上海众森计算机科技有限公司

eZ Publish 4 中文技术手册

ZerusTech

- zt-tr-ez-4-technical-manual -

提交至：ZerusTech
所有者： 上海众森计算机科技有限公司
作者： 李楠
编辑者： 李楠
版本: 1.1
日期: 2014-03-19

免责条款

本文件所有权属于上海众森计算机科技有限公司。未经上海众森计算机科技有限公司允许，任何个人或组织不得将其复制，保存，转发或用于任何商业用途。本文件内容如有任何改动，将不做特别通知。

本文件只用于信息沟通目的，不能作为上海众森计算机科技有限公司对于商品，服务的质量承诺书。上海众森计算机科技有限公司对本文件的一切内容拥有最终解释权。

注册商标

ZERUSTECH

文件修改履历

版本	修改日	修改人	注释
1.0	2009/03/15	李楠	• 初稿
1.1	2010/11/11	**李楠**	• 内容结构整理 • 修改 API 手册网址 • 增加 eZ Publish 下载地址

目录

1 安装

本章解释了如何获得并安装 eZ Publish。本章也描述了如何升级或删除 eZ Publish。如果您不需要自己安装 eZ Publish，您可以向 eZ System 寻求帮助。您也可以从 eZ Publish 的合作伙伴那里购买配置好的 eZ Publish 服务器托管环境。

可以通过三种方式安装 eZ Publish：

1. 标准安装
2. 手动安
3. 自动安装

正常安装

正常安装是最常用也是最推荐使用的方式。标准安装需要标准的系统环境，主要包括 web 服务器与数据库。您需要下载并解压 eZ Publish，并通过安装向导来安装。我们将在“正常安装”章节详细讨论这种安装。

手动安装

这种安装方式适合于有经验的用户，并不需要借助安装向导的协助。这种安装方式需要一个已经安装了 web 服务器和数据库的系统；需要下载并解压 eZ Publish。之后需要手动修改若干配置文件与数据库。我们将在“手动安装”章节详细讨论这种安装方式。

自动安装

这种安装方式（也被称为启动）适合于有经验的用户。系统管理员可以制作预配置的 eZ Publish 安装来减少对安装向导的依赖，从而节省系统安装的时间。它需要一个正确配置的系统，主要包括 web 服务器语数据库。需要下载并解压 eZ Publish。与标准安装不同，安装过程中需要用户输入的变量可以在配置文件中预配置。我们将在“自动安装”章节详细讨论这种安装方式。

1.1 正常安装

正常安装是最常用也是最推荐使用的方式。标准安装需要标准的系统环境，主要包括 web 服务器与数据库。我们将在下一章里详细讨论这种安装的系统需求。典型的正常安装包括以下几个步骤：

- 配置／创建一个数据库
- 下载 eZ Publish 安装包
- 解压 eZ Publish 安装包
- 通过安装向导安装

安装向导结束后，eZ Publish 即可使用。

您可以参阅“Linux/UNIX 安装”与“Windows 安装”章节（取决于您的目标操作系统）了解更多安装步骤信息。

1.1.1 系统需求

eZ Publish 需要五个软件系统：

1. web 服务器
2. 服务器端 PHP 脚本引擎
3. eZ Components 库
4. 数据库
5. 图像处理系统（可选）

前四个系统在安装 eZ Publish 之前就必须已经存在。图像处理系统是可选的并且只有在您需要 eZ Publish 动态处理图像时才需要。web 服务器与 PHP 引擎必须运行在同一台服务器。数据库可以运行于不同的服务器。目前，可以使用以下软件解决方案：

web 服务器

目前，只支持 Apache 服务器。在 Linux/UNIX 系统，推荐使用最新的 2.x 版本，但是 Apache 必须运行于"prefock"模式，而不是"threaded"模式。因为某些 PHP 库或扩展不是线程安全的。

在 windows 平台，建议使用最新的 1.3 版本。（Apache2.x 在 Windows 平台只支持"threaded"模式）

Apache 是一款自由，开源的软件，它是最广泛使用的 web 服务器。您可以从 http://www.apache.org 下载。

服务器端的 PHP 脚本引擎

eZ Publish 主要使用 PHP 脚本语言，PHP（hypertext preprocessor）服务器端引擎是必须的。确保已安装 PHP5.1.6 以上版本。强烈建议使用最新的 5.x 版本，因为 eZ Publish 在更新的 PHP 版本上运行得更快，而且某些扩展（如：eZ Flow）需要 PHP5.2 以上的版本。确保您的 PHP 版本符合所有软件模块的需求。

PHP 是自由软件。可以从 http://www.php.net 下载。

下表列出了需要编译的 PHP 模块。

模块名称	简介
MySQLi（建议使用）或 MySQL	需要，如果您使用 MySQL 数据库
PostgreSQL	需要，如果您使用 PostgreSQL 数据库
Zlib	需要
DOM	需要
Session	需要
PCRE	需要
GD2	需要，如果 ImageMagick 没有安装
CLI	建议
CURL	建议
mbstring	建议
Exif	建议

Zlib

确保 PHP 支持 zlib，否则安装向导无法解压下载的软件包。

DOM

大多数情况下，PHP 支持 DOM，因为 PHP 内核已经包含了对 DOM 的支持。但是，某些 Linux 发行版本的 PHP 没有预编译对 DOM 的支持，您需要安装一个称为"php-xml"的 RPM 包。这个 RPM 包会安装一个共享模块来支持 DOM。

PHP CLI

强烈建议安装 PHP CLI，否则您将无法使用某些系统特性如：通知，延迟的检索索引，升级脚本，协作系统（内容审批），从命令行清除缓存，等等。

CURL

建议支持 CURL，否则您将无法使用某些系统特性如：通过代理服务器连接，eZSoap 的 SSL 支持。

PHP 内存限制

eZ Publish 的安装向导至少需要 64MB 的内存。如果您使用的是 PHP5.2.0 或更早的版本，您需要在"php.ini"中增加"memory_limit"的值。（修改之后，您需要重启 Apache 服务器）

然而，强烈建议您保留 64MB 或更高的内存限制，因为 eZ Publish 在运信某些任务（如：重建索引，执行升级脚本等）时会消耗很多内存。另外，多语言的站点也需要至少 64MB 内存。

如果您使用的是 PHP5.2.1 或更新的版本，您不需要修改"memory_limit"（系统默认值为 128MB）。

PHP 时区

您需要在"php.ini"中设置"date.timezone"。如果不指定时区，在用 PHP5 运行 eZ Publish 时，您很有可能收到“依赖系统时区并不安全”之类的错误。PHP 时区配置示例：

```
date.timezone = Asia/Shanghai
```

参考 http://www.php.net/timezones 了解所有支持的时区代码。修改后，需要重启 Apache 服务器。

eZ Components 库

eZ Publish 是一个面向对象的应用程序。每一个类的定义保存在一个独立的 PHP 源文件中。eZ Publish 4 用__autoload()来装载所需要的类。eZ Publish 安装之后，所有 eZ Publish 内核类文件的路径都被包含在"autoload/ezp_kernel.php"中。除此以外，"autoload/ezp_extension.php"将用来保留扩展中类定义文件的路径。这些路径很可能需要被修改（例如：如果您安装了一个新的扩展或通过后台“设置－扩展”来配置现有的扩展）。您需要安装 eZ Components 2007.1.1 或更高版本来支持对 autoload path 的更新。特别需要注意的是，您至少需要安装 File 和 Base 组件("ezcBase"与"ezcFile")，否则 eZ Publish 无法更新 autoload path。

eZ Components 是一套基于 PHP 的企业级通用组件库，它可以单独或与其他库共同用于 PHP 应用程序的

开发。可以从 http://ezcomponents.org/download 下载。将来，eZ Components 将与 eZ Publish 绑定。参阅 http://ezcomponents.org/docs/install 了解如何安装 eZ Components。

重要通知

从 2008.1 版本开始，eZ Components 库需要 PHP5.2.1 或更高的版本。

数据库

eZ Publish 用数据库保存多种数据结构与数据。这意味着您的数据库必须一直可用。eZ Publish 默认支持以下数据库：

- MySQL4.1 以上版本，5.x（推荐）
- PostgreSQL7.3 以上版本

如果数据库与安装向导运行于同一台服务器，安装向导会自动检测您的数据库类型。eZ Publish 4 需要使用 UTF-8 数据库。

注意 eZ Publish 4 不支持基于 PostgreSQL 的集群。集群代码针对使用 InnoDB 存储引擎的 MySQL 数据库性能做过专门的优化。

如果您不需要在集群环境运行 eZ Publish，虽然 InnoDB 不是必须的，但是我们仍强烈推荐您使用。InnoDB 支持事务，因此可以在 MySQL 数据库中使用事务安全型表（eZ Publish 默认支持数据库事务。这一特性能保证数据库的完整性不会被错误或异常中断的处理破坏）。如果您不确定您的数据库是否支持 InnoDB，请联系您的系统管理员。

如果您希望使用 PostgreSQL，确保"pgcrypto"模块已经安装。在 Linux/UNIX 平台中，你可能需要安装一个单独的软件包－"postgresql-contrib"（参考 http://www.postgresql.org/docs/8.3/static/contrib.html 了解更多细节）。"pgcrypto"模块提供了用于 PostgreSQL 的加密函数，包括"digest"，后者是 eZ Publish 需要的。在为 eZ Publish 配置 PostgreSQL 数据库的时候，您需要在数据库中注册这些函数。参考“Linux/UNIX 安装”和“Windows 安装”（取决于目标系统）中的“配置数据库”一章。

Oracle

eZ Publish Oracle® 扩展 1.8 版本（http://ez.no/doc/extensions/database）将在 eZ Publish 4.0.1 以上版本中支持 Oracle 数据库。注意：之前的版本不能用于 eZ Publish 4。

图像处理系统（可选）

为了缩放，变换或修改图片，eZ Publish 需要调用图像处理系统。您可以选择使用以下系统中的一种（均为自由软件）：

- GD2 (由 PHP 提供)
- ImageMagick (http://www.imagemagick.org)

ImageMagick 比 GD 支持更多的图片格式并且通常比 GD 的表现更好（更好的缩放效果，等。）。安装向导会自动检测预安装的图像处理系统。

如何安装与配置所需要的软件系统（综上所述）已超出本文范围。请参阅相应软件的网站了解细节。

1.1.2 Linux/UNIX 安装

您的系统必须符合正常安装的需求。先阅读正常安装章节。确保您的系统运行于 Linux，并且已经安装了 Apache，PHP，MySQL 或 PostgreSQL。如前所述，数据库可以与 WEB 服务器运行于不同的服务器上。Linux/UNIX 安装包括以下几个步骤：

1. 配置数据库（MySQL 或 PostgreSQL）
2. 下载 eZ Publish
3. 解压 eZ Publish
4. 启动安装向导

配置数据库

运行安装向导之前，必须创建一个数据库。参考以下内容创建 MySQL 或 PostgreSQL 数据库。

MySQL

1. 以 root（或其他具有 CREATE，CREATE USER 与 GRANT OPTION 权限的用户）身份登录

```
$ mysql --host=<mysql_host> --port=<port> -u <mysql_user> -p<mysql_password>
```

注意：

如果 MySQL 安装在同一台服务器，"—host"参数可以省略。如果"--port"参数省略，MySQL 默认端口为 3306。

MySQL 客户端会显示"mysql>"提示符。

2. 创建一个数据库

```
mysql> CREATE DATABASE <database> CHARACTER SET utf8;
```

3. 设置权限

```
mysql> GRANT ALL ON <database>.* TO <user>@<ezp_host> IDENTIFIED BY '<password>';
```

注意：

如果用户帐号不存在，新用户会被创建。

<mysql_host>	MySQL 主机名或 IP 地址
<port>	MySQL 端口。默认端口 3306
<mysql_user>	MySQL 用户名（如果没有其他用户，使用 root）。
<mysql_password>	<mysql_user>的密码。
<database>	数据库名称。

<user>	使用<database>的用户。
<ezp_host>	运行 eZ Publish 的主机明（如果运行于本机，则使用 localhost）。
<password>	<user>的密码。

PostgreSQL

1. 以 postgres（或其他具有创建角色与数据库的用户）用户登录系统：

```
$ psql -h <psql_host> -p <port> -U <psql_user> -W
```

注意：

如果 PostgreSQL 安装在同一台服务器，"-h"参数可以省略。如果"-p"参数省略，默认的 PostgreSQL 端口为 5432。

PostgreSQL 客户端会要求您输入<psql_user>的密码。如果密码正确，客户端会显示"<psql_user>=#"提示符。

2. 创建数据库

```
postgres=# CREATE DATABASE <database> ENCODING='utf8';
```

3. 创建用户

```
postgres=# CREATE USER <user> PASSWORD '<password>';
```

4. 设置权限

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE <database> TO <user>;
```

5. 导入"pgcrypto"模块

```
postgres=# \c <database>
<database>=# \i '<path_to_pgcrypto>'
```

<psql_host>	PostgreSQL 主机名。
<port>	PostgreSQL 端口。
<psql_user>	PostgreSQL 用户名（默认用户"postgresql"）。
<database>	数据库名，例如："my_new_database"。
<user>	<database>的用户明。
<password>	<user>的密码。
<path_to_pgcrypto>	"pgcrypto.sql"的位置，例如"/usr/share/pgsql/contrib/pgcrypto.sql"。

下载 eZ Publish

从 http://ez.no/download/ez_publish 下载最新的稳定版本。

解压 eZ Publish

用您习惯的工具解压缩下载的 eZ Publish 至 WEB 服务器目录（可以通过浏览器访问的目录）。以下的命

令行可用 tar 解压 tar.gz 文件，假定系统已安装了"tar"和"gzip"命令。

```
$ tar zxvf ezpublish-<version_number>-gpl.tar.gz -C <web_served_directory>
```

<version_number>	eZ Publish 版本好。
<web_served_directory>	WEB 服务器目录的全路径。这可以是 WEB 服务器的"document root"，或某个个人目录（通常为"public_html"或"www"，并位于用户的主目录）。

解压工具会把 eZ Publish 解压缩至一个子目录"ezpublish-<version_number>"。可以任意重命名这个目录，如"my_site"。

启动安装向导

以上步骤完成之后即可以在浏览器中启动安装向导。首次访问 eZ Publish 解压缩目录内的 index.php 时安装向导会自动启动。假设我们使用"www.example.com"这个域名，并且我们将 eZ Publish 解压缩后的目录重命名为"my_site"。

Document root 示例

如果 eZ Publish 被解压缩至 document root 下的子目录"my_site"，可以通过访问 http://www.example.com/my_site/index.php 启动安装向导。

Home directory 示例

如果 eZ Publish 被解压缩至某个用户（如"peter"）主目录下的某个 WEB 服务器目录（通常为"public_html"，"www"，"http"，"html"，或"web"）。可以通过访问 http://www.example.com/~peter/my_site/index.php 启动安装向导。

参阅"安装向导"章节了解更多安装向导的内容。

1.1.3 Windows 安装

您的系统必须符合正常安装的需求。先阅读正常安装章节。确保您的系统运行于 Windows 上，并且已经安装了 Apache，PHP，MySQL 或 PostgreSQL。如前所述，数据库可以与 WEB 服务器运行于不同的服务器上。Windows 安装包括以下几个步骤：

1. 配置数据库（MySQL 或 PostgreSQL）
2. 下载 eZ Publish
3. 解压 eZ Publish
4. 启动安装向导

配置数据库

运行安装向导之前，必须创建一个数据库。参考以下内容创建 MySQL 或 PostgreSQL 数据库。

MySQL

1. 以 root（或其他具有 CREATE，CREATE USER 与 GRANT OPTION 权限的用户）身份登录

```
$ mysql --host=<mysql_host> --port=<port> -u <mysql_user> -p<mysql_password>
```

注意：

如果 MySQL 安装在同一台服务器，"—host"参数可以省略。如果"--port"参数省略，MySQL 默认端口为 3306。

MySQL 客户端会显示"mysql>"提示符。

2. 创建一个数据库

```
mysql> CREATE DATABASE <database> CHARACTER SET utf8;
```

3. 设置权限

```
mysql> GRANT ALL ON <database>.* TO <user>@<ezp_host> IDENTIFIED BY '<password>';
```

注意：

如果用户帐号不存在，新用户会被创建。

<mysql_host>	MySQL 主机名或 IP 地址
<port>	MySQL 端口。默认端口 3306
<mysql_user>	MySQL 用户名（如果没有其他用户，使用 root）。
<mysql_password>	<mysql_user>的密码。
<database>	数据库名称。
<user>	使用<database>的用户。
<ezp_host>	运行 eZ Publish 的主机明（如果运行于本机，则使用 localhost）。
<password>	<user>的密码。

PostgreSQL

1. 以 postgres（或其他具有创建角色与数据库的用户）用户登录系统：

```
$ psql -h <psql_host> -p <port> -U <psql_user> -W
```

注意：

如果 PostgreSQL 安装在同一台服务器，"-h"参数可以省略。如果"-p"参数省略，默认的 PostgreSQL 端口为 5432。

PostgreSQL 客户端会要求您输入<psql_user>的密码。如果密码正确，客户端会显示"<psql_user>=#"提示符。

2. 创建数据库

```
postgres=# CREATE DATABASE <database> ENCODING='utf8';
```

3. 创建用户

```
postgres=# CREATE USER <user> PASSWORD '<password>';
```

4. 设置权限

```
postgres=# GRANT ALL PRIVILEGES ON DATABASE <database> TO <user>;
```

5. 导入"pgcrypto"模块

```
postgres=# \c <database>
```

```
<database>=# \i '<path_to_pgcrypto>'
```

<psql_host>	PostgreSQL 主机名。
<port>	PostgreSQL 端口。
<psql_user>	PostgreSQL 用户名（默认用户"postgresql"）。
<database>	数据库名，例如："my_new_database"。
<user>	<database>的用户明。
<password>	<user>的密码。
<path_to_pgcrypto>	"pgcrypto.sql"的位置，例如"C:\\Program Files\\PostgreSQL\\8.2\\share\\contrib\\pgcrypto.sql"。

下载 eZ Publish

从 http://ez.no/download/ez_publish 下载最新的稳定版本。

解压 eZ Publish

用您习惯的工具解压缩下载的 eZ Publish 至 WEB 服务器目录（可以通过浏览器访问的目录）。

解压工具会把 eZ Publish 解压缩至一个子目录"ezpublish-<version_number>"。可以任意重命名这个目录，如"my_site"。

启动安装向导

以上步骤完成之后即可以在浏览器中启动安装向导。首次访问 eZ Publish 解压缩目录内的 index.php 时安装向导会自动启动。假设我们使用"www.example.com"这个域名，并且我们将 eZ Publish 解压缩后的目录重命名为"my_site"。

Document root 示例

如果 eZ Publish 被解压缩至 document root 下的子目录"my_site"，可以通过访问 http://www.example.com/my_site/index.php 启动安装向导。

参阅"安装向导"章节了解更多安装向导的内容。

1.2 手动安装

这种安装方法只适用于那些完全清楚自己在做什么的高级用户，其他用户都应该使用“正常安装”。手动安装需要一个已经安装了 WEB 服务器，数据库等必须软件的系统。需要下载并解压 eZ Publish。与安装向导不同，所有的配置均通过系统的命令行完成。以下章节（取决于您的系统）会引导您完成所有步骤。

1.2.1 系统需求

与正常安装的系统需求相同。参阅正常安装的“系统需求”一章。

1.2.2 Linux/UNIX 安装

确保系统满足手动安装的系统需求。如果不确定，参阅前一章节。确保您的系统运行于 Linux/UNIX，并且已经安装 Apache，PHP，MySQL 或 PostgreSQL。如前所述，数据库可与 WEB 服务器运行于不同服务器。手动安装需要以下几个步骤：

- 配置数据库（MySQL 或 PostgreSQL）
- 下载 eZ Publish
- 解缩 eZ Publish
- 手动配置 eZ Publish

手动安装与正常安装的不同只在于最后一步。手动安装通过手动配置若干文件来配置 eZ Publish，而不启动安装向导。前三个步骤与正常安装“Linux/UNIX 安装”所述内容一致。最后一个步骤在“手动配置 eZ Publish ”一章中说明。

1.2.3 Windows 安装

确保系统满足手动安装的系统需求。如果不确定，参阅前一章节。确保您的系统运行于 Windows，并且已经安装 Apache，PHP，MySQL 或 PostgreSQL。如前所述，数据库可与 WEB 服务器运行于不同服务器。手动安装需要以下几个步骤：

- 配置数据库（MySQL 或 PostgreSQL）
- 下载 eZ Publish
- 解缩 eZ Publish
- 手动配置 eZ Publish

手动安装与正常安装的不同只在于最后一步。手动安装通过手动配置若干文件来配置 eZ Publish，而不启动安装向导。前三个步骤与正常安装“Windows 安装”所述内容一致。最后一个步骤在“手动配置”一章中说明。

1.2.4 手动配置

本章描述了如何手动配置 eZ Publish。请牢记手动安装只适合于那些完全了解自己在做什么的高级用户。以下步骤适用于 Linux/UNIX 与 Windows。

初始化数据库

可以通过导入两个重要的 SQL 脚本（"kernel_schema"和"cleandata"）来初始化 eZ Publish 数据库。（注意：运行这两个脚本之前，您需要创建一个空数据库）。第一个脚本创建必要的数据库结构，第二个脚本导入预定义的数据。使用哪一个"kernel_schema"脚本取决于使用的数据库。"cleandata"脚本适用于 MySQL 与 PostgreSQL 数据库。

MySQL

用以下命令行执行 MySQL 的"kernel_schema"脚本：

```
$ mysql -u USERNAME -pPASSWORD DATABASE < PATH/kernel/sql/mysql/kernel_schema.sql
```

这个脚本将使用 InnoDB 存储引擎创建数据表。确保数据库的默认存储引擎也被设置为 InnoDB，否则将来的系统升级将会造成混合的表类型。参阅 MySQL 文档了解如何设置默认的存储引擎。

使用以下脚本运行通用的"cleandata"脚本：

```
$ mysql -u USERNAME -pPASSWORD DATABASE < PATH/kernel/sql/common/cleandata.sql
```

USERNAME	MySQL 用户名。
PASSWORD	MySQL 密码。
DATABASE	数据库名。
PATH	eZ Publish 的安装的绝对路径。例：/opt/ezp

文件权限

Windows 用户可以跳过这一步。如果 eZ Publish 安装于 Linux/UNIX 系统，某些文件的权限需要重新设置。可以使用 eZ Publish 自带的脚本来设置文件权限。这个脚本必须被执行，否则 eZ Publish 无法正常运行。这个脚本需要在 eZ Publish 的安装目录中运行。

```
$ cd /opt/ezp
$ bin/modfix.sh
```

把"/opt/ezp"替换为您 eZ Publish 安装目录的绝对路径。

"modifx"脚本递归地修改 eZ Publish 安装目录中以下目录的权限：

- var/*
- settings/*
- design/*

如果您知道 WEB 服务器的用户与用户组，建议使用以下的权限：

```
# chown -R user.usergroup var/
# chmod -R 770 var/
```

"user.usergroup"必须被替换为 WEB 服务器的用户与用户组。

配置 eZ Publish

eZ Publish 目录中的"settings/override/site.ini.append.php"必须被修改，否则 eZ Publish 不能正常运行。这个文件是 site.ini 文件的一个全局的重设文件。这个文件中有很多选项需要配置（数据库，邮件发送系统，var 目录等等）。以下是一个通用的 site.ini.append.php 示例：

```
<?php /* #?ini charset="iso-8859-1"?
[DatabaseSettings]
DatabaseImplementation=ezmysql
Server=localhost
User=root
Password=Database=my_database
[FileSettings]
```

```
VarDir=var/example
[Session]
SessionNameHandler=custom
[SiteSettings]
DefaultAccess=example
SiteList[]
SiteList[]=example
[SiteAccessSettings]
CheckValidity=false
AvailableSiteAccessList[]
AvailableSiteAccessList[]=example
AvailableSiteAccessList[]=example_admin
RelatedSiteAccessList[]
RelatedSiteAccessList[]=example
RelatedSiteAccessList[]=example_admin
MatchOrder=host;uri
# Host matching
HostMatchMapItems[]=www.example.com;example
HostMatchMapItems[]=admin.example.com;example_admin
[InformationCollectionSettings]
EmailReceiver=webmaster@example.com
[MailSettings]
Transport=sendmail
AdminEmail=webmaster@example.com
EmailSender=test@example.com
[RegionalSettings]
Locale=eng-GBContentObject
Locale=eng-GBText
Translation=disabled
*/ ?>
```

在上例中，"[SiteAccessSettings]"章节中的"AvailableSiteAccessList[]"数组定义了两个可用的站点入口："example" 和"example_admin"。"CheckValidity"应设置为 false，否则访问这个站点时安装向导会被触发。

除此以外，还需创建两个站点入口的配置文件。一个公共站点入口（"example"），一个管理站点入口（"examle_admin"）。在 eZ Publish 根目录下创建以下站点入口目录：

- settings/siteaccess/example

- settings/siteaccess/example_admin

两个站点入口都需要一个独立的"site.ini.append.php"文件。

公共站点入口

```
<?php /* #?ini charset="iso-8859-1"?
[SiteSettings]
SiteName=Example
SiteURL=www.example.com
LoginPage=embedded
[SiteAccessSettings]
RequireUserLogin=false
ShowHiddenNodes=false
[DesignSettings]
SiteDesign=example
[ContentSettings]
ViewCaching=disabled
[TemplateSettings]
TemplateCache=disabled
TemplateCompile=disabled
#ShowXHTMLCode=enabled
#Debug=enabled
[DebugSettings]
DebugOutput=enabled
Debug=inline
#DebugRedirection=enabled
[RegionalSettings]
SiteLanguageList[]
SiteLanguageList[]=eng-GB
ShowUntranslatedObjects=disabled
*/ ?>
```

管理站点入口

```
<?php /* #?ini charset="iso-8859-1"?
[SiteSettings]
SiteName=Example
SiteURL=admin.example.com
```

```
LoginPage=custom

[SiteAccessSettings]

RequireUserLogin=true

ShowHiddenNodes=true

[DesignSettings]

SiteDesign=admin

[ContentSettings]
CachedViewPreferences[full]=admin_navigation_content=0;admin_navigation_details=0;admin_navi
gation_languages=0;admin_navigation_locations=
0;admin_navigation_relations=0;admin_navigation_roles=0;admin_navigation_policies=0;admin_na
vigation_content=0;admin_navigation_translatio
ns=0;admin_children_viewmode=list;admin_list_limit=1;admin_edit_show_locations=0;admin_url_l
ist_limit=10;admin_url_view_limit=10;admin_sec
tion_list_limit=1;admin_orderlist_sortfield=user_name;admin_orderlist_sortorder=desc;admin_s
earch_stats_limit=1;admin_treemenu=1;admin_boo kmarkmenu=1;admin_left_menu_width=13

[DebugSettings]

DebugOutput=disabled

Debug=inline

[RegionalSettings]

SiteLanguageList[]

SiteLanguageList[]=eng-GB

ShowUntranslatedObjects=enabled

*/ ?>
```

注意：

在"settings/override/site.ini.append.php"中定义的数据库配置，邮件配置，区域配置与其它配置适用于所有的站点入口。在本例中，"Database=my_database"在这个文件中的"[DatabaseSettings]“定义，因此"example"与"example_admin"均会使用这个数据库。参阅“基本概念”中的“站点管理”与“配置”两个章节了解更多内容。

Unicode 支持

不需要重设"i18n.ini"文件，因为 eZ Publish 4 默认启用对 unicode 的支持。

语言

可用语言与优先级可以通过修改每个站点入口中的"site.ini.append.php"中的"[RegionalSettings]"章节中的"SiteLanguageList"配置来定义。如果不指定，系统会使用旧的"ContentObjectLocale“配置，因此只有默认语言会被显示。参阅“配置站点语言”章节了解更多内容。

"cleandata.sql"脚本只创建一种英式英文（eng-GB）一种语言。 其它语言需要通过管理界面的“设置 – 语言”来添加(在本例中：http://admin.example.com)。

动态树状菜单

如果您的站点有很多节点，强烈建议您为管理站点入口启用"Dynamic"选项。这会使管理界面左侧的树状菜单速度更快并减少对网络带宽的消耗。

系统管理员的登录名与密码

"cleandata.sql"会创建以下的管理员登录名与密码。

登录名： admin

密码：publish

强烈建议您立即修改管理员的密码。

注意：

如果您需要另外的管理员帐号，您可以创建新的管理员帐号；用新帐号登录管理界面；删除旧管理员帐号。

1.3 自动安装

自动安装（也被称为“kickstart”）适合于有经验的用户。它是“正常安装”的自动版本并且被设计帮助系统管理员快速完成预定义的 eZ Publish 安装。这种方法只需要与安装向导进行很少的交互，因此适合用于快速批量地安装 eZ Publish。这种方法的系统需求与“正常安装”相同。典型的自动安装包含以下几个步骤：

- 配置/创建数据库
- 下载 eZ Publish
- 解压 eZ Publish
- 配置"kickstart.ini"文件
- 启动安装向导

安装向导完成之后，eZ Publish 即可使用。

1.3.1 系统需求

自动安装的系统需求与正常安装相同。参阅正常安装的“系统需求”章节了解更多内容。

确保系统至少已安装了 WEB 服务器，PHP 引擎与数据库服务器。如果 kickstart 配置文件需要使用其它的服务器端软件，则必须安装。例如：如果"ImageMagick"被配置为主要的图像处理系统，则需要安装"ImageMaick"。

下一章将解释如何配置 eZ Publish 来自动完成安装。

1.3.2 自动安装

系统必须满足自动安装的系统需求。如果您不确定，请参阅前一章节。本章将会引导您完成以下步骤

- 配置数据库（MySQL 或 PostgreSQL）
- 下载 eZ Publish

- 解压 eZ Publish
- 配置 kickstart 系统
- 启动安装向导开始安装

取决于目标系统，请参阅"Linux/UNIX 安装"或"Windows 安装"了解前三步（配置数据库，下载，解压）的详细内容。 剩余步骤如下：

配置 kickstart 系统

自动安装的行为由"kickstart.ini"文件控制。这个文件可对每个安装步骤指定参数。例如：通过指定数据库链接参数，安装向导中对应页面的表单内容可以被预填写。也可以控制安装向导跳过某些步骤。

初始化

复制"kickstart.ini-dist"文件（在 eZ Publish 安装的根目录中）到"kickstart.ini"（在 eZ Publish 安装根目录中）。以下示例演示如何在 Linux/UNIX 中复制：

1. 进入 eZ Publish 安装根目录

```
$ cd /path/to/ezpublish/
```

2. 复制文件

```
$ cp kickstart.ini-dist kickstart.ini
```

安全问题

安装过程中，WEB 服务器必须对"kickstart.ini"文件有读权限。如果这个文件包含登录名，密码等数据，这可能成为一个安全问题。为了防止类似问题，建议使用以下解决方案：

- 安装结束后删除这个文件
- 使用 rewrite rules 保证外界无法访问这个文件

配置块

"kickstart.ini"文件中没个安装步骤有一个对应的配置块。配置块名称由方括号[]环绕。可用的配置块如下。

```
[email_settings]
[database_choice]
[database_init]
[language_options]
[site_types]
[site_access]
[site_details]
[site_admin]
[security]
```

```
[registration]
```

默认的 kickstart 文件中，所有配置均被注释。要启用某个配置，必须将其反注释。您可以删除配置开头的(“#”)符号来反注释它。确保配置行没有前导空格。

配置参数

每个参数接受字符串参数值。某些参数接受字符串数组。以下示例演示了两种参数类型。

- 单个参数：

```
Server=www.example.com
```

- 数组参数 Array parameter:

```
Title[]
Title[news]=The news site
Title[forums]=The forum site
```

文档与示例

"kickstart.ini"文件自身包含文档与示例。请参阅文件中的文档与示例了解详细内容。下表演示了示例与文档如何描述必须与可选参数。

语法	描述
<value>	尖括号表示这个参数是必须参数，例如： `#Server=<hostname>`
[value]	方括号表示这个参数是可选参数，例如： `#FirstName=[string]`

只有反注释某个参数才能启用它。删除开头的("#")字符并确保这一行中没有前导空格。

跳过步骤

可以反注释并设置"Continue"为"true"来跳过某个安装步骤。这个参数可应用于每个步骤/块。下表解释了不同"Continue"参数值的意义。

配置	结果
Continue=false	这个步骤会显示并且表单的字段会根据"kickstart.ini"的配置（如果有）预填入。如果 Continue 参数缺失或被注释，也是同样效果。
Continue=true	系统会自动使用 kickstart 文件中定义的参数值，因此这个安装步骤不会显示。然而，如果有任何错误（参数缺失或错误），这个安装步骤仍会显示。

开始安装

启动安装向导开始安装过程。请参阅“正常安装”中“启动安装向导”一章。

1.4 安装向导

本章包含一个完整的安装向导文档。安装向导被设计用来简化系统的初始化。必要的安装步骤（参阅之前相关章节）完成之后，可以在浏览器中启动安装向导。初次访问"index.php"（位于 eZ Publish 安装的根目录中）时，安装向导会自行启动。

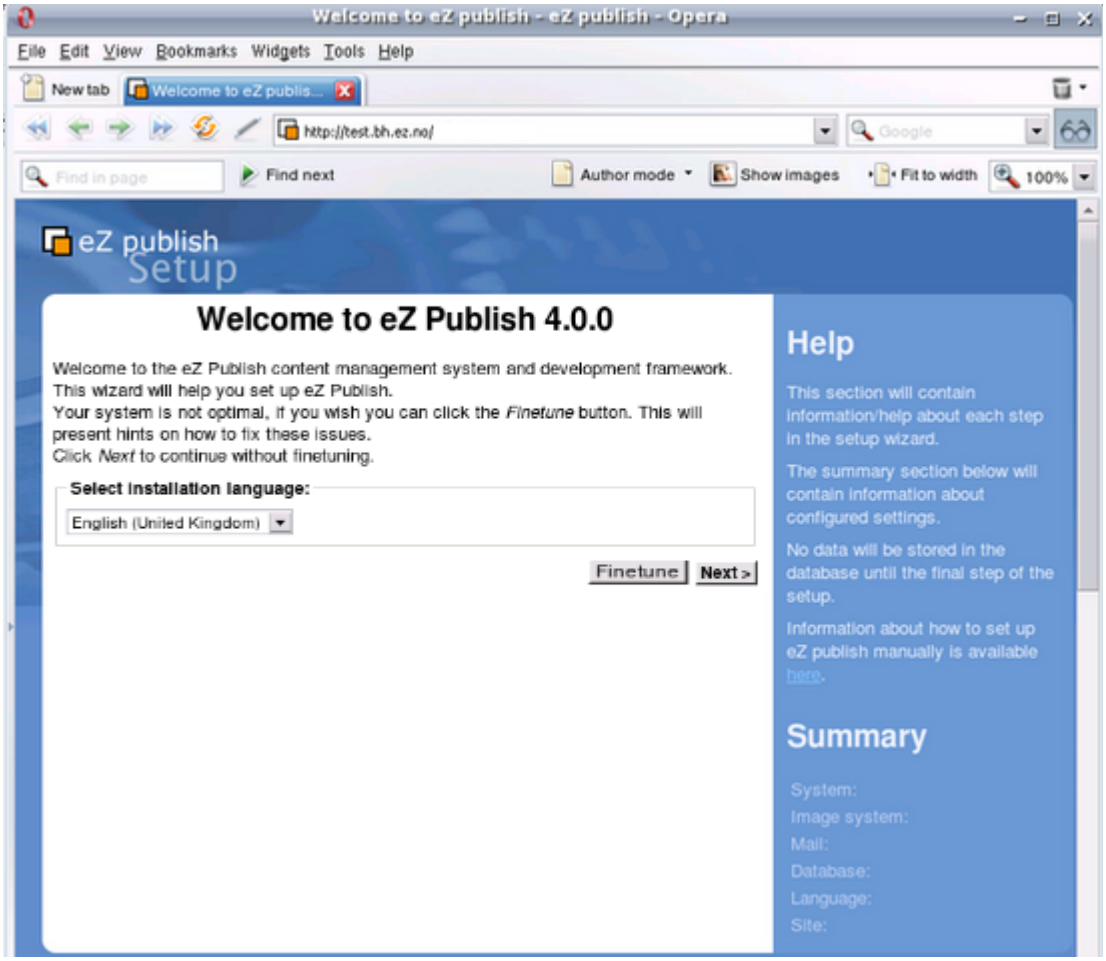
在最后一步完成之前，安装向导不保存或修改任何数据；因此，您可以安全地刷新"index.php"来重新启动安装向导。可以点击返回按钮（位于页面底端）跳回前一页修改配置。典型的安装包含 1 2 个步骤：

1. 欢迎页面
2. 系统检测
3. 邮件发送
4. 数据库选择（可选）
5. 数据库初始化
6. 语言支持
7. 站点选择
8. 访问方法
9. 站点细节
10. 站点安全
11. 站点注册
12. 完成

注意：

在安装 eZ Publish bundle 时，某些步骤可能被省略。

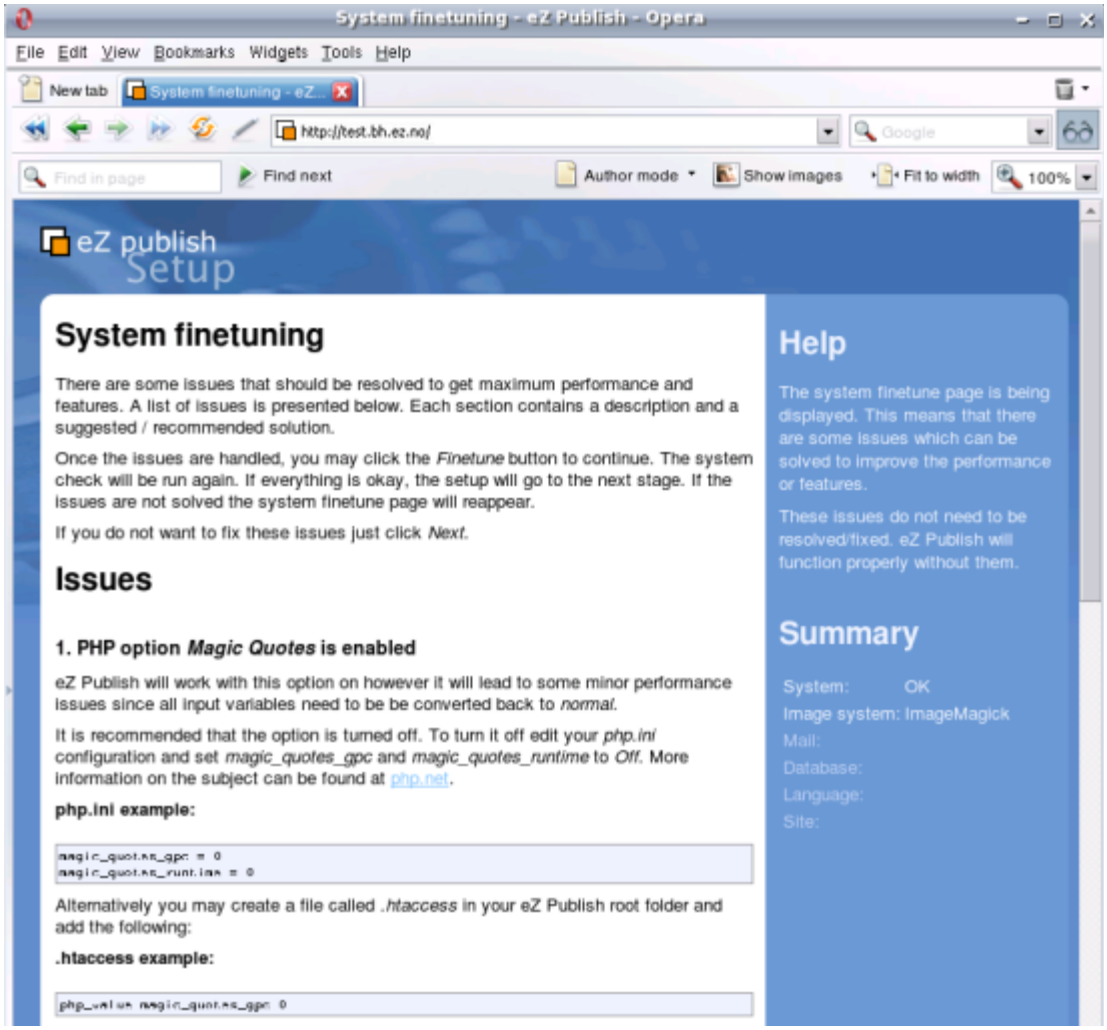
欢迎页面



这是安装向导的初始页面。这一步骤允许用户选择安装过程中使用的语言。安装向导也会检查系统配置并且，如果系统的配置有待优化，显示优化建议。（这种情况下，页面底端会显示一个“系统优化”按钮）。

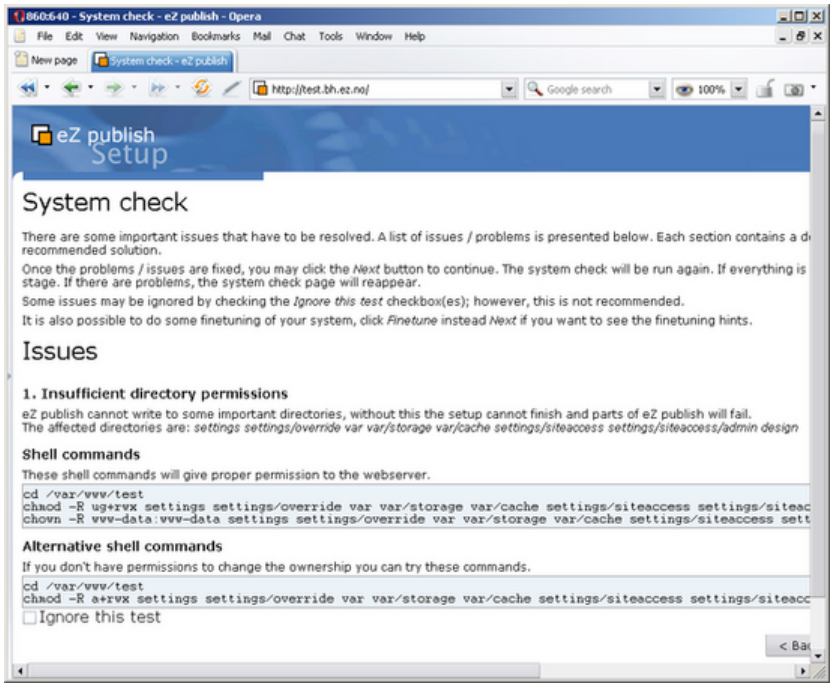
系统会根据您浏览去的设置自动选择一个默认语言。您可以从语言下拉框中选择偏好的语言。（可选语言列表来源于"share/locale"目录中的 INI 文件）。

点击“系统优化”按钮（如果存在）以后，安装向导会载入“系统优化”页面，这个页面包含系统配置问题的信息。下图为这个页面的示例。



点击“下一步”按钮之后，安装向导会载入“系统检测”页面（如果有任何关键配置需要修改）或“邮件发送”页面（如果配置没有问题）。

系统检测



这个页面通常因为安装向导检测到系统存在的关键问题才会显示。安装向导会显示与问题相关的信息并会提供推荐的解决方案。

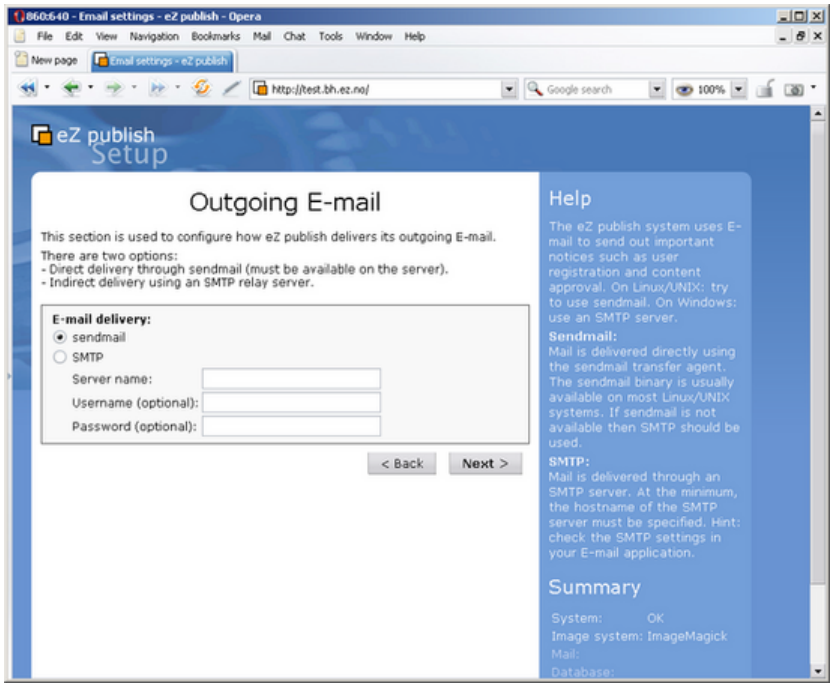
问题

系统可能存在多个问题。每个问题描述下面会显示相应的建议。安装向导可能会建议您执行若干脚本（用于修正文件属主，权限等问题）。这些脚本/命令必须通过系统命令行执行。可直接从浏览器窗口复制这些命令并粘贴到命令行窗口。点击“下一步”按钮，安装向导会再次检测。安装向导和反复检测系统配置直到所有问题被修正（或忽略，参阅下一章节）。如果没有问题，安装向导会载入下一个步骤。

忽略检测

某些问题可以通过勾选“忽略这个检测”复选框来忽略。尽管如此，我们仍然建议您修正所有问题，而不是简单地忽略它们。

邮件发送



eZ Publish 通过电子邮件发送若干通知。这一步骤用来配置 eZ Publish 如何发送邮件。有两种选择：

直接通过 sendmail 发送（服务器必须安装 sendmail）

间接通过 SMTP (Simple Mail Transfer Protocol) 服务器

在 linux/UNIX 平台：优先尝试 sendmail；如果 sendmail 不可用再使用 SMTP。在 Windows 平台：使用 SMTP。

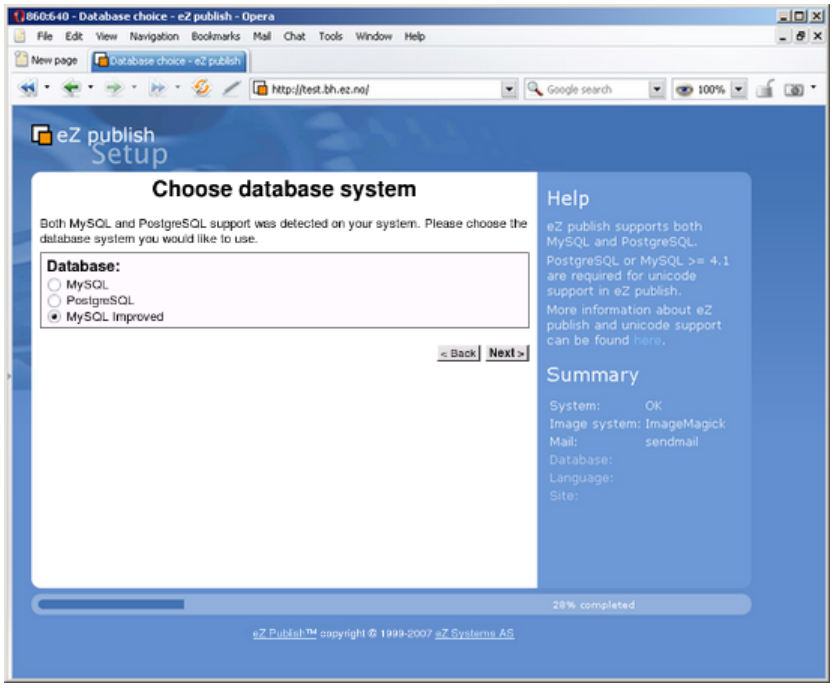
Sendmail

邮件通过 sendmail 转发代理直接发送。sendmail 必须与 WEB 服务器运行于同一台机器上。sendmail 二进制文件在大部分 Linux/UNIX 系统都存在。如果 sendmail 不可用，则使用 SMTP。

SMTP

邮件通过 SMTP 服务器转发。您至少需要提供 SMTP 服务器的主机名。

数据库类型

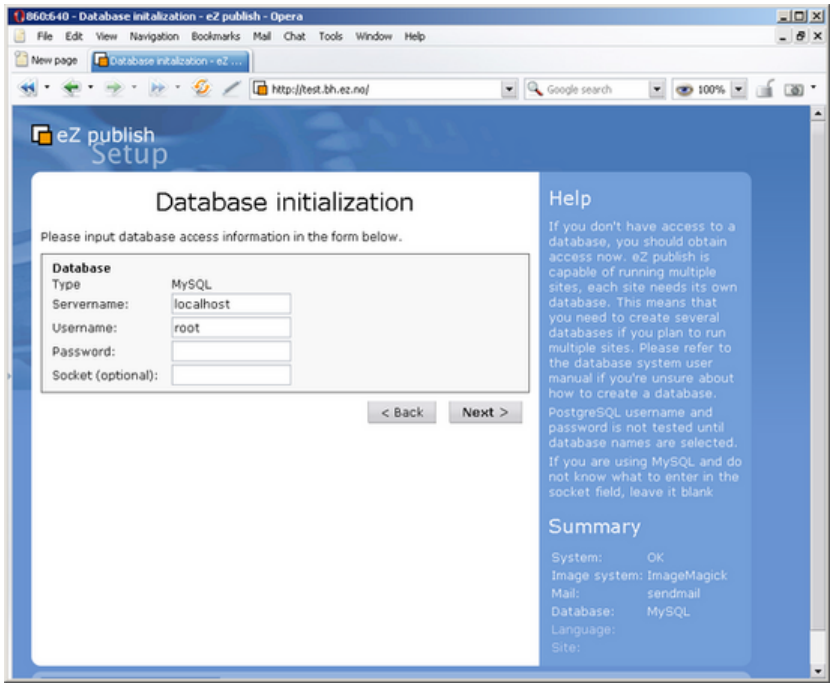


安装向导会自动检测对 PHP 脚本引擎可用的数据库。如果 MySQL 与 PostgreSQL 均可用，会显示数据库选择对话框。 如果 PHP 直支持一种数据库，安装向导会自动选择这种数据库，数据库选择对话框不会被载入。

注意：

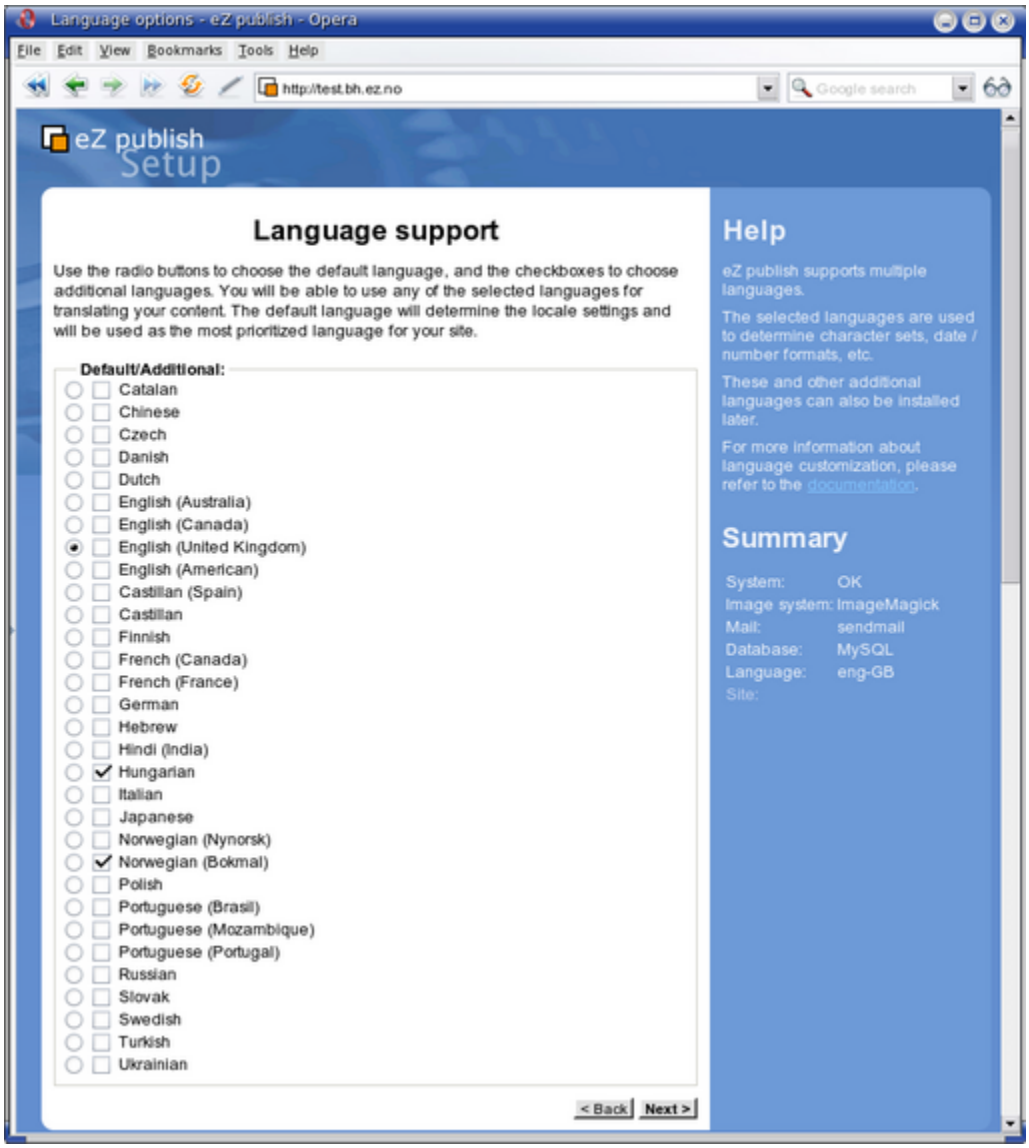
如果 PHP 启用了 MySQLi 扩展，列表中会显示"MySQL Improved"选项。如果您使用 MySQL 数据库，建议使用"MySQL Improved"，而不是"MySQL"。

数据库初始化



您需要提供数据库主机名，用户名与密码。点击“下一步”按钮，安装向导会尝试链接数据库。安装向导只有通过您提供的数据库链接信息成功连接数据库才会继续。PostgreSQL 参数会在后面的步骤测试。

语言支持



这一步允许用户选择要安装的语言。安装向导会根据您浏览器的设置预先选择一种默认语言。您可以用单选框选择一种默认语言（必选）并用复选框选择多种附加语言（可选）。所有选择的语言会被添加到系统中并在优先语言列表中显示。安装成功后，您可以使用所有语言创建与翻译内容。

注意：

选择的默认语言会决定系统默认的语言，区域以及最优先语言。如果，例如您选择了德语作为默认语言，那么系统的区域和默认语言均会被设置为"ger-DE"，您的管理界面也会被翻译为德文界面，并且这种语言也会被作为最优先的语言。语言可以通过管理界面重新设置（即使站点已在运行）。

无论您选择了何种语言，站点均会使用 UTF-8 字符集。

站点选择

这一步骤允许用户选杂某个标准安装包。这些安装包主要为演示和学习提供基本示例。当然，您也可以把这些安装包作为开发的基本框架，并对其进行扩展/定制来满足特定的需求。一个演示站点通常包含一些图片，CSS 代码，内容和模板文件。如果您希望从头开始构建一个站点，请使用 plain 类型。

安装向导自动从远端软件仓库下载安装包列表并允许用户选择其中一个安装包。默认的远端软件仓库是 http://packages.ez.no/ezpublish/4.0。

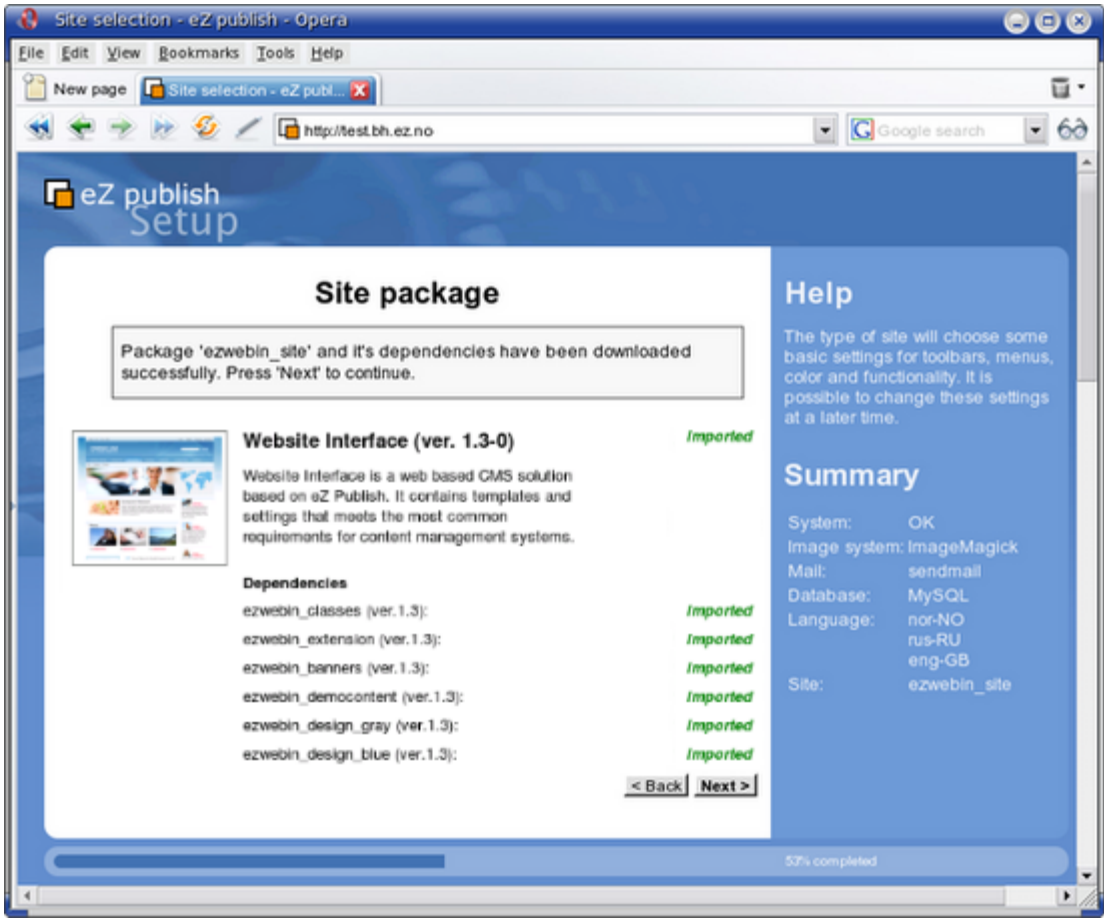
注意：

目前，默认软件仓库只包含以下三个安装包：

- Plain site
- Website Interface
- eZ Flow

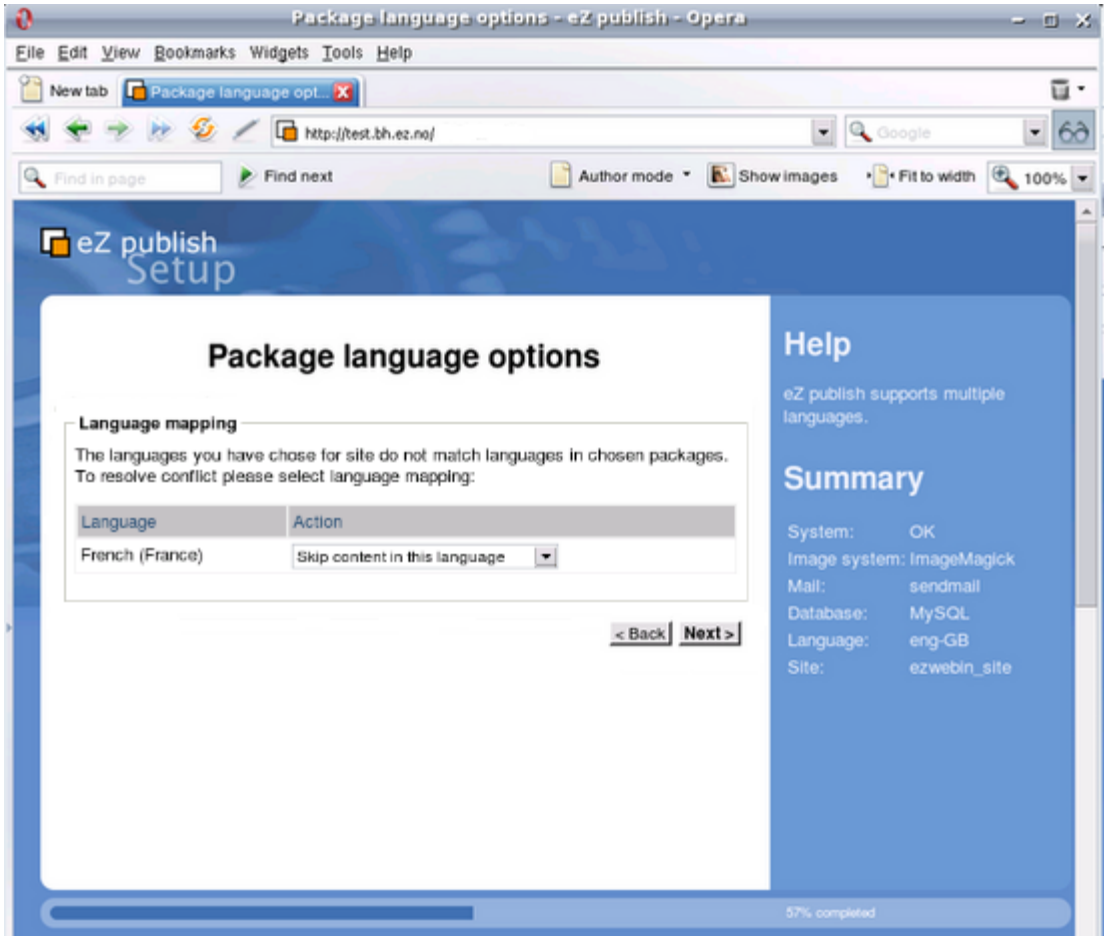
旧的站点安装包，如："新闻站点"，"商店站点"，"图片库站点"目前在 eZ Publish 4 暂不可用。

安装向导会自动下载所选择的站点安装包及其依赖包，导入安装包并在下图所示页面中显示成功导入的安装包列表（如果所有安装包已被保存在本地软件仓库，这一步会被省略）。



除站点风格包外，所有依赖包会被自动安装。

安装包语言选项



如果在“语言支持”步骤所选的语言与安装包所使用的语言不匹配，上面的“安装包语言选项”界面会显示。例如，"Website interface"站点安装包可以支持两种语言的演示内容：英文（英国）与法语。如果在“语言支持”步骤，用户选择了相同的语言，安装包会被安静地安装。否则，系统会要求用户指定如何处理未知语言。（如：存在于安装包中但不存在于当前选择的语言）。可能的动作为：

- 跳过这种语言的内容
- 创建语言（扩展站点的语言配置并创建这种语言的内容）
- 映射到其它语言（用演示内容创建其它语言的内容）

处理可能碰到的问题

如果 WEB 服务器无法与远端软件仓库通讯（例如：被火墙规则屏蔽），安装向导会在“站点选择”步骤显示错误信息。要修正这个错误，在火墙上允许对 http://packages.ez.no（80 端口）的出栈操作或手动下载安装包。

通过代理的出栈连接

如果您之允许通过代理的出栈连接，那么您需要对 eZ Publish 做如下配置：

1. 在"settings/override"目录创建"site.ini.append.php"文件并添加如下内容：

```
[ProxySettings]
ProxyServer=proxy.example.com:3128
User=myuser
Password=secret
```

2. 用真正的代理 IP 地址和端口替换"proxy.example.com:3128"。如果代理需要认证，您也需要提供用户名/密码。
3. 重启安装向导

注意：

必须启用 PHP 对 CURL 的支持，否则经过代理的出栈连接不能工作。

手动下载安装包

如果安装向导无法连接至外部软件仓库，您可以手动下载安装包及其依赖包，然后通过安装向导上传/导入至系统中。参阅以下步骤。

1. 从安装包下载页面(http://ez.no/download/ez_publish/ez_publish_4_stable_releases/4_0/packages/4_0_0)。这个页面的"Sites"章节包含可用安装包列表。每个列表包含以下信息。
   - 名称
   - 简介
   - 依赖关系（如果有）

点击名称下载安装包。（安装包被下载为".ezpkg"文件）

2. 下载所有依赖包（依赖包在"Dependencies"下列出）。您可以点击名称下载安装包。安装包被下载为".ezpkg"文件。
3. 用页面底端的导入界面上传/导入安装 包（点击“选择”按钮，选择下载的".ezpkg"文件，然后点击“上传”按钮）。导入的安装包会在列表中显示。
4. 反复使用导入界面上传/导入依赖安装包。

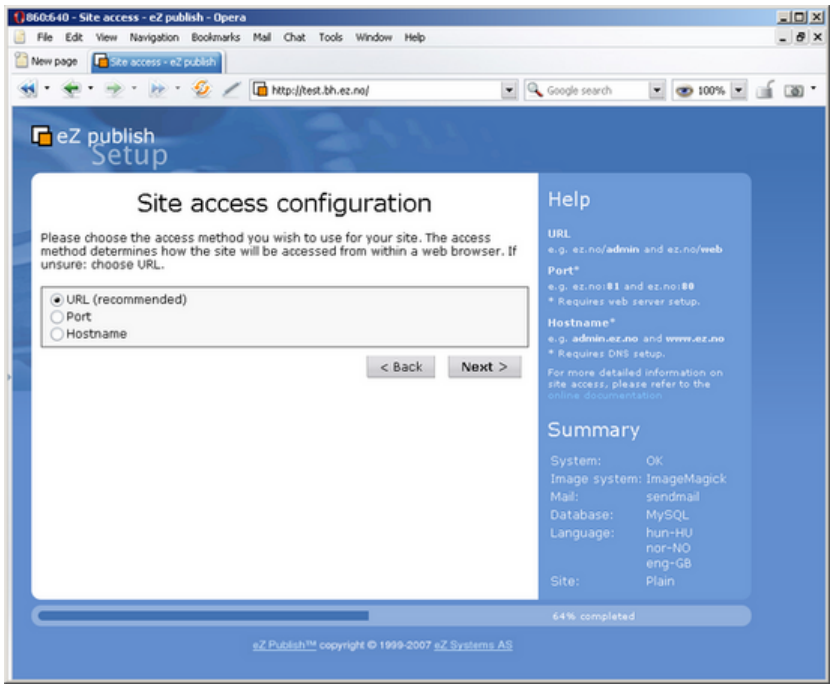
注意：手动从远端软件仓库下载安装包。参阅以下步骤。

1. 在软件仓库(http://packages.ez.no/ezpublish/4.0)找到需要的安装包并手动下载。（安装包为".ezpkg"文件）
2. 解压缩".ezpkg"文件到一个临时目录，查看"package.xml"了解依赖包（依赖包在<dependencies></dependencies>中列出）。下载所有所有需要的依赖包。

附加功能

在 eZ Publish 3.7 与早期版本中，安装向导还包含另外一个步骤“站点功能”。这个步骤允许用户选择附加的功能。这个步骤已不再使用。附加功能可以在安装成功后，从安装包下载页面的"Content objects"章节下载安装包，并导入系统。

访问方法



这一步允许配置站点接受到一个访问时所接受的访问方法。有三种访问方法：

- URL
- 端口
- 主机名

URL

当使用 URL 访问方法时，eZ Publish 通过 URL（"index.php"之后的 URL）的内容来决定站点入口。 这中方法是默认的也是最常用的方法。如果是初次安装 eZ Publish，建议使用这种方法。

端口

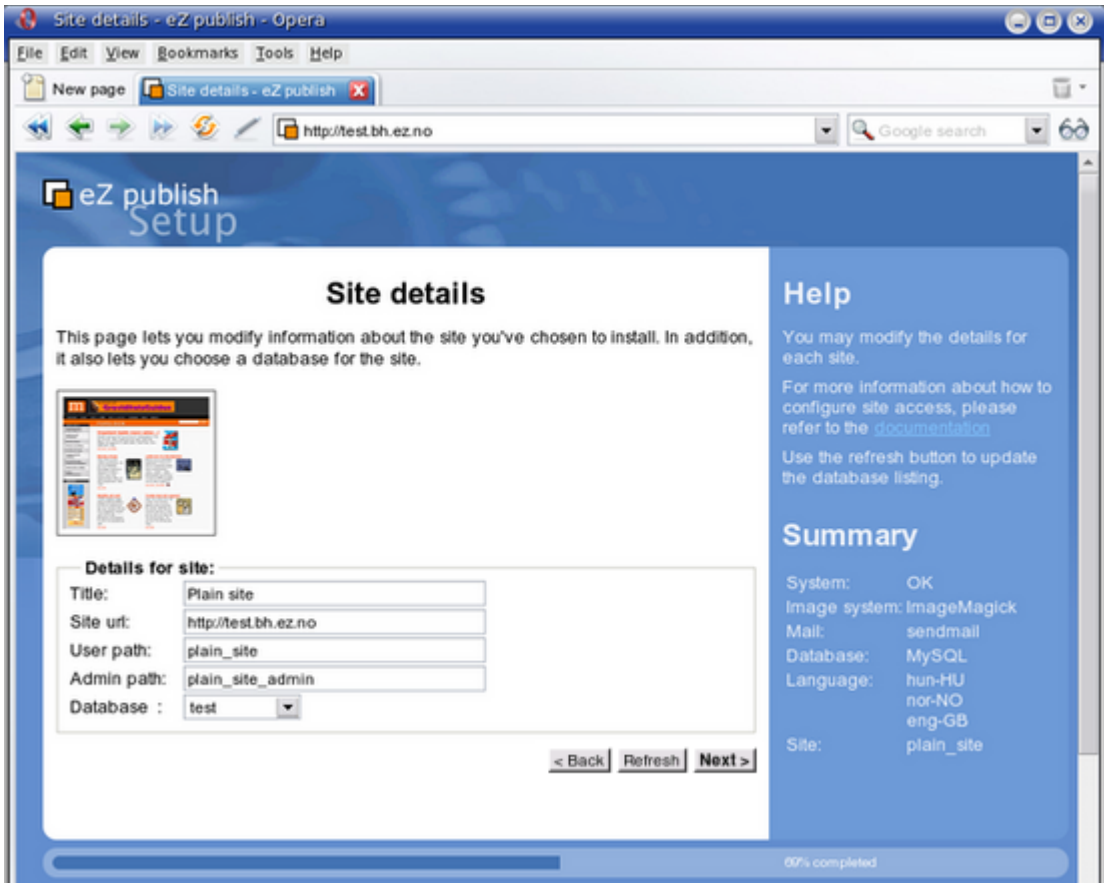
当使用这种方法时，eZ Publish 通过 URL 中的端口选择站点入口。端口必须追加到主机名后：“http://www.example.com:81/index.php"。这种方法需要额外配置 WEB 服务器和火墙。除非您完全了解您在做什么，否则不要使用这种方法。

主机名

当使用这种方法时，每个站点入口被帮定到一个独立的主机名。例

如："www.example.com"与"admin.example.com"可以被分别指派到公共站点入口和管理界面。这种方法需要额外配置 WEB 服务器和 DNS 服务器。除非您了解您在做什么，否则不要使用这种方法。

站点细节



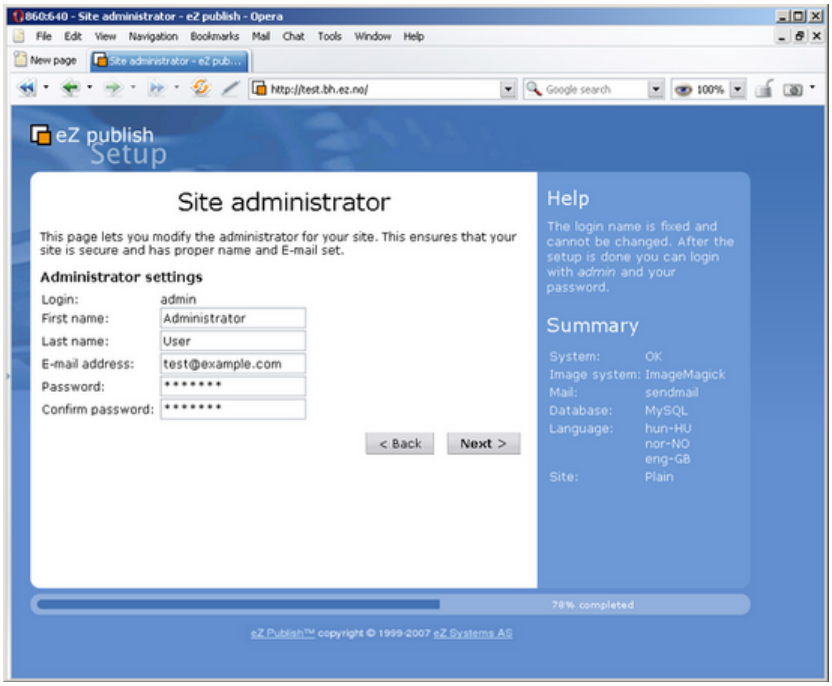
这个步骤允许修改站点相关的某些配置。注意，"User path"和"Admin path"取决于您使用的是何种访问方法。当使用端口访问方法时，在这里指定端口。如果使用 URL，"User path"和"Admin path"应该使用字符，数字和下划线("_")。如果使用主机名，可以使用其他附加的符号如："/",".",":"等但是不允许使用"_"。可用的数据库会在数据库下拉框中显示。可以用“刷新”按钮刷新数据库列表（如果您刚刚窗创建数据库）。数据库必须使用 UTF-8 字符集。

如果所选数据库已经包含数据，“站点细节”页面会重新显示并询问您下一步的操作。可能的操作：

- 保留现有数据并添加新数据
- 删除现有数据
- 保留现有数据并不做任何事
- 我已经选择了一个新的数据库

如果您选择了其它的数据库，请选择最后一个选项。

站点安全

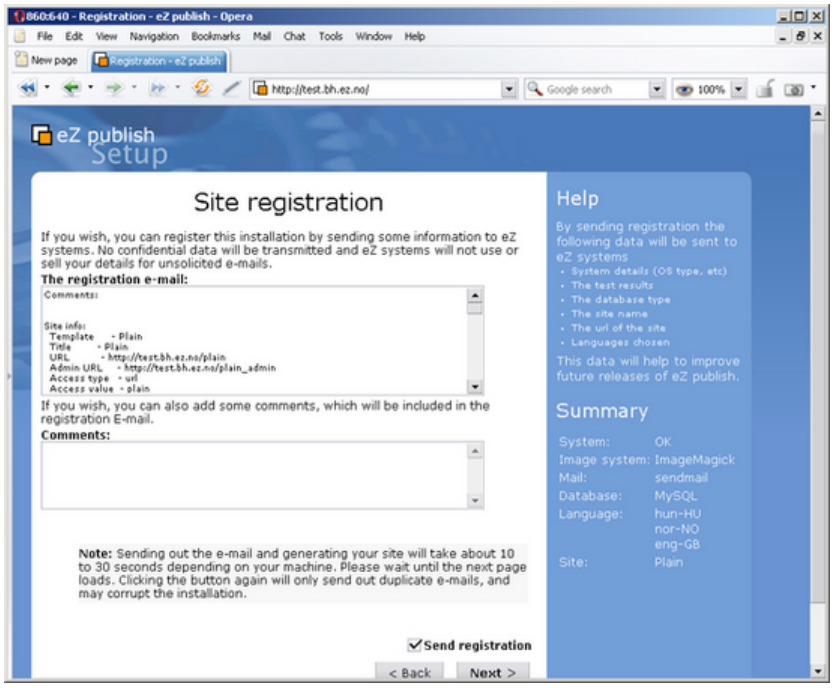


这一步骤建议一些基本的修改来保证站点的安全性。建议的安全修正保证配置文件不会被非法用户访问。如果您构建的不是一个公共站点，您不必担心安全问题。

注意：

管理员的登录名(login)默认被设置为"admin"并且不能为修改。如果您需要其他的管理员帐号，您可以安装 eZ Publish，创建一个新的管理员帐号，用这个用户登录，删除默认管理员帐号。

站点注册



这个步骤允许您控制是否向 eZ Systems 发送安装信息邮件。 发送的信息只会在 eZ Systems 内部用于统计和改善 eZ Publish。没有任何保密信息会被发送并且 eZ Systems 不会滥用或贩卖这些数据。以下数据会被发送：

- 系统细节（OS 类型等）
- 测试结果
- 数据库类型
- 站点名称
- 站点 URL
- 选择的语言

完成

安装向导已结束，现在可以使用 eZ Publish 了。点击任何一个链接访问不同的站点入口（公共站点入口，管理站点入口）。

注意：

安装成功后，您可以在"settings/override/site.ini.append.php"中设置"CheckValidity=true"来在下次访问站点时重新开始安装向导。

1.5 虚拟主机配置

本章介绍了如何在 Apache 中为 eZ Publish 配置虚拟主机。只有当你使用主机名访问方法（最安全的方法）时，您才需要配置虚拟主机。

通过使用虚拟注意，您可以在同一台服务器上运行多个站点。站点通常是通过它们各自的主机名来区分。Apache 会根据访问的域名使用相应的配置。

通用虚拟注意配置

虚拟主机通常在"httpd.conf"（Apache 服务器的主要配置文件）结尾处定义。您可以复制以下内容并替换方括号"[]"内部的内容。请参阅下一章查看真实的虚拟主机配置文件示例。

```
NameVirtualHost [IP_ADDRESS]


<VirtualHost [IP_ADDRESS]:[PORT]>
    <Directory [PATH_TO_EZPUBLISH]>
        Options FollowSymLinks
```

```
        AllowOverride None
    </Directory>

    <IfModule mod_php5.c>
        php_admin_flag safe_mode Off
        php_admin_value register_globals 0
        php_value magic_quotes_gpc 0
        php_value magic_quotes_runtime 0
        php_value allow_call_time_pass_reference 0
    </IfModule>

    DirectoryIndex index.php

    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteRule content/treemenu/?$ /index_treemenu.php [L]
        Rewriterule ^/var/storage/.* - [L]
        Rewriterule ^/var/[^/]+/storage/.* - [L]
        RewriteRule ^/var/cache/texttoimage/.* - [L]
        RewriteRule ^/var/[^/]+/cache/texttoimage/.* - [L]
        Rewriterule ^/design/[^/]+/(stylesheets|images|javascript)/.* - [L]
        Rewriterule ^/share/icons/.* - [L]
        Rewriterule ^/extension/[^/]+/design/[^/]+/(stylesheets|images|javascripts?)/.* -
[L]
        Rewriterule ^/packages/styles/.+/(stylesheets|images|javascript)/[^/]+/.* - [L]
        RewriteRule ^/packages/styles/.+/thumbnail/.* - [L]
        RewriteRule ^/favicon\.ico - [L]
        RewriteRule ^/robots\.txt - [L]
        # Uncomment the following lines when using popup style debug.
        # RewriteRule ^/var/cache/debug\.html.* - [L]
        # RewriteRule ^/var/[^/]+/cache/debug\.html.* - [L]
        RewriteRule .* /index.php
    </IfModule>

    DocumentRoot [PATH_TO_EZPUBLISH]
    ServerName [SERVER_NAME]
```

```
    ServerAlias [SERVER_ALIAS]


</VirtualHost>
```

[IP_ADDRESS]	虚拟主机的 IP 地址，例如："128.39.140.28"。Apache 允许使用通配符("*")。
[PORT]	WEB 服务器的监听端口。端口配置是可选配置，默认端口为 80。IP 地址与端口的组合通常被称为套接字。Apache 允许使用通配符("*")。
[PATH_TO_EZPUBLISH]	eZ Publish 安装目录的绝对路径。如："/var/www/ezpublish-3.6.0"。
[SERVER_NAME]	Apache 需要监控的 IP 地址或主机名。如果匹配，对应的虚拟主机配置会被使用。
[SERVER_ALIAS]	Apache 需要监控的附加的主机名/IP 地址。如果匹配，对应的虚拟主机配置会被使用。

注意：

"mod_rewrite"模块必须在"httpd.conf"中启用，才能使用 Rewrite Rules。

NameVirtualHost

"NameVirtualHost"配置可能已经存在。如果定义新的 NameVirtualHost 会造成冲突。如果“ Apache 报告如“NameVirtualHost [IP_ADDRESS]没有 VirtualHost ”或“ 不能混用*端口和非*端口的 NameVirtualHost 地址 ”的错误，尝试注释 NameVirtualHost 这一行。参阅 http://httpd.apache.org/docs/1.3/mod/core.html#namevirtualhost 了解更多关于 NameVirtualHost 的内容。

SOAP 与 WebDAV

如果您希望在 eZ Publish 中使用 SOAP 与/或 WebDAV，您需要在虚拟主机配置文件中添加以下内容：

```
RewriteCond %{HTTP_HOST} ^webdav\..*
RewriteRule ^(.*) /webdav.php [L]


RewriteCond %{HTTP_HOST} ^soap\..*
RewriteRule ^(.*) /soap.php [L]


ServerAlias soap.example.com
ServerAlias webdav.example.com
```

1.5.1 配置示例

本示例演示了如何在 Apache 中为 eZ Publish 配置虚拟主机。假设 eZ Publish 安装在"/var/www/example"，并且可以通过以下 URL 访问：

- http://www.example.com (公共站点入口)
- http://admin.example.com （管理界面站点入口）

为了达到这个目的，我们需要配置 Apache 和 eZ Publish 从而保证它们可以正确相应不同的请求。

配置 eZ Publish：站点入口配置

eZ Publish 需要被配置为使用主机名访问方法。您可以通过安装向导或手动编辑"settings/override/site.ini.append.php"。典型的配置如下：

```
...
[SiteAccessSettings]
AvailableSiteAccessList[]
AvailableSiteAccessList[]=example
AvailableSiteAccessList[]=example_admin
MatchOrder=host

HostMatchMapItems[]=www.example.com;example
HostMatchMapItems[]=admin.example.com;example_admin
...
```

以上配置要求 eZ Publish 为"www.example.com"使用"example"站点入口，为"admin.example.com"使用"example_admin"站点入口。参阅“基本概念”中的“站点管理”章节了解更多站点管理的内容。

Apache 配置：虚拟主机配置

假设。。。

- eZ Publish 被安装在"/var/www/example"
- 服务器 IP 地址为 128.39.140.28
- 我们希望通过"www.example.com"和"admin.example.com"访问站点

以下虚拟主机配置需要被添加到"http.conf"结尾处：

```
NameVirtualHost 128.39.140.28

<VirtualHost 128.39.140.28>
    <Directory /var/www/example>
        Options FollowSymLinks
        AllowOverride None
    </Directory>

    <IfModule mod_php5.c>
        php_admin_flag safe_mode Off
        php_admin_value register_globals    0
        php_value magic_quotes_gpc  0
        php_value magic_quotes_runtime  0
```

```
        php_value allow_call_time_pass_reference 0
    </IfModule>

    DirectoryIndex index.php

    <IfModule mod_rewrite.c>
        RewriteEngine On
        RewriteRule content/treemenu/?$ /index_treemenu.php [L]
        Rewriterule ^/var/storage/.* - [L]
        Rewriterule ^/var/[^/]+/storage/.* - [L]
        RewriteRule ^/var/cache/texttoimage/.* - [L]
        RewriteRule ^/var/[^/]+/cache/texttoimage/.* - [L]
        Rewriterule ^/design/[^/]+/(stylesheets|images|javascript)/.* - [L]
        Rewriterule ^/share/icons/.* - [L]
        Rewriterule ^/extension/[^/]+/design/[^/]+/(stylesheets|images|javascripts?)/.* -
[L]
        Rewriterule ^/packages/styles/.+/(stylesheets|images|javascript)/[^/]+/.* - [L]
        RewriteRule ^/packages/styles/.+/thumbnail/.* - [L]
        RewriteRule ^/favicon\.ico - [L]
        RewriteRule ^/robots\.txt - [L]
        # Uncomment the following lines when using popup style debug.
        # RewriteRule ^/var/cache/debug\.html.* - [L]
        # RewriteRule ^/var/[^/]+/cache/debug\.html.* - [L]
        RewriteRule .* /index.php
    </IfModule>

    DocumentRoot /var/www/example
    ServerName www.example.com
    ServerAlias admin.example.com
</VirtualHost>
```

注意：

您不必为"admin.example.com"配置单独的虚拟主机，它可以作为"ServerAlias"添加。

您可以在同一个虚拟主机内使用 apache1 和 apache2。这种方式允许用户为两个服务器中使用同一个虚拟主机。

```
<IfModule mod_php5.c>
# If you are using Apache 2, you have to use <IfModule sapi_apache2.c>
# instead of <IfModule mod_php5.c>.
        # some parts/addons might only run safe mode on
        php_admin_flag safe_mode Off
        # security just in case
        php_admin_value register_globals    0
        # performance
        php_value magic_quotes_gpc  0
        # performance
        php_value magic_quotes_runtime  0
        #http://www.php.net/manual/en/ini.core.php#ini.allow-call-time-pass-reference
        php_value allow_call_time_pass_reference 0
</IfModule>

<IfModule sapi_apache2.c>
# If you are using Apache 2, you have to use <IfModule sapi_apache2.c>
# instead of <IfModule mod_php5.c>.
        # some parts/addons might only run safe mode on
        php_admin_flag safe_mode Off
        # security just in case
        php_admin_value register_globals    0
        # performance
        php_value magic_quotes_gpc  0
        # performance
        php_value magic_quotes_runtime  0
        #http://www.php.net/manual/en/ini.core.php#ini.allow-call-time-pass-reference
        php_value allow_call_time_pass_reference 0
</IfModule>
```

1.6 删除 eZ Publish

本章介绍了如何从系统中完整删除 eZ Publish 安装。

删除 eZ Publish 需要以下几个步骤：

1. 删除 eZ Publish 目录
2. 删除数据库

3. 重新配置 Apache（可选）
4. 删除 cronjob（可选）

注意!

以下步骤会删除 eZ Publish 和所有的数据与内容。所有的删除动作均不可恢复。

删除 eZ Publish 目录

用您习惯的工具删除 eZ Publish 安装目录。

Linux/UNIX

在 linux/UNIX 平台，可以使用"rm"命令：

```
$ rm -Rf /path/to/ez_publish
```

注意：

某些文件/目录的权限可能已经混乱。这种情况下，普通用户可能无法删除这些文件/目录。您需要用 root 删除。

Windows

Windows 用户可以在文件管理器中删除 eZ Publish 目录。

删除数据库

MySQL

1. 启动 MySQL 客户端，用您的用户名，密码登录

```
$ mysql -u <username> -p
```

如果用户名/密码正确，客户端会显示"mysql>"提示符。

2. 用"drop"命令删除数据库：

```
mysql> drop database <database-name>;
```

PostgreSQL

1. 在命令行中用"dropdb"命令删除数据库

```
$ dropdb <database-name>
```

重新配置 Apache（可选）

如果使用了虚拟主机配置，很可能 Apache 配置文件中包含 eZ Publish 相关的配置。因为这些配置将不再被使用，所以可以将其删除。用文本编辑器打开"httpd.conf"，滚动到文件结尾处，删除 eZ Publish 相关的虚拟主机配置。修改配置后，需要重新启动 Apache。

删除 cronjob（可选）

Windows 用户应该跳过这部分。如果您配置了 eZ Publish 的 cronjob，那么应该删除他们。您可能需要编辑全局的 cron 文件（在"/etc/cron*"）或用"crontab -e"命令来编辑用户私有的 cron 文件。从 cron 文件中删除 eZ Publish 相关的记录。

1.7 扩展

eZ Publish 的插件被称为扩展。扩展可以提供附加的功能。eZ Publish 可以安装多种扩展。所有的 eZ Publish 扩展都具有统一的安装方法。本章会说明如何安装扩展：

1. 解压扩展压缩文件
2. 激活扩展

某些扩展需要额外的操作才可以正常工作，如：创建新的数据库表，添加若干 eZ Publish 内容类等等。类似的操作会在每个扩展自己的文档中说明。

如前所述，本章只涉及扩展安装的基本步骤。出于演示的目的，我们会尝试安装一个称为"ezfoo"的扩展。

1.7.1 解压文件

每个扩展都以一个压缩文件的形式发布。压缩文件的文件名由扩展的名称和版本构成。此外，压缩文件的类型由文件的后缀名表示。后缀名可能为"tgz","tar.gz","bz2",或"zip"。例如：

- ezfoo-extension-1.0.tgz
- ezfoo-extension-1.0.tar.gz
- ezfoo-extension-1.0.bz2
- ezfoo-extension-1.0.zip

Windows 用户应该下载"zip"格式的压缩文件。对于 Linux/UNIX 用户，只要对应的解压工具已安装，可以下载任何格式的文件。

扩展的基本目录

复制下载的压缩文件至 eZ Publish 安装目录下的"extension" 目录。如果这个目录还不存在，可以创建同名目录。（不要用复数"extensions"作为目录名。）

在 linux/UNIX 平台，可以用以下的命令行创建目录并复制压缩文件。

