Forum

Plug-in Guide

trunk-SNAPSHOT

escenic

A CCI COMPANY

Table of Contents

1 Introduction

The Forum plug-in is a framework for adding forum management functionality to the Escenic Content Engine. Forums are a common feature of many web sites, intended to facilitate reader participation.

1.1 What Is a Forum?

Broadly speaking, a **forum** is a web page that contains a series of messages or **postings** submitted by web site visitors. A forum is generally set up to host discussions on a particular subject: politics, a rock band, a computer program, a football team and so on.

The postings in a forum are organized in **threads**, which roughly correspond to conversations. A visitor initiates a thread by posting a message that is not a reply to any other posting (done by clicking on some kind of "post message" link on the forum page). The posting appears on the forum page together with some kind of "reply" link that allows other visitors to reply to the posting. Their postings then belong to this new thread, as do any replies to their postings.

There are many different variations on this basic idea:

- Some forums require visitors to register as a user and log in to be able to submit postings. This gives the forum publisher a minimum of control over the forum (the ability to identify and block troublesome posters, for example).
- Forums that do not require registration usually have some kind of **CAPTCHA** system to prevent the automated posting of spam messages.
- Most forum publishers also reserve the right to remove postings with illegal or offensive content. It is very common for published postings to contain a "complaints" link that allows visitors to submit complaints about such postings, thus increasing the forum publisher's ability to respond quickly.
- In a **moderated forum**, all postings must be read and approved by a **moderator** before they can be published.
- Forum functionality can be used to allow visitors to comment on articles published on a site.

1.2 The Forum Plug-in

The Escenic Forum plug-in enables all of the above types of forums to be created and managed as part of an Escenic publication. In the Forum plug-in, a forum is simply a kind of content item, defined in the `content-type` resource along with other content types. A posting, on the other hand is a special object type, and is usually stored in its own database.

The Forum plug-in supports two different moderation **schemes**:

Pre-moderation
: Postings are not published until they have been approved by a moderator.

Post-moderation
: Postings are published immediately, but may be subsequently withdrawn by a moderator.

Moderation is, however, dependent upon another plug-in, the Dashboard plug-in. If this plug-in is not installed, then no moderation is possible. For further information about this, see section 3.2.

Any posting submitted to a forum that is not a reply to an existing posting initiates a new thread. The thread consists of all the responses to that posting, plus any responses to those responses and so on, recursively. This results in a tree of postings, all related to the original unprovoked posting, which is called the thread's **root** posting.

Any content type in a publication can be made "commentable" by adding a parameter to its definition in the publication's content-type resource. It will then be possible to add a comment submission form to content items of that type, allowing readers to comment on the items. For further information see section 3.2.1.

1.2.1 The Plug-in Components

The Forum plug-in consists of the following components:

- The core code required to provide the basic forum functionality.
- A set of Java Beans (`PresentationForum`, `PresentationThread` and `PresentationPosting`) representing forum objects that can be accessed by JSP template developers. For further information see chapter 12.
- A tag library providing simplified access to these forum objects for template developers. For further information see chapter 9.
- Struts forms and form actions to simplify the process of making posting submission forms. For further information see chapter 11.
- A pingback server. For further information, see chapter 6.

2 Installation

The following preconditions must be met before you can install Forum trunk-SNAPSHOT:

- The Content Engine and Escenic assembly tool have been installed as described in the **Escenic Content Engine Installation Guide** and are in working order.
- You have the required plug-in distribution file **`forum-trunk-SNAPSHOT.zip`**.
- The **`Lucy`** plug-in must be installed to make the users and postings search work in the **`dashboard`** application. Any version of the **`Lucy`** plug-in which is compatible with Escenic Content Engine 5.2.3 can be used.

2.1 Conventions

The instructions in the following section assume that you have a standard Content Engine installation, as described in the **Escenic Content Engine Installation Guide** for releases 5.2.3 and above. *escenic-home* is used to refer to the **`/opt/escenic`** folder under which both the Content Engine itself and all plug-ins are installed).

The Content Engine and the software it depends on may be installed on one or several host machines depending on the type of installation required. In order to unambiguously identify the machines on which various installation actions must be carried out, the **Escenic Content Engine Installation Guide** defines a set of special host names that are used throughout the manual.

Some of these names are also used here:

assembly-host
: The machine used to assemble the various Content Engine components into a enterprise archive (EAR) or **`.ear`** file.

engine-host
: The machine(s) used to host application servers and Content Engine instances.

The following Forum-specific host names are used as well:

forum-database-host
: The machine used to host the Forum database server. For production purposes, you are strongly recommended to create a separate database for your forum data, located on a separate host, rather than using the main Content Engine database to store forum data. This ensures:
  - Better performance
  - The physical separation of user-generated content from editorial content

The host names always appear in a bold typeface. If you are installing everything on one host you can, of course, ignore them: you can just do everything on the same machine. If you are creating a larger multi-host installation, then they should help ensure that you do things in the right places.

2.2 Download Forum

First of all you need to download and unpack the Forum package:

1. Log in as **escenic** on your **assembly-host**.
2. Download the Forum distribution from the Escenic Technet web site (http://technet.escenic.com). If you have a multi-host installation with shared folders as described in the **Escenic Content Engine Installation Guide**, then it is a good idea to download the distribution to your shared **/mnt/download** folder:

   ```
   $ cd /mnt/download
   $ wget http://user:password@technet.escenic.com/downloads/release/54/forum-trunk-SNAPSHOT.zip
   ```

   Otherwise, download it to some temporary location of your choice.
3. If the folder **/opt/escenic/plugins** does not already exist, create it:

   ```
   $ mkdir /opt/escenic/plugins
   ```
4. Unpack the Forum distribution file:

   ```
   $ cd /opt/escenic/plugins
   $ unzip /mnt/download/forum-trunk-SNAPSHOT.zip
   ```

   This will result in the creation of an **/opt/escenic/plugins/forum** folder.

2.3 Install Database

> For test and development purposes you can skip the steps in this section: all forum-related data will then be stored in your main Content Engine database. For a production system, you are **strongly recommended** to create a separate database for your forum data, located on a separate host. This section describes how to do that.

The following instructions describe how to install and set up MySQL for use by the Forum plugin.

1. Log in as **root** on your **forum-database-host**.
2. Install the MySQL server and client packages:

   ```
   # apt-get install mysql-server mysql-client
   ```

   During the installation you will be asked to specify a root password for the database.
3. Log in to the system and create a database:

   ```
   # mysql -p
   mysql> create database db-name character set utf8 collate utf8_general_ci;
   Query OK, 1 row affected (0.00 sec)

   mysql> grant all on db-name.* to user@'%' identified by 'password';
   Query OK, 0 rows affected (0.00 sec)

   mysql> grant all on db-name.* to user@'localhost' identified by 'password';
   Query OK, 0 rows affected (0.00 sec)
   mysql> exit
   ```

Replace *db-name*, *user* and *password* in the above commands with a database name, user name and password of your choice. *db-name* must, of course be different from the name of your main Content Engine database.

4. Still on your **forum-database-host**, change user to **`escenic`**.
5. You now need to run the Forum database scripts. If you have a single-host installation, then you will find them in **`/opt/escenic/plugins/forum/misc/database/mysql`**. If you have a multi-host installation with shared folders, then you have direct access to the distribution file and can just unpack it to a temporary location as follows:

   ```
   $ cd /tmp
   $ unzip /mnt/download/forum-trunk-SNAPSHOT.zip
   ```

   You will then find the scripts in **`/tmp/forum/misc/database/mysql`**. If you don't have a shared folder set-up then you can either download the package from Escenic Technet again and unpack it, or copy the scripts over from your **assembly-host**.

   Once you have located the scripts, you can run them as follows:

   ```
   $ cd script-folder
   $ for el in tables.sql constraints.sql indexes.sql; do
   > mysql -u user -ppassword db-name < $el
   > done;
   ```

   where *script-folder* is the path of the folder containing the scripts.
6. On some platforms, external access to the MySQL server is disabled by default. To enable external access, you need to bind the **`mysql`** process to the **forum-database-host**'s IP address. To do this, you need to open **`/etc/mysql/my.cnf`** for editing (as **`root`** again) and set the **`bind-address`** parameter:

   ```
   bind-address = forum-database-host-ip-address
   ```
7. Verify that the MySQL server is running and accessible by trying to connect to port 3306 from each of your other hosts using **`telnet`**:

   ```
   $ telnet forum-database-host 3306
   ```

   where *forum-database-host* is the host name or IP address of the **forum-database-host**. If a connection is opened, then the database server is running and accessible.

2.4 Assemble and Deploy Forum

Assemble and deploy the Forum plug-in as follows:

1. Log in as **`escenic`** on your **assembly-host**.
2. Run the **`ece`** script to re-assemble your Content Engine applications

   ```
   $ ece assemble
   ```

   This generates an EAR file (**`/var/cache/escenic/engine.ear`**) that you can deploy on all your **engine-host**s.
3. If you have a single-host installation, then skip this step.

On each **engine-host**, copy **`/var/cache/escenic/engine.ear`** from the **assembly-host**. If you have installed an SSH server on the **`assembly-host`** and SSH clients on your **`engine-host`**s, then you can do this as follows:

```
$ scp -r escenic@assembly-host-ip-address:/var/cache/escenic/engine.ear /var/
cache/escenic/
```

where *assembly-host-ip-address* is the host name or IP address of your **assembly-host**.