```
mkdir /opt/ezp/extension/

cp /home/myuser/download/ezfoo-extension-1.0.tar.gz /opt/ezp/extension/
```

用真实的 eZ Publish 安装目录替换"/opt/ezp/"，用真实的压缩文件下载目录替换"/download/ezfoo-extension-1.0.tar.gz"。

解压缩压缩文件

在"extension"目录中解压缩压缩文件。如果操作无误，在"extension"目录中会创建一个"ezfoo“目录。

参阅下表了解 Linux/UNIX 平台上正确的解压命令行，取决于您要解压的文件格式：

压缩文件格式	解压命令行

tar.gz 或 tgz	`tar -zxvf ezfoo-extension-1.0.tar.gz`
	或
	`tar -zxvf zfoo-extension-1.0.tgz`

bz2	`tar -jxvf ezfoo-extension-1.0.bz2`
zip	`unzip ezfoo-extension-1.0.zip`

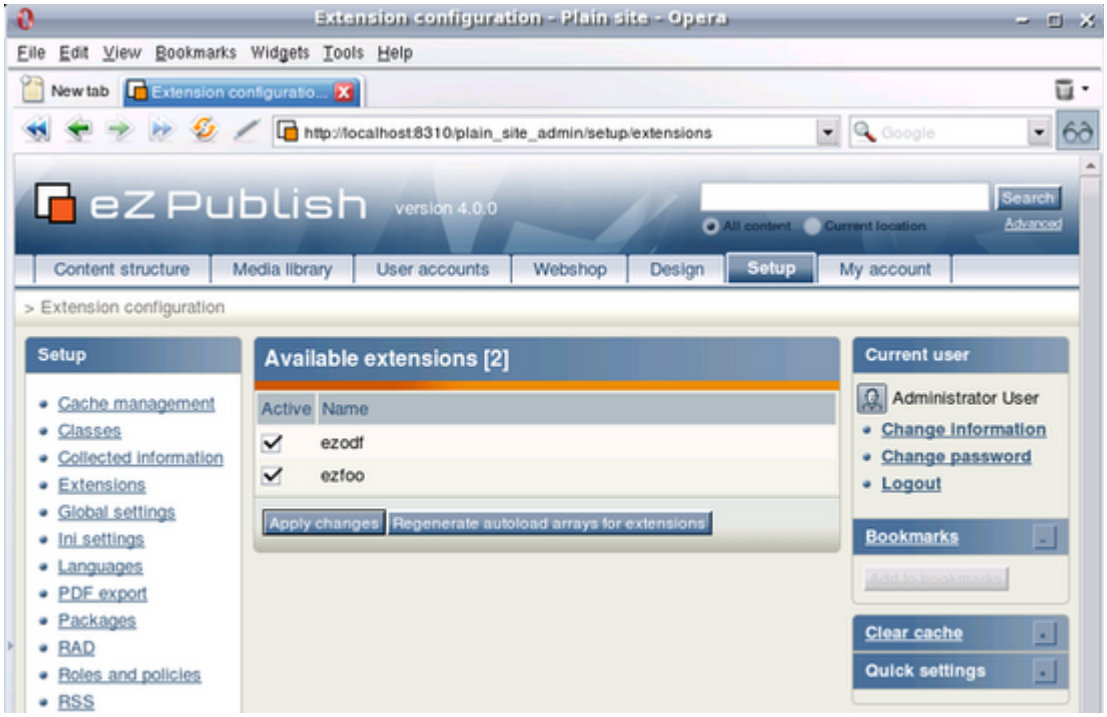
在 windows 平台，您可以用系统内建的 zip 特性来解压"zip"文件。文件会因该被解压至"extension/ezfoo"。

1.7.2 激活扩展

每个扩展都需要被激活，这意味者需要将扩展注册到 eZ Publish 框架中。每个扩展可以通过 eZ Publish 管理界面或配置文件激活。此外，您可以将扩展注册为全局扩展或某些站点入口的扩展。

管理界面

登录到 eZ Publish 管理界面，点击“设置”标签，然后点击左侧的“扩展”菜单。您可以看到一个显示所有可用扩展的列表与对应的复选框。要激活示例扩展，勾选下图中的复选框然后点击“应用”按钮。



这会将示例扩展激活为全局扩展。

配置文件

除了管理界面外，扩展也可以在"site.ini"中手动激活。

激活为全局扩展

要将示例扩展激活为全局扩展，编辑"settings/override/site.ini.append.php"。在"[ExtensionSettings]"下添加以下内容：

```
ActiveExtensions[]=ezfoo
```

"[ExtensionSettings]"中可以添加多个扩展。如果文件或配置章节不存在，您需要手动创建文件和/或配置章节。

激活为站点入口扩展

要将示例扩展激活为"example"站点入口的扩展，编辑"settings/siteaccess/example/site.ini.append.php"文件。在"[ExtensionSettings]"中添加以下内容：

```
ActiveAccessExtensions[]=ezfoo
```

注意：

激活扩展的这一行为"ActiveAccessExtensions"，而不是"ActiveExtensions"。如果文件和/配置章节不存在，您可以手动创建。

更新 autoload 数组

更新配置文件后，您需要执行"ezpgenerateautoloads.php"脚本将所有在扩展中的 PHP 类定义添加到"autoload/ezp_extension.php"文件中，否则 eZ Publish 可能无法执行新添加的扩展。以下的示例演示了如何执行脚本。

1. 进入 eZ Publish 目录
2. 用以下命令行执行脚本：

```
bin/php/ezpgenerateautoloads.php --extension
```

这个脚本会在"extensions"目录中检测 PHP 类定义并相应地更新"autoload/ezp_extension.php"。

1.8 常见问题

本章节将会解释如果安装由于一些不明原因而失败，对应的解决方法。

首先，确保所有的系统需求无一例外地得到满足。这些系统需求是很严格，也是很非常重要的。因此，请仔细阅读。

如果所有的系统需求都满足而安装仍然有问题，建议您检查安装过程中的调试信息。要启用调试输出，采用以下步骤：

1. 进入"settings/override"目录
2. 创建"site.ini.append.php"并加入以下内容：

```
[DebugSettings]

DebugOutput=enabled
```

调试信息会在页面底端显示，参阅下图。

2 基本概念

本章的目的是介绍与解释 eZ Publish 中最重要的概念。任何 eZ Publish 的开发新手都绝对应该仔细阅读这一章以了解基本术语，模型，结构和系统的各个组成部分。本章的介绍更一般性而非技术性，更倾向于传授概念和不是解释技术细节。对 eZ Publish 不了解的用户应该可以获得足够的信息来帮助他们理解以下内容：

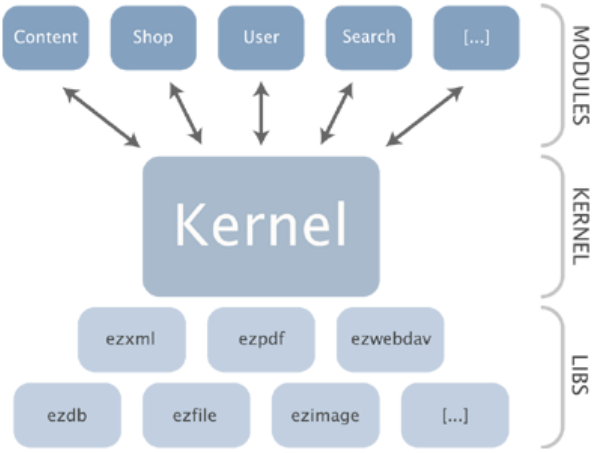
- eZ Publish 是如何构建的
- 主要的目录结构
- 内容与界面独立的概念与必要性
- eZ Publish 如何保存与管理内容
- eZ Publish 如何处理界面相关的问题
- eZ Publish 如何管理不同站点
- 模块与视图的概念
- eZ Publish 处理 URL 的方法
- 配置系统
- 工作流系统的结构
- 访问/权限系统如何工作
- 网络商店如何工作
- 典型的页面请求周期

2.1 eZ Publish 内部结构

本章通过从不同层次展示 eZ Publish 的概述来描述 eZ Publish 的内部结构。eZ Publish 是一个复杂的，基于 PHP 的，面向对象的应用程序。这个系统由三个主要部分构成：

- 库
- 内核
- 模块

下图演示了这三部分如何彼此连接。



库

库是系统的主要构成部分。库由很多一般的，可重用的 PHP 类构成。库不依赖 eZ Publish 内核。但是，它们中的某些类彼此紧密联系，因此无法分割。如果您需要了解一般 PHP 库，可以查看 eZ Publish 根目录中的"lib"目录。“参考手册”章节包含当前版本库的完整列表与简要说明。

内核

eZ Publish 内核可以被描述为系统的核心。它负责处理所有底层的功能，如：内容处理，内容版本管理，访问控制，工作流，等等。内核由各种基于或使用一般库的引擎构成。

模块

eZ Publish 模块提供了一种 HTTP 接口用来以 WEB 的方式与系统交互。虽然某些模块提供了调用内核功能的接口，其它的模块或多或少独立于内核存在。eZ Publish 包含了一系列模块以满足典型日常事务的需求。例如：content 模块提供了一种接口可以通过浏览器管理内容。“参考手册”章节包含了所有当前版本的完整列表与简要介绍。一个模块可以被分解为以下部分：

- 视图
- fetch 函数

视图提供了实际的 WEB 接口。例如："content"模块的"search"视图提供了一种调用内建检索引擎的 WEB 接口。每个 eZ Publish 模块至少提供一种视图。Fetch 函数可以在模板中被调用来从模块中提取数据。例如："user"模块中的"current_user"fetch 函数可以用来访问与当前登录的用户相关的数据。某些模块提供 fetch 函数，某些没有。

2.1.1 目录结构

eZ Publish 根目录包含多个子目录。每个子目录对应系统的一个特殊部分并且包含一系列逻辑上彼此关联的文件。下表是 eZ Publish 主要目录的一览。

目录	描述

bin	"bin"目录包含了各种 PHP,Perl 和 shell 脚本。例如：它包含了"ezcache.php"脚本可以用来从命令行删除所有 eZ Publish 缓存。这些脚本主要用于手动维护系统。
cronjobs	"cronjobs"目录包含了各种定期自动执行的维护脚本。
design	"design"目录包含了所有界面相关的文件，如：模板，图像，式样表等等。
doc	"doc"目录包含文档和系统变更日志。
extension	"extension"目录包含了 eZ Publish 插件。eZ Publish 扩展系统允许外部代码嵌入 eZ Publish 并与 eZ Publish 系统的其余部分共存。通过使用扩展，可以创建新的模块，数据类型，模板操作符，工作流事件等等。
kernel	"kernel"目录包含了所有的内核文件，如：核心内核类，模块，视图，数据类型等等。这里是系统的核心所在。只有专家才可以染指这部分。
lib	"lib"目录包含了一般用途的库。这些库由一些列完成各种底层操作的类构成。内核需要调用这些库。
packages	"packages"目录包含了绑定的安装包（主题，内容类，模板等等）。这些安装包可以通过安装向导或管理界面安装。
settings	"settings"目录包含动态的，站点关联的配置文件。
share	"share"目录包含静态的配置文件，如：代码页，区域描述，翻译，图标等等。
support	"support"目录包含了附加应用程序的源代码，可以用来完成各种高级任务。例如：它包含"lupdate"程序，可以用来创建和维护 eZ Publish 翻译文件。
update	"update"目录包含各种辅助系统升级的脚本。
var	"var"目录包含缓存与日志文件。它还包含那些没有保存在数据库中的内容（图像和文件）。这个目录的大小会随着系统的运转而增长。

2.2 内容与界面

本章解释了内容与界面的基本概念。理解内容与界面的本质，他们如何彼此联系，系统如何处理这些基本的元素非常重要。

内容

在 eZ Publish 的世界里，内容与界面是彼此分离的。当谈论内容时，我们是指会通过某种结构组织并保存的信息。例如：内容可以是一篇新闻的内容（标题，简介，正文，图片），一辆汽车的属性（厂商，车型，年份，颜色）等等。换言之，所有为了将来提取而保存的自定义信息均可以称为内容。

界面

内容结构中保存的信息总需要被以某种人类容易接受的方式显示。内容意味着真正的数据，界面则关系到内容如何被标志与显示。当谈论界面时，我们实际上在讨论构成一个 WEB 界面的元素：HTML，式样表，不属于内容范畴的图像等等。

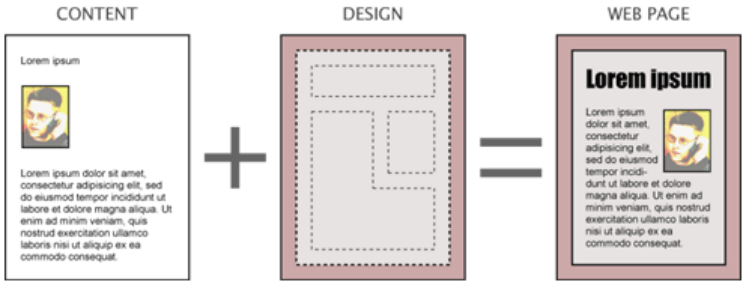
模板

eZ Publish 把模板作为构成界面的基础单元。例如：一个模板可能定义了一个页面的顶端显示标题，而下端显示主题内容。当这个页面被访问的时候，如何将真正的内容汇入模板中的正确位置就成为了内容管理系统的工作。eZ Publish 的模板基本上就是一种定制过的 HTML 文件，它描述了特定类型的内容如何被视觉化。除标准的 HTML 语法外，在模板中也可以使用 eZ Publish 的特殊代码来执行例如：从系统

中提取数据之类的任务。系统内建的模板中所使用的 HTML 语法符合 XHTML1.0 Transitional 规范。

内容与界面分离

内容是关于保存与组织自定义/原始数据，而界面的目的是定义内容应该如何被视觉化。内容与界面的组合成就了一个完整的用户界面，如下图所示。



这种内容与界面分离的特点，以及系统处理它的方式是 eZ Publish 的一个主要特性。内容与界面的分离早就了一种无法用其他方法达到的广阔的可能性。以下的列表简单列举了这种技术的一些重要优点：

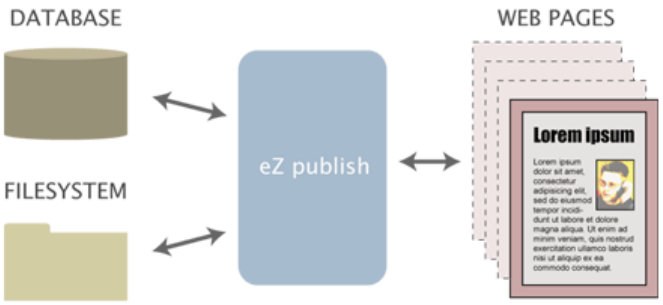
- 内容作者与设计师可以并行工作而没有冲突
- 内容可以被发布为多种格式
- 内容可以很容易地被转移
- 全局的重新设计/变化可以通过简单的修改达成

2.2.1 存储

本章解释了 eZ Publish 在何处保存站点（并非系统本身）的数据。典型的 eZ Publish 站点又以下元素构成：

- 真实的内容
- 界面相关的文件
- 配置文件

真实内容被组织并保存在数据库中。但是也有例外，对于图像和文件，它们被保存在文件系统中。这样做的主要原因在于，当存取大容量的文件时，文件系统比数据库快得多。将文件保存在文件系统中可以允许 WEB 服务器直接发送它们而无须访问数据库。此外，这种方法也令外部工具管理/扫描/索引上传文件的内容变得相对容易。例如：内建的检索引擎支持使用外部的工具为多种类型文件（PDF，Word 文档，Excel 电子表格等等）的内容建立索引。将文件保存在文件系统中可以很大程度上减少数据库的大小并因此令数据库的复制和处理变的更容易。所有与界面相关的文件（模板文件，CSS 文件，非内容图片等等）与配置文件也被保存在文件系统中。因此，一个 eZ Publish 站点的备份必须包含数据库的 dump 文件与必须文件的备份。下图演示了 eZ Publish 如何利用数据库与文件系统保存站点的不同元素。



	Content	Design	Settings
	✔ Text ✔ Structure	✘	✘
	✔ Images ✔ Files	✔ HTML + CSS ✔ Templates	✔ Ini files

2.3 内容管理

内容管理系统的角色是一个负责组织和保存任何类型和复杂度内容的系统。这样一个系统的主要目标是提供一个结构清晰，自动且灵活的解决方案来保证信息可以在各种不同的沟通渠道（如：互联网，内部网以及其它各种前台和后台系统）间自由地发布并持续地更新。本章描述了 eZ Publish 如何真正地处理内容。

典型的例子

让我们想象这样一个场景：一所大学需要保存学生相关的信息。大多数内容管理系统会提供一系列内建的内容类型。这些内容类型中，可能有一种称为"Person"的类型。这种内容类型可能由诸如“姓名”，“生日”，“电话号码”等等属性构成。然而，实际的学生信息可能无法完全匹配那些预定义的模型，因为学生信息可能包含一些大学特有的信息（如：学号，部门等等）。尽管某些系统允许创建自定义的结构，其解决方案往往过于复杂也费时费力。可能需要同时修改代码与数据库。此外，一旦应用了这种方案，系统日后的维护很可能会成为一个问题。

eZ Publish 中的内容管理

与其它系统不同，eZ Publish 并不使用预定义的“一劳永逸”的解决方案。与绝望地试图将数据填入预定义的结构中相反，eZ Publish 允许以面向对象的方法创建自定义的结构。例如：站点开发人员可以开发自定义的内容结构来完美的匹配大学的内容存储需求。这是令 eZ Publish 成为一个灵活，成功系统的一个重要特性。除提供自定义结构的自由度以外，eZ Publish 还允许在运行时修改内容的结构。换言之，如果此例中的学生结构需要修改，eZ Publish 会根据管理员的指令自动对其进行修改。

尽管创建和修改内容结构的可能是一种不错的特性，您并不是总需要用到这种特性。因此 eZ Publish 的发行版本中会内建若干预定义的内容结构并因此允许开发人员做以下选择：

- 使用标准/内建的结构

- 使用修改过的标准/内建的结构
- 只使用自定义的结构
- 使用标准，修改过的与自定义的结构的组合

面向对象的内容结构

eZ Publish 内容结构基于面向对象世界中那些流行的编程语言（Smalltalk, C++, JAVA 等）中的编程思想。

总体来讲，面向对象意味着任何事物均为对象。在显示生活中，我们周围存在着很多对象：家具，汽车，宠物，人类等等。每种对象都有区别于其它对象的特性。这也是 eZ Publish 定义与管理内容的方法。

系统提供了一系列基础的单元与机制来共同提供一种灵活的内容管理解决方案。真实的数据结构由内容类定义。内容类由属性构成。一个属性可以被想象成为一个字段，例如：在用于保存学生信息的结构中的“生日”字段。对整个学生的描述将会被定义为“学生类”。一个类属性的特性由代表这个属性的数据类型决定。

理解内容类只是某种结构的定义非常重要。换言之，内容类本身描述内容结构却并不保存任何真实数据。当一个内容类被定义之后，您可以创建这个内容类的实体。一个内容类的实体被称为内容对象。真实的数据被保存在不同类的内容对象中。一个内容对象包含一到多个版本。版本这一层次允许为相同内容保留多个版本。每个版本包含一个至多个翻译。翻译层次允许为同一个内容的同一个版本创建多种语言的版本。一个翻译由多个属性构成。属性是内容结构链中的最后一级，这里才是真正保存数据的地方。

内容对象被关联至节点。节点被系统以树状结构的形式组织并保存。这种树状结构常被称为节点树。后面的章节包含了关于以上概念更详细的说明。

2.3.1 数据类型

数据类型是最小的存储单元。它定义了某种特殊类型的信息因该如何被验证，保存，格式化等等。eZ Publish 内建了一套基础数据类型，可以用来构建功能强大，复杂的内容结构。此外，您可以开发自定义的数据类型对系统进行扩展。自定义的数据类型必须用 PHP 开发。然而，内建的数据类型已经足以满足各种典型的应用。下表列出了 eZ Publish 内建的最基本的数据类型。

数据类型	描述
文本行	保存单行純文本。
文本块	保存多行纯文本。
XML 块	验证并保存多行 XML 内容。
整数	验证并保存一个整数。
浮点数	验证并保存一个浮点数。

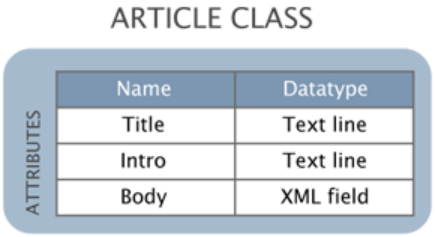
请参阅“数据类型”章节了解全部内建数据类型。此外，附加的数据类型可以从 http://ez.no/community/contribs/datatypes 下载；他们由 eZ Publish 社区用户创建。

输入验证

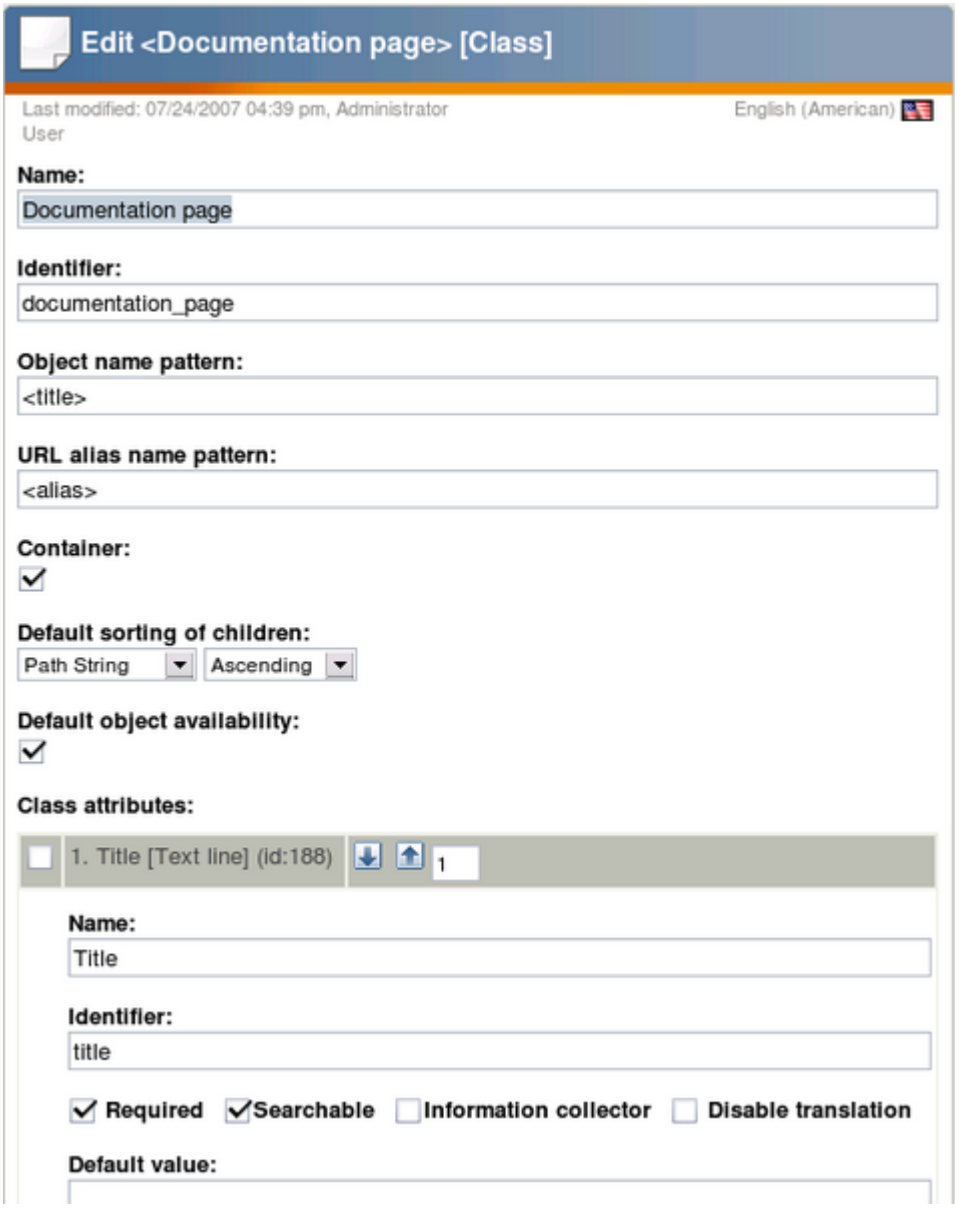
如上表所示，某些数据类型不止负责保存数据。例如：“XML 块”显然支持内容验证。这意味着输入的 XML 会被经过验证才会写入数据库。换言之，只有 XML 是合法的，系统才会接受并保存输入的数据。大部分（不是所有）内建数据类型都支持输入验证。输入验证不能被启用/禁用。换言之，如果某个数据类型刚好支持输入验证，它将会一直验证输入的数据，因而系统永远都不允许非法的数据保存如这种数据类型中。

2.3.2 内容类

内容类定义了某种特殊的数据结构。它本身并不保存任何真实数据。内容类由属性构成。属性的特性由属性对应的数据类型决定。通过组合不同的数据类型，可以描述复杂的数据结构。下图演示了一个称为“文章”的类，它定义了用于保存新闻文章的数据结构。它由用于保存标题，简介文字和主体内容的属性构成。



eZ Publish 发行版本内建了一套一般用途的内容类，这些类可以用于典型的 WEB 应用。例如：默认的图像类定义了用于图像文件的结构。它由用于保存图像名称，图像文件，标题与 ALT 文本的属性构成。内建的类可以被修改以更好地适应特殊用途。此外，您也可以创建新的内容类。内容类在可以在管理界面简单地被创建，修改，删除。当一个内容类被删除，所有这个类的实体（包含真实数据）也会被从系统删除。下图演示了内容类的编辑界面。



类结构

内容类由以下元素构成：

- 类名
- 标识符
- 对象名模式
- URL 别名模式

- 容器标记
- 默认子节点排序规则
- 默认对象可用标记
- 属性

类名

类名用来为类保存一个用户友好的名称。类名可以由字符，数字，空格和特殊字符构成。类名最大长度为 255 个字符。例如：如果一个类定义了关于毕业学生的数据结构，这个类可以被命名为“毕业学生”。类名用于在管理界面中各种有关类的列表中显示，但是系统内部并不使用类名。如果类名为空，系统会在保存类定义的时候自动为类分配一个唯一的类名。

标识符

标识符用于系统内部，特别是在配置文件，模板和 PHP 代码中使用。类标识符只可以包含小写字符，数字和下划线。标识符的最大长度为 50 个字符。例如：如果某个类定义了关于毕业学生的数据结构，标识符可以为"graduate_student"。如果类标识符为空，eZ Publish 会在保存类定义时自动为其分配一个唯一的标识符。

对象名模式

对象名模式控制内容类对象（类实体）的名称应该如何生成。模式通常由属性标识符（稍后解释）构成。通过这种方式要求 eZ Publish 哪些属性应该被用来生成对象名。每个用到的属性标识符都应该由尖括号环绕。尖括号外的文本会被直接包含在生成的对象名中。如果对象名模式为空，eZ Publish 会默认使用第一个属性标识符来定义对象名模式。

URL 别名模式

URL 别名模式控制在对象（类实体）被创建时，对象节点的虚拟 URL 应该如何被生成。注意，URL 别名模式只影响虚拟 URL 的最后一部分。URL 别名模式与对象名模式的工作原理相同。尖括号外的文本会根据指定的 URL 变换方法进行变换。如果 URL 别名模式为空，eZ Publish 会使用对象名。

容器标记

容器标记控制是否允许这个类包含子项目（常被称为子节点）。这个设置只影响管理界面，目的是希望为管理员和编辑提供更方便的工作环境。换言之，它并不控制任何实际的底层逻辑，只是控制管理界面的显示方式。

默认子节点排序规则

从 eZ Publish3.9 版本开始，您可以在编辑类的时候同时指定默认的子节点排序规则。当创建了新对象之后，它们对应的节点在左侧的节点树中显示的时候，它们会根据其父节点类定义中的子节点排序规则来排列。换言之，如果您设置“文件夹”类的默认子节点排序规则为“按优先级升序”，那么在文件夹下新创建的子节点会按照它们的优先级升序（由小到大）排列。

注意：

可以通过子节点窗口中的排序控件每个节点单独指定子节点排序规则。修改类级别的子节点排序规则不会影响到那些已经存在的节点（只对新创建的对象有效）。参阅“排序规则”和“排序顺序”了解更多。

默认对象可用标记

这个标记与都语言特性（自 eZ Publish3.8 开始）有关。它只用于标记新类实体（对象）的可用性。您可以进一步（在对象级别）控制这个标记，可以通过管理界面中“翻译”窗口中的“使用主语言如果没有优先翻译”复选框来控制。换言之，你可以单独指定每一个对象的对象可用性。如果设置了这个标记，当对象没有可用的翻译时，会使用对象的初始/主语言来显示对象。如果没有设置这个标记，当对象没有可用翻译时，对象不会被显示。

属性

如前所述，在内容类中，真正构成数据结构的是属性的结构和类型。一个内容类至少有一个属性。另一方面，理论上内容类的属性数是没有限制的。任何时候，您都可以通过管理界面创建，删除或重新排列类属性。如果添加了一个类属性，所有这个类的现存对象和新对象都将拥有这个新属性。如果一个类属性被删除，它也会从所有对象中消失。

尽管可以通过管理界面删除和添加属性，在某些情况下这种操作会导致数据库崩溃。这种问题往往由于太多对象需要更新。如果程序的处理时间超出 PHP 的最大执行时间，处理流程会被中断而数据库会处于不稳定的状态。在写本文档时候，这个问题可以通过增加 PHP 的最大执行时间来解决。可以在"php.ini"中修改"max_execution_time"。默认值为 30 秒，应该修改为若干分钟。另外一种更可靠的解决方案（通过一个 PHP 脚本在命令行删除/添加类属性）可能在将来增加到系统中。

2.3.3 类属性

类由类属性构成。每个类属性由一种数据类型代表。每个类属性的特性有它的数据类型决定。属性由以下元素构成：

- 属性名
- 标识符
- 一般控件
- 数据类型相关控件

属性名

属性名为属性保存用户友好的名称。例如：如果一个属性用于保存生日，这个属性可以被命名为“生日”。这个名称会在管理界面中各种与类属性有关的页面中显示，但不会在系统内部使用。属性名可由字符，数字，空格和特殊字符构成。属性名的最大长度为 255 个字符。如果属性名为空，eZ Publish 会在保存类定义的时候，为其自动分配一个唯一的属性名。

标识符

属性标识符在系统内部使用。特别是在配置文件，模板和 PHP 代码中使用。属性标识符只能包含小写字符，数字和下划线。属性标识符的最大长度为 50 个字符。例如：如果某个类属性用于保存生日，它的

标识符可以为"date_of_birth"。如果属性标识符为空，保存类定义的时候，eZ Publish 会为其自动分配一个唯一的标识符。

一般控件

没个属性都有一套一般控件。对于所有的属性，这些控件均相同，无论（但并非毫无关系）属性为何种数据类型。一般控件是一套可以被开启或关闭的开关：

- 必填项
- 可检索
- 信息收集器
- 可翻译

必填项

必填项开关控制内容对象（内容类实体）的保存流程。无论属性为何种数据类型，均可以使用这个开关。当对某些属性启用必填项开关，系统会拒绝保存对象直到所有的必填项都被输入内容。如果必填项开开关被关闭，eZ Publish 不会关注属性中是否有输入。新属性的必填项开关默认是关闭的。请注意，无论必填项开关启用与否，系统还是会根据属性数据类型的验证规则验证输入的数据。大部分（不是全部）内建的数据类型都支持内容验证。下例演示了这些特性如何工作。

假设定义一个类用来保存囚犯数据（想不通，为什么是囚犯？:P）。类属性如下：姓名，编号，生日，监号，管区等等。如果至少把姓名和生日设置为必填项，就可以减少数据错误的可能。如果生日属性使用内建的“日期”数据类型，系统会验证输入的日期格式。

可检索

这个开关可以控制这个属性中保存的数据是否可以被内建的检索引擎编入索引。大部分内建数据类型都支持索引。请参阅“引用”中的“数据类型”章节了解哪些数据类型支持索引。

信息收集器

这个开关可以控制属性在显示模式中的行为。默认的显示模式行为是回显编辑模式中输入的内容。例如：查看一篇文章的时候，文章的内容可以显示，但不能编辑。但是，如果属性被标记为信息收集器，在显示模式中，可以为这个属性输入内容。这种起初令人费解设计其实很实用。比如：可用来快速创建反馈表单。表单被提交后，提交的内容可以以邮件形式发送给系统管理员或其他帐号。只有一小部分内建数据类型支持信息收集器。下例演示了如何用这种技术创建反馈表单。

假设已创建了一个类“反馈表单”，属性如下：名称，标题，反馈内容。将标题和反馈内容标记为信息收集器。当显示这个类的实体（对象）时，标题和反馈内容会被显示为输入框，另外页面会有一个提交按钮。

可翻译

这个开关控制属性的数据应该只存在一种语言（默认语言）版本还是可以被翻译为多种语言版本。翻译机制与数据类型无关。换言之，这个开关对任何数据类型都有效。

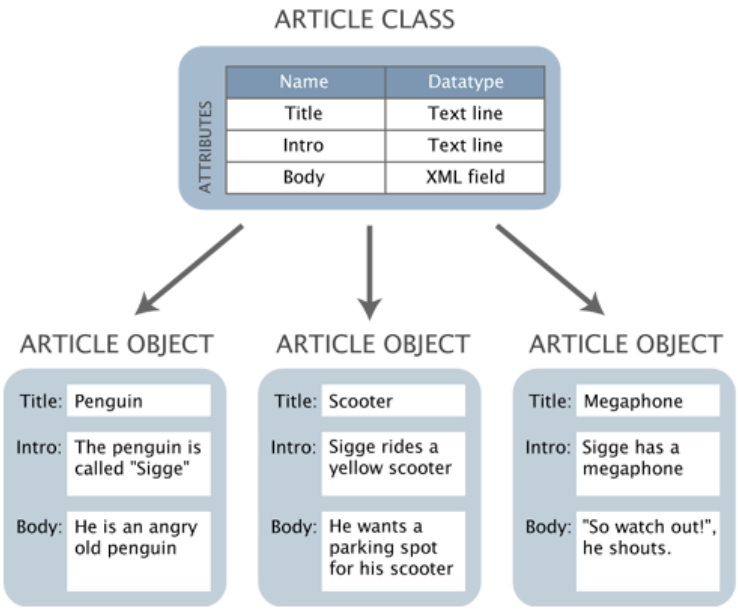
新建属性的这个属性默认为开启。这个开关的用处主要是对某些属性禁用翻译。比如：翻译日期，数字，价格，电子邮件等属性的意义就不大。

数据类型控件

除了一般控件外，某些数据类型还有一些附加的控件。某些数据类型允许定制，某些不允许。例如：内建的“文本行”数据类型有两个设置：默认值和最大长度。

2.3.4 内容对象

内容对象是内容类的实体。类只定义了内容结构，数据保存在内容对象中。内容类定义后，可以创建这种类的对象（实体）。例如：定义了存储新闻文章的类以后，就可以创建若干新闻文章对象（每个对象保存着不同的文章内容）。下图说明了数据类型，属性，内容类和内容对象间的关系。



注意：

上图是一个简化版本。它没有体现对象的全部结构，因为版本和翻译层次被省略了。以下内容更深入地探讨了内容对象结构。版本和翻译层次会在稍后的章节解释。

对象结构

内容对象由以下元素构成：

- 对象 ID
- 对象名
- 类型
- 属主

- 创建时间
- 修改时间
- 状态
- 分区 ID
- 版本
- 当前版本

对象 ID

每个对象都有一个唯一的数字 ID。系统用这个 ID 来组织和标识不同的对象。这些 ID 不会循环使用。换言之，如果对象被删除了，这个对象的 ID 不会被其他对象使用。

对象名

对象名只是对象的一个用户友好的名称，对象名在管理界面中各种与对象有关的页面中显示。对象名帮助用户通过名称而不是对象 ID 来定位对象。发布对象时，对象名会被自动生成。对象名的生成规则有对象名模式定义。对象名模式允许用对象属性生成对象名。因为对象名不用于系统内部，对象名可以重复。

例如：编辑新闻文章时，文章标题很适合用于生成对象名。发布文章时，文章对象名与标题相同。每次发布对象时，对象名都会被重新生成。换言之，如果标题改变了，对象名也会相应变化。

类型

类型表示对象的类。

属主

对象的属主包含到首次创建对象的用户的引用。任何时候，一个对象只能属于一个用户。对象第一次被创建时，系统会设置属主。属主不能被编辑，而且即使对象属主对应的用户被删除，属主也不会改变。

创建时间

创建时间是对象第一次被创建时的时间戳。创建时间由系统设置并不能被修改。无论对象作何改变，创建时间都不会改变。

修改时间

修改时间记录对象最后被修改时的时间戳。这个信息由系统设置而且不能被修改。对象每次被修改时，修改时间会改变。

状态

显示对象的状态。有三种状态：

- （0）草稿

- （1）发布
- （2）归档

新创建的对象，状态为“草稿”。在对象被发布之前，它的状态一直是“草稿”。发布之后变为“发布”状态。发布之后，对象不能变回“草稿”。当发布的对象被移入回收站后，状态变为“归档”。如果对象被从回收站删除（或直接删除而不放入回收站），对象会被物理删除。

分区

分区 ID 显示对象属于哪个分区。每个对象只能属于一个分区。通过指派不同分区，逻辑上可以将对象分成很多组。参阅“分区”章节了解更多。

版本

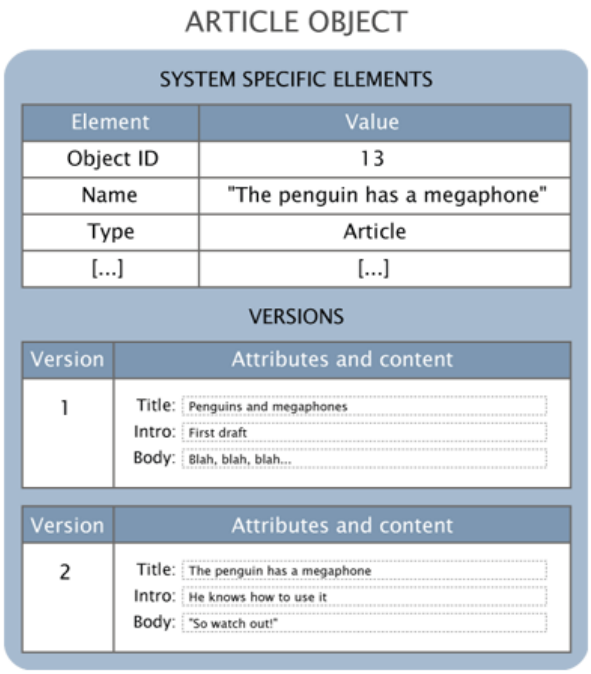
真正的数据保存在不同的版本中。一个版本可以被想象为属于某个用户的某个时间戳的数据（对象属性）集合。每次对象的内容被编辑时，系统会创建一个新版本。被编辑的永远是最新的版本。当前/发布的版本与以往的版本都不会被修改。这样用户可以撤销修改。对象总是至少有一个版本。每个版本有一个唯一的版本号。每个新版本会被系统自动分配一个自增的版本号。版本机制在下一章详细讨论。

当前版本

当前版本是当前发布版本的版本号。如上所述，对象可以有多个版本。但是其中只有一个版本可以是当前版本（也被称为发布版本）。当前/发布版本是用于显示的版本。

2.3.5 对象版本

eZ Publish 有一套内建的对象版本系统。这种机制允许对象内容（属性）有多个版本。它基本上提供一种一般的，可以直接使用的版本控制框架。这种机制可用于任何内容。不同的版本被封装在对象内。下图演示了更详细的对象结构。



每次对象被编辑，系统都会创建一个新版本。被编辑的永远都是最新的版本，旧版本不会被改变。eZ Publish 是这样保留不同用户的修改。用户可以很容易的撤销修改并恢复到之前的版本。

版本限制

由于每次修改都会产生一个新版本（除非新版本被放弃），数据库会很快被不同的版本充满。为了防止这种情况，版本系统可以被限定为每个对象最多允许保留若干版本。可以为不同的对象（类）设置不同的版本限制。默认的限制为 10，就是说每个对象最多保留 10 个版本。达到这个限制后，最旧的版本会被删除，空出的位置可以用于保留新版本。这是默认的行为。您也可以设置当版本限制达到后，之后手动删除某个版本后，才能创建新版本。

版本结构

版本由以下元素构成：

- 版本号
- 创建时间
- 修改时间
- 创建者
- 状态
- 翻译

版本号

每个版本都有一个唯一的版本号。系统用版本号组织和保管不同版本的对象。对象内每个版本的创建都会被分配一个自动自增的版本号。

创建时间

创建时间包含版本被创建时的时间戳。它由系统设置且不论版本作何改变均不会改变。

修改时间

修改时间包含版本最后被修改时的时间戳。每次版本被保存或发布，系统会设置它。当版本被发布时，对象的修改时间也会被修改（简单地设置为与版本的修改时间相同）。

创建者

版本的创建者包含一个到创建版本的用户的引用。尽管一个对象只能属于一个用户（由“属主”字段表示），每个版本却可以属于不同的用户。版本被创建时，系统会更新版本的创建者。它不能被修改且即使创建者帐号已经被删除也不会改变。

状态

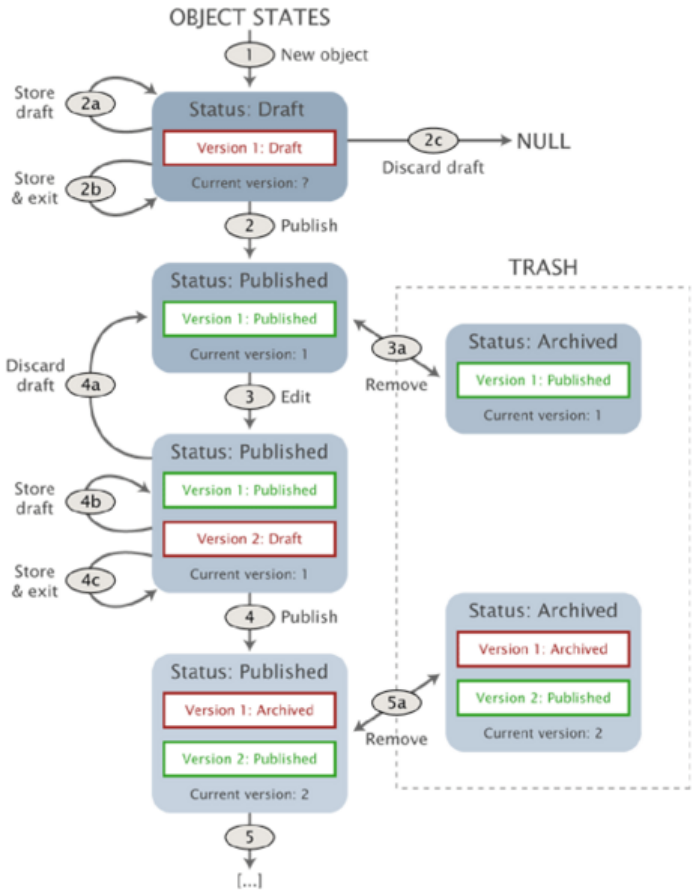
版本有五种状态：

- 草稿（0）
- 发布（1）
- 待定（2）
- 归档（3）
- 拒绝（4）

在 eZ Publish3.8 以后的版本，还有另外一种可能：如果版本被创建但是没有被修改（例如：某人点击了“添加评论”按钮但是没有真正提交），版本的状态将转为“内部草稿（5）”。在管理界面中，状态"5"被称为“为修改草稿”。从 3.9 版本开始，您可以设置内部草稿被保留的天，小时，分钟和秒。超过这个限制的内部草稿可以用"internal_drafts_cleanup.php"cronjob 脚本来删除。另外一个 cronjob 脚本"old_drafts_cleanup.php"可以被配置用来删除已经存在了特定时间的状态为"0"的草稿。

新创建的版本是一份草稿。这个状态会一直保留直到版本被发布。尽管对象可以有多个版本，但是只有一个发布版本（其余的版本通常为草稿或归档）。发布的版本可以为认为是当前版本且它是对象显示的时候用到的版本。发布的版本不能再次称为草稿。但是当另外一个版本被发布时，它将变为归档。

下图演示了版本系统如何工作。



上图演示了内容对象最常见的状态。当新对象被创建（第 1 步），eZ Publish 会同时创建一个新的草稿版本。由于对象还没有被发布，它的状态被设置为草稿且当前版本为之。保存草稿（第 2a 步，第 2b 步）不会改变对象的状态。唯一会发生的事情是内容被保存至版本 1。如果草稿（唯一存在的版本）被放弃，对象即被物理删除（第 2c 步）。当草稿被发布（第 2 步），草稿和对象的状态都被设置为发布。此外，当前版本号被设置为 1，意味着对象当前发布的版本为版本 1。发布后，对象内容可以被其他人查看。发布的对象可以被移入回收站或删除（第 3a 步）。当被移入回收站，对象的状态被设置为“归档”。对象可以从回收站恢复，恢复后的状态为移除前的状态（状态被重新设置为发布）。当发布的对象被编辑（第 4 步）时，当前版本（这个例子中，版本 1）不会被修改，一个全新的版本会被创建。新版本（版本 2）的内容直接从版本 1 复制。同样，保存草稿（第 4b 和 4c 步）不会改变对象的状态。如果草稿被放弃（第 4a 步），它会被物理删除，因而对象会恢复到编辑之前的状态。如果新创建和编辑的草稿被发布，它会成为对象的当前版本，因而之前的版本（在本例中，版本 1）会被设置为“归档”。第 5a 步，演示了如果对象（现在有两个版本）被移入回收站会发生什么。

待定和拒绝状态在协作系统中使用。当一个版本等待编辑的审批时，状态被设置为待定。如果版本得到批准，它会被自动发布，因而状态被设置为发布。反之，如果待定版本被编辑拒绝，状态被设置为拒绝。只有草稿才可以被编辑且一个版本只能被版本的创建者编辑。此外，被拒绝的版本也可以被编辑。被拒绝的版本被编辑时，会重新变为草稿。发布和归档版本不能被编辑。但是，可以复制它们。当发布或归档版本被复制时，附件被设置为草稿，因而可以被编辑。如果新草稿被发布，系统会自动将前一个发布版本设置为归档且新草稿会成为发布版本。

翻译

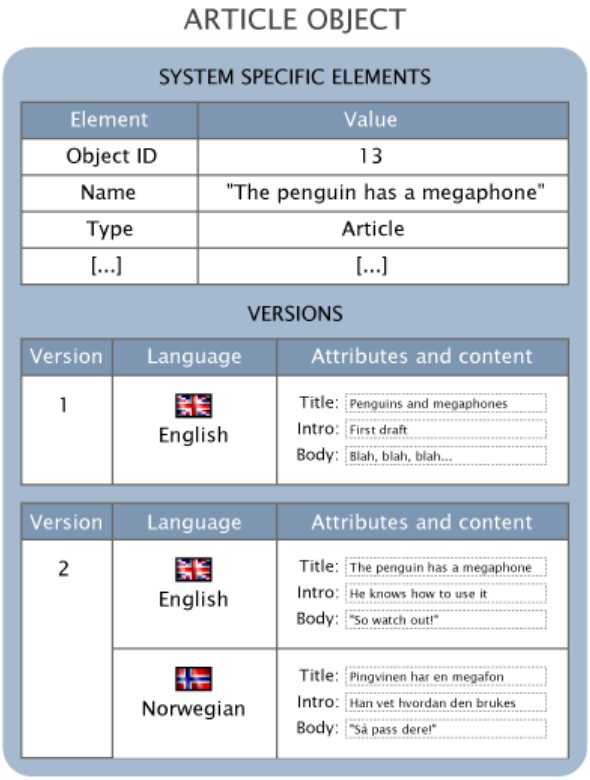
版本的内容实际上保存于不同的翻译内。翻译是信息在某种语言的表现。换言之，翻译层次允许对象的某个版本存在于多种语言中。版本总是至少有一个内容的翻译（代表默认/标准语言的内容）。

2.3.6 多语言

除版本系统外，eZ Publish 的内容模型还提供了内建的多语言框架。这个特性允许对象的内容存在于多种语言中。系统同时支持多达 30 种不同的语言。

多语言特性提供了一种一对一的翻译机制，可用于翻译任何内容。一对一的翻译方案可以确保内容得到精准的翻译。例如：如果一篇新闻文章的内容存在于英文，挪威语和匈牙利语（同样的内容，不同语言的翻译），我们说我们有对于这篇文章的一对一翻译。翻译机制完全独立于数据类型。换言之，无论内容为何种数据类型，均可以被翻译。可以以一种语言作为开始，然后在需要的时候添加其它语言的翻译，从而扩展目标用户的范围。

下图演示了一个有两个版本的对象。每个版本存在于若干语言中。语言在这里常被称为翻译。



如上图所示，每个版本可以有不同的翻译。至少，一个版本总有一个翻译（初始/主翻译）。初始/主翻译不能被删除。但是，如果对象存在于多个语言中，可以选择哪个语言是初始/主翻译，从而可以删除之前的初始/主翻译。

注意：

从 3.8 版本开始，当用户编辑对象时，并不是整个版本被编辑，而是版本和翻译的组合得到编辑。这种方案避免锁定整个版本（包含所有翻译），因而允许多个翻译者同时对同一个对象进行不同语言的翻译。

全局的翻译列表

对象只可以被用全局翻译列表中的语言编辑/翻译。默认情况下，这个列表包含我们在安装向导第 6 步选择的语言。站点运转后，仍然可以添加附加的语言。下图演示了管理界面中的全局翻译列表（在“设置”和“语言”下）。

全局翻译列表只是保管了允许用户用于内容编辑和翻译的语言。添加到这个列表中的语言即可使用。从 3.8 版本开始，除非某种语言没有被任何对象使用，否则不允许删除这种语言。全局翻译列表支持多达 30 种语言。

3.8 和早期版本的区别

在 3.7 和早期版本中，对象必须先被创建为主语言，才能进行翻译。多个翻译者不能并行工作因为编辑流程锁定整个版本也包括所有翻译。

在 3.8 版本中，再没有主语言的概念，因而对象可以用任何语言创建。这意味着，您可以有一些文章只有英文翻译，而另一些文章只有挪威语的翻译。多个翻译者可以并行工作因为当编辑对象时，他们实际上在编辑翻译本身而不是整个版本。这意味着，如果您已经写了一篇英文文章，其他翻译者可以并行地为这篇文章添加附加的翻译（例如：匈牙利语，挪威语和俄语）。他们再也毋需等待因为他们可以同时对同一个对象同一个版本的不同翻译进行编辑。但是，这也意味着用户再也不能同时处理多个翻译。真正的问题在于，用户必须离开编辑界面才能添加（然后编辑）新的翻译。此外，还有其它一些影响。比如：除非用户在编辑对象的第一个版本，否则用户不能在编辑界面中修改对象的位置。但是，位置仍然可以在非编辑界面中的“位置”窗口内修改。

无论何时，当对象被发布时，系统会自动收集所有之前版本中的最新翻译，然后把这些翻译放置到发布版本中。结果是发布的版本包含了所有最新的翻译。对象的内容可以被翻译为最多 30 种语言。

请参阅“从 3.6.x(3.7.x)升级到 3.8.0 ”中“更新多语言 INI 配置”了解更多关于多语言相关的 INI 配置。

多语言类

从 3.9 版本开始，类名与类属性名也可以被翻译。换言之，例如：您可以用"Car"和"Bil"定以同一个类的英文名和挪威语名称。同时也可以用"Top speed"和"Topphastighet"给同一个属性命名。参阅“可翻译的类属性”章节了解更多。

不可翻译的属性

类定义数据结构，属性定义类，数据结构定义属性。类属性可被设置为可翻译或不可翻译。如果一个属性可以被翻译，在编辑这个类的对象时，系统会允许对这个属性进行翻译。典型的属性为包含真正文本的属性。例如：一篇新闻文章的正文部分可以被翻译为多种语言。但是，某些属性却不适合被翻译。典型的属性如：没有文本的图片，数字，日期，电子邮件等等。此类属性可以被设置为不可翻译，因而它们的内容可以从初始/主翻译直接复制。复制的内容不能被编辑。

例如：假设我们需要用多种语言保存家具信息。我们可以用以下属性创建家具类：名称，图片，简介，高度，宽度，深度和重量。显然，除了简介以外，对其它属性的翻译并没有意义。换言之，名称，图片，高度，宽度，深度和重量对于例如：英文和挪威文是一样的。不同计量单位之间的转换需要在模板中完成。

访问控制

可以控制是否允许某个用户（或某组用户）翻译内容。这个策略可以在类，分区，语言和属主级别上控制。值得一提的是，语言限制可以控制哪个用户（或用户组）可以用不同语言编辑/翻译内容的不同部分。此外，也可以控制到全局翻译列表的访问。这允许除系统管理员以外的用户在全局级别上添加和删除翻译。

请参阅“特性”章节中的多语言部分了解更多。

2.3.7 内容节点

系统开始运作后，新的内容对象可以被即时创建。例如：当撰写一篇新闻文章时，一个新的文章对象会被相应创建。显然，内容对象本身不能漂浮在空间中，他们必须以某种方式组织起来。这时就需要节点和节点树。内容节点只是对内容对象的封装。在 eZ Publish 中，每个对象可以由一个或多个几点表示。下图演示了内容节点和它对应的内容对象（被节点引用）在系统内部的实现。

内容节点树有节点构成。节点是对象在节点树中的位置。节点树用来以树状结构组织系统中的对象。节点树在下一章节解释。

节点结构

内容节点由以下元素构成：

- 节点 ID
- 父节点 ID
- 对象 ID
- 排序方法
- 排序顺序
- 优先级

节点 ID

每个节点有一个唯一的数字 ID。系统用这个 ID 来组织与保管不同的节点。这些 ID 不能循环使用。换言之，如果某个节点被删除，这个节点的 ID 不能被其他节点重用。

父节点

节点的父节点 ID 为节点树中父节点的节点 ID。

对象 ID

系统中的每个对象有一个唯一的数字 ID。节点的对象 ID 为这个节点封装的对象的 ID。

排序方法

排序方法定义如何排列节点的子节点。以下排序方法可用：

方法	ID	描述
类标识符	6	节点按照对象的类标识符排序。
类名	7	节点按照对象的类名排序。
深度	5	节点按照它们在树中的深度排序。根节点的深度为 1 ，每下降一层，深度加 1 。
修改时间	3	节点按照对象的修改时间排序。
子节点修改时间	10	节点按照它们子节点的修改时间排序。
名称	9	节点按照对象名排序。
路径	1	节点按照它们的路径排序。
优先级	8	节点按照它们的优先级排序。每个节点有一个优先级字段，用户可以在这个字段输入优先级。这种方法允许节点按照用户的要求排序。优先级字段在稍后解释。
发布时间	2	节点按照对象的当前/发布版本的发布时间排序。
分区	4	节点按照对象的分区 ID 排序。

注意：

可以把多种排序方法组合使用来实现复杂排序。但是，因为节点无法“记住”组合（您只能对每个节点指定一种排序方法和排序顺序），您必须在模板中实现复杂排序。

排序顺序

排序顺序决定排序的顺序。两种顺序可用：

- 降序(0/FALSE)
- 升序(1/TRUE)

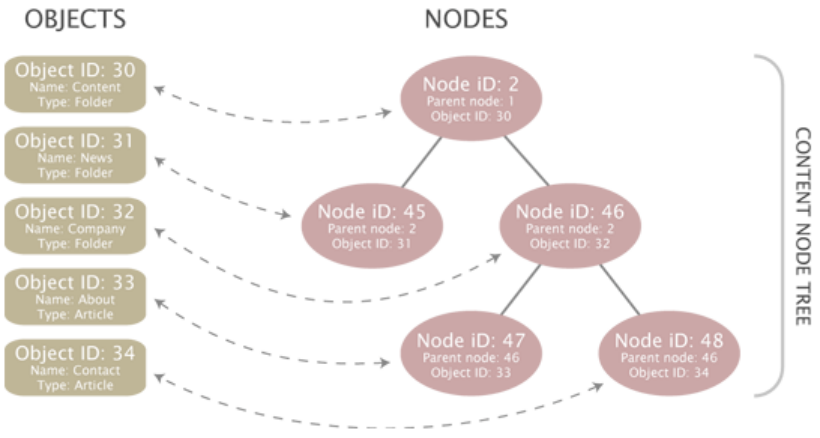
例如：如果排序方法为“名称”并且排序顺序为“升序”，子节点会按照字符表顺序从 A 到 Z 排列。如果排序顺序为“降序”，子节点会按照字符表顺序从 Z 到 A 排列。

优先级

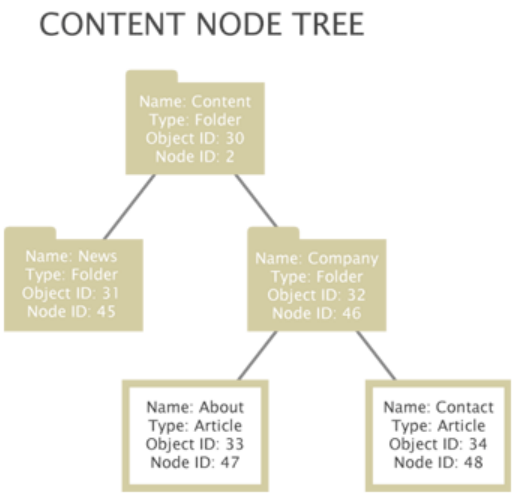
优先级字段允许用户为节点指派正整数，负整数和 0 为优先级。用这个字段可以按照用户的要求对节点排序。如果排序方法为“优先级”，子节点将会按照它们的优先级排列。

2.3.8 内容节点树

节点树是对象的树状结构。每个叶子是一个节点（也被称为位置）。每个节点引用一个对象。一般情况下一个对象被一个节点引用。因为节点封装了对象，任何类型的内容对象可以被放置在任何位置。节点树至少包含一个节点，称为根节点。根节点的 ID 为 1 。根节点是一个虚拟节点，它不封装任何真实对象。直接位于根节点下方的节点称为顶极节点（顶极节点在下一章解释）。理论上，节点树的深度和广度没有限制。下图演示了对象如何被节点引用，并共同构成节点树。

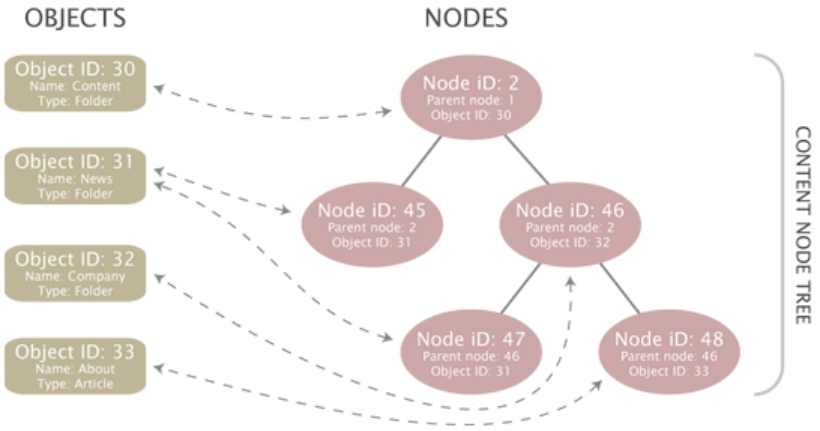


下图演示了同样节点结构的外部视图。

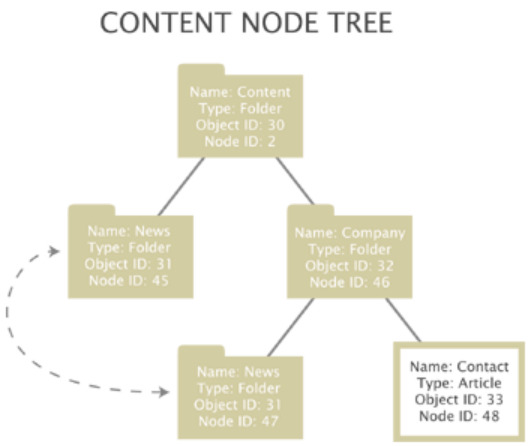


多位置

同一个对象可以被多个节点引用，这意味着同一个对象可以出现在节点树中的不同位置。这个特性可以用来例如：将某一篇新闻文章放置在两个位置：首页和归档页面。当使用多节点/位置时，只有一个节点可以作为对象的主节点。主节点通常代表对象在节点树中的原始位置。如果对象只被一个节点引用，那么当然这个节点就是主节点。主节点被用作避免重复的检索结果，无限递归循环，智能过滤等用途。下图演示了一个对象有多个位置的节点结构。



下图演示了相同结构的外部视图。



易犯的错误

在计划内容结构时，一个很容易犯的错误是把多位置想象成为文件系统中的快捷方式/链接。不幸的是，节点树并不是这样工作的。当为对象添加新位置时，eZ Publish 不会遍历并为对象原始位置的子节点添加位置。例如：如果一个文件夹包含若干子文件夹，子文件夹包含文章，图像等等。当为这个文件夹指派第二个位置时，它的子文件夹和子文件夹不会自动出现在文件夹的新位置下。

附加说明

只有发布的对象会出现在节点树中。新创建的对象（草稿）在第一次发布前没有被指派的节点。只有所有引用某个对象的节点都被从节点树移除，才能说这个对象被移除（状态设置为归档）。被移除的对象会出现在回收站。需要牢记，eZ Publish 的回收站是只用扁平结构。这与操作系统中的回收站不同。回收站中的对象可以被恢复到原始位置，但是只有在它们的原始位置的父节点没有被删除时才可行。否则，用户必须在恢复过程中手动为其指定一个备选的/新的位置。

注意：

无论对象的原始父节点是否仍然存在，用户都可以为其手动指定恢复的位置。

此外，如果一个包含新闻文章的文件夹被移除，文件夹和文章都会在回收站中以同一级别显示。只恢复文件夹不会自动恢复文件夹下的文章，因为当移除节点时，文件夹和文章之间的关联已经丢失。这种情况下，需要先恢复文件夹。之后，每篇文章需要被手动恢复并指定位置。

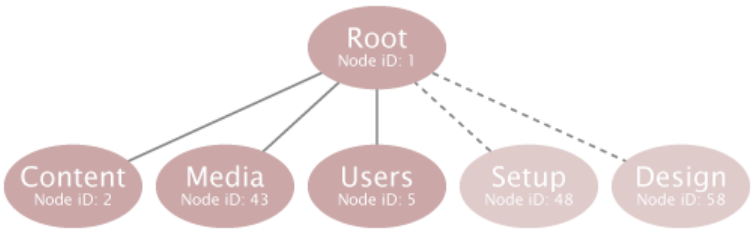
2.3.9 顶级节点

典型的 eZ Publish 安装会建立以下顶极节点：

- 内容
- 媒体
- 用户
- 设置

- 设计

顶极节点不能为删除。但是，它们可以与其他节点互换。互换可以用来修改顶极节点的类型。例如：“内容”节点引用一个文件夹对象。通过把它与另外一个不同类型的节点互换，可以替换它的类型。下图演示了虚拟根节点和标准的顶极节点：



内容

站点的真实内容保存在“内容”节点下。这个节点通常用来组织文件夹，文章，信息页等内容。因此，定义了站点的内容结构。通过遍历这个顶极节点的内容，可以很容易地生成站点地图。“内容”节点的默认 ID 为 2。这个节点的内容可以通过在管理界面中选择“内容结构”标签来查看。默认情况下，这个节点引用一个“文件夹”对象。

媒体

“媒体”节点的典型应用是用来保存和组织会被“内容”节点下的节点频繁引用的对象。它通常包含图像，动画，文档和其他文件。例如：它可以用来创建图片集保存可被其他文章使用的图片。“媒体”节点的默认 ID 为 43。在管理界面中选择“媒体”标签查看它的内容。默认情况下，这个节点引用一个“文件夹”对象。

用户

内建的多用户解决方案使用 eZ Publish 原生的内容结构。一个用户实际上就是一个包含“用户帐号”数据类型的类的实体。用户节点被保存在“用户”顶极节点下的“用户组”节点下。换言之，“用户”顶极节点实际上包含用户组和用户。“用户”节点的默认 ID 为 5。在管理界面中选择“用户帐号”标签查看这个节点的内容。默认情况下，这个节点引用一个“用户组”对象。

设置

“设置”节点包含各种与配置有关的节点。这个节点只用于系统内部。“设置”节点的默认 ID 为 48。默认情况下，这个节点引用到一个“文件夹”对象。

设计

“设计”节点包含各种与设计相关的节点。这个节点只用于系统内部。“设计”节点的默认 ID 为 58。默认情况下，这个节点引用到一个“文件夹”对象。

2.3.10 节点可见性

因为发布意味着添加对象（以节点的形式）到内容树，撤销发布则意味着从内容树中移除对象。一旦对象被发布就不能被撤销因为 eZ Publish 不直接提供类似的功能。相反，系统提供了一种隐藏机制来改变节点的可见性。隐藏功能可以用来阻止发布的内容被显示。这是通过拒绝对节点的访问来达到的。单个节点或节点子树可以被用户或系统隐藏。节点可以有以下的可见性状态：

- 可见
- 隐藏
- 由上级隐藏

所有的节点默认为可见，因此他们引用的对象都可以被访问。用户可以通过管理界面隐藏/显示节点。节点被隐藏后，它的所有子孙节点都会被标记为“由上级隐藏”，因而也变为隐藏。如果某节点的父节点被隐藏，则这个节点不能被设置为可见。

隐藏的节点将变为不可用，除非"[SiteAccessSettings]"配置区域中的"ShowHiddenNodes"被设置为"true"。这个配置可以在"site.ini"的重设文件中设置。最常用的配置方法为：对所有除管理界面外的站点入口不允许访问隐藏节点。

实现

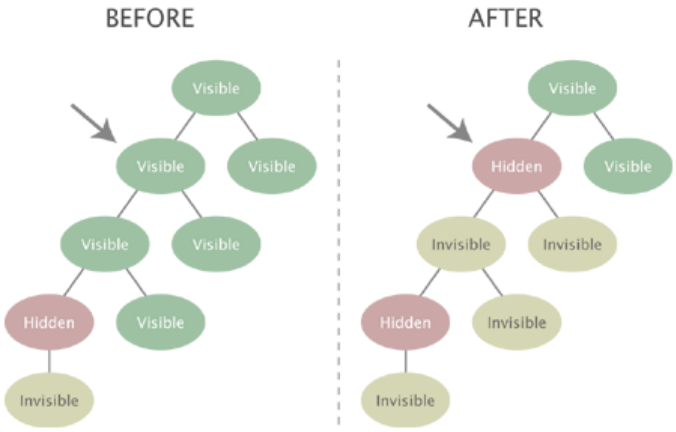
每个节点有两个标记："H"和"X"。"H"意味着“隐藏”，"X"意味着“不可见”。隐藏标记表示这个节点是否被用户隐藏。"X"为"true"意味着这个节点不可见因为它被用户或系统隐藏。两个标记共同确定了内容树的可见性：

H	X	状态
-	-	节点可见
1	1	节点不可见。因为被用户隐藏。
-	1	节点不可见。被系统隐藏因为祖先节点被隐藏或不可见。

如果用户尝试隐藏一个已经不可见的节点，在不可见标记之外，这个节点的隐藏标记也会被设置。如果一个节点被隐藏并且它的父节点变为可见，这个节点仍然为隐藏状态，而它的子孙节点仍然为不可见状态。下图演示了隐藏算法如何工作。

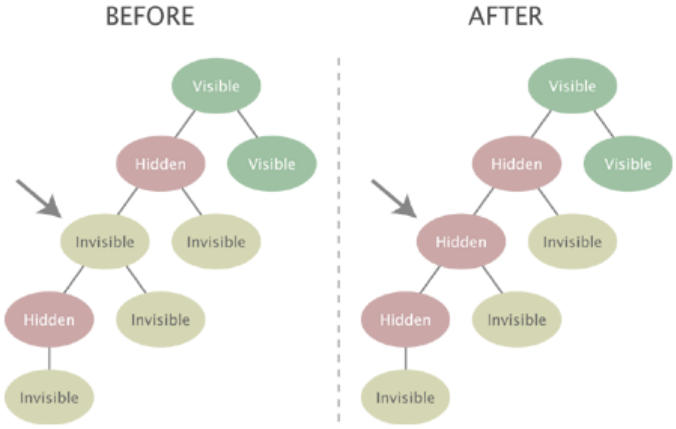
场景 1：隐藏一个可见节点

下图演示了当一个可见节点被用户隐藏后的结果。这个节点会被标记为隐藏。子孙节点会被标记为不可见（由上级隐藏）。已经被标记为隐藏或不可见的子孙节点不会变化。



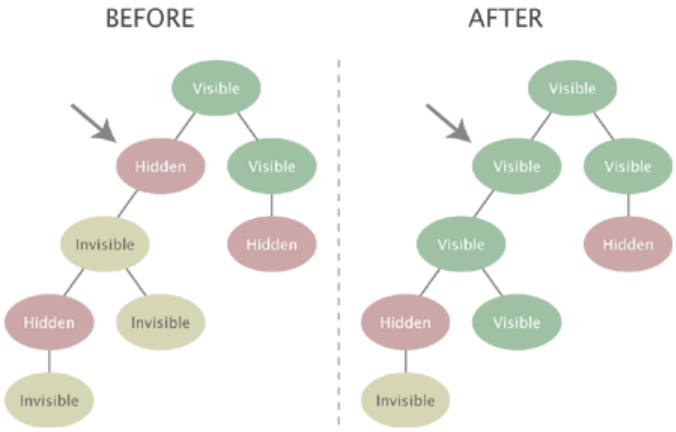
场景 2：隐藏一个不可见节点

下图演示了当一个不可见节点（由上级隐藏）被用户明确地隐藏后的结果。这个节点会被标记为隐藏。因为子孙节点已经为隐藏或不可见状态，它们的可见性不会改变。



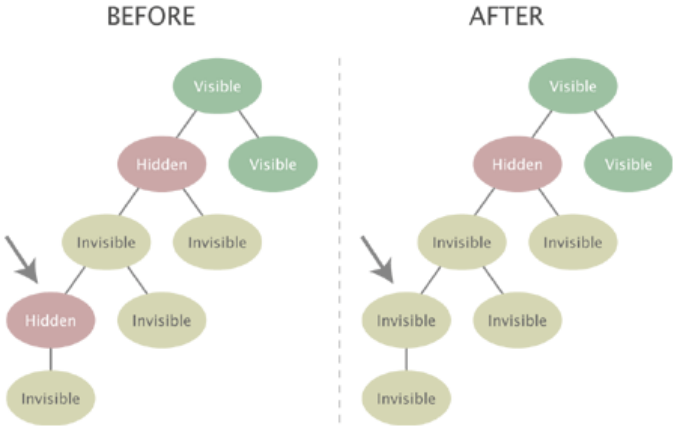
场景 3：显示一个父节点可见的节点

下图演示了当用户显示了一个有可见父节点的节点后的结果。不可见的子孙节点变为可见。隐藏的子孙节点仍为隐藏（它的子孙节点仍然为不可见）。



场景 4：显示一个有不可见父节点的节点

下图显示了当用户显示一个有不可见父节点的节点后的结果。因为目标节点在一个不可见的子树中（因为它的某个祖先节点为隐藏节点），这个节点不会变为可见。相反，它会被标记为不可见并且当它的隐藏祖先节点被显示时变为可见。



2.3.11 对象关联

eZ Publish 的内容模型允许为不同对象创建关联。任何类型的对象都可以彼此关联。这个特性特别适合用于绑定和/或重用散落在系统内部的各种信息。

例如：关联对象的概念可以把图片添加到文章中。不同于使用固定数量的图片属性，图片可以报存在其它的对象中。这些对象可以被关联到文章并在“XML 块”类型的属性中直接以插图的形式使用。这种方案很灵活，因为它不限制关联对象的数量与类型。

关联类型

对象间的关联可以在对象级别，也可以在对象属性级别实现。系统用相同的数据库表保存对象间不同类型的关联关系。对象不能与自己关联。

对象级别的关联

在 3.8 之前的版本中，对象级别的关联是一般性的并且不支持分组。从 3.9 版本开始，对象级别的关联有三种类型：

- 通用
- XML 链接
- XML 嵌入

通用

当用户手动将一个对象添加到其它对象的关联对象列表中时（大部分情况下，这是通过对象编辑界面中的“关联对象”窗口来完成的），创建的是“通用”类型的关联。这种方法总是可以使用。

XML 链接

当将一个内部链接（指向另外一个节点或对象的连接）嵌入“XML 块”类型的属性内时，系统会自动创建一个“XML 链接”类型的对象关联。注意，当这个链接标签被删除时，系统会自动删除这个对象关联。

XML 嵌入

当一个"embed"标签被嵌入一个“XML 块”类型的属性中时，系统会自动创建一个“XML 嵌入”类型的对象关联，例如：将对象嵌入到正在编辑的对象。注意，这种类型的关联在"embed“标签被删除时候，会被自动删除。

属性级别的关联

当使用“对象关联（单数）”或“对象关联（复数）”数据类型的属性时，这种类型的关联会被自动创建。第一种数据类型只允许关联一个对象，第二种数据类型允许关联多个对象。对象关联没有分组。但是，通过使用不同的这两种数据类型的属性，可以创建出具有关联分组的自定义数据结构。

2.3.12 分区

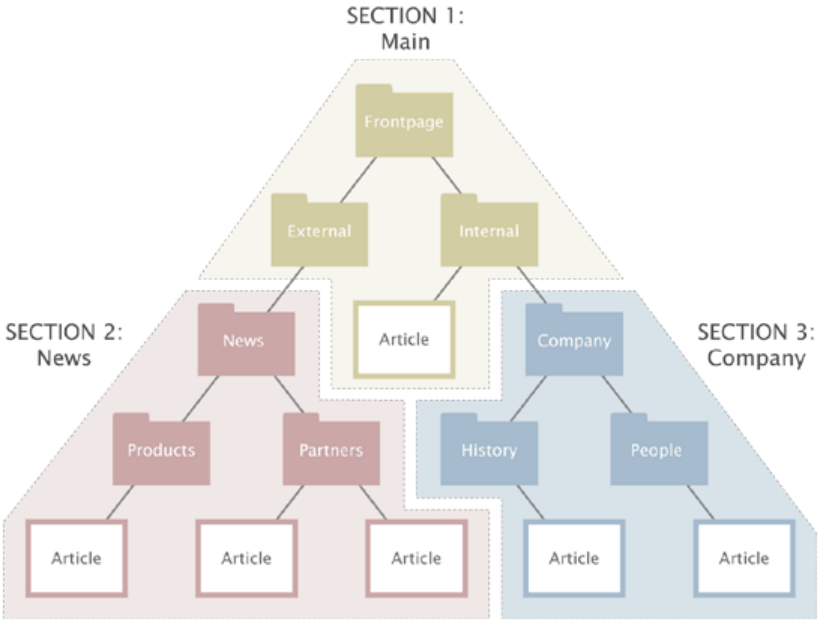
分区是可以指派给对象的一个数字。对象的分区 ID 表明这个对象属于哪个分区。每个对象可以属于一个分区。通过为不同对象指派不同的分区 ID，逻辑上可以把对象分成很多组。尽管分区机制在对象级别实现，它更多的时候是与节点树一起使用。因此在管理界面中，允许在节点级别管理分区。通过使用分区，可以实现：

- 逻辑上将节点树划分为不同的子树
- 设置自定义的模板重设规则
- 限定与控制对内容的访问
- 对一组商品指派折扣规则

eZ Publish 默认安装会创建以下分区：

ID	名称	描述
1	标准	“标准”分区是默认分区。“内容”顶极节点属于这个分区。
2	用户	“用户”分区服务于用户帐号和用户组。“用户”顶极节点属于这个分区。
3	媒体	“媒体”分区用于“媒体”顶极节点。
4	设置	“设置”分区用于“设置”顶极节点。

在管理界面中可以添加，修改和删除分区定义。下图演示了如何利用分区对内容分段。



行为

当一个新对象被创建，它的分区 ID 会被设置为默认分区（通常为标准分区）。当对象被发布时，它会继承父节点的分区。例如：如果在属于分区 13 的文件夹中创建对象，这个对象的分区 ID 也是 13。如果一个对象与多个节点关联，对象的分区 ID 为它主节点的父节点的分区 ID。此外，如果主节点的变化了，对象的分区 ID 也会变化。

可以利用管理界面通过节点树为对象指派分区。当分区被指派到节点，节点对应的对象的分区 ID 会改变。此外，这个节点的子孙节点的分区 ID 也会变化。例如，如果一个包含新闻文章的文件夹的分区 ID 被修改，那么这个文件夹下的文章的分区 ID 也会变化。

删除分区会破坏权限设置，模板输出和系统中的其它设置。换言之，除非某个分区完全没有用处，否则不要删除。当删除分区时，只有分区定义会被删除。其它对分区的引用仍然存在，因此系统会处于不稳定的状态。分区 ID 不会被循环使用。如果分区被删除，它的 ID 不会被其它分区重用。

2.3.13 URL 存储

每个在“XML 块”或“URL ”类型的属性中输入的地址都会在数据库的独立区域保存。这些属性的数据中只保存到 URL 表中记录的引用。这种特性允许调查和编辑已发布的 URL，而不需要与内容对象互动。URL 表中的地址可以用"linkcheck.php"脚本来检查。这个脚本可以通过 eZ Publish 内建的 cronjob 脚本来执行。这个脚本会通过逐个访问 URL 表中的地址来检查它们时候可用。如果目标服务器返回一个异常的反馈（404 页面丢失，500 内部服务器错误，403 拒绝访问等等）或没有反馈，URL 会被标记为非法 URL。用管理界面中的“URL 管理”功能可以很容易地过滤出非法的 URL 和使用非法 URL 的对象。URL 表中的记录由以下数据构成：

- ID
- 地址
- 创建时间
- 修改时间
- 最后检查
- 状态

每个 URL 有一个唯一的数字 ID。地址包含真正的链接。创建时间是包含这个 URL 的对象的发布时间。修改时间在每次通过管理界面中的 URL 管理功能修改 URL（不是当包含 URL 的对象被修改）时会被修改。每次 URL 被脚本检查后，最后检查字段会被修改。URL 的状态可以为“合法”和“非法”。默认情况下，所有的 URL 都是合法的。当脚本执行时，它会自动更新 URL 的状态。如果发现一个无效的 URL，它的状态会被标记为“非法”。如果一个已经存在的 URL 被保存，系统会重用现存的记录。

注意：

链接检查脚本必须可以通过 80 端口与外界通讯。换言之，火墙必须允许从 WEB 服务器出栈的 HTTP 数据包。

2.3.14 信息收集

信息收集特性允许在查看一个与信息收集器关联的节点时收集用户输入的数据。它常用于创建反馈表单，投票等功能。

一个对象可以用来收集数据，如果它至少有一个被标记为信息收集器的类属性。当对象被显示时，每个收集器属性会用对应数据类型的收集器模板显示。与直接显示内容相反，信息收集器模板提供了数据输入的接口。生成的输入接口依赖于属性的数据类型。下表列出了可用作信息收集器的数据类型。

数据类型	输入接口	输入验证
复选框	复选框。	否
电子邮件	单行文本。	是
选项	单选按钮或下拉框。	否
文本块	多行的纯文本块。	否
文本行	单行的纯文本块。	否

输入接口必须封装在 HTML 表单中，并通过名称为"ActionCollectInformation"的按钮提交

到"/content/action"（"content"模块的"action"视图）。提交的数据会被保存在数据库的特殊部分，与对象分别保存但是与对象保持某种联系。此外，如果对象收集了任何数据，收集的信息可以以电邮邮件的形式发送给特性的收件人。在管理界面中“设置”下的“收集的信息”中可以查看和删除收集到的数据。

2.4 配置

本章解释了 eZ Publish 中的配置模型。默认的配置文件后缀名为".ini"并且保存在"settings"目录。每个配置文件控制 eZ Publish 中的一个特定部分。例如："content.ini"文件控制内容引擎的行为，"webdav.ini"文件控制 WebDAV 子系统的行文，等等。最主要也是最重要的配置文件是"site.ini"。除了其它的作用，它告诉 eZ Publish 使用哪个数据库，界面等等。默认的配置文件包含所有的配置选项（设定为默认值）与对应的说明。这些文件应该之被用于参考目的。换言之，它们永远都不能被修改。“参考手册”中的“配置文件”章节中有各种配置文件的详细解释。

文件结构

eZ Publish 的配置文件被分割为多个块，每一块包含一套配置。下例演示了"site.ini"配置文件的一部分。

```
...
# This line contains a comment.
[DatabaseSettings]
Server=localhost
User=allman
Password=qwerty
Socket=disabled
SQLOutput=enabled

# This line contains another comment.
[ExtensionSettings]
ActiveExtensions[]=ezdhtml
ActiveExtensions[]=ezpaypal
...
```

上例演示了两个配置块："DatabaseSettings"和"ExtensionSettings"。每一块内部包含若干控制系统行为的配置。配置通常可以被设置为"enabled“/"disabled"，一个字符串或一个字符串数组。如果配置名以一对方括号结尾，意味着这个配置接受数组类型的值。在上例中，"ActiveExtensions"配置告诉 eZ Publish 启用两个扩展："ezdhtml"和"paypal"。以"#"开头的行为注释行。

配置重设

如前所述，默认的配置文件永远都不应该被修改因为它们很可能在日后的升级过程中被新的配置文件覆盖。即使对这些文件备份也不够，因为配置文件会随着时间变化。例如：前一个版本的配置文件不会包含最新增加的配置选项。由于这些问题，自定义的配置必须被放置在其它位置。全局的配置重设可以被放置在"settings/override"目录。这个目录中的配置文件会重设默认配置。这个目录中的文件必须使用以下

后缀名中的一个：

- .ini.append
- .ini.append.php

如果".ini.append"和".ini.append.php"后缀名都被使用，".ini.append.php"的优先级更高。处于安全考虑，".ini.append.php"应该优先被使用。特别是当 eZ Publish 运行在非虚拟主机服务器环境。".php"后缀名能强迫 WEB 服务器把配置文件作为 PHP 脚本处理。如果有人试图在浏览器中直接访问配置文件，服务器不会显示配置文件的内容。相反，服务器会把它作为 PHP 脚本处理。但是因为配置文件所有的内容都已被注释（参阅下面内容），因此服务器不会生成任何输出。这种方法让黑客试图从系统外部访问配置文件中的重要信息（如用户名，密码等等）变得更加困难。要达到这种目的，配置文件的内容必须被 PHP 风格的注释/*和*/封装。下例演示了一个重设配置文件（如："test.ini.append.php"）的格式：

```
<?php /* #?ini charset="iso-8859-1"?


# These are my example settings
[ExampleSettings]
ExampleSettingOne=enabled
ExampleSettingTwo=disabled
...


*/ ?>
```

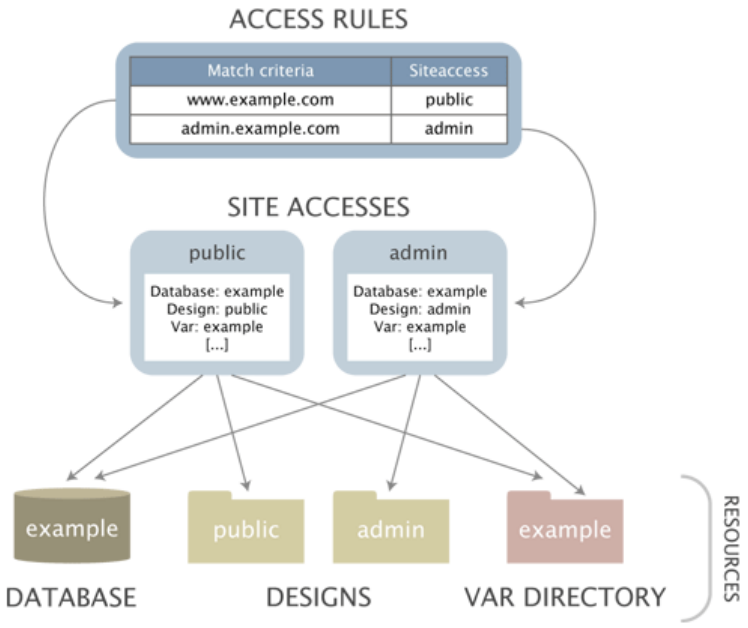
"charset"配置选项表明 ini 文件的字符集（通常为 ISO-8859-1）。

2.4.1 站点管理

eZ Publish 的一个安装可以支持多个站点，这需要使用称为“站点入口”系统。这个系统允许通过一系列规则管理不同配置。这些规则控制在不同情况下应该使用哪一组配置。站点入口规则必须在"site.ini"的全局重设文件（"settings/override/site.ini.append.php"）中定义。

站点入口

一套配置文件称为一个站点入口。当一个站点入口被使用时，默认的配置会被站点入口本身的配置重设。除其他配置内容外，站点入口定义需要用到的数据库，界面和 var 目录（这些有时也被称为“资源”）。通过使用不同的站点入口，可以把不同的内容和界面组合起来。典型的 eZ Publish 站点由两个站点入口：一个用于公共用户访问的公共站点和一个用于管理员访问的有安全限制的站点。两个站点入口使用相同内容（相同的数据库和 var 目录），但是他们使用不同的界面。管理站点入口使用系统内建的管理界面，公共站点入口使用一个自定义的界面。下图演示了这个场景。



当一个站点入口被使用时，它只是一套重设了默认配置的配置文件。一个 eZ Publish 安装理论上可以通过站点入口支持无限个站点。站点入口的配置文件被保存在"settings/siteaccess"中个一个子目录中。子目录的名称就是站点入口的名称。（注意，站点入口名称只能包含字符，数字和下划线）。下图演示了两个站点入口的配置：admin 和 public。

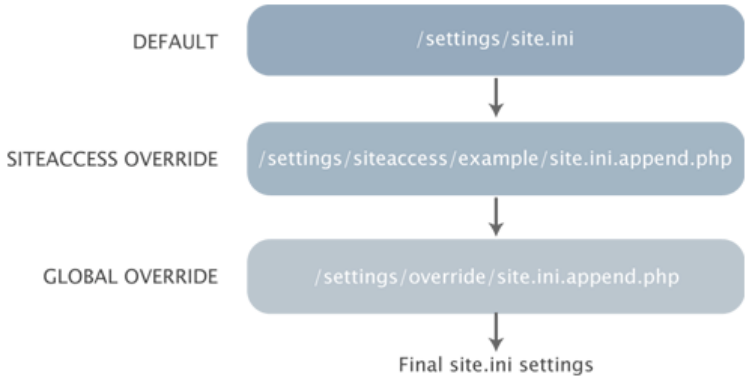
当一个站点入口被使用，eZ Publish 用以下顺序载入配置文件：

- 默认的配置文件(settings/*.ini)
- 站点入口配置(settings/siteaccess/[站点入口名称]/*.ini.append.php)
- 全局重设文件(settings/override/*.ini.append.php)

换言之，eZ Publish 首先会载入默认配置。然后，它会根据"site.ini."全局重设("settings/override/site.ini.append.php")中的配置来确定应该使用哪个站点入口。当它确定应该使用哪个站点入口，它会从这个站点入口的目录中载入这个站点入口的配置。站点入口的配置会重设全局重设，全局重设会重设默认设置。例如：如果站点入口使用"Amiga"数据库，系统将会看到这个配置并且在处理客户端请求是自动使用这个数据库。

换言之，eZ publish 会先读取默认的配置。然后，它会通过"site.ini"的全局重设（"settings/override/site.ini.append.php"）中的配置确定应该使用哪个站点入口。当它确定了应该使用哪个站点入口后，它会从站点入口中读取站点入口的配置文件。站点入口的配置会重设全局的配置。例如：

如果站点入口使用"Amiga"数据库，系统会了解这个配置并且在处理客户端请求时自动使用这个数据库。最后，eZ Publish 从全局重设目录中读取配置。全局重设配置会重设所有其它配置。换言之，如果全局重设配置中使用"CD32"数据库，eZ Publish 会尝试使用这个数据库而无论站点入口中使用的哪个数据库。如果一个配置既没有被站点入口重设也没有被全局重设，则默认配置会被使用。默认配置文件被保存在"settings"目录。下图演示了系统如何从"site.ini"文件中依次读取配置。如前所述，重设文件中的配置会重设默认设置。



2.4.2 扩展站点入口配置

扩展站点入口配置允许在扩展中放置站点入口的配置文件。

目录结构必须按如下规则组织：

```
extension/<my_extension>/settings/siteaccess/<my_siteaccess>/<file.ini.append.php>
```

示例：

```
extension/ezno/settings/siteaccess/ezno/override.ini.append.php :
```

```
<?php /*

[article_full_ezno]
Source=node/view/full.tpl
MatchFile=article/full.tpl
Match[class_identifier]=article
Subdir=templates

*/ ?>
```

注意：

除了 debug 配置和 include/activate 扩展配置，其余所有的配置都可以在扩展中配置。

2.4.3 访问方法

eZ Publish 在每次处理一个客户端的请求时，会根据一套规则决定应该使用哪个站点入口。这些规则必须在"settings/override/site.ini.append.php"中设置。站点入口系统的行为由[SiteAccessSettings]中的"MatchOrder"配置决定。这个配置控制 eZ Publish 如何解释客户端的请求。有三种可用的 MatchOrder：

- URI
- Host（主机）
- Port（端口）

以下的内容简单介绍了三种访问方法。注意，这些访问方法可以被组合使用。关于"MatchOrder"的文档说明了如何做到。

URI

这是"MatchOrder"的默认设置。当使用 URI 访问方法时，目标站点入口的名称为 URL 中"index.php"之后的第一个参数。例如：以下的 URL 会告诉 eZ Publish 使用"admin"站点入口：http://www.example.com/index.php/admin。如果还有一个"public"站点入口，可以用 http://www.example.com/index.php/public 访问。如果 URL 的最后一部分被省略，默认站点入口会被使用。默认站点入口在[SiteSettings]中的"DefaultAccess"配置。以下的示例演示了如何配置"settings/override/site.ini.append.php"来使用 URI 访问方法并使用"public"作为默认站点入口：

```
...
[SiteSettings]
DefaultAccess=public


[SiteAccessSettings]
MatchOrder=uri
...
```

URI 访问方法对于测试/演示很有用。此外，它非常容易因为不需要配置 WEB 服务器和 DNS。

Host

Host（主机）访问方法允许把主机/域名组合映射到不同的站点入口。这中方法需要 eZ Publish 以外的配置。首先，DNS 服务器必须被配置从而可以将需要的主机/域名组合解析到 WEB 服务器的 IP 地址。其次，WEB 服务器必须被配置以触发对应的虚拟主机配置（除非 eZ Publish 被部署在主要的 DocumentRoot）。请参阅安装章节中“虚拟主机配置”一章了解如何配置虚拟主机。DNS 服务器和 WEB 服务器正确配置之后，eZ Publish 可以被配置来通过主机/域名组合来确定使用的站点入口。下例演示了如何配置"settings/override/site.ini.append.php"来启用 Host 访问方法。此外，它也表明了 Host 匹配机制的基本用法。

```
...
[SiteAccessSettings]
```

```
MatchOrder=host

HostMatchType=map

HostMatchMapItems[]=www.example.com;public

HostMatchMapItems[]=admin.example.com;admin

...
```

上例告诉 eZ Publish 如果请求的 URL 以"www.example.com"开头，就使用"publich"站点入口。换言之，"settings/siteaccess/public"下的配置文件会被使用。如果请求 URL 以"admin.example.com"开头，则使用 admin 站点入口。上例只演示了 eZ Publish Host 匹配功能的一部分。请参阅参考文档了解全部关于 HostMatchType 的配置。

Port

Port 访问方法允许将不同端口映射到不同的站点入口。这种访问方法需要 eZ Publish 之外的配置。WEB 服务器比如被配置来监听在希望的端口（默认情况下，WEB 服务器监听在 80 端口，这是标准的 HTTP 端口）。此外，火墙也很可能需要被配置从而允许客户端请求可以通过 81 端口到达 WEB 服务器。下例演示了如何配置"settings/override/site.ini.append.php"来使用 Port 访问方法。它也演示了如何把不同的端口映射到不同的站点入口。

```
...