4. On each **engine-host**, deploy the EAR file and restart the Content Engine by entering:

```
$ ece deploy
$ ece restart
```

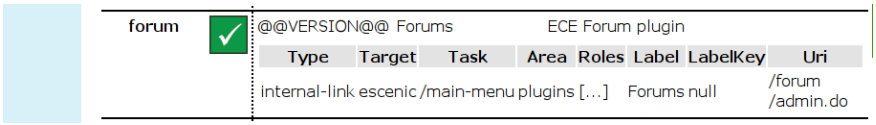
2.5 Verify The Installation

To verify the status of the Forum plug-in, open the Escenic Admin web application (usually located at **`http://`***server*`/escenic-admin`**) and click on **View installed plugins**. The status of all currently installed plug-ins is shown here, and indicated as follows:

The plug-in is correctly installed.

The plug-in is not correctly installed.

So if the Forum plug-in is correctly installed, you should see something like this in the displayed plug-in list:



2.6 Install the Moderator Webapp

The **`dashboard`** web application can be used to moderate forums. **`Dashboard`** is provided as separate plug-in since this component also may be used to manage other components of an Escenic Content Engine based system. Please refer to the **Dashboard Plug-in Guide** for detailed information on how to install and use it.

2.7 Configure the Forum Plug-in

The Forum plug-in does not **require** any configuration in order to work. You can, however, control some aspects of the plug-in's behavior by modifying configuration parameters in one or more of your **configuration layers**, as described below. For background information about configuration layers and how they work, refer to the **Escenic Content Engine Server Administration Guide**.

In these instructions, the placeholder *configuration-root* is used to represent the root folder of the configuration layer in which you are working. In the case of the **common** configuration layer, this will be **`/etc/escenic/engine/common`** if you have a standard Content Engine installation as described in the **Escenic Content Engine Installation Guide**. If you have a single-host installation or multi-host installation with configuration layers on a shared (i.e NFS) file system, then you will only need to carry out the changes described below once. Otherwise you will have to repeat them on each of your **engine host**s.

The general procedure for configuring the Forum plug-in is:

1. If the *configuration-root*`/com/escenic/forum` folder does not exist, create it. In the rest of these instructions, the placeholder *forum-config* is used to represent this folder.
2. If the *forum-config* folder does not contain the **`.properties`** file(s) you require (see below for details), copy it/them from **`/opt/escenic/engine/plugins/forum/misc/siteconfig/com/escenic/forum`**.
3. Edit the **`.properties`** files to meet your requirements.

2.7.1 Set Up Database Pooling

In order to set up database pooling for the forum database, you need to:

1. Copy **`ForumDBManager.properties`** component configuration to your *forum-config*`/persistent` folder. To copy it to your common configuration layer, for example:

   ```
   $ mkdir -p /etc/escenic/engine/common/com/escenic/forum/persistent/
   $ cp /opt/escenic/plugins/forum/misc/siteconfig/com/escenic/forum/persistent/
   ForumDBManager.properties \
     /etc/escenic/engine/common/com/escenic/forum/persistent/
   ```

2. Open *configuration-root*`/com/escenic/forum/persistent/ForumDBManager.properties` for editing and modify the **`contentManager`** property setting to use the Forum-specific content manager:

   ```
   contentManager=/com/escenic/forum/persistent/ForumContentManager
   ```

3. **`ForumContentManager`** uses a JNDI datasource for storing and retrieving data, called **`jdbc/FORUM_DS`** by default. If for any reason you want to use different one, you can do so by editing the following file:

   - *configuration-root*`/com/escenic/forum/persistent/DataConnector`

4. Open your application server configuration file (**`/opt/tomcat/conf/server.xml`**) for editing and insert a resource definition for the data source name. The resource definition must be inserted as children of the **`GlobalNamingResources`** element. If you have left the default name unchanged you will need to enter:

   ```
   <Resource
     name="FORUM_DS"
     auth="Container"
     type="javax.sql.DataSource"
     username="user"
     password="password"
     driverClassName="com.mysql.jdbc.Driver"
     maxActive="30"
     maxIdle="10"
     maxWait="5000"
     url="jdbc:mysql://database-host/db-name?
   autoReconnect=true&amp;useUnicode=true&amp;characterEncoding=UTF-8"
   ```

```
/>
```

Replace *database-host*, *db-name*, *user* and *password* in the above definitions with the details of your Forum database.

5. Finally, open **/opt/tomcat/conf/context.xml**) for editing and insert a resource link for the data source name. The resource link must be inserted as children of the root **Context** element. For example:

```
<ResourceLink
    global="FORUM_DS"
    name="jdbc/FORUM_DS"
    type="javax.sql.DataSource"/>
```

2.7.2 Caches

The Forum plug-in maintains a number of caches for various purposes. In a production environment, it may sometimes be necessary to change the sizes of one or more of these caches. In order to change a cache size, copy one of the following **.properties** files from **/opt/escenic/engine/plugins/forum/misc/siteconfig/com/escenic/forum/presentation** to your *forum-config*/**presentation** folder, and set the **maxSize** property to the required size.

ForumCache.properties
PostingCache.properties
SubmittedFormsCache.properties
ThreadCache.properties

It is unlikely that you will need to modify any of the other properties in any of these files.

2.7.3 Solr Configuration

To enable flagging and moderation of those flagged content, Forum needs some fields to be indexed into Solr. To do this, the user needs to add the following XML fragment into the Escenic Content Engine's Solr configuration file which is usually (if you have the followed the Escenic Content Engine Installation Guide) : **/var/lib/escenic/solr/<ece instance >/conf/schema.xml**

```
<field name="qualification-isflagged_b" type="boolean" indexed="true" stored="false" /
>
```

```
<field name="type_text" type="string" indexed="true" stored="true" />
```

In order to filter posting search in Dashboard for individual forums, the following configuration is also required in Solr:

```
<field name="forumid_double" type="double" indexed="true" stored="true"/>
```

3 Adding a Forum To a Publication

To add a forum to a publication, you need to:

- Add Struts configuration information to the publication's **WEB-INF** folder.
- Create the necessary forum content types.
- Create templates (or modify existing templates) to:
  - Display forum postings.
  - Display posting submission forms.
  - Display user registration and log-in forms (if required)
  - Limit posting submission to logged in users (if required)

3.1 Struts Configuration

The Forum plug-in uses the Apache Struts framework to manage the posting forms displayed in publications. You therefore need to add some configuration information to the publication's **WEB-INF/web.xml** file, and also add a Struts configuration file called **struts-config.xml** to the **WEB-INF** folder. These steps are described in the following sections.

For a proper introduction to Struts, see http://struts.apache.org/primer.html.

3.1.1 Editing web.xml

Every Escenic publication has a **web.xml** file in its **WEB-INF** folder. To enable Struts, **web.xml** must contain code like this:

```
...
<listener>
  <description>Escenic Forum Presentation layer bootstrap listener</description>
  <listener-class>com.escenic.forum.presentation.servlet.PresentationBootstrapper</
listener-class>
</listener>
...
<!-- Standard Action Servlet Configuration -->
<servlet>
  <servlet-name>action</servlet-name>
  <servlet-class>
    org.apache.struts.action.ActionServlet
  </servlet-class>
  <init-param>
    <param-name>config</param-name>
    <param-value>/WEB-INF/struts-config.xml, /WEB-INF/struts-config-forum.xml, /WEB-
INF/struts-config-profile.xml</param-value>
  </init-param>
  <init-param>
    <param-name>validate</param-name>
    <param-value>true</param-value>
  </init-param>
  <init-param>
    <param-name>locale</param-name>
```

```
    <param-value>true</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>

<!-- Standard Action Servlet Mapping -->
<servlet-mapping>
  <servlet-name>action</servlet-name>
  <url-pattern>*.do</url-pattern>
</servlet-mapping>
```

The most important items in the above code are highlighted:

com.escenic.forum.presentation.servlet.PresentationBootstrapper
: This listener definition must be added somewhere in the file (preferably alongside any other listener definitions).

action
: The name of the Struts action servlet. It is only used inside the **web.xml** file, so you can use any name you like. It must, however, be the same in both places it appears.

org.apache.struts.action.ActionServlet
: The name of the class that is to be used as the Struts action servlet. You should not in general change this line.

/WEB-INF/struts-config.xml, /WEB-INF/struts-config-forum.xml, /WEB-INF/struts-config-profile.xml
: The paths of the Struts configuration files to be used, described below. You will need at least to specify **struts-config,xml** and **struts-config-forum.xml**. **struts-config-profile.xml** is only needed if your publication is to require forum user registration and login. Note also that Escenic's **Viz Community Expansion** provides an alternative and more flexible means of providing user registration functionality - it allows users to log in using credentials they already have, such as Google or Facebook credentials.

Several Escenic plug-ins use Struts, so the publication **web.xml** file may already contain some of the above code, in which case you do not need to add it again.