[SiteAccessSettings]

MatchOrder=port


[PortAccessSettings]

80=public

81=admin

...
```

上例告诉 eZ Publish 如果请求被发送到 80 端口，则使用"public"站点入口。换言之，"settings/siteaccess/public"下的配置文件会被使用。如果请求被发送到 81 端口（通常在 URL 后追加端口号，如：http://www.example.com:81），则 admin 站点入口会被使用。

2.5 模块与视图

模块提供了一种 HTTP 接口可以用于在 WEB 上与 eZ Publish 互动。某些模块提供了调用内核功能的接口，其余的模块在不同程度上独立于内核存在。系统内建的一套模块可以满足典型的日常事务的需求。例如：内容模块提供了通过 WEB 浏览器管理内容的接口。通过开发自定义的模块，可以扩展系统来满足特定的业务需求。自定义的模块必须用 PHP 开发。下表列出了一些常用的内建模块。

模块	描述
content	"content"模块提供了调用 eZ Publish 内核中内容引擎的接口。这个模块允许显示，编辑，检索和翻译内容对象，管理节点树等等。
user	"user"模块提供了调用内核中用户管理系统的接口。这个模块允许登录用户，登出用户。此外，它还提供了与用户注册，激活，修改密码等相关的功能。
role	"role"模块提供了调用内核中的访问控制系统的接口。此外，它还提供指派角色给不同用户和用户组的功能。

模块扩展

每次通过 WEB 浏览器访问 eZ Publish，客户端其实是在间接地与系统中的模个模块互动。请求的 URL 告诉 eZ Publish 应该执行哪个模块来处理客户端的请求。URL 的第一部分表明模块的名称。这通常是 URL 中"index.php"后面的第一部分（URI 访问方法除外）。下例演示了典型的 eZ Publish URL：

```
http://www.example.com/index.php/content/edit/13/03
```

这个 URL 表明请求会调用 content 模块。另外一个典型的 eZ Publish URL 如下：

```
http://www.example.com/index.php/user/login
```

这个 URL 表明 eZ Publish 会调用 user 模块。显然，这两个 URL 还包含附加的信息。在第一个例子中，模块名的后面追加了"/edit/13/03"。在第二个例子中，模块名的后面追加了"/login"。这些附加的信息控制被请求的模块的行为，将在下面解释。

模块视图

模块由一系列视图构成。视图可以被想象成为一种与模块的接口。通过使用视图，可以调用模块提供的各种功能。例如：除其它视图外，content 模块提供了用于显示，编辑，检索和翻译内容对象的视图。

视图的名称在 URL 中紧跟在模块名后面（由"/"分割）。在第一个例子中，eZ Publish 被要求访问 content 模块的"edit"视图。在第二个例子中，eZ Publish 被要求访问"user"模块的"login"视图。

当视图被调用时，eZ Publish 启动与那个视图相关的代码。执行结束后，视图把返回结果返回给模块，模块再把结果返回给系统的其余部分。结果被设置到一个模板变量中，变量名为 $module_result.content。在主模板（pagelayout）中可以访问这个变量。请参阅“模板”章节中的“模板生成”一章了解更多关于这部分的信息。

视图参数

某些视图支持一到多个参数。视图参数允许用户向视图传送信息从而通过 URL 来控制视图。视图参数在 URL 中被追加到视图名之后。在第一个例子中，以下的参数被传输给视图："13"和"03"。这些参数会要求 content 模块的 edit 视图提供一个界面用来编辑对象 13 的版本 3。第二个例子中的 URL 并没有使用视图参数。视图机制支持两种类型的参数：

- 有序参数
- 无序参数

有序参数在 URL 中必须用"/"分割且必须紧跟视图名之后。此外，他们的顺序必须与模块定义中的顺序相同。例如：如果第一个例子中的参数顺序对调，eZ Publish 会尝试去编辑对象 3 的版本 13（而不是对象 13 的版本 3）。

如同名字的含义，无序参数可以以任何参数提供。如果模块支持有序参数，无序参数必须排在有序参数之后。如果视图不支持有序参数，无序参数会跟随在视图名之后。无序参数必须成对出现。一个无序参数对有参数名和参数值构成（由"/"分割）。下例演示了如何在 eZ Publish URL 中使用无序参数：

```
http://www.example.com/index.php/video/dvd/button/play
```

上例中的地址告诉 eZ Publish 运行"video"模块并且执行"dvd"视图。无序参数的名称为"button"，参数值为"play"。如何处理这些参数由"dvd"视图的 PHP 代码决定。

POST 变量

某些视图会用到通过表单以 HTTP POST 方式提交到服务器的参数。例如："content"模块的"action"视图支持可扩展的 POST 变量。

GET 变量

视图也可以支持 GET 变量。例如：content 模块的 treemenu 视图的参数是通过 GET 变量传输的。

默认的请求

为了生成合适的输出，eZ Publish 必须知道应该运行哪个模块与执行哪个视图。换言之，每个 URL 都至少要包含模块名和视图名。如果输入了不完整的或错误的 URL，eZ Publish 会显示一个出错页面表明错误的内容（缺失/错误的模块或视图）。如果在 URL 中"index.php"后不包含任何内容（或许除了"/"），默认的模块/视图组合会被执行。默认的模块/视图组合可以在"site.ini"的重设文件中配置。配置的位置为："[SiteSettings]"下的"IndexPage"。默认的设置是"/content/view/full/2"。它要求 eZ Publish 显示节点 2（内容顶极节点）的全视图。换言之，如果请求以下 URL：

```
http://www.example.com/index.php
```

与请求如下 URL 的效果相同。

```
http://www.example.com/index.php/content/view/full/2
```

eZ Publish 不会重定向或是重载页面，这意味着浏览器的地址栏中的内容不会改变。

2.6 URL 翻译

本章解释了 eZ Publish 中可以使用的不同类型的 URL 以及 URL 翻译如何工作。默认情况下，eZ Publish 可以处理两种 URL：

- 系统 URL
- 虚拟 URL

系统 URL

系统 URL 告诉 eZ Publish 应该运行哪个模块并执行哪个视图。它可能包含传送给视图的附加的参数/值。每个系统 URL 都遵循相同的结构并且可以被分割为以下部分：

- 模块名
- 视图名
- 视图参数

视图参数是可选的且可以由有序参数和/或无序参数构成。在“模块和视图”章节有完整的介绍。以下的模型演示了不同 URL 部分的序列。

```
http://www.example.com/index.php/<module>/<view>/[<ordered_view_parameters>]/
[<unordered_view_parameters>]
```

URL 部分	描述
模块	应该运行的模块名称。
视图	应该执行的视图名称。
有序视图参数	某些视图接受有序参数。
无序视图参数	某些视图接受无序参数。

下例演示了典型的系统 URL：

```
http://www.example.com/index.php/content/edit/13/3
```

这个 URL 要求 eZ Publish 运行"content"模块并执行"edit"视图。"13"和"3"是将要传送给视图的参数。注意，确切的 URL 格式取决于“访问方法”和 WEB 服务器的配置方法。例如：WEB 服务器可以被配置以隐藏"index.php"。

虚拟 URL

虚拟 URL（也被称为 URL 别名或友好的 URL）其实只是现存的系统 URL 的别名。虚拟 URL 更友好，更容易记忆且有时比系统 URL 更短。系统 URL 揭示了很多有关 eZ Publish 被要求做何种处理的信息，虚拟 URL 却不显露任何系统级别的信息。虚拟 URL 不能如系统 URL 一样分解。下例演示了典型的虚拟 URL：

```
http://www.example.com/company/about
```

系统中其实有两种类型的虚拟 URL，一种是系统自动生成的，另外一种是由站点管理员手动创建和维护的。然而，所有的虚拟 URL 都会被等同处理。

从 3.10 版本，系统开始支持“多语言虚拟 URL　”。系统在一张由三个字段的数据库表中保管 URL：

Virtual address	Action	Language mask
company/about	eznode:46	2

使用上表中虚拟 URL 的真实地址可以如下：

```
http://www.example.com/company/about
```

根据上表，虚拟 URL 会被系统在内部翻译为如下 URL：

```
http://www.example.com/content/view/full/46
```

两个 URL 都是正确的且会生成相同的输出，在本例中为节点 46 的全视图。当使用虚拟 URL 时，重定向/映射会在系统内部进行，因此用户不会经历重定向或是页面重载等问题。Language mask（语言掩码）在系统内部使用，被用来标记 URL 别名所用的语言（与内容对象的语言掩码算法相同）。

如果站点管理员为"content/search"创建一条虚拟 URL，系统会在上表中添加如下记录：

Virtual address	Action	Language mask
findme	module:content/search	4

实际的虚拟 URL 为：

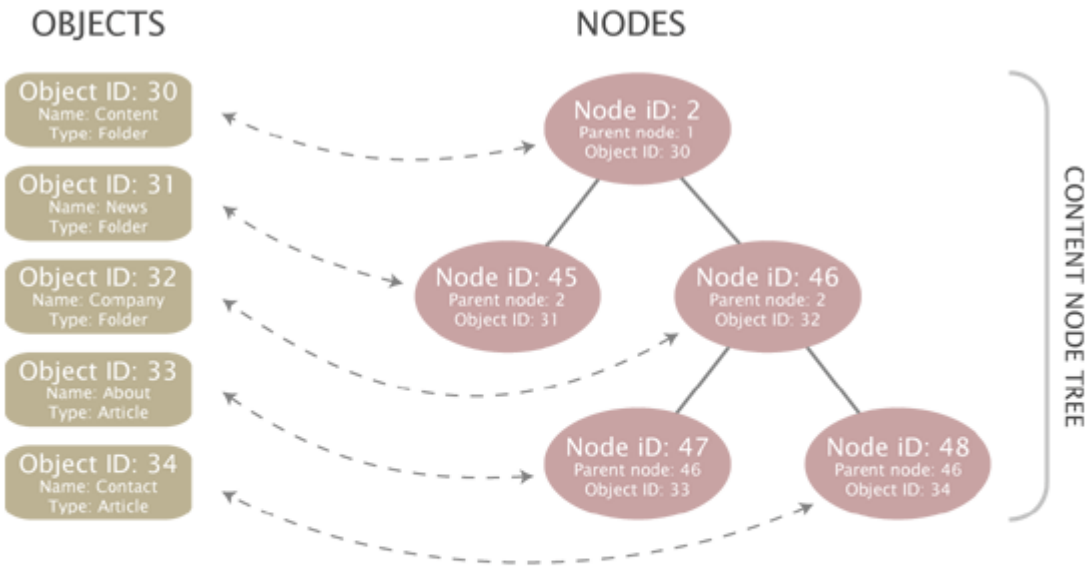
```
http://www.example.com/findme
```

根据上表，这个 URL 会在系统内部被翻译为：

```
http://www.example.com/content/search
```

自动虚拟 URL 的生成和维护

每次对象被发布，系统都会为对象的每个节点重新生成一个虚拟 URL。如果对象存在于多种语言中，系统会为每种翻译生成虚拟 URL。节点 URL 的生成是基于节点在节点树中的位置和节点对象的对象名。节点虚拟 URL 的生成完全由系统处理且不能在管理界面修改。下图演示了对象，节点以及对应的 URL。



Source/virtual address	System/destination address
news	content/view/full/45
company	content/view/full/46
company/about	content/view/full/47
company/contact	content/view/full/48

上图清楚地演示了虚拟 URL 是如何被生成的。对于每个节点，系统生成一个路径（由"/"分割）。路径中的字符串为节点祖先节点所对应对象的名称和目标节点本身对应对象的名称。

在 eZ Publish 3.9 版本之前，URL 变换规则更严格且之支持 ASCII 字符（小写拉丁字符"a"到"z"，数字和下划线）。特殊符号被转换为下划线且特殊字符由系统内建的字符翻译功能翻译。例如：挪威字符"æ"，"ø"和"å"会被翻译为"ae"，"oe"和"aa"。如果系统要生成一个已经存在的虚拟 URL，它会在新 URL 后简单地追加一个下划线从而消除重复 URL 的风险。

自 3.10 版本开始，可以在 URL 中支持 Unicode 且字符翻译不再需要因为大部分字符都允许出现在虚拟 URL 中。如果相同位置有两个节点的名字相同或几乎相同，系统会在新节点的 URL 后添加数字（如："company","company2","company3"等等）。

如果对象的名称改变了，系统会自动修改对应节点的虚拟 URL。此外，一条内部重定向记录会被创建。这条记录可以确保旧的 URL 仍然有效。旧的虚拟 URL 会一直有效，直到某个节点用到了相同的 URL。在这种情况下，重定向记录会被删除。

手动虚拟 URL

在管理界面中（全局别名与节点 URL 别名）可以手动添加，编辑与删除虚拟 URL。参阅“管理 URL 别名”了解更多。此外，也可以创建“基于通配符的 URL 转发”。（在实现多语言 URL 功能时，这个特性被从 3.10.0 版本中删除。不过又在随后的版本中重新加回了系统）

2.7 界面

本章解释了界面的概念以及 eZ Publish 如何处理不同的界面。如前所述，界面关于真实的内容如何被标记和显示。当谈论界面时，我们在讨论构成 WEB 用户界面的元素：HTML，式样表，不属于内容的图片等。所有与外观有关的文件都被放置在"design"目录中。一个 eZ Publish 安装理论上可以处理无限个界面。每个界面在 design 目录下有一个属于自己的子目录。子目录的名称也是界面的名称。典型的 eZ Publish 界面有以下内容构成：

- CSS 文件
- 图片文件
- 字体文件
- 模板文件

除了其它配置，一个站点入口会指定使用哪个界面。通过使用不同的站点入口，可以将内容与界面进行组合。典型的 eZ Publish 站点由两个站点入口构成：一个公共界面和一个受限的管理界面。两个站点入口使用相同的内容（数据库和 var 目录），但是却使用不同的界面。管理站点入口使用内建的管理界面。公共站点入口使用自定义界面。

默认界面

一个 eZ Publish 发行版本内建至少两个默认界面：

- admin
- standard

"admin"目录包含所有用于构成内建管理界面相关的文件。"standard"目录包含一套标准/默认的界面相关的文件如默认/标准模板，图片等。这些目录中的文件不应该被修改。相反，如有必要，可以创建自定义的界面。一个自定义的界面可以通过在"design"目录下创建一个新的子目录来实现。

界面目录结构

所有属于某个界面的文件都位于这个界面的子目录中。子目录的名称也是界面的名称。一个典型的 eZ Publish 界面目录包含以下子目录：

子目录	描述
fonts	用于"texttoimage"模板操作符的字体文件。"texttoimage"可以把文字转换为图片。
images	非内容图片（页眉，logo，图片布局元素等）。
override	替换默认/标准模板的自定义模板。这些文件会被模板重设规则触发。模板重设规则在"override.ini"的重设文件中配置。请参阅“模板”章节中“模板重设系统”一章了解更多。
stylesheets	CSS 文件。
templates	主模板（如：pagelayout，header，footer 等）与替换标准/默认模板的自定义模板。

2.7.1 界面组合

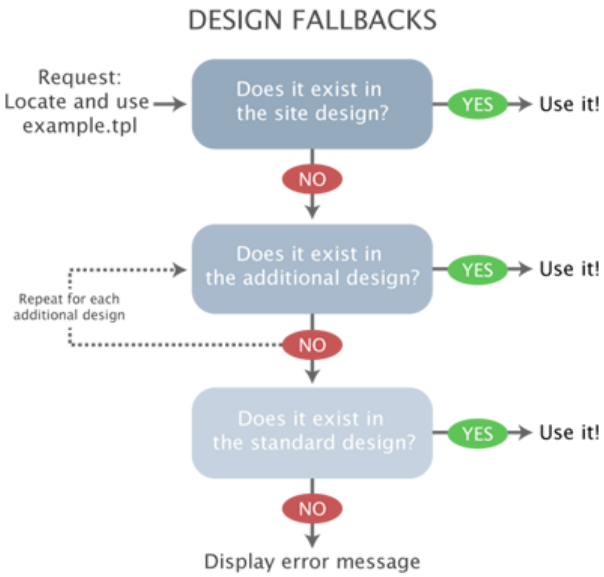
一个站点入口可以使用若干个界面。这意味着 eZ Publish 最终生成的结果可以是不同界面的组合。一个站点入口可以使用如下界面的组合：

- 一个主界面
- 0 到若干个附加界面
- 一个标准界面

一个站点入口总是至少有一个主界面和一个标准界面。主界面可以被任意修改，但是标准界面不能被修改。默认配置是使用系统内建的"standard"界面。这可以确保 eZ Publish 总能找到必要的模板，因而任何类型的内容都可以被显示。以下对此做深入讨论。

自动备选

如果 eZ Publish 在主界面中找不到某个文件（式样表，模板，图片等），它会自动尝试到其它位置寻找这个文件。系统会依次便利所有的附加界面（如果有），寻找这个文件。最后，如果仍然没有发现请求的文件，eZ Publish 会尝试在标准界面中寻找这个文件。下图演示了这种功能。



配置

对于不同界面的使用必须在"site.ini"的一个重设文件中配置。应该在"[DesignSettings]"中配置。以下的配置选项可以使用：

- SiteDesign
- AdditionalSiteDesignList
- StandardDesign

"SiteDesign"选项指定主界面。"AdditionalSiteDesignList"选项指定一组附加界面。"StandardDesign"指定标准界面。尽管可以修改标准备选界面，修改它并不明智。因此，"StandardDesign"应永远被设置为系统内建的"standard"界面。这已经在默认"site.ini"中设置，因此不需要在重设文件中修改它。如果您需要自定义的备选界面，您可以在"AdditionalSiteDesignList"中指定。自动备选机制提供了很多灵活性。例如：它简化了界面的重用与组合。

例子

下例演示了如何在"site.ini"的重设文件中配置以下界面：

- "my_design"为主界面
- "fallback_one"为第一个附加界面
- "fallback_two"为第二个附加界面
- "standard"为标准备选界面

```
...
[DesignSettings]
```

```
SiteDesign=my_design
AdditionalSiteDesignList[]=fallback_one
AdditionalSiteDesignList[]=fallback_two
StandardDesign=standard
...
```

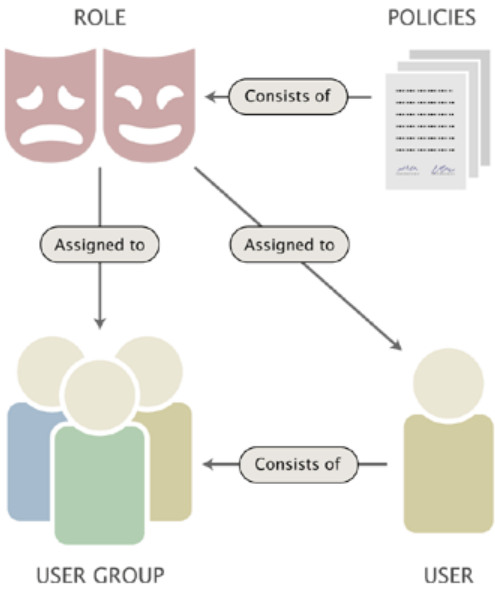
在这个特殊的例子中，如果 eZ Publish 在"my_design"中找不到请求的文件，它会自动退回到附加界面。首先，系统会在"fallback_one"界面目录中寻找。如果仍然找不到，系统会在"fallback_two"目录中寻找。如果仍然找不到，系统会尝试在"standard"界面目录中寻找。"standard"目录很可能包含请求的文件（除非请求的是一个自定义的模板/重设）。

2.8 访问控制

本章解释了 eZ Publish 如何管理用户帐号和访问权限。系统有一个内建的访问控制机制，可以用来限制对内容和功能的访问。访问控制系统基于以下元素：

- 用户
- 用户组
- 策略
- 角色

下图演示了这些元素的关系：



用户定义了一个系统内合法的用户帐号。用户组由用户和其他用户组构成。策略规则是一条规则，这条规则授权对内容和功能的访问。例如：一条策略可以授权对一组节点的读访问。角色是被命名的一组策

略。角色可以被指派到用户和用户组。以下内容深入探讨用户/用户组/策略/角色。

用户

一个真实的用户帐号由一个包含这个用户个人信息的内容对象（至少有一个节点）代表。默认的“用户”类允许存储后面这些元素：名，姓，电子邮件，用户名和密码。最后三个元素（电子邮件，用户名和密码）由“用户帐号”数据类型提供。这是一种特殊的数据类型，它会作为合法的用户存在于系统中。换言之，如果需要保存附加的用户数据，可以修改默认的用户类或创建一个自定义类且包含这种数据类型的属性。

启用和禁用用户帐号

在管理界面中可以启用或禁用用户帐号。当被禁用时，用户帐号仍然存在，但是用户不能登录系统，除非用户帐号被启用。新创建的用户帐号默认被启用。

锁定/解锁用户帐号

除了启用和禁用，用户帐号还可以被锁定和解锁。一个帐号会自动被系统锁定如果密码输入错误的次数超过最大登录失败数。失败的登录是一个正确的用户名和错误的密码的组合。帐号被锁定后，帐号对应的用户将不能登录，除非帐号被有管理员权限的其他用户解锁或者从可信 IP 地址段访问。

登录失败数保存在数据库中的"ezuservisit"表。登录成功后，登录失败数会被清零。

电子邮件

注意，默认配置不允许不同用户用相同电子邮件注册。这只是一个内部预警机制。您可以在"site.ini"的重设文件中将"RequireUniqueEmail"设置为 false。这个配置在[UserSettings]块内。

用户 ID

每个用户有一个唯一的数字 ID。这个 ID 与代表用户帐号的对象 ID 相同。用户 ID 被系统中的其他对象引用。一个对象包含到创建者（通过用户 ID）和所有版本创建者的引用。删除用户帐户会造成数据库不一致，因为对象的属主/修改者会引用到不存在的用户帐号。因为这个原因，不建议从系统中删除用户，删除用户的功能应该被从系统中禁用。

用户组

用户组是一个内容对象（有至少一个节点）。这个对象包含用户帐号和其他用户组。换言之，用户组只是用户的集合（与文件系统中的目录类似）。

策略

策略是一条规则。这条规则授权访问某个模块特定功能或全部功能。策略由以下元素构成：

- 模块名
- 函数名
- 函数限制

模块名表明了要授权的模块名。模块名指定要被限定的函数名。一个策略可以限定单个函数或授权访问一个模块的所有函数。一个模块可以有 0 到多个函数。函数被指派到模块的视图，因此对视图的访问控制由指派到这个视图的函数控制。函数-视图指派不能在管理界面中修改。对模块函数的授权可以用函数限制进一步控制。这只有在函数本身支持限制才可以。一个函数可以有 0 个，1 个到多个限制。下表演示了可用的函数限制。

限制	描述
类	“类”限制允许将策略限定到某些对象类型。
语言	“语言”限制允许将策略限定到特定节点。
节点	“节点”限定允许将策略限定到一个特定节点。
属主	“属主”限定允许将策略限定到属于当前登录用户的对象。
父节点类	“父节点类”限定允许基于父节点类型限制策略。
分区	“分区”限定允许将策略限定到特定分区内的对象。
站点入口	“站点入口”限定允许将策略限定到特定站点入口。
状态	“状态”限定允许将策略限定到特定的版本状态（发布，归档，等）。
子树	“子树”限定允许将策略限定到内容节点树的子树。

角色

角色是被命名的一组策略。角色可以被指派到用户和用户组。指派角色的时候可以有附加的限制。角色限定特性在多个具备类似权限的用户必须管理内容树中的不同部分时很有用。在这种情况下，不是为每个用户创建一个独立的角色，而是创建一个通用的角色然后为不同用户指派不同的角色限定。角色限定会重设策略限定。下表演示了可用的角色限定。

限制	描述
分区	“分区”限制允许将角色限定到某些对象类型。
子树	“子树”限制允许将角色限定到特定子树。

2.9 网络商店

本章解释了 eZ Publish 的电子商务特性。系统有一个内建的直接嵌入对象/节点树模型的商店机制。网络商店围绕着以下组件构建：

- 商品
- 增值税（VATs）
- 折扣规则
- 首选购物清单
- 购物篮
- 订单

下图演示了不同的组件如何协同运作。

一个商品由一个内容对象（至少有一个节点）代表。这个对象应该包含商品的信息和一个价格。价格必须是一个使用内建的“价格”或“多价格”数据类型的属性。它们是两种特殊的数据类型。区别在于“价格”数据类型只能对一个对象指定一种价格，而“多价格”数据类型可以对一个对象（多价格对象）指定不同货币的多个价格。一个内容类只能有一个价格属性或一个多价格属性。不能同时在购物篮中放置单价格商品和多价格商品，事实上不建议在同一个站点同时使用单价格和多价格。

价格会受到增值税和/或折扣规则的影响。折扣规则被配置用来以折扣率形式减少某些商品的价格。商品可以被添加到用户的优先购物清单和/或购物篮中。用户可以在任何时候修改自己的优先购物清单和购物篮。用户可以通过开始结帐流程来购买购物篮中的商品。结帐流程结束后，会有一个订单被创建。

系统会通过电子邮件自动通知站点管理员和这个买家。在管理界面可以查看订单和销售报表。订单的状态可以被有权限的用户修改。每个订单的状态变化会被系统记账。

增值税(VAT)

系统允许站点管理员设置不同类型的增值税。增值税类型由名称和税率构成。管理界面可以用来添加，删除和修改 VAT 类型。VAT 类型会被价格和多价格数据类型用到。您还可以配置系统按照买家的国家和商品的分类来收取不同金额的增值税。（请参阅“特性”章节中的“VAT 收费系统”了解更多）。

价格数据类型

如上所述，商品只是一种包含价格的内容对象。价格可以由一个使用内建价格数据类型的属性代表。任何包含价格的类的实体会被系统自动识别为简单价格商品。由价格数据类型代表的类属性使用某个预定义的 VAT。有两种使用 VAT 的方法。这个配置取决于创建对象时，价格信息是如何被输入的。第一种方法（含税价格），如果输入的价格已经包含了增值税。第二种方法（不含税价格），如果输入的价格不含增值税。当使用第一种方式时，如果查看商品，输入的价格会被直接显示。当使用第二种方法时，如果查看商品，显示的价格为输入价格加上增值税。当查看购物篮时，您可以查看含税和不含税价格（无论使用的哪中方法）。

注意：

价格数据类型只允许为每个商品指定一种价格（系统会用您的本地货币显示这个价格）。这个数据类型不支持多货币。

多价格数据类型

价格可以由一个使用系统内建的多价格数据类型的属性代表。这个数据类型允许您为每个商品设置多种货币的多种价格（与您的本地货币无关）。任何包含多价格数据类型的类的实体会被系统自动识别为多价格商品。（请参阅“特性”章节中的“多价格”章节了解更多。）这种数据类型与 VAT 的关系与价格数据类型一样。

折扣规则

商品的最终价格会受到折扣规则影响。折扣规则可以被配置来按照折扣率减少特定商品的价格。折扣规则可以被放置在不同折扣规则组内且一直有效（其实不能开启/关系折扣规则）。折扣规则组允许选择折扣规则应用于哪一组客户。这可以通过将某个折扣组指派到一个目标用户组来达到。

默认情况下，新创建的折扣规则对所有系统中的商品都有效。但是，您可以很容易地把折扣规则限定到一组商品。有两种方法限定折扣规则，它们是彼此排斥的。第一种方法是通过使用“商品类型”和“分区”的组合，如下表。

限制	描述
商品类型	“商品类型”允许限定策略到特定类型的商品/对象（只有使用价格数据类型的类会被显示）。默认设置为“任何”，意味着它会影响所有类型的商品对象。
分区	“分区”限定允许限定策略到特定分区内的商品/对象。默认设置为“任何”，意味着折扣规则会影响多有分区内的商品对象。

商店相关的数据类型

下表演示了嵌入 eZ Publish 电子商务子系统的数据类型。

数据类型	描述
价格	在内容类中使用一个属性时，“价格”数据类型会将这个类的实体（对象）与网络商店系统连接起来。只要对象有一个价格属性，用户就可以把它放入自己的购物篮和/和优先购物清单中。这中数据类型只允许对每个商品设置一个价格（系统会使用您本地的货币显示这个价格）。
多价格	在内容类中使用一个属性时，“多价格”数据类型也会将这个类的实体（对象）与网络商店系统连接起来。只要对象有一个多价格属性，用户就可以把它放入自己的购物篮和/和优先购物清单中。没有价格或多价格属性的对象不能被放入用户的购物篮和/或优先购物清单中。多价格数据类型允许您为每个商品设置不同货币的不同价格。

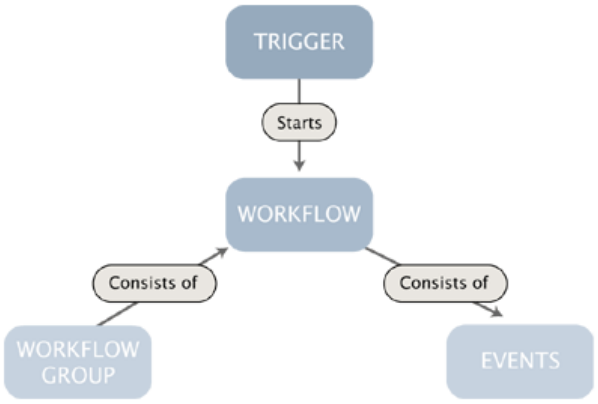
选项	选项数据类型允许为每个对象创建一组选项。每个选项可以被指派一个文本简介和一个附加价格。例如：它可以用来销售不同颜色的 T 恤衫，而某些（或全部）颜色的价格不同。
多选项	注意，这种数据类型不应再被使用。从 3.10 版本开始，这种数据类型已经不推荐使用。取而代之的是“多选项 2 ”数据类型（如下）。 “多选项”数据类型允许对每个对象创建多组选项。每个选项可以被指派一个文本简介和一个附加价格。这种数据类型与“选项”数据类型的工作方式一样。唯一的区别在于，它允许创建为每个对象创建多个选项组而不仅是一个选项组。
多选项 2	“多选项 2 ”数据类型允许为一个对象创建多个不同的多选项组。多选项可以嵌套。对每个选项，您可以指定一个附加的价格，一张图片，选项是否被默认选中以及是否可以选择（有时，您希望强制选中一些选项而不为用户提供一个默认选项；在这种情况下，第一个选项可以被设置为“请选择”）。此外，这种数据类型还允许设置允许/不允许特定的选项组合规则。
范围选项	“范围选项”数据类型允许为一个对象创建一组枚举选项。例如：它可以用于这样一个场景，目标是销售不同尺寸的鞋且尺寸不影响价格。对于每个对象，管理员需要设置可用的范围（如果有）。

2.10 工作流

本章解释了 eZ Publish 的工作流特性。系统内建了一种工作流机制，可以通过或不通过用户的参与来完成各种工作。工作流的实现基于以下组件：

- 事件
- 工作流
- 工作流组
- 触发器

下图演示了这些元素间的联系。



事件是工作流系统中的最小单位，它执行一个特定的任务。eZ Publish 内建了一套事件，可以满足典型日常工作的需要。例如：内建的审批时间允许对象在发布前由编辑（一个用户）对它的内容进行审查。内建的事件在“参考手册”中的“工作流事件”章节有详细的描述。可以通过开发用于特定用途的自定义的事件来扩展系统。自定义工作流事件必须用 PHP 开发。

工作流是一组事件。换言之，它定义了工作流运行时一个有序的动作序列。工作流可以被放置在不同组内。工作流组只是工作流的集合。工作流由触发器触发。尽管一个触发器只能触发一个工作流，内建的多路控制器事件（在由触发器最初启动的工作流内部）可以启动多个其它的工作流。触发器与模块的函数关联在一起。它会在模块函数开始或结束时启动指定的工作流。下表演示了标准/内建的触发器。

ID	模块	函数	连接类型
1	content	publish	before
2	content	publish	after
3	shop	confirmorder	before
4	shop	checkout	before
5	shop	checkout	after

3 模板

这一章的目的是讲授关于模板系统的一切基本知识。它描述了模板语言以及系统如何处理模板文件。之前不熟悉 eZ Publish 的用户应该可以得到足够的信息来帮助他们理解如下问题：

- 模板是什么又不是什么
- 模板类型（pagelayout，节点和系统模板）
- 模板结构
- 模板语言
- 主模板（pagelayout）
- pagelayout 中可以使用的模板变量
- 如何完成基本的模板任务
- 如何从 CMS 中提取信息
- 模板重设系统

3.1 模板基础

本章解释了模板系统和模板背后的概念。eZ Publish 把模板作为站点界面的基础单元。基本上模板是一个定制过的 HTML 文件，这个文件描述了特殊类型的内容如何被显示。模板文件总是以".tpl"作为后缀名。内建/默认模板中的 HTML 代码符合 XHTML1.0 Transitional 规范。除标准 HTML 语法外，模板还由 eZ Publish 特有的代码构成。eZ Publish 特有的代码允许模板从系统中提取信息并解决通用的编程问题如：条件分支，循环等。所有 eZ Publish 特有的代码必须被放置在一对大括号"{"和"}"中。下例演示了某个模板的一部分，用于显示当前系统时间。

```
...
<h1>Time machine</h1>
<p>
    The current time is: {currentdate()|l10n( time )}
</p>
...
```

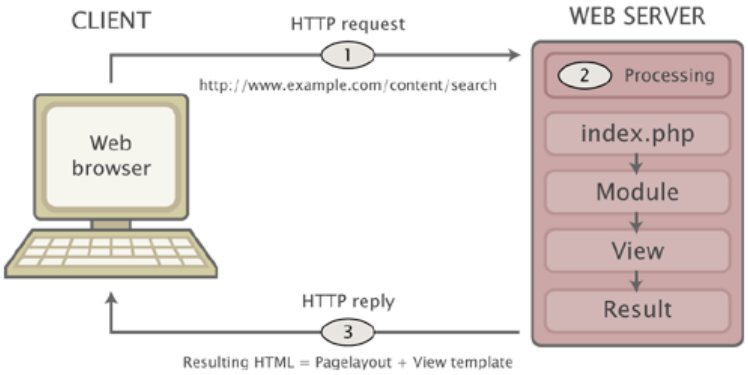
上列演示了标注 HTML 如何与 eZ Publish 特有的代码混合在一起。演示了"currentdata"和"l10n"模板操作符

的用法。因为"currentdata"返回一个 UNIX 时间戳，它必须经过"l10n"本地化操作符格式化（否则输出的数据无法被理解）。这是通过将"currentdata"的输出用管道操作符连接到"l10n"，后者会根据当前区域设置来输出结果。"Time"参数告诉操作符只输出时间（它可以是"date"，"shortdate"，"datetime"等等）。

模板生成

模板系统基于组件。换言之，一个真实的 HTML 页面通常由多个模板构成。至少，eZ Publish 总是生成主模板，称为 pagelayout。pagelayout 由 HTML，HEAD 和 BODY 标签构成；它决定了站点的全局版式。除了其他外特性外，它描述了系统所有 HTML 页面的全局视觉结构（主版式，logo，主菜单，页脚，等等）。

每个客户端的请求告诉 eZ Publish 运行特性的模块并执行模块的某个视图。当执行结束后，请求的模块/视图组合会生成一个结果。这个结果可以在 pagelayout 中通过$module_result 数组访问。下图演示了一个 3 步的流程来解释 eZ Publish 如何回应一个 HTTP 请求。



每个视图会用一个模板生成一段 HTML 代码。被视图使用的模板通常被称为“视图模板”。每当一个视图执行结束后，它会调用一个内部模板请求。被请求的模板会被解释，处理然后被转换为 HTML。处理之后，系统会把结果的 HTML 放入模块的结果数组中。模块/视图的结果可以用".content"来访问：。{$module_result.content}。通过显示这个变量的内容，可以在 pagelayout 中包含视图生成的 HTML 内容。下图演示了模块/视图结果（由不同的模块/视图生成－取决于不同的请求）如何被包含到 pagelayout 中。

视图模板

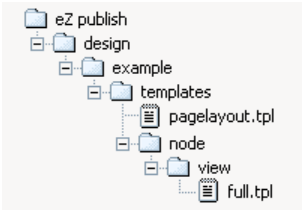
被视图使用的模板可以是一个节点模板或是系统模板。节点模板只有在一个节点被显示时，才会被用到，例如：当一个包含"conent/view"的系统 URL 或一个节点的虚拟 URL 被请求。系统模板通常用来提供调用 eZ Publish 某个特定特性的 HTML 接口。例如：被"content"模块的"search"视图使用的模板提供了调用系统内建的检索引擎的接口。

上述模板类型间的区别在于模板中可以使用的变量和可用的重设规则组合。在节点模板中有一个特殊变量($node)，这个变量包含了当前被显示的节点的信息。取决于被执行的视图，系统模板中通常会有一些可用变量。模板重设规则允许在特定条件下使用自定义模板。节点模板的重设规则比系统模板的规则要灵活很多。例如：可以设定复杂的重设规则组合来通过节点的类型，节点树的深度，节点对象所在的分区等条件来决定所用的节点模板。请参阅“模板重设系统”了解更多关于模板重设系统的信息。

3.1.1 节点模板

任何时候，当 eZ Publish 被要求输出某个节点的信息时（通过系统 URL 或虚拟 URL），它都会执行"content"模块的"view"视图。如果使用系统 URL，视图模式和目标节点必须在 URL 中指定。如果使用虚拟 URL，eZ Publish 会通过查阅内部的 URL 表来得知应该使用什么视图模式以及显示哪个节点。当使用虚拟 URL 时，系统总是会使用"full"视图模式。

服务于不同视图模式的模板必须被放置在某个界面的"templates/node/view/"目录中。如果在主界面中没找到所需模板，系统会在附加和标准界面中搜索。请参阅“自动备选系统”章节了解更多。"standard"界面中的"templates/node/view"目录包含了不同视图模式的模板。一个最基本的自定义界面通常会包含 pagelayout 和一个"full"视图模板。下图演示了一个称为"example"的自定义界面中上述文件的位置。



当节点被请求（且没有针对节点模板的模板重设规则）时，eZ Publish 将会生成一个由以下模板构成的页面。

自定义节点模板

典型的 eZ Publish 站点总是会用到自定义节点模板。主要的原因在于，用户总会希望用不同的方式显示不同的内容。例如：信息页会与新闻页有所不同；欢迎页面需要有特殊的格式等等。不同于自定义系统模板（基本上是复制一份系统模板到自定义界面，然后做适当调整），自定义节点模板被创建为重设模板。重设模板被模板重设系统触发。系统提供了灵活的机制来根据不同条件使用不同的模板。例如：它可以被配置为在显示一个文章类节点时，使用"article.tpl"，而在显示某个特殊的文章节点时，使用"spacical.tpl"。请参阅“模板重设系统”了解更多关于这种机制如何运作以及如何出发重设系统的信息。

$node 变量

任何时候当系统使用节点变量时（无论使用何种视图模式，目标节点是什么还是模板是重设模板与否），$node 变量在模板中总是可用。这个变量是系统自动设置的且它包含一个"ezcontentobjecttreenode"对象（此处的对象指的是 PHP 语言的对象，而不是内容对象）来代表当前显示的节点。从这个变量中可以提取并显示各种关于节点以及节点封装的内容对象的信息。请参阅“输出节点与对象数据”一章了解更多关于如何显示节点/对象数据的信息。

3.1.2 系统模板

任何时候，当 eZ Publish 被要求做除显示节点（换言之，URL 不包含"/content/view"或 URL 不是某个节点的虚拟 URL）以外的工作时，它会使用系统模板。系统模板与节点模板主要有两点不同：

- 系统模板提供各种变量（取决于执行的视图）。节点模板之提供$node 变量来代表当前显示的节点。
- 节点模板的重设系统比系统模板的重设系统要复杂得多。

eZ Publish 的发行版本提供所有视图的默认模板。这些模板位于"standard"界面下的"templates"目录中。一个视图使用的模板通常位于一个与其模块同名子的目录下。这个模板的名称通常与视图同名（加上.tpl 后缀）。例如："user"模块的"login"视图会在"user"子目录中寻找名为"login.tpl"的模板文件。另外一个例子

是"shop"模块的"basket"视图。这个视图会在"shop"子目录中寻找"basket.tpl"这个模板文件。

自定义系统模板

尽管 eZ Publish 提供了所有必要的系统模板（通过"standard"界面），一个典型的 eZ Publish 站点总是需要自定义的系统模板。主要原因在于，默认系统模板通常需要被定制以完美地匹配全局的外观。不同于节点模板大多数情况下通过模板重设系统提供，自定义系统模板通常只是通过修改一份系统模板在自定义界面中的复件来完成。例如：在"example"界面中针对"user"模块的"login"视图的自定义模板可能为"design/example/templates/user/login.tpl"。针对"content"模块的"search"视图的自定义模板可能为"design/example/templates/content/search.tpl"

界面组合

如前所述，自定义界面通常包含一套自定义的系统模板。但是，创建一个提供所有可能模板的自定义界面需要太多/不必要的工作。因此"standard"界面总是应该被用作最后一个备选界面。“自动备选系统”允许将多个界面组合使用，从而主界面（通常为自定义界面）不需要提供所有必须的模板。如果 eZ Publish 在主界面中找不到模板，它就会自动在附加界面和标准界面中寻找。

通用的系统模板

下表列出了一些最常用的通用系统模板。

请求	URL	模块	视图	模板
检索界面	/content/search	content	search	templates/content/search.tpl
购物篮	/shop/basket	shop	basket	templates/shop/basket.tpl
登录页面	/user/login	user	login	templates/user/login.tpl
用户注册	/user/register	user	register	templates/user/register.tpl

3.2 Pagelayout

pagelayout 是主模板。除了其他以外内容外，它决定了站点的全局外观。pagelayout 模板必须被命名为"pagelayout.tpl"。它必须被放置在某个界面的"templates"目录下。如果 eZ Publish 在当前界面（有站点入口指定）中找不到这个模板，它会使用备选界面中的第一个匹配的 pagelayout 模板。下图演示了"example"界面中的 pagelayout 模板的位置。

Pagelayout 包含 HTML，HEAD 和 BODY 标签（外围 HTML 框架）。此外，它还决定了站点的全局外观。出了其他用途外，它被用来描述所有页面的视觉结构（主版式，logo，主菜单，页脚等）。下例演示了一个最基本的 pagelayout。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

```
<head>

<style type="text/css">
    @import url({'stylesheets/core.css'|ezdesign});
    @import url({'stylesheets/debug.css'|ezdesign});
</style>

{include uri='design:page_head.tpl'}

</head>

<body>

{$module_result.content}

<!--DEBUG_REPORT-->

</body>
</html>
```

文档类型

pagelayout 的第一行用来定义页面的文档类型。根据 HTML 和 XHTML 标准，DOCTYPE（"document type declaration"的缩写）用来告知浏览器与语法验证引擎所使用的(X)HTML 文档类型。这个信息必须被放置在每个 WEB 页面的第一行，因此 pagelayout 的第一行必须是 DOCTYPE。

为了确保页面被正确渲染与页面的兼容性，DOCTYPE 定义是必不可少的一个关键元素。一个 DOCTYPE 包含一个完整的 URL 字段，这个 URL 告知浏览器用遵循标准的模式渲染页面，根据标准来对待(X)HTML，CSS 和 DOM 结构。缺失的，不完整的或过期的 DOCTYPE 会导致主流的浏览器运行于所谓的“扭曲模式”。在这种模式下，浏览器假定文档是基于老式的，不规范的标记和代码以及 20 世纪后期混乱的业界规范来书写的。换言之，页面很可能不会根据现有的工业标准来渲染，因而当然也不规范。

html 标签

HTML 标签封装了真实 WEB 页面的标记内容。处标签本身外，上列中的 HTML 标签还包含了一个到 XHTML 规范的 URL。XHTML 是一个现存和未来文档的类型和模块的家族。它可以被用来重新实现，简化和扩展 HTML4。XHTML 家族基于 XML，这意味着他们可以与基于 XML 的用户代理联动。

在处理文档时，确定内容所使用的自然或官方语言通常会比较有用。"lang"和"xml:lang"属性用来指定整个 HTML 元素的语言。xml:lang 的属性值的优先级更高。语言值应该被设置为整个站点使用的语言。语言代码在"ISO 3166-1"（以及与之相关的 ISO 3166-1-alpha-2）标准中定义。

head 标签

head 标签包含文档本身的信息。这里的信息不会在浏览器中显示。只有 title 标签的内容会在浏览器窗口的标题栏内显示。head 标签通常包含关于诸如：使用哪些 CSS 文件，文档的简介，关键字之类的信息。

式样表

上列中的 pagelayout 用到两个 CSS 文件："core.css"和"debug.css"。大括号封装的代码为 eZ Publish 特有的代码。引号里的文本通过管道操作符传送给"ezdesign"模板操作符。这个操作符将当前界面（在"SiteDesign"中指定）目录的路径添加到文本之前。这个技术确保指向 CSS 文件的路径总是正确的，与所使用的访问方法无关。例如：如果当前的界面为"my_design"并且它包含一个 CSS 文件"example.css"，系统会生成以下的输出。

```
@import url("/design/my_design/stylesheets/example.css");
```

"core.css"和"debug.css"属于"standard"界面。不需要在自定义界面中包含这些文件。如果 eZ Publish 在当前/自定义界面中找不到指定的文件，它会自动使用"standard"界面中的文件。请参阅“自动备选系统”了解更多信息。因为备选系统，上例中关于 CSS 部分的代码很可能会被生成如下输出：

```
...
<style type="text/css">
    @import url("/design/standard/stylesheets/core.css");
    @import url("/design/standard/stylesheets/debug.css");
</style>
...
```

core 式样表

"core.css"定义了一套用于一般 HTML 元素与 eZ Publish 特有的 CSS 类的标准风格（字体风格，尺寸，边距，等）。eZ Publish 特有的 CSS 类在标准模板中使用。一个站点如果继承使用默认的模板，就应该有"core.css"。否则，缺失的风格定义会导致很多元素无法正确渲染。

永远不要修改标准的"core.css"。如果"core.css"中的某些风格不符合需求，可以把一个修改过的"core.css"放置在自定义界面中。当时更好的做法是创建一个新的 CSS 文件来重设"core.css"中那些不符合需求的风格。

debug 式样表

"debug.css"包含用于格式化调试信息的风格定义。当调试输出被启用时，调试信息显示在页面底端。"core.css"只在开发过程中需要（特别是需要显示调试信息时），因此它可以在网站上线后被删除。

文档信息

系统可以自动生成页面本身的一些信息（标题，meta 标签，关键字等）。这可以通过包含页头模板("page_head.tpl")来完成。这个模板位于"standard"界面的"templates"目录中。如果 eZ Publish 在当前/自定义界面中找不到这个模板，它会自动使用"standard"界面中的文件。

body 标签

body 标签以有序的方式定义了文档的主体。文档的主体包含 WEB 页面真正内容（文本，图片，等）的标记。eZ Publish 的 pagelayout 至少应该包含来自所请求模块的结果。

模块结果

对于每个客户端请求，eZ Publish 自动生成一个称为"module_result"的数组。这个数组只在 pagelayout 模板中可用。它包含了所有关于运行的模块，执行的视图，输出的内容等方面的信息。实际的输出（例如：一篇新闻文章的内容）可以通过访问$module_result 数组的"content"元素在 pagelayout 中得到显示。语法如下：

```
{$module_result.content}
```

当生成 pagelayout 时，{$module_result.content}会被替换为真正的输出结果。请参阅“pagelayout 中的变量”一章了解更多。

调试信息

典型 pagelayout 模板的最后一部分是一段 HTML 注释代码：

```
<!--DEBUG_REPORT-->
```

如果调试信息被启用，在处理 pagelayout 时，eZ Publish 会把这段注释代码替换为实际的调试信息。换言之，调试信息会被作为内容的一部分包含到生成的页面中，因此不会破坏 HTML 结构，也不会造成页面的不规范。eZ Publish 的调试输出符合 XHTML1.0 Transitional 规范，因而调试输出也是规范的。

3.2.1 页头信息

标准界面中包含一个页头模板，它可以用来自动生成重要的页头标签（包含在每个页面的 head 标签内）。标准页头模板(design/standard/templates/page_head.tpl)的输出包括以下标签：

- title 标签
- meta 标签
- link 标签

下例演示了标准页头模板的输出：

```
<title>Current / Parent / Top - Site name</title>


<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<meta http-equiv="Content-language" content="eng-GB" />
<meta name="author" content="eZ Systems" />
```

```
<meta name="copyright" content="eZ Systems" />
<meta name="description" content="Content Management System" />
<meta name="keywords" content="cms, publish, e-commerce, content management, development
framework" />
<meta name="MSSmartTagsPreventParsing" content="TRUE" />
<meta name="generator" content="eZ publish" />


<link rel="Home" href="/" title="Front page" />
<link rel="Index" href="/" />
<link rel="Top"  href="/" title="Current / Parent / Top - Site name" />
<link rel="Search" href="/content/advancedsearch" title="Search Site name" />
<link rel="Shortcut icon" href="/design/standard/images/favicon.ico" type="image/x-icon" />
<link rel="Copyright" href="/ezinfo/copyright" />
<link rel="Author" href="/ezinfo/about" />
<link rel="Alternate" href="/layout/set/print/content/view/full/64" media="print"
title="Printable version" />
```

title 标签

title 标签的内容取决于当前显示的位置（内容节点树或系统本身的位置）与站点的名称。当前显示的元素的路径被倒序排列，因此当前元素成为标题的第一部分。路径的各个部分由"/"分割。当某个节点被显示时，路径实际上是当前节点所有祖先节点的名称再追加上当前节点的名称。当某个系统功能被调用时（例如：user 模块的 login 视图："/user/login"），路径很可能是"模块/视图"的倒序排列。站点名称被追加到路径的最后，由"-"分割。站点名称可以在"site.ini"的重设文件中的"SiteName"配置选项中指定。

上例演示了当显示一个节点时页头模板的输出。当前节点封装的对象名为"Current"。其他对象的名称为"Parent"和"Top"。站点名称为"Site name"。

meta 标签

除 WEB 页面的真实数据外，页面的 HTML 还应包含文档本身的信息。这是通过使用 meta 标签来达到的。由 meta 标签给出的信息通常不会在浏览器中显示。然而，这些 meta 标签的内容会被浏览器和检索引擎用来对 WEB 页面的内容建立索引与排名。标准的页头模板输出最常用的 meta 标签。它包括以下三类标签：

- HTTP-EQUIV meta 标签
- 一般 meta 标签
- 附加 meta 标签

HTTP – EQUIV meta 标签

具有 HTTP-EQUIV 属性的 meta 标签与 HTTP 信息头的作用相同。这些标签通常被用来控制浏览器如何解

释文档。这种形式的标签与使用 HTTP 信息头的作用相同。某些 WEB 服务器会自动将这些标签的内容翻译为真正的 HTTP 信息头。页头模板中的 HTTP-EQUIV meta 标签确保浏览器（与检索引擎）明了文档使用的字符集和语言。语言和字符集是由 eZ Publish 根据站点的语言和字符集自动生成的。

一般 meta 标签

一般 meta 标签可以揭示文档本身的信息。尽管规范并没有规定哪些 meta 标签必须被包括，包含例如：作者，站点简介，版权声明，关键字等信息确是一种惯例。通过使用"site.ini"重设文件中的"MetaDataArray[...]"，站点管理员可以设定一套自定义的一般 meta 标签。eZ Publish 将会循环显示这些 meta 标签的名称和值。上例演示了没有自定义 meta 标签时候，系统默认的输出。

附加 meta 标签

最后一组被标准页头模板设置的 meta 标签禁用"智能标签"并揭示了生成输出的软件的名称。

link 标签

head 标签内的 link 标签允许可以将文档与其他文档关联起来。这是通过 rel 和 rev 属性来完成的。"rel"链接用来建立关联，"rev"链接用来建立反向关联。某些浏览器用这些标签来生成快速导航栏。link 标签包含在标准界面的"templates"目录下的"links.tpl"中。标准页头模板使用"links.tpl"模板。标准页头模板的默认输出包含了一组基本的 link 标签，可以用来导航到站点的不同部分。下例演示了页头模板生成的标签：

链接	描述
Home	"Home"链接指向站点的根/起始点。它总会将用户带回首页（如：http://www.example.com）。
Index	"Index"链接指向站点的根/起始点。它总会将用户带回首页（如：http://www.example.com）。
Top	"Index"链接指向站点的根/起始点。它总会将用户带回首页（如：http://www.example.com）。
Search	"Search"链接指向"content"模块的"advancedsearch"视图。它会将用户带入高级检索页面。(http://www.example.com/content/advancedsearch)
Shortcut icon	"Shortcut icon"定义了网站收藏夹/快捷方式图标的位置。大部分浏览器会在地址栏的开头和收藏夹的记录前显示这个图标。默认的图标为两个橙色方块的 eZ Systems logo。这可以通过在自定义界面的"images"目录中放置一个名为"favicon.ico"的 16x16 像素的图标文件（16 色 BMP/Windows 图标格式）来替换。
Copyright	"Copyright"链接指向"ezinfo"模块的"copyright"视图。eZ Publish 的默认的版权页面会被显示(http://www.example.com/ezinfo/copyright)
Author	"Author"链接指向"ezinfo"模块的"about"视图。eZ Publish 的默认"关于"页面会被显示(http://www.example.com/ezinfo/about)。
Alternative	"Alternative"链接指向页面的替换的/打印机友好的版本。打印机友好的版本是通过使用"layout"模块的"set"视图来达到的。这种技术允许使用其他的 pagelayout（通常会去除除实际内容外的所有内容：菜单，logo 等）来显示内容。

link 参数

当引入页头模板的时候，可以通过传送参数"enable_link=false()"来禁用这些 link 标签。

```
{include uri='design:page_head.tpl' enable_link=false()}
```

指向替换/打印机友好版本的 link 标签可以在导入页头模板时通过传送参数"enable_print=false()"来达到。

```
{include uri='design:page_head.tpl' enable_print=false()}
```

3.2.2 Pagelayout 变量

pagelayout 模板中有各种可以用来显示系统状态和/或用来控制输出。下表列出了所有可用的变量与描述。

变量	类型	描述
$access_type	数组	站点入口的名称("name")和访问方法的 ID("type")（1=URL,2=Host,3=Port）。
$anonymous_user_id	整数	匿名用户帐号的对象 ID（默认/标准值为 10）。
$current_user	对象	当前登录用户对应的"ezuser"对象。如果没有用户登录，则为匿名用户对应的"ezuser"对象。
$ezinfo	数组	包含三种字符串的数组："version"，"version_alias"和"revision"。这些字符串揭示了 eZ Publish 发行版本的基本信息。
$module_result	数组	包含由模块/视图生成的结果的信息（以及结果本身）。
$navigation_part	数组	包含当前导航部分的名称与标识符的哈西表（键值为"name"和"identifier"）。例如："Content structure"和"ezcontentnavigationpart"。管理界面通过导航部分来确定用户与哪部分系统交互。
$requested_uri_string	字符串	包含请求 URL 中站点特有的部分，例如："content/view/full/44"（系统 URL）或"company/about'（虚拟 URL）。
$site	数组	包含关于站点入口的各种信息（站点名称，界面资源，meta 标签等）。
$uri_component	字符串	显示当前页面时，eZ Publish 使用的用户界面组件。这个变量被管理界面使用。
$ui_context	字符串	显示当前页面时，eZ Publish 所在的用户界面上下文。管理界面用这个变量来区别不同的工作模式（例如：“导航”，“编辑”，“浏览”等）。
$uri_string	字符串	请求 URL 的系统版本（例如："/content/view/full/13"）。
$warning_list	数组	包含与页面被生成时碰到的问题有关的警告信息。

$module_result

$module_result 数组包含被执行的模块/视图组合生成的结果。如果 eZ Publish 被要求显示某个节点的内容，这个变量将会包含这个节点的附加信息。如果 eZ Publish 被要求做其他任务（除了显示节点内容以外的任务），结果不会包含附加信息。下表列出了不同场景下$module_result 的内容。

默认$module_result

元素	类型	描述
content	字符串	被请求的视图（模板的结果）所生成的实际内容。
path	数组	哈西数组。包含了当前视图的路径信息。每个哈西包含后面这些键值："text","url"。"text"元素通常为模块/视图名（例如："Collected information"）。"url"元素包含地址。数组中最后一个元素的"url"键值通常为 false。 标准页头模板使用 path 数组来构造 title 标签。此外，path 数组可以例如被用来构造当前页面的脚印导航(breadcrumbs)（当前页面的全路径与链接）。
is_default_navigation_part	布尔	如果默认的导航部分被用到（在 PHP 代码中设置），则返回 TRUE。如果当前模块/视图的导航部分被站点管理员重新配置过，则返回 FALSE。可以在"module.ini"重设文件中的"[ModuleSettings]"修改导航部分。
navigation_part	字符串	当前导航部分的标识符（例如："ezcontentnavigationpart"）。管理界面用这个变量确定用户与哪部分系统互动。

ui_context	字符串	当前页面被显示时，eZ Publish 所在的用户界面上下文。管理界面用这个变量来区别不同的工作模式（“导航”，“编辑”，“浏览”等等）。
ui_component	字符串	eZ Publish 当前使用的用户界面组件。管理界面使用这个变量。
uri	字符串	包含 URL 中站点特有的部分，例如："/content/view/full/44"（系统 URL）或"company/about"（虚拟 URL）。

显示节点时的$module_result

元素	类型	描述
content	字符串	被请求的视图（模板的结果）所生成的实际内容。
view_parameters	数组	包含发送给视图的参数的数组（例如："limit"，"offset"等）。
path	数组	一个哈希数组。每个哈希包含到当前节点路径中的某个节点的路径信息。每个哈希包含以下元素： 键值 / 描述 text：节点引用的对象名。 url：节点的系统 URL（例如："/content/view/full/44"）。 url_alias：节点的虚拟 URL（例如："company/about_us"）。 node_id：节点的 ID 当前显示的节点会把它的"url"和"url_alias"设置为 false。此外，"node_id"不可用。path 数组可以被用来如构造脚印导航(breadcrumbs)。
title_path	数组	与"path"数组几乎相同。显示某个节点时，标准页头模板用"title_path"构造 title 标签。
section_id	字符串	节点对应对象所在的分区 ID
node_id	字符串	节点的 ID
navigation_part	字符串	当前导航部分的标识符（例如："ezcontentnavigatonpart"）。管理界面用这个变量确定用户与哪部分系统交互。
content_info	数组	包含当前节点的各种信息： 变量 / 类型 / 描述 node_id：字符串：节点 ID。 parent_node_id：字符串：父节点 ID。 object_id：字符串：节点引用的对象 ID。 class_id：字符串：节点引用的对象的类 ID。 class_identifier：字符串：节点引用的对象的类的标识符。 offset：整数：offset 视图参数 viewmode：字符串：显示当前节点的视图模式（例如："full"，"line"，等）。 node_depth：字符串：节点在节点树中的深度。 url_alias：字符串：节点的虚拟 URL（例如："company/about_us"）。

		persistent_variable	未知	在视图的某个模板中设置的变量。不论是否使用缓存，这个变量会一直在 pagelayout 中可用。这个变量的类型取决于它包含的内容。如果这个变量没有被设置，则返回 FALSE。
		class_group	数组	当前类（当前节点对应的对象对应的类）所属类组的 ID。这个变量与系统的一个特性有关，这种特性允许根据类组来重设模板。 默认情况下，"class_group"总是返回 FALSE，因为类组重设选项被禁用。可以在"content.ini"的重设文件中启用它（在"[ContentOverrideSettings]"中将"EnableClassGroupOverride"设置为"true"）。
cache_ttl	整数	模块/视图结果的 TTL(Time To Live)值（以秒为单位）。负的 TTL 意味着视图缓存永不过期。TTL 为 0 意味着结果不应该被缓存。		
is_default_navigation_part	布尔	如果默认的导航部分被用到（在 PHP 代码中设置），则返回 TRUE。如果当前模块/视图的导航部分被站点管理员重新配置过，则返回 FALSE。可以在"module.ini"重设文件中的"[ModuleSettings]"修改导航部分。		
ui_context	字符串	当前页面被显示时，eZ Publish 所在的用户界面上下文。管理界面用这个变量来区别不同的工作模式（“导航”，“编辑”，“浏览”等等）。		
ui_component	字符串	eZ Publish 当前使用的用户界面组件。管理界面使用这个变量。		
uri	字符串	包含 URL 中站点特有的部分，例如："/content/view/full/44"（系统 URL）或"company/about"（虚拟 URL）。		

3.3 模板语言

eZ Publish 模板语言允许从系统内部提取信息并可用来解决通用的程序问题如：条件分支，循环等。所有 eZ Publish 特有的代码都需要被放置在一对大括号"{"和"}"内部。eZ Publish 模板是 HTML 与 eZ Publish 模板代码的组合。所有在大括号内部的代码都会被 eZ Publish 解析。大括号外部的内容会被 eZ Publish 忽略，因而会被直接发送给浏览器。

大括号问题

因为大括号是 eZ Publish 的保留符号，它被用来定义代码块，这些符号不能在代码中直接使用。例如：Javascript 代码不能被直接嵌入代码因为 Javascript 会用到大括号。所有非模板特有的代码/文本必须为放置在"literal"标签内部。"literal"标签内的内容会被 eZ Publish 模板解释器忽略。下例演示了 literal 标签的用法：

```
...
{literal}
<script language="JavaScript" type="text/javascript">
<!--
    window.onload=function()
    {
```

```
        document.getElementById( 'sectionName' ).select();
        document.getElementById( 'sectionName' ).focus();
    }
-->
</script>
{/literal}
...
```

输出大括号

可以用两个模板函数："ldelim"和"rdelim"（left delimiter 和 right delimiter 的缩写）来输出大括号。下例演示了这些函数的用法：

```
...
This is the left curly bracket: {ldelim} <br />
This is the right curly bracket: {rdelim} <br />
...
```

输出：

```
This is the left curly bracket: {
This is the right curly bracket: }
```

3.3.1 注释

eZ Publish 模板的注释由"{*"和"*}"封装。注释标签不能嵌套。

单行注释

```
{* This is a single line comment. *}
```

多行注释

```
{* This is a long comment that
    spans across several lines
    within the template file. *}
```

嵌套注释（非法）

```
{* {* Nested comments are not supported! *}
This text will be displayed. *}
```

输出：

```
This text will be displayed.
```

3.3.2 变量类型

eZ Publish 模板语言支持以下变量类型：

- 数字
- 字符串
- 布尔
- 数组
- 对象

某些变量可以直接创建，其它的变量需要通过操作符创建。数字和字符串可以直接创建。布尔和数组需要通过操作符创建，对象需要通过各种函数和操作符创建。此外，也可以创建和使用自定义的变量。自定义变量必须以对象形式存在。

数字

数字可以是正数，0，负整数或浮点数。下例演示了如何在模板中使用不同的数字：

```
{13}
{1986}
{3.1415}
{102.5}
{-1024}
{-273.16}
```

字符串

字符串可以由单引号或双引号，'或"，封装。如果省略引号，字符串会被解释为函数名。字符串通常以下面的方法定义：

```
{'This is a string.'}
{"This is another string."}
```

输出：

```
This is a string.
This is another a string.
```

使用引号

字符串内部可以包含引号。可以通过使用不同类型的引号（例如：在双引号封装的字符串中使用单引号）或通过使用转义符"\"：

```
{'The following text is double quoted: "Rock and roll!"  '}
{"The following text is single quoted: 'Rock and roll!'  "}
{'Using both single and double quotes: "Rock\'n roll!"   '}
```

```
{'Using both single and double quotes: \'Rock\'n roll!\' '}
{"Using both single and double quotes: 'Rock'n roll!'    "}
{"Using both single and double quotes: \"Rock'n roll\"   "}
```

输出：

```
The following text is double quoted: "Rock and roll!"
The following text is single quoted: 'Rock and roll!'
Using both single and double quotes: "Rock'n roll!"
Using both single and double quotes: 'Rock'n roll!'
Using both single and double quotes: 'Rock'n roll!'
Using both single and double quotes: "Rock'n roll!"
```

因为模板代码的工作方式（用"{"和"}"定义代码块），右侧的大括号"}"也不要用转义符转义。下例演示了这个问题：

```
{'{ This text is inside curly brackets.\}'}
```

输出：

```
{This text is inside curly brackets.}
```

模板字符串不支持行内变量代入（这一点与 Perl 和 PHP 不同）。因此，不能在字符串中直接嵌入变量。但是，可以用"concat"操作符将变量内容追加到字符串。

布尔

布尔变量必须用"true"或"false"操作符来创建。例如：

```
{true()}
{false()}
```

对于某些操作符和函数，可以用整数代替布尔型。但是，它们不是真正的布尔类型。0 表示 FALSE；非 0 表示 TRUE。某些操作符可以把数组当作布尔值处理。空数组为 FALSE，非空数组为 TRUE。

数组

数组是保存不同类型变量的集合。数据可以是简单的向量或是哈希表（关联数组）。向量的元素可以用数组下标访问（下标从 0 开始）。哈希表的元素可以通过键值访问。数组可以通过"array"操作符创建。哈希表可以通过"hash"操作符创建。下例演示了如何创建数据和哈希：

例 1：数字数组

```
{array( 2, 4, 8, 16 )}
```

上例创建包含四个数字的数组。数组包含如下元素：

下标	元素值
0	2
1	4
2	8
3	18

例 2：字符串数组

```
{array( 'This', 'is', 'a', 'test' )}
```

上例创建了一个包含四个字符串的数组。数组包含以下元素：

下标	元素值
0	'This'
1	'is'
2	'a'
3	'test'

例 3：哈希表

```
{hash( 'Red', 16, 'Green', 24, 'Blue', 32 )}
```

上例创建了包含三对元素的哈希表。数组包含以下元素：

键值	元素值
Red	16
Green	24
Blue	32

对象

模板对象由 PHP 代码或模板操作符创建。系统用对象代表不同种类的数据结构。例如：对象被用来代表节点，翻译，网络商店订单，用户帐号，角色，策略等。参阅“参考手册”中的“对象”章节了解更多。

对象属性

对象由属性构成。每个属性有一个唯一的标识符。属性的类型不可不同。属性可以代表任何类型的数据（数字，字符串，数组等等）甚至其它对象。因为属性是有名字的（每个属性有唯一的标识符），可以通过不同的标识符访问它们的内容。这点与哈希表的用法相同。

下图演示了一个对象("ezdate")的结构。

Attribute	Type	Value
timestamp	string	567990000
is_valid	boolean	TRUE
year	string	1988
month	string	01
day	string	01

上图表明"ezdate"对象由五个属性("timestamp"，"is_valid"，"year"，"month"，"day")构成。"is_valid"是布尔型，其余属性为字符串型。"value"为属性的真是内容。

属性可用性

当从系统中提取一个对象时，有些属性被预提取/计算，而有些属性则没有。这意味着，访问某些属性将需要额外的处理（通常为额外的数据库查询）。在“参考手册”中“对象”章节中，“静态”列揭示了这个属性是否为预提取属性。这一信息将有助于您优化您的代码。

3.3.3 变量用法

模板变量必须以'$'符号开头，如：$my_variable，$object_array，等。变量是大小写敏感的，所以$lollipop与$LolliPop 是不同变量。模板变量可以由系统创建（在 PHP 代码中）或由模板的作者创建（在模板代码中）。不论变量在哪里定义，变量可以被"set"函数重新赋值。某些模板变量有预设值，如：主模板(pagelayout)提供了一套预设变量。

创建与销毁变量

模板中的变量在使用前必须先通过"def"函数定义。变量定义后会一直存在，直到变量被"undef"（稍后解释）函数销毁。在模板中定义的变量在模板的结尾处会被自动销毁。下例演示了"def"和"undef"的基本用法。

```
{def $temperature=32}


    {$temperature}


{undef}
```

上例会输出"32“。{undef}之后，$temperature 被销毁。"def“和"undef"可同时用于多个变量。此外，"undef"可以不使用任何参数。当不使用参数时，"undef"会销毁所有之前在当前模板定义的变量。下例演示了如何用"def"和"undef"来定义和销毁多个变量。

```
{def $weather='warm' $celsius=32 $fahrenheit=90}


The weather is {$weather}: {$celsius} C / {$fahrenheit} F <br />


{undef $celsius $fahrenheit}
```

```
The weather is still {$weather}. <br />


{undef}
```

输出：

```
The weather is warm: 32 C / 90 F

The weather is still warm.
```

上例中，"def"定义了三个变量：$temperature,$celsius 和$fahrenheit。"undef"使用了两次。第一次用来销毁$celsius 和$fahrenheit 变量。第二次没用到任何参数，因而所有剩余的变量（在本例中，只有$temperature）会被销毁。参阅“参考手册”中的"def"和"undef"文档了解更多。

修改变量内容

可以用"set"函数为变量赋值。这个函数可以为任何变量赋值，不论它是在模板中还是在 PHP 代码中创建的。如果系统变量被赋值，系统不会发出警告。"set"会忽略变量的新/旧类型，换言之，"set"可以修改变量的变量类型。"set"不能用来修改数组的元素，哈希的元素和对象。事实上，数组，哈希表和对象不能在模板中被修改。下例演示了"set"函数的用法。

```
{def $weather='warm'}


The weather is {$weather}. <br />


{set $weather='cold'}


The weather is {$weather}.


{undef}
```

输出：

```
The weather is warm.

The weather is cold.
```

与"def"和"undef"一样，"set"可以同时为多个变量赋值。

访问数组元素

向量数组的元素只能用下表访问。这种方法被成为“下标寻址”。哈希的元素可以通过键值访问。这种方法称为“键值寻址”。下例演示了不同寻址方法。

下标寻址

下表寻址可以通过在数组名后追加"."和下标来实现。也可以用常规的"[]"方式。下例演示了不同的下表寻址方法（注意不同的语法，"."和"[]"）。

```
{def $sentence=array( 'Life', 'is', 'very', 'good!' )}

The 1st element is: {$sentence.0} <br />
The 2nd element is: {$sentence.1} <br />
The 3rd element is: {$sentence[2]} <br />
The 4th element is: {$sentence[3]} <br />

{undef}
```

输出：

```
The 1st element is: Life
The 2nd element is: is
The 3rd element is: very
The 4th element is: good!
```

键值寻址

与下表寻址类似。但是用键值替换下表。下例演示了不同的键值寻址方法。

```
{def $sentence=hash( 'first',  'Life',
                     'second', 'is',
                     'third',  'very',
                     'fourth', 'good!' )}

The 1st element is: {$sentence.first} <br />
The 2nd element is: {$sentence.second} <br />
The 3rd element is: {$sentence[third]} <br />
The 4th element is: {$sentence[fourth]} <br />

{undef}
```

输出：

```
The 1st element is: Life
The 2nd element is: is
The 3rd element is: very
The 4th element is: good!
```

访问对象属性

与哈希的键值寻址类似。但是不支持"[]"方式。下例演示了如何访问对象的不同属性。

```
The ID of the node: {$node.node_id} <br />

The ID of the object encapsulated by the node: {$node.object.id} <br />

The name of the object: {$node.object.name} <br />

First time the object was published: {$node.object.published|l10n( shortdate )} <br />
```

如果$node 变量的节点 ID 是 14，对象 ID 是 13，对象名为"Birthday“，发布时间是"2003 年 4 月 1 日"，则输出如下：

```
The ID of the node: 44
The ID of the object encapsulated by the node: 13
The name of the object: Birthday
First time the object was published: 01/04/2003
```

3.3.4 调查数组和对象

通过使用"attribute"操作符，可以快速调查数组和对象的内部数据。这个操作符生成对象或数组键值，属性名和/或方法名的一览。默认情况下，之显示数据的键值和对象的属性标识符。通过传送"show"作为第一个参数，操作符也可以显示值。第二个参数可以指定遍历的深度（默认为 2）。下例演示了如何用这个操作符调查"ezcontentobjectnode"对象的内容。

```
{$node|attribute( show, 1 )}
```

输出：

Attribute	Type	Value
node_id	string	2
parent_node_id	string	1
main_node_id	string	2
contentobject_id	string	1
contentobject_version	string	10
contentobject_is_published	string	1
depth	string	1
sort_field	string	8
sort_order	string	1
priority	string	0
modified_subnode	string	1108118324
path_string	string	'/1/2/'
path_identification_string	string	''
is_hidden	string	0

is_invisible	string	0
name	string	'eZ publish'
data_map	array	Array(6)
object	object[ezcontentobject]	Object
subtree	array	Array(114)
children	array	Array(44)
children_count	string	44
contentobject_version_object	object[ezcontentobjectversion]	Object
sort_array	array	Array(1)
can_read	boolean	true
can_create	boolean	false
can_edit	boolean	false
can_hide	boolean	false
can_remove	boolean	false
can_move	boolean	false
creator	object[ezcontentobject]	Object
path	array	Array(0)
path_array	array	Array(2)
parent	object[ezcontentobjecttreenode]	Object
url	string	''
url_alias	string	''
class_identifier	string	'folder'
class_name	string	'Folder'
hidden_invisible_string	string	'-/-'
hidden_status_string	string	'Visible'

如上所示，从节点对象中可以提取很多信息。出字符串和数字外，对象也可以包含其它对象。例如：节点的"creator"是一个"ezcontentobject"对象。可以对"creator"进一步调查：

```
{$node.creator|attribute( show, 1 )}
```

输出如下：

Attribute	Type	Value
id	string	14
section_id	string	2
owner_id	string	14
contentclass_id	string	4
name	string	'Administrator User'
is_published	string	0

published	string	1033920830
modified	string	1033920830
current_version	string	1
status	string	1
current	object[ezcontentobjectversion]	Object
versions	array	Array(1)
author_array	array	Array(1)
class_name	string	'User'
content_class	object[ezcontentclass]	Object
contentobject_attributes	array	Array(5)
owner	object[ezcontentobject]	Object
related_contentobject_array	array	Array(0)
related_contentobject_count	string	0
reverse_related_contentobject_array	array	Array(0)
reverse_related_contentobject_count	string	0
can_read	boolean	false
can_create	boolean	false
can_create_class_list	array	Array(0)
can_edit	boolean	false
can_translate	boolean	false
can_remove	boolean	false
can_move	boolean	false
data_map	array	Array(5)
main_parent_node_id	string	13
assigned_nodes	array	Array(1)
parent_nodes	array	Array(1)
main_node_id	string	15
main_node	object[ezcontentobjecttreenode]	Object
default_language	string	'eng-GB'
content_action_list	boolean	false
class_identifier	string	'user'
class_group_id_list	array	Array(1)
name	string	'Administrator User'
match_ingroup_id_list	boolean	false

如上所述，"attribute"操作符也可以用于调查数组。下例演示了如何用它调查"data_map"数组：

```
{$node.creator.data_map|attribute( show, 1 )}
```

输出如下：

Attribute	Type	Value
first_name	object[ezcontentobjectattribute]	Object
last_name	object[ezcontentobjectattribute]	Object
user_account	object[ezcontentobjectattribute]	Object
signature	object[ezcontentobjectattribute]	Object
image	object[ezcontentobjectattribute]	Object

3.3.5 控制结构

eZ Publish 模板语言提供了以下的控制结构：

- IF-THEN-ELSE
- SWITCH
- WHILE
- DO...WHILE
- FOR
- FOREACH

IF-THEN-ELSE

参阅下例：

例 1:

```
{if eq( $var, 128 )}
    Hello world <br />
{else}
    No world here, move along. <br />
{/if}
```

例 2:

```
{if eq( $fruit, 'apples' )}
    Apples <br />
{elseif eq( $fruit, 'oranges' )}
    Oranges <br />
{else}
    Bananas <br />
{/if}
```

SWITCH

```
{switch match=$fruits}

    {case match='apples'}
        Apples <br />
    {/case}

    {case match='oranges'}
        Oranges <br />
    {/case}

    {case}
        Unidentified fruit! <br />
    {/case}

{/switch}
```

如果$fruits 初始值为'oranges'，则输出：

```
Oranges
```

WHILE

```
{while ne( $counter, 8 )}

    Print this line eight times ({$counter}) <br />
    {set $counter=inc( $counter )}

{/while}
```

如果$counter 初始值为 0，则输出：

```
Print this line eight times (0)
Print this line eight times (1)
Print this line eight times (2)
Print this line eight times (3)
Print this line eight times (4)
Print this line eight times (5)
Print this line eight times (6)
Print this line eight times (7)
```

DO...WHILE

```
{do}

    Keep printing this line ({$counter}) <br />
    {set $counter=inc( $counter )}

{/do while ne( $counter, 8 )}
```

如果$counter 初始值为 0，则输出：

```
Keep printing this line (0)
Keep printing this line (1)
Keep printing this line (2)
Keep printing this line (3)
Keep printing this line (4)
Keep printing this line (5)
Keep printing this line (6)
Keep printing this line (7)
Keep printing this line (8)
```

FOR

```
{for 0 to 7 as $counter}

    Value of counter: {$counter} <br />

{/for}
```

输出：

```
Value of counter: 0
Value of counter: 1
Value of counter: 2
Value of counter: 3
Value of counter: 4
Value of counter: 5
Value of counter: 6
Value of counter: 7
```

FOREACH

```
{foreach $objects as $object}


    {$object.name} <br />


{/foreach}
```

如果$objects 是一个数组，且包含 4 个对象，对象名依次为："Emmett Brown","Marty McFly","Lorraine Baines"和"Biff Tannen"，则输出：

```
Emmett Brown

Marty McFly

Lorraine Baines

Biff Tannen
```

3.3.6 函数与操作符

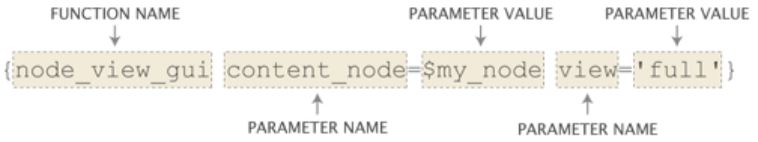
eZ Publish 模板语言提供了一套不同功能的函数和操作符。此外，可以开发特殊用途的模板操作符来扩展系统。自定义操作符必须用 PHP 开发。

模板函数

模板函数可以用如下语法调用：

```
{function_name parameter1=value1 parameter2=value2 ...}
```

函数接受 0，1 或更多参数。参数必须跟在函数名之后，用空格分割。因为参数有参数名，参数可以用任何顺序传送。每个参数必须用'='赋值。下图演示了一个常用函数的用法。



上例调用"node_view_gui"函数。这个函数引入视图模式对应的视图模板来显示指定的节点。节点用"content_node"指定，视图模式用"view"参数指定。

模板操作符

操作符只支持匿名参数。操作符支持管道操作符。

操作符用以下方法调用：

```
{$input_parameter|operator_name( parameter1, parameter2 ... )}
```

因为操作符只支持匿名参数，参数必须按照操作符文档中指定顺序传送。此外，参数必须用逗号分割。下图演示了一个常用操作符用法。

上例演示了"datetime"操作符的用法。这个操作符可以把 UNIX 时间戳格式化为时间格式。时间戳由 $yesterday_evening 变量提供。第一个参数告诉操作符，输出应该用自定义格式。格式由第二个参数指定（小时:分钟）。

管道

操作符从左侧接受输入，在右侧生成输出。多个操作符可以用管道操作符"|"连接。管道确保前一个操作符输出会被作为下一个操作符的输入。下例演示了如何用管道和操作符创建一个字符串。

```
{concat( 'To ', 'The ' )|prepend( 'Back ' )|append( 'Future' )}
```

输出：

```
Back To The Future
```

3.4 模板基本任务

本章特别介绍了一些常用的与模板开发相关的任务。

模板引入

模板文件可以通过"include"函数导入。因为这个函数可以用来引入 eZ Publish 目录下的任何位置的任何文件，比如显式告诉 eZ Publish 在"design"目录中寻找文件。可以在"路径/文件名"前添加"design:"前缀来达到这个目的。下例演示了如何引入"footer.tpl"，这个文件位于"design"下的"templates"目录中。

```
{include uri='design:footer.tpl'}
```

如果请求的文件在主界面中没有找到，系统会在附加界面和标准界面中继续寻找。请参阅“自动备选系统”了解更多。

输出清理

包含非法字符的变量在输出之前都应该用"wash"操作符来清理。这个操作符确保输出中不会含有破坏 HTML 结构的符号。下例演示了它的用法。

```
{def $bogus_string='hello < world'}

{$bogus_string|wash()}
```

输出：

```
hello < world
```

电子邮件模糊化

除了确保输出内容的安全外，"wash"操作符也可以用来对 WEB 页面上的电子邮件地址进行模糊化处理。模糊化处理的电子邮件地址会减少邮件地址被检索机器人加入垃圾邮件列表的概率。下例演示

了"wash"操作符如何用于电子邮件模糊化。

```
{def $email_address='allman@example.com'}
{$email_address|wash( 'email' )}
```

输出：

```
allman[at]example[dot]com
```

字符串连接

"concat"操作符可以把多个字符串连接成一个字符串。下例演示了它的用法。

```
{def $my_string='sausage'}
{concat( 'Liver ', $my_string, ' sandwitch' )}
```

输出：

```
Liver sausage sandwitch
```

自定义的视图参数

用于显式节点的 URL 中可以包含自定义的视图参数。自定义的视图参数必须追加在 URL 的最后。每个参数必须有一个参数名和参数值。参数名必须由小括号"("和")"封装。参数的每个元素必须由"/"分割。下例演示了如何使用在节点系统 URL 中是使用自定义的视图参数（除系统视图参数外）。

```
http://www.example.com/content/view/full/13/(color)/green/(amount)/34
```

相同的参数也可以追加在虚拟 URL 后：

```
http://www.example.com/company/about_us/(color)/green/(amount)/34
```

当使用自定义视图参数时，系统会为这些参数在哈希表中创建记录，参数名为哈希表的键值。所有的参数值都会被作为字符串处理。参数的键值和值会被保存在$view_paramters 中。上例的参数会生成如下的哈希表：

Key	Type	Value
color	string	green
amount	string	34

下例演示了如何在模板中提取上述参数：

```
The color is: {$view_parameters.color} <br />
The amount is: {$view_parameters.amount} <br />
```

输出：

```
The color is: green
The amount is: 34
```

"edit.tpl"模板中的自定义视图参数

在 3.9 版本之前，您不能向"content"模块的"edit"视图传送自定义视图参数。从 3.9 版本，系统开始支持这

种特性。下例演示了如何向"edit.tpl"传输这种参数：

```
http://www.example.com/content/edit/13/03/eng-GB/(color)/green/(amount)/34
```

这会要求 eZ Publish 在编辑对象 13 的版本 3 的 eng-GB 翻译的时候同时接受上述两个自定义参数。

3.4.1 URL 处理

当链接，非内容的图片，CSS 文件等被引入模板时，必须使用一个相应的模板操作符来确保指向被引入文件的路径正确。下面操作符中的一个可以被使用：

- ezurl
- ezimage
- ezdesign

ezurl

ezurl 确保 URL 与 eZ Publish 的安装目录，访问方法以及运行环境（非虚拟主机模式，虚拟主机模式，等）无关。换言之，通过使用 ezurl 操作符，URL 总是可用。使用 ezurl 操作符时，只需要提供 URL 中 eZ Publish 特有的部分。操作符会生成其余部分（http://，主机名，域名，目录，站点入口，端口，等）最终的输出为一个合法的地址。这种解决方案可以确保在站点被迁移并/或当访问方法被改变的时候，不需要修改所有的 URL。默认情况下，"ezurl"操作符输出的 URL 会自动被双引号封装。换言之，操作符的输出可以被直接追加到 HTML 代码中的超链接中。下例演示了这个操作符的用法。

链接到模块/视图（使用系统 URL）

```
<a href={'/user/login'|ezurl()}>Login</a>
```

上例演示了如何创建一个链接到"user"模块的"login"视图的链接。"/user/login"只是一个示例，另外一个例子，链接到一个节点："/content/view/full/34"。如果 eZ Publish 运行于"ezpublish"目录，主机名"www.example.com"，使用"URL 访问方法"且站点入口的名称为"my_company"，这个操作符会输出以下内容：

```
"http://www.example.com/ezpublish/index.php/my_company/user/login"
```

如果运行于"虚拟主机模式"且使用"主机访问方法"，下记 URL 会被生成：

```
"http://www.example.com/user/login"
```

链接到一个节点（使用节点的虚拟 URL）

当创建一个链接到节点的链接（使用节点的虚拟 URL，也被称为 URL 别名）时，地址通过"ezurl"处理。这样做的原因是内部 URL 表只包含 eZ Publish 特有的 URL 部分。下例演示了如何用"ezurl"创建节点的虚拟 URL。

```
<a href={$node.url_alias|ezurl()}>Link to a node</a>
```

如果节点的 URL 别名为"company/about_us"且 eZ Publish 运行于虚拟主机模式且使用主机访问方法，以下 URL 会被生成：

```
"http://www.example.com/company/about_us"
```

参阅"基本概念"中的"URL 翻译"了解更多。

ezimage

ezimage 与 ezurl 工作方式完全相同。但是它不包含"index.php"部分。每次在模板中引入非内容图片时，都需要用到这个操作符。图片必须被放置在某个界面的"images"目录中。无论 eZ Publish 安装在哪个目录，使用何种访问方法和/或使用何种运行模式，这个操作符都会生成一个正确的指向图片的地址。下例演示了"ezimage"的用法：

```
<img src={'women.jpg'|ezimage()} alt="This is my image." ... />
```

如果 eZ Publish 使用主机访问模式且站点入口使用"my_design"界面，操作符会生成如下输出：

```
"http://www.example.com/design/my_design/images/women.jpg"
```

如果图片被放置在"images"目录下的某个子目录中，子目录的名称必须在模板中明示。如果请求的文件在主界面中没有找到，系统会在附加界面和标准界面中继续搜寻。参阅“自动备选系统”的文档了解更多。

ezdesign

"ezdesign"操作符与"ezurl"的工作方式完全相同。但是它不包含"index.php“部分。在模板中引入界面元素（CSS 文件，Javascript 等）时，这个操作符负责生成指向目标文件的正确 URL。下例演示了如何用这个操作符引入 CSS 文件。

```
...
<style type="text/css">
    @import url({'stylesheets/my_stuff.css'|ezdesign()});
</style>
...
```

如果 eZ Publish 在主界面中没有找到指定的文件，系统会在附加界面和标准界面中继续搜寻。请参阅“自动备选系统”的文档了解更多。

3.5 内容提取

eZ Publish 中保存的信息可以用"fetch"模板操作符提取。这个操作符会调用模块的 fetch 函数。它通常被用来调用 content 模块以提取节点，对象等内容。fetch 操作符只能用于定义了 fetch 函数的模块。请参阅“参考手册”中的“Fetch 函数族”了解详细的信息。以下的模型演示了 fetch 操作符的语法。

```
fetch( <模块>, <函数>, <参数> )
```

参数	描述
模块	目标模块名。
函数	目标模块的 fetch 函数名。
参数	模块 fetch 函数的参数列表（哈西表形式）。

模块的 fetch 函数和参数列表在模块的"function_definition.php“中定义，这个文件位于模块的目录中。

提取单个节点

下例演示了如何从数据库中提取单个节点。

```
{def $my_node=fetch( content, node, hash( node_id, 13 ) )}


...


{undef}
```

上例要求 eZ Publish 从 content 模块中提取单个节点。只用到了一个参数，要提取的节点 ID。这个操作符会返回一个"ezcontentobjecttreenode"对象，这个对象被设置到$my_node 变量中。这个变量可以用来提取节点与节点对象的信息。例如：可以提取对象的名称，属性和发布时间等。如果节点不可用/不存在或当前用户没有权限访问它，这操作符会返回 FALSE。

提取多个节点

可以从提取某个节点下的多个节点。这可以通过用"list"替换"node"作为"fetch"操作符的第二个参数来达到。下例演示了如何提取节点 13 下的所有节点。

```
{def $my_node=fetch( content, list, hash( parent_node_id, 13 ) )}


...


{undef}
```

这个操作符会返回一个"ezcontentobjecttreenode"对象的数组。"content"模块的"list" fetch 函数可以接受多个参数。这些参数是可选的且可以用来优化内容提取，例如：筛选特定的节点。下表揭示了一部分常用参数。

参数	描述
sort_by	节点排序的方法和顺序（必须用数组指定）。
limit	提取的节点数量。
offset	提取开始的位置（偏移量）。
class_filter_type	过滤的类型，"include"或"exclude"
class_filter_array	应该被过滤器包含或排除的节点类型（必须用数组指定）。

下例演示了如何提取节点 13 下最新的 10 篇文章。

```
{def $my_node=fetch( content,
                     list,
                     hash( parent_node_id,     13,
                           limit,              10,
                           class_filter_type,  include,
```

```
                            class_filter_array, array( 'article' ) ) )}


...


{undef}
```

请参阅"list" fetch 函数的文档页了解更多。

3.5.1 输出节点与对象数据

只要一个代表某个节点的"ezcontentobjectreenode"对象在模板中可用，就可以用它来输出关于节点和节点对象的内容。以下内容演示了如何提取最常用的数据。

一般信息

对象明

```
{$node.name|wash}
```

对象名可以从节点直接访问（换言之，不需要用$node.object.name）。"wash"操作符确保输出不包含任何破坏 HTML 的字符且/或序列。

对象最后修改日期/时间

```
{$node.object.modified|l10n( 'shortdatetime' )}
```

因为修改时间是一个 UNIX 时间戳，它必须被格式化后输出。可以用"l10n"操作符输出本地化的日期/时间。

对象属主的名称

```
{$node.object.owner.name|wash}
```

最后修改者的名称

```
{$node.object.current.creator.name|wash()}
```

对象的类名

```
{$node.object.class_name|wash()}
```

对象属性

可以通过"data_map"方法访问对象的属性。这个方法返回"ezcontentobjectattribute"对象的哈希表。每个对象代表一个属性。键值为属性标识符。下例演示了如何提取"first_name"属性。

```
{$node.object.data_map.first_name}
```

上例不产生任何输出，因为请求的数据需要被格式化。有两种方法输出属性的内容：

- 原始输出（".output"）
- 格式化输出（"attribute_view_gui"函数）

原始输出与格式化输出的区别在于格式化输出通过一个模板输出属性的内容。原始输出只是在同一个模板内简单的输出属性内容。永远应该用"attribute_view_gui"函数输出属性内容。原始输出只有在必要的情况下使用（例如：当用 IF 语句检查某个属性的值的时候）。

原始输出

原始输出正如其定义一样：输出属性内保存的原始内容。实际的语法取决于属性的数据类型。大部分情况下，可以用".output"输出原始数据。

一般解决方案

下例演示了如何输出"my_attribute"的内容

```
{$node.object.data_map.my_attribute.content}
```

XML 块

下例演示了如何输出一个 XML 块"my_xml"的内容。

```
{$node.object.data_map.my_xml.content.output.output_text}
```

图片

下例演示了如何输出"my_image"内保存的图片。

```
<img src="{$node.object.data_map.my_image.content[image_size].full_path}" ... />
```

格式化输出

每种数据类型都有一套用于不同上下文中显示内容的模板。每种数据类型至少有两种模板："view"模板和"edit"模板。"view"模板用于显示属性内容，"edit"模板用于编辑数据。默认的数据类型模板位于"design/standard/templates/content/datatype"目录。

"attribute_view_gui"函数允许用数据类型的"view"模板来显示属性。下例演示了如何用这个函数。

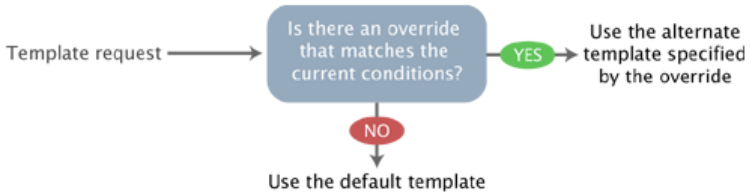
```
{attribute_view_gui attribute=$node.object.data_map.name_of_any_attribute}
```

上例将会正确生成任何属性的内容（与数据类型无关）。

3.6 模板重设系统

模板重设系统允许用自定义的模板替换默认模板。理论上，这种机制允许对系统内任何模板（包括通过"include"函数与"design:"前缀引入的模板）进行重设。要以不同方式显示不同节点时，模板重设就变个很有用。

一个视图模板的重设通常由一组重设条件触发。如果条件匹配，系统会用重设模板替换原始模板。不同的视图支持不同的重设条件，某些视图不提供重设条件。请参阅“参考手册”中的“模板重设条件”章节了解更多。"content"模块的"view"视图提供的重设条件最为灵活（在显示节点时使用）。下图演示了重设机制如何被嵌入系统的其余部分。



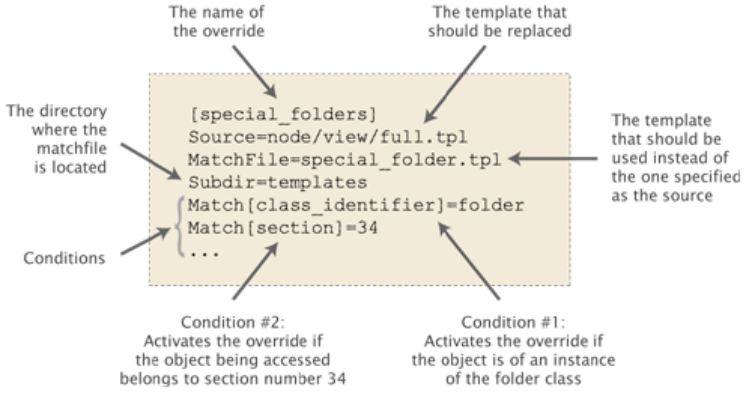
模板重设必须在站点入口的"override.ini.append.php"中定义。这个文件有重设块构成。一个重设块是一组告知 eZ Publish 在某种特殊条件下使用某个重设模板的规则。没个重设块有一个唯一的名称。对于一个重设块，以下信息必须被设置：

- 一个唯一的名称
- 被重设的原始模板
- 重设模板
- 重设模板所在目录名称（通常为"templates"）
- 一组触发重设模板的条件/规则

注意：

规则/条件是可选的。如果没有指定规则，这个重设会一直有效。

下图演示了一个典型的模板重设场景。



上列定义了一条重设"special_folders"。这条重设在系统用"full"视图模式显示某个节点时会被使用。这条重设只有在节点的类为文件夹类且属于 section 34 才会被激活。当重设被激活时，系统会使用重设模板（"override/templates/special_folder.tpl"，在主界面中）。如果 eZ Publish 在主界面中找不到这个模板，它会在附加界面和标准界面中继续搜寻。请参阅“自动备选系统”了解更多。

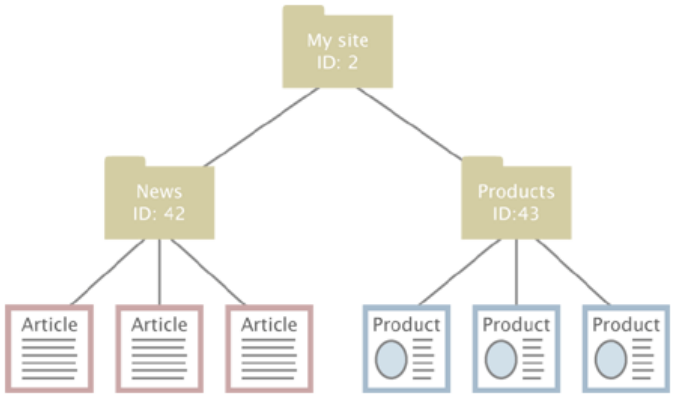
多重/冲突重设

重设的优先级是由它们在配置文件中的位置决定的。如果有多条规则类似/等价的重设，eZ Publish 会使用最上面的一条，因而其余重设会失效。因为这个缘故，由节点 ID 或对象 ID 出发的重设应被放在最上面，否则他们可能永远都不会被触发，因为具备更一般规则的重设会因为位于靠前的位置而有更高的优先级。

3.6.1 重设示例

下例演示了如何利用模板重设系统在不同条件下显示不同模板。

假设我们有一个简单的节点树。它由两个文件夹构成："News"和"Products"。"News"文件夹包含新闻文章，"Products"文件夹包含商品。下图演示了这个节点树。



没有任何重设时，eZ Publish 很可能会用同一个模板显示所有节点。这可能是 standard 界面中的默认的"full"视图模式模板。然而，我们希望的是用不同模板显示不同节点。我们可能希望系统以下面的方式工作：

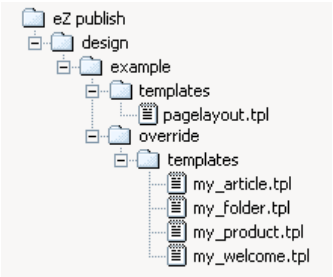
- 访问"My site"节点时，显示一个特殊的"welcome"模板。
- 访问文件夹时显示一个自定义文件夹模板
- 访问新闻文章时，显示一个自定义新闻文章模板
- 访问商品时，显示一个自定义的商品模板

上述需求可以通过创建几条重设来达到。"welcome"页面应该用"My site"节点 ID 来触发。其余的模板可以用类标识符来触发。下例演示了"override.ini.append.php"文件的内容：

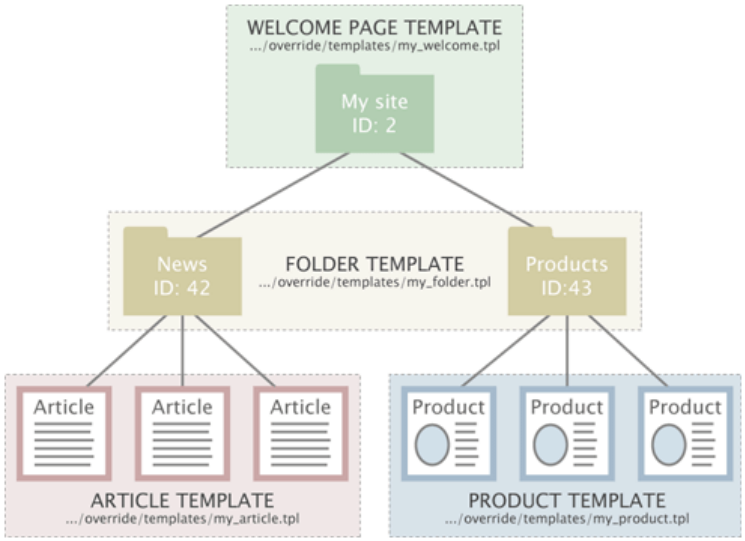
```
# Override for welcome page
[welcome_page]
Source=node/view/full.tpl
MatchFile=my_welcome.tpl
Subdir=templates
```

```
Match[node]=2


# Override for folders
[my_folder]
Source=node/view/full.tpl
MatchFile=my_folder.tpl
Subdir=templates
Match[class_identifier]=folder

# Override for articles
[news_articles]
Source=node/view/full.tpl
MatchFile=my_article.tpl
Subdir=templates
Match[class_identifier]=article

# Override for products
[products]
Source=node/view/full.tpl
MatchFile=my_product.tpl
Subdir=templates
Match[class_identifier]=product
```

重设模板应该被放置在主界面的"override/templates"目录下。下图演示了可能的目录结构，假设主界面为"example"。



当系统运行时，不同的重设会根据实际的条件被触发。下图演示了哪里/何时不同的重设会被触发。



每次访问一个文件夹节点时，系统会使用"my_folder.tpl"模板。当访问文章节点时，系统会使用"my_article.tpl"。当访问商品节点时，系统会使用"my_product.tpl"模板。当访问节点 2（"My site"节点）时，系统会使用"my_welcome.tpl"。

4 系统特性

本章包含了关于 eZ Publish 各种特性的介绍，配置与使用方法。

4.1 系统记帐

系统可以根据用户的操作来生成系统记账。这个特性可以用于有很多管理员和编辑的大型网站来记录用户的行为。例如：记账功能可以用来找出哪个用户删除了内容以及用户的请求来自于哪个 IP 地址等等。

系统提供了一套内建的记账功能，它可以用来为不同操作生成记账。系统至少会记录以下信息：

- 何时发生（时间戳）
- 来源（IP 地址）
- 操作员（用户名和 ID）

注意，大部分记账功能提供附加信息。下例演示了节点被删除后的一条记账内容。

```
[ May 23 2007 14:47:58 ] [127.0.0.1] [editor:16]
Node ID: 124
Old parent node ID: 2
New parent node ID: 59
Object ID: 114
```

```
Content Name: Folder

Comment: Moved the node to the given node: eZContentObjectTreeNode::move()
```

下表列出了可用的内建记账功能，触发条件，记账信息类型以及默认日志文件。

记账功能	操作	记账信息	默认日志文件
user-login	登录成功	时间戳 Ip 地址 用户（用户名:ID）	login.log
user-failed-login	登录失败	时间戳 Ip 地址 用户（用户名:ID）	failed_login.log
content-move	内容位置变更	时间戳 Ip 地址 用户（用户名：ID） 旧父节点 ID 新父节点 ID 对象 ID 对象名 注释	content_move.log
content-delete	删除内容	时间戳 Ip 地址 用户（用户名：ID） 旧父节点 ID 新父节点 ID 对象 ID 对象名 注释	content_delete.log
role-change	角色与策略变更	时间戳 Ip 地址 用户（用户名：ID） 角色 ID 角色名 注释	role_change.log
role-assign	指派角色给用户和用户组。	时间戳 Ip 地址 用户（用户名：ID） 角色 ID 角色名 注释	role_assign.log
section-assign	分区指派	时间戳 Ip 地址 用户（用户名：ID） 分区 ID 分区名 节点 ID 对象 ID 对象名 注释	section_assign.log

order-delete	删除网络商店订单	时间戳 IP 地址 用户（用户名：ID） 订单 ID 注释	order_delete.log

配置

默认情况下，记账功能被禁用，可以在"audit.ini"重设文件中"[AuditSettings]"下的"Audit"设置。
用"AuditFileNames"数组配置日志文件。用记账函数名称作为键值，文件名作为值。注意，默认情况下，记账会被写入不同文件（参考上表）。

"LogDir"用来指定记账日志文件保存目录。默认目录为"var/log/audit"。

示例

假设您只希望为成功登录和角色变更记账，可以创建一个"audit.ini"的重设文件并做如下配置：

```
[AuditSettings]
Audit=enabled
LogDir=var/log/my_audit
AuditFileNames[]
AuditFileNames[user-login]=login.log
AuditFileNames[role-change]=role_change.log
```

成功登录信息会被写入到"login.log"。关于角色变更的信息会被写入到"role_change.log"。两个文件都位于"var/log/my_audit"目录。两个文件中的记录都会包含：时间戳，用户（用户名和 ID），IP 地址。与角色变更的记录还会包含其他附加信息。

创建新的记账函数

本章为要开发新的记账函数的 PHP 程序员提供指导。

有时您可能需要开发新的记账功能，例如：要求系统将某个特殊的操作记录到特殊的日志文件。例如：如果你希望创建一个新的记账函数"my-new-audit"，并且将某个操作的信息记录到"info.log"，您可以：

1. 确定记账功能已经启用且已添加如下配置：

```
AuditFileNames[my-new-audit]=info.log
```

2. 在定义操作的 PHP 代码中，添加如下代码：

```
include_once( "kernel/classes/ezaudit.php" );
eZAudit::writeAudit( 'my-new-audit', array( 'User id' => $userID,
```

```
        'Comment' => 'The operation XYZ was performed.' ) );
```

"名称" => "值"定义了当操作发生时，哪些信息需要被写入哪个日志文件。

例如，日志中的记录如下：

```
[ May 23 2007 14:44:04 ] [127.0.0.1] [anonymous:10]
User id: 10
Comment: The operation XYZ was performed.
```

4.2 策略功能

eZ Publish 内建的访问控制机制基于角色和策略。策略是一组授权访问模块的特定或全部函数的规则。函数被指派到模块的视图，因而对视图的访问是由指派给它的函数控制的。

以下代码（摘自 eZ Publish 源码）演示了在"kernel/notification/module.php"中如何指派"notification"模块的函数－视图。

```
<?php

$Module = array( "name" => "eZNotification",
                 "variable_params" => true );

$ViewList = array();
$ViewList["settings"] = array(
    "functions" => array( 'use' ),
    "script" => "settings.php",
    'ui_context' => 'administration',
    "default_navigation_part" => 'ezmynavigationpart',
    "params" => array( ),
    'unordered_params' => array( 'offset' => 'Offset' ) );

$ViewList["runfilter"] = array(
    "functions" => array( 'administrate' ),
    "script" => "runfilter.php",
    'ui_context' => 'administration',
    "default_navigation_part" => 'ezsetupnavigationpart',
    "params" => array( ) );
```

```
$ViewList["addtonotification"] = array(
    "functions" => array( 'use' ),
    "script" => "addtonotification.php",
    'ui_context' => 'administration',
    "default_navigation_part" => 'ezcontentnavigationpart',
    "params" => array( 'ContentNodeID' ) );

$FunctionList['use'] = array( );
$FunctionList['administrate'] = array( );

?>
```

如上面代码所示，有两个函数被指派给三个函数。"administrate"函数被指派给"runfilter"视图，"use"函数被指派给"addtonotification"和"settings"视图。

多函数指派

一个视图可以有多个函数指派。从 3.9.3 版本开始，系统在函数－视图指派中使用逻辑操作符（"and","or"）。下例演示了这个特性如何工作。

例 1

"content"模块的"tipafriend"视图有两个函数指派。以下代码摘自"kernel/content/module.php"。

```
$ViewList['tipafriend'] = array(
    'functions' => array( 'tipafriend', 'read' ),
    'default_navigation_part' => 'ezcontentnavigationpart',
    'script' => 'tipafriend.php',
    'params' => array( 'NodeID' ) );
```

本例中的代码限定用户必须被同时授权访问"tipafriend"和"read"函数才能使用"tipafriend"视图（"content"模块的一部分）。注意，有如下一种方法来指派同样的函数。

```
...
'functions' => array( 'tipafriend and read' ),
...
```

注意，"and"操作符也可以用"&&"代替。

例 2

"section"模块的"list"视图有三个函数指派。下面的代码摘自"kernel/section/module.php"。

```
$ViewList['list'] = array(
    'functions' => array( 'view or edit or assign' ),
    'script' => 'list.php',
    'default_navigation_part' => 'ezsetupnavigationpart',
    "unordered_params" => array( "offset" => "Offset" ),
    'params' => array( ) );
```

上例的代码限定用户只要被授权访问"view","edit"或"assign"函数中的一个就可以使用"list"视图（"section"模块的一部分）。注意，"or"操作符也可以用"||"来代替。

缺失函数

某些模块没有函数（如，"search"和"collaboration"模块）。这种情况下，授权访问这个模块意味着用户被授权访问这个模块的所有视图。

如果一个模块既有有函数指派的视图也有没有函数指派的视图，只有当用户被授权访问整个模块才可以访问那些没有函数指派的视图。

对早期版本的补充说明

在 3.9.3 版本（除 3.8.9 及以后的 3.8.x 版本）以前版本中，授权用户访问模块的函数将会授权用户访问：

- 被指派该函数的视图
- 没有函数指派的视图

例如，在 eZ Publish 3.9.2，"shop"模块的"discountgroupview"视图没有函数指派。可以访问"shop"模块的"buy"函数的匿名用户也可以访问"discountgroupview"视图（以及其它没有函数指派的视图）。基于安全考虑，这一特性在"3.10.0 beta1“，"3.9.3"和"3.8.9"版本中被修改。参照“发布声明”了解更多。

自定义的模块如果有函数不应该包含没有函数指派的视图。

函数限制

策略（授权访问模块的函数）可以通过函数限制被进一步限定。如果函数支持限定条件就可以做这种限定。函数可以支持 0，1 或多个限定条件。下面的代码演示了如何在"kernel/content/module.php"对"diff","hide"和"tipafriend"函数指定限定条件。

```
...
$FunctionList['diff'] = array( 'Class' => $ClassID,
'Section' => $SectionID,
'Owner' => $Assigned,
'Node' => $Node,
'Subtree' => $Subtree);
...
```

```
$FunctionList['hide'] = array( 'Subtree' => $Subtree );

...

$FunctionList['tipafriend'] = array();

...
```

如上面的代码所示，"diff"函数支持五个限定条件，"hide"函数支持一个，"tipafriend"函数不支持限定条件。参考“基本概念”中的“访问控制”一章了解更多。

4.3 多语言

在 eZ Publish3.7 及早期版本中，你必须为所有对象指定主语言（例如，每个对象必须至少存在于一种语言）。此外，您可以指定附加语言用于内容翻译。“多语言功能”实现在版本级别且允许内容的版本存在于多种语言（语言在此被称为“翻译”）中。旧方案的缺点在于当需要多种翻译时，只有一个翻译者可以编辑对象。换言之，编辑者必须彼此等待并串行工作，因为同一时间只能有一个用户编辑对象的版本。这种功能已经被修改。

从 3.8 版本开始，系统不再需要主语言。您可以有一篇只存在于英文的文章同时另一篇文章却存在于法语。为对象选择语言后，您可以将它翻译为任何可用语言。同一个对象的翻译者可以并行工作（一个用户只能同时编辑一个版本与翻译）。下一章会简单解释一些与多语言功能相关的主要规则与术语。

可翻译的类属性

从 3.9 版本开始，类属性名也可被翻译。参阅“可翻译类属性”文档了解更多。

区域

一个区域是一组与国家/地区特有的配置，如：语言，字符集，数字格式，货币符号，日期时间格式，月份，星期的表示等。eZ Publish 在"share/locale"下保存了很多预定义的区域 INI 文件。这些文件用区域标识符命名。

区域标识符有三个语言代码字符与两个大写的国家代码构成。如："eng-US“（英文，美国）或"nor-NO"（挪威语，挪威）。语言与国家代码由"ISO 639"和"ISO 3166-1 alpha-2"标准定义。

eZ Publish 默认使用"eng-GB"区域。请参阅“配置站点区域”章节了解如何配置站点区域，翻译管理界面，创建自定义区域。

默认语言

从 3.8 版本开始，"ContentObjectLocale"指定的不是主语言而是默认语言。这个语言会被用来作为某些支持可选语言参数的 PHP 函数的默认值。INI 配置中的默认值为"eng-GB"。

例

假设已在"ContentObjectLocale"中指定"nor-NO"。这种情况下，如果用"eZContentClass::instance()"函数初

始化一个对象且不指定语言，那么挪威语会被使用。

可翻译的国家名

从 3.9 版本开始，可以把国家名翻译为多种语言。例如：您可以要求系统在使用挪威区域时候，在语言列表中用"Frankrike"替换"France"，"Norge"替换"Norway"。参考“可翻译的国家名”文档页面了解更多。

翻译语言

可以选择创建和/或翻译对象的语言。所使用的语言集合称为翻译语言。可以用管理界面管理翻译语言。系统目前最多支持 30 种翻译语言。

初始化/主语言

可以用任何通过安装向导或管理界面中“设置/语言”安装的语言来创建对象。创建对象时，所使用的语言会被设置为对象的初始化/主语言。例如，如果用匈牙利语创建一篇文章，它的初始化/主语言就理所当然为匈牙利语。

对象主语言的内容不能被删除。但是，如果对象存在于多种语言中，主语言可以被改变，因而原来的主语言内容可以被删除。可以在管理界面中的“语言”窗口修改主语言或删除翻译。

重要说明

注意：

“初始化语言 (initial language) ”与“主语言 (main language) ”是同一件事。代码和数据库表用“初始化语言 (initial language) ”，而管理界面用“主语言 (main language) ”。这种不一致有望在将来的发行版本中得到修正。

站点语言

从 3.8 版本开始，站点显示的语言。这种配置称为“站点语言”。可以在站点入口的"site.ini.append.php"中"[RegionalSettings]"下的"SiteLanguageList[]"数组配置。您可以在这个数组中添加区域标识符来配置站点语言与其优先级（排在上面的区域有更高的优先级）。系统会尝试用最优先的语言显示内容，如果内容不存在于这种语言中，系统会依按优先级依次尝试其它语言。如果对象不存在于任何一种站点语言中，对象的内容不会被显示，除非它总是可用（稍后解释）或您配置站点为“显示未翻译的内容”。

注意：

如果没有指定"SiteLanguageList“，系统会用"ContentObjectLocale"中的设置，因而只有默认语言可用。

例

假设您的翻译语言为英文，法文和挪威语。如果您指定它们中的两种语言为公共站点的站点语言（如：英文作为最优先语言和法文），系统会尝试用英文和法文显示对象内容，因而挪威语的内容不会被显示。

如果您用英文创建第一篇文章，法文创建第二篇，第三篇用挪威语，那只有第一，二篇文章会被显示。如果您把第三篇文章翻译为英文或法文，那翻译过的内容会得到显示而挪威语内容仍然不会被显示。如果文章有英文和法文的翻译，那英文的内容会被显示（因为英文是站点的优先语言）。

请参阅“配置站点语言”一章了解更多。

永远可用的对象

某些对象应该总是可用，尽管它们不存在于任何一种站点语言中。例如：无论在哪个站点入口内，系统都必须可以提取用户对象。因为如此，在对象级别出现了一个新的标记"总是可用"。这个标记可以用来单独控制不同对象的可用性。如果一个对象不存在于任何站点语言中但是它的这个标记被启用，那么系统仍然会使用对象的初始化/主语言来显示它。

默认的对象可用性可以在类级别控制。默认情况下，这一标记为“文件夹”，“用户”，“用户组”，“图片”，“文件”等类启用，因而在创建这些类的对象时，这个标记会被启用。在类级别修改这个标记不会影响已经存在的对象因为它只决定对象被保存时这个标记的默认值。

例

假设您有一个文件夹，它之存在于挪威语中，它被标记为“总是可用”且包含了若干文章（文章存在于英文，法文和挪威语且都没有被标记为“总是可用”）。如果英文和法文为公共站点的语言，文件夹仍然会被显示，因为它被标记为“总是可用”站点的用户因而可以访问它下面的文章。如果文件夹没有被标记为“总是可用”，则它不会被显示，因而网站用户也不能访问它下面的文章，除非您将文件夹翻译为英文或法文。

4.3.1 配置地区

eZ Publish 默认使用"eng-GB"区域。这是因为系统在"settings/site.ini"中的"[RegionalSettings]"下的"Locale"中默认使用"eng-GB"。如果您需要其它区域，您需要重设这个配置。注意，指定的区域会被"l10n"操作符用作默认区域，除非您显式给出区域参数。下例演示了如何配置站点区域。

例 1

假设您需要对所有站点入口使用"nor-NO"区域。参照以下步骤。

1. 编辑"settings/override/site.ini.append.php"
2. 添加如下配置

```
Locale=nor-NO
```

3. 清除缓存

系统会为所有站点入口使用"share/locale/nor-NO.ini"作为区域配置。

例 2

假设您需要为某个站点入口"example"使用"nor-NO"区域。则编辑"settings/siteaccess/example/site.ini.append.php"，并按照例 1 添加配置。但要确

保"settings/override/site.ini.append.php"中不包含区域设置。清除缓存后，"example"开始使用"nor-NO"区域。但是这并不会翻译站点入口界面的所有部分（如："Login"和"Sign up"链接/按钮等。）。要翻译界面，您需要在"[RegionalSettings]"内添加如下配置：

```
TextTranslation=enabled
```

上述配置会要求系统根据当前区域配置翻译模板中所有使用"i18n"操作符的字符串。这意味者您可以为管理站点入口设置挪威语区域并启用文本翻译，从而将管理界面翻译为挪威语。（"TextTranslation"默认被禁用。）

您也可以用相同方法为其余站点入口指定不同的区域，否则默认区域会被使用。

添加缺失区域

eZ Publish 在"share/locale"目录中提供了很多默认的区域 INI 文件。INI 文件由区域标识符命名。如果这里没有您需要的区域配置，参阅 http://ez.no/community/contribs/internationalization。下例演示了如何添加区域。

例

假设您需要使用"ell-GR"区域。您需要：从 http://ez.no/community/contribs/internationalization 下载希腊翻译文件并把文件解压到临时目录。您应该可以找到一个"share"子目录，它包含区域文件"share/locale/elle-GR.ini"和翻译文件"share/translations/ell-GR/translation.ts"。此外，下载包可能还包含一个国旗图标"share/icons/flags/elle-GR.gif"和/或用于某些扩展的翻译文件，在"extension"子目录中。

注意，"translation.ts"文件包含 eZ Publish 特有字符串的希腊文翻译（模板中和 PHP 代码中使用的字符串）。如果启用"TextTranslation"配置，这个文件中的字符串会被用来翻译界面，系统信息，警告等。

如果您把"share"目录复制到 eZ Publish 的安装目录中，设置"ell-GR"区域（如前例所述）并清除缓存，系统就会开始使用希腊区域。

自定义区域

除系统内建的区域，您也可以开发自定义的区域。下例演示了如何创建自定义区域。

例 1

假设您需要使用冰岛区域。您可以基于 eng-GB 的区域文件创建自定义的配置文件。参阅以下步骤：

1. 在"share/locale"目录中把"eng-GB.ini"复制为"ice-IS.ini"。
2. 编辑区域配置。
3. 设置站点区域为"ice-IS"。

例 2

假设您需要修改挪威区域。不要修改原始文件，因为升级时会被覆盖。相反，您应该基于原始文件创建

一个自定义的区域文件。参阅如下步骤。

1. 在"share/locale"目录复制"nor-NO.ini"到"nor-NO@custom.ini"。
2. 编辑这个区域文件
3. 确保站点使用"nor-NO@custom"区域
4. 清除缓存

4.3.2 配置语言

默认情况下，站点语言为空（如：下载并解压 eZ Publish 之后）。在安装过程中，安装向导允许用户选择站点语言。可用语言列表由"share/locale"目录中的 INI 文件构成。用单选按钮选择默认语言（必须），服选框选择附加语言（可选）。

注意，选择默认语言会同时影响默认语言与系统区域。注意，取决于浏览器的配置，某个单选按钮会被预先选择。但是，您可以选择其它语言。如果您例如：选择德语，那么默认语言与区域都会被设置为"ger-DE"且您的管理界面也会被翻译为德文界面（此外，"TextTranslation"会被启用）。

所有选中的语言会被添加到系统的翻译语言列表中且被作为公共和管理站点入口的站点语言的一种。默认语言会被作为最优先使用的站点语言。安装完成后，您可以用它们中的任何语言创建和翻译内容。您也可以在管理界面中或配置文件中修改站点语言。

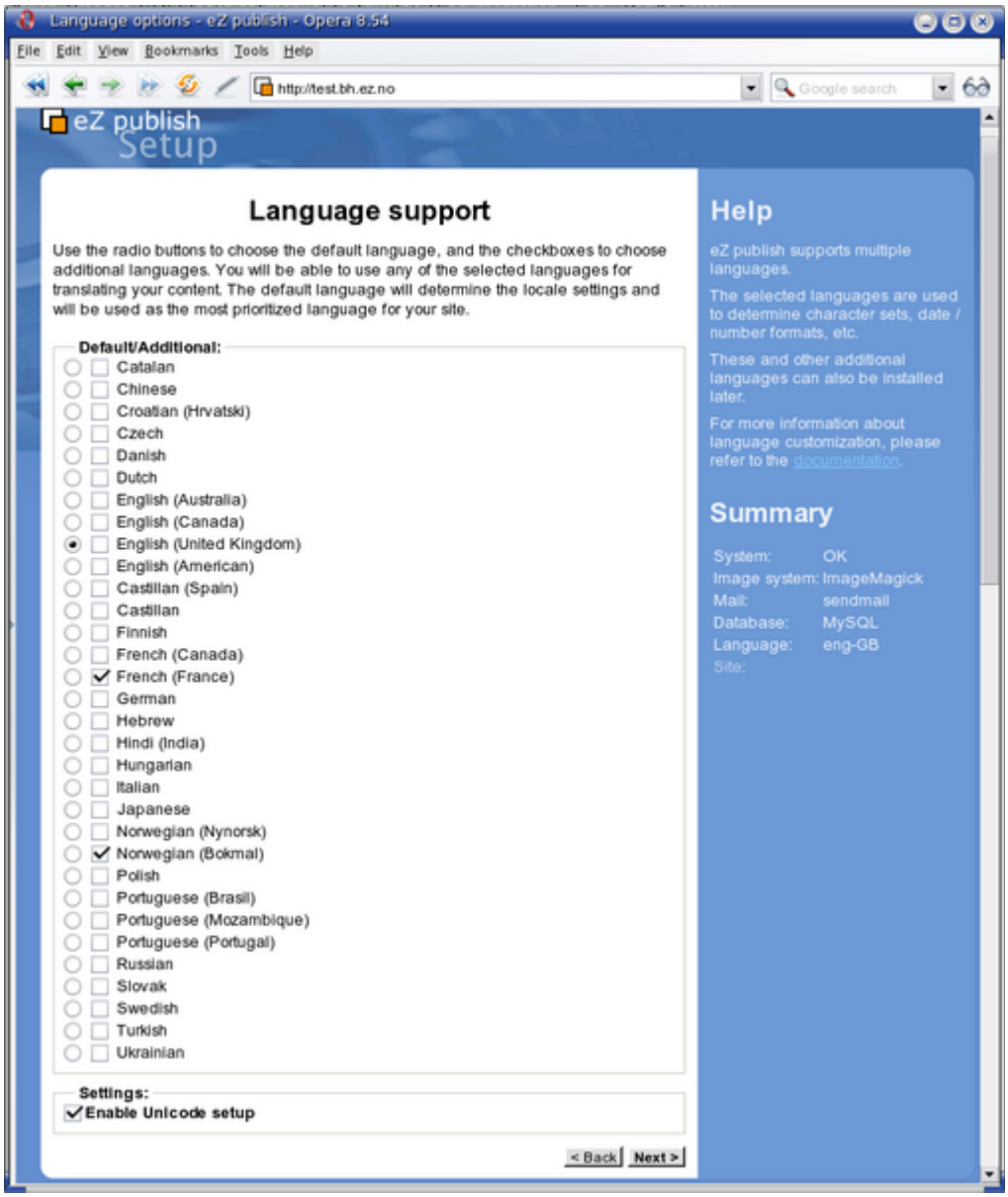
显示未翻译的内容

因为有时可能需要显示所有语言，系统提供了一个称为"ShowUntranslatedObjects"的配置选项。它可以被设置为"enabled"或"disabled"。如果启用这个选项，系统会根据站点语言的优先级显示内容，但是系统不会过滤不在站点语言列表中的内容。换言之，所有的内容都会被显示，无论存在于何种语言中而且存在于站点语言列表中的内容会被系统以优先语言显示。

"ShowUntranslatedObjects"选项默认被禁用。但是，安装向导通常会对管理界面启用这个选项。这允许站点管理员用任何翻译语言创建和编辑对象，尽管某些语言不在站点语言列表中。

例

假设您选择英式英文作为默认语言，法文与挪威语作为附加语言（参阅下图）。



这种情况下，安装结束后，您将会有以下的区域，默认语言和站点语言设置。

```
[RegionalSettings]
Locale=eng-GB
ContentObjectLocale=eng-GB
SiteLanguageList[]=eng-GB
SiteLanguageList[]=fre-FR
```

```
SiteLanguageList[]=nor-NO
```

这意味着站点区域被设置为"eng-GB"，默认语言为英文，最优先语言为英文，其次为法文和挪威语。安装向导会把这些配置写入公共和管理站点入口的"site.ini.append.php"中。对两个站点入口，"TextTranslation"选项都会被禁用，因为使用了"eng-GB"区域选项。

安装向导会在管理站点入口的"site.ini.append.php"中添加下面一行配置：

```
ShowUntranslatedObjects=enabled
```

这会告诉系统管理界面中可以使用所有翻译语言。

您可以用管理界面添加新的翻译语言。例如：在"设置－语言"部分添加德语。这种语言不会在公共站点显示因为它还不是站点语言（没有在"SiteLanguageList[]"数组中配置）。但是，清除缓存后你可以在管理界面中使用德语创建对象，因为管理站点入口的"ShowUntranslatedObjects"选项被启用。

改变语言优先级

站点入口的"site.ini.append.php"中的"SiteLanguageList[]"指定站点语言。语言在列表中的顺序揭示了它们的优先级。排在上面的语言有更高的优先级。系统首先会尝试用最优先的语言显示内容。如果对象不存在于这种语言中，系统会按优先级依次尝试其它语言。如果对象不存在于任何一种站点语言，它不会被显示，除非被标记为"总是可用"且站点入口的"显示未翻译内容"选项被启用。

要修改站点语言的优先级，编辑"site.ini.append.php"并重新排序"SiteLanguageList"数组中的元素。

例

假设您公共站点入口使用以下配置：

```
[RegionalSettings]
SiteLanguageList[]
SiteLanguageList[]=eng-GB
SiteLanguageList[]=fre-FR
SiteLanguageList[]=ger-DE
SiteLanguageList[]=nor-NO
```

如果一篇文章存在于法语和挪威语中，系统会根据站点语言的优先级用法语显示这篇文章。如果您把文章翻译为德语，这种行为也不会改变。但是，如果将文章翻译为英文，那么它会被显示为英文。

如果把"SiteLanguageList[]=nor-NO"移动到顶端，挪威语会成为最优先语言。这会要求系统优先显示挪威语内容并只有在挪威语内容不存在时才会用其它语言显示内容。

使用多个公共站点入口

在前一个例子中，只用到一个公共站点入口。一个多语言站点通常会使用几个公共站点入口。如果站点内容存在于例如：英文和法文，那么建议使用如下两个公共站点入口：

站点入口"gb"	站点入口"fr"
[RegionalSettings] SiteLanguageList[] SiteLanguageList[]=eng-GB	[RegionalSettings] SiteLanguageList[] SiteLanguageList[]=fre-FR SiteLanguageList[]=eng-GB

现在，假设您希望在站点中使用挪威语。在本例中，您可以添加挪威语作为翻译语言，创建一个新的站点入口"no"并在"site.ini.append.php"中作如下配置：

```
[RegionalSettings]

SiteLanguageList[]

SiteLanguageList[]=nor-NO
```

这会告诉系统，这个站点唯一的站点语言为挪威语。换言之，如果文章不存在于挪威语中，则不会被显示。

当然，也可以添加如下配置：

```
SiteLanguageList[]=eng-GB
```

在本例中，挪威语为"no"站点入口的最优先语言，英语为次优先语言。

	站点入口"gb"	站点入口"fr"	站点入口"no"
最优先语言	eng-GB	fre-FR	nor-NO
第二优先语言	-	eng-GB	eng-GB
第三优先语言	-	-	-

只存在于英语中的文章会在三个站点入口中显示为英语。如果文章之存在于挪威语，它只会在"no"站点入口中显示。

假设我们用法语创建一个新文章"Lundi"（法语星期一的意思）。这篇文章会在"fr"站点入口中显示，但是不会在"gb"和"no"站点入口中显示（因为法语不是另外两个站点的站点语言）。如果把这篇文章翻译为挪威文"Mandag"，那么这篇文章可以在"no"站点入口中显示，但是仍然不能在"gb"站点入口中显示。如果再将它翻译为英文"Monday"，它会在"gb"站点入口中显示，但是不会影响到"fr"和"no"站点入口，因为英文是它们的第二优先语言。

4.3.3 管理翻译语言

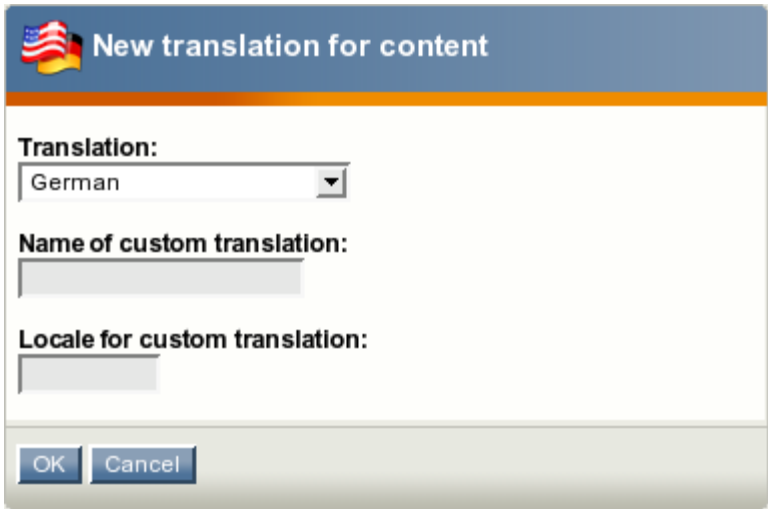
管理界面允许您为您的站点管理翻译语言。这可以通过管理全局翻译语言列表来达到。要访问翻译语言列表，在管理界面中点击“设置”标签，然后点击左侧的“语言”链接（也可以通过"/content/translations"直接访问）下图演示了这个列表。

列表的最后一列包含翻译数字信息（例如：多少对象被翻译成这种语言）。上图显示所有对象只存在于英文中，但还没有被翻译成法语和挪威语。如果点击语言名，系统会显示关于这种语言和区域的配置。

下一章解释了如何添加和/或删除翻译语言。

添加新语言

您可以通过点击“添加语言”按钮，从“翻译”下拉框中选择需要的语言来达到。注意，这个下拉框中的内容取决于"share/locale"中 INI 文件的内容。如果您希望使用的语言不在这里，您需要首先添加缺失区域。



点击“确定”按钮保存您的修改。清除缓存后，您将可以为您的内容使用这种语言。

删除语言

只有在没有任何对象使用某个语言时，您才能删除这种语言（当“翻译”这一列为"0"）。

要从系统中删除一种或多种语言，勾选第一列中的复选框并点击“删除所选”按钮。

4.3.4 可翻译的类属性

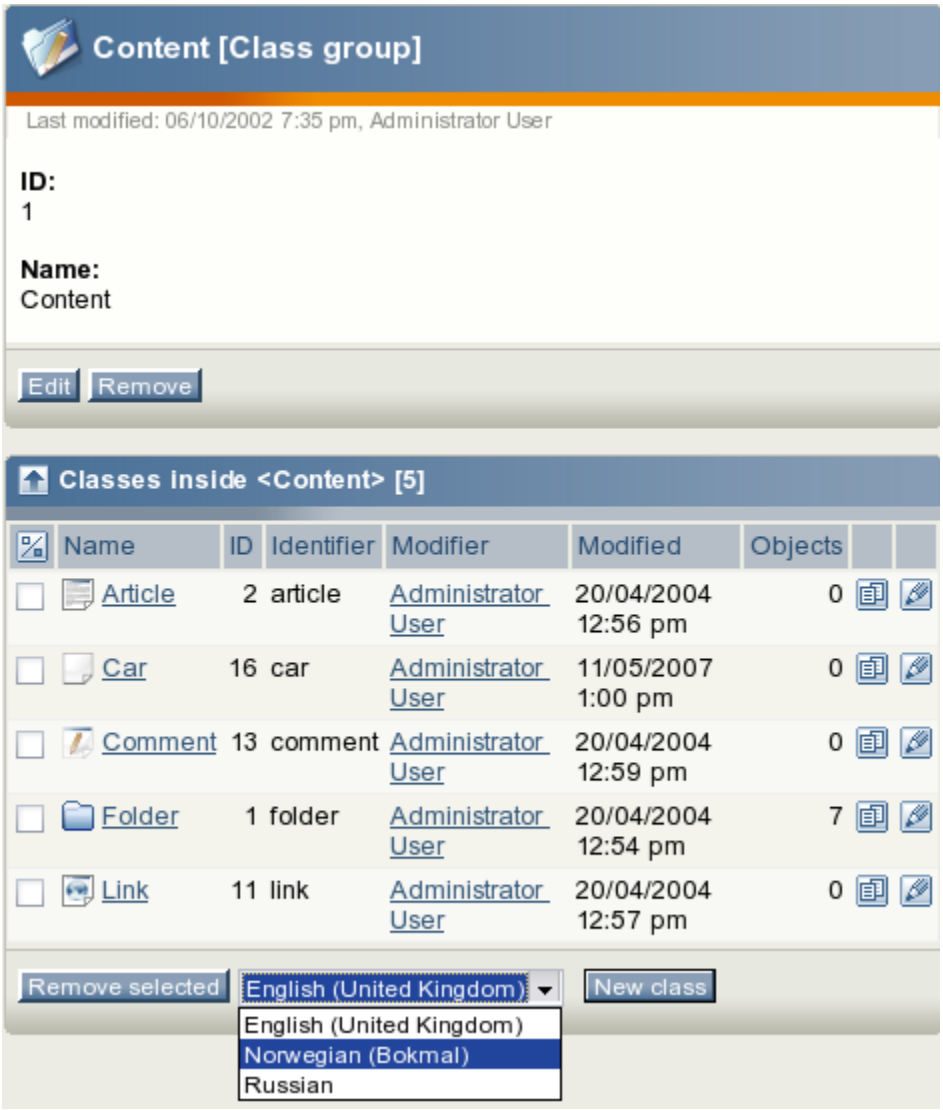
从 3.9.0 版本开始，编辑类的时候可以翻译类属性名。用户用不同语言编辑对象时，类属性标签会被用对应的翻译显示。例如：如果一个类用来保存英文和挪威语的汽车信息，类属性可以被翻译从而“颜色”属性会被显示为"Color"或"Farge"。

可以用任何被安装的语言（通过安装向导或管理界面安装）创建类。创建类时所使用的语言会被设置为类的主语言。类主语言的类名和类属性名不能被删除。但是，如果类存在于多种语言，您可以修改类的主语言，因而非主语言的类名和类属性名可以被删除。可以在类视图的“翻译”窗口修改主语言或删除类的翻译。

用不同语言创建类

管理界面允许您用不同的翻译语言创建内容类。参阅如下步骤。

1. 在管理界面中访问“设置”标签，点击左侧的“类”再选择希望创建类的类组。您应该可以看到若干已经存在于这个组中的类。
2. 从页面底端的下拉框中选择用于创建类的语言。点击“新建类”按钮（参阅下图）。



如果希望使用的语言不在下拉框中，确认它在全局翻译列表中。参阅“管理翻译语言”章节了解如何添加语言到全局翻译列表。注意，新加的语言在清除缓存后才能生效。

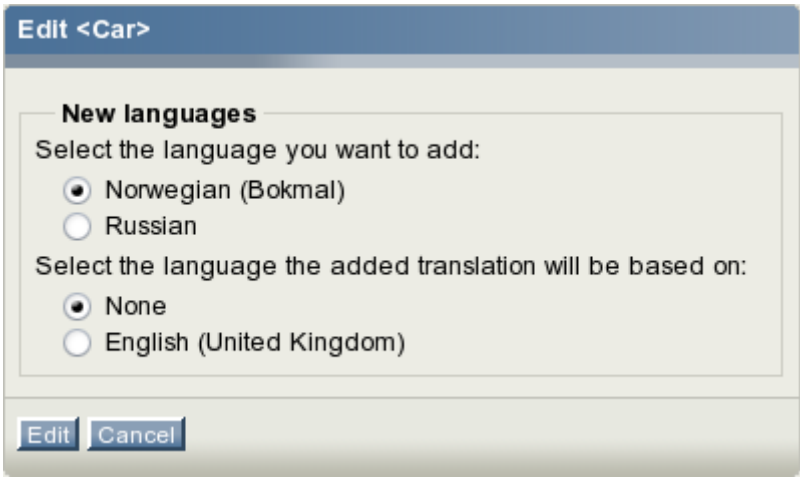
3. 系统会显示类编辑界面，右上角会显示类的主语言。指定类名，类标识符，对象名模式和容器标记，然后用页面底端的下拉框添加希望使用的类属性。

将类翻译为多种语言

管理界面允许您将内容类的类名和属性名翻译成任何一种可用的翻译语言。参阅如下步骤。

1. 在管理界面中，点击希望编辑的类的类名，会显示类的显示界面。
2. 从下拉框中选择“另一种语言”并点击“编辑”按钮。如下图。

系统会显示语言选择页面（如下图）。



用单选按钮选择语言（在上图挪威语被选中），您还可以选择翻译源。您可以选择"None"或者其它现存的语言作为翻译源。如果您选择了某个现存语言作为翻译源，系统会从选定的语言中复制类名和属性名，您可以随后修改它们（否则，您必须手动输入所有内容）。

3. 点击“编辑”按钮后，系统会显示类的编辑界面，你可以在这个界面中编辑类名和类属性名。结束后，点击“确认”按钮保存修改。

用不同语言编辑类

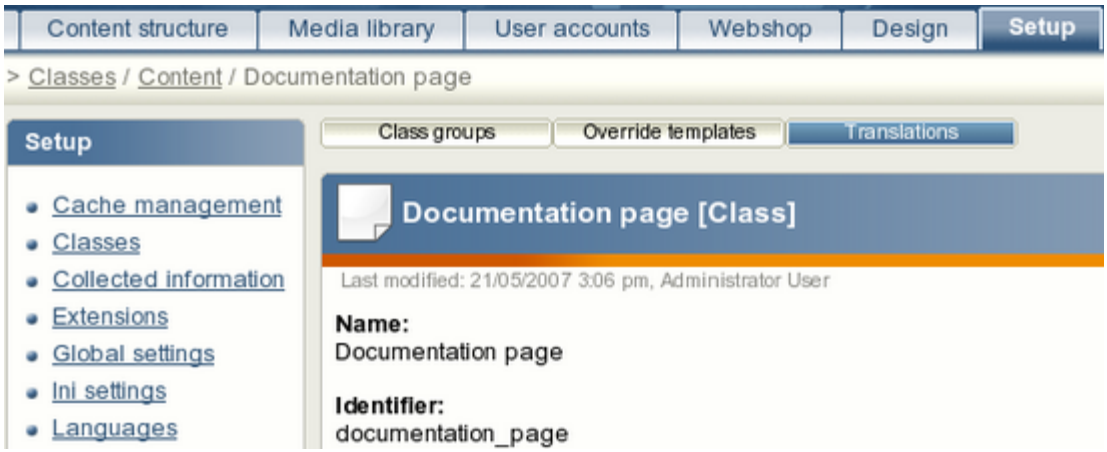
您可以用任何现存语言编辑类。参阅如下步骤。

使用“编辑”按钮

1. 在管理界面中，找到要编辑的类，点击类名，系统会显示类查看视图。
2. 从下拉框中选择语言然后点击“编辑”按钮。系统会显示类编辑界面，您可以在这里修改类名和属性名。修改结束后，点击“确定”保存修改。

用翻译窗口

1. 在管理界面中，找到要编辑的类，点击类名，系统会显示类查看视图。
2. 页面顶端的水平开关可以控制不同窗口的可见性。点击“翻译”开关显示翻译窗口（参阅下图）



开关的蓝色背景表示开关被打开，因而“翻译”窗口被启用。下图演示了这个窗口的外观（在本例中有三种现存语言）。

找到需要编辑的语言，然后点击语言右侧的编辑图标。系统会显示类编辑界面。

修改主语言

如果类存在于多种语言中，那么您可以选择其中任何一种语言作为主语言。

1. 在管理界面中，找到要编辑的类，点击类名，系统会显示类查看视图。
2. 启用翻译窗口，用单选框选择新的主语言，点击“设置主语言”按钮。

删除语言

可以从类中删除语言（除了主语言外）。可以在翻译窗口中完成。用复选框勾选要删除的语言，然后点击“删除所选项”按钮。

4.3.5 可翻译的国家名

从 3.9 版本开始，可以把国家名翻译成不同语言。例如：当使用挪威区域时，您可以要求系统用"Frankrike"代替"France"，用"Norge"代替"Norway"显示国家名。下例演示了如何做到。

例

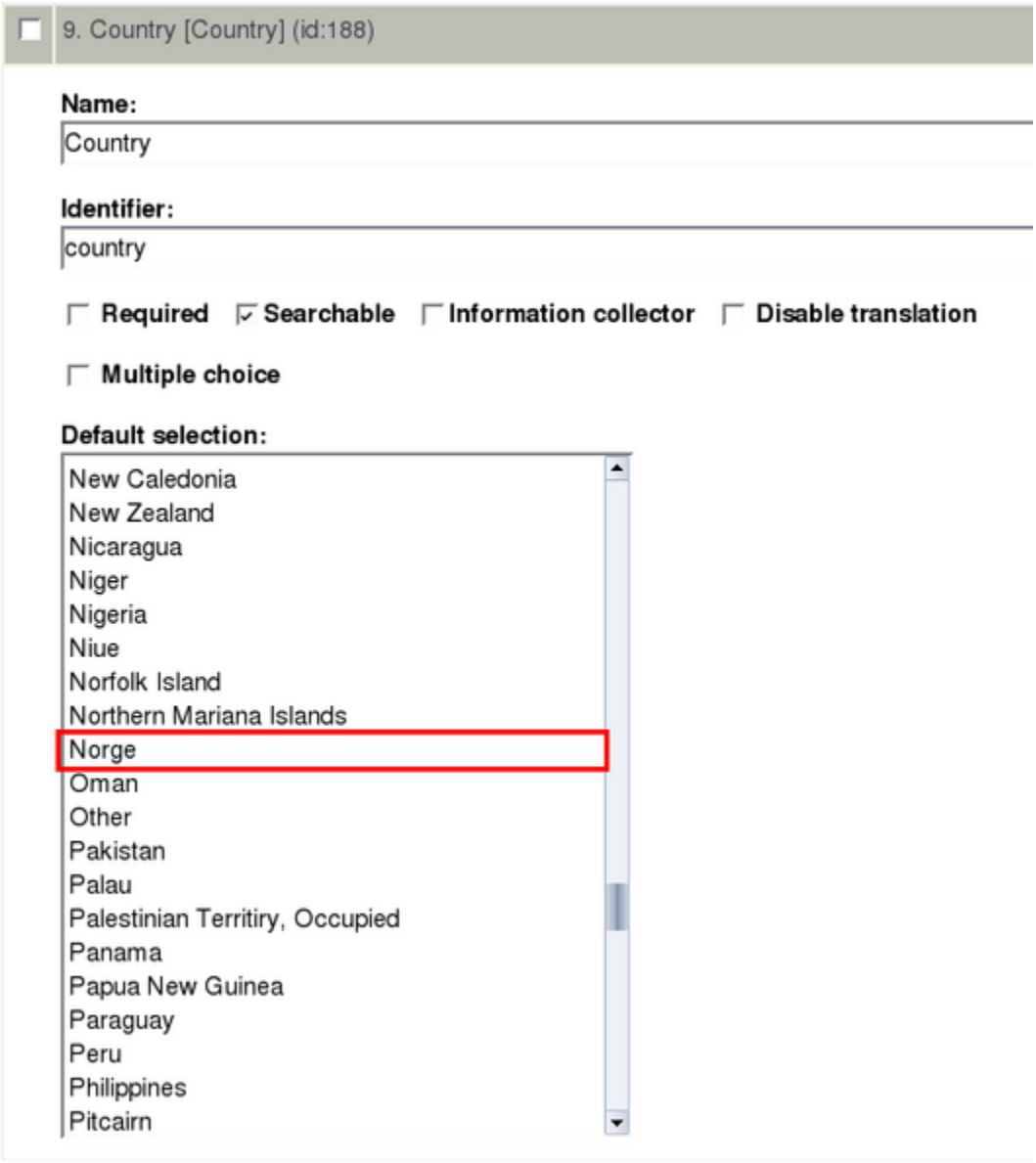
如果您希望在挪威区域中用挪威语翻译“法国”和“挪威”，参阅如下步骤。

1. 在"share/locale"目录中复制"nor-NO.ini"到"nor-NO@custom.ini"。
2. 在"[RegionalSettings]"下添加如下配置：

```
[CountryNames]
Countries[]
Countries[FR]=Frankrike
Countries[NO]=Norge
```

3. 设置站点区域为"nor-NO@custom"（参阅“配置站点区域”文档了解更多）

清除系统缓存后，系统会用挪威语显示法国和挪威国名。下图显示了翻译后的国名在编辑"ezcountry"数据类型时的显示效果。



4.3.6 多语言对象

以下步骤描述了如何创建多语言对象，使对象总是可用，设置对象的初始化/主语言等等。

创建新对象

管理界面允许您用任何语言创建对象。在“在此创建”界面中从语言下拉框中选择语言作为对象的初始化/主语言然后点击“在此创建”按钮（参阅下图）。

如果需要的语言不在下拉框中，参阅如下步骤：

1. 在翻译语言列表页面添加需要使用的语言。如果仍然没有所需要的语言，参阅“管理翻译语言”章节
2. 确保管理界面站点入口的"site.ini.append.php"中包含如下配置：

```
ShowUntranslatedObjects=enabled
```

改变初始化/主语言

如果对象存在于多种语言，您可以选择其中一种作为初始化/主语言。在“翻译窗口”中用单选按钮选择语言然后点击“设置为主语言”按钮。

改变对象可用性

要令对象总是可用，在对象查看视图中的翻译窗口勾选“如果没有优先翻译则使用主语言”复选框然后点击“更新”按钮。

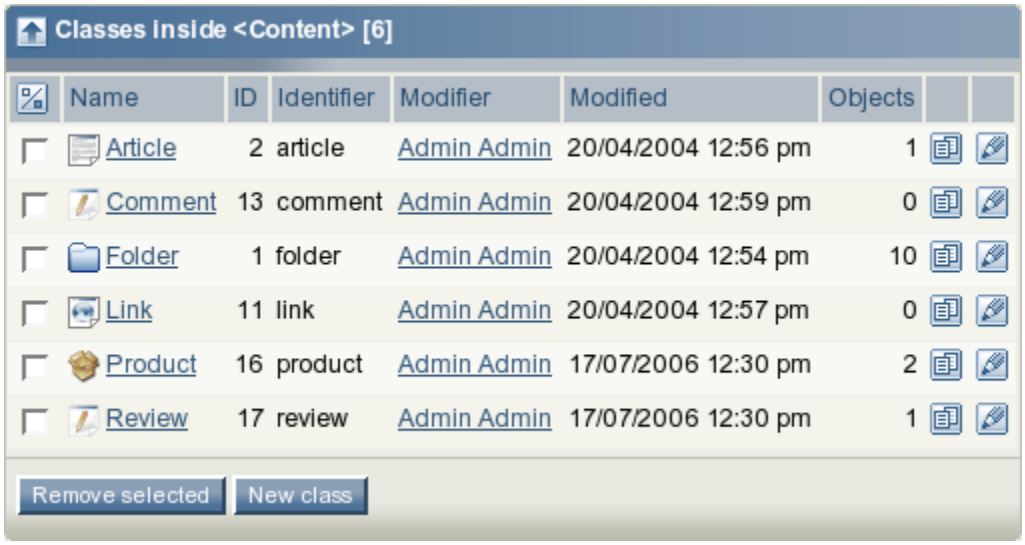
类的默认对象可用性

可以在类级别设置默认对象可用性。默认情况下，可用性对“文件夹”，“用户”，“用户组”，“图片”，“文件”等类启用，因而这些类的新对象会被默认设置为“总是可用”。注意，无论类的设置如何，这一属性都可以为每个对象单独重新配置。下例演示了如何做到。

例

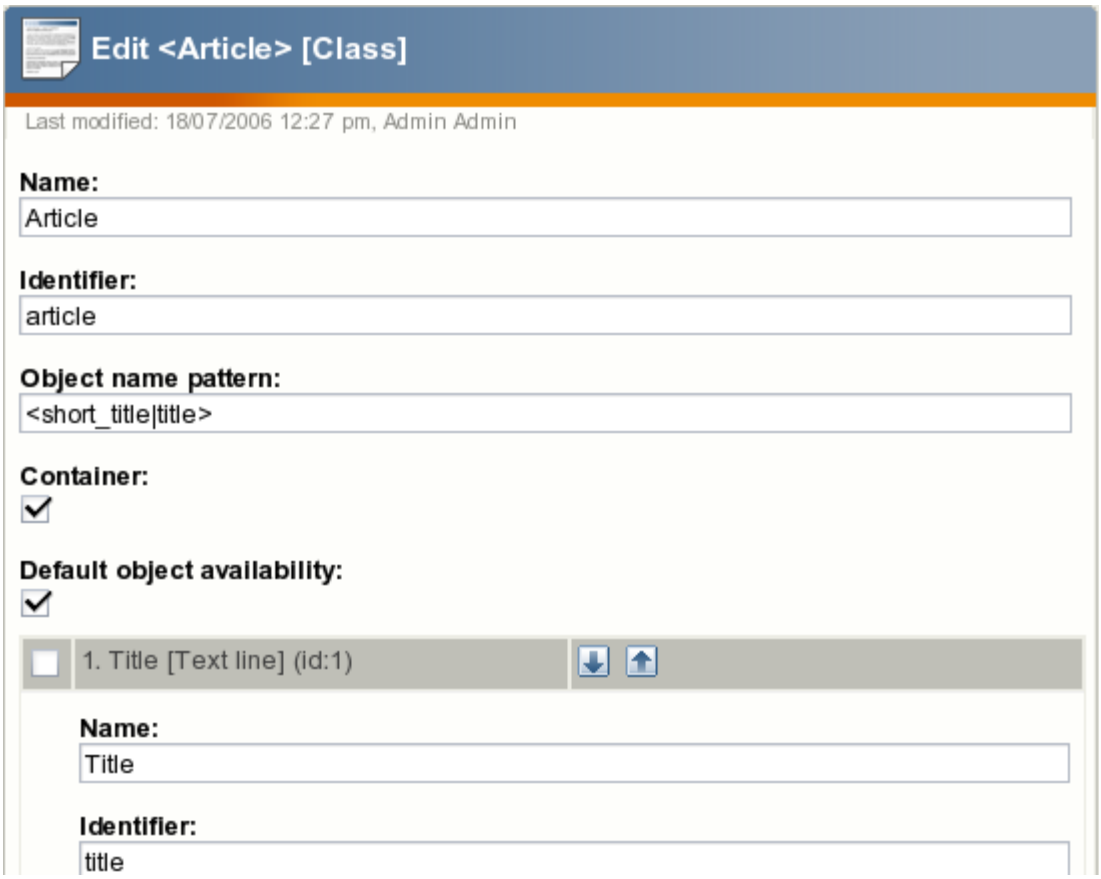
假设您需要创建一些英文文章，但是需要它们在任何站点入口都可以显示，无论站点入口使用何种语言。您可以对“文章”类启用“总是可用”属性，从而每个新创建的文章默认都成为“总是可用”的对象。下图揭示了如何做到：

1. 访问管理界面中的“设置”标签，点击左侧的“类”并选择"Content"类组。您应该会看到所有指派到这个组中的类。如下图。



找到“文章”类，然后点击类右侧的编辑图标。系统会显示类编辑界面。

2. 勾选“默认对象可用性”复选框然后点击“确定”按钮保存修改



注意：

这一修改不会影响已经存在的对象。只有新文章会受到影响。

4.3.7 使用翻译

您可以用翻译窗口查看对象存在的语言。以下步骤揭示了您可以如何创建，编辑和删除对象翻译。

编辑翻译

所有内容的编辑都是通过对象编辑界面完成的。在编辑或创建对象时，这个界面会自动显示。如果一个对象存在于多种语言，您可以选择编辑的语言。以下步骤揭示了您可以如何用不同方法编辑现存翻译。

用翻译窗口

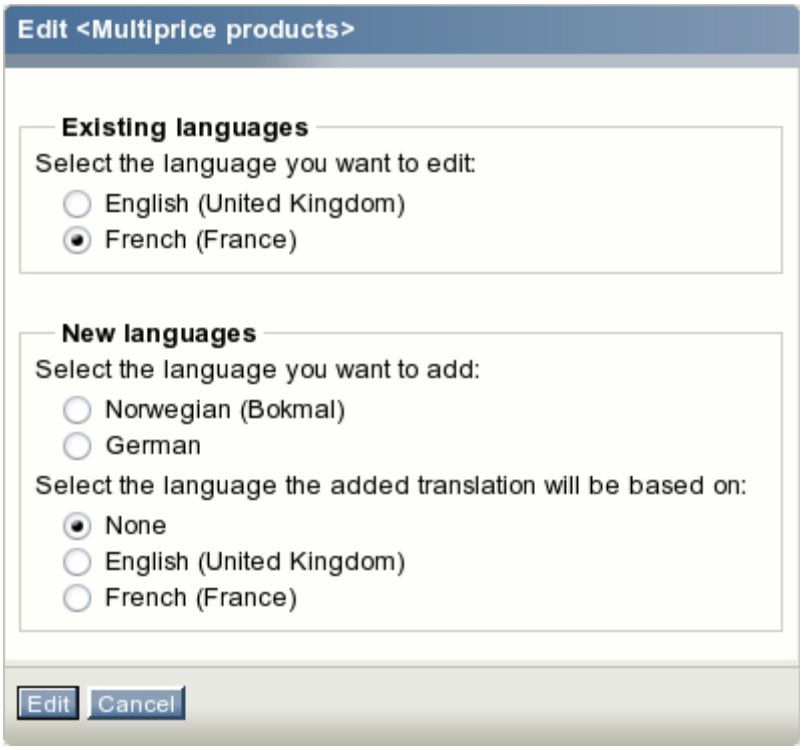
1. 找到要编辑的对象，换言之，确保对象正在被查看。
2. 启用对象的翻译窗口。点击语言右侧的编辑按钮。系统会显示编辑界面。

使用“子项目”窗口

1. 找到对象的父节点。换言之，确保对象的父节点在被查看。
2. 在“子项目”窗口中找到要编辑的对象。点击编辑图标。系统会显示语言选择界面。
3. 在“现存语言”框架中用单选按钮选择要编辑的语言然后点击“编辑”按钮。系统会显示编辑界面

语言选择界面

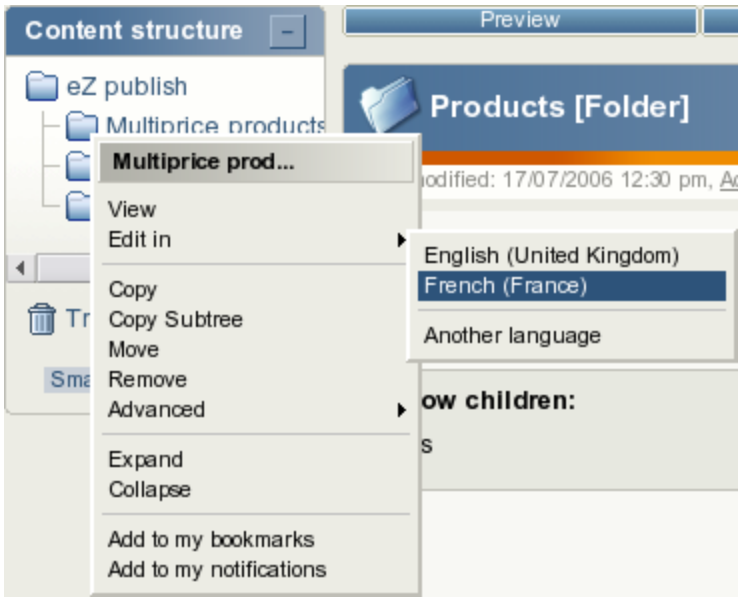
语言选择界面（全视图或精简视图）在你需要选择编辑/创建语言时显示。下图演示了一个存在于英文和法文的文件夹的语言选择界面。



如您所见，语言选择单选按钮被分为两组。“现存语言”组包含了对象已经存在的语言。这个列表允许您选杂编辑已经存在的语言。“新语言”组包含没有被用过的语言。后者允许将对象翻译成还不存在的语言。添加新翻译的时候，您可以选择翻译源。如果选择"None"以外的某种语言，编辑界面会包含翻译界面，而不是显示为标准编辑界面。

用内容树和上下文菜单

1. 通过左侧的内容树定位需要编辑的对象
2. 点击对象图标（鼠标左键）来激活上下文菜单
3. 访问“编辑”子菜单，然后选择要编辑的语言。如下图。



上图演示了一个存在于英文和法文的文件夹的上下文菜单。选择语言后，系统会显示编辑界面。

使用“编辑”按钮

1. 在管理界面中定位要编辑的对象。换言之，确保对象在被查看。
2. 在预览窗口中从下拉框中选择需要编辑的语言然后点击“编辑”按钮（参阅下图）。

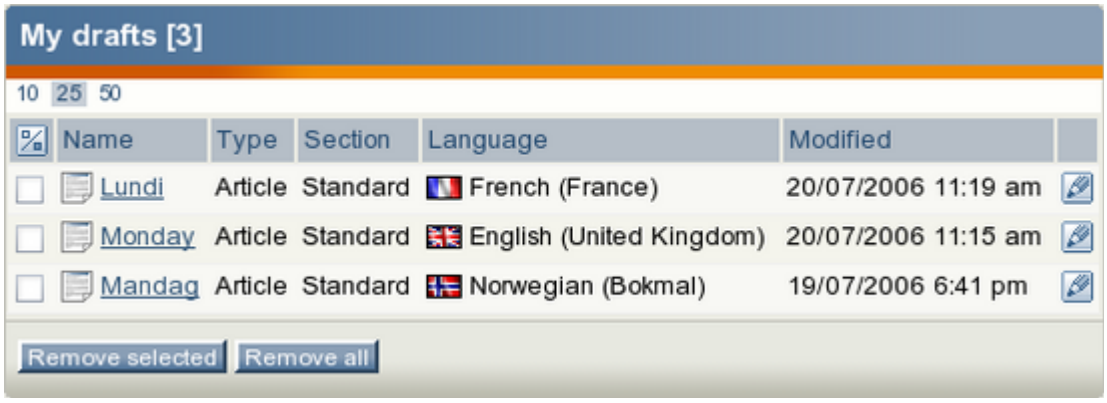
系统会显示编辑界面。

使用书签

1. 确保您的书签正被显示（用"+"符号打开书签窗口）。
2. 定位需要编辑的对象然后点击它的图标激活上下文菜单。
3. 访问“在...编辑”子菜单然后选择需要编辑的语言。系统会显示编辑界面。

编辑多语言

系统其实可以允许编辑同一个对象的多个版本。一个草稿只包含一种语言的对象属性数据。当草稿被发布时，系统复制早期版本中的翻译到发布版本中。下图演示了当用户编辑一个有三种语言的文章时版本窗口（这个界面可以通过点击“我的帐号”标签里左侧的“我的草稿”来显示）的外观。



同一个对象的不同翻译可以被分别创建和编辑并被不同编辑并行修改（一个用户同一时间只可以编辑同一个对象的一个版本和一个翻译）。

添加新翻译

您可以在管理界面中将对象翻译为任何翻译语言。以下步骤揭示了如何用不同方法翻译对象。

使用“子项目”窗口

1. 在管理界面中定位到对象的父节点。
2. 查看“子项目”窗口然后找到要编辑的对象。点击对象的编辑按钮。语言选择界面会被显示。在“新语言”中选择语言，然后点击“编辑”按钮。系统会显示编辑界面。

使用节点树和上下文菜单

1. 在左侧节点树中定位要编辑的对象
2. 点击对象图标激活上下文菜单
3. 访问“编辑”子菜单然后选择“其他语言”。系统会显示精简版语言选择界面。它包含对象不存在的一组语言（参与下图）和一组翻译源。

选择需要编辑的语言。同时可以选择一种现存语言作为翻译源。点击“编辑”按钮后，系统会显示编辑界面。

使用“编辑”按钮

1. 在管理界面中定位需要编辑的对象。
2. 在预览窗口中从下拉框中选择“另一种语言”然后点击“编辑”按钮。系统会显示语言选择界面的精简版。选择需要的语言然后点击“编辑”按钮。系统会显示编辑界面。

使用书签

1. 确保您的书签被显示（使用"+"标志打开窗口）。
2. 定位需要编辑的对象然后点击它的图标激活上下文菜单。
3. 访问“在...编辑”子菜单然后选择“另一种语言”。系统会显示语言选择界面的精简版。选择需要的参数然后点击“编辑”按钮。系统会显示编辑界面。

4.3.8 数位算法

以下文字揭示了一些与数位算法相关的技术细节。这些算法被用于语言过滤与排序。

系统在"ezcontent_language"数据库表中保存了所有翻译语言的信息。语言由 2 的指数标识。例如：他们的 ID 为 2,4,6,8,16,32 等等。(2^0)被用来“总是可用的”对象。当对象被标记为“总是可用”，即使对象的语言不在站点语言中（"SiteLanguageList"数组），对象仍然会被显示。

"always_available"字段

"ezcontentclass"数据库表中有一个"always_available"字段（默认为 0），它被用来控制是否将这个类的新

对象设置为“总是可用”。如果某个类的这个字段的值为"1"，则所有这个类的新对象都会被标记为“总是可用”。注意，之后这个标记可以在对象级别修改。类中的设置只会影响新对象的"总是可用”标记的初始值。

"language_mask"字段

在"ezcontentobject"数据库表中保存关于对象的信息时，系统用一个特殊的数位类型的字段"language_mask"来对象最后被发布版本的语言。这个字段包含了对象所有语言 ID 的和再加上"1"，如果对象是永远可用的。当创建新对象时，则为初始化语言的 ID 加上默认的"alwasy_available"的值（在类中指定）。

每次对象的语言配置被修改时，它的"language_mask"都会被更新。通常在添加或删除翻译的时候。

例

假设您有两种翻译如下：

语言名	ID
英文（英国）	2
法文（法国）	4

这允许您的内容对象使用以下可能的"language_mask"：

Language mask	Bitmap	Languages
2	00010	对象存在于英文。
3	00011	英文，总是可用。
4	00100	法文。
5	00101	法文，总是可用。
6	00110	英文和法文。
7	00111	英文，法文，总是可用。

在"ezcontentobject_version"数据库表中保存对象版本的信息时，"language_mask"字段包含语言 ID 的和。加上"alwasy_available"的值。

"initial_language_id"字段

"ezcontentobject"数据库表中有一个"initial_language_id"字段，它用来保存对象的初始化语言 ID。

当在"ezcontentobject_version"数据库表中保存关于对象版本的信息时，系统记录将当前版本的语言 ID 保存在"initial_language_id"字段。

"language_id"字段

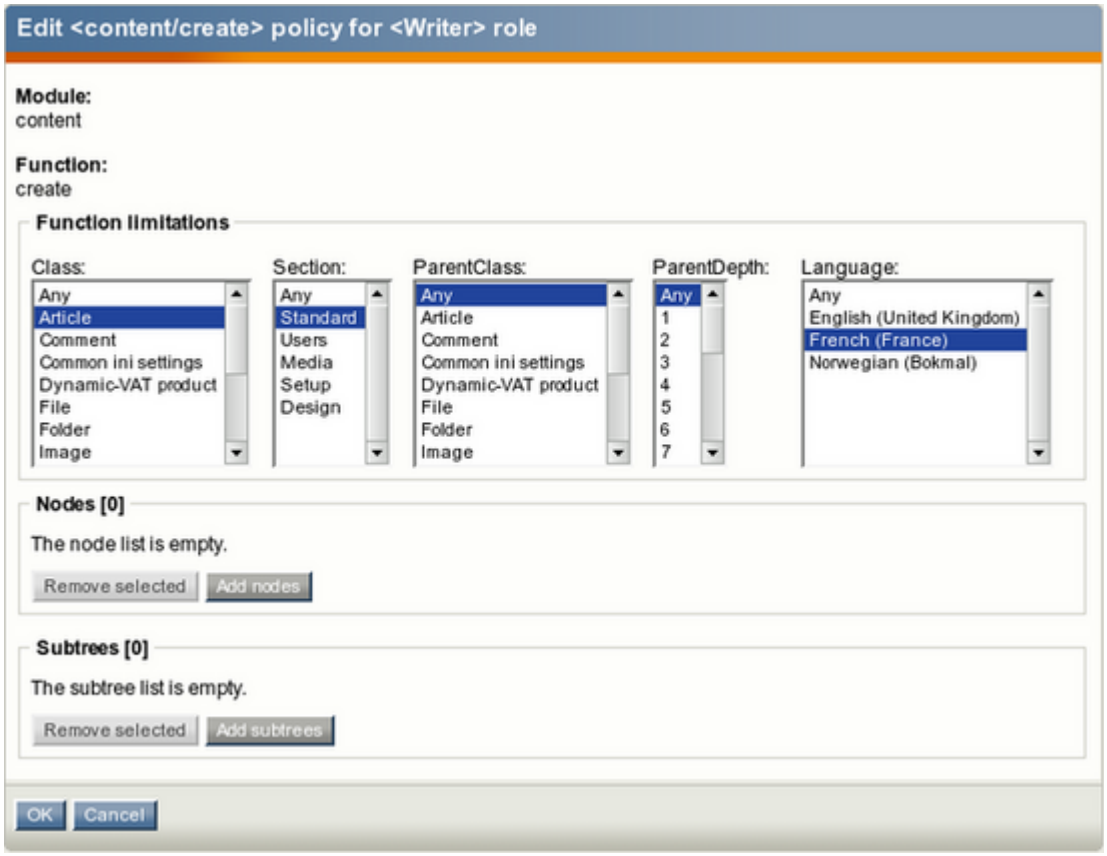
当在"ezcontentobject_attribute“数据库表中保存对象属性的信息时，系统在"language_code"字段中保存语言代码（例如："eng-GB"），而"language_id"则用来保存语言 ID。

4.3.9 基于语言的权限

"content"模块的"create“和"edit"函数支持语言级别的权限。例如，可以配置系统来允许一组用户用英文和挪威文创建和翻译对象，而另一组用户之被允许将现存的内容翻译成法文。"read"函数支持语言级别的权限，因而对象的所有翻译都可以被有"read"权限的用户访问。

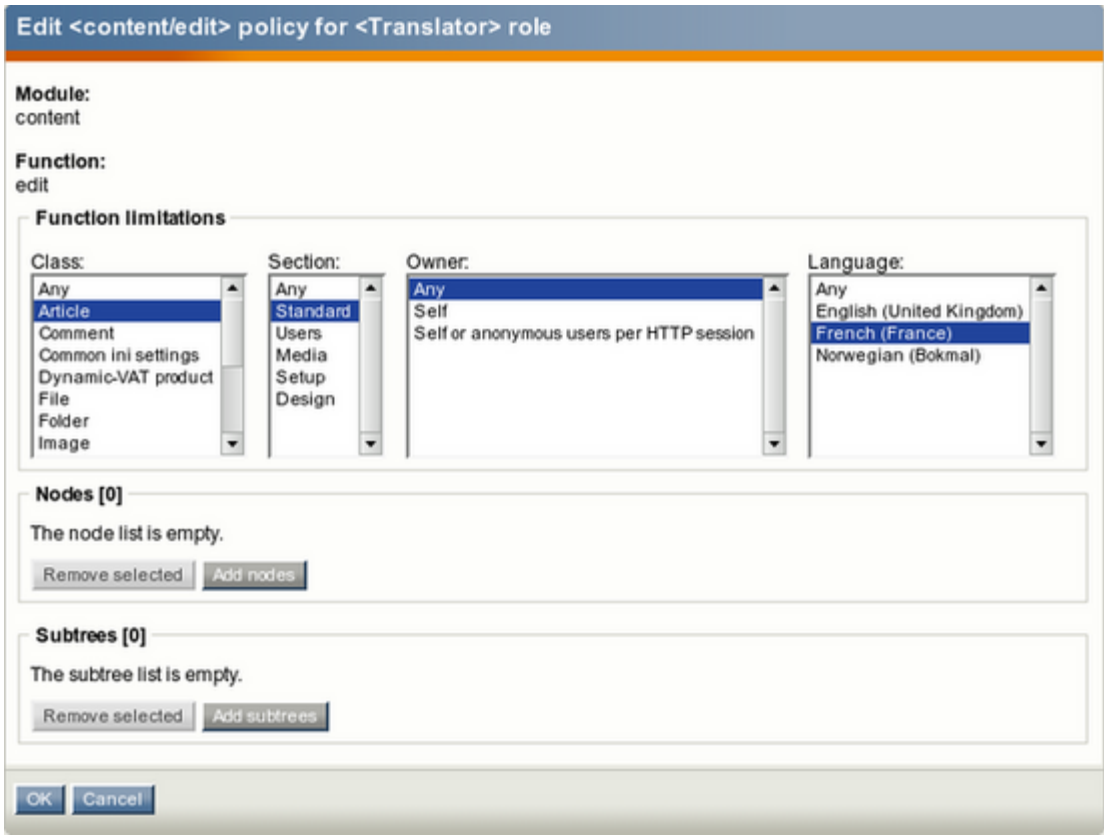
content/create

"create"函数的“语言”限制控制当对象被创建时，哪些语言可以被使用。下图演示了策略编辑界面。本例中，只允许在某些分区中创建法语文章。

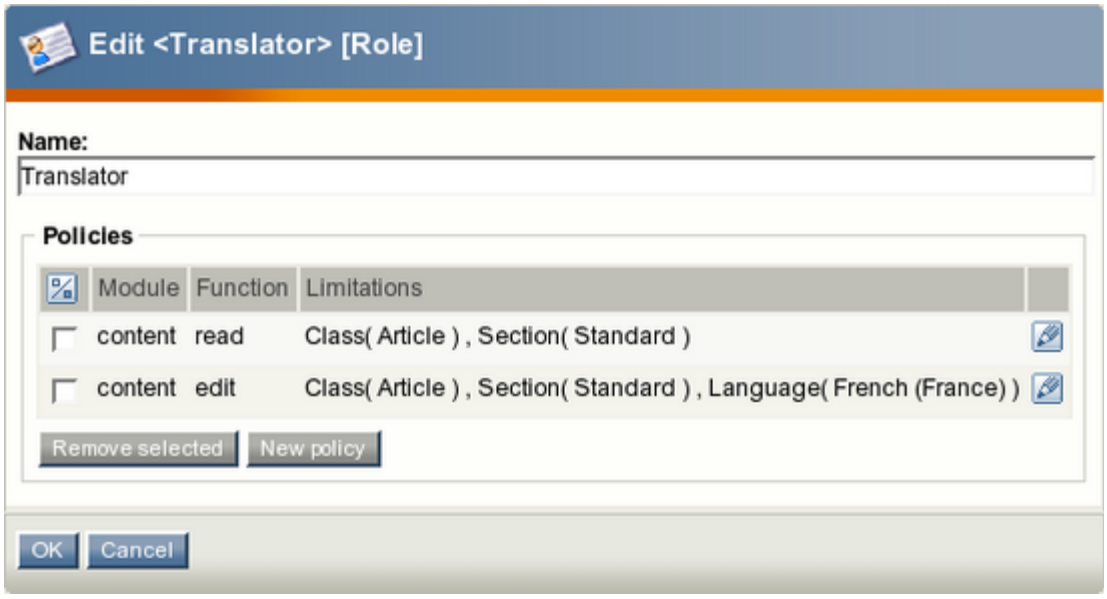


content/edit

"edit"函数的“语言”限制控制可以用哪几种语言编辑对象。它也控制哪些翻译可以被添加到对象中。下图演示了一个策略编辑界面。本例中，只允许编辑法语内容（文章）或添加法语的翻译到现存文章中。



与"content/read"函数组合（不支持语言限制），本例中的策略允许任何文章被从任何语言翻译成法语。下图演示了这种组合。注意，只负责翻译的用户不需要访问"create"函数。



4.4 多语言 URL 别名

在 eZ Publish 3.10，系统引入了一种新特性，它允许使用多语言虚拟 URL（也被称为友好 URL 或 URL 别名）。这种特性允许 URL 别名存在于多种语言中。

自动生成的别名

从 3.10 版本开始，自动生成的虚拟 URL 机制允许 URL 别名存在于多种语言，当然这取决于对象实际使用的语言。换言之，节点的 URL 别名现在由节点封装的对象所存在的翻译决定。当为对象添加新翻译时，系统会为这种翻译自动生成一套新的 URL 别名。

类编辑界面中多了一个称为“URL 别名模式”的字段。它控制对象被保存时，节点的虚拟 URL 如何被生成。

不能在管理界面中创建，编辑或删除自动生成的 URL 别名。对象被改变时，它们会被系统自动更新。唯一修改自动生成的 URL 别名的方法就是用对应语言编辑对象本身。

手动/用户定义的 URL 别名

以下两种 URL 别名可以在管理界面中管理：

- 全局 URL 别名
- 节点 URL 别名

全局 URL 别名列表包含了所有用户定义的虚拟 URL，但不包括那些指向最终目标（系统 URL）的 URL 别名如"content/view/full/node_id"。这些被称为节点 URL 别名且可以为节点单独管理。

全局别名总是以站点根目录开始，节点别名却可是从根或它的父节点开始。这是通过“相对于父节点”标记控制的。

基于通配符的 URL 转发

eZ Publish 支持基于通配符的 URL 转发。这意味着您可以包含通配符(*)的 URL 别名，系统会根据目标 URL 自动替换别名中的通配符。例如，您可以创建类似"pictures/*/*"的通配符 URL 别名，然后指定"media/images/{1}/{2}"为目标。在本例中，类似于"http://www.example.com/pictures/home/photo/"的 URL 会载入"http://www.example.com/media/images/home/photo/"。换言之，在访问内容时，您可以在 URL 中用"pictures"代替"media/images"来访问位于"media/images"下两层以下的内容。

可以选择别名为“直接/direct ”还是“转发/forward ”类型。在上例中，直接别名意味着当访问"http://www.example.com/pictures/home/photo/"时，输入的 URL 会保留在浏览器的地址栏而真正的节点内容会直接显示在浏览器中。如果别名为“转发”类型，系统会被重定向到"http://www.example.com/media/images/home/photo/"。

通配符 URL 别名可以通过管理界面管理。

可用性

只有别名的语言匹配当前站点入口的某个站点语言，别名才可用。如果站点入口被配置为“显示未翻译的内容”，则任何语言的别名都可用。

总是可用的别名

某些全局的别名需要总是可用而不论站点使用何种语言。因此，系统为全局别名引入了一个新的属性“包含在其他语言”。这一属性允许单独控制不同别名的可用性。

语言

多语言 URL 别名不控制被请求页面的显示语言。当访问某个节点的虚拟 URL 时，系统会根据站点的语言配置确定正确的语言（参阅下例）。

例

如果您创建一篇称为"Company"的文章并把它翻译成法语，则会有两个自动生成的 URL 别名："Company"和"Compagnie"。

假设您有两个公共站点入口"gb"和"fr"，配置如下：

"gb"站点入口	"fr"站点入口
[RegionalSettings] SiteLanguageList[] SiteLanguageList[]=eng-GB SiteLanguageList[]=fre-FR	[RegionalSettings] SiteLanguageList[] SiteLanguageList[]=fre-FR SiteLanguageList[]=eng-GB

如上表所示，"gb"站点入口被配置为使用英文作为优先语言，法文作为第二语言。这以为着"Company"和"Compagine"两个别名都可用。当输入如下 URL 时，系统会显示文章的英文内容：

- http://www.example.com/gb/Company
- http://www.example.com/gb/Compagnie

注意，如果您只为"gb"站点入口配置英文，则法文别名不可用。

"fr"站点入口的优先语言为法文，英文为第二语言。所以，两种 URL 别名都有效且下面的 URL 会要求系统显示文章的法文内容：

- http://www.example.com/fr/Company
- http://www.example.com/fr/Compagnie

字符转换

多语言 URL 别名机制支持三种字符转换的类型/方法。可以在"site.ini"的重设文件中的"[URLTranslator]"下的"TransformationGroup"配置。下表揭示了可用的转换方法。

名称	描述
urlalias_compat	这种方法支持在 URL 中使用小写拉丁字符"a"到"z"，数字和下划线。它提供了与 eZ Publish3.9.x 之前版本的行为。大写字符不被支持。
urlalias	这种方法支持更多字符，但是 URL 字符仍然被限定在 ASCII 表（除了少数例外）内。大写字符保持不变。
urlalias_iri	这种方法允许在 URL 中使用所有 Unicode 字符（除了少数例外）。它尽可能多地保留原始文字，这会生成对用户更友好的 URL。多个连续空格会被转换为一个。大写字符被保留。这是建议使用的方法。

如果您使用"urlalias_iri"转换类型，注意某些浏览器会用百分号"%"对 Unicode 字符进行编码。例如：如果用户输入"http://www.example.no/Ostehøvel"，它可能会被浏览器自动转换为"http://www.example.no/Osteh%C3%B8vel"。但是这不会影响内容的正常显示。在火狐浏览器中可以禁用这种行为（在地址栏中输入"about:config"并编辑"network.standard-url.escape-utf8"属性）。

参阅下例了解多语言 URL 如何工作。

例

假设有如下站点结构：

- Company（节点 ID：10）
  - *About us（节点 ID：11）*
  - *Contacts （节点 ID：12）*

如果节点 10（"Company"）被翻译成法语，它会得到第二个 URL 别名"Compagnie"。站点结构如下：

- Company | Compagnie（节点 ID：10）
  - *About us（节点 ID：11）*
  - *Contacts （节点 ID：12）*

这时，如果站点入口支持英文和法文，节点 10 可以通过两个 URL 别名访问。如果法文为优先语言，两个 URL 别名都会载入法文页面。

"About"页面（节点 11）可以通过"Company/About"或"Compagnie/About"访问。"Company/About"别名会在任何支持英文的站点入口可用。"Compagnie/About"别名只在支持英文和法文的站点入口可用。两种情况下，只有英文内容会被显示（因为对象之存在于英文中）。如果您编辑"About"页面并启用“总是可用”标记，这一页会对所有站点入口可用，而无论它们的语言配置如何（即使站点入口不支持英文）。

如果"Contacts"页面（节点 12）被翻译成德语，它会得到第二个别名"Kontakten"。在本例中，站点结构如下：

- Company | Compagnie（节点 ID：10）
  - *About us（节点 ID：11）*
  - *Contacts | Kontakten（节点 ID：12）*

在这里，可以用以下四个 URL 别名访问"Contacts|Kontakten"（节点 12）。下表揭示了为了让每个 URL 别名工作，站点入口所需要的语言设置。

URL 别名	所需的站点语言
"Company/Contacts"	英文
"Compagnie/Contacts"	英文和法文
"Company/Kontakten"	英文和德文
"Compagnie/Kontakten"	法文和德文

4.4.1 管理 URL 别名

在管理界面中可以简单地管理站点的虚拟 URL。可以在两个列表中管理。它们中的一个与节点 URL 别名相关，另一个负责处理全局别名。此外，还可以通过 URL 通配符界面管理通配符 URL 转发规则（被称为“通配符别名”）。

管理节点别名

节点 URL 别名的管理界面可以从节点的上下文菜单中的高级选项中的“管理 URL 别名”项目来触发。

也可以通过"content/urlalias/<node_id>"直接访问（用节点的 ID 代替<node_id>）。下图演示了节点 URL 别名管理界面的外观。

这个界面列出了所有属于所选节点的 URL 别名。此外，它可以用来创建与删除别名。本例列出了属于"Company"节点的所有虚拟 URL。有三个手动别名："articles/company_info","MaCompagnie"和"mycompany"。"MaCompagnie"别名与法语关联，"article/company_info"和"mycompany"别名为英文。这意味着如果站点入口同时支持英文和法文，同一个节点可以同这些 URL 别名中的任何别名访问。

下拉框可以用来选择别名关联的站点语言。例如：如果选择了“西班牙”，这个别名会在所有使用西班牙语的站点入口中可用。下拉框包含了所有可用于管理站点入口的语言。如果"ShowUntranslatedObjects"选项被启用，则所有的翻译语言会被显示。例如：尽管对象不存在于西班牙语，也可以创建关联到这种语言的别名。注意，多语言别名不控制内容显示的语言（这由当前站点入口的语言配置决定）。

“相对于父节点”复选框可以用来控制别名从何处开始。如果勾选，它会从父节点开始。否则，别名从站点根开始。例如：如果您为"/country/state/city"这个节点添加一个称为"test"的别名，新的 URL 别名可以为"/country/state/test"或"/test"，取决于是否勾选“相对于父节点”复选框。

“一般别名”窗口

在上例中，"Company"节点存在于英文和法语，因而有两个自动生成的别

名："Company"和"Compagnie"。这些别名是系统基于现有翻译而自动生成的。自动生成的别名在“一般别名”窗口显示。“一般别名”窗口位于界面的底端。如果站点入口同时支持法文和英文，则可以用任何上述别名访问"Company"节点。

注意，无论节点的父节点在不同语言中有多个别名，“一般别名”窗口为每种语言只显示一个别名。换言之，并非所有的 URL 别名组合都会被显示。例如：如果在"Company"节点下创建一个新节点"Employees"，它可以通过以下别名访问：

- Company/Employees
- Compagnie/Employees
- articles/company_info/Employees
- MaCompagnie/Employees
- mycompany/Employees

但是，对于"Employees"节点，只有上述别名中的一个会被显示。系统会自动选择一个父节点的自动生成的别名（"Company"或"Compagnie"），取决于站点入口的语言配置。如果管理站点入口最优先的语言为英文，在"About"节点的“一般别名”窗口中，只有"Company/Employees"别名会被显示。如果最优先语言为法文，则显示"Compagnie/Employees"。父节点的别名会以粗体显示。上图中，管理站点入口的最优先语言为英文，因而"Company"别名以粗体显示。

创建新节点别名

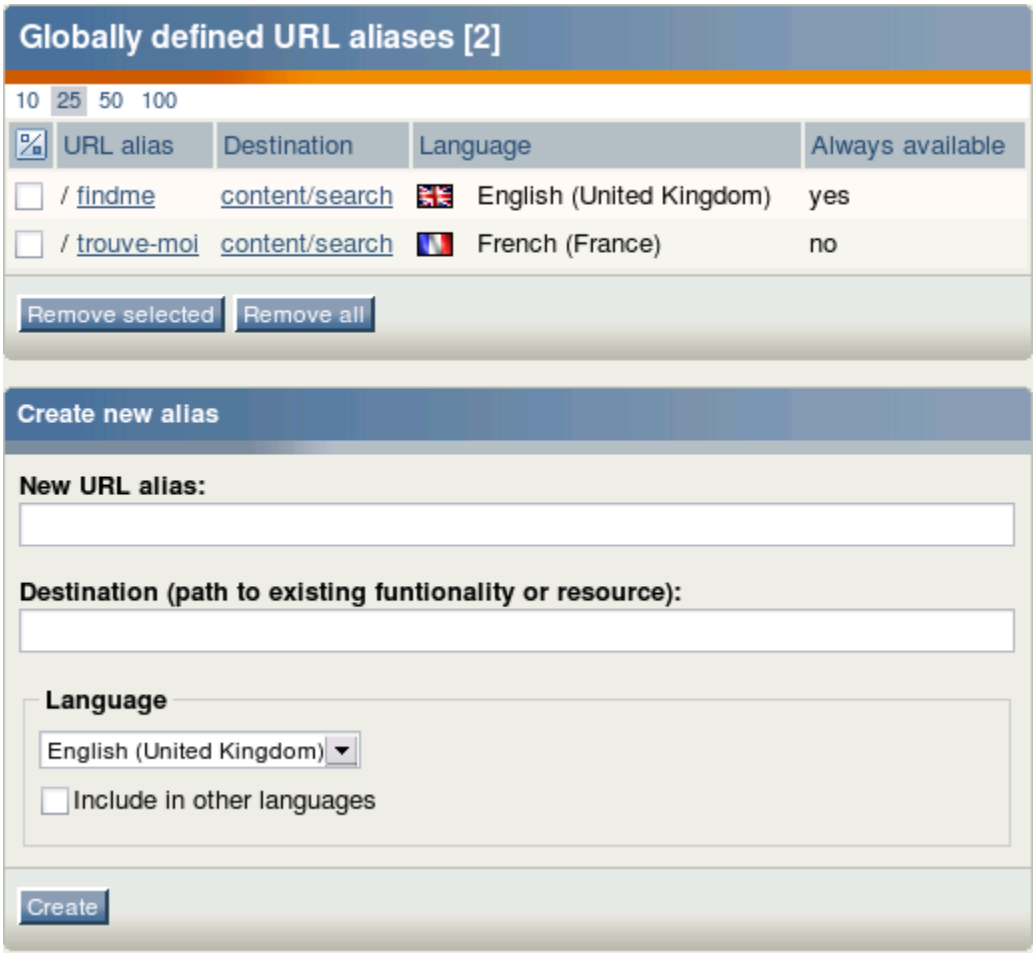
要创建新别名，首先选择关联到哪个站点语言。然后输入别名的文本，再点击“创建”按钮。可以用别名让节点看起来似乎被置于节点树中完全不同的位置。例如：您可以为"Articles/Article"这篇文章创建一个别名"my_dummy_folder/my_article"。注意：不能勾选“相对于父节点”复选框。

附加说明

假设您在节点树中的某个位置（位置不重要）有一个节点"About us"。如前说述，您可以创建一个想象的 URL（由非法的/不存在的父节点构成）。例如：您可以创建"company/about_us"并且它可以工作（系统会显示"About us"节点）。假设"Company"节点之前不存在，如果有人直接访问"company"，系统会返回“对象不存在”错误页面。但是，如果创建了"Company"节点，系统会为它自动创建一个 URL 别名（很可能是"company"），于是"company"别名会工作（它会显示"Company"节点）。

管理全局别名

管理全局别名的界面很早就被引入系统。但是在 3.10 版本中被修改。这个界面可以通过点击“设置”标签下左侧的“URL 翻译器”链接访问。下图演示了这个界面的外观。



如上图所示，这个界面与管理节点的 URL 别名的界面类似。这个列表显示了系统中所有的别名。列表由别名文本排序（不是由别名的路径排序）。

在上例中，有两个对"content"模块的"search"视图的别名。第一个别名与英文关联，第二个与法文关联。这允许用"findme"或"trouve-moi"别名访问"search"视图，如果站点入口同时支持英文和法文。换言之，"http://www.example.com/content/search"，"http://www.example.com/content/findme"和"http://www.example.com/trouve-moi"均会显示检索界面。“总是可用”列揭示这个别名是否总是可用。在上图中，"findme"别名总是可用（不论站点入口如何配置，它总是可用）。

注意:

与 3.10 之前版本不同，这个列表不再显示节点别名。节点别名可以在节点别名管理解明中为每个节点单独编辑/创建。

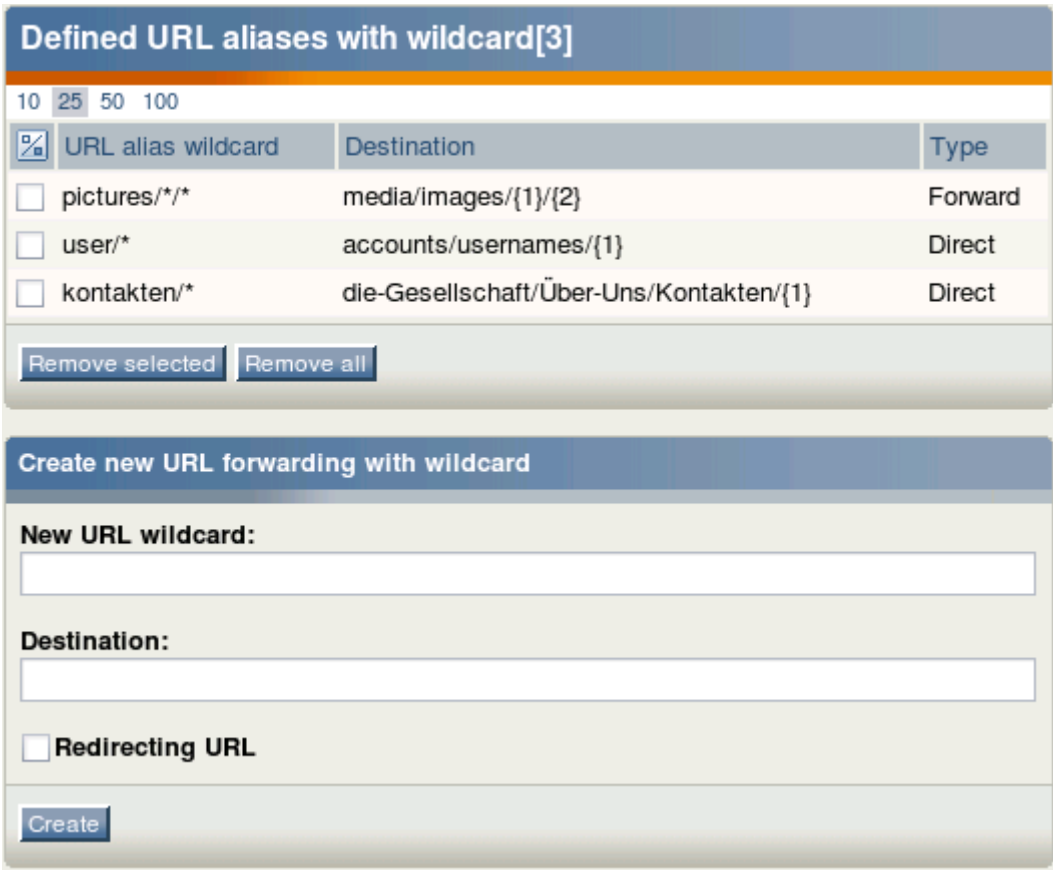
创建新的全局别名

要创建新的全局别名，输入虚拟 URL 和指向现存功能或资源的路径；这可以是模块/视图组合或节点的地址。指定的别名总是从站点根开始。语言下拉框可以用来别名关联的站点语言。如果“包含在其它语言”复选框被勾选，新创建的别名会对所有站点入口可用，不论站点入口语言配置如何。

注意，可以用全局别名界面创建到节点的别名（例如：指向"content/view/full/<node_id>或虚拟 URL）。但是，这样的别名不会在全局列表中显示。他们会在节点别名界面中显示。

管理通配符 URL 别名

管理通配符 URL 别名的界面可以通过点击“设置”标签下左侧的“URL 通配符”链接来访问。下图演示了通配符 URL 别名的外观。



上图显示了系统中所有的通配符别名并且允许您创建新的别名。在上例中，有三个通配符别名。前两个别名为英文，第三个为德文。

在浏览器中输入 URL 后，系统会根据指定的通配符规则寻找与输入 URL 匹配的节点。在 eZ Publish 中，只有(*)才可以用于通配符别名。在别名中(*)可以重复使用。注意，别名中的每个通配符会被系统自动分配一个数字：1,2,3...。这些数字可用于引用通配符说匹配的字符串：{1},{2},{3},...（这一点与正则表达式中的"back reference"类似）。

“类型”列揭示了别名为“直接”还是“转发”类型的。在上图中，第一个别名为“转发”类型，意味着系统会重定向到目标 URL。换言之，当在浏览器的地址栏中输入"http://www.example.com/pictures/home/photo/"，系统会重定向到"http://www.example.com/media/images/home/photo/"。

创建通配符别名

要创建一个新的通配符别名，输入别名的文字和目标地址。如果“重定向 URL ”复选框被勾选，别名会工作于“转发”模式。点击“创建”按钮后，新创建的别名会在列表中显示。

4.4.2 URL 变换规则

当站点管理员输入一个新的虚拟 URL，系统会所谓的 URL 变换规则来清理输入的 URL。这是为了避免由输入的字符造成的问题并且确保别名符合站点内其它 URL 的标准。如果修改了别名，用户会被通知。

注意：

在 eZ Publish 3.10 版本中，对输入/自动生成别名的变换规则已经改变。

支持 Unicode

在 3.10 版本之前，只有下划线可被用作单词分割符。从 3.10 开始，可以选择分割符。可以在"site.ini"的重设文件中"[URLTranslator]"下的"WordSeperator"设置。分割符可以为"-","_"或" "（空格）。注意，使用"urlalias_compat"方法时，这个配置会被忽略（因为它只支持下划线）。

大小写敏感

当使用"urlalias"或"urlalias_iri"方法时，URL 会包含大小写字符。这与传统的/旧的行为不同，那时字符都会被转换为小写字符。相反，系统会保留大小写并相应保存 URL 别名。但是，URL 本身并非大小写敏感。例如：节点"About Us"的 URL 别名可能为"About-Us"（假设分割符号为"-"）。您可以用任何大小写组合的 URL 访问这个节点。换言之，以下 URL 都可以用来访问这个节点："www.example.com/about-us","www.example.com/About-us","www.example.com/ABOUT-US"；等等。

注意：

如果在同一个位置有两个节点，它们的名字相同（几乎），（例如："My article"和"My Article"），系统会为通过为新创建的节点的 URL 后追加数字来为它生成唯一的 URL 别名。例如：如果节点"My article"已经存在且"My Article"有被创建，第二篇文章的 URL 别名会被生成为"My-Article2"。如果第三篇文章"MY-Article"被创建，则 URL 别名为"MY-Article3"。

别名文本过滤

别名过滤被实现用来在生成 URL 别名时引入更多灵活性。过滤由系统在转换 URL 别名之前完成。过滤器可以用扩展实现。以下文字解释了如何创建一个过滤器。

1. 编辑"site.ini"重设文件并在"[URLTranslator]"中加入一个新的扩展（如："myfilters"）。