3.1.2 Editing struts-config.xml

If your publication does not already have a **struts-config.xml** file in the **WEB-INF** folder, then create one and add the following content to it:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
  <global-forwards>
    <forward name="not-logged-in" path="/"/>
  </global-forwards>
  <controller forwardPattern="$P" pagePattern="$P" debug="2" locale="true"

 processorClass="com.escenic.forum.struts.presentation.Utf8RequestProcessor"/>
  <message-resources parameter="com.escenic.forum.Messages"/>
</struts-config>
```

If the **WEB-INF** folder **does** already contain a **struts-config.xml** file, then add the highlighted section of the above content to it.

3.1.3 Editing struts-config-forum.xml

Create **struts-config-forum.xml** file in the **WEB-INF** folder of your publication and add the following content to it:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
  <form-beans>
    <form-bean name="com.escenic.forum.struts.presentation.CommentForm"
               type="com.escenic.forum.struts.presentation.CommentForm"/>
    <form-bean name="com.escenic.forum.struts.presentation.PostingForm"
               type="com.escenic.forum.struts.presentation.PostingForm"/>
    <form-bean name="com.escenic.forum.struts.presentation.ComplaintForm"
               type="com.escenic.forum.struts.presentation.ComplaintForm" />
  </form-beans>

  <action-mappings>
    <action path="/insert/comment"
            type="com.escenic.forum.struts.presentation.InsertCommentAction"
            name="com.escenic.forum.struts.presentation.CommentForm"
            input="/template/art/comments/add-comment.jsp"
            validate="false"
            scope="request">
      <forward name="target" path="/template/art/comments/add-comment-success.jsp"/>
    </action>
    <action path="/insert/posting"
            type="com.escenic.forum.struts.presentation.InsertPostingAction"
            name="com.escenic.forum.struts.presentation.PostingForm"
            input="/"
            validate="true"
            scope="request">
      <forward name="target" path="/"/>
    </action>
    <action path="/insert/auth/posting"

 type="com.escenic.forum.struts.presentation.AuthenticatedInsertPostingAction"
            name="com.escenic.forum.struts.presentation.PostingForm"
            input="/"
            validate="true"
            scope="request">
      <forward name="target" path="/"/>
    </action>
    <action path="/insert/complaint"
            type="com.escenic.forum.struts.presentation.InsertPostingAction"
            name="com.escenic.forum.struts.presentation.ComplaintForm"
            input="/"
            validate="true"
            scope="request">
      <forward name="target" path="/"/>
    </action>
    <action path="/insert/auth/complaint"

 type="com.escenic.forum.struts.presentation.AuthenticatedInsertPostingAction"
```

```
            name="com.escenic.forum.struts.presentation.ComplaintForm"
            input="/"
            validate="true"
            scope="request">
      <forward name="target" path="/"/>
    </action>
  </action-mappings>
</struts-config>
```

3.1.4 Editing struts-config-profile.xml

This files is only needed for publications that require forum user registration and login.

> Escenic's **Viz Community Expansion** provides an alternative and more flexible means of providing user registration functionality. It allows users to log in using credentials they already have, such as Google or Facebook credentials.

If you want to use the Forum plug-in's built-in user registration support, create a **`struts-config-profile.xml`** file in the **`WEB-INF`** folder of your publication and add the following content to it:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.1//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_1.dtd">
<struts-config>
  <form-beans>
    <form-bean name="com.escenic.profile.presentation.struts.LoginForm"
               type="com.escenic.profile.presentation.struts.LoginForm"/>
    <form-bean name="com.escenic.profile.presentation.struts.UserForm"
               type="com.escenic.profile.presentation.struts.DefaultUserForm"/>
    <form-bean name="com.escenic.profile.presentation.struts.DeleteProfileForm"
               type="com.escenic.profile.presentation.struts.DeleteProfileForm"/>
    <form-bean name="com.escenic.profile.presentation.struts.PasswordForm"

 type="com.escenic.profile.presentation.struts.ValidateUserPasswordForm"/>
    <form-bean name="com.escenic.profile.presentation.struts.ChangePasswordForm"
               type="com.escenic.profile.presentation.struts.ChangePasswordForm"/>
  </form-beans>
  <action-mappings>
    <action path="/login"
            type="com.escenic.profile.presentation.struts.LoginAction"
            name="com.escenic.profile.presentation.struts.LoginForm"
            scope="request"
            validate="true">
      <forward name="success" path="/" redirect="true"/>
    </action>
    <action path="/logout"
            type="com.escenic.profile.presentation.struts.LogoffAction"
            name="com.escenic.profile.presentation.struts.LoginForm"
            scope="request"
            validate="false">
      <forward name="success" path="/" redirect="true"/>
    </action>
    <action path="/user/add"
            type="com.escenic.profile.presentation.struts.AddUserAction"
            name="com.escenic.profile.presentation.struts.UserForm"
            input="/?service=register"
            scope="request">
```

```
      <forward name="success" path="/template/profile/register-success.jsp"/>
      <forward name="failure" path="/?service=register" />
      <forward name="error" path="/?service=register" />
    </action>
    <action path="/user/newPassword"
            type="com.escenic.profile.presentation.struts.NewPasswordAction"
            name="com.escenic.profile.presentation.struts.PasswordForm"
            input="/?service=forgot"
            scope="request">
      <forward name="success" path="/" redirect="true"/>
      <forward name="error" path="/?service=forgot" />
    </action>
    <action path="/user/setPassword"
            type="com.escenic.profile.presentation.struts.SetPasswordAction"
            name="com.escenic.profile.presentation.struts.ChangePasswordForm"
            scope="request">
      <forward name="success" path="/" redirect="true"/>
    </action>
    <action    path="/user/populate"
               type="com.escenic.profile.presentation.struts.PopulateUserAction"
               name="com.escenic.profile.presentation.struts.UserForm"
               scope="request"
               validate="false">
      <forward name="success" path="/template/profile/update_profile.jsp"
 redirect="false"/>
      <forward name="failure" path="/" redirect="true" />
    </action>
    <action    path="/user/update"
               type="com.escenic.profile.presentation.struts.UpdateUserAction"
               name="com.escenic.profile.presentation.struts.UserForm"
               scope="request"
               input="/template/profile/update_profile.jsp"
               validate="true">
      <forward name="success" path="/" redirect="true" />
      <forward name="failure" path="/" redirect="true" />
    </action>
    <action    path="/user/delete"
               type="com.escenic.profile.presentation.struts.DeleteUserAction"
               scope="request">
      <forward name="success" path="/" redirect="true" />
    </action>
    <action    path="/profile/delete"
               type="com.escenic.profile.presentation.struts.DeleteProfileAction"
               name="com.escenic.profile.presentation.struts.DeleteProfileForm"
               validate="true"
               scope="request">
      <forward name="success" path="/" redirect="true" />
    </action>
  </action-mappings>
</struts-config>
```

3.2 Creating a Forum Content Type

A forum is displayed in a content item with some special characteristics. You therefore need to add at least one forum content type definition to your publication's **`content-type`** resource (located in the publication's **`META-INF/escenic/publication-resources`** folder).

A forum content type must have the following minimum characteristics:

- A parameter called **`com.escenic.forum.articleType`**, which must be set to the value **`forum`**.
- A title field (that is, a field that is designated as the title field using the **`ui:title-field`** element).
- An enumeration field called **`moderation-scheme`**, with two enumeration values:

  `post-moderation`
  This means that postings will be posted before they have been moderated, and may subsequently be withdrawn by the moderator.

  `pre-moderation`
  This means that postings will not be posted until they have been approved by the moderator.

  Moderation functionality is dependent on the Dashboard plug-in. If the Dashboard plug-in is not installed, therefore, the contents of this field will be ignored. You should, nevertheless, always include a moderation-scheme field in your Forum content types.

The following example shows a **`content-type`** resource containing such a content type:

```
<content-types
    xmlns="http://xmlns.escenic.com/2008/content-type"
    xmlns:ui="http://xmlns.escenic.com/2008/interface-hints"
    version="4">
  <content-type name="forum">
    <parameter name="com.escenic.forum.articleType" value="forum"/>
    <ui:title-field>title</ui:title-field>
    <panel name="Standard">
      <field name="title" type="basic" mime-type="text/plain">
      </field>
      <field name="moderation-scheme" type="enumeration">
        <enumeration value="post-moderation"/>
        <enumeration value="pre-moderation"/>
      </field>
    </panel>
  </content-type>
</content-types>
```

In most cases you should also add a boolean field (called **`active`**, for example) that you can use to control whether or not the Forum is open for posting. For example:

```
<content-types
    xmlns="http://xmlns.escenic.com/2008/content-type"
    xmlns:ui="http://xmlns.escenic.com/2008/interface-hints"
    version="4"
  <content-type name="forum">
    <parameter name="com.escenic.forum.articleType" value="forum"/>
    <ui:title-field>title</ui:title-field>
    <panel name="Standard">
      <field name="title" type="basic" mime-type="text/plain">
      </field>
      <field name="moderation-scheme" type="enumeration">
        <enumeration value="post-moderation"/>
        <enumeration value="pre-moderation"/>
      </field>
      <field name="active" type="boolean">
    </panel>
  </content-type>
</content-types>
```

The content type may in addition contain any other fields you want (a **body** field, for example).

Once you have added such a content type to your publication's **content-type** resource, a Content Studio user with sufficient privileges will be able to start a forum by simply creating and publishing a new content item of the type **forum**.

Note that the above examples are deliberately kept as short as possible: labels are omitted, for example.

3.2.1 Commentable Content Types

If you want readers to be able to comment on content items, then you need to modify the content types on which they are based. To do this, add a **decorator** called **com.escenic.forum.presentation.ForumPresentationArticleDecorator** to the **content-type**. This decorator will give the template developer access to related postings for this article. The following example shows a **news** content type to which such a decorator has been added.

```
<content-type name="news">
  <ui:title-field>title</ui:title-field>
  <ui:decorator
 class="com.escenic.forum.presentation.ForumPresentationArticleDecorator"/>
  <panel name="article">
    <ref-field-group name="common"/>
  </panel>
  <panel name="front-page">
    <ref-field-group name="frontpagefields"/>
  </panel>
</content-type>
```

3.3 Creating Forum Templates

The following examples are based on the templates in the **forum-demo** publication. They show how to use the Forum plug-in to add forum functionality to a publication, concentrating on the use of the Forum plug-in components. User access control is achieved using standard components delivered with the Content Engine: this is therefore not described in detail here.

You will find all the examples given below in the **forum-demo** publication's **index.jsp** file. The examples require the following tag library inclusions:

```
<%@ taglib uri="http://struts.apache.org/tags-html" prefix="html" %>
<%@ taglib uri="http://www.escenic.com/taglib/escenic-forum" prefix="forum" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

3.3.1 Listing Forums

To list all the forums in a publication:

```
<h2>Choose a Forum</h2>
<forum:forums id="forums" recursive="true" sectionIds="${publication.rootSection.id}"/
>
<c:if test="${!empty forums }">
  <ul>
    <c:forEach var="forum" items="${forums}">
      <li>
```

```
        <a href="${pageContext.request.requestURL}?forumId=${forum.id}">
          <c:out value="${forum.fields.title}"/> (forum ID ${forum.id})
        </a>
      </li>
    </c:forEach>
  </ul>
</c:if>
```

The **forum:forums** tag (see section 9.3) is used to create a collection of **PresentationForum** beans (see section 12.1) representing all the forums in the publication: a link is then output for each bean in the collection.

3.3.2 Displaying a Forum

To lists all the threads in a forum:

```
<c:if test="${!empty param.forumId}">
  <forum:forum id="forum" forumId="${param.forumId}"/>
  <h2>Threads in this forum (with forum ID ${param.forumId})</h2>
  Number of threads: ${forum.threadCount}.
  Number of postings: ${forum.postingCount}.
  <ul>
    <c:forEach var="theThread" items="${forum.threads}">
      <li>
        <a href="${forum.url}/posting/${theThread.id}">
          Title is "${theThread.title}"
          and thread ID is ${theThread.id}
        </a>
      </li>
    </c:forEach>
  </ul>
</c:if>
```

The most significant points in the above listing are highlighted and described below:

<forum:forum id="forum" forumId="${param.forumId}"/>
The **forum:forum** tag (see section 9.5) is used to create a **PresentationForum** bean (see section 12.1) called **forum**.

<c:forEach var="theThread" items="${forum.threads}">
The code cycles through the first 10 threads in the forum, creating a **PresentationThread** bean (see section 12.2) called **theThread** to represent each thread.

3.3.3 Displaying a Posting

To display the content of a posting (plus a list of replies to the posting):

```
<c:if test="${!empty param.postingId}">
  <forum:posting id="posting" postingId="${param.postingId}"/>
</c:if>

<c:if test="${!empty posting}">
  <h2>
    You're viewing posting with ID ${posting.id}
    from forum with ID ${posting.forum.id}
  </h2>
  <p>
```

```
    <c:out escapeXml="false" value="${posting.body}"/>
  </p>
  <p>
    Created ${posting.creationDate}
    <c:if test="${posting.thread.id != posting.id}">
      as a reply to
      <a href="${pageContext.request.requestURL}?postingId=${posting.thread.id}">
        posting with ID ${posting.parent.id}
      </a>
    </c:if>
    <c:if test="${posting.thread.id == posting.id}">
      (first posting on this thread)
    </c:if>
  </p>
  <c:if test="${posting.repliesCount > 0}">
    <h3>Has ${posting.repliesCount} replies:</h3>
    <ul>
      <c:forEach var="reply" items="${posting.replies}">
        <li>
          <a href="${reply.url}">
            Posting ID ${reply.id}, created ${reply.creationDate}
          </a>
        </li>
      </c:forEach>
    </ul>
  </c:if>
</c:if>
```

The most significant points in the above listing are highlighted and described below:

`<forum:posting id="posting" postingId="${param.postingId}"/>`
The **`forum:posting`** tag (see section 9.7) is used to create a **`PresentationPosting`** bean (see section 12.3) called **`posting`**.

`<c:out escapeXml="false" value="${posting.body}"/>`
This displays the content of the posting (returned from the **`posting`** bean's body property).

`<c:if test="${posting.thread.id != posting.id}">` and `<c:if test="${posting.thread.id == posting.id}">`
These tests check whether or not the posting is a root posting (that is, a thread). If a posting's thread ID is the same as its posting ID, then it is a root posting or thread, and not a reply to another posting.

`<c:forEach var="reply" items="${posting.replies}">`
This code cycles through the posting's replies, creating a new **`PresentationPosting`** bean called **`reply`** to represent each thread.

3.3.4 Adding Postings/Comments/Complaints

Here is a template for displaying form that allows users to add postings to a forum. The template is at **`/template/v0/components/forum/post.jsp`**:

```
<c:if test="${!empty param.forumId}">
  <h2>Post a message to forum with ID ${param.forumId}</h2>
</c:if>
<c:if test="${!empty posting}">
  <h3>Post a follow up to this post</h3>
</c:if>
```

```
<html:form action="/insert/posting">
  <html:hidden property="com.escenic.context.path.initial" value="/" />
  <html:hidden property="targetUrl" value="${pageContext.request.requestURL}"/>
  <html:hidden property="errorUrl" value="${pageContext.request.requestURL}"/>
  <html:hidden property="cancelUrl" value="${pageContext.request.requestURL}"/>
  <html:hidden property="relatedArticleId" value="-1" />
  <html:textarea property="body" rows="5" cols="40" />
  <html:hidden property="title" value="no title"/>
  <html:hidden property="email" value="john@example.com"/>
  <html:hidden property="typeName" value="posting" />


  <c:if test="${!empty param.forumId}">
    <html:hidden property="forumId" value="${param.forumId}" />
    <html:hidden property="parentId" value="-1" />
  </c:if>
  <c:if test="${!empty posting}">
    <html:hidden property="forumId" value="${posting.forum.id}" />
    <html:hidden property="parentId" value="${posting.id}" />
  </c:if>

  <html:errors property="body" />
  <br/>
  <html:submit />
</html:form>
```

Here is a template for displaying form that allows users to add complaints to a posting. The template is at **/template/index.jsp**:

```
<c:if test="${!empty param.forumId}">
  <h2>Post a message to forum with ID ${param.forumId}</h2>
</c:if>
<c:if test="${!empty posting}">
  <h3>Post a complaint to this posting </h3>
</c:if>

<html:form action="/insert/complaint">
  <html:hidden property="com.escenic.context.path.initial" value="/" />
  <html:hidden property="targetUrl" value="${pageContext.request.requestURL}"/>
  <html:hidden property="errorUrl" value="${pageContext.request.requestURL}"/>
  <html:hidden property="cancelUrl" value="${pageContext.request.requestURL}"/>
  <html:hidden property="relatedArticleId" value="-1" />
  <html:textarea property="body" rows="5" cols="40" />
  <html:hidden property="title" value="no title"/>
  <html:hidden property="email" value="john@example.com"/>
  <html:hidden property="typeName" value="complaint" />


  <c:if test="${!empty param.forumId}">
    <html:hidden property="forumId" value="${param.forumId}" />
    <html:hidden property="parentId" value="-1" />
  </c:if>
  <c:if test="${!empty posting}">
    <html:hidden property="forumId" value="${posting.forum.id}" />
    <html:hidden property="parentId" value="${posting.id}" />
  </c:if>

  <html:errors property="body" />
```

```
  <br/>
  <html:submit />
</html:form>
```

The most significant points in the above listing are highlighted and described below:

`<html:form action="/insert/posting">`
The **`action`** attribute here associates the form with a Struts action path. The **`struts-config-forum.xml`** file (see section 3.1.3) specifies that these action paths require a **`PostingForm`** to be submitted for processing by an **`InsertPostingAction`** or **`AuthenticatedInsertPostingAction`**. The rest of the form defines the fields specified by **`CommentForm`**.

`<html:hidden property="relatedArticleId">`
The **`relatedArticleId`** determines whether or not this form is an article comment form. For an ordinary posting, it should be set to **`-1`** as above. For a comment posting, it should be set to the ID of the relevant article.

3.3.5 Article Commenting

The Forum plug-in can be used to allow readers to comment on articles. In order for an article to be "commentable", its content type must include a decorator:

```
<content-type name="content-type-name">
  ...
  <ui:decorator
 class="com.escenic.forum.presentation.ForumPresentationArticleDecorator"/>
  ...
</content-type>
```

3.3.6 Listing Comments

The decorator described in section 3.3.5 provides easy access to any comment made in relation to a given content type: you simply retrieve a list of comments using **`article.relatedElements`**.