```
Extensions[]
Extensions[]=myfilters
```

2. 在"[URLTranslator]"中的"Filters[]"中添加新的过滤器（如："StripWords"）。

```
Extensions[]
Extensions[]=myfilters


Filters[]
Filters[]=StripWords
```

系统会在"stripwords.php"中寻找"StripWords"过滤器类。

3. 在"extension/myfilters/urlfilters"中创建"stripwords.php"。注意，这个文件必须被放置在"urlfilters"目录内。确保这个文件包含如下内容：

```
<?php
     class StripWords
     {
          function process( $text, $languageObject, $caller )
          {
              return str_replace( "hell", "", $text );
          }
     }
?>
```

过滤器类"StripWords"实现了"process"方法，它有三个参数：被过滤的文字，语言对象（eZContentLanguage）和调用过滤方法的对象。这个方法返回过滤后的文字。在本例中，所有的"hell"都被删除（由空串替换）。换言之，过滤之后，新的 URL 中不包含"hell"。

参阅"site.ini"中的"[URLTranslator]"了解更多。

4.4.3 自定义变换命令

为了按照特殊需求变换 URL，可以创建并使用所谓的自定义命令。可以用扩展来实现并通过"transform.ini"的重设文件来加入系统。以下文字解释了如何创建自定义 URL 变换命令。

假设由于某种原因，我们希望倒序显示 URL。这可以通过创建一个自定义的变换命令来实现。

1. 创建一个新的扩展

必须在一个扩展的目录中创建一个新文件。文件必须包含一个类且这个类必须包含一个静态方法"executeCommand"。它有三个参数："$text","$command","$charsetName"。

- $text – 需要被变换的文字
- $command – 要执行的命令名称，这可以用于在一个函数中保留多条命令

- $charsetName - $text 的字符集名称，通常可以省略

在本例中，我们将在"extension/myextension/transformation"目录下创建"myreverse.php"。代码如下：

```
<?php
   class MyReverse
   {
       function executeCommand( $text, $command, $charsetName )
       {
           $text = strrev( $text );
           return $text;
       }
   }
?>
```

如代码所示，函数会返回输入文字的倒序序列。

2. 在"transform.ini"重设文件中注册一个新的变换命令

新的命令必须在"transform.ini"重设文件中注册。您需要在"[Extensions]"下的"Commands[]"数组中增加一行。这一行必须包含指向这个 PHP 文件的路径，一个冒号":"和要使用的类名。下例演示了如何配置：

```
[Extensions]
Commands[]
Commands[my_reverse]=extension/myextension/transformation/myreverse.php:MyReverse
```

3. 在对应的变换组中添加新的命令

新创建的命令必须被添加到"transform.ini"重设文件中用于 URL 变换的组中。在本例中，自定义命令"my_reverse"被加入"urlalias_iri"组。

```
[urlalias_iri]


Commands[]
Commands[]=url_cleanup_iri
Commands[]=my_reverse
```

现在，新生成的 URL 会以倒序显示在使用"urlalias_iri"变换方法时，URL 会被"my_revers"命令变换。

4.5 集群

集群特性允许在多台 WEB 服务器上运行同一个 eZ Publish 站点。集群站点会有更好的性能也可以承受更多的访问量。

可以将所有内容相关的缓存，图片和二进制文件保存在数据库中。数据库事务被用来保证所有的集群节点使用相同的缓存文件并访问相同的图片和二进制文件。换言之，当内容被上传，变化会自动对所有节

点生效。这个特性在 3.10 中得到了很大的改善。

注意：

当使用集群时，建议使用虚拟主机模式运行 eZ Publish。

3.10 引入的变化

在 3.10 版本以前，清除缓存会物理删除缓存文件。这一操作可能非常耗时。

从 3.10 版本开始，系统会将缓存标记为不可用而不是从数据库或文件系统中物理删除。这可以标记每个特别缓存文件为过期或设置全局过期时间（在需要很多变化时，例如：当清除特定类型的所有缓存，后者常被使用）。全局过期时间是一个时间戳，它可以被用来作为系统中所有缓存的过期时间。如果全局过期时间被设置为一个特定日期，所有比这个日期旧的缓存文件都不会被使用。注意，在重建缓存的时候，系统会重用旧的/过期的文件内容。

如果要从数据库中物理删除缓存文件，需要添加"—purge"参数来执行"bin/php/ezcache.php"。下例演示了如何删除所有两天前的内容缓存。

```
php bin/php/ezcache.php --clear-id=content --purge --expiry='-2 days'
```

如果要了解更多信息，可以用"—help"参数：

```
php bin/php/ezcache.php --help
```

注意：

3.10 不支持基于 PostgreSQL 和 Oracle 数据库的集群。代码针对使用 InnoDB 引擎的 MySQL 数据库做了性能优化。MySQL 的数据库连接数必须被增加 30-50%。这样做的原因是新的集群代码在将内容写入数据库时会建立另外一个连接（这个连接用于检查得到写入锁后，文件是否被修改过）。如果持续连接被启用，集群将不会与普通数据库操作共用连接，所以之前的连接数将翻倍。

3.9 版本引入的修改

从 3.9 版本开始，系统引入了一个附加的 HTTP 头"Served-by"。这一特性用于测试和调试目的。当您需要在客户端检查内容来源于哪个服务器时，这一特性变得很有用。下例演示了服务器回应的片段。

```
...
Last-Modified: Fri, 29 Jun 2007 09:35:54 GMT
Served-by: 62.70.12.230
Content-Language: en-GB
...
```

集群如何工作

需要在不同服务器见保持同步的数据被保存在数据库中：

- 二进制文件
- 图片和图片别名
- 内容相关的缓存
  - *内容视图缓存*
  - *模板缓存块*
  - *过期缓存*
  - *URL 别名缓存*
  - *RSS 缓存*
  - *用户信息缓存*
  - *类标识符缓存*
  - *排序键值缓存*

其它文件在文件系统中保存，包括（但不限于）：

- INI 文件
- 模板文件
- 编译的模板
- PHP 文件
- 日志文件
- 与内容无关的缓存
  - *全局 INI 缓存*
  - *INI 缓存*
  - *代码页缓存*
  - *字符变换缓存*
  - *模板缓存*
  - *模板重设缓存*

内容视图缓存

当 eZ Publish 显示一个页面（一个内容节点）时，它会执行"content"模块的"view"视图并在 pagelayout 中包含视图的输出。如果视图的输出被缓存，缓存文件会被读取并使用。否则，系统会从 eZ Publish 数据库中提取内容，生成必要的模板，生成 WEB 页面并在返回结果前将生成的 XHTML 保存在文件系统中。如前说述，这些文件现在可以（从 3.8 版本开始）被保存在数据库中因而文件（以及改动）可以被简单迅速地在所有的服务器中共享。

图片和图片别名

上述解决方案也被用于图片和图片别名（图片变种）。但是，解决方案有一点复杂因为直到最近(3.8)，

图片都是被 Apache 直接提供的。因为 WEB 服务器不能直接与数据库沟通，图片需要由一个称为"index_image.php"的 PHP 脚本来提供。这条规则应用于所有与内容有关的图片，但是不适用于界面中的图片。请注意，您需要为 Apache 添加新的 rewrite 规则来要求 Apache 在提供图片的时候使用"index_image.php"。

集群文件处理器

在 3.8 中引入了一种新的文件管理器机制。它允许在数据库中对文件进行保存，提取，重命名，删除等操作。集群文件处理器在"kernel/classes/clusterfilehandlers"目录中。以下为系统内建的集群文件处理器：

- ezfs(eZFSFileHandler)
- ezdb(eZDBFileHandler)

eZFSFileHandler

这个处理器允许使用文件系统来处理文件。

eZDBFileHandler

这个处理器允许使用数据库处理文件（在集群环境中，这通常包括图片，上传的二进制文件，内容有关的缓存等等）。它被划分为不同的后台实现以支持不同的数据库（注意：目前只有 MySQL 被支持）。

自定义处理器

可以开发自己的处理器和/或后台实现来扩展系统。这可以通过扩展系统实现（不是通过修改 eZ Publish 内核文件）。

"file.ini"中的"[ClusteringSettings]"下的"[ExtensionDirectories[]"数组是 eZ Publish 用来寻找附加集群文件处理器的位置。默认情况下，eZ Publish 会在您扩展的"clusterfilehandlers"子目录内寻找。

例

如果您有一个扩展"myExtension"，它包括一个集群文件处理器"cfh"。您应该在您的"file.ini.append.php"文件中的"[ClusteringSettings]"下添加如下配置：

FileHandler=cfh

ExtensionDirectories[]=myExtension

这些配置会要求 eZ Publish 使用位于"extension/myExtension/clusterfilehandlers/cfhfilehandler.php"的自定义集群文件处理器。

4.5.1 配置

以下步骤演示了如何配置 eZ Publish 在数据库中保存图片，二进制文件和内容相关的缓存。

1. **清除缓存（可选）**

建议（但不是必须）在启动集群功能之前清除所有 eZ Publish 缓存。步骤如下：

```
php bin/php/ezcache.php --clear-all --purge
```

运行缓存脚本之后，确保所有缓存文件已经被清除（通常在"var/cache/"和"var/<站点入口名称>/cache/"）。如果还有残留文件，手动删除它们。

2. **修改"file.ini"配置**

在"file.ini"的重设文件中添加如下配置：

```
[ClusteringSettings]
FileHandler=ezdb
DBBackend=mysql
DBHost=localhost
DBPort=3306
DBSocket=
DBName=name
DBUser=user
DBPassword=pass
DBChunkSize=65535
```

用实际的数据库连接信息替换"localhost","name","user"和"pass"（大部分情况下，这些设置与"[DatabaseSettings]"下的配置相同）。确保"DBSocket"的配置正确（如果在"site.ini"的重设文件中"[DatabaseSettings]"下的配置为"Socket=disabled"，则设置为空）。

在"FileHandler"配置中指定"ezdb"，这会要求 eZ Publish 使用指定的数据库保存图片，二进制文件和内容相关的缓存。"DBBackend"配置指定"ezdb"文件处理器应该使用何种数据库后端（目前，只支持"MySQL"）。"DBChunkSize"指定了从数据库中提取文件时，内容块的大小（字节）。

3. **创建新脚本用来提供图片**

所有的图片（除了界面图片）将会由 PHP 脚本提供。Apache 会被要求用特定的 PHP 脚本"index_cluster.php"来处理图片。这个脚本必须引入"index_image.php"以及一系列配置。这个技巧可以另图片的处理更快因为系统不需要从数据库读取配置。在 eZ Publish 的安装目录中创建"index_cluster.php"并输入如下内容：

```
<?php
define( 'STORAGE_BACKEND',     'mysql'           );
define( 'STORAGE_HOST',        'localhost'       );
define( 'STORAGE_PORT',        3306              );
define( 'STORAGE_SOCKET',      ''                );
```

```
define( 'STORAGE_USER',         'user'             );
define( 'STORAGE_PASS',         'pass'             );
define( 'STORAGE_DB',           'name'             );
define( 'STORAGE_CHUNK_SIZE',   65535              );


include_once( 'index_image.php' );
?>
```

确保数据库的配置与"file.ini.append.php"中的"[ClusterSettings]"配置相同。

4. **创建新的数据库表**

您需要在数据库中手动创建一些表。您可以在"kernel/classes/clusterfilehandlers/dbbackends/mysql.php"中找到表结构。表结构的定义在文件开头的注释中。

5. **将文件导入数据库**

您需要将"var"目录中的文件复制到数据库。在"eZ Publish"的根目录中执行如下脚本（用实际的站点入口名替换"ezwebin_site"）：

```
php bin/php/clusterize.php -s ezwebin_site
```

这个脚本会将"var"目录中的文件，图片，图片别名复制到数据库中。

6. **编译模板（可选）**

因为所有缓存为空，您需要重新编译模板。注意，这一步可以省略，而模板会在被请求时被编译。在 eZ Publish 根目录中执行以下脚本：

```
php bin/php/eztc.php -s ezwebin_site
```

为每个站点入口执行这个脚本

7. **更新 Apache 配置文件**

Apache 需要知道在提供图片时应该使用哪个 PHP 脚本。这个脚本只是简单地从数据库提取并提供图片。将以下规则添加到.htaccess 文件中其它规则之上：：

```
Rewriterule ^/var/([^/]+/)?storage/images-versioned/.*  /index_cluster.php  [L]
Rewriterule ^/var/([^/]+/)?storage/images/.*            /index_cluster.php  [L]
```

如果".htaccess"文件不存在，在 Apache 配置文件中添加以上规则。

8. **重启 Apache 并测试站点**

重启 Apache。重启 Apache 之后，系统会以集群模式启动。确认站点正常工作，内容图片可以被显示且内容二进制文件可以被访问（在浏览器中登录到管理界面，试着点击不同链接）。

9. **从文件系统中删除导入的文件**

如果站点正常工作，你可以从文件系统中删除原始的内容图片和二进制文件（因为它们已经被成功导入数据库）。您需要调查"var"目录（特别是"var/storage/"和"var/<name_of_siteaccess>/storage/"目录）。如果这些目录中有任何内容图片或二进制文件，可以手动删除它们。

4.5.2 撤销集群配置

您总是可以将站点出集群模式切换回标准模式：在文件系统中保存图片，二进制文件和内容相关的缓存。以下步骤解释了如何切换。假设 eZ Publish 站点现在使用 MySQL 保存文件。

1. **清除缓存（可选）**

建议（不是必须）在撤销集群之前清除缓存。

```
php bin/php/ezcache.php --clear-all --purge
```

2. **撤销图片和文件的集群**

您需要将图片，二进制文件，内容相关的缓存从数据库复制回文件系统。可以用"bin/php/clusterize.php -u"来实现。执行之前确保您有足够的磁盘空间。

下例演示了如何运行这个脚本（用实际的站点入口替换"ezwebin_site"）。

```
php bin/php/clusterize.php -u -s ezwebin_site
```

以上脚本将会遍历数据库中保存的文件并将它们复制到"var"下面正确的子目录中。

3. **修改"file.ini"配置**

编辑"file.ini"的重设文件并注释以下内容：

```
[ClusteringSettings]
#FileHandler=ezdb
#DBBackend=mysql
#DBHost=localhost
#DBPort=3306
#DBSocket=
#DBName=name
#DBUser=user
#DBPassword=pass
#DBChunkSize=65535
```

注意，您也可以直接删除以上配置。

4. **禁用提供图片的脚本**

重命名或删除"index_cluster.php"。

5. **修改 Apache 配置**

注释或删除以下 Apache 配置：

```
Rewriterule ^/var/([^/]+/)?storage/images-versioned/.*  /index_cluster.php  [L]
Rewriterule ^/var/([^/]+/)?storage/images/.*            /index_cluster.php  [L]
```

6. **重启 Apache 并测试站点**

重启 Apache 服务器。重启之后，您的站点应该可以使用"var"目录下的图片，文件和内容相关的缓存。确保站点正常工作（在浏览器中登录管理界面，试着点击不同链接）。

7. **删除数据库中的集群表**

如果站点正常工作，您可以从数据库中删除之前使用的图片，文件和内容缓存。

```
DROP TABLE ezdbfile;
DROP TABLE ezdbfile_data;
```

4.6 安装包

从 3.8 版本开始，标准安装包不包含在 eZ Publish 发行版本中。它们以".ezpkg"文件是形式单独发行。可以通过安装向导自动从远端软件仓库下载活手动从安装包下载地址下载。

一个安装包是一组特殊格式的方便安装/卸载的文件。系统允许创建安装包并把它导出为".ezpkg"文件。这是安装包发行的一般方法。当".ezpkg"文件被导入后，它会被加入本地 eZ Publish 的系统软件仓库中。它们中的大部分都可以被安装和卸载。下表揭示了在管理界面中可用于安装包的全部操作。

操作	描述
创建安装包	可以通过创建不同类型的安装包导出您的类定义，内容对象，配置，界面风格等对象。新创建的安装包会被保存在本地的软件仓库中。
导入安装报	要导入一个新的安装包，您需要从本地选择需要的".ezpkg"文件。系统会上传，解压这个文件并把解压后的内容保存在正确的内容软件仓库中。
删除安装包	您可以从系统的软件仓库中删除安装包。请注意安装包本身不会被删除。只有安装包文件会从内容软件仓库删除。
安装	可以安装本地软件仓库这中的安装包。当安装一个安装包时，系统会创建内容类，内容对象，应用配置等等 。（注意，不支持安装站点安装包和界面安装包）
导出安装包	系统软件仓库中的安装包可以被导出为".ezpkg"文件。系统会询问你在什么位置保存新创建的文件。
卸载	卸载安装包时，所有安装造成的改动会被撤销。

远端软件仓库

位于远端软件仓库中的安装包可以被安装向导下载并安装。但是，管理界面不能从远端软件仓库下载安装包。

系统会用"http://packages.ez.no/ezpublish/4.0"作为默认的远端软件仓库。如果您希望使用其它的远端软件仓库，您需要在"settings/override/package.ini.append.php"中的"[RepositorySettings]"配置"RemotePackagesIndexURL"。

系统/内部软件仓库

默认情况下，所有的安装包都位于"var/storage/packages"目录中。这个目录是系统主软件仓库且它的子目录被称为“系统软件仓库”或“内部软件仓库”。子目录名也被用作软件仓库的名称。安装包根据开发商的名称排序。例如：从 ez.no 下载的安装包在"ez_systems"内部软件仓库（"var/storage/packages/ez_systems"）内保存。没有开发商的和本地的安装包会被保存在"local"软件仓库（"var/storage/packages/local"）。

您也可以将"var/storage"下的其它位置作为系统主软件仓库。下例演示了如何通过修改"settings/override/package.ini.append.php"将主软件仓库的位置换成"var/storage/importedpackages"。

4.6.1 安装包类型

系统支持以下类型的安装包：

- 内容类安装包
- 内容对象安装包
- 扩展安装包
- 站点风格安装包（界面安装包）
- 站点安装包

内容类和内容对象安装包

内容类安装包可以保存类定义。如果您定义了一些类，又希望在其它站点中使用它们，您可以把这些类定义导出到一个内容类安装包中。这个安装包可以随后被导出为".ezpkg"文件。这个文件可以在别的站点被导入并安装。

内容对象安装包用来保存实际的内容对象。如果您创建了一些对象又希望在其它站点中使用它们，您可以把对象导出为内容对象安装包。

安装包向导会让您选择希望包含在安装包中的节点和/或子树。您也可以在安装包中包含对象的类定义，不同站点入口中相关的模板。选中的对象可以与它所有的版本和翻译一起导出活您可以指定自定义的参数。您可以选择导出所有的节点指派或只导出主节点，也可以指定如何处理关联对象。

数据类型串行化支持

在 3.8 版本中，所有的内建数据类型都与安装包系统兼容。都支持对象和类串行化。如果您使用附加数据类型但是不支持串行化，那么您在导出/导入包含这种数据类型的类定义和/或内容对象时会收到系统警告。

扩展安装包

这些安装包保存扩展文件。如果您创建了一个扩展又希望在其它站点中使用它，您可以把扩展导出到扩展安装包。管理界面允许创建扩展安装包。

界面/站点风格安装包

站点风格安装包保存站点界面风格。这类安装包允许简单地改变站点的外观。站点风格安装包包括非内容的图片（logo，页眉，图像外观元素等）和两个 CSS 文件（"site-color.css"包含 pagelayout 的颜色代码，背景图等信息，"classes-color.css"包含类模板的风格）。

注意：

站点风格安装包不能被安装/撤销。如果您导入多个界面安装包，下一章会解释可以通过管理界面来切换它们。

改变站点风格

假设您已经导入了一个新的站点风格安装包。要根据安装包改变您站点的风格，参阅如下步骤：

1. 在管理界面点击“设计”标签，然后从左侧菜单中选择“外观”。
2. 从站点风格列表中选择需要的风格
3. 点击“发布”按钮保存改动
4. 到实际的站点刷新页面。如果改动不生效，请清除缓存。

站点安装包

这种特殊的安装包提供站点示例如“新闻”，“商店”，“图片库”等等。主要用于演示和学习目的。站点安装包不包含任何对象。但是，他们包含对其它安装包的依赖以及特殊的配置和脚本。这些安装包不能被安装或撤销。站点安装包可以被想象为在安装向导中使用的一种“描述安装包”。（如果您从内部软件仓库中删除站点安装包时，不会影响以安装系统的行为。）不能在管理界面中创建站点安装包。

安装向导会自动从远端软件仓库下载可用的站点安装包并要求用户用户选择安装。它会自动下载选择的站点安装包和它的所有依赖包，将它们导入系统并显示成功导入的安装包列表。（如果所有安装包已经被保存在内部软件仓库，这一步会被忽略。）除站点风格安装包外，所有依赖安装包都会被自动安装。

例

"Shop site"安装包 v.2.0.6 包含对以下安装包的依赖：

- 三个内容对象安装包"Products","Multi-price products","Dynamic VAT products"。

- 一个站点风格安装包"Theme 04"。
- 一个扩展安装包"ezpaypal extension"。

从安装向导中选择"Shop site"安装包会从"http://packages.ez.no/ezpublish/4.0"下载这个安装包与其它五个依赖安装包。下载的".ezpkg"文件会被解压到"var/storage/packages/ez_systems"目录。安装向导会自动安装"Products","Multi-price products","Dynamic VAT products"和"ezpaypal extension"安装包。"Shop site"和"Theme 04"安装包不会被安装。当安装向导结束了，您可以手动或通过管理界面安全删除"shop_site"安装包。删除站点安装包不会影响它的任何依赖包。

4.6.2 创建安装包

管理界面允许您将类定义，内容对象，配置，界面风格等导出为不同类型的安装包。这个功能通过内建的安装包创建处理器实现。您可以创建以下类型的安装包：

- 内容类安装包
- 内容对象安装包
- 扩展安装包
- 站点风格安装包（界面安装包）

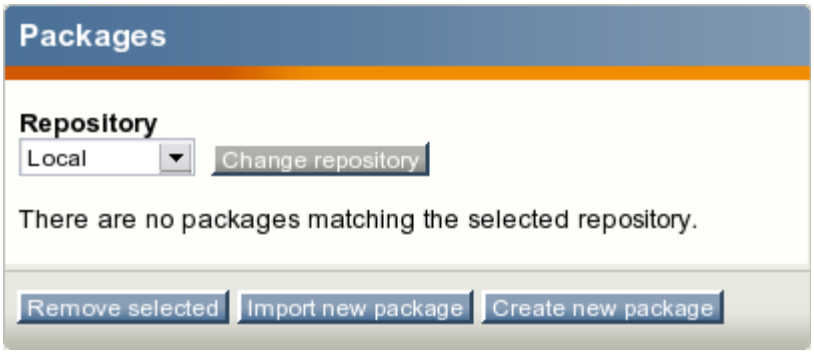
内建的安装包处理器被保存在"kernel/classes/packagecreators"目录。注意，没有对应站点安装包的处理器，因此不能在管理界面中创建此类安装包。本地创建的安装包被保存在"local"系统软件安装包。

下一章解释了如何创建不同类型的安装包。

内容类安装包

下例演示了如何创建内容类安装包。

1. 在管理界面中点击“设置”标签，然后点击左侧的“安装包”链接。系统会显示"local"系统软件仓库中的安装包列表。（这个界面也可以通过"/package/list“访问。）



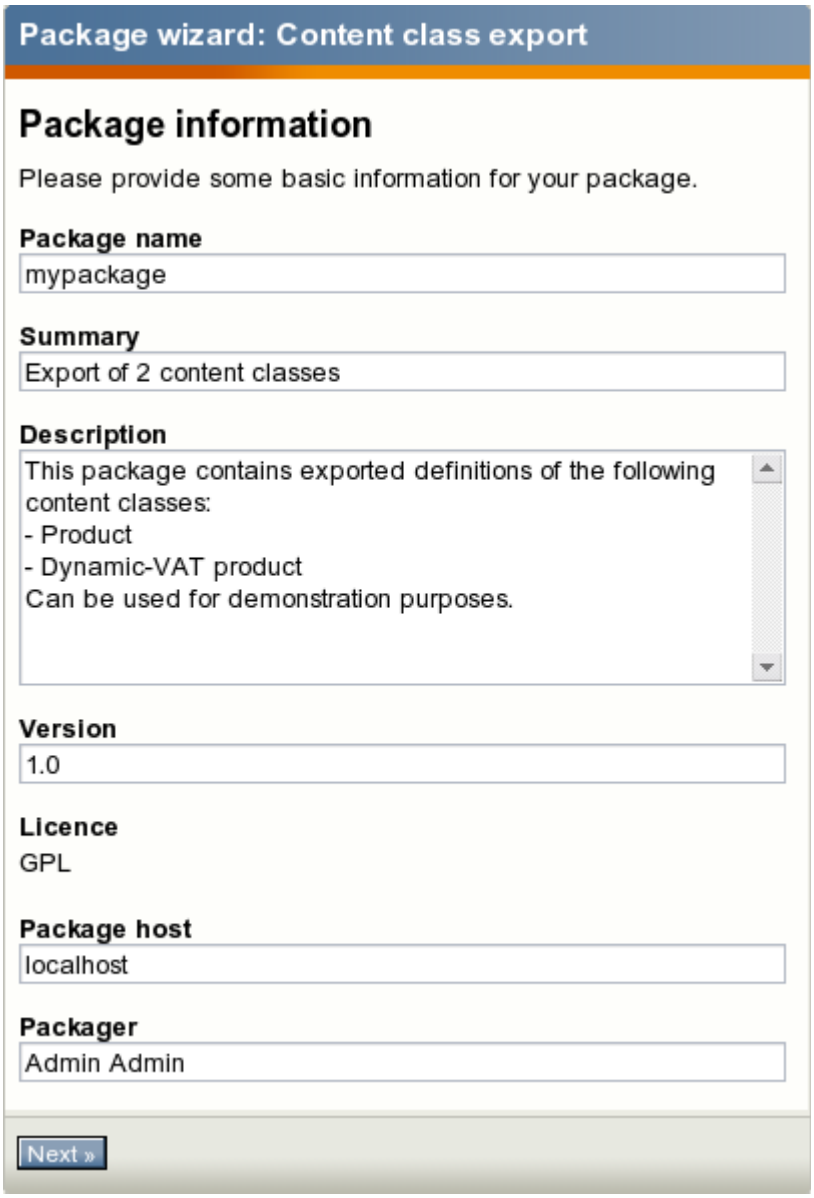
2. 点击页面底端的“创建新安装包”按钮。系统会显示安装包创建对话框。您可以在这个对话框中选择四种安装包创建向导中的一种（这个界面也可以通过"/package/create"访问）。

在上图中选择“导出内容类”然后点击“创建安装包”按钮

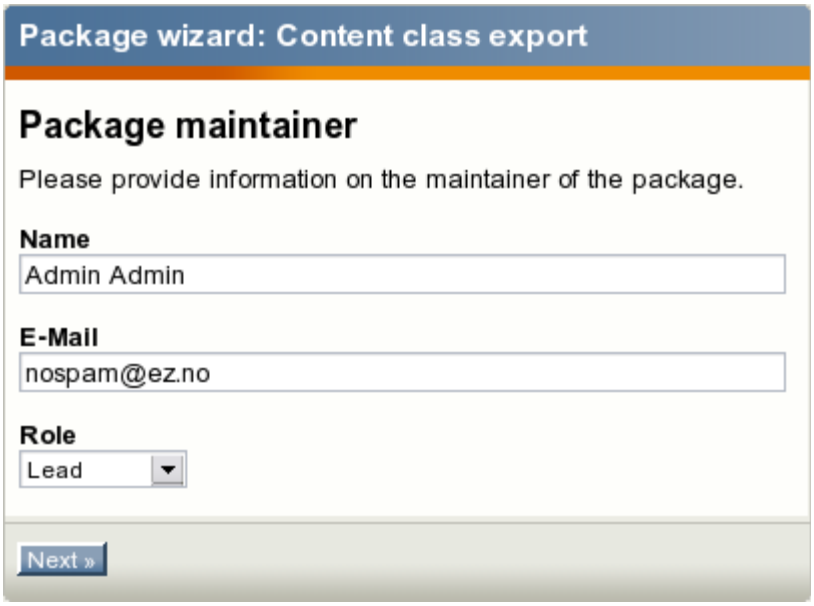
3. 安装包会询问您导出何种类（参阅下图）。

从列表中选择需要的类然后点击“下一步”按钮。

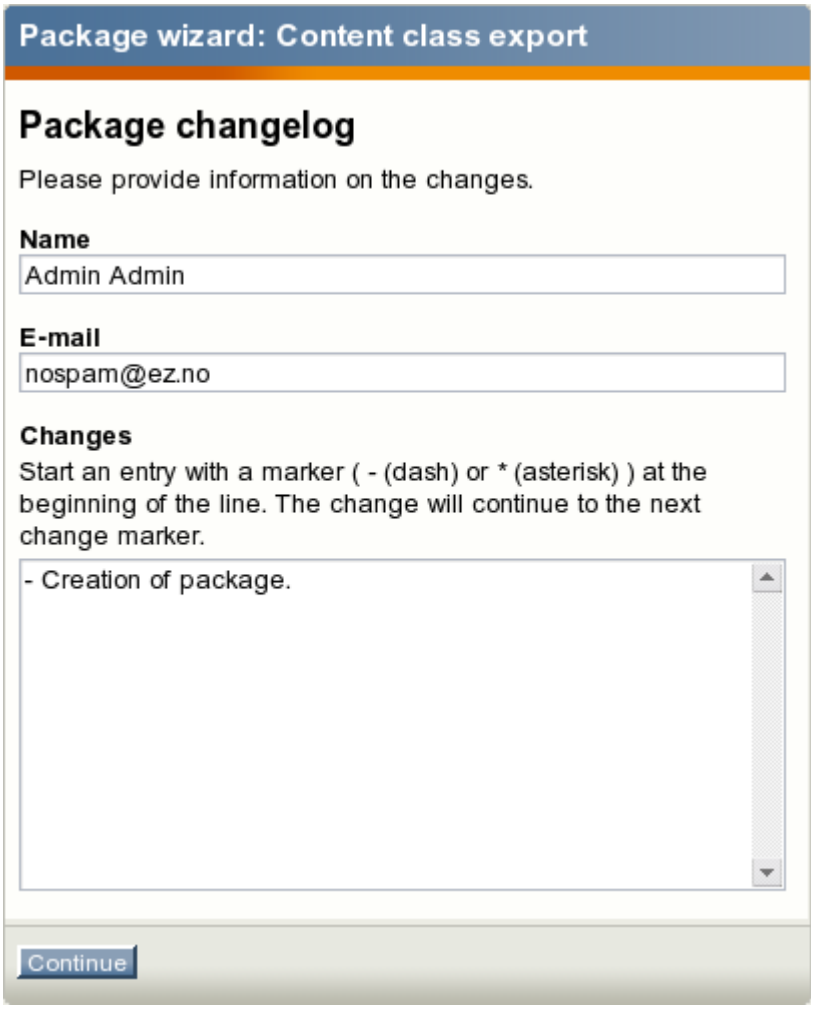
4. 现在您可以输入关于内容类安装包的信息。输入名称与简介。然后点击“下一步”按钮。



5. 系统还需要一些安装包的维护信息。输入这些信息然后点击“下一步”按钮。



6. 最后一步，您可以输入一些这个版本中的修改（参阅下图）。



点击“继续”按钮，安装向导会创建安装包并显示它的简介。

内容对象安装包

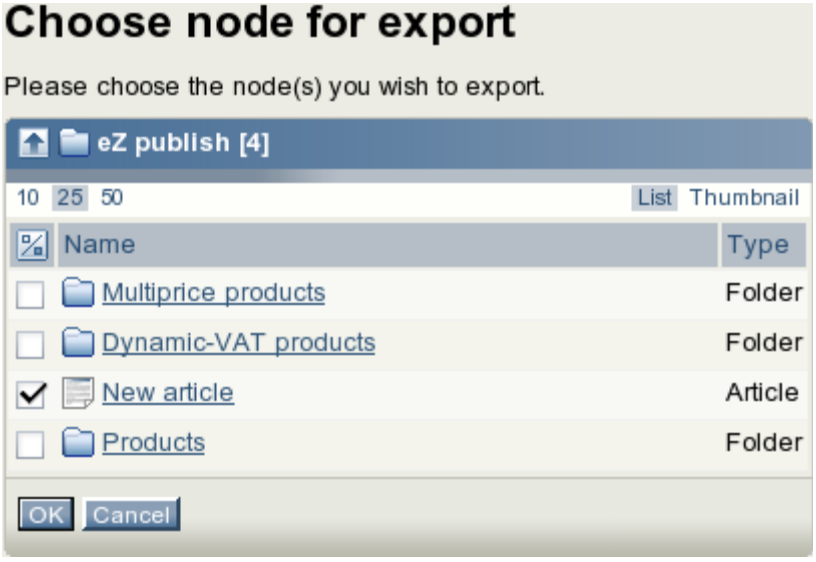
下例演示了如何创建内容对象安装包。

1. 在管理界面中点击“设置”标签，然后在左侧选择“安装包”链接再点击“创建新安装包”按钮。在安装包创建对话框中选择“导出内容对象”然后点击“创建安装包”按钮。（参阅下图）

2. 安装向导会询问您包含哪些对象（参阅下图）。

以下文字描述了如何使用“添加节点”，“添加子树”和“删除所选”按钮来选择对象。

- “添加节点”按钮允许添加单独节点到安装包中。点击这个按钮后，系统会显示“内容结构”对话框。用这个列表选择希望包含的节点。下图演示了这个对话框（"New article"节点被选择）。

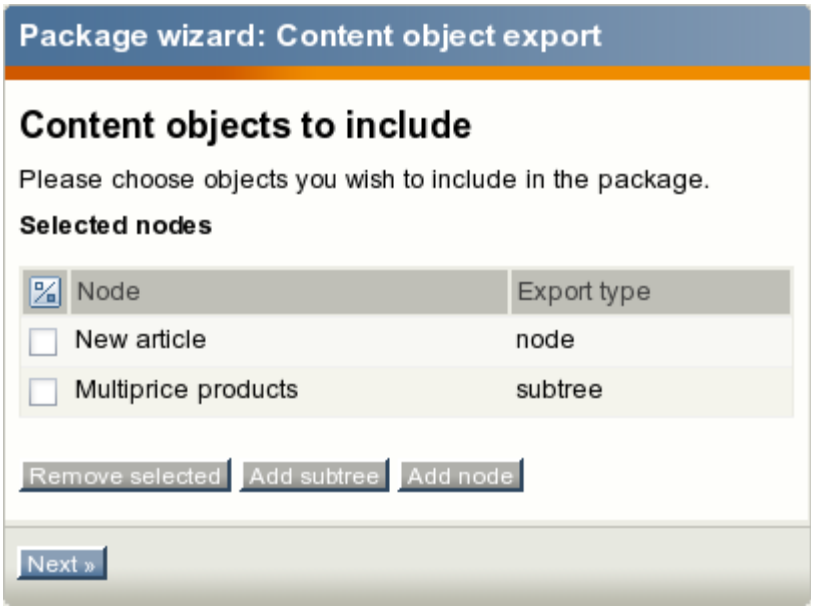


注意，您同时可以选择多个节点。您可以通过点击节点的名称在列表中导航。如果需要的节点在“内容结构树”外，点击上箭头图表直到到达内容树根节点。（这一操作将允许您例如：切换到“媒体库”内容树并选择图片对象）可以重新配置如何显示这个列表。例如：您可以设置每页显示的项目数（点击"10" / "25" 和 "50"链接）。如果您希望浏览以缩略图形式浏览图片对象，点击“缩略图”按钮。选择好需要的节点后，点击“确定”按钮保存您的选择。

- “添加子树”按钮允许添加整个子树到安装包中。如果点击这个按钮，您会看到“选择导出子树”对话框。这个对话框与节点选择对话框很类似。唯一的区别在于在这个界面选择节点意味着整个子树会被包含在安装包中。假设选择某个文件夹下个某个子树（参阅下图）。

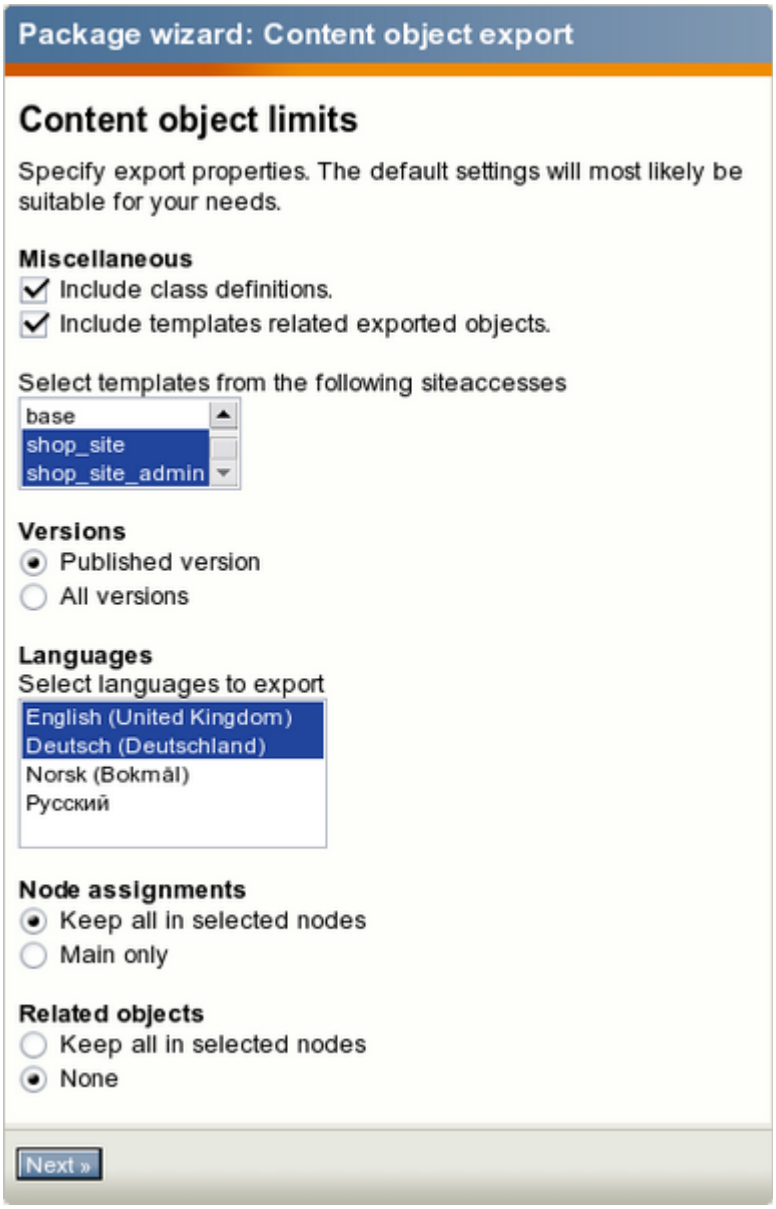
选择所需子树后，点击“确定”。

- 向导会显示选择的节点和子树。



如果您选择了错误的节点/子树，用复选框勾选项目然后点击“删除所选项目”按钮。如果内容正确，点击“下一步”按钮。

- 在下一个对话框中您应该选择需要的导出属性。您可以包含对象，类定义和关联的模板（模板可以来源于多个站点入口）。选择的对象可以与他们所有的版本和翻译一起导出或您也可以指定导出选项。下图演示了这个对话框的外观。



选择适当的属性然后点击“下一步”按钮。

- 剩下的三个步骤允许您输入关于安装包，维护和修改信息。这些已经在上述“内容类导出”步骤中讨论过。

扩展安装包

下例演示了如何创建扩展安装包。

1. 在管理界面的“设置—安装包”页面中点击“创建新安装包”按钮。在安装包创建对话框中选择“导出扩展”然

后点击“创建安装包”按钮。

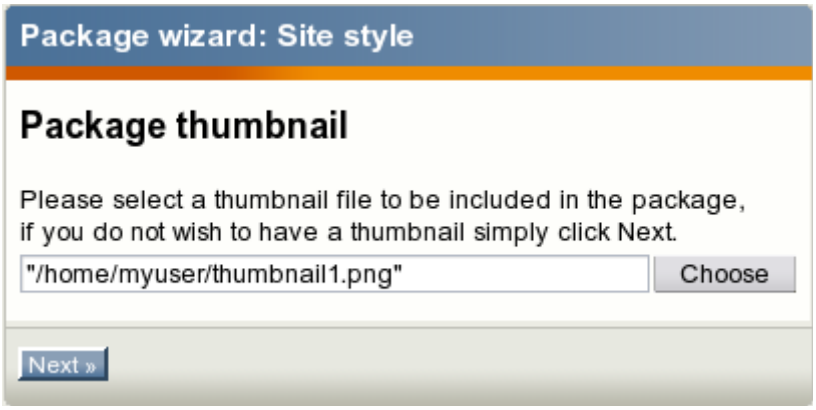
2. 导出向导会显示现存扩展的列表。选择需要导出的扩展（如下图所示）然后点击“下一步”按钮。

3. 剩下的三个步骤用于输入安装包，维护和变更信息。

界面/站点风格安装包

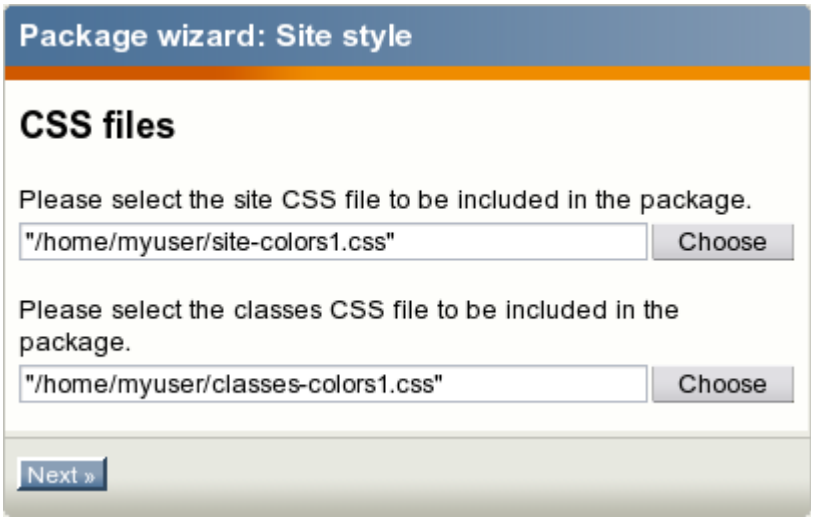
下例演示了如何创建站点风格安装包。

1. 在管理界面的“设置—安装包”页面中点击“创建新安装包”按钮。在安装包创建对话框中选择“站点风格”然后点击“创建安装包”按钮。
2. 导出向导会要求您选择一个缩略图，它应该是一个屏幕截图或描述外观的图标。图片应该为 120 像素 x103 像素（长 x 宽）。下图演示了这个对话框。

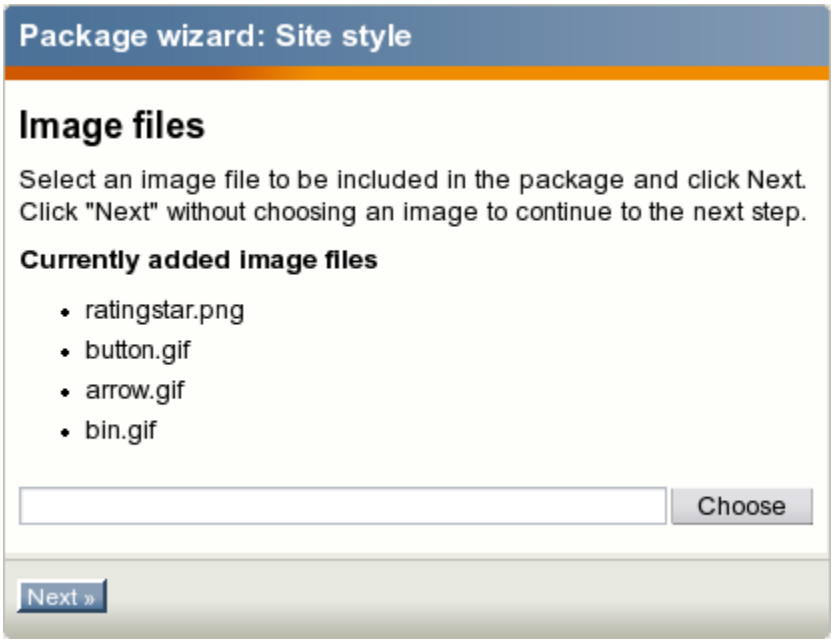


选择图片文件，然后点击“下一步”按钮。

3. 下一个对话框要求您提供两个 CSS 文件："site-colors.css"文件包含 pagelayout 的颜色代码和背景图和"classes-colors.css"包含类模板的风格。如下图所示，选择文件然后点击“下一步”按钮。



4. 如果在您的主题中使用图片，您可以在下一个页面中上传这些图片。



上传好图片后，点击“下一步”。

5. 剩下的三个步骤用于输入安装包，维护和变更信息。

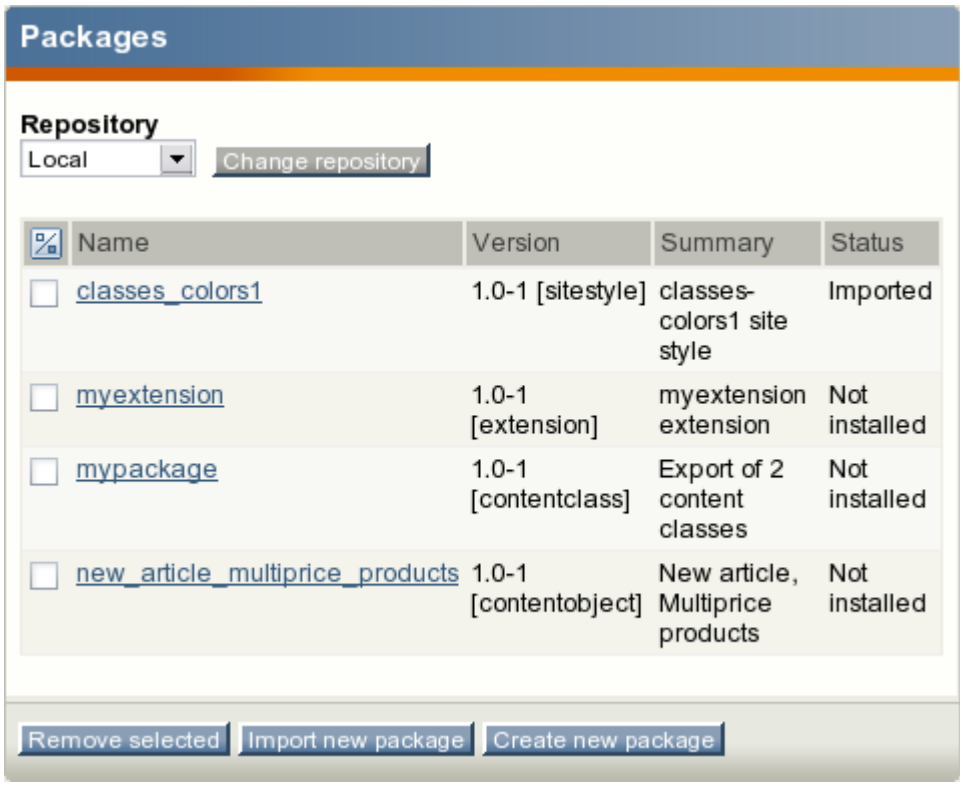
站点安装包

不能通过管理界面创建站点安装包（没有针对站点安装包的处理器）。它们只能手动创建，这意味者您必须手动编辑"package.xml"文件。

4.6.3 导出安装包

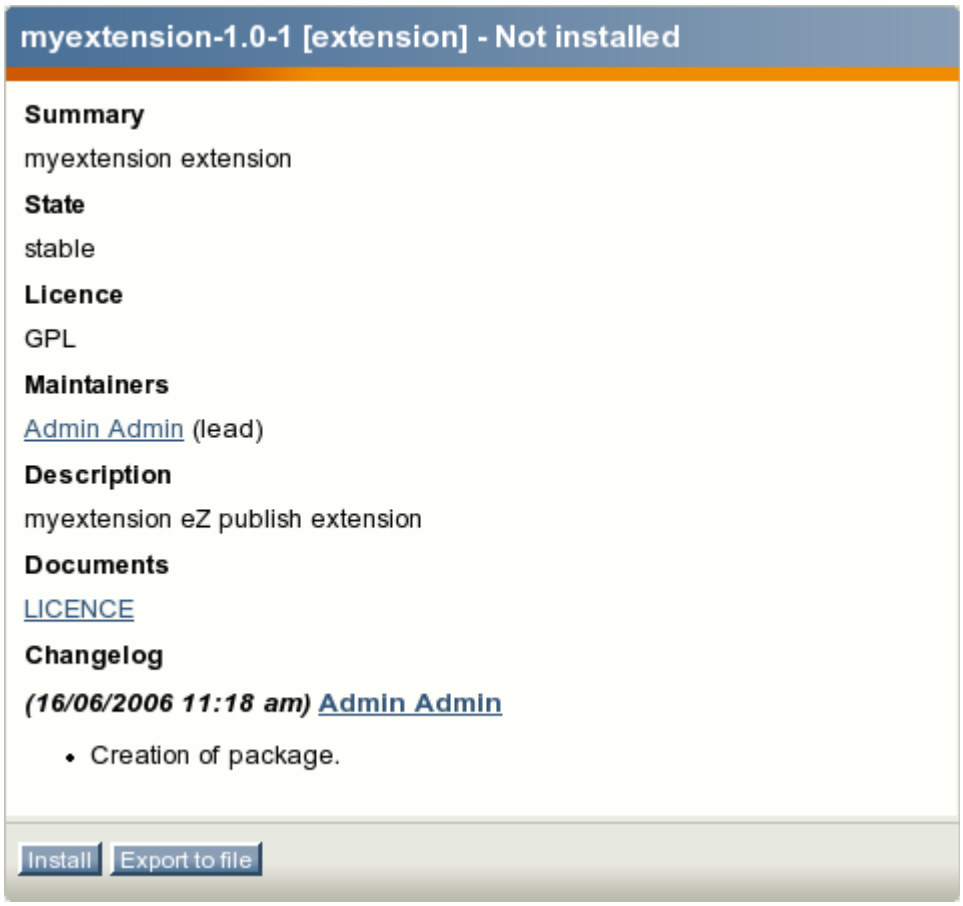
内部软件仓库中的安装包可以被导出为".ezpkg"文件。以下步骤演示了如何导出安装包。

1. 在管理界面中选择“设置”标签，然后选择左侧的“安装包”链接。系统会显示位于"local"系统软件仓库中的安装包列表（参阅下图）。



这个界面也可以通过"/package/list"访问。如果您希望浏览其它内部软件仓库中的安装包，从下拉框选择软件仓库名，然后点击“改变软件仓库”按钮。

2. 找到希望导出的安装包然后点击名称。系统会显示安装包简介页面（如下图）。

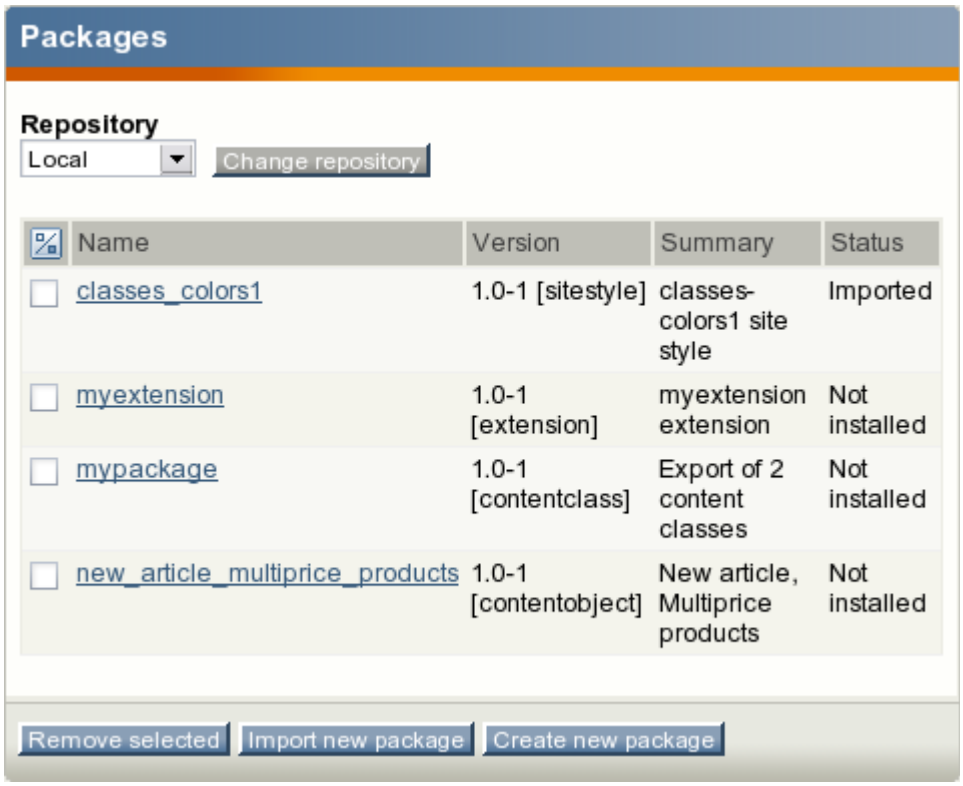


点击“导出到文件”按钮来下载".ezpkg"文件。

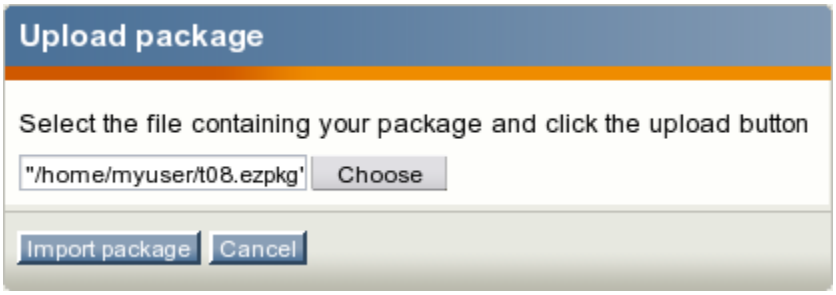
4.6.4 导入安装包

".ezpkg“文件可以被导入系统。例如：上传，解压，保存到适当的内部软件仓库。下例演示了如何导入站点风格安装包。

1. 在管理界面中选择“设置－安装包”，然后点击“导入新安装包”按钮（参阅下图）。



2. 从您的电脑上选择需要的".ezpkg"文件（如下图）并点击“导入安装包”按钮。



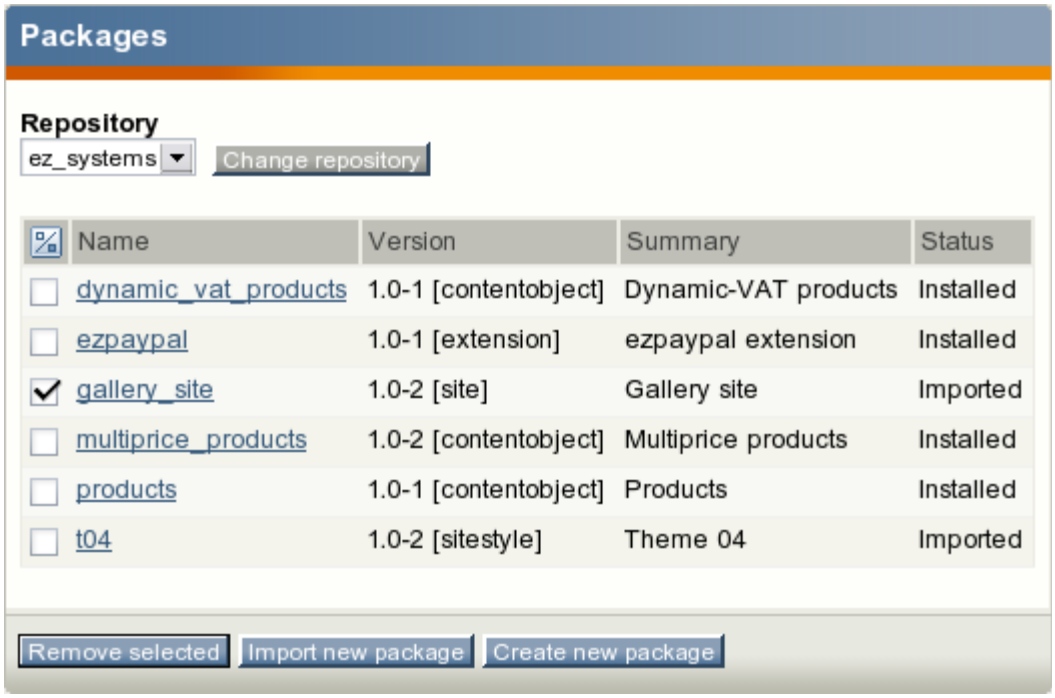
系统会从".ezpkg"文件中导入安装包并显示安装包简介。

请参阅“改变站点风格”章节了解如何应用导入的界面风格。

4.6.5 删除安装包

以下步骤演示了如何从系统软件仓库中删除安装包。

1. 在管理界面中选择“设置－安装包”，然后选择一个内部软件仓库并点击“改变软件仓库”按钮。
2. 选择希望删除的安装包（如下图）然后点击“删除所选”按钮。



注意:

如果您删除一个已经安装的安装包，它不会被卸载。只有安装包文件会被从内部软件仓库中删除。

4.6.6 安装安装包

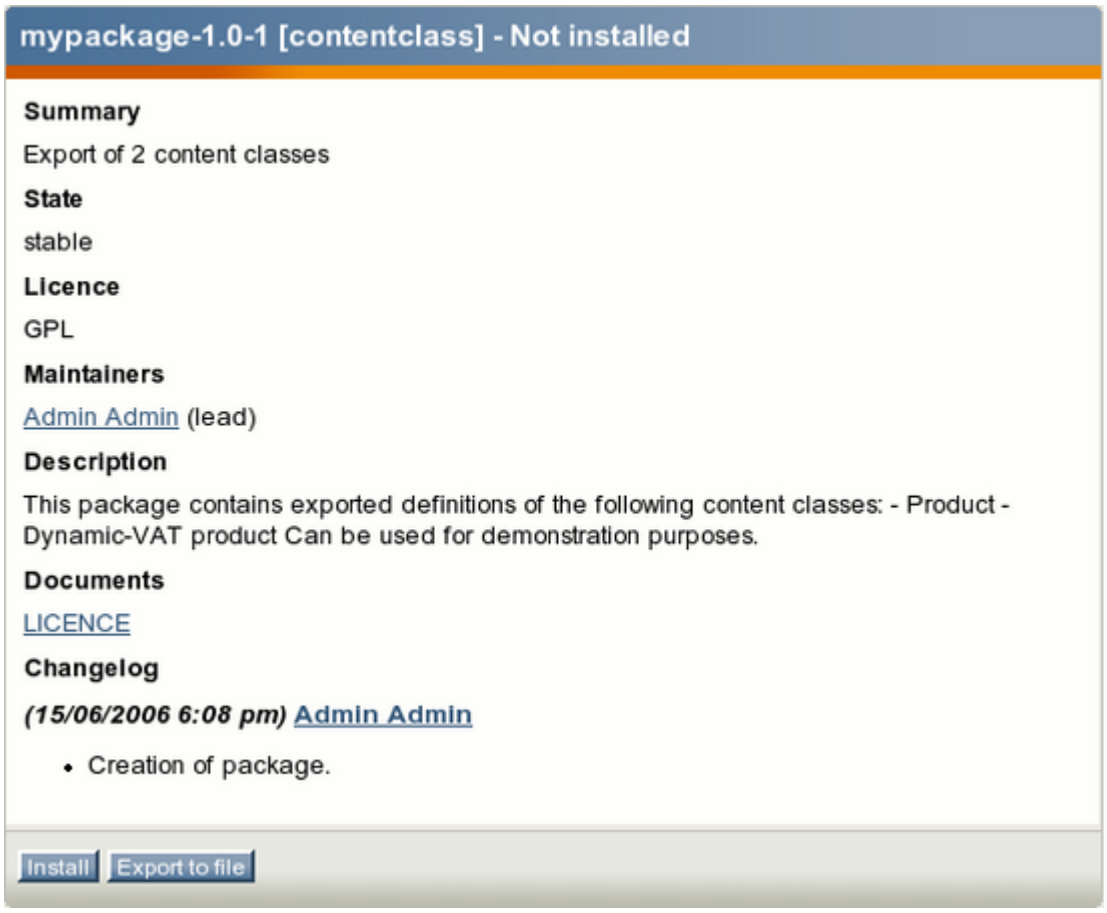
在内部软件仓库中的安装报可以被安装。注意，不能安装站点安装包和界面安装包。在安装安装包时，系统会创建内容类，内容对象，应用配置等等。注意，从 3.8 版本开始，与安装包有关的信息会被保存在"ezpackage"数据库表中。

下面的章节解释了如何安装不同类型的安装包。

内容类安装包

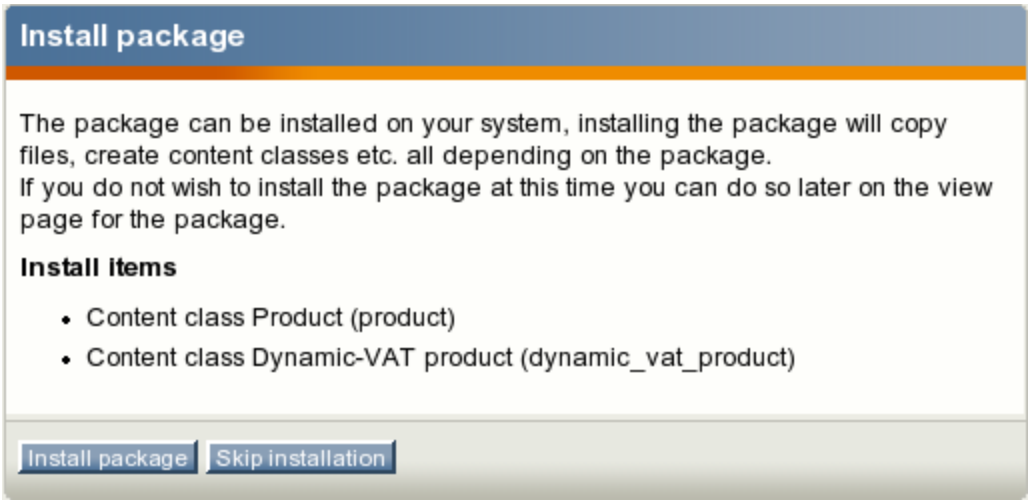
下例演示了如何安装内容类安装包。

1. 在管理界面中选择“设置－安装包”，选择要使用的软件仓库，然后点击“改变软件仓库”按钮。找到需要的安装包，然后点击它的名字。系统会显示安装包的简介（如下图）。



点击“安装”按钮。

2. 系统会显示安装包的项目（如下图）。



仔细阅读这些信息然后点击“安装安装包”按钮。如果要取消安装，点击“跳过安装”按钮。

3. 如果某些类已经存在，系统会提示用户如何解决冲突（如下图）。

如果您希望用新的类替换现存类（注意，旧类对应的所有的对象会被删除）。只有您了解您在做什么，否则不要使用这种方法。其余的选项允许您跳过类安装或创建新类（两种情况下，现存的类和对象都不会被修改）。点击“继续”按钮后，系统会安装安装包并显示简介。

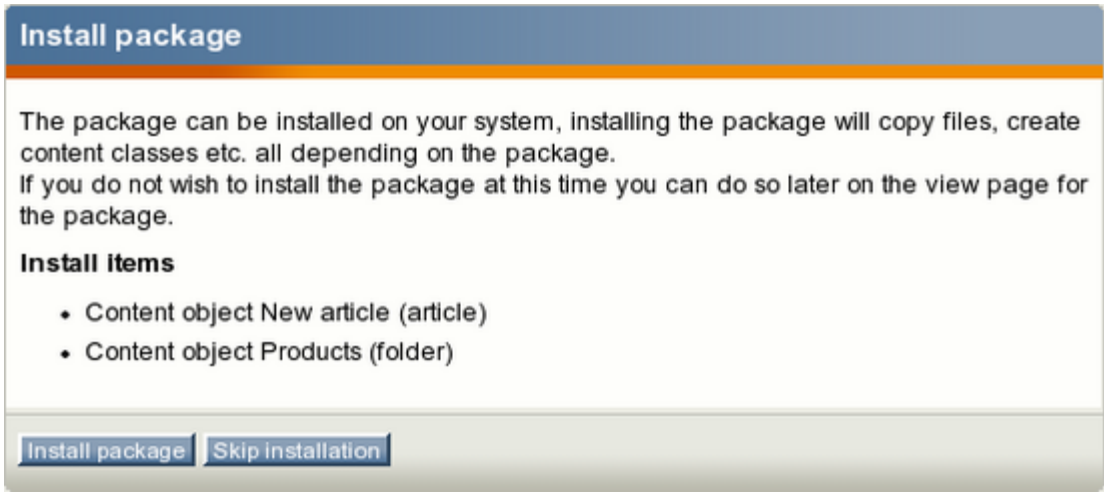
内容对象安装包

下例演示了如何安装内容对象安装包。（因为处理类冲突的步骤已在上例中解释过，本例中假设没有类冲突。）

1. 在管理界面中选择“设置－安装包”，选择要使用的内部软件仓库，然后选择“变更软件仓库”按钮。找到要

安装的安装包，点击名称然后点击“安装”按钮。

2. 系统会显示安装包的内容（如下图）。

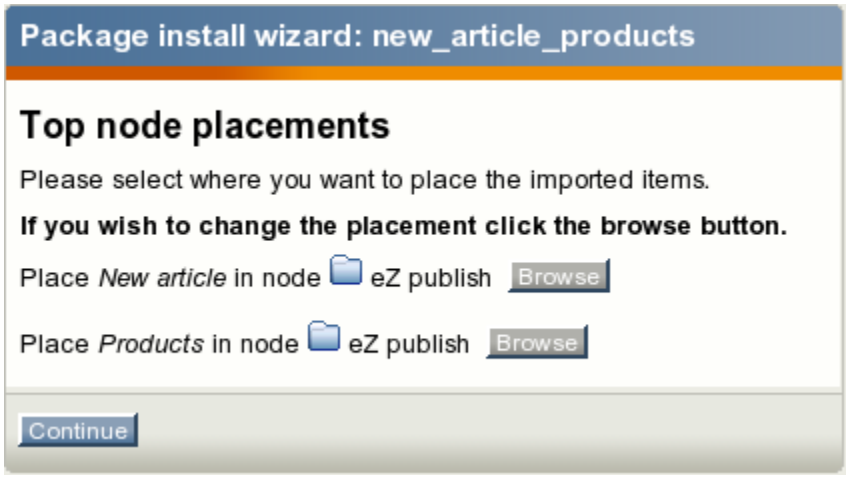


仔细阅读这里的信息然后点击“安装”按钮。如果要取消安装，点击“跳过安装”按钮。

3. 如果安装包除了包含对象外，还包含这些对象的模板，系统会询问您这些模板应该被安装到哪个站点入口（参阅下图）。

选择选项，然后点击“下一步”按钮。

4. 下一个对话框揭示了对象会被安装到什么位置，同时也允许您选择其它安装位置。



选择需要的位置然后点击“继续”按钮。

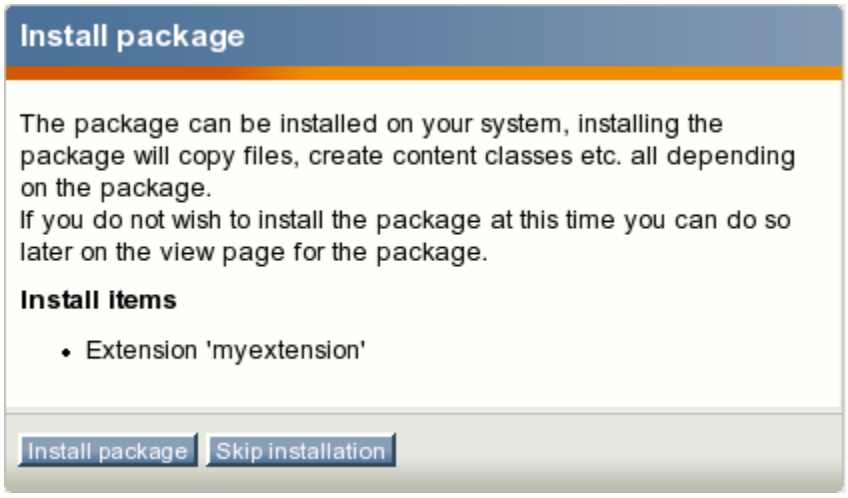
5. 如果某些被安装的对象已经存在（例如：如果与另外一个对象有相同的 remote_id），系统会询问如何解决冲突（如下图）。

点击“继续”按钮后，系统会安装安装包并显示简介。

扩展安装包

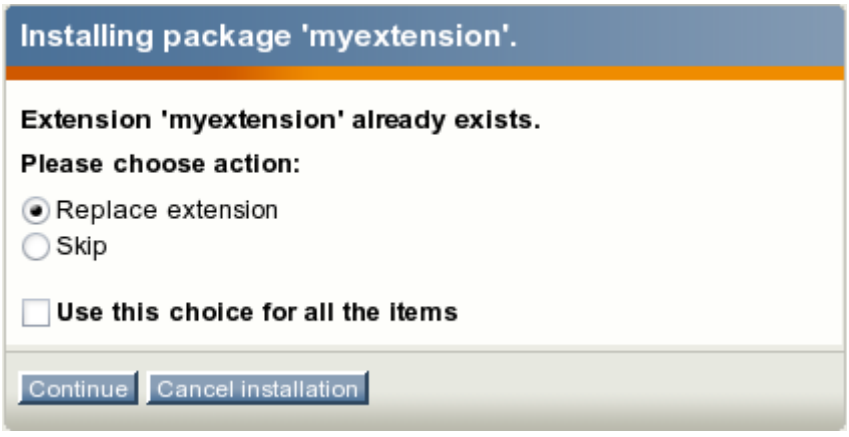
下例演示了如何安装扩展安装包。

1. 在管理界面中选择“设置－安装包”，选择要使用的软件仓库，然后点击“变更软件仓库”按钮。找到要安装的安装包，点击名称，然后点击“安装”按钮。
2. 系统会显示安装包的内容（如下图）。



点击“安装安装包”按钮。如果要取消安装，点击“跳过安装”按钮。

3. 如果某些项目已经存在，系统会询问如何解决冲突（如下图）。

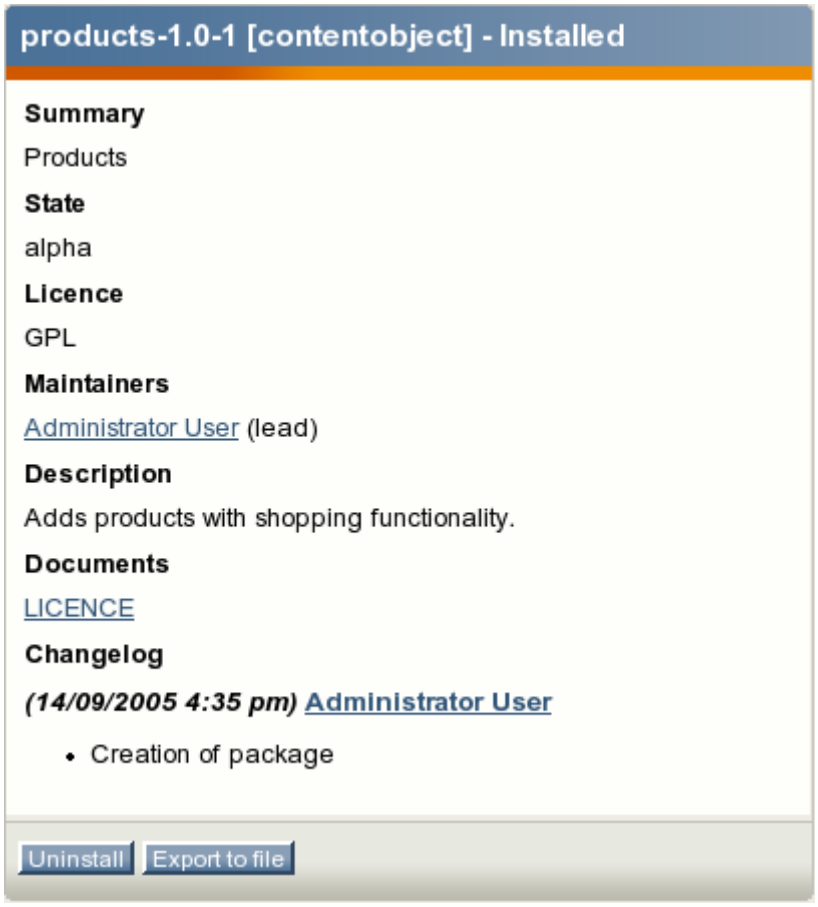


点击“继续”按钮后，系统会安装安装包并显示简介。

4.6.7 卸载安装包

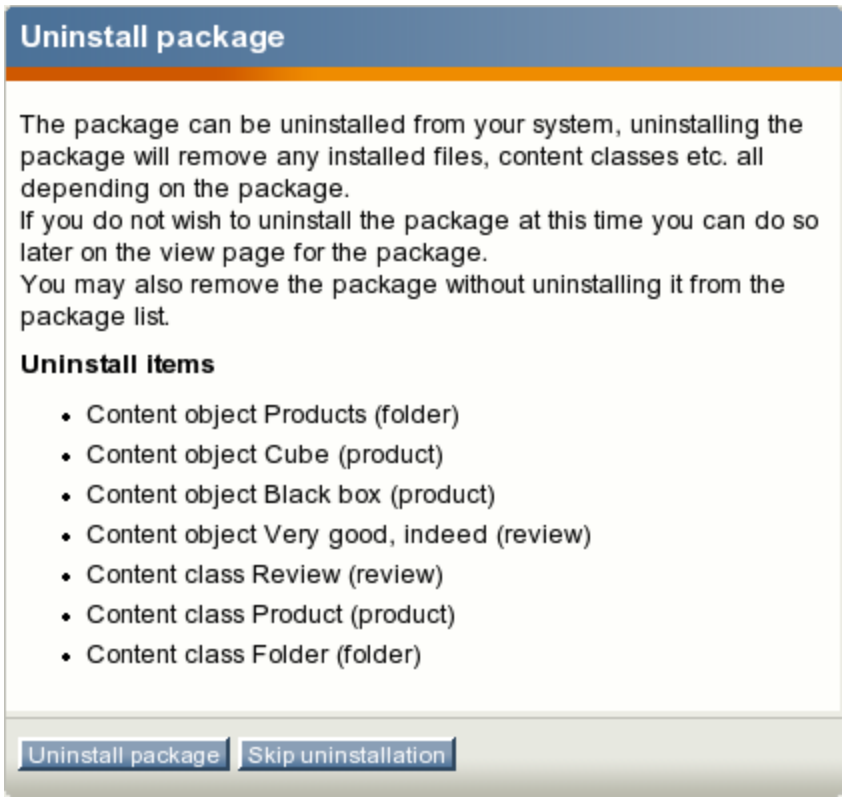
以安装的安装包可以被卸载。例如：假设您已经通过安装向导安装了"Shop"标准站点安装包，然后您决定只保留"multi-price products"，而不要简单价格商品。在本例中，您可以安全地卸载"Products"安装包（这个内容类安装包包含简单价格商品）。以下步骤演示了如何卸载。

1. 在管理界面中选择“设置－安装包”，选择"ez_systems"软件仓库然后点击“变更软件仓库”按钮。找到要卸载的安装包然后点击名称。系统会显示安装包简介（如下图）。



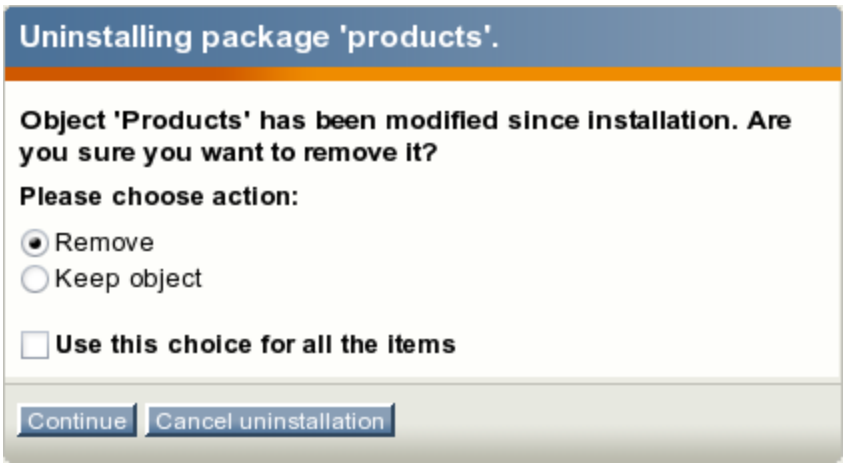
点击“卸载”按钮

2. 系统会显示要被删除的项目（如下图）。



点击“卸载安装包”按钮或点击“跳过卸载”取消卸载。如果列表中的某个项目不能被删除，卸载流程会自动终止。例如：“文件夹”类不能被删除，如果它被某个定级节点使用。

3. 如果某些被删除的项目在安装之后被修改过，系统会在删除它们之前询问您（如下图）。



您可以选择保留或删除这些项目。选择后点击“继续”按钮。

4. 系统会显示简介。注意，被卸载的安装包仍然会被保留在系统软件仓库中（它们会被标记为“已经导入”）并且可以被再次安装。

4.6.8 package.xml 格式

本章描述了安装包系统所使用的重要的 XML 标签。这些标签在安装包目录中的"package.xml"中。

安装项目

安装包安装流程由安装项目决定。这总是在<install>和</install>标签中指定。（注意，不允在在"package.xml"文件中使用两个<install>标签。）

安装项目由<item>标签定义。下表揭示了这个标签的属性。

属性	值
type	项目类型("ezcontentclass","ezcontentobject","ezextension","ezinstallscript"等。)
filename	包含安装项目信息的 xml 文件名（不包含文件扩展名）。
sub-directory	包含安装项目 xml 文件的子目录名。

例

假设一个类安装包包含三个类"myarticle","myfolder"和"myproduct"。这些类的 xml 文件为："class-myarticle.xml",“class-myfolder.xml"和"class-myproduct.xml"。这些 xml 文件被保存在"myclassdir"子目录中。在本例中，"package.xml"将包含以下内容：