To retrieve the number of comments made on an article:

```
${article.relatedElements.forum.commentCount}
```

To retrieve the actual comments:

```
<c:forEach var="comment" items="${article.relatedElements.forum.comments}">
  <p>${comment.title}</p>
</c:forEach>
```

If threaded comments (that is, comments with replies) are being used, then retrieving the next level of comments is simply a matter of iterating over the current comment's replies:

```
<c:forEach var="comment" items="${article.relatedElements.forum.comments}">
  <p>${comment.title}</p>
  ...
  <p>Replies:<p>
  <c:forEach var="reply" items="${comment.replies}">
    <p>${comment.title}</p>
  </c:forEach>
```

```
  ...
</c:forEach>
```

3.3.7 Listing Pingbacks

Pingbacks are accessible via the decorator described in section 3.3.5 and can be listed in the same way as article comments. To retrieve the number of pingbacks:

```
${article.relatedElements.forum.pingBackCount}
```

To retrieve the actual pingbacks:

```
<c:forEach var="pingback" items="${article.relatedElements.forum.pingBacks}">
  <p>${pingback.title}</p>
</c:forEach>
```

4 Controlling User Input

4.1 CAPTCHA Support

CAPTCHA is a technique for preventing forms being filled in automatically by programs. A **challenge** (usually an image containing distorted characters) is displayed, and the user submitting a form is required to provide a correct **response** (usually by entering the characters displayed in the image).

Escenic Content Engine comes with built-in feature to add Captcha to different actions. The Forum plug-in action classes use this Captcha support allowing you to easily add CAPTCHA fields to your posting submission forms. Please see the **Advanced Developer Guide** for instructions on how to setup Captcha.

In order to configure your Forum action classes to verify captcha response, Open **`WEB-INF/struts-config-forum.xml`** for editing, and enable the CAPTCHA checks you want by setting the parameter **`checkCaptcha`** to **`true`**. For example:

```
<action path="/insert/posting"
type="com.escenic.forum.struts.presentation.InsertCommentAction"
parameter="checkCaptcha=true"
...
/>
```

The following Struts actions support CAPTCHA checks:

- **`com.escenic.forum.struts.presentation.InsertCommentAction`**
- **`com.escenic.forum.struts.presentation.AuthenticatedInsertCommentAction`**
- **`com.escenic.forum.struts.presentation.InsertPostingAction`**
- **`com.escenic.forum.struts.presentation.AuthenticatedInsertPostingAction`**

You will then need to redeploy the publications. For general information about developing and deploying Escenic publications, see the **Escenic Content Engine Template Developer Guide**.

In order to show the Captcha image and include captcha field, please see **Advanced Developer Guide**.

You can include a JSP segment like this in your posting submission HTML form to show the error when the captcha response is invalid:

```
<p class="fieldError" id="errorfield(CAPTCHA)">
  <html:messages
    bundle="Validation"
    id="error"
    message="true"
    property="CAPTCHA">
    <bean:write name="error"/>
  </html:messages>
</p>
```

4.2 Stop Words

The Forum plug-in supports **stop words**. Stop words are words that are disallowed in postings. If you specify a list of stop words for your publication, then postings containing any of those words will not be created. To help page developers display information on stop words `InsertPostingAction` and `InsertCommentAction` save stop word information in a session attribute called `escenic-forum-posting-stopword-occurrences`. You can also use the constant defined as: `com.escenic.forum.presentation.PresentationConstants.RQ_ATTR_POSTING_STOPWORD_OCCURRENCE`

The session attribute contains a `java.util.Map` which consists of (*fieldName*, `neo.xredsys.api.stopword.StopwordOccurrenceInfo`) pairs. The `neo.xredsys.api.stopword.StopwordOccurrenceInfo` object offers the following facilities:

- A `SourceStr` property containing a copy of the offending field.
- A `getOccurredStopwordIterator()` method that returns an iterator for accessing the individual stop words in the field.
- A `getStopwordOccurrences()` method that returns a `java.util.List` of `Integer` values specifying the positions of the stop words in the field.

See the Javadoc for further information about `neo.xredsys.api.stopword.StopwordOccurrenceInfo`.

4.3 Controlling HTML Input

An important feature of Forum is that it allows you to define white lists of HTML elements and attributes that the users may use when creating postings and article comments.

Controlling that the user does not input unwanted markup is important for many reasons. First of all, the markup may alter the design of your web page. More seriously, it may inject markup and JavaScript which loads content from other sites and displays them on your page or they may insert markup which re-directs your visitors to external web sites without they wanting - or knowing - it.

For these reasons, Forum will per default allow the user to only input the following HTML elements:

- p
- br
- b
- strong
- i
- em
- u
- code
- cite
- blockquote
- acronym
- strike

Similalry, the default white list for attributes is:

- title
- cite

You may configure change these white lists by editing the **`/com/escenic/forum/presentation/PresentationManager`** Nursery component inside your publication.

4.3.1 A Note To Template Developers

Some web sites want to offer a rich text editor for their users so that they do not have to enter the HTML tags themselves. There are several JavaScript based rich text editors around that you can use. They can all be configured with what buttons you want to allow your users to use and it is important that these correspond to the ones you have defined in your white lists.

Below, we show you how to add TinyMCE to the posting form shown in the **`forum-demo`** web application:

Inside the **`head`** element, add the following:

```
<script
  type="text/javascript"
  src="http://tinymce.moxiecode.com/js/tinymce/jscripts/tiny_mce/tiny_mce.js">
</script>
<script type="text/javascript">
tinyMCE.init({
  mode : "textareas"
});
</script>
```

When you reload your web page, you will now see a simple, rich text editor. For production environments, we recommend downloading TinyMCE and refer to your local copy rather than the absolute URI above.

4.3.2 A Note To System Integrators

Technically speaking, the user input is written as it is to the database and it is first "washed" when presenting it. Thus, the template developer does not need to worry about escaping what comes from Forum, such as outputting the body text with **`${posting.body}`**.

Similarly, system integrators can know that the posting data in the database can be full XML, HTML or whatever, allowing for easy import from 3rd party systems as well as integrating with other export channels (mobile devices, tablets, PDFs++) which may want to use more markup elements than you want on the web site.

4.4 Controlling Flooding

Forum has some protection mechanism so that the users cannot flood the forum with too many postings. The basic idea is to stop the hackers who may write script to send requests automatically. This feature may not be needed if you are already using CAPTCHA support.

Basically, Forum keeps track of how many postings are created from a specific client domain at a certain time period. If it exceeds the threshold, Forum blocks that client domain for a configured amount of time. However, the user can unblock his domain by posting again with CAPTCHA verification if flooding control mechanism is configured to support this.

4.4.1 Configuring Flood Control

Flooding control is disabled by default. Please find the following configuration properties file in Forum distribution and copy it to your configuration directory. Change it to enable this feature in your site and configure the threshold and other properties.

```
misc/siteconfig/com/escenic/forum/FloodPreventionConfig.properties
```

4.4.2 Action Class Support

This support has been added to following action classes

```
com.escenic.forum.struts.presentation.InsertPostingAction
com.escenic.forum.struts.presentation.AuthenticatedInsertPostingAction
```

If the client domain is blocked, the action classes will set ActionMessage to the request. The action message key **`forum.validate.error.flooding`** will be set when **`useCaptchaToUnblock`** is set to false indicating that the user should be informed that too many requests are posted from his domain.

If **`useCaptchaToUnblock`** is set to true, the action message key is **`forum.validate.error.flooding`** in this case. This way, the template developer can show CAPTCHA verification input interface by checking this message key. Please have a look at the **Advanced Developer Guide** for using CAPTCHA support.

5 Flagging Posting

The site users can flag a posting if they find it obscene or inappropriate. A user can flag a posting once. The flag count of a posting is kept. If flag count of a posting reaches a configured threshold, it will be marked as flagged and shown in Dashboard plugin in the flagged tab. This way the moderator can take action on the flagged item.

Please note that, in order to show the flagged postings in Dashboard, Solr schema has to be configured mentioned in section 2.7.3 section.

5.1 Feature properties for Flagging

Some behavior of flagging functionality in Forum can be configured using the **`feature`** publication resource.

1. **`com.escenic.qualification.flag.threshold`**: This feature property configures the threshold of flag count to mark a posting as flagged. If this property is not configured, flag count of a posting will be increased when a posting is flagged, but it will not appear in Dashboard.
2. **`com.escenic.qualification.flag.unpublish`**: This feature property configures whether a posting will be unpublished from the site when it is marked as flagged. In that case, the posting item will have a moderation state as 'pending'. The default value of this features is true. This can be configured as:

```
com.escenic.qualification.flag.unpublish=false
```

5.2 Template for flagging

Forum provides an action class **`com.escenic.forum.qual.presentation.SubmitPostingQualification`** to flag a posting. A JSP tag is also provided to determine whether the user has flagged this posting. Here is an example template to use this Action and JSP tag.