```
<install>
<item type="ezcontentclass"
            filename="class-myarticle"
            sub-directory="myclassdir" />
<item type="ezcontentclass"
            filename="class-myfolder"
            sub-directory="myclassdir" />
<item type="ezcontentclass"
            filename="class-myproduct"
            sub-directory="myclassdir" />
```

```
</install>
```

这会要求系统按照以下顺序安装三个"ezcontentclass"类型的项目：

- myarticle
- myfolder
- myproduct

卸载项目

安装包卸载流程由卸载项目决定。卸载项目在<uninstall>和</uninstall>标签中。（注意，"package.xml"中不能有两个<uninstall>标签。）

卸载项目由<item>标签定义。

依赖安装包

一个站点安装包通常会依赖于其它安装包。在安装向导中选择一个站点安装包，系统会下载，导入并安装它的依赖包。安装包的依赖包总是在<dependencies>和</dependencies>中的<requires>和</requires>标签中。

依赖包通过<requires>标签定义。下表揭示了标签的属性。

属性	值
type	安装包类型(通常为"ezpackage")
name	安装包的内部名。
min-version	安装包的最低版本。

例

"News"站点安装包的"package.xml"包含以下内容：

```
<dependencies>
    <provides />
    <requires>
      <require type="ezpackage"
               name="news"
               min-version="1.0" />
      <require type="ezpackage"
               name="media"
               min-version="1.0-3" />
```

```
      <require type="ezpackage"
               name="t01"
               min-version="1.0" />
    </requires>
    <obsoletes />
    <conflicts />
  </dependencies>
```

这意味着"News"安装包依赖于以下安装包：

- news（1.0 以上版本）
- media（1.0-3 以上版本）
- t01（1.0 以上版本）

在安装向导中选择"News"站点安装包，系统会下载这个站点安装包以及它的依赖包。除了站点风格安装包外，所有的依赖包都会被自动安装。

4.6.9 自定义安装脚本

可以通过管理界面安装的安装包可以包含特定的自定义安装和卸载脚本。一个自定义的脚本在安装/卸载流程的任何位置为调用。这些脚本可以是互动脚本且可以显示附加的安装向导步骤。互动脚本基于"eZPackageInstallationHandler"的“安装步骤”机制。

下例演示了如何实现一个自定义安装脚本。

例

假设您需要对内容对象安装包"Products"（位于"ez_system"软件仓库，"var/storage/packages/ez_systems/products"）开发一些安装后步骤。要实现一个安装后互动脚本，参阅如下步骤：

1. 在 package 目录中创建以下子目录
   - post-install
   - post-install/templates

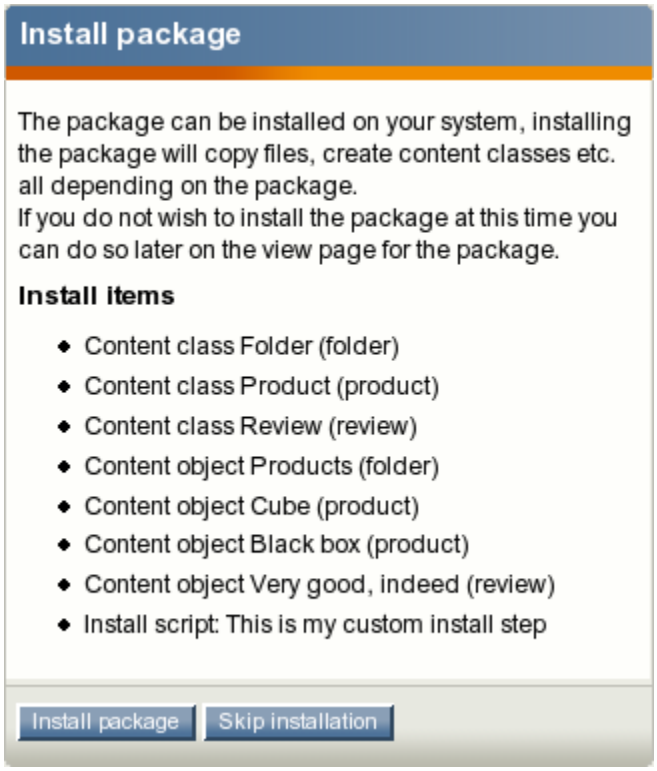
2. 编辑"package.xml"并在<install>和</install>中添加如下内容：

```
<item type="ezinstallscript"
      filename="myinstallscript"
      sub-directory="post-install" />
```

3. 在"post-install"中创建"myinstallscript.xml"（这个文件必须包含您安装脚本的简介）并在其中添加如下内容：

```
<?xml version="1.0" encoding="UTF-8"?>
<install-script filename="myinstallscript.php"
                classname="myInstallScript"
                description="This is my custom install step" />
```

这会告诉系统"post-install/myinstallscript.php"中的 PHP 类"myInstallScript"实现了附加的安装步骤。安装脚本的介绍文本会在安装包流程开始时显示（如下图）。

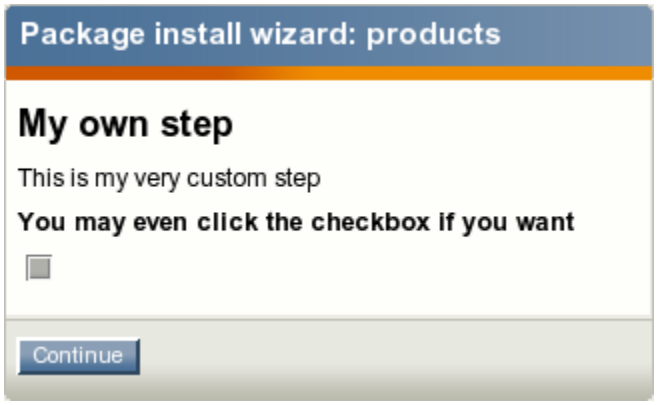


在"post-install/templates"中创建"myownstep.tpl"（用于您的安装脚本的模板）并添加如下代码：

```
<form method="post" action={'package/install'|ezurl}>

    {include uri="design:package/install/error.tpl"}
    {include uri="design:package/install_header.tpl"}

    <p>This is my custom step</p>
```

```
    <label>You may even click the checkbox if you want</label>

    <div class="block">
      <input class="button" type="checkbox" name="MyCheckBox" />
    </div>

    {include uri="design:package/navigator.tpl"}

</form>
```

安装包的最后一步会显示这个模板（如下图）。



在"post-install"目录中创建"myinstallscript.php"。这个文件必须包含"myInstallScript"PHP 类，它实现了所有的附加安装步骤（根据"post-install/myinstallscript.xml"文件中的描述）。并在"myinstallscript.php"中添加如下代码：

```
<?php
class myInstallScript extends eZInstallScriptPackageInstaller
{
  function myInstallScript( &$package, $type, $installItem )
    {
        eZDebug::writeDebug( $installItem, "Hello from myInstallScript" );

        $steps = array();
        $steps[] = array(
          'id' => 'my_own_step',
          'name' => 'My own step',
          'methods' => array( 'initialize' => 'initializeMyOwnStep',
               'validate' => 'validateMyOwnStep',
```

```
                'commit' => 'commitMyOwnStep' ),
          'template' => 'myownstep.tpl' );
        $this->eZPackageInstallationHandler( $package,
                                             $type,
                                             $installItem,
                                             'My own custom step',
                                             $steps );
    }

  // Function that is called before the step is displayed.
  // You can use it to set variables for your template.

  function initializeMyOwnStep( &$package, &$http, $step, &$persistentData, &$tpl,
&$module )
    {
        eZDebug::writeDebug( "Hello from initializeMyOwnStep()" );
        return true;
    }

  // This function is called after user has submitted the form.
  // If this function returns "false", the step will be
  // displayed again.

  function validateMyOwnStep( &$package, &$http, $currentStepID, &$stepMap,
&$persistentData, &$errorList )
    {
        eZDebug::writeDebug( "Hello from validateMyOwnStep()" );
        return true;
    }

  // This function is called after the form is submitted
  // and validated.

  function commitMyOwnStep( &$package, &$http, $step, &$persistentData, &$tpl )
    {
        eZDebug::writeDebug( "Hello from commitMyOwnStep()" );
        return true;
```

```
    }
}
?>
```

4.7 Cronjob

某些 eZ Publish 的特性需要依赖于一个运行于后台的脚本。这个脚本位于 eZ Publish 根目录且应该按照一定时间间隔被反复执行。这个脚本是"runcronjobs.php"。除其它功能外，它负责处理工作流，检查/验证 URL，发送通知邮件等等。尽管如果"runcronjobs.php"不被定期执行，eZ Publish 也可以使用，还是建议在后台运行这个脚本。某些特性，如“通知”系统必须需要这个脚本。

最常用的解决方案是要求系统每隔 30-60 分钟便自动执行这个脚本。但是，某些任务应该执行得更频繁。因此，建议将 cronjob 分成不同的组并分别执行它们。请参阅“配置 cronjob ”和“运行 cronjob ”章节了解更多。

4.7.1 Cronjob 脚本

"cronjobs"目录包含各种 cronjob，它们被用于各种自动的周期和计划维护任务。如下表所示。

名称	描述	频率	默认状态
basket_cleanup.php	清除被删除的用户会话的购物篮。	每周。	启用
hide.php	当当前时间超过指定的日期/时间后，隐藏特定的节点。	每天不少于一次。	启用
indexcontent.php	为新创建的对象建立非实时索引。	每天不少于一次。	启用
old_drafts_cleanup.php	删除旧的/不用的草稿。	每月一次。	禁用
internal_drafts_cleanup.php	删除不用的内部草稿。	每天一次。	启用
ldapusermanage.php	与 LDAP 服务器同步用户帐号信息。	每天不少于一次。	禁用
linkcheck.php	验证发布的 URL。	每周一次。	启用
notification.php	向订阅的用户发送通知。	每 15-30 分钟	启用
rssimport.php	导入 RSS。	每天不少于一次。	启用
subtreeexpirycleanup.php	删除相对于"subtree_expiry"过期的缓存块。	每天不少于一次。	启用
unpublish.php	当系统时间超过指定的日期/时间时，删除特定的对象。	每天不少于一次。	启用
updateviewcount.php	通过解析 Apache 日志文件来更新页面的访问量。	每天不少于一次。	禁用
workflow.php	处理工作流	每 15-30 分钟。	启用

为网络商店清除过期数据

eZ Publish 网络商店功能允许用户将商品放入购物篮。购物篮中的商品可以通过启动结帐流程来购买。系统在数据库表"ezbasket"中保存用户的购物篮信息。与用户会话有关的信息被保存在"ezsession"表中。

如果用在购物篮中添加了商品然后终止了购物流程（例如：关闭了浏览器）而没有结帐，用户的会话在一段时间后会过期。过期的会话可以被自动或被管理员手动删除。当系统删除用户会话时，系统不会处理与这个会话对应的购物篮。换言之，系统会从"ezsession"中删除一条记录，但是"ezbasket"中的对应记录（如果有）不会被删除。这种行为由"site.ini"重设文件中"[Session]"下的"BasketCleanup"配置。如果它被设置为"cronjob"（默认值），您将需要通过"basket_cleanup.php"来删除不用的购物篮。

请注意，对于一个有很多用户的站点而言，删除不用的购物篮通常会耗费很多时间。建议每周执行一次购物篮清理脚本。如果您希望与其它（更频繁）的脚本一起使用，可以配置"BasketCleanupAverageFrequency"来指定当"basket_cleanup.php"被执行时，购物篮实际清除的频度。如够您希望将"basket_cleanup.php"与其它脚本分开执行，则配置"BasketCleanupAverageFrequency"如下：

```
BasketCleanupAverageFrequency=1
```

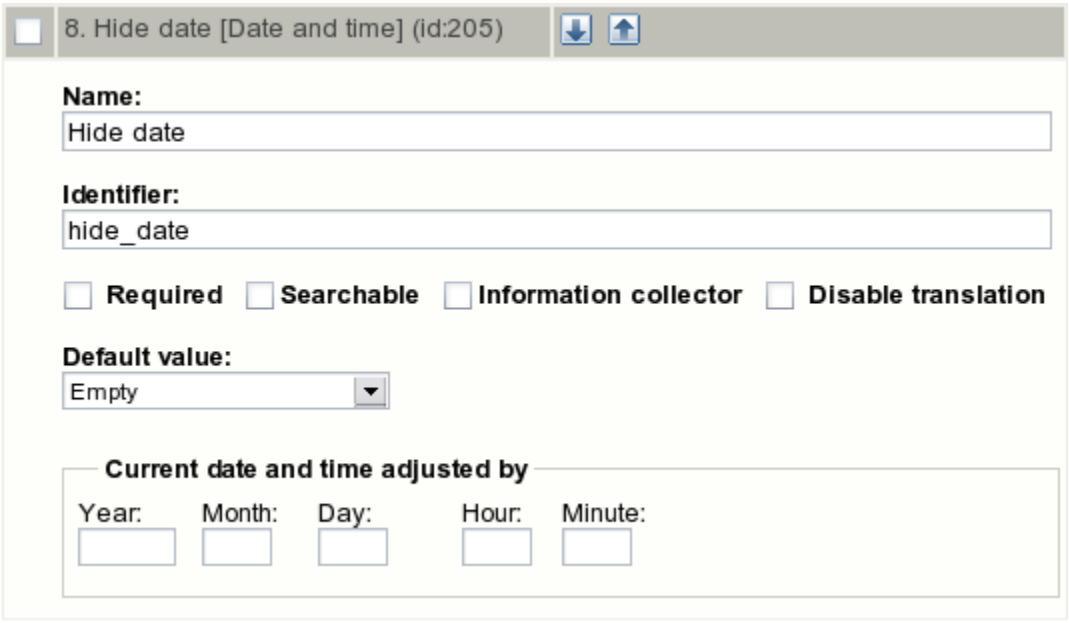
请注意，如果您的站点不使用网络商店功能，没有必要运行这个脚本（或者执行它但是配置"BasketCleanup"为"pageload"）。

在特定时间隐藏节点

当系统时间超过指定的时间后，系统可以自动隐藏特定节点。例如：如果您希望发布的一篇文章在一天/周/月后被隐藏。但是，您不想删除这篇文章，您只是希望隐藏这篇文章。在这种情况下，你需要为您的文章类添加一个日期时间类型的属性并配置隐藏 cronjob。以下文字描述了如何做到。

为"文章"类添加一个属性

在管理界面中选择"设置－类"然后选择"内容"类组并显示这一组内的类。找到文章类然后点击编辑按钮。系统会显示类编辑界面。选择"日期时间"数据类型，点击"添加属性"按钮并做如下编辑。



注意，属性名和属性标识符可以随意指定（参阅以下内容）并点击“确定”按钮。系统会为类添加一个新属性"Hide date"（新添加的属性名称），因而它会在对象编辑界面中显示。当编辑文章时，这个字段可以被用来指定何时文章应该被隐藏。如果属性被设为空，文章不会受隐藏 cronjob 影响。请注意，"hide.php"必须定期执行。

配置"hide" cronjob

在"content.ini.append.php"中添加如下内容：

```
[HideSettings]
RootNodeList[]=2
HideDateAttributeList[article]=hide_date
```

您应该在"HideDateAttributeList"中添加新添加的属性标识符并用类标识符作为键值。此外，您需要在"RootNodeList"中指定文章的父节点 ID。

推迟的检索索引

如果“推迟的索引”特性被启用，新的和重新发布的对象不会被立刻编入索引。换言之，eZ Publish 不会在发布过程中为内容建立索引。相反，索引 cronjob 会在后台建立索引，因而发布过程会变得更快（因为您不需要等待内容被编入索引）。要启用推迟的索引，必须在后台执行"indexcontent.php" cronjob（否则内容会被发布但是永远不会被编入索引）。

请注意，您不需要执行这个脚本除非您已经在"site.ini"的重设文件的"[SearchSettings]"中设置了"DelayedIndexing=enabled"。

清除旧的/不用的草稿

一般草稿（状态"0"）

"old_drafts_cleanup.php"脚本的用途是从数据库中删除旧的草稿。如果启用，这个脚本会删除系统中超过 90 天的草稿。要设置草稿过期的时间，在"content.ini"重设文件中"[VersionManagement]"下配置"DraftsDuration[]"。每次脚本执行最多可以删除草稿数量（默认值 100）由"DraftsCleanUpLimit"指定。

内部/未处理草稿（状态"5"）

"internal_drafts_cleanup.php"的用途是删除那些可能永远都不会被发布的草稿。如果一个对象的版本被创建但是没有被改变（例如：如果某个用户点击了“添加注释”按钮但是没有真正提交任何内容），版本的状态为"5"。"internal_drafts_cleanup.php"脚本会删除系统中超过 24 小时（一天）的内部草稿。要设置内部草稿的过期时间，可以修改"content.ini"重设文件中"[VersionManagement]"下的"InternalDraftsDuration[]"。脚本每次最多可以删除的内部草稿数（默认值 100）由"InternalDraftsCleanupLimit"控制。

与 LDAP 服务器同步用户帐号信息

如果通过 LDAP 服务器验证用户，eZ Publish 会从外部数据源提取用户帐户信息并保存在本地数据库。系统会在用户成功登录后，在本地创建帐号。

"ldapusermanage.php"脚本可以用来与 LDAP 服务器同步用户帐号信息。建议站点与 LDAP 服务器连接后，定期执行这个脚本。这个脚本会负责典型的维护任务。例如：如果用户被从 LDAP 服务器删除，eZ Publish 的帐号会被禁用。

请注意，这个脚本只会更新 eZ Publish 数据库。不支持对外部数据（保存在 LDAP 服务器中）的修改。

检查 URL

在 eZ Publish 中，每个在 XML 块中输入的链接或在"URL"数据类型中输入的链接都会被保存在 URL 表中，因此这些发布的 URL 可以被调查，编辑而不需要处理对象。这意味着，您不需要编辑或重新发布对象，如果您只是想修改 URL。

URL 表包含所有关于地址的信息，包括：状态（有效或无效）和最后检查时间（系统最近一次检查 URL 的时间）。默认情况下，所有的 URL 都是有效的。"linkcheck.php"脚本通过访问 URL 表中的每个 URL 来逐个检查这些链接。如果发现不可访问的链接，链接的状态被设置为“无效”。最后检查时间总会被更新。

你必须在"cronjob.ini.append.php"中的"[linkCheckSettings]"配置"SiteURL"来指定您站点的 URL。这会确保"linkcheck.php"可以正确处理相对 URL（内部链接）。

请注意，链接检查脚本必须可以通过 80 端口与外部网站通讯。换言之，火墙必须允许 80 端口的 HTTP 出栈数据。从 3.9 版本开始，可以通过 HTTP 代理服务（在"site.ini"重设文件中的"[ProxySettings]"配置）器获得数据（PHP 需要支持 CURL）。

发送通知

eZ Publish 内建的通知系统允许用户接收不同时间的信息。当对象被修改或发布时，当工作流被执行等等。如果您需要在您的站点使用通知功能，您需要定期运行"notification.php"脚本。它会负责向信息订阅者发送通知（这是通过启动主通知处理脚本"kernel/classes/notificationeventfilter.php"来完成的）。

如果您使用通知系统，建议每 15-30 分钟执行一次这个脚本。

RSS 导入

RSS 导入功能允许从不同站点接收 RSS 数据。例如：来源于 ez.no 的最新的社区新闻(http://ez.no/rss/feed/communitynews)。您需要在管理界面中用“设置－RSS　”来配置 RSS 导入。然后定期运行"rssimport.php"脚本。这个脚本会从您的活动 RSS 导入获得最新的项目并将它们发布到您的站点（如果项目已经存在，则逃过这个项目）。

请注意，RSS 导入脚本必须可以通过 80 端口与外部网站通讯。换言之，火墙必须允许 80 端口的 HTTP 出栈数据。从 3.9 版本开始，可以在"site.ini"重设文件中的"[ProxySettings]"中配置 HTTP 代理服务器（需要 PHP 支持 CURL）。

清除过期的模板缓存块

如果您使用模板"cache-block"功能，且指定了"subtree_expiry"参数，缓存块只有在指定的子树（而不是整个内容树）下有对象被发布才会过期。"site.ini"重设文件中"[TemplateSettings]"下的"DelayedCacheBlockCleanup"用来控制使用"subtree_expiry"的缓存块是否应该立刻被删除。如果这个设置被启用，过期的缓存块必须用"subtreeexpirycleanup.php"来清除。

在指定时间删除对象

当系统时间超过指定时间时，"unpublish.php"脚本可以用来删除特定的对象。例如：您可能希望在一天/周/月后删除某些文章。参阅如下步骤。

1. 为"文章"类添加一个日期时间数据类型的属性。用"unpublish_date"作为属性标识符；这个属性在编辑文章时候会显示。这个字段可以被用来指定何时删除对象。如果这个属性为空，对象不会受这个脚本的影响。
2. 在"content.ini.append.php"中做如下配置：

```
[UnpublishSettings]
RootNodeList[]=2
ClassList[]=2
```

您应该在"ClassList"中指定文章类的 ID 并将文章父节点 ID 添加到"RootNodeList"中。

分析 Apache 日志文件

可以通过分析 Apache 日志文件来得到站点页面的页面访问量，并把结果保存在 eZ Publish 中。您需要定期运行"updateviewcount.php"。这个脚本会通过分析 Apache 日志文件来更新每个节点的访问量（访问量被保存在"ezview_counter"表中）。当执行脚本时，它会更新"var/example/log/updateview.log"（用"site.ini"重设文件中的"varDir"代替 example）。这个文件包含的信息告诉脚本下次执行时应从 Apache 日志的哪一行开始。

您需要重设"logfile.ini"并添加如下配置：

```
[AccessLogFileSettings]
StorageDir=/var/log/httpd/
LogFileName=access_log
SitePrefix[]=example
SitePrefix[]=example_admin
```

用 apache 日志目录替换"/var/log/httpd"，用实际的日志文件替换"access_log"，用实际的站点入口名替换"example"和"example_admin"（如果您有超过两个站点入口，将它们全部列出）。

一旦您指定了正确的配置且定期执行"upateviewcount.php"脚本，您可以用"view_top_list"模板函数提取最流行（最高访问量）的节点或检查一个节点被访问了多少次。

处理工作流

要使用工作流，必须定期执行"workflow.php"。这个脚本会处理工作流。例如：假设您使用协作系统且所有在标准分区中的改动必须经过您的批准才能发布（可以在一个由"content-publish-before"触发器触发的工作流中创建一个“审批”事件来实现）。如果某个用户（除站点管理员外）修改了文章 A，系统会为您生成一个新的协作消息“文章 A 等待您的批准”，并为编辑者生成一个消息“文章 A 等待编辑的批准”。（定期运行"notification.php"来通过电子邮件向用户发送新的协作消息。）您可以在管理界面中“我的帐号－协作”中查看协作消息并查看/批准/拒绝对内容的修改。但是，在您批准了对文章的修改后，修改不会立刻被应用。修改会在下次运行"workflow.php"时生效。

4.7.2 配置 cronjob

您可以在"settings/cronjob.ini"的重设文件中配置哪些 cronjob 会被启用（可以被"runcronjobs.php"执行）。以下列表揭示了在"[CronjobSettings]"中可以使用的配置选项。

- ScriptDirectories 数组指定 eZ Publish 会在哪些位置搜索内建 cronjob 脚本（默认使用"cronjobs"目录）。
- ExtensionDirectories 数组指定 eZ Publish 会在哪些位置搜索附加的/自定义的 cronjob 脚本。默认情况下，eZ Publish 会在您的扩展中的"cronjobs"子目录中搜索。
- Scripts 数组包含了一组 cronjob 脚本，如果在执行"runcronjobs.php"脚本时不指定"group_of_tasks"选项，这些脚本会被执行。在这个数组中的脚本被称为主脚本。

cronjob parts

某些 cronjob 必须比其它脚本更频繁地执行。可以在"cronjob.ini"重设文件中添加特定的 cronjob parts 来配置附加的 cronjob 组。

下例演示了如何配置。

例 1（默认配置）

以下配置是在"cronjob.ini"中的"[CronjobSettings]"的默认配置：

```
[CronjobSettings]
ScriptDirectories[]=cronjobs
Scripts[]=unpublish.php
Scripts[]=rssimport.php
Scripts[]=indexcontent.php
Scripts[]=hide.php
Scripts[]=subtreeexpirycleanup.php
Scripts[]=internal_drafts_cleanup.php
ExtensionDirectories[]
```

这意味着主脚本包含以下六个任务：

- unpublish.php

- rssimport.php
- indexcontent.php
- hide.php
- subtreeexpirycleanup.php
- internal_drafts_cleanup.php

如果不指定"group_of_tasks"选项，这些脚本会在每次执行"runcronjobs.php"脚本时被执行。系统会期望这些脚本被放置在"cronjobs"目录中。

以下的 cronjob parts 定义了两个 cronjob 的附加组："infrequent"和"frequent"：

```
[CronjobPart-infrequent]
Scripts[]=basket_cleanup.php
Scripts[]=linkcheck.php


[CronjobPart-frequent]
Scripts[]=notification.php
Scripts[]=workflow.php
```

在"[CronjobPart-infrequent]"中的配置要求系统用如下选项执行"runcronjobs.php"时执行"basket_cleanup.php"和"linkcheck.php"：

```
php runcronjobs.php infrequent
```

在"[CronjobPart-frequent]"中的配置要求系统用如下选项执行"runcronjobs.php"时执行 notification 和 workflow 脚本：

```
php runcronjobs.php infrequent
```

通过以上配置，可以独立运行每一组脚本，如：

- workflow.php 和 notification.php － 每 15 分钟
- basket_cleanup.php 和 linkcheck.php － 每周
- 主脚本 － 每天

例 2

如果您希望每个月执行"old_drafts_cleanup.php“，您可以在"settings/siteaccess/example/cronjob.ini.append.php“中添加如下配置：

```
[CronjobPart-monthly]
Scripts[]=old_drafts_cleanup.php
```

可以用如下命令行运行脚本：

```
php runcronjobs.php monthly -s example
```

例 3

可以开发自定义脚本扩展系统。例如：如果您有一个扩展"nExt"包含一个脚本"myjob.php"，您可以在"cronjob.ini"重设文件中添加如下配置：

```
[CronjobSettings]
ExtensionDirectories[]=nExt
Scripts[]=myjob.php
```

或

```
[CronjobSettings]
ScriptDirectories[]=extension/nExt/cronjobs
Scripts[]=myjob.php
```

以上配置会要求 eZ Publish 把"extension/nExt/cronjobs/myjob.php"作为一个 cronjob 脚本。这个脚本会被添加到主脚本中并且每次不指定"group_of_tasks"选项执行"runcronjobs.php"时，这个脚本都会被执行。

4.7.3 运行 cronjob

在 eZ Publish 根目录中的"runcronjobs.php"负责在后台执行您的 cronjob。这个脚本应该被定期执行。最常用的解决方案是要求操作系统（或某些应用程序）按固定的时间间隔自动执行这个程序。在 Linux/UNIX 操作系统中，这可以通过使用"cron"来实现。在 Windows 上，可以用“计划任务”系统服务来定期执行这个脚本。

以下文字描述了如何执行这个脚本。

在命令行中执行 cronjob

可以在命令行中手动执行"runcronjobs.php"脚本：

1. 进入 eZ Publish 根目录
2. 运行脚本（用实际的站点入口名替换"example"）：

```
php runcronjobs.php group_of_tasks -s example
```

"group_of_tasks"选项表示只有在"cronjob.ini"重设文件中"[CronjobPart-group_of_tasks]"组内的脚本会被执行。如果省略这个选项，"cronjob.ini"重设文件中"[CronjobSettings]"下的脚本会被执行。（请参阅“配置 cronjob”一章了解更多。）

"-s example"表示这个脚本应该对哪个站点入口执行。如果您执行脚本的时候不指定站点入口，默认的站点入口会被使用。

也可以用"-d"选项要求脚本在结束时显示调试信息，例如：

```
php runcronjobs.php group_of_tasks -d -s example
```

您也可以用"all"选项来获得更多的调试信息：

```
php runcronjobs.php group_of_tasks -dall -s example
```

您还可以使用以下选项："accumulator","debug","error","include","notice","timing","warning"。请注意，这些选项需要用","分隔。下例会告诉脚本显示调试注意信息并生成 include 列表：

```
php runcronjobs.php group_of_tasks -dinclude,notice -s example
```

默认情况下，这个脚本不会修改任何日志文件（"var/log"目录中的文件）。如果您需要记录日志，您可以使用"-d"和"—logfiles"选项：

```
php runcronjobs.php group_of_tasks -d -s example --logfiles
```

在 linux/UNIX 系统中运行 cronjob

"cron"是系统中在后台运行任务的工具。它主要用来定期自动执行系统管理与维护的任务（例如：创建每周的备份）。被称为"cron daemon"（cron 精灵）的程序通常在后台安静地等待执行间隔到达，然后执行 cronjob。一个"cronjob"是由 cron daemon 定期执行的一个脚本或一个程序。cronjob 必须在 crontab 中设置。crontab 是一个文本文件，它不能直接被编辑。下表揭示了哪些命令可以用来维护 crontab。

shell 命令	描述
crontab /var/www/ezpublish/ezpublish.cron	从"ezpublish.cron“文件中安装新的 crontab。（用 eZ Publish 的目录替换"/var/www/ezpublish"）。
crontab -l	显示当前 crontab
crontab -e	编辑当前 crontab。修改后的 crontab 会被自动安装。
crontab -r	删除当前 crontab。

下例演示了如何在 crontab 中设置 eZ Publish 的 cronjob。它假设 eZ Publish 被安装在"/var/www/ezpublish/“，PHP 命令行接口在"/usr/local/bin/php"且目标站点入口名称为"example"。

```
# The path to the eZ Publish directory.
EZPUBLISH=/var/www/ezpublish


# Location of the PHP command line interface binary.
PHPCLI=/usr/local/bin/php


# Instruct cron to run the main set of cronjobs
# at 6:35am every day
35 6 * * * cd $EZPUBLISH && $PHPCLI runcronjobs.php -q -s example 2>&1


# Instruct cron to run the "infrequent" set of cronjobs
# at 5:20am every Monday
```

```
20 5 * * 1 cd $EZPUBLISH && $PHPCLI runcronjobs.php infrequent -q -s example 2>&1


# Instruct cron to run the "frequent" set of cronjobs
# every 15 minutes
0,15,30,45 * * * * cd $EZPUBLISH && $PHPCLI runcronjobs.php frequent -q -s example 2>&1


# Instruct cron to run the "monthly" set of cronjobs
# at 4:10am the first day of every month
10 4 1 * * cd $EZPUBLISH && $PHPCLI runcronjobs.php monthly -q -s example 2>&1
```

当添加到 crontab 后，cron daemon 会用 PHP 命令行接口定期执行"runcronjobs.php"。基于这个配置，主脚本会在每天的 6:35am 执行。这意味着"[CronjobSettings]"下的所有脚本每天会被执行一次。

"infrequent"组中的脚本会在每周一的 5:20am 执行。例如："[CronjobPart-infrequent]"下的脚本。

"frequent"组中的脚本会每 15 分钟执行一次。只有"[CronjobPart-frequent]"中的脚本会被执行。

"monthly"组中的脚本会在每个月第一天的 4:10am 执行。"[CronjobPart-monthly]"下的脚本会被执行。

"-q"参数要求脚本运行于安静模式（去除不必要的输出）。"-s example"表明脚本应该使用哪个站点入口的配置。"2>&1"符号要求系统将系统标准输出和标准出错合并为一个流。

Windows 计划任务

与 linux/UNIX 系统不同，Windows 不支持 cron。但是，Windows 提供了称为“计划任务”的解决方案。一个计划任务可以通过在控制面板中选择“计划任务”来创建。这会启动一个安装向导并帮助您配置计划任务。它应该被设置为定期执行一个批处理（.bat）文件。这个.bat 文件应该进入 eZ Publish 目录并执行"runcronjobs.php"脚本。

4.8 登录后高级重定向

在 eZ Publish 3.8 这，您可以配置用户登录后的重定向页面。参阅如下步骤：

1. 在用户类中添加一个“文本行”数据类型的属性。如果您有多个用户类并希望对所有用户类启用高级重定向，您需要为每个用户类添加这个属性（确保所有的属性使用相同的标识符）。
2. 在"settings/siteaccesses/example/site.ini.append.php"中的"[UserSettings]"配置"LoginRedirectionUriAttribute"。如下：

```
LoginRedirectionUriAttribute[key]=attribute_id
```

key	**您可以使用两种键值："user"代表用户类或"group"代表用户组类。**
attribute_id	新添加的类属性标识符（不是类属性 ID）。

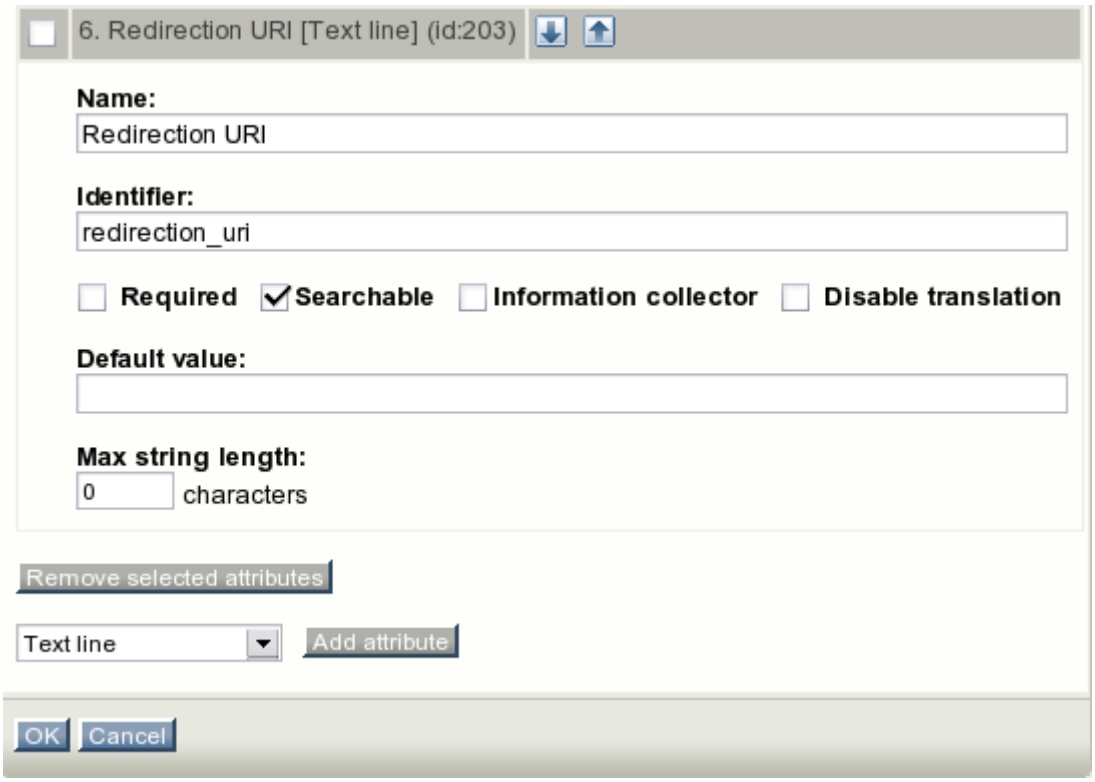
现在在创建/编辑用户的时候，您可以在文本行字段指定重定向 URI。

您可以对整个用户组指定重定向 URI。这意味着您也需要在用户组类中添加这个属性并在"LoginRedirectionUriAttribute"中用"group"作为键值。

例 1

假设用户"John"登录系统后必须被重定向到"News"文件夹。参阅如下步骤：

1. 在管理界面中选择“设置”标签，然后在左侧选择“类”并选择“用户”类组。您应该可以看到指派到这一组的类。找到您的用户类并点击“编辑”按钮。系统会显示类编辑界面。从下拉框中选择“文本行”数据类型，点击“添加属性”按钮并做如下编辑。



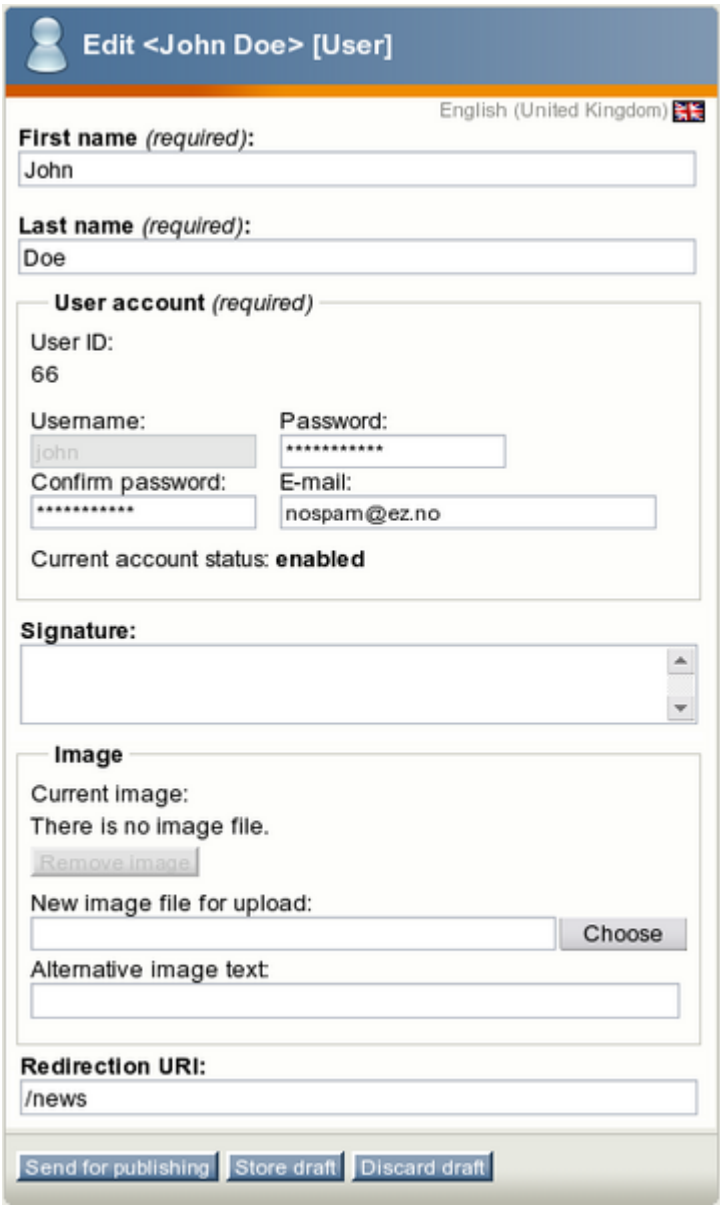
点击“确定”保存您的修改。

2. 在"LoginRedirectionUriAttribute"指定新加入的类属性的标识符。您应该在"site.ini"重设文件中"[UserSettings]"下添加如下内容：

```
LoginRedirectionUriAttribute[user]=redirection_uri
```

"redirection_uri"为属性标识符。

3. 在管理界面中选择“用户帐号”标签，通过“子项目”列表找到用户"John"并点击“编辑”按钮。系统会显示编辑界面。在"Redirection URI"字段中输入"/news"。参阅下图。

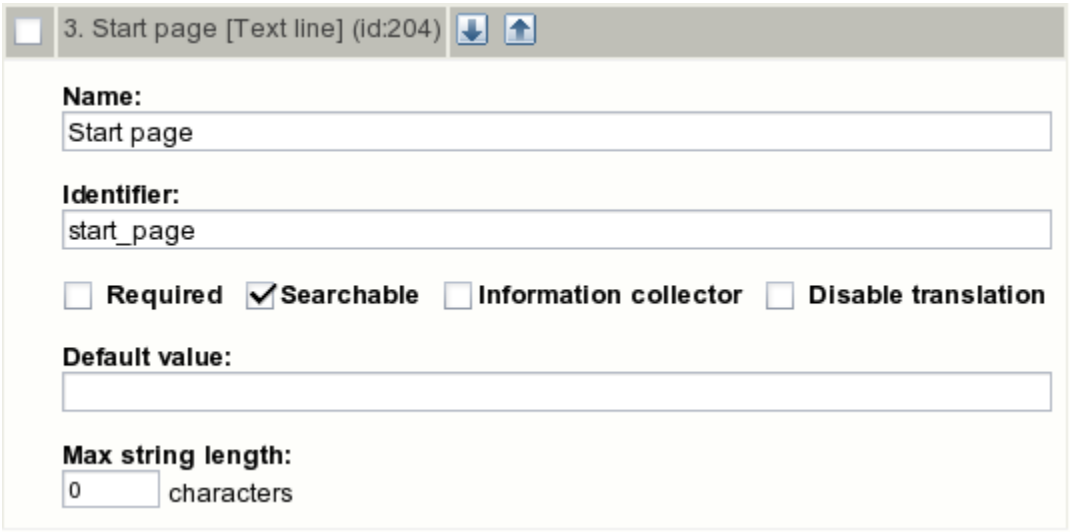


点击“发布”按钮保存您的修改。用户"John"登录系统后总是会被重定向到"News"文件夹。

例 2

假设您希望将所有"Guest accounts“组下的用户重定向到"News"文件夹。参阅如下步骤：

1. 编辑用户组类并添加如下文本行属性：

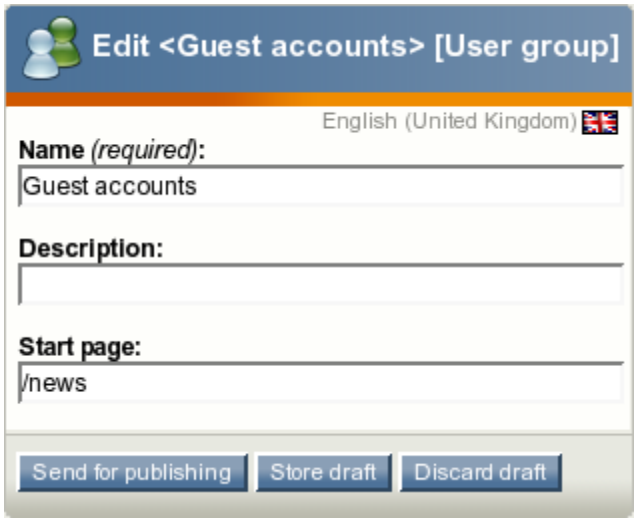


2. 在"site.ini"重设文件中"[UserSettings]"下添加如下内容：

```
LoginRedirectionUriAttribute[group]=start_page
```

"start_page"为类属性标识符。

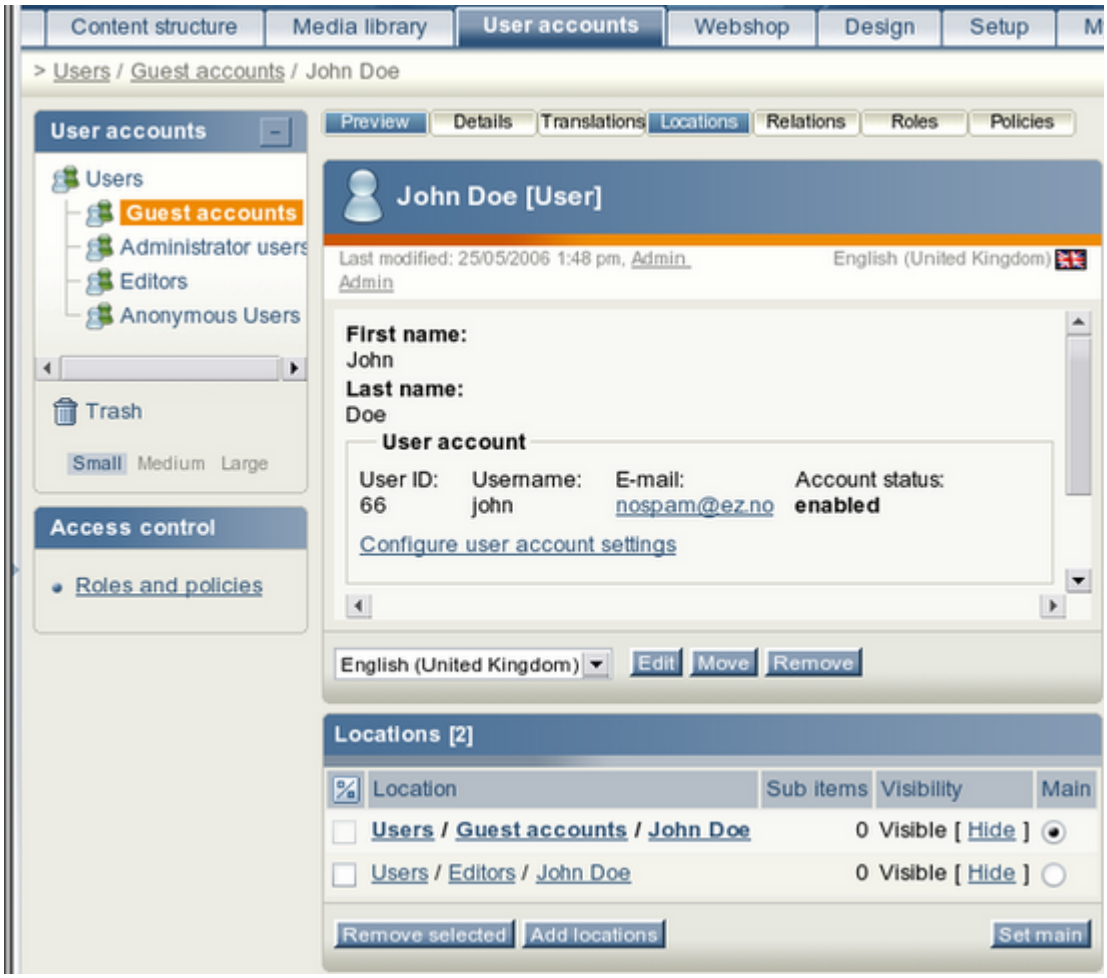
3. 编辑"Guest accounts“用户组并在"Start page"中指定"/news"。如下图。



点击“发布”按钮保存您的修改，则所有位于"Guest accounts"下的用户在登录系统后都会被重定向到"News"文件夹。

重要说明

如果一个用户属于多个组（多个“用户组”节点的子节点），系统会使用“主”用户组（主父节点）。下图演示了属于两个组（"Guest accounts"和"Editors"）的用户"John"的用户界面。“位置”列表可以用来查看和管理用户对象的位置。主位置名称用粗体显示（在本例中"Users/Guest accounts/John Doe"）。



请注意，如果重定向 URI 已经通过其它方法设置了（如：通过"LastAccessURI" session 变量）。假设您已经为用户"John"指定"/news"为重定向 URI。如果"John"打开浏览器并直接直接访问"http://yoursite.com/media_files"，那么他登录之后不会被重定向到"http://yoursite.com/news"。

4.9 增值税（VAT）系统

您的网络商店系统中的增值税取决于使用的 VAT 类型。VAT 类型包含一个名称和一个固定税率。例如："Std，0%"。管理界面可以用来添加，删除或修改 VAT 类型（参阅“管理 VAT 类型”章节）。尽管 VAT

类型的数量没有限制，您的网络商店系统中至少应该有一种 VAT 类型。这种 VAT 类型的唯一用途是用税率保存一些固定的 VAT 税率，因此您可以称它们为“静态 VAT 类型”或“固定 VAT 类型”。

如果您为商品指定静态 VAT 类型，那么系统总是为这个商品征收固定税率的 VAT。（这是“按商品征收 VAT ”的工作方法。）

含税价格/不含税价格

有两种方法来使用指派的 VAT 类型。这取决于创建商品对象时如何输入价格。如果输入的价格为含税价格，可以使用“含税价格”方法；否则使用“不含税”价格。使用含税价格查看商品时，会显示输入的价格。使用不含税价格查看商品时，显示的价格为输入价格加上 VAT 的金额。

动态 VAT 类型

动态 VAT 不保存固定税率也不能通过管理界面配置。在您编辑商品对象时，这种类型可以从 VAT 类型列表中选择。默认情况下，这种类型被称为"Determined by VAT charging rules"。（参阅“VAT 设置”章节了解如何通过"DynamicVatTypeName"修改这个显示明。）选择这种类型的 VAT（为商品指派动态 VAT 类型）会告诉系统这个商品没有固定税率，因此 VAT 金额应该通过某些复杂的逻辑动态计算。例如：如果 VAT 金额取决于用户的居住地。

动态 VAT 类型与含税价格不兼容。您应该为您的商品设置不含税价格。请注意，VAT 类型与 VAT 处理器连接，因此如果 VAT 处理器不可用，VAT 类型会被禁用。

VAT 处理器

如果您希望使用某些复杂的征收逻辑，必须通过自定义的 VAT 处理器实现（一个可以动态决定 VAT 金额的 PHP 类）。您可以使用系统内建的默认 VAT 处理器（它支持基于国家的 VAT）或开发您自己的 VAT 处理器扩展（扩展 VAT 解决方案）。不能同时使用两种 VAT 处理器。

必须在“VAT 设置”章节中描述的"Handler"配置要使用的 VAT 处理器。要启用内建的 VAT 处理器，您需要在"shop.ini"重设文件中"[VATSettings]"下添加如下内容：

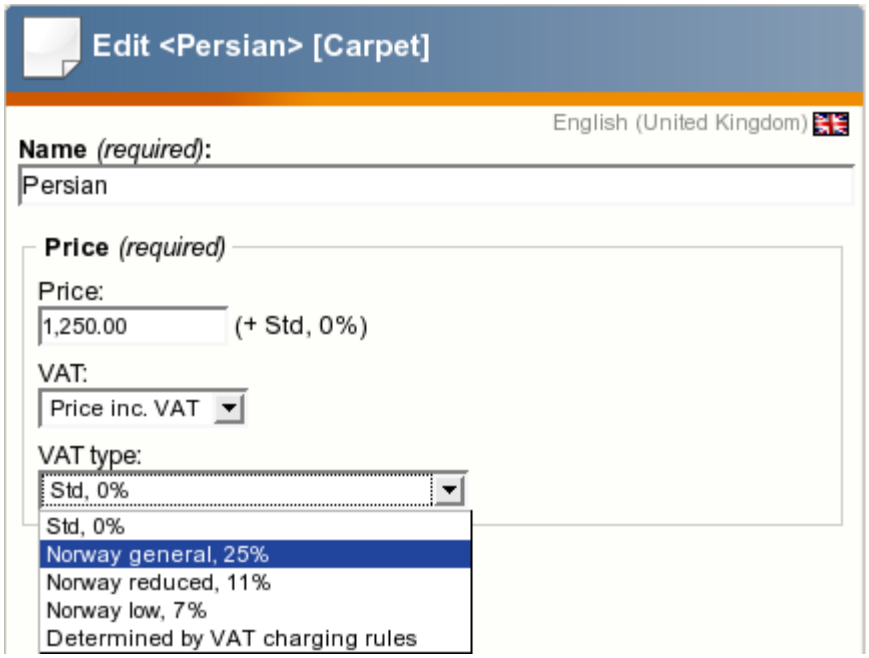
```
Handler=ezdefault
```

系统会开始使用默认的 VAT 处理器并在您创建/编辑商品或商品类时显示动态的 VAT 类型。

请注意，启用 VAT 处理器后，动态 VAT 是一种虚拟的结构。如果没有指定 VAT 处理器，动态 VAT 类型不会被显示。

4.9.1 指派商品增值税类型

要为商品指派 VAT 类型，编辑商品并从“VAT 类型”下拉框中选择需要的 VAT 类型（如下图）。对简单价格和多价格商品（由“价格”和“多价格”数据类型支持）都可以使用这种方法。



上图显示了一个简单价格商品编辑界面的一部分。因为价格被设置为$1,250 并且选择了“含税价格”选项，显示给客户的实际价格为$1,250。如果您指派 25%的固定税率，则 VAT 为$250。

如果您选择“不含税价格”，则 VAT 计算如下：

1,250*25/100 = 312.50

实际的商品价格如下：

1,250.00 + 312.50 = 1,562.50

如果您选择“根据 VAT 征收规则动态决定”，那么动态 VAT 类型会被指派到这个商品。这种类型只与“不含税价格”兼容。

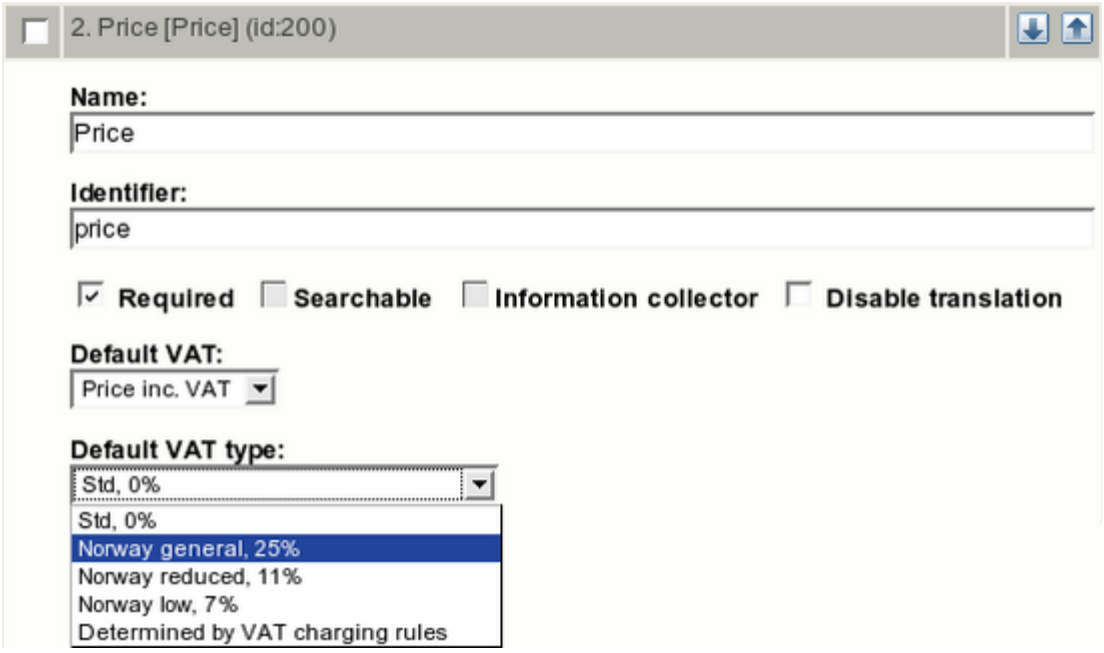
商品类中的默认 VAT 类型

可以在类级别上选择默认 VAT 类型（当您创建或编辑商品类时）。在创建新商品对象时会默认使用这个 VAT 类型。但是，您仍然可以为每个商品单独指派 VAT 类型。

下例演示了如何为商品类指派默认 VAT 类型。

1. 在管理界面中选择“设置”标签，然后选择左侧的“类”，再选择“内容”类组。您应该可以看到属于这个类组的类列表。找到您的商品类然后点击“编辑”按钮。系统会显示类编辑界面。

2. 找到价格或多价格类属性。您可以看到“默认 VAT 类型”下拉框。选择需要的 VAT 类型然后点击“确定”保存您的修改。参阅下图。



上图演示了一个简单价格商品类（"Carpet"）编辑界面的一部分。这里您可以选择“含税价格”或“不含税”价格为所有这种类型的新对象的默认配置。您也选择例如：25%为默认 VAT 类型。请注意，您可以在对象级别修改以上默认设置（为每个 Carpet 类型的商品）。如果您选择最后一个项目“根据 VAT 征收规则动态决定”，则动态 VAT 会被指派为新对象的默认 VAT 类型。这种 VAT 类型只能与“不含税”价格合用。

4.9.2 三种 VAT 征收方法

网络商店系统支持以下三种 VAT 征收方法：

1. 基于商品的 VAT
2. 基于国家的 VAT
3. 扩展 VAT

下一章解释了它们的区别。

基于商品的 VAT

基本的“基于商品的 VAT ”解决方案允许您创建/编辑商品时，为其选择一个预定义的静态 VAT 类型。因此，您可以为每个商品指定一个固定税率。也可以在类级别指定默认的 VAT 类型，从而所有这种类型的新对象都使用这种默认的 VAT 类型。

例

假设您在挪威销售各种商品并需要征收以下 VAT：

- 对大部分商品，按通用税率 25%征收。
- 对食品，按缩减税率 11%征收。
- 对个人交通工具，按低税率 7%征收。

你需要创建三种 VAT 类型（参阅“创建 VAT 类型”章节），然后您可以为每个商品指派正确的 VAT 类型。

您也可以创建不同的商品类并为它们指定不同的默认 VAT 类型。

例如，假设您已经创建了如下三种商品类：

- “地毯”，默认 VAT 类型为 25%
- “食品商品”，默认 VAT 类型为 11%
- “摩托车”，默认 VAT 类型为 7%

在本例中，系统会为每个新创建的“地毯”使用 25%的固定税率，每个新创建的“食品商品”使用 11%的固定税率，为每个新创建的“摩托车”使用 7%的固定税率。

基于国家的 VAT

大多数情况下，VAT 金额取决于客户的居住地。“基于国家的 VAT ”解决方案允许 VAT 税率依赖于商品分类（如果指定）与客户的国家。这可以通过使用动态 VAT 类型和内建的默认 VAT 处理器来实现。这个处理器通过 VAT 征收规则来决定正确的 VAT 税率。

VAT 征收规则

VAT 征收规则由以下组件构成：

- 用户国家
- 商品分类（一个或多个）
- VAT 类型

VAT 规则还根据客户的国家和商品的分类来决定使用哪个固定税率。管理界面允许添加，删除或修改 VAT 征收规则（参阅“管理 VAT 规则”章节）。默认的 VAT 规则指派到“任何国家”和“任何分类”决定了如果其它 VAT 规则都不匹配，应该使用哪个税率。

对于特定的“国家－分类”组匹配度越高的规则，优先级越高。换言之，默认的 VAT 处理器会尝试选择匹配度最高的 VAT 税率。要理解这个算法如何工作，参阅下表：

国家	分类	示例	优先级
完全匹配	完全匹配	挪威 – 食品	4
完全匹配	弱匹配	挪威 – 任何商品	3
弱匹配	完全匹配	任何国家 – 食品	2
弱匹配	弱匹配	任何国家 – 任何商品	1

如果没有匹配的国件和/或没有匹配的分类，则优先级最低（0）的规则生效。

配置基于国家的 VAT

如果您销售地毯并需要对德国的客户征收 16%的 VAT 而对挪威的客户征收 25%的 VAT，那么“基于商品的 VAT ”不适合这种情况。以下文字描述了如何实施“基于国家的 VAT ”方案。

1. 参阅“VAT 处理器”章节启用内建的默认 VAT 处理器。
2. 参阅“创建 VAT 类型”章节创建以下两种 VAT 类型：
   - 挪威一般，25%
   - 德国一般，16%
3. 在您的用户类中添加一个“国家”数据类型的属性，并参阅“为用户类添加国家属性”章节在 INI 文件的"UserCountryAttribute"中指定类属性标识符。
4. 参阅“创建 VAT 规则”章节创建以下两种 VAT 规则：

用户国家	商品分类	VAT 类型
挪威	任何	挪威一般，25%
德国	任何	德国一般，16%

系统也会要求您创建一条用于其它国家用户的默认 VAT 规则。

因为您只销售一种类型的商品，没哟必要创建商品分类。对“任何”商品指定的 VAT 规则会应用到您的所有商品。

5. 要另您的商品被 VAT 征收规则影响，您应该参阅“为商品指派 VAT 类型”章节为您的商品指派动态 VAT 类型。

设置基于国家和分类的 VAT

如果您的网络商店销售不同类型的商品且 VAT 税率不同，那么税率将同时依赖于用户的国家和商品分类。这意味着您必须创建商品分类，将它们指派到您的商品并为这些分类指派 VAT 规则（不是指派到“任何”分类）。

例如：假设您向挪威和德国的客户销售不同类型的商品并根据商品分类征收以下 VAT：

- 德国
  - *大部分商品，通用税率，16%*
  - *食品，缩减税率，7%*

- 挪威
  - *大部分商品，通用税率，25%*
  - *食品，缩略税率，11%*
  - *个人交通工具，低税率，7%*

以下文字解释了如何在本例中使用“基于国家的 VAT ”

1. 参阅“VAT 处理器”章节启用内建的 VAT 处理器。
2. 参阅“创建 VAT 类型”章节创建以下四种 VAT 类型：
   - *挪威一般，25%*
   - *德国一般，16%*
   - *挪威缩减，11%*
   - *挪威低，德国缩减，7%*
3. 为您的用户类添加一个国家类型的属性，并参阅“为用户类添加国家属性”章节在 INI 文件中的"UserCountryAttribute"指定类属性标识符。
4. 为您的商品类添加一个商品分类的属性，并参阅“为商品类添加商品分类”章节在 INI 文件中的"ProductCategoryAttribute"指定类属性标识符。
5. 参阅“创建商品分类”章节创建以下两种商品分类：
   - *食品*
   - *个人交通工具*
6. 参阅“创建 VAT 规则”章节创建以下五种 VAT 规则：

用户国家	商品分类	VAT 类型
德国	食品	挪威低，德国缩减，7%
德国	任何	德国一般，16%
挪威	个人交通工具	挪威低，德国缩减，7%
挪威	食品	挪威缩减，11%
挪威	任何	挪威一般，25%

系统同时会要求您为任何分类和任何国家（如果其它 VAT 规则都没有匹配，则使用这条规则）创建默认 VAT 规则。

7. 为您的商品指派动态 VAT 类型（参阅“为商品指派 VAT 类型”）并为它们指派正确的商品分类（参阅“为商品指派分类”章节）。

扩展 VAT

如果您需要更复杂的 VAT 征收逻辑，您可以开发您自己的 VAT 处理器扩展来满足特定的需求。这种解决方案与前面的解决方案不兼容，因为不能同时使用两个 VAT 处理器。在“VAT 配置”章节中介绍的"Handler"INI 配置确定 VAT 处理器的类型。

由您的处理器实现的 VAT 征收逻辑会应用到所有使用动态 VAT 类型的商品。牢记动态 VAT 类型不支持“含税价格”。

请参阅“创建新 VAT 处理器”章节了解更多。

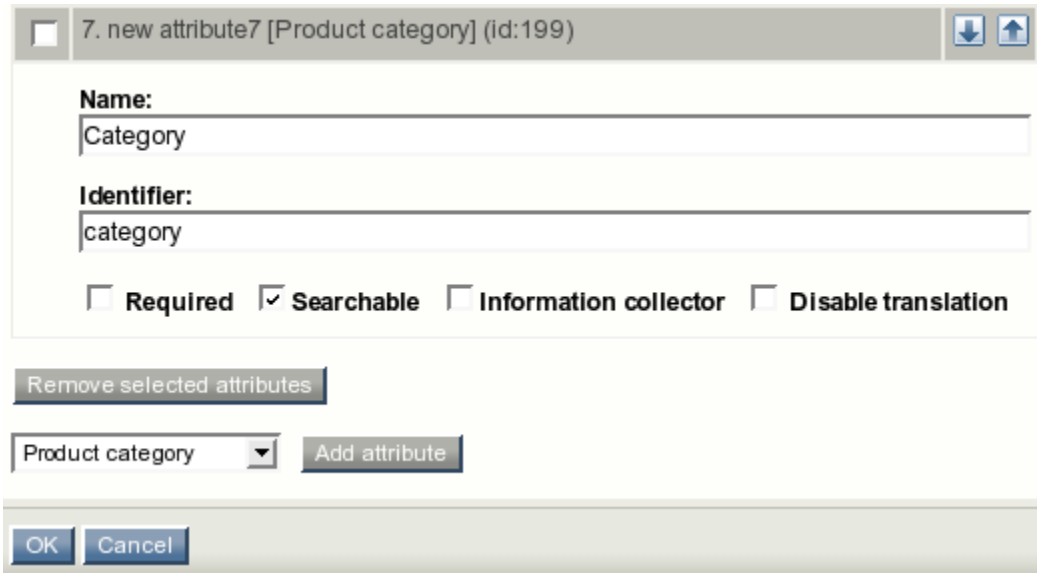
4.9.3 商品分类

“基于国家的 VAT ”解决方案假定您的商品可以被指派到商品分类。下一章揭示了如何创建商品分类。管理界面可以用来创建，删除或重命名商品分类（参阅“管理商品分类”章节）。

为商品类添加新属性

要为商品指派分类，必须添加一个商品分类类型的类属性。参阅以下步骤。

1. 在管理界面中选择“设置”标签，然后选择左侧的“类”，再选择“内容”类组。您应该会看到属于“内容”组的类列表。找到您的商品类然后点击“编辑”按钮。系统会显示类编辑界面。
2. 从下拉框中选择商品分类数据类型，然后点击“添加属性”并做如下编辑。下图演示了在类编辑界面中添加商品分类数据类型属性时的页面片段。

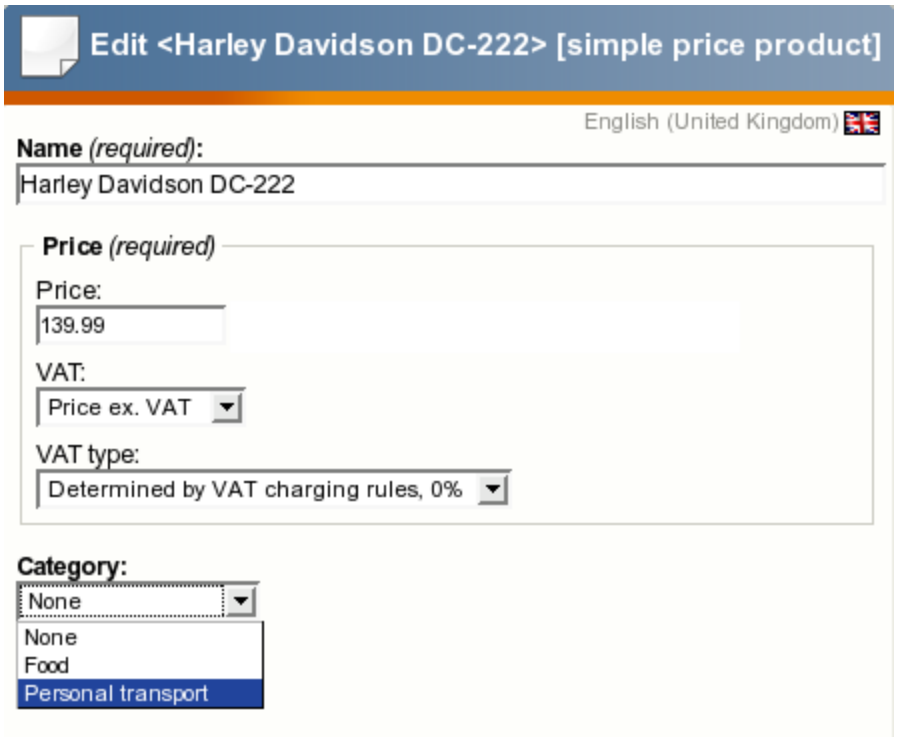


系统会在上篇对象的编辑界面中增加一个称为“分类”的（新属性的名称）下拉框。您可以在下拉框中选择需要的分类来为商品指派分类。

3. 在"shop.ini"重设文件中的"[VATSettings]"下配置"ProductCategoryAttribute"为新添加属性的标识符。

为商品指派分类

如果您的商品类包含一个商品分类数据类型的属性，那么在编辑这个商品的时候，您可以为它指派分类。要做到这一点，编辑商品然后从“分类”下拉框中选择需要的分类。参阅下图。



注意：

商品分类总是与动态 VAT 类型（如上图所示，“根据 VAT 征收规则确定”被选中）一起使用。为固定税率的商品选择分类没有任何意义。

4.9.4 用户国家

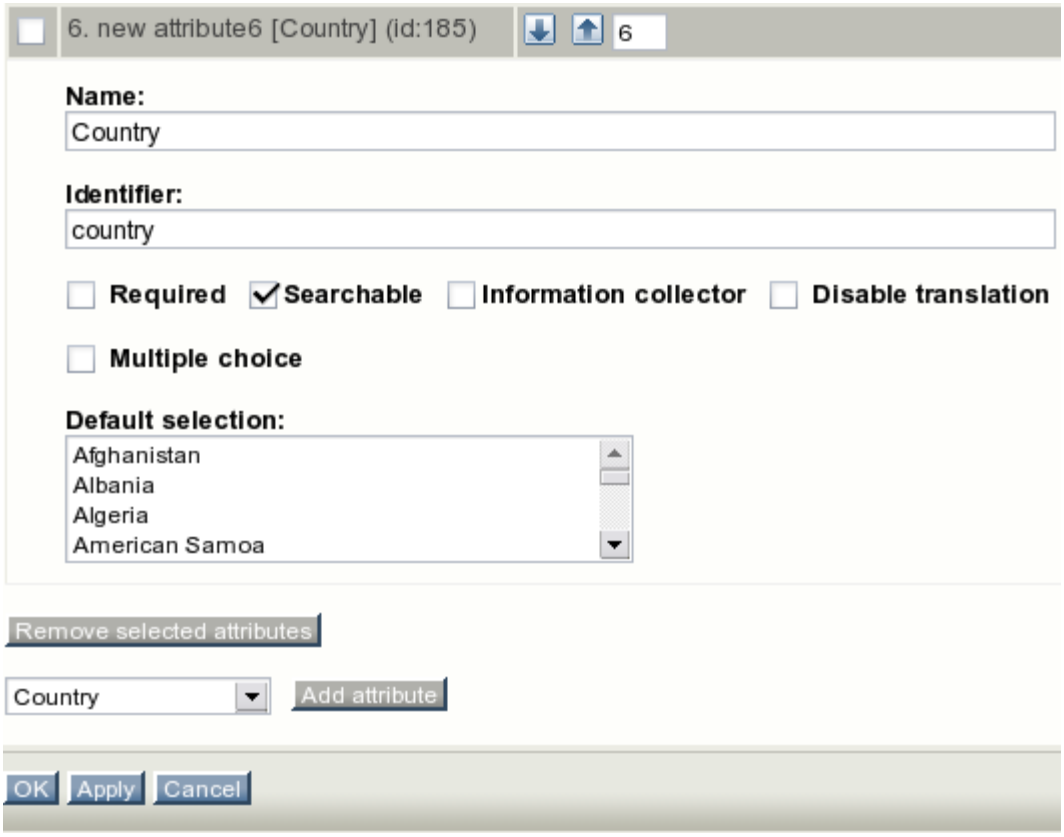
“基于国家的 VAT ”假设您的每个用户可以被指派一个国家。下一章节揭示了具体步骤。注意，可用国家以及他们的属性（电话区号等等）可以在"country.ini"重设文件中指定。

为用户类添加新属性

要为用户指定国家，需要为用户类添加国家数据类型的属性。参阅以下步骤。

1. 在管理界面中选择“设置”，然后选择左侧的“类”，再选择“用户”组。您应该可以看到属于用户组的类列表。找到您的用户类然后点击“编辑”按钮。系统会显示类编辑界面。

2. 从下拉框中选择“国家”数据类型，点击“添加属性”按钮并对新属性做如下编辑。参阅下图。



在编辑用户时，系统会显示国家下拉框。您可以从这个下拉框中选择国家。

3. 在"shop.ini"重设文件的"[VATSettings]"中配置"UserCountryAttribute"为国家属性标识符。

为用户指定国家

如果您的用户类包含国家数据类型的属性，则可以用以下两种方法为用户指派国家：

- 新用户需要在注册表单中的国家下拉框中选择自己的国家。
- 站点管理员可以在管理界面中编辑用户并为用户指派国家。
- 站点管理员可以添加一个工具栏允许用户修改自己的国家。

添加客户工具栏

建议允许用户通过“用户国家”工具栏修改自己的国家。要启用工具栏，在"settings/siteaccess/example/toolbar.ini"中的"[Toolbar_right]"下添加如下代码：

```
Tool[]=user_country
```

这个设置要求系统在页面右侧显示国家工具栏。当用户选择了国家，系统会立刻根据用户所选国家对应的 VAT 规

则更新商品价格。

为了避免缓存问题，您需要在所有的站点入口的"CachedViewPreferences[full]"中指定"user_preferred_country"。可以编辑"site.ini"的重设文件。如果文件中已经存在如下配置：

```
CachedViewPreferences[full]=<list_of_user_preferences>
```

则您需要在行尾追加一个":"和"user_preferred_country"，例如：

```
CachedViewPreferences[full]=admin_navigation_content=0;
admin_navigation_details=0;<...>;admin_bookmarkmenu=1;
admin_left_menu_width=13;user_preferred_country=''
```

注意，这一行配置通常会很长。上例中我们用<...>省略了中间部分。

如果"[ContentSettings]"下没有这条配置，可以创建它：

```
CachedViewPreferences[full]=user_preferred_country=''
```

如果不指定这个配置，您的客户将无法改变国家（由于缓存的原因）。

使用其它国家数据类型

除了系统内建的国家数据类型，您也可以使用其它的国家类型。这意味着您可以把其它数据类型集成到系统中用来保存用户国家，并且用户国家可以通过数据类型，VAT 规则管理界面和商店用户注册模块(shop/userregister)以相同的方式保存。以下列表揭示了具体步骤。

1. 确保您的数据类型内容是一个有键值的哈希或支持 get 和 set 值属性的对象（如 eZPersistentObject）。无论内容实际上是如何保存在数据库中的，objectAttributeContent()方法必须返回一个数组/对象。返回值（通常是国家代码）接着会和 VAT 规则中的国家比较。
2. 在您的数据类型扩展中重设"standard"界面下"templates/shop/country"目录中的"view.tpl"与"edit.tpl“，从而保证国家可以在 VAT 规则管理界面和商店用户注册模块中显示和编辑。

4.9.5 显示增值税

下一章解释了在使用“基于商品的 VAT ”和/或“基于国家的 VAT ”时，如何在真实的站点中显示 VAT。

基于商品的 VAT

假设某个固定税率的 VAT 被指派到每个商品。当用户查看商品页面时，系统会显示这个商品的含税价格。用户可以把商品添加到购物篮中，在购物篮中会显示 VAT 的税率。用户可以点击“结帐”按钮来购买商品。

点击“结帐”按钮后，系统会要求用户输入他/她的姓名，电子邮件，国家以及其它客户帐号和订单需要的信息。这些信息由"shopaccount.ini"重设文件中"[AccountSettings]"下指定的商店帐号处理器负责处理。通常用户需要选择一个国家（这个行为不依赖于“VAT 设置”章节中的"RequireUserCountry"INI 配置）。

国家信息会与客户的订单信息一起保存在系统中。客户由客户唯一的电子邮件标识，所以对应不同邮件的订单属于不同的客户。客户的帐号包含关于国家的信息。国家信息在客户第一次创建订单时指定。请注意，客户帐号是一种特殊的数据结构，它在网络商店系统中使用但不与真正的用户对象关联。

如果您已经在您的用户类中添加了国家类型的属性就可以为用户指派国家。如果您在"UserCountryAttribute"中指定了这个属性的标识符，那么在用户第一次创建订单的时候，国家会被自动指派给用户。请注意，对于“基于商品的 VAT ”，这一特性通常是不需要的，因此如果没有指定 VAT 处理器，它是被禁用的。此外，如果用户的国家已经被设置了（用户的国家可以通过注册时在表单中选择国家，编辑用户并选择国家或从国家工具栏中选择国家来设置），系统不会自动修改用户的国家。

基于国家的 VAT

假设您使用“基于国家的 VAT ”解决方案，您的用户类包含一个国家类型的属性并且它的属性标识符已经在"UserCountryAttribute"中设置。

当用户查看一个商品页面时，系统会显示商品的含税价格。用户可以把商品加入购物篮，在购物篮中会显示每个商品的 VAT 税率。点击“结帐”按钮后，系统会要求用户输入他/她的姓名，电子邮件，国家以及其它用户帐号和订单需要的信息。系统会用最匹配指定国家的 VAT 规则重新计算 VAT 并将 VAT 包含在最终商品价格中。最终价格会在确认订单页面显示。

如果为用户指派了国家，系统会根据最匹配用户的国家和商品分类的 VAT 规则重新计算 VAT。如果在点击“结帐”按钮之后选择了其它国家（不是用户配指派的国家），这个国家不会被指派给用户但是会用于这个特定的订单。

如果没有为用户指派国家，那么查看商品时显示的 VAT 金额会根据默认 VAT 规则计算。用户点击“结帐”按钮之后选择的国家会被指派给这个用户。

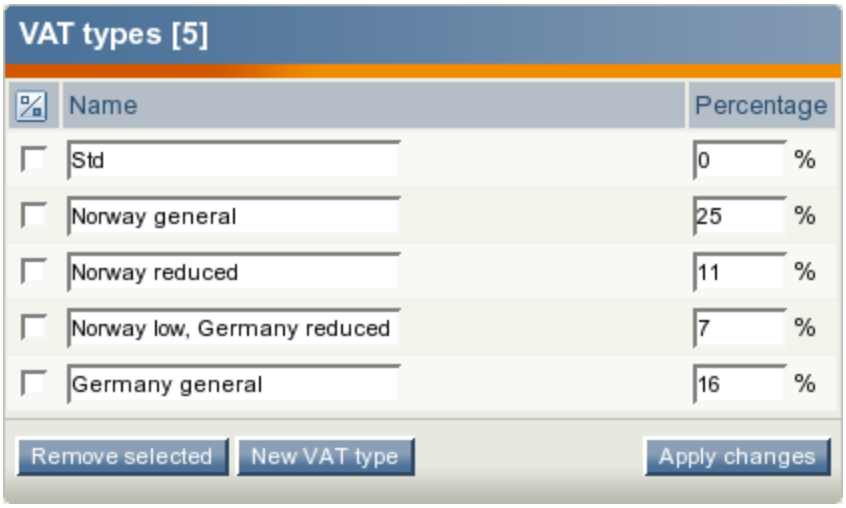
4.9.6 管理增值税类型

本章描述了如何在管理界面中添加，删除，修改静态 VAT 类型。

创建 VAT 类型

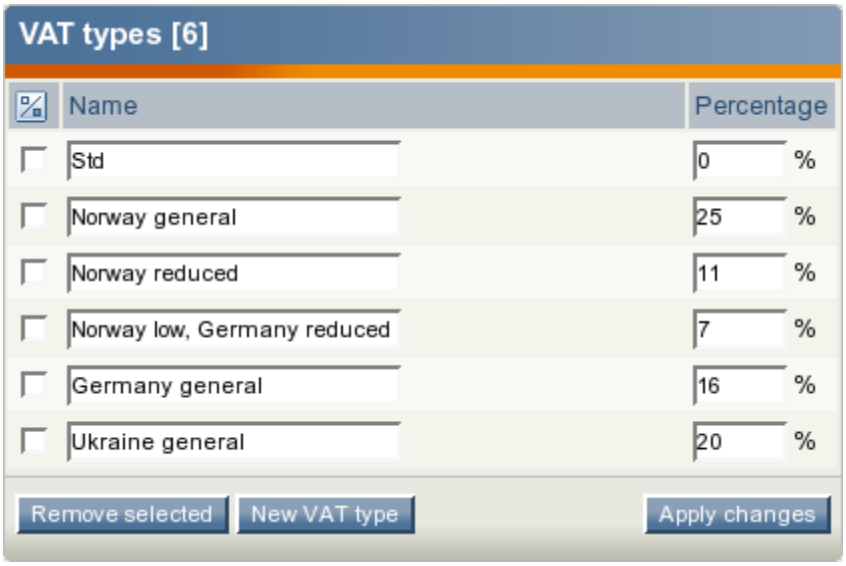
要在您的网络商店中使用 VAT，您需要创建 VAT 类型。参阅以下步骤。

1. 在管理界面中选择“网络商店”，然后选择左侧的“VAT 类型”。系统会显示现存的 VAT 类型列表。如下图。这个界面也可以通过"/shop/vattype"来访问。



点击“新 VAT 类型”按钮。系统会添加一个新的项目"VAT type 1"，默认的税率为 0%。

2. 如下图所示，为这个项目指定名称和税率。



3. 点击“应用修改”按钮保存您的修改或点击“新建 VAT 类型”继续创建 VAT 类型。

编辑 VAT 类型

如果您希望修改您的 VAT 类型，参阅如下步骤：

1. 在管理界面选择“网络商店”并选择左侧的“VAT 类型”，系统会显示 VAT 类型的列表。
2. 为您需要修改的 VAT 类型指定名称和税率（您可以同时修改多个 VAT 类型）。
3. 点击“应用修改”按钮保存您的修改。

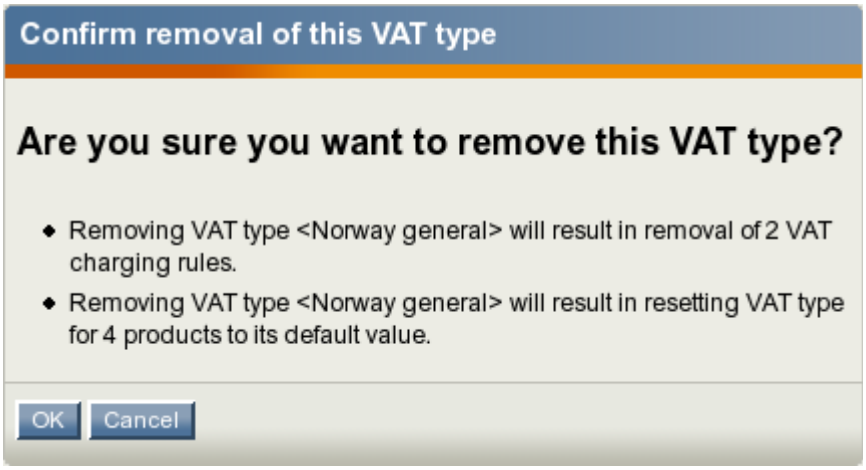
删除 VAT 类型

您不能删除商品类的默认 VAT 类型。可以删除被指派到商品对象和/或用于 VAT 规则的 VAT 类型，但是不建议这样做（这些 VAT 规则会被删除并且默认的 VAT 类型会被指派给商品）。大部分情况下，您应该修改 VAT 类型的名称和税率，而不是删除它。

请注意，您不能删除所有的 VAT 类型。如果您不希望对您的商品征收 VAT，您可以保留一个 VAT 类型并将它的税率设为 0。

以下文字揭示了如何从网络商店系统中删除一个或多个 VAT 类型。

1. 在管理界面中选择“网络商店”标签，然后选择左侧的“VAT 类型”，系统会显示 VAT 类型列表。
2. 用复选框够选希望删除的 VAT 类型。不要选择所有的 VAT 类型。
3. 点击“删除所选”按钮
4. 如果您的某些商品使用这个 VAT 类型并且/或您的某些 VAT 规则基于这个 VAT 类型，系统会显示一个确认对话框（如下图）。



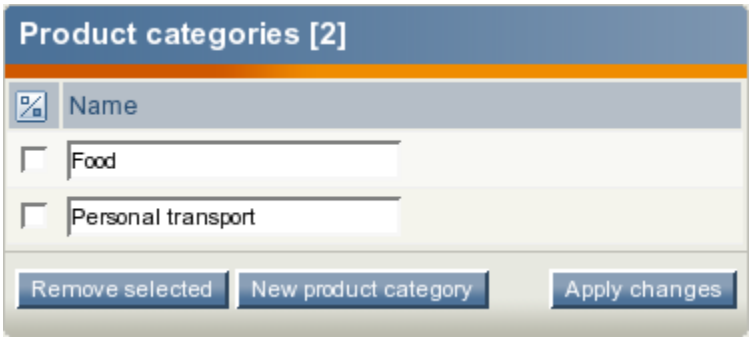
4.9.7 管理商品类型

本章揭示了如何在管理界面中添加，删除或修改商品分类。

创建商品分类

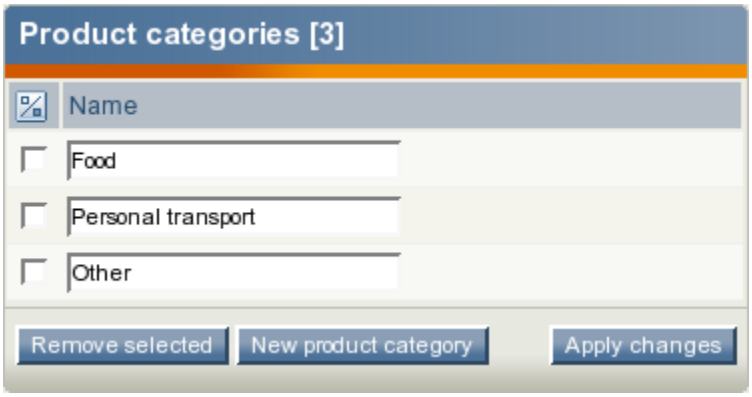
管理界面允许您在网络商店系统中添加新的商品分类。参阅如下步骤。

1. 在管理界面中选择“网络商店”标签，然后选择左侧的“商品分类”。系统会显示现存的商品分类列表（如下图）。这个界面也可以通过"/shop/productcategories"访问。



点击“新建商品分类”按钮。系统会添加一个新项目"Product category 1"。

2. 为这个分类指定名称（如下图）。



3. 点击“应用修改”按钮保存您的修改。

删除商品分类

您可以删除被指派到商品的分类和/或用于 VAT 规则的商品分类，但是不推荐这样做。大多数情况下，您应该修改这个分类的名称活修改这个分类的 VAT 规则，而不是从系统中删除它。删除一个商品分类，不会删除属于这个分类的商品。系统会将这些商品的分类属性复位并修改或删除用于这个分类的 VAT 规则。

例

假设您有如下的 VAT 规则。

用户国家	商品分类	VAT 类型
任何	巧克力，咖啡，果汁	挪威缩减，7%

任何	洗发水	挪威一般，25%

如果您删除洗发水分类，第二条 VAT 规则会被删除。如果您删除“巧克力”分类，第一条规则会被修改如下。

用户国家	商品分类	VAT 类型
任何	咖啡，果汁	挪威缩减，7%

以下文字揭示了如何删除商品分类。

1.在管理界面中选择“网络商店”标签，然后选择左侧的“商品分类”。

2.用复选框够选要删除的分类。

3.点击“删除所选”按钮。如果有商品和/或 VAT 规则在使用这个分类，系统会显示删除确认对话框，如下图。点击“确定”按钮确认删除。

4.9.8 管理增值税规则

这一章描述了如何在管理界面中添加，删除或修改 VAT 征收规则。

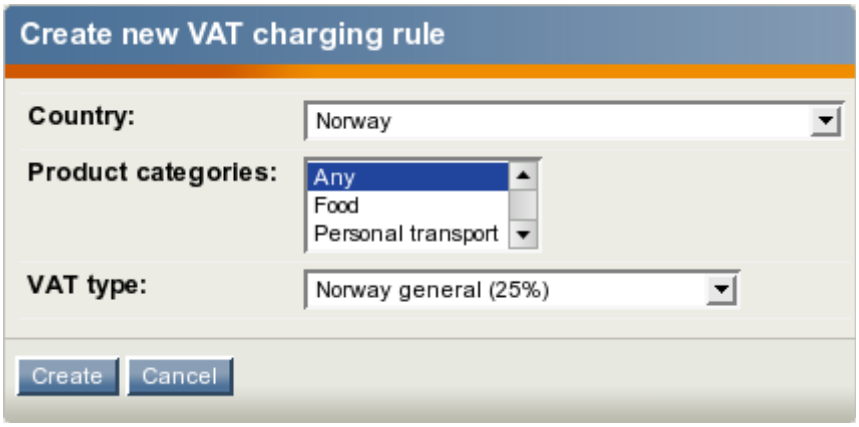
创建 VAT 规则

管理界面允许您为网络商店添加新的 VAT 征收规则。参阅以下步骤。

请注意，建议（但不是一定）在创建 VAT 征收规则之前，先创建静态 VAT 类型和商品分类（如果有）。

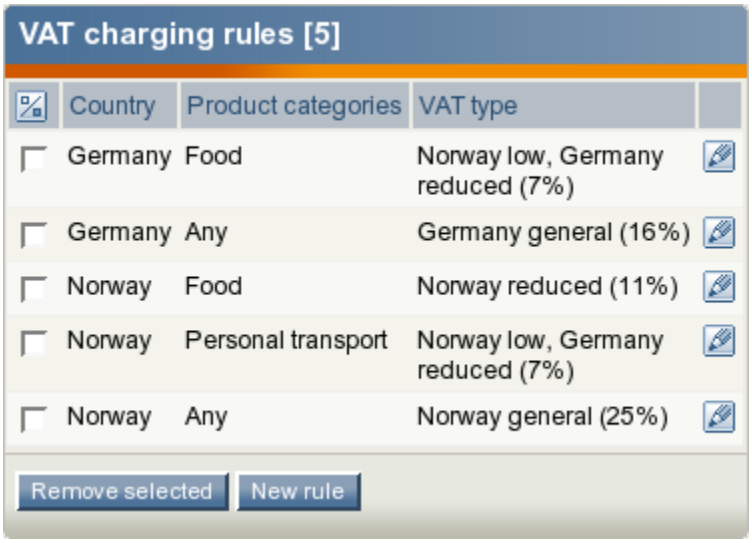
1. 在管理界面中选择“网络商店”标签，然后选择左侧的“VAT 规则”。系统会显示现存 VAT 规则列表，如下图。这个界面也可以通过"/shop/vatrules"访问。

点击“新建规则”按钮。系统会显示 VAT 规则编辑界面（如下图）。



2. 为 VAT 规则指定如下参数：
   - 从国家下拉框中选择需要的国家。这个 VAT 规则会用于属于这个国家的用户。
   - 选择一或多个受这条规则影响的商品分类。
   - 从下拉框中选择一种静态 VAT 类型。选定的 VAT 类型决定实际的 VAT 税率。
   - 点击“创建”按钮。

新的 VAT 规则会在列表中显示，如下图。



编辑 VAT 规则

以下文字揭示了如何编辑 VAT 规则。

1. 在管理界面中选择“网络商店”标签，然后选择左侧的“VAT 规则”，系统会显示 VAT 规则列表。
2. 找到要修改的 VAT 规则并点击“编辑规则”按钮。
3. 系统会显示 VAT 规则编辑界面。在编辑界面中指定参数并点击“保存修改”按钮。

删除 VAT 规则

以下文字揭示了如何删除 VAT 规则。

1. 在管理界面中选择“网络商店”标签，然后选择左侧的“VAT 规则”，系统会显示 VAT 规则列表。
2. 用复选框勾选要删除的 VAT 规则。
3. 点击“删除所选”按钮

4.9.9 增值税配置

在"shop.ini"文件中的"[VATSettings]"分区定义了 VAT 处理器。在这个分区内可以指定以下配置：

- "Handler"指定应该使用的 VAT 处理器。
- "RepositoryDirectories[]"数组指定了 eZ Publish 应该从哪里搜索内建的 VAT 处理器。
- "UserCountryAttribute"指定了用户类中的国家属性标识符。
- "ProductCategoryAttribute"指定了商品分类属性标识符。
- "RequireUserCountry"默认为 true，因此系统总是需要用户国家。如果设置为 false，如果没有指定用户国家，系统不会报错。
- "DynamicVatTypeName"指定系统如何显示动态 VAT 类型的名称。默认名称为“根据 VAT 规则”，您可以设置

为例如：“动态 VAT”，“基于国家的 VAT”，“扩展 VAT”或“我的 VAT”等等。系统会将这个名称作为类/对象查看/编辑界面中的 VAT 下拉框中最后一项的名称。

例 1

以下内容可以被添加到"shop.ini"重设文件中的"[VATSettings]"内：