Configuring struts-config.xml

```
<form-beans>
  <form-bean
      name="com.escenic.forum.qual.presentation.QualificationForm"
      type="com.escenic.forum.qual.presentation.QualificationForm"/>
  </form-beans>

<actions>
  <action
      path="/flag/posting"
      type="com.escenic.forum.qual.presentation.SubmitPostingQualification"
      name="com.escenic.forum.qual.presentation.QualificationForm">
  </action>
</actions>
```

Template for using flagging action

```
<%@ taglib prefix="qual" uri="http://www.escenic.com/taglib/escenic-content-
qualification" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="html" uri="http://struts.apache.org/tags-html" %>

<!-- To check whether the user has flagged this posting already -->
<qual:qualInfo id="qualInfoBean"
               objectId="${posting.id}"
               objectType="posting"
               userId="${user.id}"
               qualificationType="flag" />
<c:choose>
    <c:when test="${qualInfoBean != null}">
        This posting is flagged by you!
    </c:when>
    <c:otherwise>

      <!-- Using the action to flag the posting -->
      <html:form action="/flag/posting">
        <html:hidden property="contentId" value="${posting.id}"/>
        <html:hidden property="qualificationType" value="flag"/>
        <html:hidden property="qualificationValue" value="1"/>
        <html:submit value="Flag This!"/>
      </html:form>
    </c:otherwise>
</c:choose>
```

6 Pingback

Pingback is a means for web authors to request notification when somebody links to one of their documents. Typically, web publishing software automatically informs the relevant parties on behalf of the user, making it possible to automatically create links to referring documents.

For example:

1. Alice writes an interesting article in her blog.
2. Bob reads the article writes about it on his blog and links back to Alice's original post.
3. Bob's blogging software uses pingback to automatically notify Alice that her post has been linked to.
4. Alice's blogging software includes this information on her site.

Forum provides a web application called **`pingback`**, which listens for pings, verifies them and stores them as as special type of posting.

6.1 Adding Pingback Support

This section describes how to enable pingback support on your site.

6.1.1 Deploy Pingback Web Application

The **`pingback`** web application supplied with the Forum plug-in listens for pings, verifies and stores them. To deploy the application:

1. Log in as **`escenic`** on your **assembly-host**.
2. Copy **`/opt/escenic/plugins/forum/wars/forum-pingback-server.war`** to **`/opt/escenic/assemblytool/publications/`**.

   ```
   $ cp /opt/escenic/plugins/forum/wars/forum-pingback-server.war /opt/escenic/
   assemblytool/publications/
   ```

3. Create a **`pingback.properties`** file in the **`/opt/escenic/assemblytool/publications`** folder with the following contents:

   ```
   context-root=/pingback
   name=pingback
   source-war=forum-pingback-server.war
   ```

4. Rebuild, deploy and restart the Content Engine as described in section 2.4.

6.1.2 Add Publication Feature

Once you have deployed the **`pingback`** web application, you need to add a publication feature that tells the Forum plug-in to use it. You do this by adding the following to your publication's **`feature`** resource:

```
forum.pingback=true
```

For general information about publication resources and how update them, see the **Escenic Content Engine Template Developer Guide**.

6.1.3 Modify Content Type Resource

You can enable pingback support for specific content types in a publication. To do so, you need to modify the publication's **`content-type`** resource. For each content type resource that is to support pingbacks, you must add a parameter called **`com.escenic.forum.pingback`** and set it to **`true`**. For example:

```
<content-type name="blog">
  <parameter name="com.escenic.forum.pingback" value="true"/>
  ...
</content-type>
```

It may be the case that you want to be able to disable this pingback support for some specific content items. In this case you should also add a boolean field called **`ping-back`** to the content type as follows:

```
<content-type name="blog">
  <parameter name="com.escenic.forum.pingback" value="true"/>
  ...
  <panel name="pingback">
    <field type="boolean" name="ping-back">
      <ui:value-if-unset>true</ui:value-if-unset>
    </field>
  </panel>
</content-type>
```

Any content item in which **`ping-back`** is set to false will then not support pingbacks.

6.1.4 Create Pingback Forum

All pingbacks are stored as postings in a forum. So you need to create a forum that will be used exclusively for storing pingbacks, and then configure your publication to use this forum for storing pingbacks. To do this:

1. Create a forum content item in Content Studio. For a description of how to do this, see section 7.2.You then need to
2. When you have created the forum content item in Content Studio, click on it's **Properties** tab and make a note of:
   - **Either** the content item's **Id**
   - **Or** the content item's **Source** and **Source id**
3. Open your publication in Web Studio and add the following parameters to the root section:
   - **Either** set **`forum.pingbackForum.id`** to the content item id that you recorded in step 2
   - **Or** set **`forum.pingbackForum.source`** and **`forum.pingbackForum.sourceId`** to the content item source and source id that you recorded in step 2

   For a description of how to set section parameters in Web Studio, see the **Escenic Content Engine Publication Administrator Guide**.

6.1.5 Configure Pingback Server URL

In order for the pingback set-up to work, every pingback-enabled publication must know the URL of the pingback server you deployed (as described in section 6.1.1). In order to achieve this you will need to log on to the system where your publication web applications are developed and maintained, and do the following in each pingback-enabled publication:

1. Add a **`WEB-INF/localconfig/com/escenic/forum/presentation`** folder to the publication web application. For example:

   ```
   $ cd /path/to/web-app
   $ mkdir -p WEB-INF/localconfig/com/escenic/forum/presentation
   $ cd WEB-INF/localconfig/com/escenic/forum/presentation
   ```

2. Create a text file called **`PingbackHeaderProcessor.properties`** in the new folder.
3. Open the new file for editing and add the following line:

   ```
   pingbackURL=http://host-name/pingback
   ```

   where *host-name* is the host name or IP address of the host on which you deployed the pingback server.

You will then need to redeploy the publications. For general information about developing and deploying Escenic publications, see the **Escenic Content Engine Template Developer Guide**.

7 Managing Forums

This chapter contains the information a publication administrator needs to be able to manage the forums in a publication.

7.1 Access Rights

The Forum plug-in adds one forum-related role to the Content Engine's access rights system. You can see the role and assign it to users of a publication in Web Studio:

1. Start a browser and enter the following URL in the address bar:

   ```
   http://host/escenic/
   ```

   where *host* is the host name or IP address of your Content Engine host.
2. A login form is displayed - enter the publication administrator's user name and password.
3. Select **Person and user archive**.
4. Select a user to whom you wish to grant a forum-related role and then click on **Global roles**. You will see that **Forum moderator** role has been appended to the list of global roles. See section 7.1.1 for a description of these roles.

For general information about Content Studio, see the **Escenic Content Engine Publication Administrator Guide**.

7.1.1 Forum Moderator Role

A user with this role has forum posting moderation rights for the whole publication. This means that:

- The user is allowed to log in to the **`Dashboard`** application.
- When the user logs in to the **`Dashboard`** application, she will be able to moderate the postings of the publication.

7.2 Creating Forums

To create a forum:

1. Start Content Studio and log in to the publication in which you want to create the forum.
2. Select **File** > **New** > *forum-content-type*, where *forum-content-type* is the name of a forum content type defined in your publication. A content item of the selected type is then created and opened for editing. The number and type of fields in the content item depend on the content type definition, but there will always be:
   - Some kind of name or title field that is used as the title of the forum
   - A field that allows you to select the forum's moderation scheme
3. Fill in the fields. The moderation scheme options are:

Post-moderation
This means that postings will be posted before they have been moderated, and may subsequently be withdrawn by the moderator.

Pre-moderation
This means that postings will not be posted until they have been approved by the moderator.

The options may not have these names, since the names displayed in Content Studio are defined in the content item's content type definition.

4. Add the forum to the section(s) in which you want it to appear.
5. Set the **State** to **Published** and click on **Save**.

8 Performance Testing Forum

Forum has a built-in security mechanism to ensure that malicious bots and scripts on the Internet do not flood your forums. However, if you want to load test a forum in order to find out exactly how many postings per second the forum can handle on **your** website with your JSP templates and server stack, then that is exactly what you want to do. For performance testing, therefore, you need to be able to circumvent this Forum security feature.

You can disable the security feature by setting a configuration parameter called **`/com/escenic/forum/ForumManager/performanceTestingAllowed`** to **`true`**.

To do this:

1. Start a browser and go to the Content Engine's administration application by entering:

   ```
   http://host-name/escenic-admin
   ```

   in the address field (where *host-name* is the host name or IP address of your Content Engine host).
2. Click on **Component Browser** to display the Content Engine's component browser application.
3. Click down to the required component: **com** > **escenic** > **forum** > **ForumManager**.
4. Click on **performanceTestingAllowed** in the displayed **Properties** table.
5. Enter **`true`** in the **New value** field and click on **Submit**.

The security mechanism is now disabled and will remain disabled until you set **`/com/escenic/forum/ForumManager/performanceTestingAllowed`** back to **`false`** or until the Content Engine is restarted.

With the security mechanism disabled, you can use load testing tools such as siege to load test your forum installation.

> It is of course also possible to set **`/com/escenic/forum/ForumManager/performanceTestingAllowed`** to **`true`** by editing the appropriate **`.properties`** file in a configuration layer. You are, however, strongly advised **not** to do this, since it carries the risk of forgetting to set the parameter back to **`false`**, and thus exposing your production system to real attacks.

9 forum Tag Library

9.1 forum:expiresCache

Expires the Jsp-cache based on the Forum-object specified.

Syntax

```
<forum:expiresCache
   objectId="..."?
   objectType="..."?/>
```

Attributes

`objectId`
Id of the Forum-object(forum, thread or posting) we will use to expire the Jsp-cache. This attribute must be used together with objectType.