```
[VATSettings]
Handler=ezdefault
RepositoryDirectories[]=kernel/classes/vathandlers
```

这些配置会要求 eZ Publish 使用内建的 VAT 处理器。这个文件位于"kernel/classes/vathandlers/ezdefaultvathandler.php"。

例 2

您可以通过开发自定义的 VAT 处理器扩展来满足特定的需求。例如，如果您有一个扩展"myextension"，它包含一个 VAT 处理器"myrule"，您可以将以下内容加入"shop.ini"重设文件中：

```
[VATSettings]
Handler=myrule
ExtensionDirectories[]=myextension
```

或

```
[VATSettings]
Handler=myrule
RepositoryDirectories[]=extension/myextension/vathandlers
```

这些配置会要求 eZ Publish 使用位于"extension/myextension/vathandlers/myrulevathandler.php"的 VAT 处理器。

4.9.10 创建增值税处理器

本章为希望开发新 VAT 处理器的程序员（只适合那些熟悉 PHP 的程序员）提供了一些有用的信息。请注意，不建议修改 eZ Publish 内核，因此您应该在扩展中实现您的 VAT 处理器。

处理器接口

本章描述了一些实现细节。

VAT 处理器是一个 PHP 文件，它包含一个实现了如下方法的类：

```
/**
*
* \public
* \static
```

```
* \param $object      The product content object.
* \param $country     Country the buyer is from, or false if not specified.
* \return             VAT percent (integer), or null in case of an error.
*/
mixed function getVatPercent( eZContentObject $object, mixed $country );
```

处理器不是被系统直接调用而是通过 eZVATManager 类调用。这个类的 getVAT()方法返回对于某个商品因该使用的 VAT 税率：

```
$vatPercent = eZVATManager::getVAT( $object, $country );
```

getVAT()方法实际上会调用"Handler"INI 配置中指定的处理器的 getVatPercent()方法。

下一章解释了如何实现您自己的 VAT 处理器。

创建您自己的 VAT 处理器

假设您需要根据商品所在的分区动态地决定 VAT 税率。您可以按照如下步骤创建您自己的 VAT 处理器"mysectionbased"。

1. 在 eZ Publish 的 extension 目录下创建以下目录：
   - myextension
   - myextension/settings
   - myextension/vathandlers
2. 在"myextension/vathandlers/"目录中创建"mysectionbasedvathandler.php"（这个文件必须包含一个名为"MySectionBasedVATHandler"的 PHP 类）并加入以下内容：

```
<?php
class MySectionBasedVATHandler
{
    /**
     * \public
     * \static
     */
    function getVatPercent( $object, $country )
    {
        $section = $object->attribute( 'section_id' );
        if ( $section == 1 )
            $percentage = 10;
        else
            $percentage = 20;
        return $percentage;
```

```
    }
}
?>
```

3. 在"myextension/settings"目录中创建"shop.ini.append.php"并添加如下内容：

```
[VATSettings]
ExtensionDirectories[]=myextension
Handler=mysectionbased
RequireUserCountry=false
DynamicVatTypeName=Section based VAT
```

这会要求 eZ Publish 使用"extension/myextension/vathandlers/mysectionbasedvathandler.php"中的 VAT 处理器。因为由这个处理器决定的税率不依赖于用户的国家，"RequireUserCountry"必须被设置为 false。因为这个处理器不使用 VAT 规则，将动态 VAT 类型显示为“基于分区的 VAT”（而不是“由 VAT 征收规则决定”）将更合理。这是通过"DynamicVatTypeName"来配置的。

4. 要启用您的扩展，登录 eZ Publish 管理界面，选择“设置”标签，然后选择左侧的“扩展”。系统会显示可用的扩展列表。选择"myextension"然后点击“应用修改”按钮。

4.10 改进的商品配送系统

eZ Publish 3.8 允许为你的网络商店定制配送选项（例如：配送费用可以依赖于商品分类）。这可以通过实现配送处理器来实现。配送处理器是一个 PHP 类，它提供了保存商品集（购物篮或订单）信息与计算配送费用的机制。eZ Publish 没有内建任何配送处理器，因此您需要开发您自己的配送处理器来支持配送选项。不能在一个站点入口中同时使用两个配送处理器。配送处理器必须在"Handler"INI 配置中指定（参阅“INI 配置”章节）。

当用户查看商品页面时，配送费用不会被显示也不会被包含在商品费用中。向系统的购物篮中添加商品后，系统会计算它们的配送价格，配送价格会显示在商品列表中，也会被包含到订单总价中。配送处理器不但返回配送费用，也会返回配送选项简介和一个到配送管理界面（可以修改配送选项）的链接。这些信息会在购物篮中与配送费用一起显示。在订单确定页面和管理界面中的订单视图中也会显示配送费用和配送选项简介。

INI 配置

"shop.ini"文件中"[ShippingSettings]"分区定义了用于计算配送费用的配送处理器。在这个分区内，可以指定一下配置：

- "Handler"配置指定使用的配送处理器
- "RepositoryDirectories[]"数组指定 eZ Publish 从哪里搜索内建的配送处理器。
- "ExtensionDirectories[]"数组指定 eZ Publish 从哪里搜索附加的配送处理器。默认情况下，eZ Publish 会在 extension 的"shippinghandlers"子目录中搜索。

例 1

以下配置可以在"shop.ini"重设文件中"[ShippingSettings]"下添加：

```
[ShippingSettings]
Handler=ezcustom
RepositoryDirectories[]=kernel/classes/shippinghandlers
```

这些配置会要求 eZ Publish 使用"kernel/classes/shippinghandlers/ezcustomshippinghandler.php"定义的配送处理器。

例 2

如果您有一个扩展"myextension"，它包含一个配送处理器"mycost"，您可以在"shop.ini"重设文件中加入如下配置：

```
[ShippingSettings]
Handler=mycost
ExtensionDirectories[]=myextension
```

或

```
[ShippingSettings]
Handler=mycost
RepositoryDirectories[]=extension/myextension/shippinghandlers
```

这些配置会要求 eZ Publish 使用"extension/myextension/shippinghandlers/mycostshippinghandler.php"中定义的配送处理器。

创建新的配送处理器

本章为希望开发新配送处理器的程序员（只适用于熟悉 PHP 的程序员）提供了一些有用的信息。请注意，不建议修改 eZ Publish 内核，因此您应该通过扩展来实现。

实现细节

配送处理器是一个 PHP 类，它实现以下方法：

```
/**
* Invoked to get shipping information for given product collection.
* \public
* \static
*/
function getShippingInfo( $productCollectionID );


/*
* Invoked when shopping basket contents is changed
```

```
* to update shipping info/cost appropriately.
* \public
* \static
*/
function updateShippingInfo( $productCollectionID );


/**
* Invoked when the associated product collection is removed
* to clean up shipping information.
* \public
* \static
*/
function purgeShippingInfo( $productCollectionID );
```

处理器被 eZShippingManager 类中相同方法调用。

```
$shippingInfo = eZShippingManager::getShippingInfo( $productCollection );
```

getShippingInfo()方法只用于调用自定义的配送处理器的 getShippingInfo()方法。这个方法返回指定商品的配送信息，它是一个包含以下信息的哈希：

名称	类型	描述
description	字符串	配送选项简介。
cost	整形	指定商品的配送费用。
management_link	字符串	链接到配送管理界面，它可以用来修改配送选项。

下一章解释了如何实现您自己的配送处理器。

创建您自己的处理器

以下文字描述了如何实现一个简单的用于演示目的的配送处理器。

1. 在"extension"目录中创建如下子目录：
   - myextension
   - myextension/settings
   - myextension/shippinghandlers
2. 在"myextension/shippinghandlers/"中创建"mycostshippinghandler.php"（这个文件必须包含 PHP 类"MyCostShippingHandler"）并添加如下内容：

```
<?php
class MyCostShippingHandler
{
    function getShippingInfo( $productCollectionID )
```

```
    {
        return array(
        'description' => 'Manual',
        'cost' => 10,
        'management_link' => '/shop/basket/' // dummy
        );
    }
    function purgeShippingInfo( $productCollectionID )
    {
        // nothing to purge
    }
    function updateShippingInfo( $productCollectionID )
    {
        // nothing to update
    }
}
?>
```

3. 在"myextension/settings"目录中创建"shop.ini.append.php"并添加如下内容：

```
[ShippingSettings]
Handler=mycost
ExtensionDirectories[]=myextension
```

4. 要在 eZ Publish 中启用您的扩展，登录到 eZ Publish 管理界面，选择“设置”标签，然后选择左侧的“扩展”。系统会列出选存的扩展。选中"myextension"然后点击“应用修改”按钮。

系统会为您站点中所有被购买的商品添加固定的配送费用。

如果您需要更复杂的配送选项，您可以使用 http://ez.no/community/contribs/examples/sample_shipping_handler 中的高级示例或开发您自己的处理器。

4.11 多货币

本章的目的是介绍与解释 eZ Publish 3.8 内建的多货币特性。不熟悉 eZ Publish 网络商店子系统的用户，应该先阅读“基本概念”中的“网络商店”一章。下一章会帮助您理解以下问题：

- 自定义价格和自动价格的概念
- 什么是基本自定义价格
- 如何为自动价格指定您的价格取舍规则
- 如何使用汇率（自动汇率和自定义汇率）

- 什么是基础货币
- 如何管理货币
- 什么是优先货币
- 如何对多价格商品使用附加的视图模板
- “商品一览”界面的用途
- 如何使用默认的汇率更新处理器
- 如何创建您自己的汇率更新处理器
- 如何将您的商品转换为多价格格式

4.11.1 自定义价格与自动价格

多价格数据类型允许您为每个商品设置多货币价格。如果您使用，例如：五种货币，那么商品总是有五种价格。但是，您不需要全部手动为它们输入价格，尽管您可以这样做。您需要至少为商品指定一个价格，它被称为基本自定义价格（简称为基础价格）。系统会使用适当的汇率自动将基础价格转换为不同货币的价格（在本例中，其余四种货币的价格）。这些被称为自动价格。与此相反，用户手动输入的价格被称为自定义价格。

自定义价格完全独立于汇率。当谈到“基础价格”时，我们是在说一个特殊的自定义价格，它会被用来计算其它自动价格。所有其它的自定义价格被成为非基础价格。请注意，非基础自定义价格独立于基础价格。如果您修改非基础价格，没有自动价格会被修改。

在对象编辑界面中，自动价格通常被标记为“自动”，自定义价格则不是。基础价格没有特殊标记，因为它的值通常在自动价格之后，在其余非基础自定义价格之前（如果有）。

如果您对某些货币中的自动价格不满意，您可以设置一个非基础自定义价格。这个值会独立于基础价格存在。

如果您删除一个非基础自定义价格，系统会为这种货币创建一个自动价格。

如果您删除基础价格，系统会：

- 设置最早的非基础自定义价格为新的基础价格。
- 删除旧的基础价格并为其创建一个新的自动价格。
- 根据新的基础价格更新所有的自动价格。

建议每个商品至少有一个自定义价格。如果您删除某个商品的所有自定义价格，系统会对所有货币使用自动价格。

例

价格你使用如下表所示的货币及汇率。

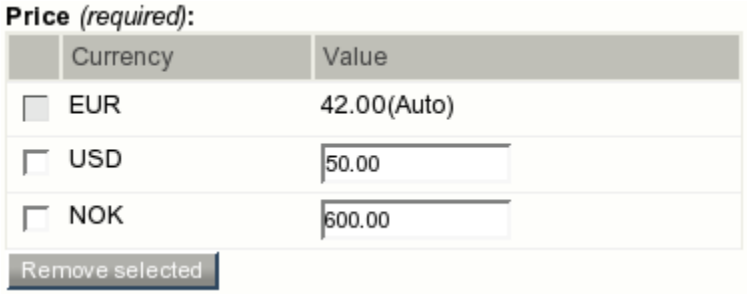
货币代码	汇率
NOK	1.32015
EUR	0.16380
USD	0.2

如果您指定，例如：$50，为基础价格，系统会为这个商品自动计算其余两种自动价格（参阅下图）。

这些自动价格是通过将$50 转换为欧元和挪威克朗来得到的：

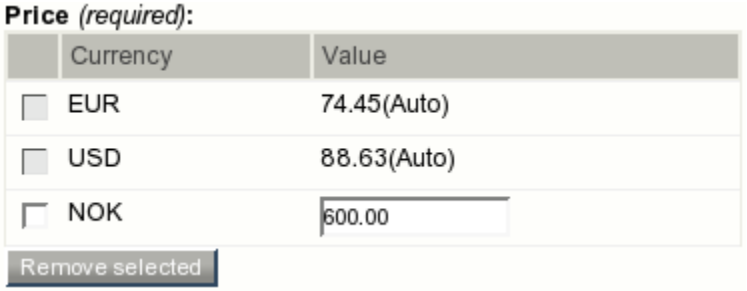
- EUR/USD 交叉汇率 = 0.16380 / 0.19500 = 0.84000
- EUR 价格 = 50*0.84 = 42.00
- NOK/USD 交叉汇率 = 1.32015/0.19500 = 6.77000
- NOK 价格 = 50*6.77 = 338.50

如果您觉得这个商品在某些国家，例如：挪威，价格太高，您可以为 NOK 创建自定义价格。结果如下图。



如您所见，有两个自定义价格（基础价格$50 和非基础价格 600nok）和一个自动价格（€42）。请注意您总是可以删除非基础自定义价格，从而系统会自动设置自动价格。

如果您删除基础价格（$50），则非基础自定义价格（600nok）会成为新的基础价格，因此系统会自动更新自动价格（如下图）。



这些自动价格会通过将 600 nok 变换为欧元和美元来得到：

- EUR/NOK 交叉汇率 = 0.16380 / 1.32015 = 0.12408
- EUR 价格 = 600*0.12408 = 74.45
- USD/NOK 交叉汇率 = 0.19500 / 1.32015 = 0.15
- NOK 价格 = 600*0.15 = 88.63

4.11.2 自动价格的精度取舍

价格精度取舍过程由"shop.ini"中的"[MathSettings]"决定。可用的配置：

- RoundingPrecision
- RoundingType
- RoundingTarget

您可以重设这个文件并指定您自己的精度取舍类型。

RoundingPrecision

这个配置指定小数点后应该保留几位数字。默认情况下，精度为 2。通常没有必要设置，例如："RoundingPrecision=3"，因为数据库中每个价格只保留两位小数。

RoundingType

这个配置定义应该使用哪中取舍方法。有以下方法可选。

配置	描述	属性值	取舍值
RoundingType=round	0.00000	0.124	0.12
RoundingType=round	0.00000	0.125	0.13
RoundingType=ceil	向上取舍	0.121	0.13

RoundingType=floor	向上取舍	0.129	0.12
RoundingType=none	0.00000	1/3	*0.333333...

* 只要数据库只保留两位数字，结果将被保存为 0.33

请注意，上例假设 RoundingPrecision 为 2。

"RoundingType"默认值为"round"。

RoundingTarget

这个配置允许您强制取舍到特定的目标。例如，如果您倾向于用$2.49 而不是$2.50 作为零售价（尾数定价法），您可以要求系统在自动价格中用 9 作为最后一位数字。

默认的"RoundingTarget"为"false"。

请参阅下表了解具体的用法。

配置	实际值	取舍值
RoundingTarget=false	89.468543	89.47
RoundingTarget=5	89.468543	89.45
RoundingTarget=99	89.468543	89.99

请注意，上例假设 RoundingPrecision 为 2 且 RoundingType 为"round"。

修改过 rounding 配置之后，不要忘记更新自动价格。

4.11.3 汇率

有两种汇率：

- 自动汇率
- 自定义汇率

自动汇率

自动汇率从外部资源自动取得。可以用内建的"eZECB"处理器或开发自己的更新处理器从欧洲中央银行的网站获得汇率信息。请注意，您应该参阅“汇率更新处理器”一章在"ExchangeRatesUpdateHandler"INI 配置中指定需要使用的处理器，否则系统不能自动更新汇率信息。

一种货币的自动汇率为要买入一个本币的货币单位需要花费的这种货币的金额。本币（或称为基础货币）由"BaseCurrenty"INI 配置决定（参阅“汇率更新处理器”章节）。建议（但不是必须）在这里指定一种现存货币。

欧洲央行的网站允许程序取得基于欧元(EUR)的汇率。如果您用"eZECB"处理器并将美元(USD)设置为本币，系统会取得基于 EUR 的汇率然后相对于 USD 计算自动汇率。请注意如果 USD 的汇率没有取得，您会收到一个错误信息。

例

假设以下相对于 EUR 的汇率在欧洲央行的网站上公布：

货币	汇率
NOK	7.85620
USD	1.20940
UAH	not available

如果您将 EUR 作为本币，系统会设置如下的自动汇率：

货币	自动汇率
EUR	1.00000
NOK	7.85620
USD	1.20940
UAH	N/A

如果您不指定本币，结果相同。

如果您将 USD 作为本币，系统会基于 USD 计算自动汇率，如下表：

货币	自动汇率
EUR	0.82685 (1 / 1.20940)
NOK	6.49594 (7.85620 / 1.20940)
USD	1.00000
UAH	N/A

如果您指定 UAH 为本币，当更新汇率时，系统会显示以下错误信息：“当更新自动汇率时，无法为货币兑换 EUR/UAH 计算交叉汇率”。因为 UAH 的汇率没有成功从欧洲央行网站取得，系统将无法完成基于 UAH 的自动汇率计算。

自定义汇率

您可以为某种货币指定一个固定的汇率，因此这个汇率会取代自动汇率（自动汇率不会被使用，因此会显示为灰色）。请注意，自定义汇率必须与自动汇率使用相同的本币。如果您有，例如：五种，货币并使用相对于 EUR 的自动汇率。如果您希望为某种货币指定自定义汇率，确保这个汇率也相对于 EUR。

如果您要为您所有的货币指定自定义汇率，您可以用其它货币作为本币。建议（但不是必须）用现存的货币作为本币。

例

假设您有四种货币：USD,EUR,NOK 和 UAH。您可以把 USD 作为本币并且指定相对于这种货币的自定义汇率，例如：

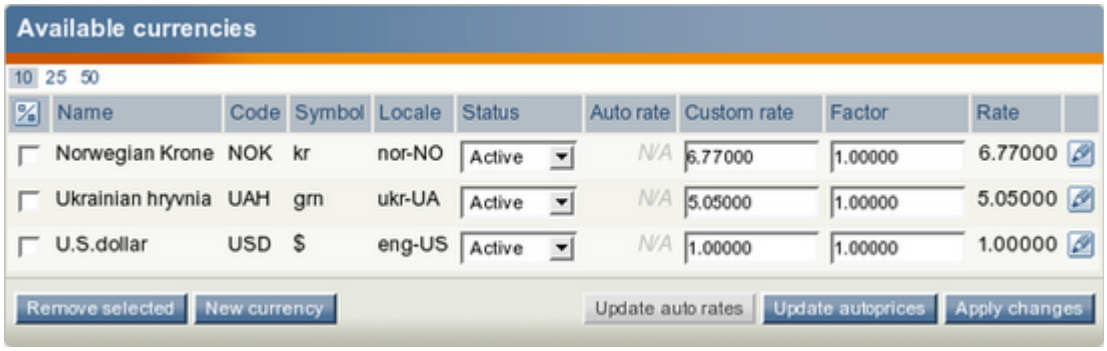
- USD 1
- EUR 0.84
- NOK 6.52
- UAH 5.05

如果您随后删除"USD"货币，EUR,NOK 和 UAH 的自定义汇率不会改变。

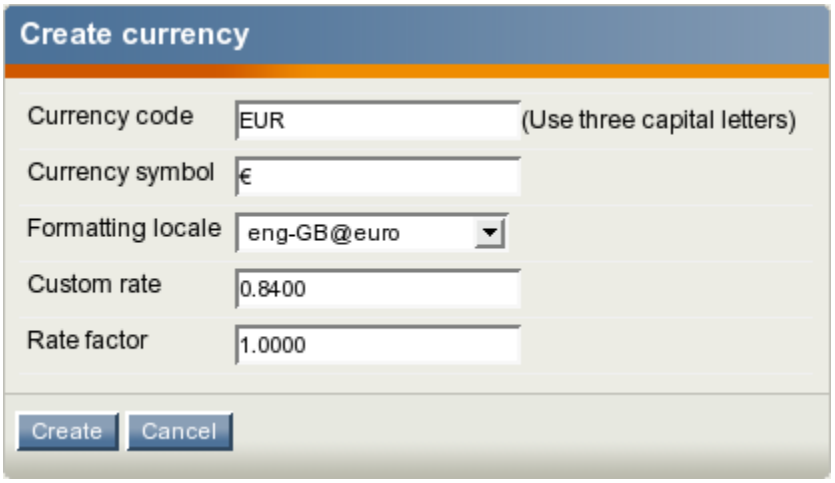
4.11.4 创建货币

管理界面允许您为网络商店系统添加新货币。假设您已经有三种货币（USD,UAH 和 NOK）并且您希望添加另外一种货币（EUR）。下例演示了当您已经有三种货币的情况下，如果添加 EUR。

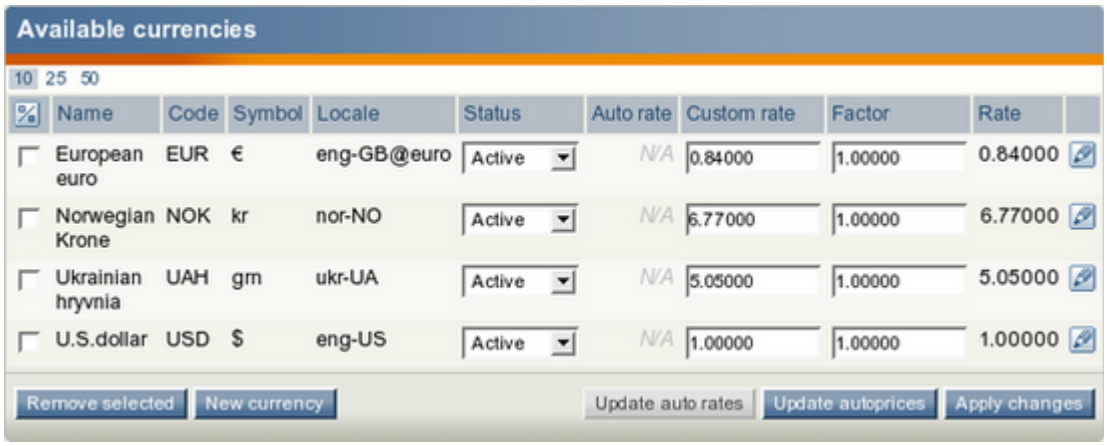
1. 在管理界面中选择“网络商店”标签，然后选择左侧的“货币”，再选择“新建货币”按钮。（这个界面也可以通过"/shop/currencylist"访问。）



系统会显示货币编辑界面，这里您可以指定新货币的属性（参阅下图）。



2. 指定货币的属性（稍后解释）并点击“创建”按钮。系统会添加新货币（如下图）。



请注意，创建一种新货币之后，系统会将您所有的商品在这种货币中的自动价格设置为 0。建议在您完成货币管理之后，点击“更新自动价格”按钮。这会要求系统为所有商品更新自动价格。

货币代码

三个字符的货币代码（例如："USD","EUR"等等）通常被用来代表这种货币。这个参数是必须的。这个代码可以被作为货币的唯一标识符。您不能将同一个货币代码指派给两种货币。货币代码有三个大写英文字符构成并且经常（但不一定）与 ISO 4217 标准一致。

一旦指定了货币代码，系统将可以显示货币名称（例如：“欧元”，“美元”等等）。这些货币名可以通过自定义版本的"currencynames.tpl"（在"templates/shop/"目录中）来修改。这个模板不会影响站点用户的功能。这些货币名称只在管理界面中显示。如果您创建了一种新货币，并为其指定了未知的货币代码（例如："ABC"），它没有包含在"currencynames.tpl"，系统会用“未知货币”作为这种货币的名称。

货币符号

货币符号是一个字符串，它会在数字附近显示（例如："$","€"等等）。货币符号在日常生活中被用来表示一个数字代表某种货币的金额。这个参数不是必须的。如果没有定义货币符号，使用这种货币的用户会看到没有货币符号的数字价格。请注意，如果您不能输入需要的货币符号，您可以从您的浏览器窗口或文本编辑器中复制/粘贴。

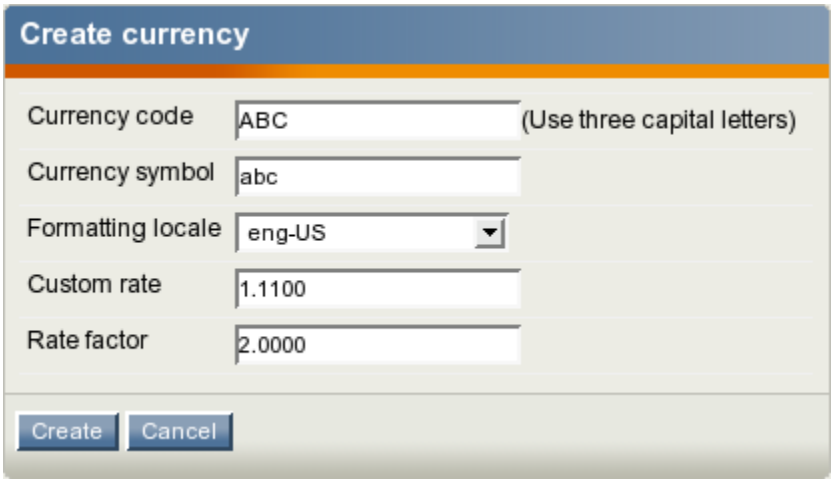
格式化区域

一个格式化区域是一个用于格式化价格的区域。这个参数是必须的。您可以从区域下拉框中选择需要的区域。默认情况下，当前系统区域（由"site.ini"重设文件中的"[RegionalSettings]"下的"Locale"指定）被选中。可用的区域由"share/locale"中的 INI 文件决定。

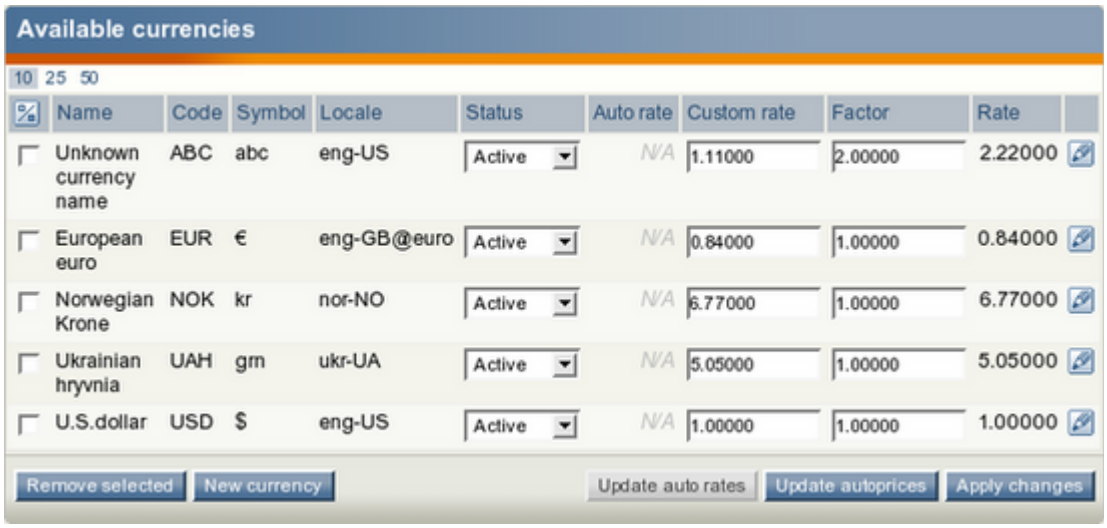
指定格式化区域之后，系统会用选定区域 INI 文件中"[Currency]"下的"DecimalSymbol","ThousandsSeparator","FractDigits"和"PositiveFormat"来格式化价格。请注意，"Symbol","Name"和"ShortName"配置不会影响价格的格式。

例

假设创建一种新货币"ABC"，并指定它的属性如下。



因为"ABC"代码没有在"currencynames.tpl"模板中列出，系统会将它显示为"未知货币"（参阅下图）。



这个问题可以通过创建一个自定义版本的"currencynames.tpl"来解决，这个模板在 standard 界面下的"templates/shop/"子目录中。要重设这个模板，将这个文件复制到"admin"界面的"templates/shop/"子目录下并编辑它。在"set_currency_names"中添加一对键值/值。参阅以下代码：

```
{set currency_names = hash( 'ABC', 'AB-Currency',
'AUD', 'Australian dollar',
...
'USD', 'U.S.dollar' ) }
```

清除 eZ Publish 缓存后，系统会将这种货币显示为"AB-Currency"。

"share/locale"目录中的"eng-US.ini"包含以下配置：

```
[Currency]
Symbol=$
Name=US Dollar
ShortName=USD
DecimalSymbol=.
ThousandsSeparator=,
FractDigits=2
PositiveSymbol=
NegativeSymbol=-
PositiveFormat=%c%p%q
NegativeFormat=%c%p%q
```

因为您为"ABC"选择了"eng-US"区域，基于"DecimalSymbol","ThousandsSeparator","FractDigits"和"PositiveFormat"配置，

系统会用"."作为小数点，","作为千分符，小数点后保留两位数字并且会把货币符号放在数字之前。（"Symbol","Name"和"ShortName"不会被使用。）

假设某种商品在这种货币内的价格为 550 个货币单位。在本例中，使用这种货币的用户看到的价格格式如下：

abc550.00

自定义汇率

这个必须的参数告诉系统应该用哪个汇率来计算这种货币的自动价格。默认情况下，自定义汇率被设置为 0，因此系统会为这种货币使用自动汇率。但是，可以指定一个非 0 的固定自定义汇率用于计算这种货币的自动价格。

汇率系数

这个必须的参数目的在于支持一种用于计算这种货币的自动价格的虚拟汇率。如果指定了一个非 0 的自定义汇率，系统会用汇率系数乘以自定义汇率来计算最终的汇率，否则系统会用自动汇率乘以汇率系数。汇率系数的默认值为 1。下表揭示了如何通过自动汇率，自定义汇率和汇率系数计算最终汇率。

自定义汇率	汇率系数	自动汇率	最终汇率
0	1	0.85	0.85
0	1.4	0.85	0.85*1.4 = 1.19
0.75	1	0.85	0.75
0.75	1.4	0.85	0.75*1.4 = 1.05

状态

货币的状态可以为：“可用”或“不可用”。当您创建一种新货币时，状态会被自动设为“可用”。不可用货币对站点用户不可见。换言之，如果您不希望您的用户使用某种货币，您可以禁用这种货币。

不可用货币在货币列表中以红色显示。如下图。

4.11.5 编辑货币

管理界面允许您编辑货币。以下文字揭示了具体步骤。

1. 在管理界面中选择“网络商店”，然后选择左侧的“货币”。系统会显示货币列表。找到目标货币，然后点击“编辑”按钮。系统会显示货币编辑界面，你可以在这里修改如下属性（在前一章中详细解释过）：
   - 货币代码
   - 货币符号
   - 格式化区域
   - 自定义汇率
   - 汇率系数
2. 指定需要的货币属性
3. 点击“保存修改”按钮

改变货币状态

货币编辑界面不能用于修改货币的状态。以下文字揭示了具体的步骤。

1. 在管理界面中选择“网络商店”标签，然后选择左侧的“货币”。系统会显示货币列表。
2. 从状态下拉框中选择需要的状态（可以对多种货币选择状态）。
3. 点击“应用修改”来保存您的修改。

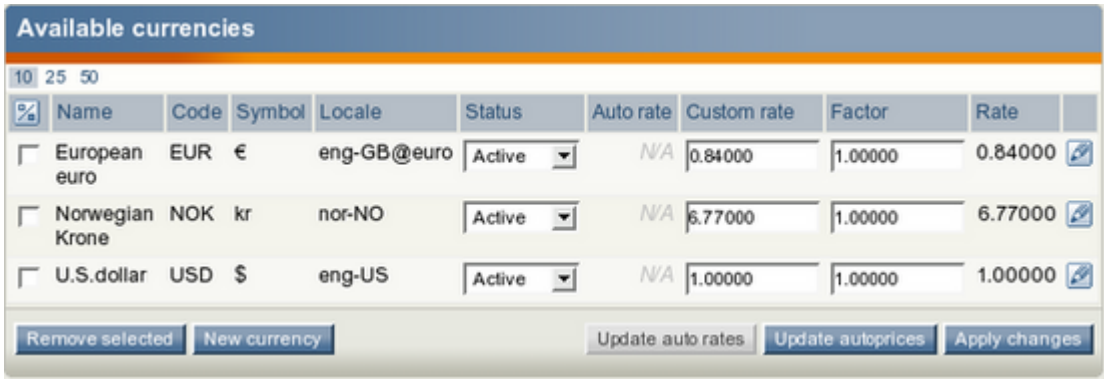
为多个货币修改汇率

你可以在货币列表中同时修改多个货币的汇率和/或汇率系数。

更新自动汇率

要更新自动汇率，参阅以下步骤：

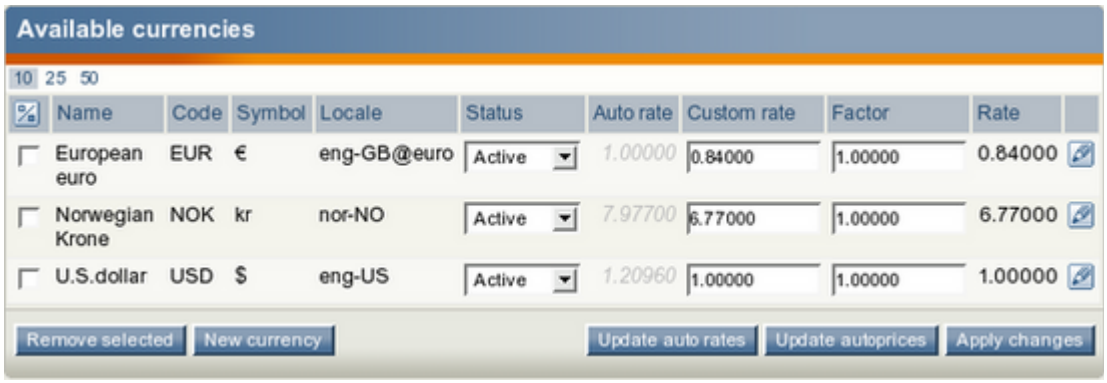
1. 在管理界面中选择“网络商店”标签，然后选择左侧的“货币”。系统会显示货币列表。
2. 如果“更新自动汇率”按钮不可用，这意味着系统不能更新自动汇率因为没有指定自动更新处理器。下图演示了这种情况，所有的货币都被指定了自定义汇率，因而系统不会使用自动汇率。在本例中，自动汇率以灰色显示。因为没有更新任何内容，"N/A"在“自动汇率”列中显示。



3. 因为没有指定更新处理器，“更新自动汇率”按钮不可用。要启用这个按钮，您应该参阅“汇率更新处理器”

章节指定更新处理器并清除 eZ Publish 缓存。“更新自动汇率”按钮会变为可用。

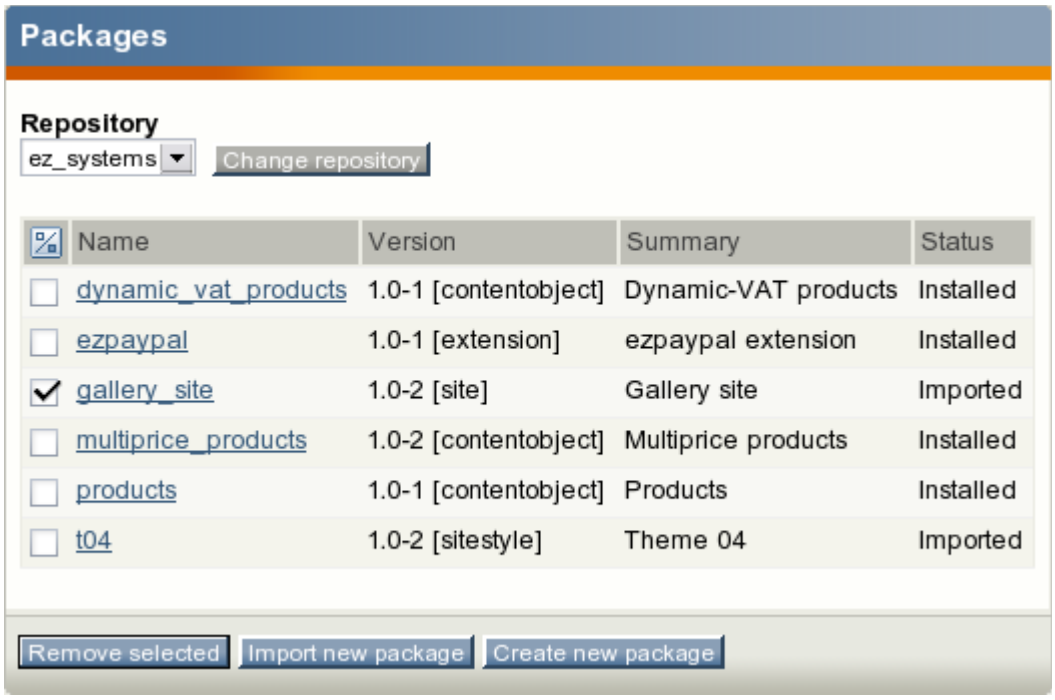
4. 点击“更新自动汇率”按钮。自动汇率会被自动更新。



上图演示了一种情况，自动汇率用 EUR 作为本币（EUR 自动汇率为 1.00000）。如您所见，取得的自动汇率仍然以灰色显示，因为自定义汇率仍然存在。但是，您可以删除自定义汇率来启用自动汇率。

启用自动汇率

如果有非 0 的自定义汇率，这种货币的自动汇率不会被使用。如果您删除自定义汇率或将它设置为 0，自动汇率会被启用。例如：如果您从“自定义汇率”列删除所有值并点击“应用修改”按钮，系统会开始使用自动汇率（参阅下图）。

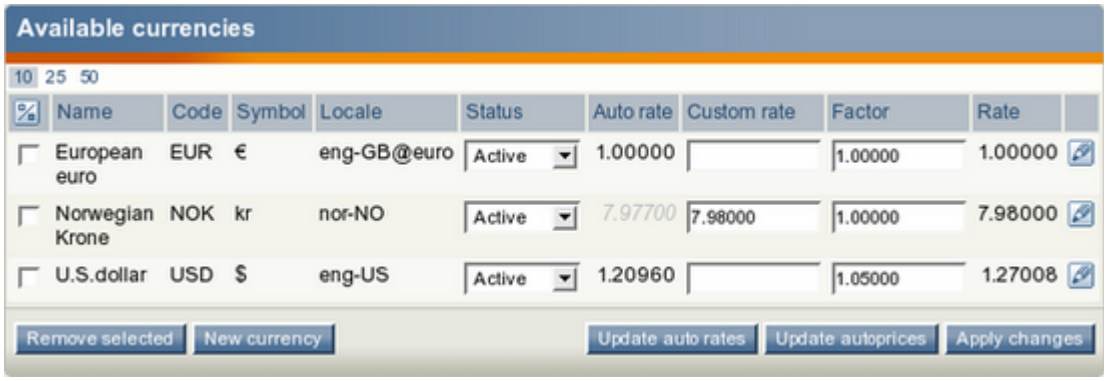


请注意，修改汇率不会自动更新您商品的自动价格。当您完成对货币的修改，建议您点击“更新自动”按钮来要求系统更新您商品的自动价格。

修改自定义汇率和/或汇率系数

您可以同时为多个货币修改自定义汇率和/或汇率系数。以下文字揭示了具体步骤。

1. 在管理界面中选择“网络商店”，然后选择左侧的“货币”。系统会显示货币列表。
2. 在自定义汇率列中指定自定义汇率并/或在汇率系数列中指定汇率系数（可以对多个货币指定这些值）。
3. 点击“应用修改”按钮来保存您的修改。系统会自动重新计算最终汇率并在“汇率”列中显示。



上图演示了修改了 NOK 的自定义汇率和 USD 的汇率系数之后的页面。

请注意，点击“应用修改”按钮不会更新您商品的自动价格。所以在您完成对货币的修改后，建议您点击“更新自动价格”按钮。这会要求系统更新所有商品的自动价格。

4.11.6 删除货币

你可以（但不建议）从网络商店系统中删除货币。如果您需要希望对您的用户隐藏某些货币，您应该把它们的状态设置为“不可用”，而不是从系统中删除它们。

请注意，删除一种货币会删除所有商品中这种货币的价格。如果您的某些商品使用这种货币作为本币，这将会造成很多问题。

例

假设您的某些商品使用 USD 作为本币（如下表）。

货币	商品 1		商品 2	
USD	50.00	基本自定义价格	50.00	基本自定义价格
NOK	338.50	自动价格	600.00	非基本自定义价格

EUR	42.00	自动价格	42.00	自动价格

这些基本价格会被删除，如果您删除 USD 货币（参阅下表）。

货币	商品 1		商品 2	
NOK	0.00	自动价格	600.00	自定义基本价格
EUR	0.00	自动价格	74.45	自动价格

如您所见，删除基础价格会造成不期望的结果。如：所有商品的价格被设为 0。因此，不建议删除货币。

以下文字揭示了如何从网络商店系统中删除一种或多种货币。

1. 在管理界面中选择“网络商店”标签，然后选择左侧的“货币”。系统会显示货币列表。
2. 用复选框勾选希望删除的货币。
3. 点击“删除所选”按钮。

4.11.7 优先货币

用户可以选择一种“可用”货币作为“优先货币”。系统会接着对用户使用这种货币。这可以通过访问"shop/preferredcurrency"，从下拉框中选择需要的货币，再点击“设置”按钮来实现。您也可以通过访问"shop/preferredcurrency/(currency)/NOK"（用需要设置的货币代码替换 NOK）来设置优先货币。您可以在您的站点中创建不同货币的链接或使用如下的工具栏。

如果没有指定优先货币，系统会使用"shop.ini"重设文件中"[CurrencySettings]"下的"PreferredCurrency"指定的货币作为优先货币。强烈建议您在这个设置中指定一种可用的货币。

当您的用户查看一个多价格商品时（请参阅“多价格商品显示模板”了解更多），可以限制只显示优先货币的价格。注意，如果您不在"PreferredCurrency"中指定优先货币，系统会在用户第一次查看这个商品的时候将价格显示为 0。

例

假设您有两种货币：EUR,NOK 并且"shop.ini"重设文件中包含如下配置：

```
[CurrencySettings]

PreferredCurrency=USD
```

如果用户第一次访问您的站点，系统并不知道用户的优先货币，因此会尝试使用默认值。但是"USD"没有在系统中定义，因此没有这种货币的价格。系统会将价格显示为 0，并使用您的区域设置中的货币符号。

添加工具栏

您可以为用户添加一个优先货币工具栏来允许用户改变优先货币。要启用优先货币工具栏，

在"toolbar.ini"重设文件中添加如下配置：

```
Tool[]=preferred_currency
```

这个设置会要求系统用"standard"界面中的"templates/toolbar/full/preferred_currency.tpl"模板显示工具栏。

站点管理员的优先货币

站点管理员可以通过访问"shop/preferredcurrency"，从货币下拉框中选择优先货币然后点击“设置”按钮来设置优先货币。这个界面也可以通过选择“网络商店”标签，然后选择左侧的“优先货币”来访问。选中的货币会用于在商品一览页面显示价格。

4.11.8 多价格商品

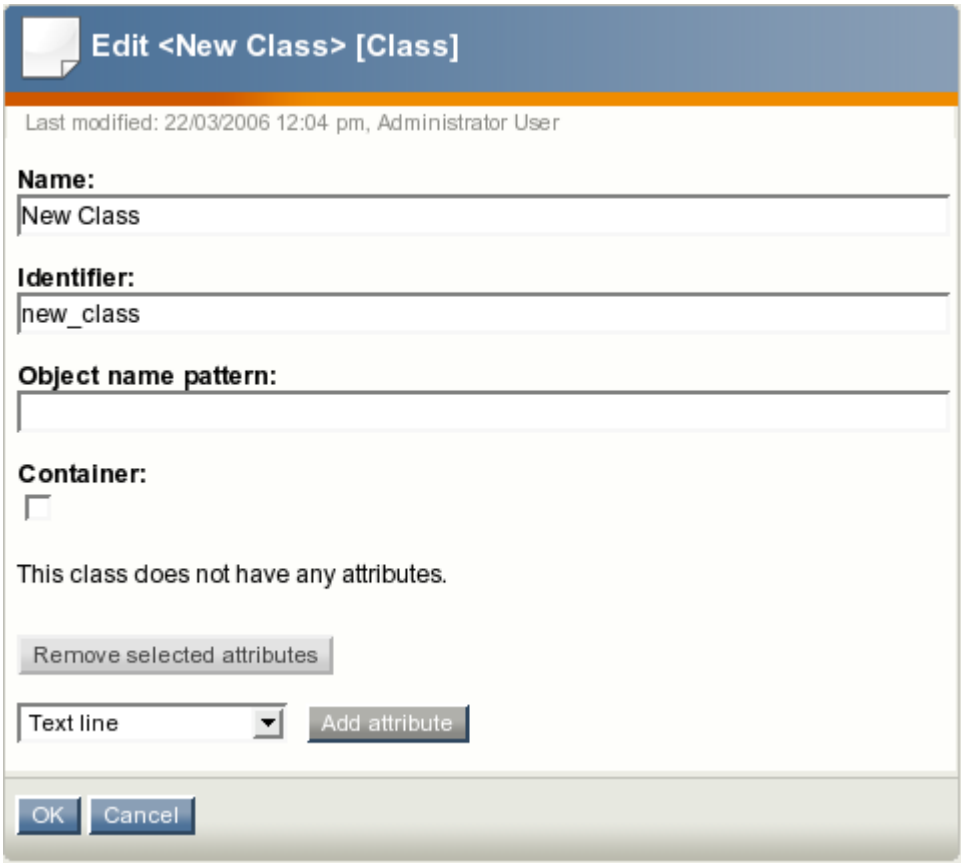
一个商品由一个内容对象代表（至少有一个节点），它包含商品本身的信息以及一个价格。价格可以由一个“价格”或“多价格”数据类型的属性表示。这两种数据类型与系统联系更加紧密并将内容对象与网络商店系统连接起来。它们主要的区别在于“价格”数据类型为每个商品指定一个价格而“多价格”数据类型为一个商品指定多个价格（每种货币一个价格）。请注意，简单价格商品不支持多种货币。

一个内容类只能包含一个“价格”属性或一个“多价格”属性。不能在购物篮中同时使用简单价格商品和多价格商品。因此，不建议在您的站点中同时使用价格和多价格数据类型。

如果您要使用多价格商品，您应该至少创建一个内容类，它应该包含一个多价格类型的属性（稍后解释）。这个类的对象会被系统识别为多价格商品。如果您已经在使用简单价格商品，您可以将它们自动转换为多价格商品（参阅“升级网络商店”章节）。

创建商品类

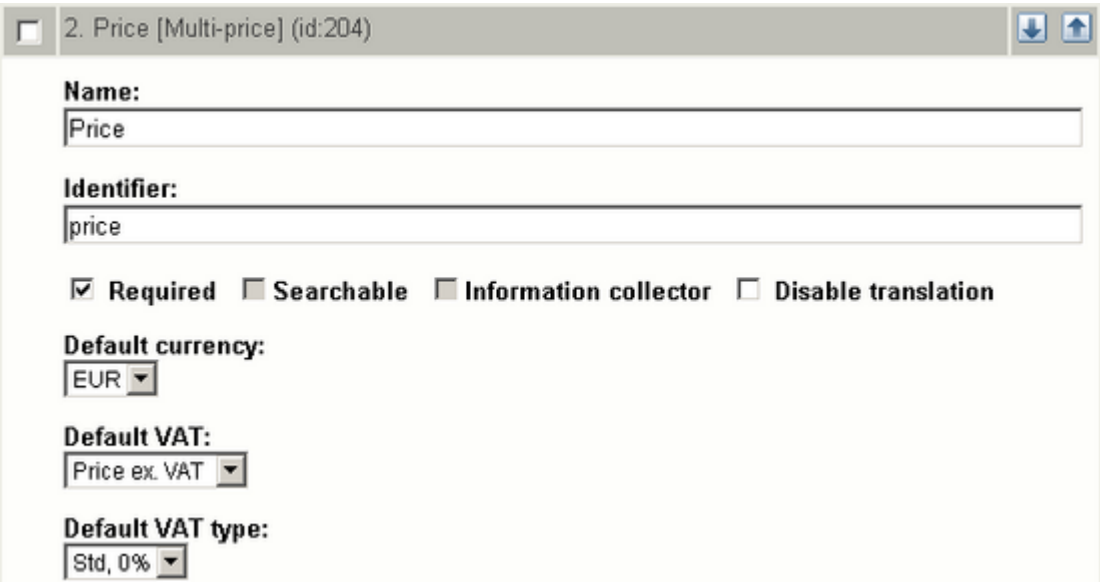
在管理界面中选择“设置”标签，然后选择左侧的“类”，再选择“内容”类组并点击“新建类”按钮。系统会显示类编辑界面（如下图）。



为这个类指定名称，标识符，对象名模式和容器标记并通过下拉框添加需要的属性。

多价格属性

要添加多价格属性，从下拉框中选择多价格数据类型，点击“添加属性”按钮并对新属性做如下编辑。



建议用"price"作为它的标识符（这个标识符在附加视图模板中使用）。您必须选择一种预定义的货币作为“默认货币”。默认情况下，这种货币会被用于自定义价格。

例

假设有四种预定义货币：NOK,EUR,USD,UAH，并且您创建了一个类“商品”，它有一个多价格属性。如果您选择 EUR 作为默认货币，系统会为新的“商品”对象使用 EUR 创建基本价格，并为 NOK,USD 和 UAH 创建自动价格。当创建新商品时，系统会将基本价格设为 0.00，但是您可以为其指定希望的价格（例如：€50）。您也可以删除这个价格并创建其他货币的基本价格（例如：$60）。

添加属性后，点击“确定”保存类。

请注意，如果您需要不同的结构来保存您的商品，您可以创建不同的多价格类。如果您销售，例如：计算机硬件，您可能需要创建不同的类：“显示器”，“打印机”，“扫描仪”等等。在本例中，您可以在商品一览页面用商品类作为过滤条件。

创建商品

如果您有多价格类，您可以为这种类创建对象（多价格商品）。

多价格商品显示模板

默认情况下，系统会向用户显示所有货币的价格。这是由"standard"界面中的"templates/content/datatype/view/ezmultiprice.tpl“模板决定的。

如果您希望之显示优先货币的价格，您可以使用"base"界面中的"override/templates/datatype/multiprice.tpl"。要做到这一点，在您的"override.ini.append.php"中添加如下配置：

```
[multiprice]
Source=content/datatype/view/ezmultiprice.tpl
MatchFile=datatype/multiprice.tpl
Subdir=templates
```

建议您为站点用户配置货币工具栏，以允许用户修改优先货币。参阅“为用户添加工具栏”章节。

以下模板可以用于显示多价格商品：

- design/base/override/templates/full/multiprice_product.tpl
- design/base/override/templates/line/multiprice_product.tpl
- design/base/override/templates/embed/multiprice_product.tpl
- design/base/override/templates/listitem/multiprice_product.tpl

要使用这些模板，在 override.ini.append.php 中添加如下内容：

```
[multiprice_product_full]
Source=node/view/full.tpl
MatchFile=full/multiprice_product.tpl
Subdir=templates
Match[class_identifier]=myproduct

[multiprice_product_line]
Source=node/view/line.tpl
MatchFile=line/multiprice_product.tpl
Subdir=templates
Match[class_identifier]=myproduct

[multiprice_product_embed]
Source=content/view/embed.tpl
MatchFile=embed/multiprice_product.tpl
Subdir=templates
Match[class_identifier]=myproduct

[multiprice_product_listitem]
Source=node/view/listitem.tpl
```

```
MatchFile=listitem/multiprice_product.tpl

Subdir=templates

Match[class_identifier]=myproduct
```

用实际的类标识符替换"myproduct"。（要查看类标识符，在管理界面中选择“设置”，选择左侧的“类”，选择“内容”类组然后找到您的多价格商品类。）

如果您要使用这些模板，您必须对所有站点入口设置"CacheViewPreferences[full]"。要设置这个选项，编辑所有的"site.ini.append.php"，并且如果"[ContentSettings]"已经包含如下内容：

```
CachedViewPreferences[full]=<list_of_user_preferences>
```

那么，您需要在行尾追加一个":"和"user_preferred_currency"，例如：

```
CachedViewPreferences[full]=admin_navigation_content=0;

admin_navigation_details=0;<...>;admin_bookmarkmenu=1;

admin_left_menu_width=13;user_preferred_currency=''
```

注意，这一行配置通常会很长。在本例中我们用<...>代替中间的内容。

如果"[ContentSettings]"下没有这样一行，则创建它。

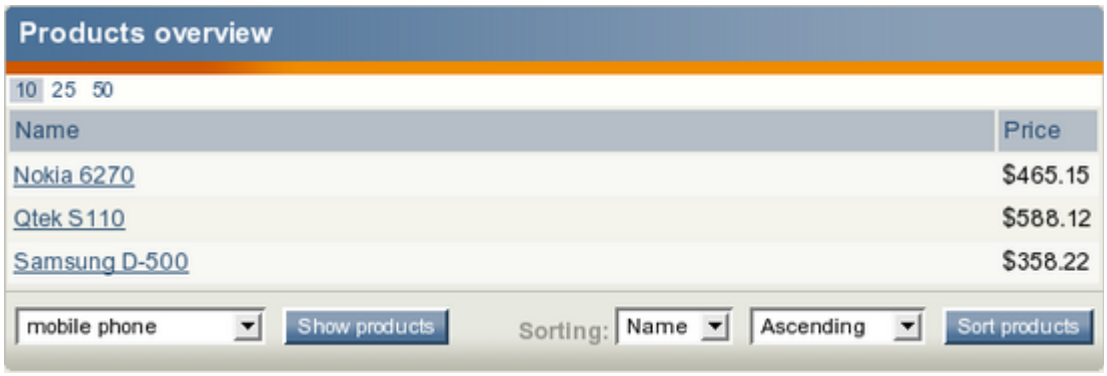
```
CachedViewPreferences[full]=user_preferred_currency=''
```

如果不指定这个配置，您的用户将无法切换优先货币（因为缓存界面将不会刷新）。

4.11.9 商品一览

用户/站点管理员可以查看所有的商品，按照类分组并按照价格或商品名称排序。商品一览界面可以通过"shop/productsoverview"访问。站点管理员也可以通过选择“网络商店”标签，然后选择左侧的“商品一览”来访问商品一览界面。

下图演示了管理界面中的商品一览界面。请注意，对多价格商品，只有管理员的优先货币中的价格会被显示。



按类过滤

上图中显示了所有“移动电话”类的商品。如果您希望查看其他类的商品，从下拉框中选择类名，然后点击“显示商品”按钮。

选择排序方法

上图中商品按照字母表排序。如果您希望用其它方法排序（如：按价格排序），选择需要的排序参数然后点击“商品排序”按钮。

4.11.10 汇率更新处理器

汇率更新处理器可以从外部资源获得最新的汇率信息，因此可以用来更新自动汇率。您应该在"ExchangeRatesUpdateHandler"（在下一章中解释）中指定要使用的处理器，否则系统不能更新自动汇率。您可以使用内建的"eZECB"处理器从欧洲央行网站获得汇率信息或开发您自己的更新处理器。

配置

"shop.ini"中的"[ExchangeRatesSettings]"定义了用于更新汇率的更新处理器。可以对以下内容配置：

- "RepositoryDirectories[]"数组指定了 eZ Publish 从哪里搜索内建的更新处理器。具体的处理器由"ExchangeRatesUpdateHandler"指定。
- "ExtensionDirectory[]"数字指定了 eZ Publish 应该搜索处理器的扩展目录。默认情况下，eZ Publish 会在您扩展目录中的"exchangeratehandlers"子目录搜索。具体的处理器由"ExchangeRatesUpdateHandler"指定。
- "ExchangeRatesUpdateHandler"指定使用的处理器
- "BaseCurrenty"为自动汇率指定本币。默认本币为"EUR"。建议（但不一定）指定某一种现存的货币为本币。

"shop.ini"中的"[ECBExchangeRatesSettings]"定义了"eZECB"特有的配置。"ServerName","ServerPort"和"RatesURI"的配置组合可以用来指定包含汇率信息 XML 文件的网址。

例 1

可以在"shop.ini"重设文件中"[ExchangeRatesSettings]"下添加如下内容：

```
ExchangeRatesUpdateHandler=eZECB

RepositoryDirectories[]=kernel/shop/classes/exchangeratehandlers

ExtensionDirectories[]

BaseCurrency=EUR
```

这些配置会要求 eZ Publish 使用内建的汇率更新处理器。这个处理器位于"kernel/shop/classes/exchangeratehandlers/ezecb/ezecbhandler.php"并使用 EUR 作为本币。

例 2

您可以开发您自己的汇率更新处理器来扩展系统满足特定需求。例如：如果您有一个扩展"myshop"，它包含一个更新处理器"mybank"，您可以将以下内容加入"shop.ini"重设文件中。

```
[ExchangeRatesSettings]

ExchangeRatesUpdateHandler=mybank

ExtensionDirectories[]=myshop/classes
```

或

```
[ExchangeRatesSettings]

ExchangeRatesUpdateHandler=mybank

RepositoryDirectories[]=extension/myshop/classes/exchangeratehandlers/
```

这些配置要求 eZ Publish 使用"extension/myshop/classes/exchangeratehandlers/mybank/mybankhandler.php"作为汇率更新处理器。

例 3

可以在"shop.ini"重设文件中的"[ECBExchangeRatesSettings]"章节下作如下配置：

```
ServerName=http://www.ecb.int

ServerPort=80

RatesURI=stats/eurofxref/eurofxref-daily.xml
```

这些配置会要求 eZECB 处理器从 http://www.ecb.ini:80/stats/eurofxref/eurofxref-daily.xml 出导入汇率信息。

创建新的处理器

本章为希望开发新的汇率更新处理器的程序员（只适用于熟悉 PHP 的程序员）提供了一些有用的信息。请注意，不建议修改 eZ Publish 内核文件，因此您应该在扩展中实现新的处理器。以下列表揭示了如何实现您自己的汇率更新处理器。

1. 在 eZ Publish 的 extension 目录中创建以下子目录：
   - myextension
   - myextension/settings
   - myextension/exchangeratehandlers

- myextension/exchangeratehandlers/mybank

2. 在"myextension/exchangeratehandlers/mybank"中创建"mybankhandler.php"。这个文件必须包含一个 PHP 类"MyBankHandler"。这个类应该继承"eZExchangeRatesUpdateHandler"（在 kernel/shop/classes/exchangeratehandlers/ezexchangeratesupdatehandler.php）并实现"initialize"和"requestRates"函数。"initialize"函数在初始化处理器对象时被调用。它允许对处理器对象的变量进行初始化（例如：从 INI 文件中读取配置）。"requestRates"函数实现了实际的汇率更新过程。这个函数会将取得的汇率数组设置在"$RateList"成员变量中。汇率数组的格式如下：

```
$RateList = array( 'currencyCode1' => 'rateValue1',

.....

'currencyCodeN' => 'rateValueN' );
```

在"myextension/settings"中创建"shop.ini.append.php"并添加如下内容：

```
[ExchangeRatesSettings]

ExchangeRatesUpdateHandler=mybank

ExtensionDirectories[]=myextension
```

这会要求 eZ Publish 使用"extension/myextension/exchangeratehandlers/mybank/mybankhandler.php"作为处理器。

3. 要激活您的扩展，登录到 eZ Publish 管理界面。选择“设置”标签，然后选择左侧的“扩展”。系统会列出可用的扩展。选择"myextension"然后点击“应用修改”按钮。

4.11.11 升级网络商店

如果您为了使用多货币而刚刚升级了您的 eZ Publish，您需要创建货币并用"bin/php/convertprice2multiprice.php"将您的商品转化为多价格格式。注意，您应该从 eZ Publish 根目录执行这个脚本。

这个脚本会遍历所有包含“价格”数据类型的类和对象并为其创建一个多价格数据类型的属性。这意味着您的重设规则对原来的类和对象仍然有效（请参阅“多价格商品显示模板”章节，如果您只希望显示优先货币价格）。

建议创建所有货币（包括您的区域货币）并在执行这个脚本之前检查汇率。

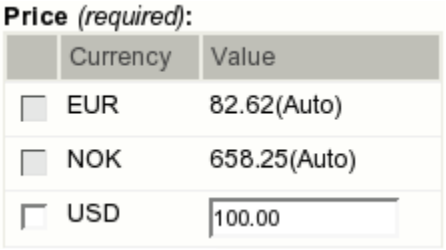
例

如果您的区域货币为 USD，您的某个简单价格商品的价格为$100 并且您希望使用多货币特性，您可以创建如下货币：

货币代码	汇率
EUR	1
NOK	7.9675
USD	1.2104

这些汇率可以是从欧洲央行网站取得的自动汇率。

成功执行"convertprice2multiprice.php"脚本之后，价格为$100 的商品的简单价格会被自动转化为多价（包含一个自定义价格：基本价格）和两个自动价格。如下图所示。



如您所见，这个脚本用现存的汇率将基本价格从 USD 转化为 EUR 和 NOK。

如果您忘记创建您的区域货币，这个脚本会自动创建它并将自定义汇率设置为 1。货币表会如下表所示：

货币代码	汇率
EUR	1
NOK	7.9675
USD	1

这些汇率是错误的（1 USD 不等于 1 EUR）。转化后的自动价格也将是错误的因为它们是基于错误的汇率。

4.12 视图缓存

缓存是一种广泛使用的技术，它将经常使用的信息保留在临时的高速存储介质中来提高系统的性能。当原始数据的提取或计算/生成相对于存取缓存所需的系统开销很大（通常指访问时间）时，缓存的效用会非常明显。一旦数据被保存在缓存中，之后可以直接从缓存中提取而不需要重新从系统中提取或重新计算原始数据，因此平均访问时间会比较低。

eZ Publish 内建了功能强大的缓存机制，它可以改善系统的性能。本章描述了 eZ Publish 缓存系统的重要部分，这部分被成为“内容视图缓存”（简称“视图缓存”）。这种机制只对 content 模块的"view"和"pdf"视图有效。

节点视图缓存

这一章描述了当节点被访问时，视图缓存是如何被生成的。

当 eZ Publish 被请求输出关于一个节点的信息时（通过系统 URL 或虚拟 URL），它会执行与"content"模块的"view"视图相关的程序代码。执行结束后，视图将结果返回给模块，模块负责将结果返回给系统的其余部分。eZ Publish 会自动生成一个称为"module_result"的数组，它包含包含了以下信息：运行的哪个

模块，执行的哪个视图，视图的输出结果等等。视图的实际输出内容（例如：通过某个节点模板生成的 XHTML 代码）被设置为"$module_result.content"并且通过以下代码在 pagelayout 中显示：

```
{$module_result.content}
```

当 pagelayout 被生成时，{$module_result.content}部分会被实际的输出替换。如果视图缓存被启用，模块的完整结果会被缓存。这意味着"module_result"变量的内容会被存入"var/example/cache/content"目录（"example"由"site.ini"重设文件中的"VarDir"设置）中的一个缓存文件。

请注意，eZ Publish 会根据角色和用户偏好设置为一个节点创建多个视图缓存。这意味着，例如：对于不同用户（已经登录，且具有不同的权限/偏好设置）会被提供不同的缓存，而匿名用户或有相同权限和用户偏好设置的会被分别提供相同的缓存。换言之，当启用视图缓存时，"content"模块的"view"视图只有在系统不能定位一个缓存文件才会被执行，否则缓存的内容会被直接嵌入 pagelayout。请注意 pagelayout 默认情况下不会被缓存。

另外一个需要注意的问题是，视图缓存还依赖其它参数。例如：

- 视图模式
- 语言
- URL 中的视图参数
- Layout（例如：'print' layout 会使用不同的缓存文件）
- 其它

例

假设节点 46 对应“关于我们”页面并且自定义的模板"aboutpage.tpl"对这个特殊节点重设了默认的"node/view/full.tpl"。

现在虚拟 URL "http://www.mysite.com/company/about"和系统 URL“http://www.mysite.com/content/view/full/46”都指向这个页面。当访问它们中的某个 URL 时，系统会执行"content"模块的"view"视图并使用"46"作为节点 ID，"full"作为视图模式。包含“关于我们”信息的 XHTML 结果会由"aboutpage.tpl“生成。输出会被设置到"$module_result.content"并随后被嵌入 pagelayout 中。

如果视图缓存对"full"视图模式启用，模块的完整内容会被报存在一个缓存文件。每个缓存文件有一个类似于"46-122bc591bf62e87a4e9ddcb5ba352bc4.cache"的很长的文件名。下一次访问关于我们页面时，系统不会执行"content"模块的"view"视图，而是会直接从缓存文件中装载内容。

$node 变量

在 eZ Publish 3.9 之前的版本中，在启用视图缓存被禁用时，"$node"变量可以在 pagelayout 中使用。从 3.9 版本开始，无论是否启用视图缓存，这一变量在 pagelayout 中都不再可用。如果您仍然需要访问当前节点，建议从$module_result 中取得需要的信息（例如："$module_result.node_id"代表当前节点的 ID）。

PDF 缓存

以下文字解释了当访问某个站点页面（内容节点）的 PDF 版本时，视图缓存是如何生成的。注意，在 eZ Publish4.0 中 PDF 导出机制已经不再推荐使用并且会被从将来的发行版本中剔除。

当 eZ Publish 被请求生成一个节点的 PDF 版本时，它会执行"content"模块的"pdf"视图对应的程序代码。系统会使用"pdf.tpl"提取实际的页面内容（节点封装的对象中的属性内容），然后使用"execute_pdf.tpl"生成 PDF 文件。与节点视图缓存不同，系统不会把输出内容嵌入$module_result.content。这些默认的模板位于"standard"界面中的"templates/node/view"子目录。

如果为"pdf"视图启用缓存，PDF 文件会被缓存。这意味着系统会将实际 PDF 的一份副本保存为"var/example/cache/content"（"example"可以在"site.ini"重设文件中的"VarDir"设置）中的一个缓存文件。

例

如果节点 46 是“关于我们”页面，则访问"http://www.mysite.com/content/pdf/46"会执行"content"模块的"pdf"视图。系统会生成关于我们页面的 PDF 版本并将它显示给用户。

如果为"pdf"视图启用视图缓存，结果 PDF 文档会被缓存在一个文件中，例如"46-3579d18de31e99fc84d2d9a5f113c3be.cache"。请注意，这个文件可以用 PDF 阅读器打开（某些情况下，您需要将文件扩展名改为".pdf"）。

4.12.1 配置视图缓存

视图缓存机制默认被启用。但是，您可能希望在网站开发阶段将它禁用（否则您对节点模板做的改动每次必须清除缓存才能生效）。您可以在"site.ini"重设文件中的"[ContentSettings]"章节下添加以下内容来禁用视图缓存：

```
ViewCaching=disabled
```

注意，强烈建议您在开发完成之后重新启用视图缓存。这可以通过将"disabled"变为"enabled"来完成：

```
ViewCaching=enabled
```

"site.ini"中"[ContentSettings]"章节下的"CachedViewModes"配置控制为哪些视图或视图模式启用视图缓存。默认值为"full","sitemap"视图模式和"pdf"视图：

```
CachedViewModes=full;sitemap;pdf
```

但是，注意"pdf"视图已经不推荐使用。

如果您需要为一个特定的模板禁用缓存，在这个模板的第一行加入如下内容：

```
{set-block scope=global variable=cache_ttl}0{/set-block}
```

这会在当前模板内将全局变量"cache_ttl"设置为 0。"cache_ttl"变量包含以秒为单位的 TTL(Time To Live)。TTL 为 0 意味着结果不应该被缓存。TTL 为"-1"意味着缓存永远都不应该过期，参阅下例：

```
{set-block scope=global variable=cache_ttl}-1{/set-block}
```

角色

不同角色组合会使用不同的缓存文件。这意味着，尽管视图缓存被启用，模板中仍然可以存在基于角色的条件分支。

用户偏好设置

以下文字描述了如何处理用户偏好设置以及当前用户的偏好设置如何影响内容视图缓存的生成。

例如，任何时候如果用户通过管理界面执行以下操作：

- 启用/禁用收藏夹菜单
- 调整内容树的宽度（小/中/大）
- 选择子项目窗口的视图模式（列表/缩略图/详细）或每页的项目数（10/25/50）
- 改变不同窗口的可见性（预览/详细/翻译/位置/关联）

...或设置任何其它偏好设置，系统会执行"user"模块的"preferences"视图来保存这些选项。选择的偏好设置通过视图参数传递：

```
.../user/preferences/set/<name_of_preference>/<value>
```

例如

```
http://my.com/myadmin/user/preferences/set/admin_left_menu_width/medium
```

保存选择的值之后，"user"模块的"preferences"视图会重定向到上一次访问的页面。

如果您被缓存的模板中有基于用户偏好设置的条件分支，您应该指定哪些偏好组合会与不同的视图模式一起使用。这可以在"site.ini.append.php"中的"[ContentSettings]"章节下的"CachedViewPreferences"设置。

例

假设您为您的商品使用多种货币，并且对所有商品"node/view/full.tpl"已经被重设，因而可以用用户优先货币显示商品的价格。如果为"full"视图模式启用视图缓存，系统会为商品显示模板创建缓存。如果缓存忽略优先货币，当选择另外一个优先货币时，系统会返回刚才缓存的内容（换言之，货币以及价格不会改变）。

为了避免这种情况，您需要在"CachedViewPreferences[]"数组中用"full"作为键值，并在在数组中添加"user_preferred_currentcy"。要修改这个设置，编辑"site.ini"重设文件。如果"[ContentSettings]"章节下已经有如下设置：

```
CachedViewPreferences[full]=<list_of_user_preferences>
```

那么您应该在这一行的行尾追加一个":"和"user_preferred_currency"。例如：

```
CachedViewPreferences[full]=admin_navigation_content=0;
admin_navigation_details=0;<...>;admin_bookmarkmenu=1;
admin_left_menu_width=13;user_preferred_currency=''
```

注意，这一行通常会很长。在本例中我们用<...>代替了中间部分。

如果"[ContentSettings]"章节下没有这一行，则可以直接添加：

```
CachedViewPreferences[full]=user_preferred_currency=''
```

关联站点入口

"site.ini"重设文件中的"[SiteAccessSettings]"章节下的"RelatedSiteAccessList"设置控制当当前站点入口的视图缓存被清除后，还有哪些站点入口的视图缓存应该被清除。（这需要所有关联的站点入口使用相同的"VarDir"）。如果没有指定"RelatedSiteAccessList"，系统会使用"AvailableSiteAccessList"。

注意，从 eZ Publish 3.8 版本开始，缓存系统不再使用"content.ini"重设文件中"[VersionView]"章节下的"AvailableSiteDesignList"。在 3.7 版本以前，它可以包含一组界面，它们会在缓存清除之后被更新。

4.12.2 清除视图缓存

当一个对象的一个新版本（也包括第一个版本）被发布时，系统会自动清除以下项目的视图缓存。

- 所有指派给这个对象的已经发布的节点
- 父节点
- 与这个对象有相同关键字（如果至少有一个关键字类型的属性）的对象的节点。

此外，以下节点的视图缓存也会被清除：

- 一般类型关联和一般类型逆向关联的对象的节点
- "XML 嵌入"类型的逆向关联对象的节点

这是由"ClearRelationTypes"配置决定的。

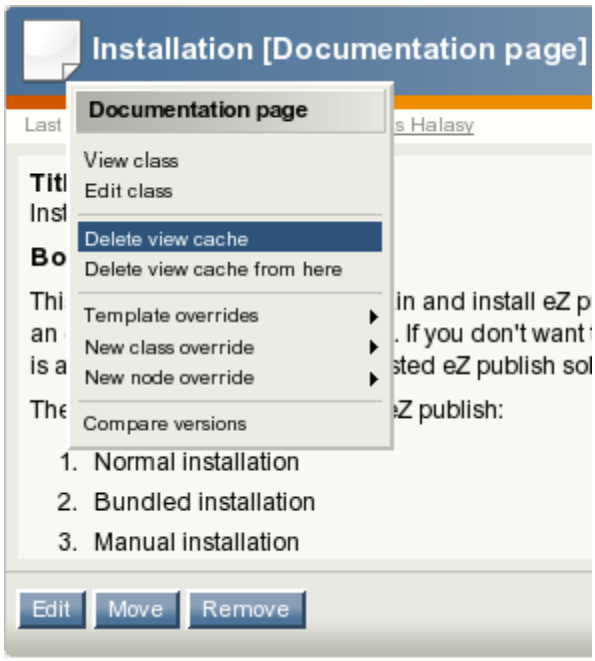
这种默认行为可以通过配置“智能视图缓存清除系统”来扩展。

注意，清除某个站点入口内的节点的视图缓存，也会清除相同节点在所有相关站点入口内的缓存。

使用管理界面

管理界面可以用来清除当前查看的节点的视图缓存：

1. 导航到您希望为其清除缓存的节点。换言之，确保目标节点正在被显示。
2. 在预览窗口的标题栏中，点击表示节点类型的图标（鼠标左键）并从上下文菜单中选择“清除视图缓存”。如下图：



您也可以选择“在此”清除整个子树的视图缓存（包括当前节点）。

使用脚本

可以通过在 eZ Publish 根目录中执行"bin/php/ezcontentcache.php"脚本来清除特定节点或子树的视图缓存。下例演示了具体步骤。

例 1

假设节点 46 为“关于我们”页面（http://www.mysite.com/company/about）并且您对这个节点的自定义模板做了修改。如果启用了视图缓存，除非您清除节点 46 的视图缓存，否则您的修改不会生效：

1. 进入 eZ Publish 根目录
2. 用以下命令行执行脚本

```
./bin/php/ezcontentcache.php --clear-node=46
```

或

```
./bin/php/ezcontentcache.php --clear-node=/company/about
```

这会清除节点 46（以及同一个对象的其它位置），它们的父节点，有相同关键字的对象节点，“一般”类型的关联和逆向关联对象节点以及“XML 嵌入”类型的逆向关联对象节点的视图缓存。如果您需要清除多个节点的视图缓存，用逗号","分隔它们的节点 ID 或者虚拟 URL：

```
./bin/php/ezcontentcache.php --clear-node=46,59,63
```

这个脚本会清除所有指定节点对应的视图缓存。

例 2

假设节点 72 为公司新闻文件夹（http://www.mysite.com/company/news），它包含很多新闻文章。要清除这个文件夹以及子孙节点的视图缓存，参阅如下步骤：

1. 进入 eZ Publish 根目录
2. 运行以下命令行：

```
./bin/php/ezcontentcache.php --clear-subtree=72
```

或

```
./bin/php/ezcontentcache.php --clear-subtree=/company/news
```

如果您需要清除多个子树，用逗号","分隔它们的虚拟 URL（或节点 ID）：

```
./bin/php/ezcontentcache.php --clear-subtree=/company/news,/partners
```

4.12.3 视图缓存智能清除

智能视图缓存清除系统（在本章中简称"svcs"），允许您设置自定义的规则来控制当对象被发布时，哪些节点的视图缓存应该被清除。这个特性默认被禁用，因此当某个对象被发布时，系统只会清除以下节点的视图缓存：

- 这个对象的所有被发布的节点
- 父节点
- 与这个对象有相同关键字的对象节点（如果至少有一个关键字类型的属性）

此外，以下节点的视图缓存也会被清除：

- “一般”类型的关联和逆向关联对象节点
- “XML 嵌入”类型的逆向关联对象节点

这是由"ClearRelationType"控制的。

如果您希望启用智能视图缓存清除系统，确保您的"viewcache.ini"重设文件中包含以下内容：

```
[ViewCacheSettings]
SmartCacheClear=enabled
```

以上配置会要求系统在默认缓存清除规则的基础上，根据这个文件中的规则清除其它节点的视图缓存。这个配置文件通常包含一个通用配置"[ViewCacheSettings]"和多个特殊的章节，这些章节定义了哪些附加节点的视图缓存应该被清除。

注意：

这些附加节点的视图缓存也会根据 svcs 规则被清除（参阅例 5）。

这些特殊的配置章节由类标识符命名。

当一个已发布的对象被修改时，svcs 将对象的类标识符作为一个输入参数。它会检查这个对象的类标识符并在"viewcache.ini"重设文件中找与类标识符相同的章节。在匹配到的章节下定义的规则会应用到这个节点的所有祖先节点（在节点的"path_string"中列出的节点）。如果发布的对象有多个节点/位置，svcs 会对它们逐个处理。以下的列表揭示了 svcs 如何处理发布对象的每个节点：

1. 自下向上扫描节点"path_string"中的祖先节点（扫描的最大数由"MaxParents"配置指定）
2. 对每个祖先节点执行以下操作：
   - 检查节点的类标识符
   - 如果标识符在"DependentClassIdentifier[]"数组中，将这个祖先节点加入到“附加节点”中
3. 如果"ObjectFilter[]"配置为空，清除附加节点的视图缓存。否则，检查附加节点的类标识符，然后之清除那些类标识符在"ObjectFilter[]"数组中列出的节点的视图缓存。在两种情况下，缓存都会通过"ClearCacheMethod[]"中定义的方法来清除。

从 eZ Publish 3.9 版本开始，当特定类型的对象被修改时，您还可以清除一组特定对象的视图缓存。如果"AdditionalObjectIDs[]"数组包含一组对象 ID（不是节点 ID），系统会清除这些对象所有位置（节点）的视图缓存，不论这些节点是否是被发布的对象节点的祖先节点。

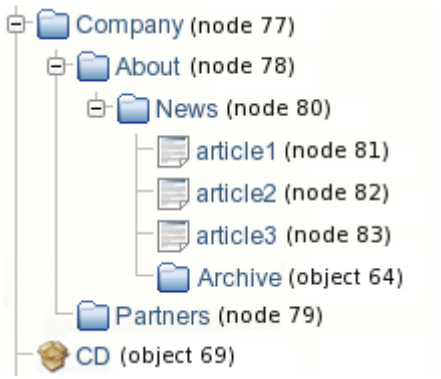
下表详细解释了以上讨论过的各种配置。

名称	类型	描述
DependentClassIdentifier	类标识符数组	指定哪些内容类会被识别为“依赖类”。如果当前对象节点的某个祖先节点的类标识符在"DependentClassIdentifier"中列出，它会被加入到附加节点中。附加节点的视图缓存会用下一个参数中指定的方法清除。
ClearCacheMethod	字符串数组	指定应该用什么方法清除附加节点的缓存。这个配置是一个字符串数组，只支持六种可能的配置（参阅下表）。 名称 / 描述 object：清除这个对象所有位置（节点）的视图缓存。 parent：清除父节点的视图缓存。 relating：清除“一般”关联和“一般”逆向关联对象节点的视图缓存；清除“XML 嵌入”类型逆向关联对象节点的视图缓存（根据"ClearRelationTypes"配置）。 keyword：清除与当前对象有相同关键字的对象节点的视图缓存。 siblings：清除当前节点的所有兄弟节点的视图缓存。 all：清除所有以上列出的视图缓存。
ObjectFilter	对象 ID 数组	如果指定了，只有附加节点中那些对象 ID 在"ObjectFilter"中列出的节点的视图缓存才会被清除。
MaxParents	整数	设置对于每个节点，它有多少个祖先节点会被检查。如果不指定则 svcs 会扫描它所有的祖先节点。

AdditionalObjectIDs	对象 ID 数组	允许清除一组特定对象的视图缓存，无论这些对象节点是否是当前节点的祖先节点。

例 1

假设对如下的内容结构，视图缓存与 svcs 都被启用：



如果您不指定任何 svcs 规则，修改一篇文章会导致如下节点的视图被清除（默认的 svcs 行为）：

- 它所有发布的节点
- 他们的父节点
- 包含相同关键字的对象节点的视图缓存
- “一般”类型的关联和逆向关联对象节点的视图缓存
- “XML 嵌入”类型的关联对象节点的视图缓存。

如果"article2"对象只有一个位置，不包含关键字且不与任何其它对象关联，那么修改它会清除文章本身以及"News"文件夹的视图缓存。"About"和"Company"节点的视图缓存不会被清除。

但是，您可以通过在管理站点入口（例：admin)的"viewcache.ini.append.php“中添加如下配置来扩展系统的默认行为：

```
[article]

DependentClassIdentifier[]

DependentClassIdentifier[]=folder

ClearCacheMethod[]

ClearCacheMethod[]=object
```

现在，如果一篇文章被修改，系统会根据它的"path_string"从下向上依次提取它的祖先节点（"article2"的"path_string"为"/77/78/80/82/"），检查它们中哪些是文件夹节点，并清除这些文件夹节点的视图缓存。这意味着修改"article2"会导致清除"article2","News","About","Company"以及所有位于"Company"节点之上的祖先节点的视图缓存。

例 2

您可以限制提取祖先节点的深度：

```
[article]
DependentClassIdentifier[]
DependentClassIdentifier[]=folder
ClearCacheMethod[]
ClearCacheMethod[]=object
MaxParents=2
```

这会要求系统只处理节点的最下两级祖先节点。这意味着修改"article2"会导致"article2","News"和"About"节点的视图缓存被清除。"Company"节点的视图缓存不会被删除。

例 3

您可以用"ObjectFilter[]"数组来指定一组对象 ID，从而在"path_string"中列出的所有文件夹祖先节点中，只有那些对象 ID 在"ObjectFilter[]"中列出的节点的视图缓存才会被清除。

```
ObjectFilter[]
ObjectFilter[]=<object_id1>
ObjectFilter[]=<object_id2>
...
```

假设在例 1 中的"Company"对象 ID 为 74（它的节点 ID 为 77），您可以在管理站点入口（例：admin）的"viewcache.ini.append.php"中指定如下设置：

```
[article]
DependentClassIdentifier[]
DependentClassIdentifier[]=folder
ClearCacheMethod[]
ClearCacheMethod[]=object
ObjectFilter[]
ObjectFilter[]=74
```

如果"article2"被修改，svcs 会扫描"path_string"中的祖先节点（节点 80,78,77,...）并检查它们的对象 ID 是否在"ObjectFilter[]"数组中列出，因此"Company"页面会被包含在附加节点中。系统将只会清除"article2","News"和"Company"节点的视图缓存（根据默认的行为以及附加的 svcs 规则）。"About"页面的视图缓存不会被清除，因为它的对象 ID 不是 74。

例 4

如果您希望在特殊类型的对象被修改时，清除一组特定对象的视图缓存，您必须在"AdditionalObjectIDs[]"数组中指定一组对象 ID。假设"Archive"文件夹和"CD"商品的对象 ID 为 64 和 69，您可以在管理站点入口（例：admin）的"viewcache.ini.append.php"中指定以下配置：

```
[article]
AdditionalObjectIDs[]
AdditionalObjectIDs[]=64
AdditionalObjectIDs[]=69
```

这会要求系统当一篇文章（任何文章）被修改时，总是清除"Archive"文件夹和"CD"商品的视图缓存。这意味着修改"article2"会导致"article2","News","Archive"和"CD"的视图缓存。

例 5

假设您已经在管理站点入口的"viewcache.ini.append.php"中指定了如下配置：

```
[article]
AdditionalObjectIDs[]
AdditionalObjectIDs[]=69


[product]
AdditionalObjectIDs[]
AdditionalObjectIDs[]=64
```

"[article]"配置块中的配置会要求系统当一篇文章被修改时，总是清除"CD"商品的视图缓存。这意味着。修改"article2"会导致"article2","News"和"CD"的视图缓存被清除。当清除"CD"商品的缓存时，svcs 会应用"[product]"配置块内的规则，因此"Archive"文件夹的视图缓存也会被清除。

4.12.4 预生成视图缓存

以上介绍的"请求时缓存"解决方案假定页面第一次被访问时，需要即时生成页面的视图缓存。此外，"发布时缓存"功能允许在创建/编辑节点时生成节点的视图缓存。这会对发布过程的速度有一些影响（对于有很多内容编辑的站点，不建议使用），但是却可以减少页面的响应时间。

"发布时缓存"特性默认被禁用。这个行为由"site.ini"中的"[ContentSettings]"章节下的"PreViewCache"设定。请注意，启用这一特性只会影响节点的"full"视图模式。当一个对象被发布时，系统会为这个对象所有的发布节点以及它们的父节点生成视图缓存。相同位置的"PreCacheSiteaccessArray"设置控制应该为哪些站点入口生成视图缓存（通常需要为公共站点入口生成视图缓存）。

如果您希望在发布对象时创建视图缓存，在您的管理站点入口的"site.ini.append.php"中的"[ContentSetings]"章节下添加如下配置：

```
PreViewCache=enabled
PreCacheSiteaccessArray[]
PreCacheSiteaccessArray[]=example
```

这会启用"发布时缓存"特性并要求系统为"example"站点入口创建发布时缓存。如果您有一个新闻文件夹，它包含很多文章，编辑其中一篇文章会导致文章本身和它的父节点（新闻文件夹）的视图缓存被重新生

成。当一篇新文章被发布在新闻文件夹中时，系统会为新文章创建视图缓存并重新生成新闻文件夹的视图缓存。

请注意，默认情况下，系统只会生成匿名用户的视图缓存。这是在"site.ini"重设文件中"[ContentSettings]"章节下的"PreviewCacheUsers"配置项来指定的。

4.13 通知

eZ Publish 内建的通知系统可以在各种事件发生时通知用户。例如：当对象被更新或发布时，当工作流被执行时，等等。

有两种内建的通知类型：

- 子树通知
- 协作通知

子树通知

用户可以订阅关于一棵子树的通知。例如：如果您在"Business"文件夹下有一组文章，用户可以对这个文件夹订阅子树通知。每次"Business"文件夹下有内容改动，系统都会给用户发送一封电子邮件。以改动会触发通知：

- 当一个新节点在子树中发布
- 当子树中现存内容被修改

用户可以选择对每个消息接收单独的邮件还是将多个消息合并为一封摘要邮件。

协作通知

eZ Publish 协作系统允许您与其他用户共同工作，从而允许您批准/拒绝其他人对内容的改动。例如：您可以指定在“标准”分区内所做的改动必须经过您的批准才能被发布（这可以通过创建一个新的工作流，在工作流中创建一个“审批”事件，并用"content-publish-before"触发器来触发这个工作流来实现）。如果某人（除了您和站点管理员）在“标准”分区内编辑了内容，系统会生成新的协作消息。例如：如果某人修改了文章"A“，系统会为您生成一个新的协作消息“文章 A 等待您的审批”，并为编辑文章的用户生成一个新的协作消息“文章 A 等待编辑的审批”。

要查看您的协作消息，在管理界面中选择“我的帐号”标签，然后选择左侧的“协作”。系统会显示协作界面，您可以在这里查看/批准/拒绝改动。

每次当系统为您生成一条新的协作消息时，也会给您发送一封电子邮件。

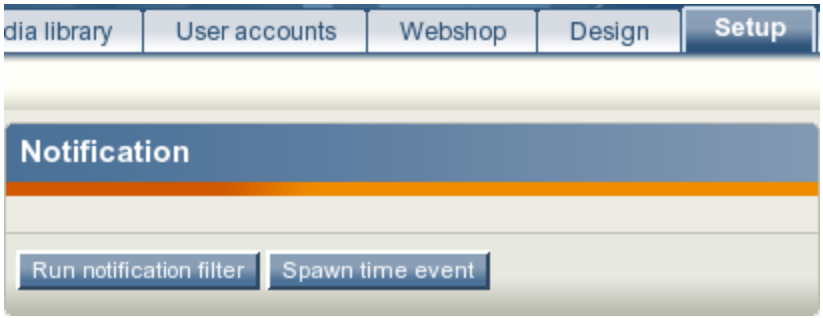
处理通知

在 eZ Publish 根目录中，有一个脚本"runcronjobs.php"。它负责处理工作流，通知和其它应该在后台运行的任务。如果您要使用通知系统，"runcronjobs.php"必须被定时执行。最常用的方法是设置一个计划任务

并每 30-60 分钟执行一次。请参阅“配置 cronjob ”和“运行 cronjob ”章节了解更多。

"runcronjobs.php"会根据"cronjobs/notification.php“中的指令启动主通知处理脚本"kernel/classes/notification/eznotificationeventfilter.php”。

如果您需要手动启动这个脚本，在管理界面中访问 URL"notification/runfilter"，并点击“运行通知过滤器”按钮（参阅下图）。



请注意，如果数据库中有大量的通知事件，处理通知可能会导致超时错误。由于这个原因，"notification"模块的"runfilter"视图只应用作测试和调试用途。

4.13.1 使用管理员界面

子树通知

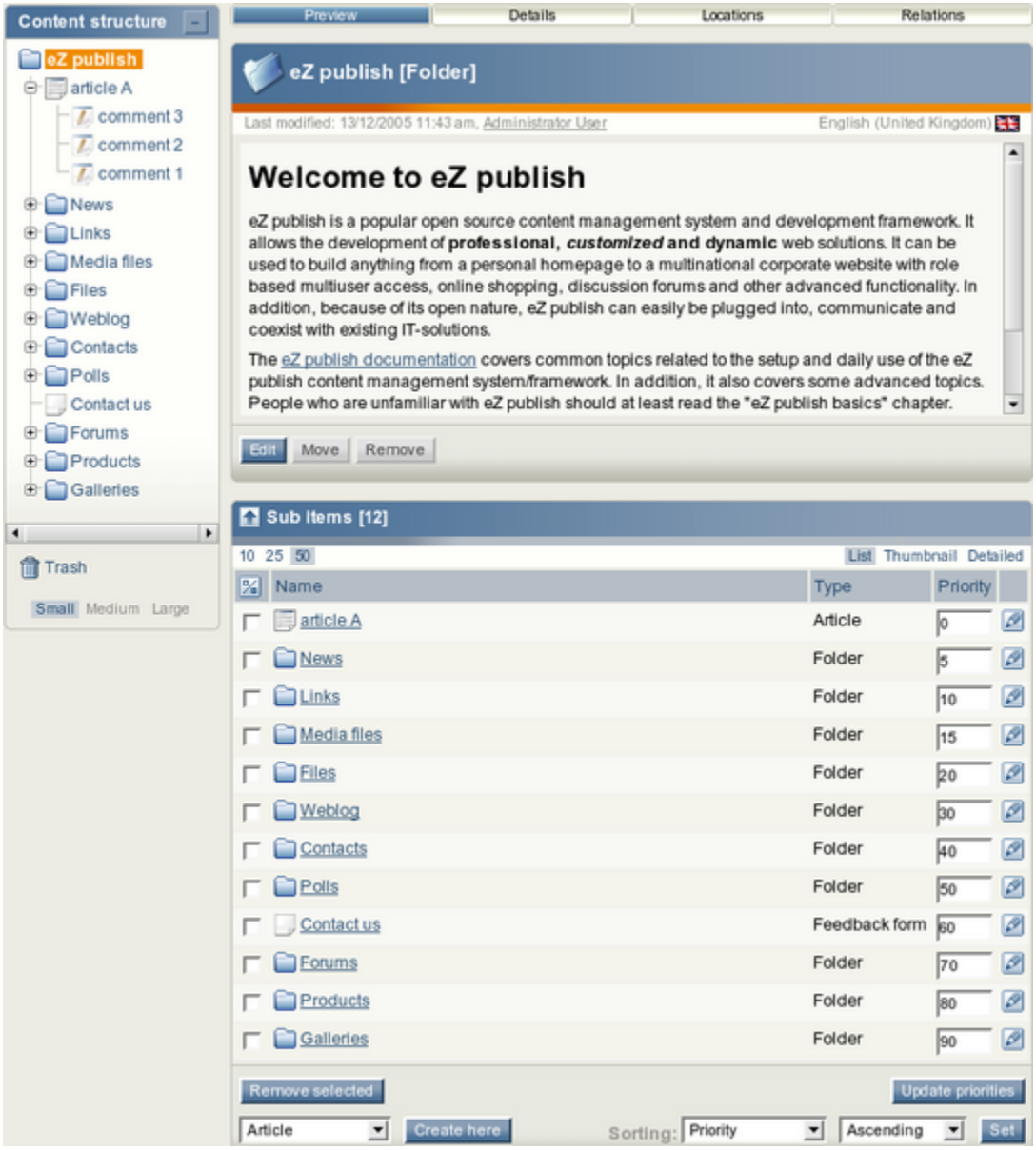
订阅

您可以通过上下文菜单或通知配置界面来简单地订阅某个对象的子树通知。

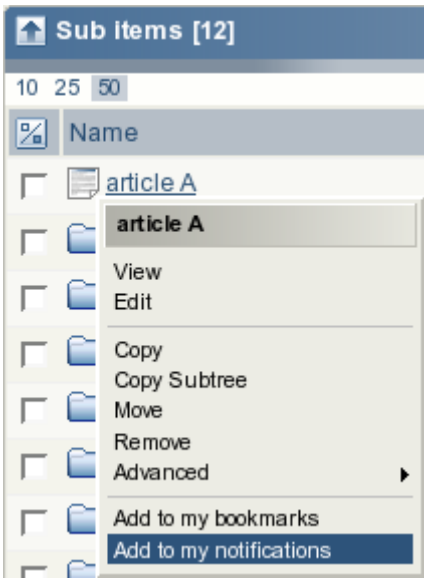
使用上下文菜单

要订阅子树通知，您应该：

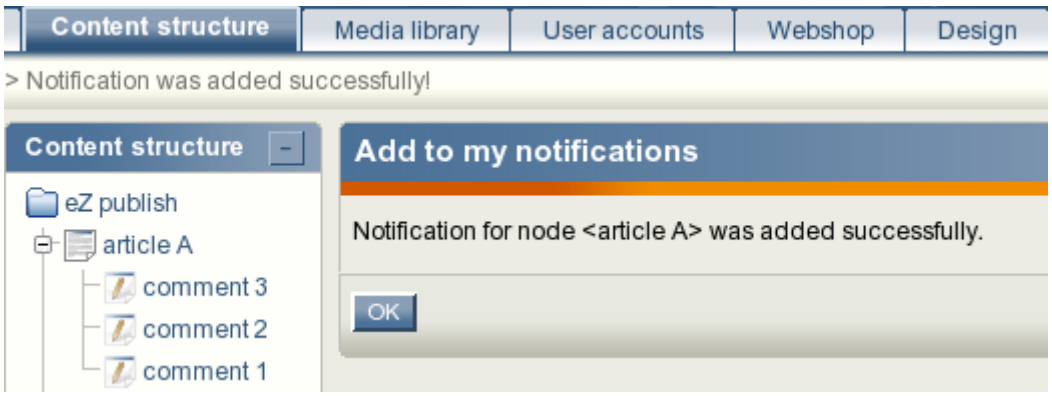
1. 登录管理界面。您应该可以在左侧看到“内容结构”树，根节点默认被选中。下图揭示了系统如何显示选中的节点以及其子节点。



2. 从“内容结构”树或“子项目”中选择需要订阅的节点，点击它的图标并从上下文菜单中选择“添加到我的通知”，如下图所示。



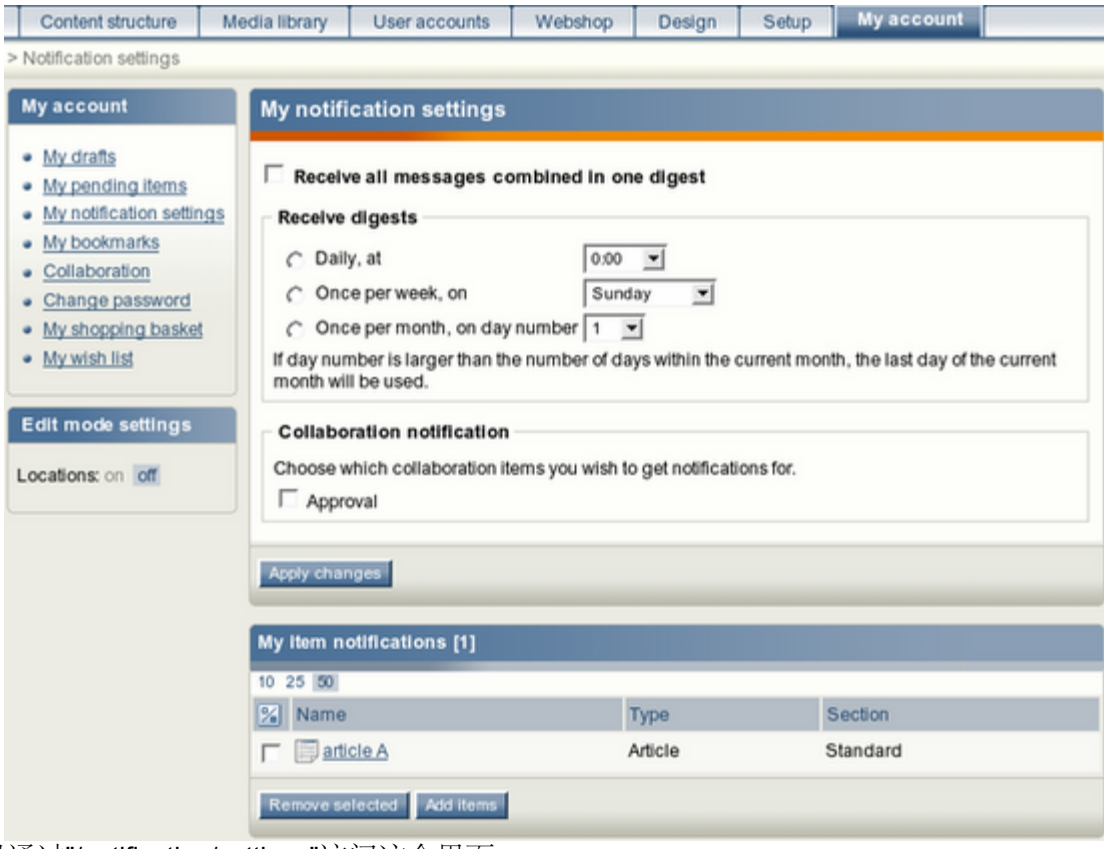
3. 系统会添加一个新的子树通知并显示确认页面：



使用通知配置界面

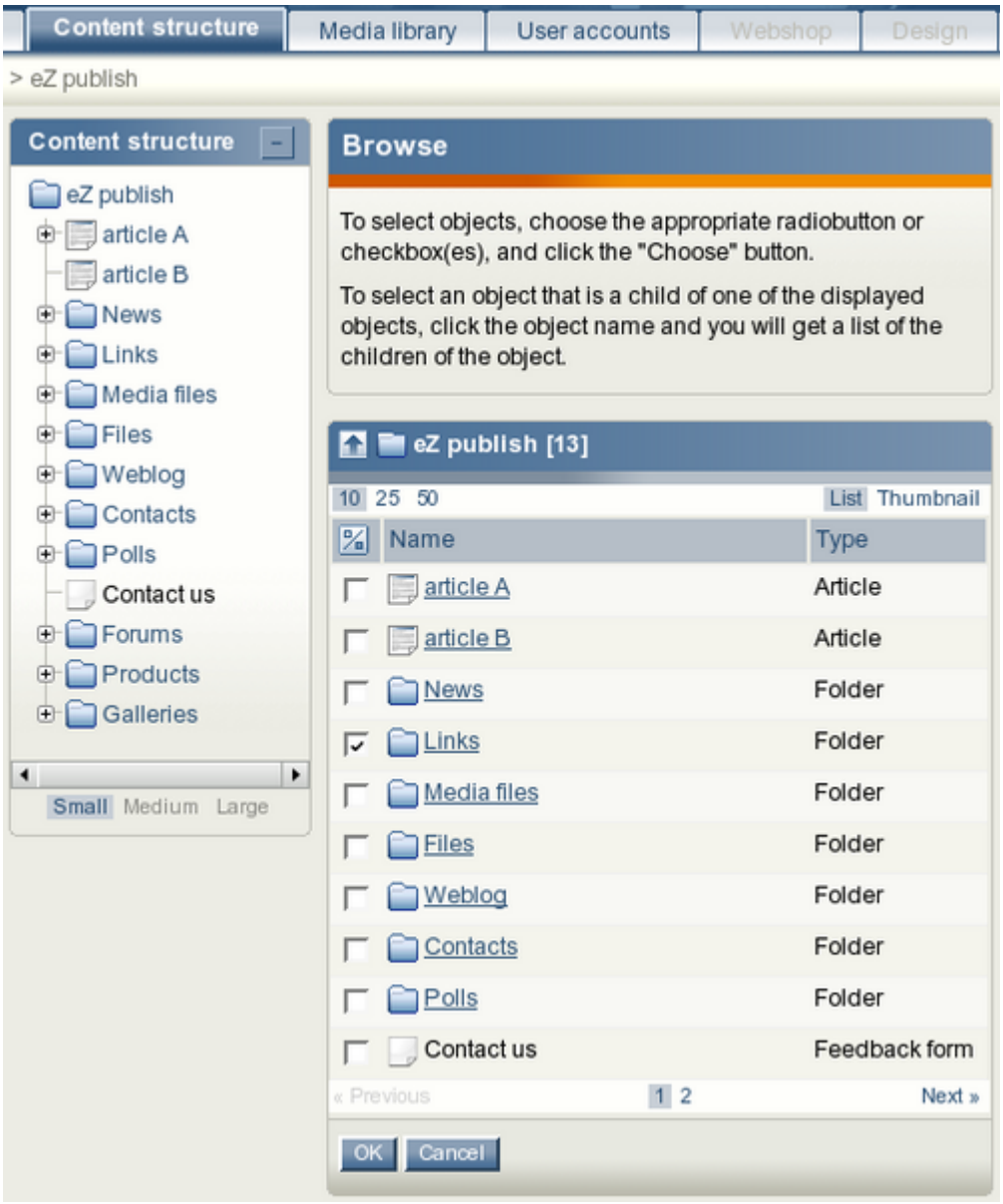
可以通过将一个对象添加到通知配置界面底部的“我的通知项目”列表中来订阅这个对象的子树通知。以下文字揭示了具体步骤：

1. 在管理界面中点击“我的帐号”标签，然后选择左侧的“我的通知配置”连接。系统会显示以下的通知配置界面。



您也可以通过"/notification/settings"访问这个界面。

2. 查看通知配置界面底端的“我的通知项目”列表。这里列出了所有您已经订阅的通知项目。点击“添加项目”按钮来添加新的通知。
3. 系统会显示浏览界面，在这里您可以选择需要的节点：



在列表中选择您希望订阅通知的对象节点。注意，您可以同时选择多个节点。您可以通过点击节点名称在列表中导航。如果需要的节点在“内容结构树”之外，您可以不断点击上箭头图标直到到达树的根节点。这个操作允许您，例如，切换到“用户帐号”树并选择树中的用户组。下图演示了上箭头。

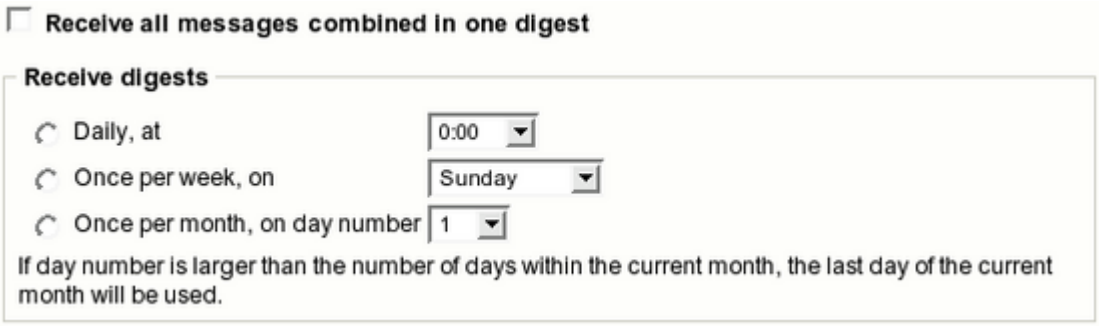
您可以配置如何显示这个列表。例如，您可以通过点击"10", "25", "50“这些链接来设置每页显示的项目数量。如果您希望以缩略图的形式浏览图片对象，您可以点击“缩略图”按钮。

4. 当您完成对象的选择后（通过勾选复选框），点击“确定”按钮。系统会为您订阅这些对象的子树通知并把这些订阅添加到“我的通知项目”列表中。

配置摘要模式

如果您希望以每天/每周/每月摘要的形式收取子树通知，参阅以下步骤启用摘要模式。

1. 在管理界面中通过"/notification/settings"或选择“我的帐号”－“我的通知配置”访问通知配置界面。
2. 摘要配置位于通知配置界面的顶端。默认情况下，摘要模式被禁用（如下图所示）。



要启用摘要模式，勾选“将所有消息合并为一篇摘要”复选框并选择摘要发送的频率。

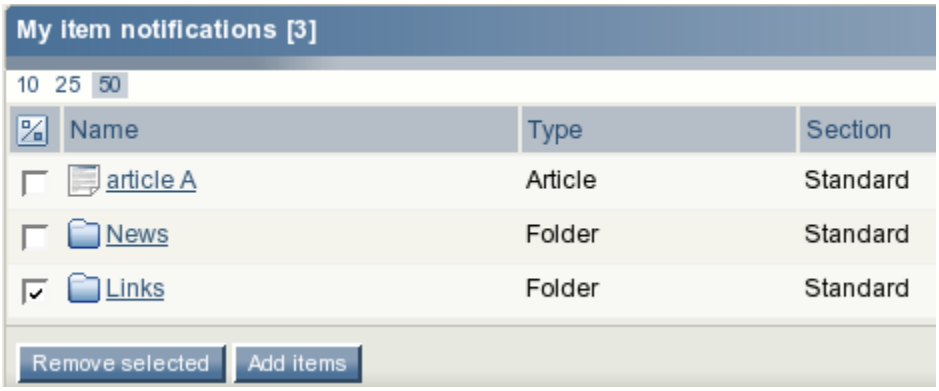
- 每天一次，在特定时间（从 0:00 到 23:00）。
- 每周一次，每周特定日（从星期天到星期六）。
- 每月一次，每月特定日（从 1 号到 3 1 号）。

3. 点击“应用修改”按钮保存您的配置。

退订

如果您不再希望接收关于某个对象的通知，参阅以下步骤退定您的订阅。

1. 在管理界面中通过"/notification/settings"或选择“我的帐号”－“我的通知配置”来访问通知配置界面。
2. 在界面底端的“我的通知项目”列表中，勾选希望退定的项目（如下图）。



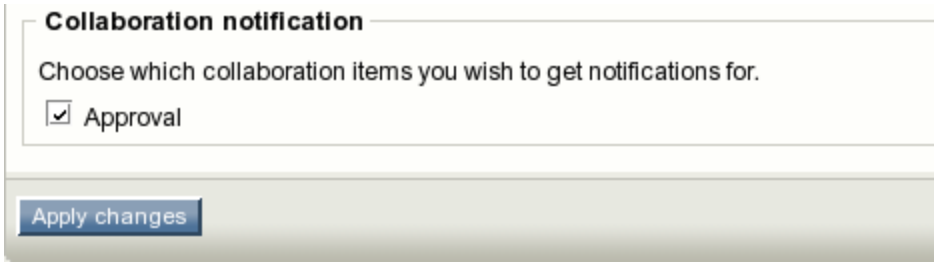
3. 点击“删除所选”按钮。系统会从列表中删除所选项目，因而您不会再收到关于这个/这些对象的通知。

协作通知

如果您通过协作系统与其他人一起工作，您可能希望每次当一个协作消息被创建后，您能收到一封通知邮件。以下步骤揭示了如何配置。

1. 在管理界面中通过"/notification/settings"或选择“我的帐号”－“我的通知配置”来访问通知配置界面。
2. 查看位于摘要配置下的“协作通知”章节。默认情况下，协作通知被禁用。如果您希望接收协作通知，勾选

“批准”复选框（参阅下图）。



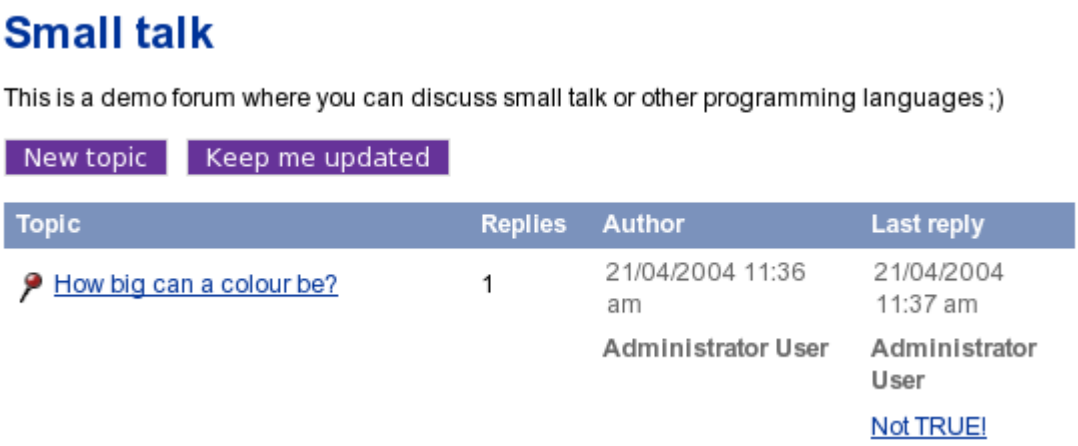
3. 点击“应用修改”按钮保存您的配置。每次当一个协作消息被创建时，系统会给你发送一封通知邮件。

4.13.2 使用真实站点

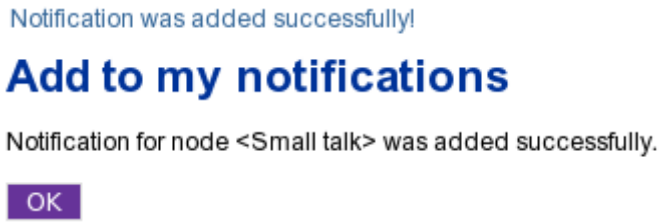
子树通知功能只对那些已通过角色/策略授权访问“通知”模块的用户可用。请参阅“授权访问通知”了解更多关于访问权限的知识。协作通知只在管理界面中可用。

订阅子树通知

用户可以通过点击正在查看的项目的“通知我”按钮来订阅这个对象的子树通知。下图某个站点的论坛。



点击这个按钮后，系统为您添加子树通知并显示如下确认页面：



在标准站点中，“通知我”按钮总是在论坛页面中显示，但是其它页面中没有这个按钮。论坛由以下模板控制：

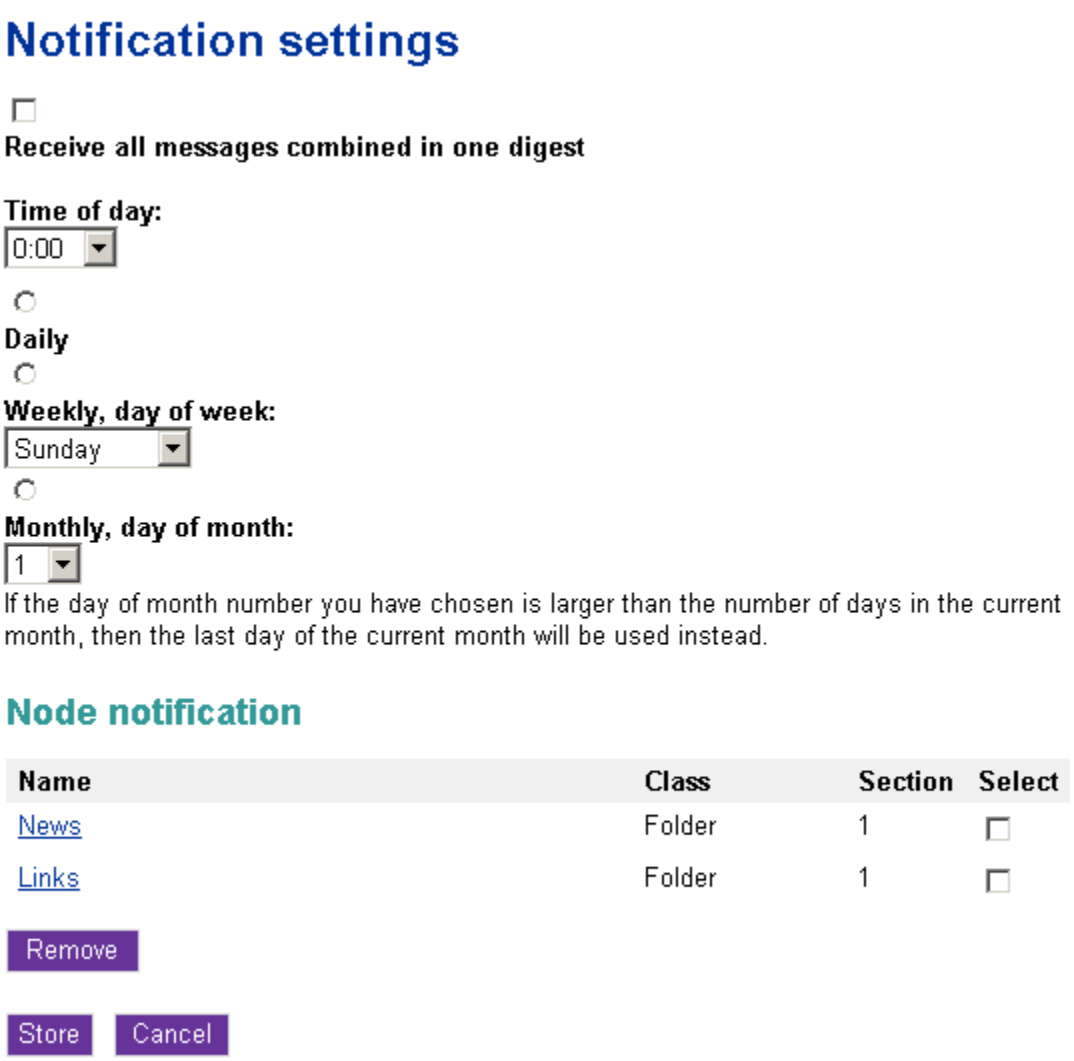
- design/your_siteaccess/override/templates/full/forum.tpl
- design/your_siteaccess/override/templates/full/forum_topic.tpl

请参阅“添加（通知我）按钮”了解更多关于为其它模板添加通知我按钮的信息。

配置摘要模式

无论您使用何种站点入口/界面（只要您有权限），您就可以访问通知配置。您可以用以下方式访问这个配置界面：

1. 登录系统后，访问"/notification/settings"（例如：http://www.example.com/notification/settings）。您应该可以看到摘要配置和“通知”列表（如下图）。



2. 默认情况下，摘要模式被禁用。要启用摘要模式，勾选“将所有消息合并为一篇摘要”复选框，然后选择摘要发送频率。
   - 每天一次，某个时间（从 0:00 到 23:00）
   - 每周一次，每周的某天（从星期天到星期六）
   - 每月一次，每月的某天（从 1 号到 31 号）
3. 点击“保存”按钮保存您的配置。（如果您希望放弃修改，点击“取消”按钮。）

退定

如果您不想再收到关于某个对象的子树订阅，参阅如下步骤：

1. 通过"/notification/settings"（例如：http://www.example.com/notification/settings）访问通知配置界面。

2. 界面中摘要配置下的“通知”列表包含了您的所有订阅（参阅上图）。勾选希望退定的项目。
3. 点击“删除”按钮。系统会从列表中删除所选的订阅。

注意，您可以通过复制 standard 或 admin 界面中的默认模板来重设通知配置模板以配合您的站点需求。

4.13.3 添加“通知我”按钮

用户可以通过使用“通知我”按钮来订阅当前浏览页面的子树通知。大部分默认模板不包含这个按钮。只有论坛中的以下模板包含这个按钮：

- design/your_siteaccess/override/templates/full/forum.tpl
- design/your_siteaccess/override/templates/full/forum_topic.tpl

在前面的例子中，如果您在"Business"文件夹中有一组文章，如果文件夹的模板中没有“通知我”按钮，您的用户将无法订阅这个文件夹的子树通知。请注意用户必须登录才能使用这个特性。

您可以通过在模板中添加以下代码来添加“通知我”按钮。例如，您可以把代码添加到"design/your_siteaccess/override/templates/full/folder.tpl"：

```
<form method="post" action={'/content/action'|ezurl}>
<input type="hidden" name="ContentNodeID" value="{$node.node_id}" />
<input type="submit" name="ActionAddToNotification" value="Keep me updated" />
</form>
```

清除缓存后，每次用户访问文件夹时，“通知我”按钮都会显示。同样的修改也可以应用于您的文章和其它对象的模板。

请注意，某些默认模板可能已经包含一个指向"/content/action"的表单，在这种情况下，确保所有上面代码中的变量在这个表单中。您也可以在一个模板中使用多个提交到"content/action"的表单。

如果您希望在 pagelayout 中使用“通知我”按钮，您需要使用不同的代码。这样做的原因在于，$node 变量在 pagelayout 中不可用。

```
{* Check if we have a node... *}
 {if $module_result.node_id}


 <form method="post" action={'/content/action'|ezurl}>


 <input type="hidden" name="ContentNodeID" value="{$module_result.node_id}" />
 <input type="submit" name="ActionAddToNotification" value="Keep me updated" />
```

```
</form>


{/if}
```

4.13.4 定制电子邮件

可以通过修改模板来定制通知电子邮件。例
如，"design/standard/templates/notification/handler/ezgeneraldigest/view/plain.tpl"是主要的通知模板。它控制通知邮件如何被生成。

如果您需要修改这个模板，您应该把这个模板复制到您的自定义界面并修改它。例如，您可以复制它并添加一些在邮件中显示的附加的/静态的文本。记住在调试修改之前需要清除缓存。注意，您应该复制默认模板到您的界面中，而不能直接修改默认模板。

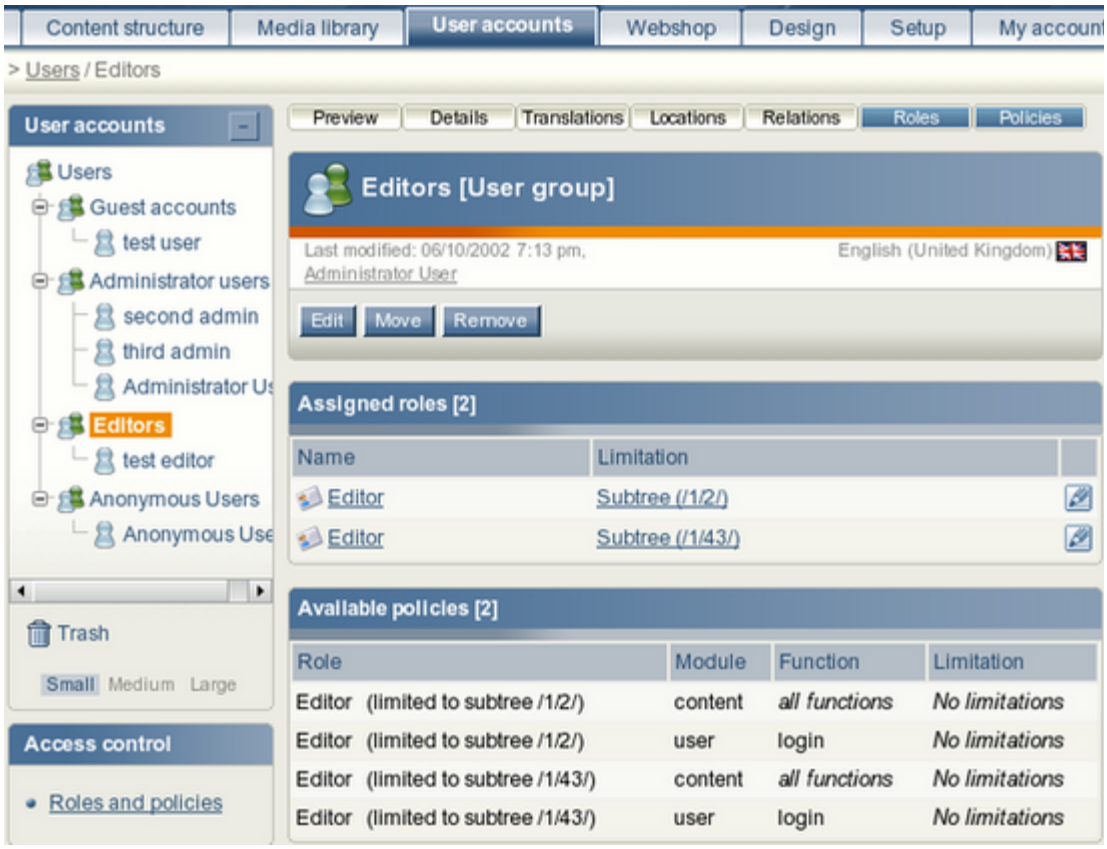
4.13.5 授权访问提醒

内建的权限系统可以控制用户是否允许使用通知。以下文字揭示了如何检查和指派必要的权限。

检查访问权限

以下文字揭示了如何查看一个用户或用户组并检查他们是否允许访问"notification"模块。

1. 登录到管理界面并点击“用户帐号”标签。您可以在页面左侧看到用户和用户组。
2. 在左侧的节点树或子项目窗口中选择要查看的用户或用户组。



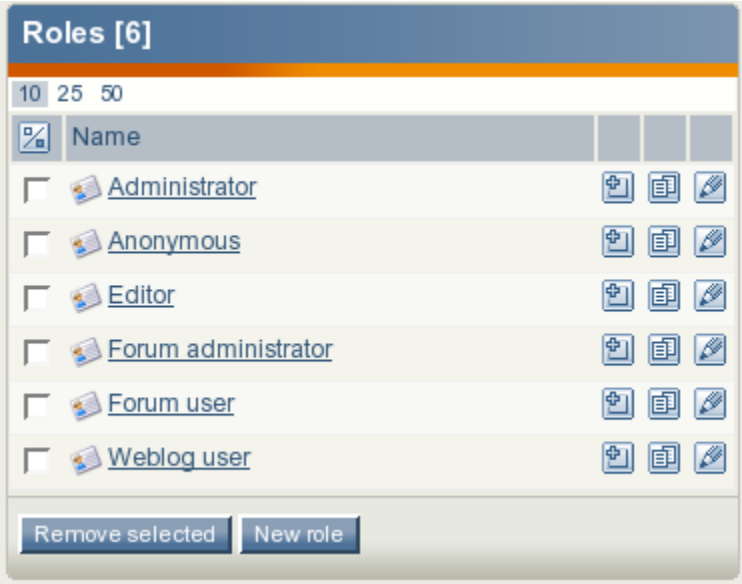
上图中，"Editors"用户组被选中。您可以通过点击页面上面的菜单启用“角色与权限”窗口来显示指派给当前用户组的角色与策略。

3. 查看策略表中的“模块”列。如果"notification"模块没有在此列出，则选中的用户/用户组不允许使用通知。请参阅下一章了解如何创建新的角色（授权访问这个模块）并指派到用户/用户组。

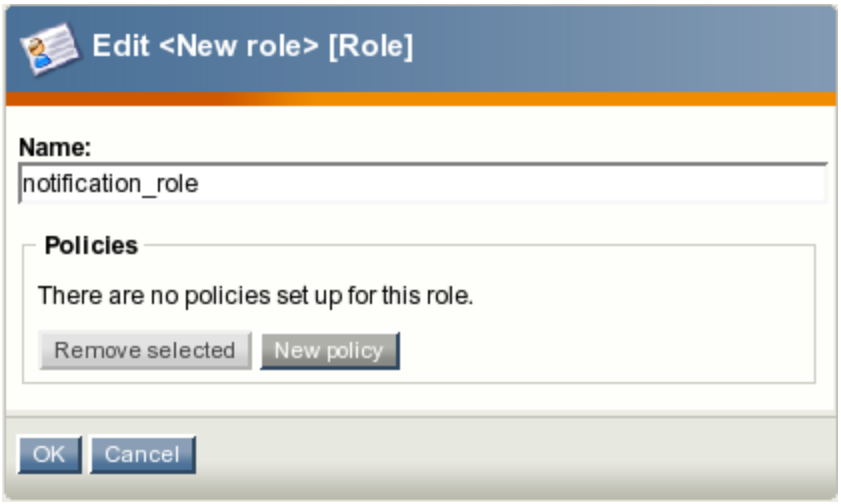
创建一个新角色

以下文字揭示了如何创建一个新的角色并授权访问通知。

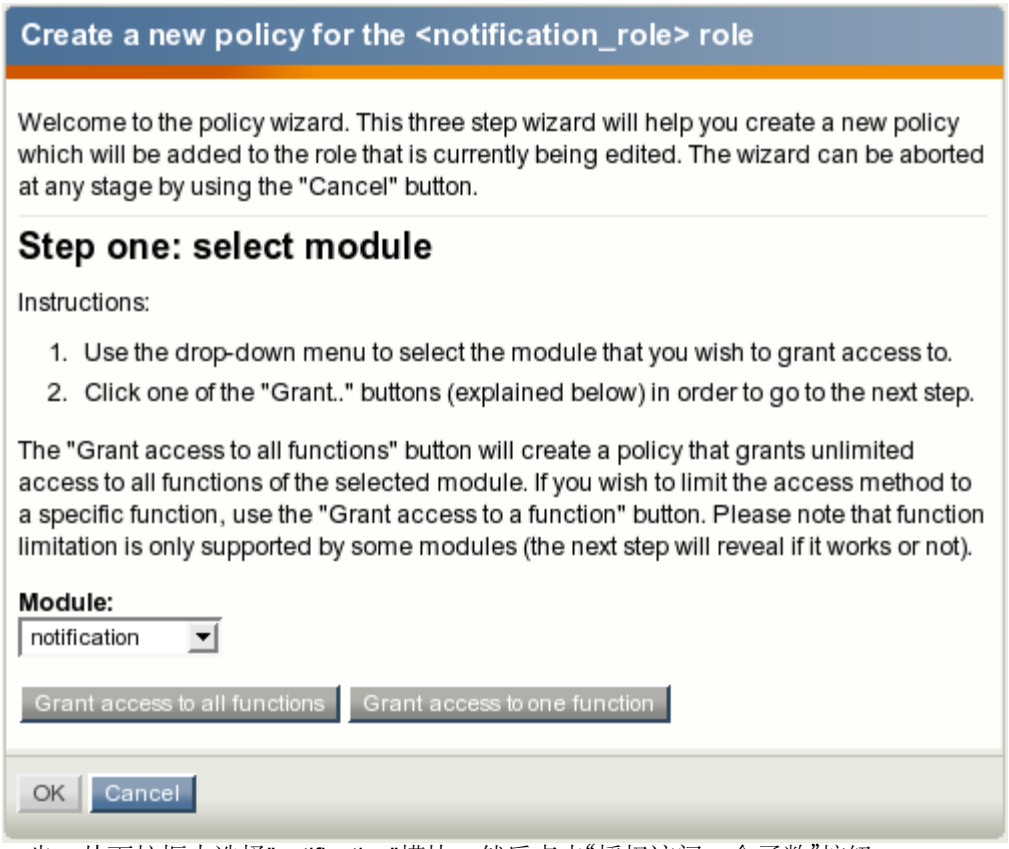
1. 在管理界面中点击“用户帐号”标签，然后访问左侧的“角色与策略”链接。系统会显示现存的角色，如下图所示。



2. 假设要创建一个新的角色“我的通知角色”。点击角色列表下的“新建角色”按钮。系统会显示角色编辑界面，如下图。

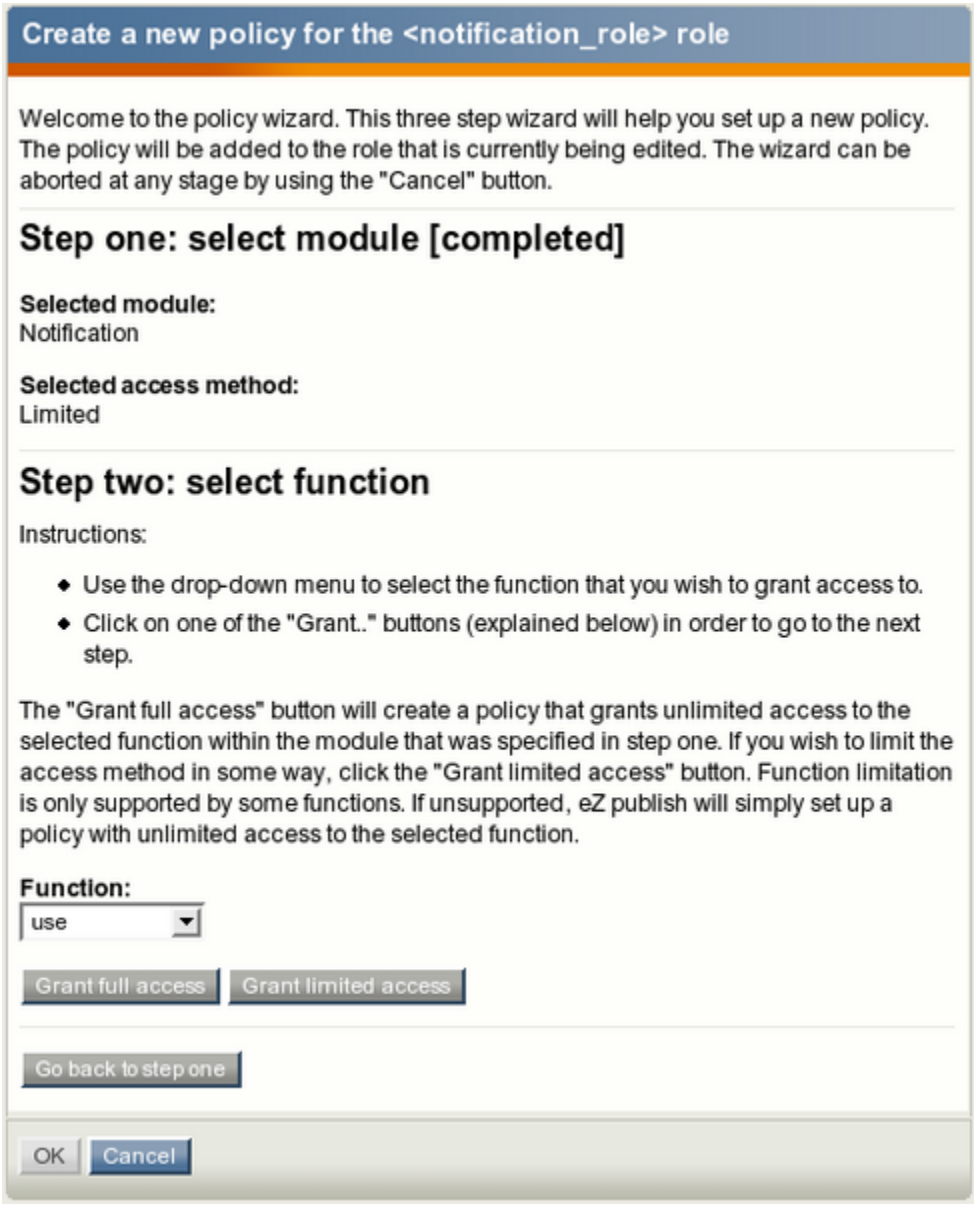


3. 输入角色名称然后点击“新建策略”按钮。
4. 设置向导会引导您用两个步骤创建新的策略。



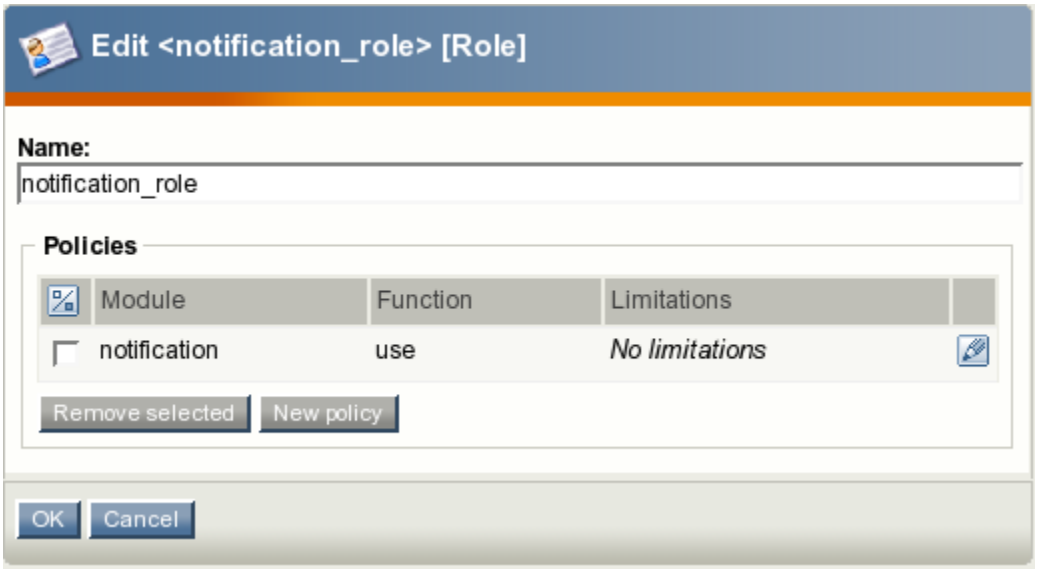
上图演示了第一步。从下拉框中选择"notification"模块，然后点击“授权访问一个函数”按钮。

5. 系统会显示第二步，如下图。

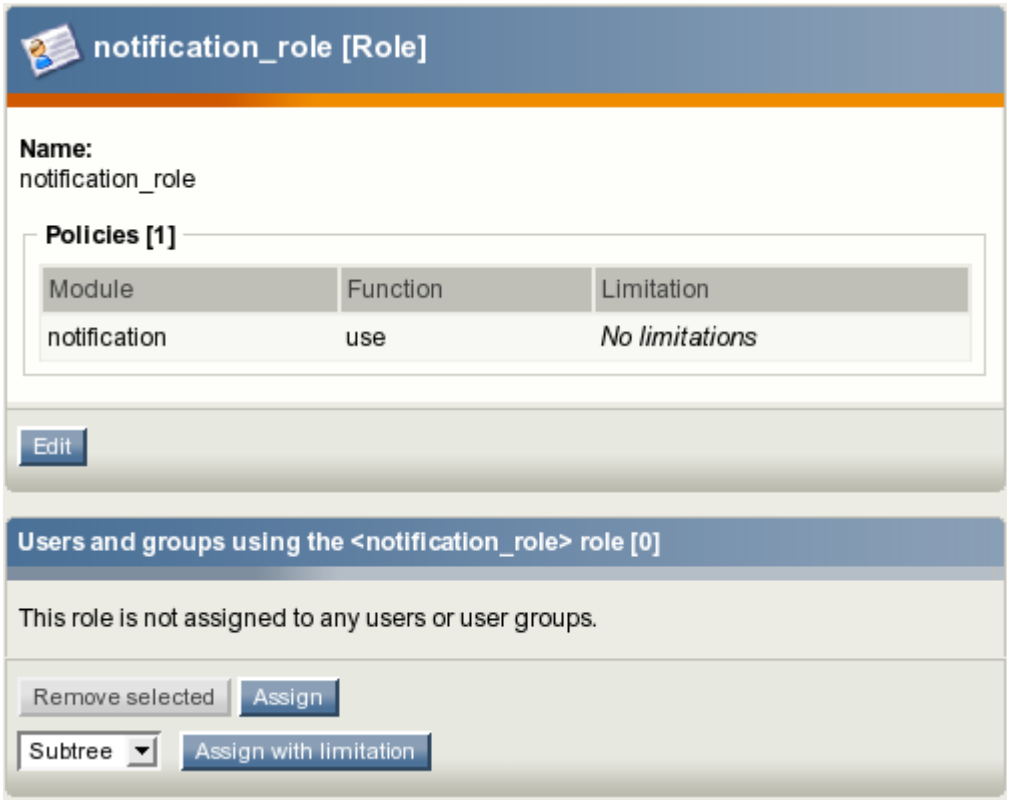


从下拉框中选择"use"函数。注意，您不应该选择"administrate"函数，因为它会授权访问"notification"模块的"runfilter"视图。

6. 点击“授权访问所有”按钮。（此处点击“授权限制访问”没有任何意义，因为"notification"模块不支持授权限制。）
7. 新的策略会在角色编辑界面显示，如下图。



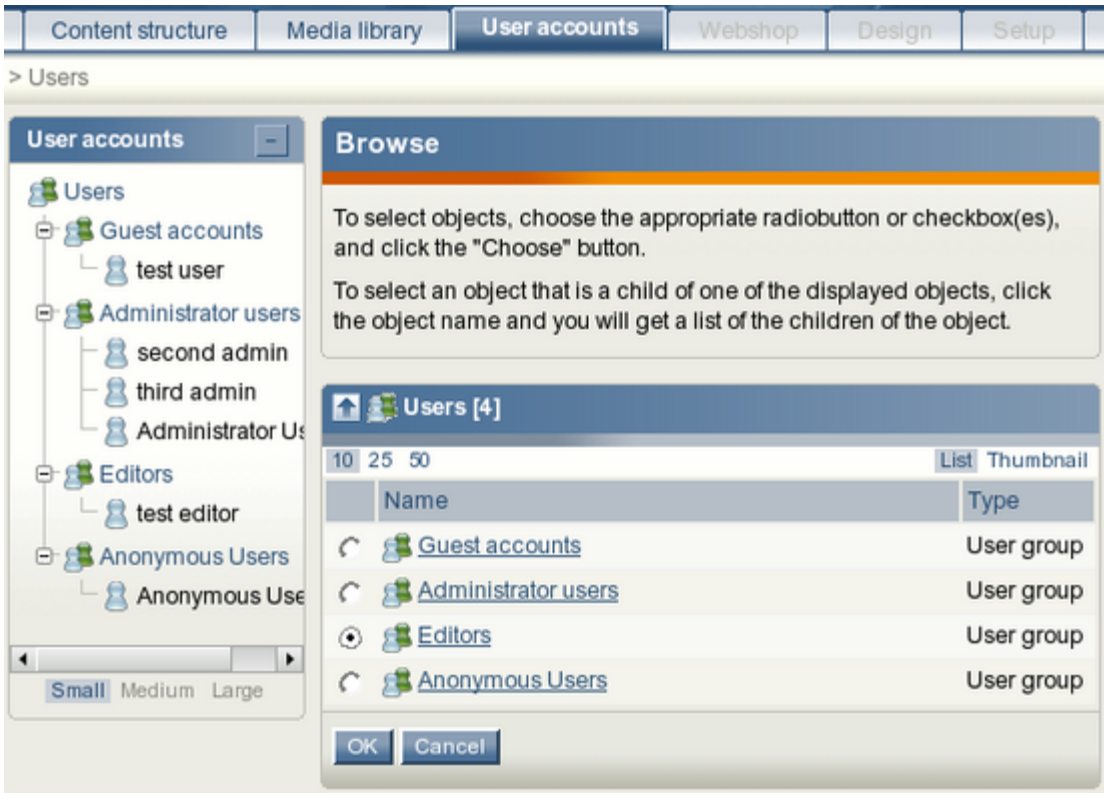
8. 点击“确定”按钮保存您的修改并返回角色查看界面。



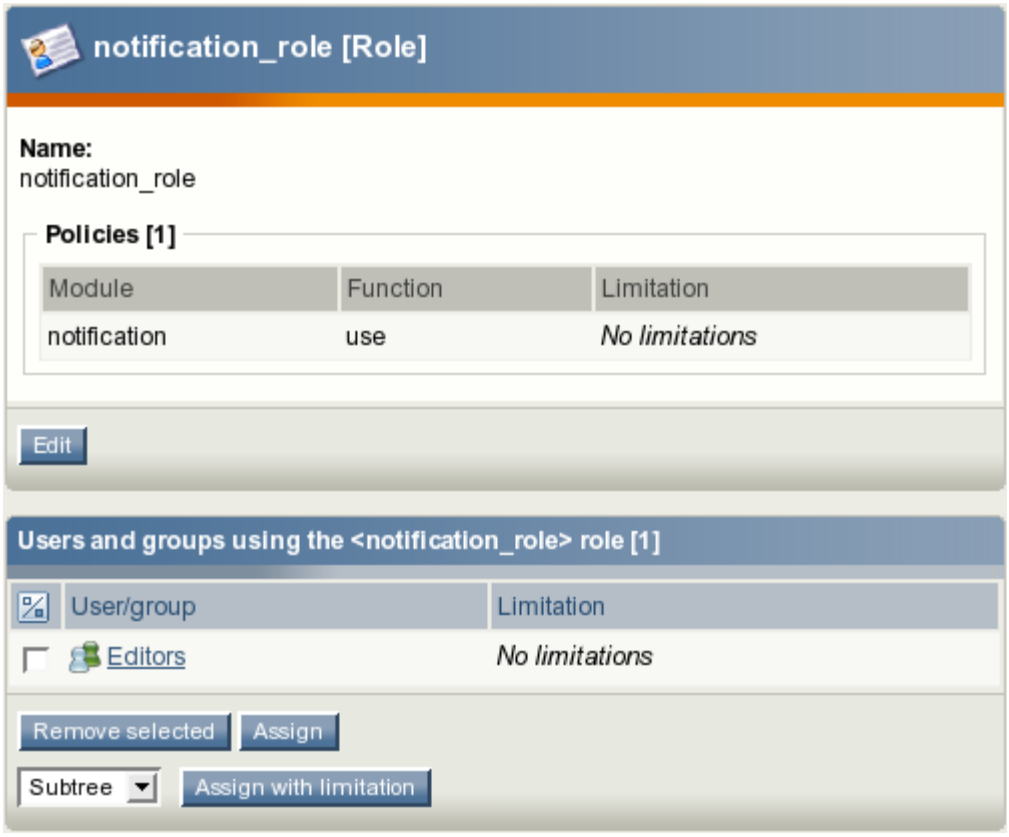
新策略会在角色查看界面显示（如上图）。您现在可以把这个角色指派给任何用户或用户组（这会在下一章节做详细解释）。

当您查看一个角色时，页面底端应该有一个用户／用户组的列表。这个列表揭示了当前角色指派的用户和用户组。以下文字揭示了如何用这个列表把当前角色指派到"Editors"用户组。

1. 点击角色查看试图中用户列表下的“指派”按钮。
2. 选择下图中的"Editors"用户组并点击“确定”按钮。



3. "Editors"用户组会在用户列表中显示。下图显示了“我的通知角色”的角色查看视图，这个角色被指派给了"Editors"用户组（这意味着这个组中的所有用户都可以使用通知）。



注意，您也可以用相同的方法把角色指派给单独的用户。

4.13.6 通知事件

默认情况下，系统支持以下三种通知事件：

- Publish
- Collaboration
- Current time

Publish

每次当对象被发布时，一个新的"ezpublish"事件会被创建。

Collaboration

每次当一个协作（collaboration）消息被生成时，一个新的"ezcollaboration"事件会被创建。

Current time

每次当"runcronjobs.php"脚本被执行，一个新的"ezcurrenttime"事件会被创建。这种行为可以在"cronjobs/notification.php"中指定。系统使用"ezcurrenttime"事件来生成摘要通知。

如果您需要手动生成这个事件，可以在管理界面中访问"notification/runfilter"，然后点击“生成时间事件”按钮。注意，"notification"模块的"runfilter"视图只应用于测试或调试的目的。

内建的通知事件类型保存在"kernel/classes/notification/event/"目录。您也可以为了特定的需求开发自定义的通知事件。

创建和存储

假设您的站点中有一篇文章，并且一个用户已经订阅了这篇文章的子树通知。每次当这篇文章的一篇评论被发布或文章被更新，系统都会生成一个新的"ezpublish"时间并把它存储在数据库中。这个事件可以被 0 个，一个或多个通知处理器处理。

配置

可用的通知事件类型是在"notificatin.ini"重设文件中的"[NotificationEventTypeSettings]"章节中配置的。以下的配置可以用于这个章节：

"RepositoryDirectories[]"数组指定 eZ Publish 会在哪些目录中搜索内建的通知事件类型。事件的确切位置由"AvailableNotificationEventTypes"指定。

"ExtensionDirectories[]"数组指定 eZ Publish 会在哪些扩展目录中搜索附加的通知事件类型。默认情况下，eZ Publish 会在您的扩展中的"notificationtypes"子目录中搜索。事件在这个子目录中的确切位置由"AvailableNotificationEventTypes"指定。

"AvailableNotificationEventTypes[]"数组包含一组事件类型。

例 1

以下内容可以在"notification.ini"重设文件中指定：

```
[NotificationEventTypeSettings]
RepositoryDirectories[]=kernel/classes/notification/event/
ExtensionDirectories[]
AvailableNotificationEventTypes[]=ezpublish
AvailableNotificationEventTypes[]=ezcurrenttime
AvailableNotificationEventTypes[]=ezcollaboration
```

以上配置会要求 eZ Publish 在以下文件中搜索内建的通知事件：

- kernel/classes/notification/event/ezpublish/ezpublishtype.php
- kernel/classes/notification/event/ezcurrenttime/ezcurrenttimetype.php
- kernel/classes/notification/event/ezcollaboration/ezcollaborationtype.php

例 2

您可以通过开发自定义的通知时间来扩展系统。例如，如果您有一个扩展"nExt"，它包含一个通知事件"nev"，把以下内容添加到"notification.ini"重设文件中：

```
[NotificationEventTypeSettings]
ExtensionDirectories[]=nExt
AvailableNotificationEventTypes[]=nev
```

或

```
[NotificationEventTypeSettings]
RepositoryDirectories[]=extension/nExt/notificationtypes/
AvailableNotificationEventTypes[]=nev
```

这些配置会要求 eZ Publish 假定附加的通知事件定义在"extension/nExt/notification/nev/nevtype.php"中。

请注意，您必须至少清除 ini 缓存来要求系统重新读取修改过的配置文件。

4.13.7 通知处理器

默认情况下，系统中有以下几种处理通知事件的处理器：

- 子树通知
- 一般摘要
- 协作通知

子树通知

"ezsubtree"通知处理器处理"ezpublish"事件。

一般摘要

"ezgeneraldigest"通知处理器处理"ezcurrenttime"事件。

协作通知

"ezcollaborationnotification"通知处理器处理"ezcollaboration"事件。内建的通知处理器保存在"kernel/classes/notification/handler"目录中。可以为了特定的需求开发自定义的处理器。

处理通知事件

任何时候，当"eznotificationeventfilter.php"被执行时，系统会尝试用每个可用的通知处理器来运行每个未处理的通知事件。

请注意，处理一个事件可能会发送/生成多个通知。例如，如果一篇文章的新版本被发布，所有订阅的用户都会被通知。

如果每个通知都被成功发送，事件会被系统从数据库中删除。或者，如果某些生成的通知必须被推迟来生成每天/每周/每月的摘要，系统会在用户的通知集中添加新的通知项目。一个通知项目包含通知事件的数据，它的处理器，订阅者的电子邮件和发送通知的事件。

摘要处理器通过访问通知项目集合来处理"ezcurrenttime"事件。在"ezcurrenttime"事件中指定的时间会被用来与每个通知项目的时间比较，从而确定哪些项目必须被处理。只要每个通知项目包含关于通知事件和它的处理器的数据，系统就会用正确的处理器来处理它。结果通知会被合并为摘要信息并发送给订阅者。

如果通知项目被成功处理，系统会将这个项目从项目集中删除。如果没有剩余的通知项目引用当前事件，这个事件会被系统从数据库中删除。

当处理结束后，"ezcurrenttime"事件会被系统从数据库中删除。注意，用摘要处理器处理"ezcurrenttime"事件并不总是以发送/生成摘要通知告终（例如，如果没有任何订阅者启用了摘要模式）。

配置

"notification.ini"重设文件中的"[NotificationHandlerSettings]"定义了用于处理通知事件的处理器。在这个章节下，可以使用以下配置。

"RepositoryDirectories[]"数组告知 eZ Publish 在哪些目录中搜索内建的通知处理器。确切的处理器位置由"AvailableNotificationEventTypes"配置。

"ExtensionDirectories[]"数组告知 eZ Publish 应该在哪些 extension 目录中搜索附加的处理器。默认情况下，eZ Publish 会在您的 extension 中的"notification/handler"子目录中搜索。确切的处理器位置由"AvailableNotificationEventTypes"配置。

"AvailableNotificationEventTypes[]"数组包含了一组处理器。

例 1

以下配置可以在"notification.ini"的重设文件中指定：

```
[NotificationEventHandlerSettings]
RepositoryDirectories[]=kernel/classes/notification/handler/
ExtensionDirectories[]
AvailableNotificationEventTypes[]=ezgeneraldigest
AvailableNotificationEventTypes[]=ezcollaborationnotification
AvailableNotificationEventTypes[]=ezsubtree
```

这些配置会要求 eZ Publish 在以下文件中搜索通知处理器。

- kernel/classes/notification/handler/ezgeneraldigest/ezgeneraldigesthandler.php
- kernel/classes/notification/handler/ezcollaborationnotification/ezcollaborationnotificationhandler.php
- kernel/classes/notification/handler/ezsubtree/ezsubtreehandler.php

例 2

您可以通过创建自定义的通知处理器来扩展系统。例如，如果你有一个扩展"nExt"，它包含一个通知处理器"nh"，把以下内容添加到"notification.ini"重设文件：

```
[NotificationEventHandlerSettings]
ExtensionDirectories[]=nExt
AvailableNotificationEventTypes[]=nh
```

或者

```
[NotificationEventHandlerSettings]
RepositoryDirectories[]=extension/nExt/notification/handler/
AvailableNotificationEventTypes[]=nh
```

这些配置会要求 eZ Publish 在"extension/nExt/notification/handler/nh/nhhandler.php"中搜索通知处理器。

4.13.8 常见问题

问题：是否有一种标准用户会自动收到关于站点的任何变更（创建/修改内容对象）的通知?

回答：默认情况下，没有任何用户会收到站点所有变更的通知。如果您希望任何时候当内容被修改或添加都收到通知，您可以在“内容结构树”中订阅根节点的子树通知。

问题：可能收到新用户注册的通知吗?

回答：您可以在“用户帐号”树中订阅根节点的子树订阅。这样，每次当新用户注册时，您都会收到通知。要订阅新用户注册通知，在管理界面中点击“用户帐号”标签，在树中选择要订阅的节点（新注册的用户对象被保存在这个节点下），然后在上下文菜单中选择“添加到我的通知”。

问题：每次当我需要审批一篇文章时，我是否可以收到电子邮件？文章的作者是否可以被通知是否文章被批准?

回答：当您需要审批一篇文章时，您可以收到通知（文章作者也会收到通知）。这可以通过启用协作通知来达到。目前当文章被批准/拒绝时，文章的作者不会被通知。

问题：我已经订阅了通知但是却收不到任何邮件。

回答：您可能忘记了执行"runcronjobs.php"脚本。如果您希望使用通知系统，这个脚本必须被定期执行。

问题：我同时使用子树和协作通知。子树通知可以正常工作但是我收不到协作通知。

回答：每次当一个协作消息被生成的时候，协作通知会被发送。在管理界面中点击“我的帐号”标签，然后点击左侧的“协作”链接来查看您的协作消息。如果没有任何协作消息，点击“设置”标签，然后选择左侧的“工作流”和/或“触发器”链接来查看您的协作配置。请参阅“工作流”和“审批”文档了解更多关于工作流，触发器和审批事件的知识。（一个简单审批流程包括：创建工作流，把它与触发器函数和审批单元连接。）

问题：我的某个用户无法使用通知配置。

回答：用户首先必须要登录才可能使用通知配置。如果登录后仍然不能使用通知配置，请检查用户的角色/策略配置。参阅“授权访问通知”。

问题：为什么用户点击“通知我”按钮后看到“拒绝访问”页面?

回答：可能他们没有权限使用通知。参阅“授权访问通知”检查用户的角色/策略配置。

问题：在我的站点中有内建/默认的论坛，我希望每个注册的用户都可以订阅/退定关于论坛/主题/回复的子树通知。我应该怎么做?

回答：默认情况下，所有属于"Guest accounts"用户组的用户允许使用通知。这是由指派给"Guest accounts"用户组的默认的"Forum user"角色决定的。可以把这个角色指派给其他用户（参阅“授权访问通知”章节中的“指派角色到用户/用户组”了解更多）。没有必要把这个角色指派给"Administrator users”用户

组。默认的指派给"Administrator users"用户组的"Administrator"角色允许用户访问所有模块，包括"notification"模块。

问题：在角色/策略配置中，当授权访问"notification"模块时，我可以从函数下拉框中选择"administrate"函数。这是否意味着可以在管理界面中的某个位置查看/编辑每个订阅用户的通知配置。

回答：尽管允许管理员查看/编辑所有用户的通知配置可能是个好主意，但是这个功能还没有实现。"use"与"administrate"函数唯一的区别在于后者授权访问"notification"模块的"runfilter"视图。请注意这个视图只应该被用于测试和调试目的。

问题：是否可以对通知强制使用摘要模式，从而对所有订阅用户默认都启用摘要模式（使用预设时间）？

回答：这个功能还未实现。默认情况下，摘要模式被禁用并且数据库中不包含任何关于这个配置的记录。如果用户设置摘要模式，它会被记录在数据库中。

问题：是否可以为子树通知设置过滤器？我在某个文件夹内有一组文章，并且任何时候当在这个位置创建新文章时，用户都会被通知。但是，当现存的文章被修改或新文件夹被创建时，用户也会被通知。我希望指定“只有当文章类的对象被创建时才通知用户”或其他类似的配置。

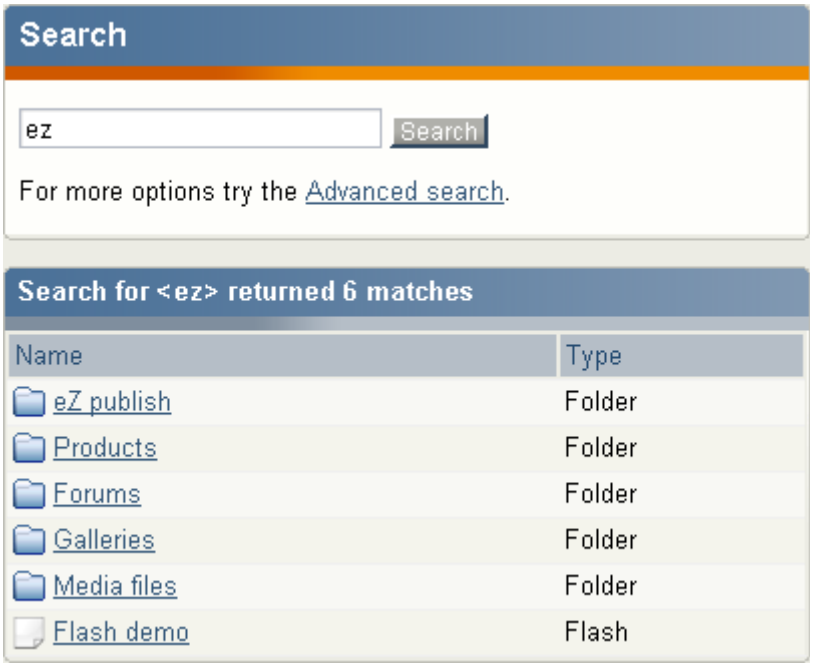
回答：这个功能目前没有实现。

4.14 检索引擎

系统内建了一个与内容结构紧密结合的检索引擎。它可以为任何通过本地的内容模型输入的内容建立索引。

在 eZ Publish 中，内容类描述了真实的数据结构（例如，新闻文章，商品等）。类由属性构成，属性由数据类型构成。一个属性可以是文章的标题，商品的价格等等。可以控制哪些属性应该被编入索引。这可以在编辑类的时候使用“可检索”复选框来完成。某些数据类型（例如：浮点数，价格等等）不支持索引。请参阅“数据类型一览”查看哪些数据类型可以被编入索引。

当一个对象被发布时，被标记为可检索的属性会被检索引擎编入索引。之后就可以在检索界面中检索被发布对象内容中的关键字或短语。例如，如果用户检索"backpack"，系统会返回所有包含"backpack"的所有类型的对象。这是默认的行为。下图演示了标准的检索界面。



高级检索

默认情况下，检索引擎只检索完整的单词或短语。如果用户检索"demo"，系统不返回包含如"demolition"，"demonstration"等内容的对象。但是，eZ Publish 事实上支持通配符检索，但是您必须在"site.ini"重设文件中添加如下内容：

```
[SearchSettings]

EnableWildcard=true
```

当启用通配符时，可以用星号'*'作为通配符，例如："demo*"。在本例中，eZ Publish 会返回一系列包含"demo"的对象。例如，它会返回包含如"demonstration","demolition"等内容的对象。在启用通配符的时候，通配符前面的关键字也会被匹配。换言之，包含"demo"的对象也会被返回。

请注意，通配符只能被追加到某个关键字之后。这意味着以下的检索是不合法的："*demo"，"some*thing"。

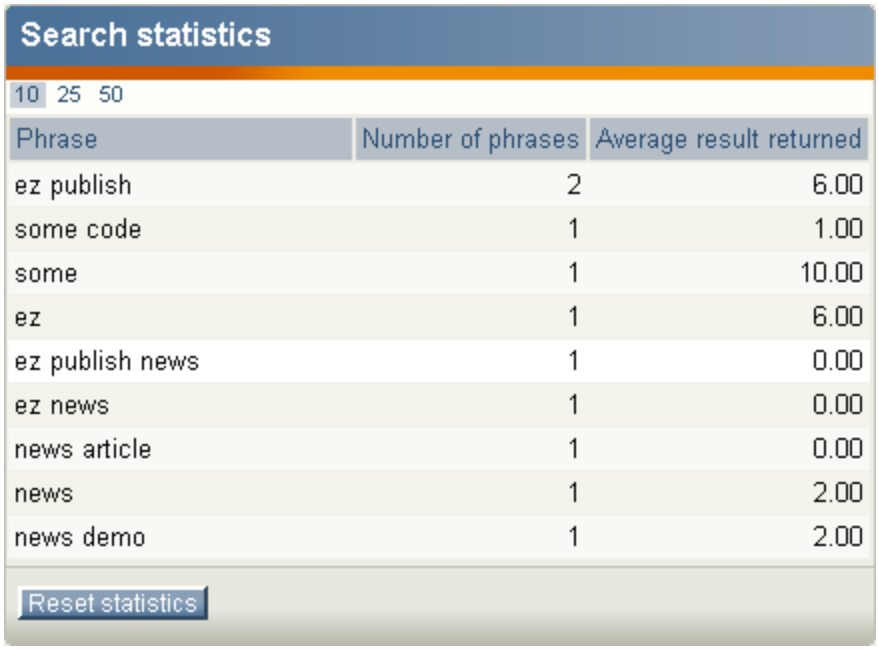
注意！默认情况下，建议禁用通配符检索，因为通配符检索比标准检索需要更多的处理时间。这意味着您必须升级您的服务器来更快地生成结果并保持更少的系统负载。

逻辑操作符

检索引擎不支持“AND”和“OR”操作符。这意味着您不能指定如"cars AND minivans"或"trucks OR vans"的检索条件。但是实际上您可以实现一个“与”检索。这可以通过使用高级检索中的“检索以下所有关键字”输入框来实现。例如，如果用户输入"cars bikes"，那么系统会返回所有同时包含这两个关键字的对象列表，关键字的顺序不重要。

检索统计

管理界面中的配置部分提供了一个页面，这个页面揭示了已经被检索的关键字/短语以及平均检索结果数量。下图为检索统计界面。



Search statistics

10 25 50

Phrase	Number of phrases	Average result returned
ez publish	2	6.00
some code	1	1.00
some	1	10.00
ez	1	6.00
ez publish news	1	0.00
ez news	1	0.00
news article	1	0.00
news	1	2.00
news demo	1	2.00

Reset statistics

“重置统计”按钮会清除检索日志。

4.15 WebDAV

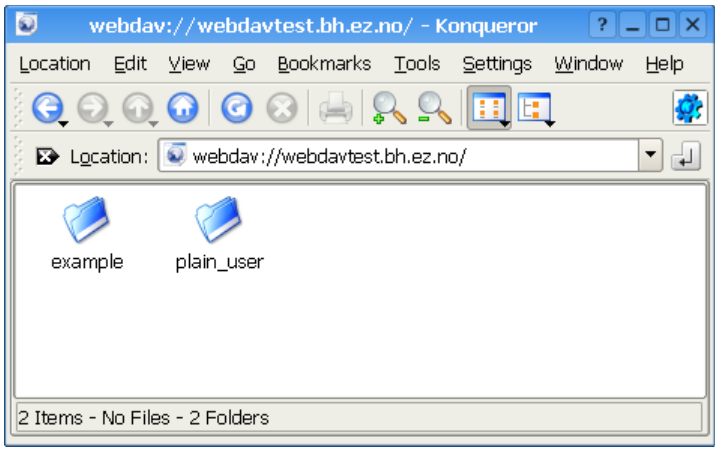
WebDAV 是"Web-based Distributed Authoring and Versioning"（基于 Web 的分布式认证与版本控制）的缩写（在一种基于 RFC2518 的开放标准）。WebDAV 是一组针对 HTTP 协议的扩展，它允许用户协同编辑和管理网站服务器上的文件。这可以通过使用一个兼容 WebDAV 的客户端来完成。例如，可以使用最近的 KDE 版本中的 Konqueror 或微软的 Internet Explorer。通过使用兼容 WebDAV 的客户端，用户可以连接到网站服务器并可以浏览和管理文件，如同使用一个网络文件夹或者 FTP 服务器。换言之，这个协议所允许用户对 Web 服务器上的文件和目录进行浏览，创建，删除，上传，下载，重命名等操作。这种技术最重要的优点之一在于它使用 80 端口作为网络通讯。这意味着如果您可以从您的工作站访问这个网站，您就可以通过 WebDAV 管理它，而并不需要重新配置火墙。

eZ Publish 与 WebDAV

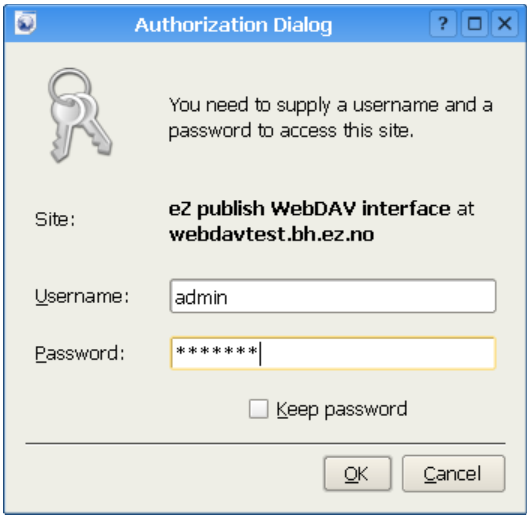
从 3.2 版本开始，eZ Publish 提供了一个内建的 WebDAV 服务器。这个实现允许用户通过兼容 WebDAV 的客户端与 eZ Publish 通讯。一旦建立连接，就可以浏览和管理站点的节点树。节点树会被显示为一个文件系统（由目录和文件构成）。

当用户第一次连接到 eZ Publish 的 WebDAV 服务器，系统会显示一个兼容 WebDAV 的站点入口列表。这个列表可以在"site.ini"重设文件中"[SiteSettings]"章节下的"SiteList[]"中配置。请注意，此时系统还不会要求您输入用户名和密码。换言之，任何可以访问网络的用户都可以看到这些可用的站点入口。下图演示

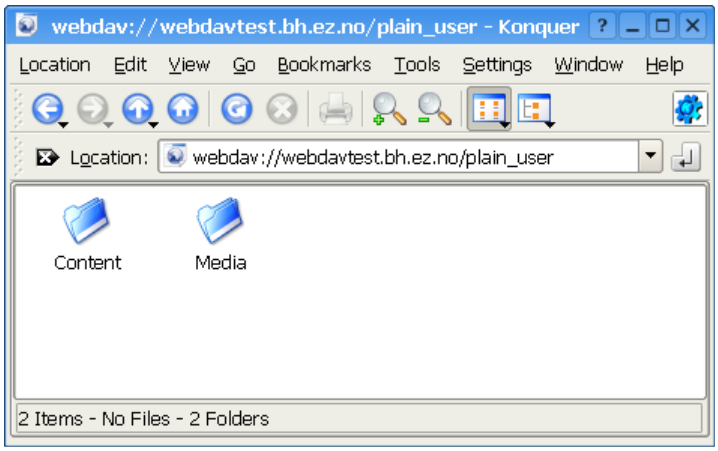
了两个站点入口的 WebDAV 界面，"example"和"plain_user"。



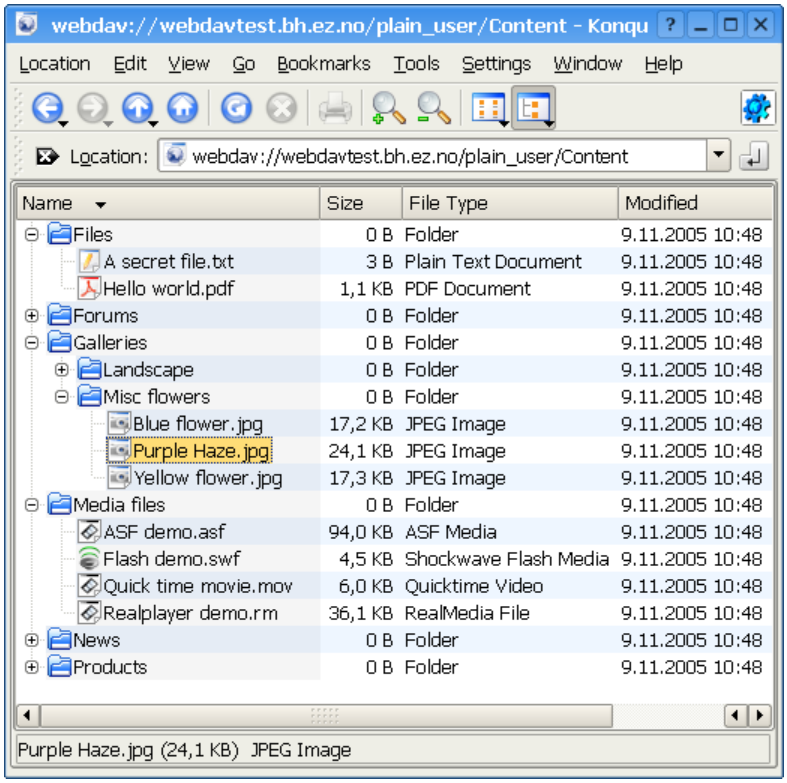
当选中一个站点入口，系统会提示您输入用户名和密码。参阅下图。



提供的用户名和密码必须属于一个存在于选择的站点入口中的有效 eZ Publish 用户。此外，用户必须有足够的权限才可以看到节点树的内容。下图演示了在一个 WebDAV 客户端中显示了名称为"plain_user"的 eZ Publish 站点入口的根节点的内容。



如上图所示，用户可以浏览和管理"Content"和"Media"顶极节点的内容。下图演示了用户在"Content"顶极节点中浏览了若干节点后的页面。



浏览和下载

默认情况下，所有的节点都被显示为目录。这是因为在 eZ Publish 中任何对象都可以以子节点的形式存

在于其它节点下。将节点显示为目录允许用户浏览节点树的结构。但是并不是所有的节点都被显示为目录。

那些包含文件数据类型的对象的节点会被显示为其所在目录中的文件。这意味着使用图片，媒体或文件数据类型的节点会被显示为文件。当下载这些节点时，eZ Publish 会把节点中的文件属性所包含的文件发送给用户。如果节点中有多个文件类型的属性，第一个属性中的文件会被传送给用户。

"webdav.ini"重设文件中的"FolderClasses[]"可以用来配置哪些类型的节点在 WebDAV 客户端中应该被显示为目录。默认的配置确保“文件夹”类型的节点总是被显示为目录。如果把某个包含文件数据类型的类添加到"FolderClases[]"中，会重设上述的行为。换言之，尽管某些节点包含文件数据类型，这个配置还是可以把这些节点显示为目录。

上传

任何类型的文件都可以被上传至 eZ Publish。文件会被保存为文件类的对象。换言之，每次上传一个文件，eZ Publish 会创建一个文件对象，它的文件属性会包含这个上传的文件。此外，在文件上传的位置，一个节点会被创建。这是系统默认的行文。但是，并非所有的文件都会被创建为文件对象。

可以配置系统，从而系统可以根据上传的文件类型创建不同类型的对象。例如，默认的配置确保上传的图片被创建为图片对象。这种行为是由 MIME 类型和类的映射来控制的。映射可以通过"upload.ini"重设文件中"CreateSettings"章节下的"MimeClassMap[]"来配置。"DefaultClass"配置确定如果没有合适的匹配，哪个类应该被用作默认类。这个选项通常被设置为"file"，这意味着系统不识别的类型会被创建为文件对象。下例演示了默认的映射。

```
MimeClassMap[]

MimeClassMap[image]=image

MimeClassMap[video/quicktime]=quicktime

MimeClassMap[video/x-msvideo]=windows_media

MimeClassMap[video/vnd.rn-realvideo]=real_video

MimeClassMap[application/vnd.rn-realmedia]=real_video

MimeClassMap[application/x-shockwave-flash]=flash
```

"MimeClassMap[]"中的每个单元必须通过另外一个配置块来进一步配置。这个配置块揭示了被映射的类的细节。这个配置块必须包含以下信息：

- 目标类的类标识符（后面追加"_ClassSettings"后缀）。
- 用于保存文件的类属性标识符。
- 用于保存文件名的类属性标识符。
- 对象名模式

下例演示了默认的图片类映射配置块。

```
[image_ClassSettings]

FileAttribute=image
```

```
NameAttribute=name

NamePattern=<original_filename_base>
```

上例告知 eZ Publish 当图片被上传，实际的文件数据会被保存在"image"属性中。图片的名称会被保存在"name"属性中（"name"为属性的标识符）。"NamePattern"告知系统应该如何生成上传图片的名称。它可以包含文本和特殊字符（由尖括号环绕"<"和">"）。下表揭示了可以使用的标签。

标签	描述
original_filename	上传文件本地的原始文件名（例如："test.jpg"）
original_filename_base	上传文件原始文件名除去扩展名的部分（例如："test"）。
original_filename_suffix	上传文件的扩展名（例如：".jpg"）。
mime_type	上传文件的 MIME 类型（例如："image/jpeg"）。

自定义上传处理器

可以通过使用自定义的上传处理器来对上传的文件做特殊处理。自定义的上传处理器必须被包含在扩展中。当某种特殊类型的文件被上传后，处理器必须被自动触发。这可以通过"upload.ini"重设文件中"[CreateSettings]"章节下的"MimeUploadHandlerMap[]"来配置。例如，以下的配置会确保所有上传的图片（无论为何种类型的图片）都会被"ezimageuploadhandler.php"中定义的"ezimageuploadhandler"类来处理。

```
MimeUploadHandlerMap[image]=ezimageuploadhandler
```

您也可以配置某些特定类型的文件被这个上传处理器处理。下例演示了如何配置只处理 JPEG 图片。

```
MimeUploadHandlerMap[image/jpeg]=ezimageuploadhandler
```

上传处理器本身必须被放置在扩展中名称为"uploadhandlers"的目录中，如下：

```
eZ Publish
|
 -extensions
  |
   -example
    |
     -uploadhandlers
      |
       -ezimageuploadhandler.php
```

以下代码演示了一个自定义上传处理器的框架。

```
include_once( 'kernel/classes/ezcontentuploadhandler.php' );


class eZExampleUploadHandler extends eZContentUploadHandler
{
```

```
    function eZExampleUploadHandler()
    {
        $this->eZContentUploadHandler( 'Example file handling', 'example' );
    }

    /*!
      Handles the uploading of example files.
    */
    function handleFile( &$upload, &$result,
                         $filePath, $originalFilename, $mimeInfo,
                         $location, $existingNode )
    {
        // Implement your import/conversion routine here
        copy( $filepath, "var/cache/example.jpeg" );
    }
}
```

4.15.1 配置

本章介绍了如何将 eZ Publish 配置为 WebDAV 服务器。请注意 DNS 与 Web 服务器也需要被配置。

第一步：启用 WebDAV 服务器

WebDAV 的主开关必须被打开。为"webdav.ini"创建一个全局的重设文件并确保它包含以下内容：

```
[GeneralSettings]
EnableWebDAV=true
```

第二步：添加需要的站点入口

为了允许 WebDAV 访问特定的站点入口，站点入口的名称必须在"site.ini"重设文件"[SiteSettings]"章节中的"SiteList[]"中配置。确保"site.ini"的全局重设文件包括必要的配置。下例演示了如何为"plain_user"和"example"站点入口启用 WebDAV。

```
[SiteSettings]
SiteList[]
SiteList[]=plain_user
SiteList[]=example
```

第三步：清除所有的缓存

至此，eZ Publish 部分的配置就完成了。现在清除所有的缓存以确保系统使用最新的配置。

第四步：设置 DNS 记录

设置用于访问 WebDAV 的 DNS 记录（例如：子域名）。这些记录必须指向 web 服务器的 IP 地址。例如，如果您通过"www.example.com"访问 web 页面，您应该为 WebDAV 配置"webdav.example.com"。

第五步：配置 web 服务器

在 eZ Publish 根目录中有一个文件"webdav.php"。这个文件提供了真正的 WebDAV 界面。每次当一个 WebDAV 客户端给服务器发送一条命令时，web 服务器必须都自动执行这个文件。以下内容演示了如何在 Apache 配置文件中配置 WebDAV。

```
<Virtualhost 128.39.140.28>
  <Directory /path/to/ezpublish>
    Options FollowSymLinks Indexes ExecCGI
    AllowOverride None
  </Directory>
  DocumentRoot /path/to/ezpublish
  RewriteEngine On
  RewriteRule . /webdav.php
 ServerAdmin admin@example.com
 ServerName webdav.example.com
</VirtualHost>
```

注意：确保您在 virtual hosts 定义之前有一条"NamedVirtualHost"记录。

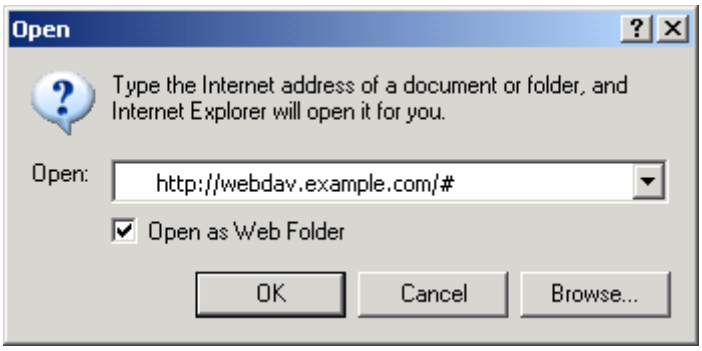
第六步：测试

启动一个兼容 WebDAV 的客户端/应用程序并尝试连接服务器。

Internet Explorer

最近版本的 Microsoft Internet Explorer (6.0.2800.1106 或更新的版本)包含一个内建的 WebDAV 客户端。目标地址必须以 web 文件夹方式打开。

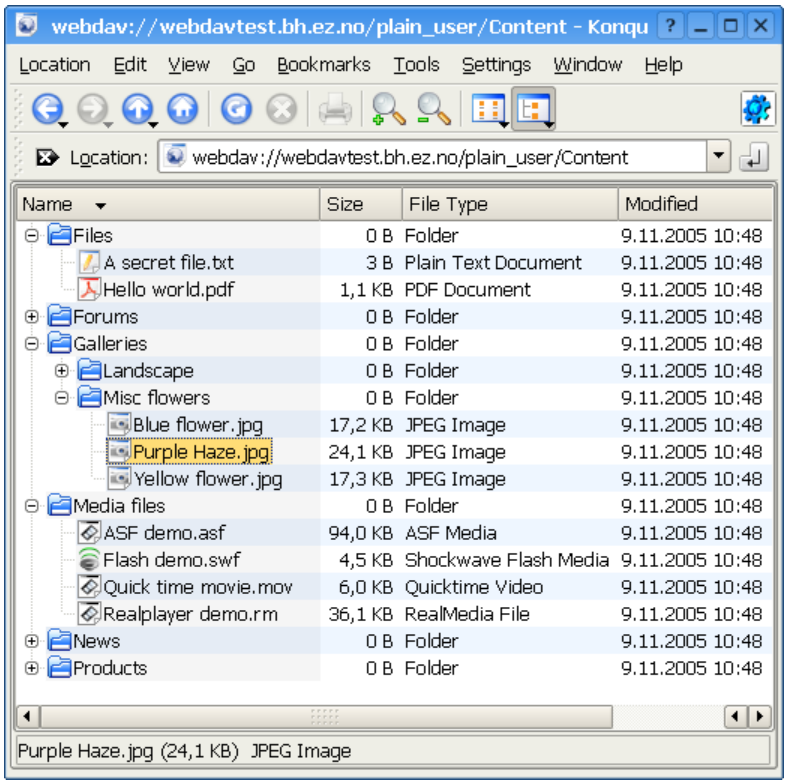
1. 启动 Internet Explorer
2. 访问“文件”菜单，然后选择“打开”，系统会显示一个对话框。
3. 在对话框的地址栏中输入 WebDAV 服务器的地址，在最后追加一个"#"，例如：http://webdav.example.com/#



4. 确保勾选“以 web 文件夹方式打开”复选框。
5. 点击确定。您应该可以可用的站点入口以目录的形式显示出来。

KDE/Konqueror

确保您有最新版本的 Konqueror（3.1.3 或更高版本）。打开 Konqueror 窗口并通过访问 WebDAV 的 URL，例如："http://webdav.example.com/"，来浏览 WebDAV 服务器。



5 参考手册

参考手册的翻译还在进行中，因此目前请先参阅

http://doc.ez.no/eZ-Publish/Technical-manual/4.x/Reference。

6 附录

6.1 词汇表

词汇	解释
Setup wizard	安装向导
Virtual host	虚拟主机
Extension	扩展
Design	界面（直译应为"设计"，但"界面"更符合其技术含义)
Module	模块
View	视图
Workflow	工作流
Override	重设
VAT	增值税
Storage	存储
Datatype	数据类型
Content class	内容类
Class attribute	类属性
Content object	内容对象
Object versonning	对象版本
Content node	内容节点
Content node tree	内容节点树
Top level nodes	顶级节点
Node visibility	节点可见性
Object relation	对象关联
Section	分区
URL storage	URL 存储
Information collection	信息收集
Site	站点
Siteaccess	站点入口

Locale	地区
URL alias	URL 别名
Package	安装包
Assign	指派
Handler	处理器
Event	事件
Approve	审批
Multiplexer	多路控制器
Payment gateway	支付网关
Simple shipping	简单配送
Delayed until date	推迟发布
Kickstart	启动
Library	库
Kernel	内核
Template	模板
Template operator	模板操作符
Instance	实体
Translation	翻译
Node tree	节点树

附录 *1:* 词汇表

6.2 引用

引用	注释
上海众森计算机科技有限公司	www.zerustech.com
eZ Publish 下载	www.zerustech.com/about_us/all/download_ez_publish

附录 *2:* 引用