`objectType`
Name of type of Forum-object(forum, thread or posting) we will use to expire the Jsp-cache. This attribute must be used together with objectId. Valid values are: 'forum', 'thread', 'posting'

9.2 forum:groups

Returns a **`Collection`** containing all the forums in one or more sections. This tag is deprecated. Use **`forum:forums`** instead.

Syntax

```
<forum:groups
   id="..."
   recursive="..."?
   sectionId="..."?
   sectionIds="..."?/>
```

Attributes

`sectionId`
This attribute is deprecated. Use **`sectionIds`** instead.

`sectionIds`
The sections from which forums are to be retrieved, specified as a list of one or more section IDs.

`recursive`
If set to **`true`**, then the sub-trees of the sections specified with the **`sectionIds`** attribute are also searched for forums. For performance reasons, you should avoid using this attribute for sections with large section subtrees. The default value is **`false`**.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned forum **`Collection`**.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`java.util.Collection`**.

9.3 forum:forums

Returns a **`Collection`** containing all the forums in one or more sections.

Syntax

```
<forum:forums
   id="..."
   publicationIds="..."?
   recursive="..."?
   sectionIds="..."?/>
```

Attributes

`publicationIds`
A comma separated list of IDs of the publications from which the forums to be retrieved. If this attribute is specified then the others will be ignored. It is convenient when all the forums from a specific publication or several publications are required.

`sectionIds`
The sections from which forums are to be retrieved, specified as a list of one or more section IDs. If **`publicationIds`** attribute is specified then, it will not take effect.

`recursive`
If set to **`true`**, then the sub-trees of the sections specified with the **`sectionIds`** attribute are also searched for forums. For performance reasons, you should avoid using this attribute for sections with large section subtrees. The default value is **`false`**.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned forum **`Collection`**.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`java.util.Collection`**.

9.4 forum:group

Returns a specified forum. This tag is deprecated. Use **`forum:forum`** instead.

Syntax

```
<forum:group
   forumId="..."?
   id="..."/>
```

Attributes

`forumId`
The ID of the forum to retrieve. If this attribute is omitted, then the ID of the default content item is used.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned forum.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.presentation.PresentationForum`**.

9.5 forum:forum

Returns a specified forum.

Syntax

```
<forum:forum
   forumId="..."?
   id="..."/>
```

Attributes

`forumId`
The ID of the forum to retrieve. If this attribute is omitted, then the ID of the default content item is used.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned forum.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.presentation.PresentationForum`**.

9.6 forum:latestThreads

Returns a **`List`** of the most recently active threads in one or more forums. The list is sorted by the creation date of the most recent posting in each thread. This means that an old thread may be at the top of the list if a reply has been posted to it recently.

This tag is optimized for performance, at the risk of losing some accuracy. It is assumed that the list can be sorted by article id, rather than actual creation date. This greatly improves performance on a high-traffic site.

Syntax

```
<forum:latestThreads
   forumIds="..."?
   id="..."
   max="..."
   name="..."?/>
```

Attributes

`name`
The name of a JSP attribute that specifies the forums to use. The attribute may be of type **`PresentationForum`**, **`Forum`**, **`String`** or **`Integer`** or a **`Collection`** of such objects - for example, the **`Collection`** returned by the **`groups`** tag:

If the **`forumIds`** attribute is not specified then this attribute is required.

`forumIds`
A forum ID (or a comma-separated list of forum IDs) from which the most recently active threads are to be retrieved. If the **`name`** attribute is not specified then this attribute is required.

`max`, mandatory
The maximum number of threads to return.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned **`List`** of threads.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.presentation.PresentationList`**.

9.7 forum:posting

Returns a specified posting.

Syntax

```
<forum:posting
   id="..."
   postingId="..."?/>
```

Attributes

`postingId`
The ID of the posting to retrieve.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned posting.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.presentation.PresentationPosting`**.

9.8 forum:relatedForum

Returns the related forum of a specified forum.

Syntax

```
<forum:relatedForum
   forumId="..."?
   id="...">
   ...
</forum:relatedForum>
```

Attributes

`forumId`
The ID of the forum from which the related forum is to be retrieved.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned forum.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.presentation.PresentationForum`**.

9.9 forum:threads

Returns a **`Collection`** containing the threads in a specified forum.

Syntax

```
<forum:threads
   forumId="..."?
   id="..."/>
```

Attributes

`forumId`
The ID of the forum from which the threads are to be retrieved.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned thread **`Collection`**.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`java.util.Collection`**.

9.10 forum:thread

Returns a specified thread.

Syntax

```
<forum:thread
   id="..."
   threadId="..."?/>
```

Attributes

`threadId`
The ID of the thread to be retrieved.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned thread.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.presentation.PresentationThread`**.

9.11 forum:postings

Returns a **`Collection`** containing the postings in a specified forum.

Syntax

```
<forum:postings
   forumId="..."?
   id="..."/>
```

Attributes

`forumId`
The ID of the forum from which the postings are to be retrieved.

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned posting **`Collection`**.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`java.util.List`**.

10 qual Tag Library

10.1 qual:qualInfo

Retrieves qualification information for a specific content item by a specified user

Syntax

```
<qual:qualInfo
   id="..."
   objectId="..."
   objectType="..."
   qualificationType="..."
   userId="..."/>
```

Attributes

`objectId`, mandatory
The object ID for which to retrieve the qualification information

`objectType`, mandatory
The type of object to retrieve qualification information. Supported types are - article, posting

`userId`, mandatory
Qualification information is retrieved only for the user having this user ID

`qualificationType`, mandatory
The type of qualification information to retrieve. Supported values are - flag, rate, favorite

`id`, mandatory, no runtime expressions
The name of the scripting variable that will be created to access the returned qualification information i.e. **`com.escenic.forum.qual.QualificationBean`** for a posting.

Scripting variable (id)

A scripting variable will be defined using the value of the **`id`** attribute as its name. The variable is of type **`com.escenic.forum.qual.QualificationBean`**.

11 Struts Reference

The Forum plug-in includes the Struts components described in the following sections, which simplify the process of adding posting submission functionality to Escenic publications. If you are an Escenic template developer with a working knowledge of Struts then these descriptions should be sufficient to enable you to create the necessary HTML forms and set up the Struts actions required to submit them.

The components are configured in the Forum plug-in's Struts configuration files (see section 3.1). For a proper introduction to Struts, see http://struts.apache.org/primer.html.

11.1 PostingForm

`PostingForm` inherits properties from:

- **`ActionForm`**

It also has the following properties of its own:

`body`
String
The body of the posting.

`parentId`
String
The ID of the parent posting. If not specified, a new thread is created.

`postingType`
Type
The type of this posting. One of two values may be specified: 'normal' or 'complaint'.

`forumId`
String
The forum to which the posting is to be directed.

`title`
String
The title of the posting.

`email`
String
The email address of the person submitting the posting.

`field`
String
A field to be stored in the posting.

`errorUrl`
String
The URL to which the request is to be forwarded if the action fails. If not specified, the error mapping specified in struts-config.xml is used.

`targetUrl`
String
The URL to which the request is to be forwarded when the action is successfully completed. If not specified, the success mapping specified in struts-config.xml is used.

11.2 CommentForm

`CommentForm` inherits properties from:

- **`com.escenic.forum.struts.presentation.PostingForm`**

It also has the following properties of its own:

`threadTitle`
String
The title of the related thread. Only used when creating an article discussion.

`threadBody`
String
The body of the related thread. Only used when creating an article discussion.

`relatedArticleId`
String
The id of the content item being commented on.

11.3 ComplaintForm

`ComplaintForm` inherits properties from:

- **`com.escenic.forum.struts.presentation.PostingForm`**

It has no properties of its own.

11.4 InsertPostingAction

Creates a Forum plug-in posting from the information submitted in a PostingForm. When the action is completed, it forwards the request to the URL specified in the form's 'targetUrl' property. If the request fails or the user clicks Cancel, the action is instead forwarded to URL specified in the form's 'errorUrl' or 'cancelUrl' property as appropriate. Note that this action class has a protected method with the signature 'protected int getSectionId(final HttpServletRequest pRequest, final PostingForm pForm);'. You can call this method to find out whether or not the new posting is to be added to any additional section. The method returns either a valid section ID or -1. If it returns -1 then the posting will not be added to any extra sections, only the forum home section.

Struts config

```
<action path="/forum/createPosting"
 type="com.escenic.forum.struts.presentation.InsertPostingAction"
 name="com.escenic.forum.struts.presentation.PostingForm"
```

```
 input="/forum/createPosting-input.jsp"
 validate="true">
</action>
```

Action Form

This action can be used with the following form:

`com.escenic.forum.struts.presentation.PostingForm`
The form contains the posting to be added to the forum. The form's 'targetUrl' field is particularly important because it contains the URL to which the request is to be redirected after the posting has been added.

Action Forwards

If no errors occur adding the posting, the request is forwarded to the URL specified in the form's 'targetUrl' field. A 'postingId' parameter containing the numeric ID of the new posting is appended to the forwarded request. If the user clicks the form's 'Cancel' button, then the request is forwarded to the URL specified in the form's 'cancelUrl' field. If an error occurs, then the request is forwarded to the URL specified in the form's 'errorUrl' field.

Action Error

The pseudo-properties 'user' and 'developer' are used to classify errors as user error or developer errors. User errors should be handled in such a way as to allow the user to recover, while developer errors can be considered fatal. Note that if the 'errorUrl' property is not set, then an 'IllegalArgumentException' will be thrown, which will usually produce a stack trace.

`forum.user.postingAlreadySubmitted`
(On the **`user`** property.) If the posting has been submitted already (typically the user clicked reload in the browser, trying to resubmit the same posting).

`forum.user.generic.error`
(On the **`user`** property.) If there is a fatal error when trying to add the posting.

`forum.error.server`
(On the **`developer`** property.) If there is a fatal error when trying to add the posting.

11.5 AuthenticatedInsertPostingAction

Creates a Forum plug-in posting from the information submitted in a PostingForm, in the same way as InsertPostingAction. AuthenticatedInsertPostingAction, however, requires the current user to be logged in. When the action is completed, it forwards the request to the URL specified in the form's 'targetUrl' property. If the request fails or the user clicks Cancel, the action is instead forwarded to URL specified in the form's 'errorUrl' or 'cancelUrl' property as appropriate. Note that this action class has a protected method with the signature 'protected int getSectionId(final HttpServletRequest pRequest, final PostingForm pForm);'. You can call this method to find out whether or not the new posting is to be added to any additional section. The method returns either a valid section ID or -1. If it returns -1 then the posting will not be added to any extra sections, only the forum home section.

Struts config

```
<action path="/forum/createPosting"
```

```
 type="com.escenic.forum.struts.presentation.AuthenticatedInsertPostingAction"
 name="com.escenic.forum.struts.presentation.PostingForm"
 input="/forum/createPosting-input.jsp"
 validate="true">
</action>
```

Action Form

This action can be used with the following form:

`com.escenic.forum.struts.presentation.PostingForm`
: The form contains the posting to be added to the forum. The form's 'targetUrl' field is particularly important because it contains the URL to which the request is to be redirected after the posting has been added.

Action Forwards

If no errors occur adding the posting, the request is forwarded to the URL specified in the form's 'targetUrl' field. A 'postingId' parameter containing the numeric ID of the new posting is appended to the forwarded request. If the user clicks the form's 'Cancel' button, then the request is forwarded to the URL specified in the form's 'cancelUrl' field. If an error occurs, then the request is forwarded to the URL specified in the form's 'errorUrl' field.

Action Error

The pseudo-properties 'user' and 'developer' are used to classify errors as user error or developer errors. User errors should be handled in such a way as to allow the user to recover, while developer errors can be considered fatal. Note that if the 'errorUrl' property is not set, then an 'IllegalArgumentException' will be thrown, which will usually produce a stack trace.

`forum.user.postingAlreadySubmitted`
: (On the **`user`** property.) If the posting has been submitted already (typically the user clicked reload in the browser, trying to resubmit the same posting).

`forum.user.generic.error`
: (On the **`user`** property.) If there is a fatal error when trying to add the posting.

`forum.error.server`
: (On the **`developer`** property.) If there is a fatal error when trying to add the posting.

11.6 InsertCommentAction

Creates a Forum plug-in posting from the information submitted in a CommentForm. When the action is completed, it forwards the request to the URL specified in the form's 'targetUrl' property. If the request fails or the user clicks Cancel, the action is instead forwarded to URL specified in the form's 'errorUrl' or 'cancelUrl' property as appropriate. Note that this action class has a protected method with the signature 'protected int getSectionId(final HttpServletRequest pRequest, final PostingForm pForm);'. You can call this method to find out whether or not the new posting is to be added to any additional section. The method returns either a valid section ID or -1. If it returns -1 then the posting will not be added to any extra sections, only the forum home section.

Struts config

```
<action path="/forum/createPosting"
 type="com.escenic.forum.struts.presentation.InsertCommentAction"
 name="com.escenic.forum.struts.presentation.CommentForm"
 input="/forum/createPosting-input.jsp"
 validate="true">
</action>
```

Action Form

This action can be used with the following form:

`com.escenic.forum.struts.presentation.CommentForm`
: The form contains the posting to be added to the forum. The form's 'targetUrl' field is particularly important because it contains the URL to which the request is to be redirected after the posting has been added.

Action Forwards

If no errors occur adding the posting, the request is forwarded to the URL specified in the form's 'targetUrl' field. A 'postingId' parameter containing the numeric ID of the new posting is appended to the forwarded request. If the user clicks the form's 'Cancel' button, then the request is forwarded to the URL specified in the form's 'cancelUrl' field. If an error occurs, then the request is forwarded to the URL specified in the form's 'errorUrl' field.

Action Error

The pseudo-properties 'user' and 'developer' are used to classify errors as user error or developer errors. User errors should be handled in such a way as to allow the user to recover, while developer errors can be considered fatal. Note that if the 'errorUrl' property is not set, then an 'IllegalArgumentException' will be thrown, which will usually produce a stack trace.

`forum.user.postingAlreadySubmitted`
: (On the **`user`** property.) If the posting has been submitted already (typically the user clicked reload in the browser, trying to resubmit the same posting).

`forum.user.generic.error`
: (On the **`user`** property.) If there is a fatal error when trying to add the posting.

`forum.error.server`
: (On the **`developer`** property.) If there is a fatal error when trying to add the posting.

11.7 AuthenticatedInsertCommentAction

Creates a Forum plug-in posting from the information submitted in a CommentForm, in the same way as InsertCommentAction. AuthenticatedInsertCommentAction, however, requires the current user to be logged in. When the action is completed, it forwards the request to the URL specified in the form's 'targetUrl' property. If the request fails or the user clicks Cancel, the action is instead forwarded to URL specified in the form's 'errorUrl' or 'cancelUrl' property as appropriate. Note that this action class has a protected method with the signature 'protected int getSectionId(final HttpServletRequest pRequest, final PostingForm pForm);'. You can call this method to find out whether or not the new posting is to be added to any additional section. The method returns either a valid section ID or -1. If it returns -1 then the posting will not be added to any extra sections, only the forum home section.

Struts config

```
<action path="/forum/createPosting"
 type="com.escenic.forum.struts.presentation.AuthenticatedInsertCommentAction"
 name="com.escenic.forum.struts.presentation.CommentForm"
 input="/forum/createPosting-input.jsp"
 validate="true">
</action>
```

Action Form

This action can be used with the following form:

`com.escenic.forum.struts.presentation.CommentForm`
: The form contains the posting to be added to the forum. The form's 'targetUrl' field is particularly important because it contains the URL to which the request is to be redirected after the posting has been added.

Action Forwards

If no errors occur adding the posting, the request is forwarded to the URL specified in the form's 'targetUrl' field. A 'postingId' parameter containing the numeric ID of the new posting is appended to the forwarded request. If the user clicks the form's 'Cancel' button, then the request is forwarded to the URL specified in the form's 'cancelUrl' field. If an error occurs, then the request is forwarded to the URL specified in the form's 'errorUrl' field.

Action Error

The pseudo-properties 'user' and 'developer' are used to classify errors as user error or developer errors. User errors should be handled in such a way as to allow the user to recover, while developer errors can be considered fatal. Note that if the 'errorUrl' property is not set, then an 'IllegalArgumentException' will be thrown, which will usually produce a stack trace.

`forum.user.postingAlreadySubmitted`
: (On the **`user`** property.) If the posting has been submitted already (typically the user clicked reload in the browser, trying to resubmit the same posting).

`forum.user.generic.error`
: (On the **`user`** property.) If there is a fatal error when trying to add the posting.

`forum.error.server`
: (On the **`developer`** property.) If there is a fatal error when trying to add the posting.

12 Bean Reference

The beans described in the following sections are supplied as part of the Forum plug-in.

12.1 PresentationForum

Represents an Escenic forum. A forum can contain threads (represented by **`PresentationThread`** beans) and postings (represented by **`PresentationPosting`** beans).

`PresentationForum` has the properties described in the following sections.

12.1.1 acceptingPostings

If **`true`**, this forum is currently active (that is, open for postings). If **`false`**, then it is not.

Type: **`boolean`**

Example usage

```
${myPresentationForum.acceptingPostings}
```

12.1.2 active

If **`true`**, then this forum is **active**, meaning that it is possible to add new threads and postings to it. If **`false`**, then the forum is not active and new threads and postings cannot be added.

Type: **`boolean`**

Example usage

```
${myPresentationForum.active}
```

12.1.3 answering

If **`true`**, then this qa forum is open for questions. If **`false`**, then it is not.

Type: **`boolean`**

Example usage

```
${myPresentationForum.answering}
```

12.1.4 articleTypeName

The name of the content type on which this forum is based.

Type: **`java.lang.String`**

Example usage

```
${myPresentationForum.articleTypeName}
```

12.1.5 continue

If **`true`**, then this qa forum will remain active (that is, open for postings) after the qa session has finished. If **`false`**, then it will not. If false

Type: **`boolean`**

Example usage

```
${myPresentationForum.continue}
```

12.1.6 description

A description of the forum.

Type: **`java.lang.String`**

Example usage

```
${myPresentationForum.description}
```

12.1.7 fields

The fields of this forum (a forum is actually a special type of content item).

Type: **`java.util.Map<String,PresentationProperty>`**

Example usage

To get all the fields:

```
${myPresentationForum.fields}
```

To get one particular field:

```
${myPresentationForum.fields.field}
```

where *field* is the key of the field you want.

12.1.8 hasRelatedForum

If **`true`**, then this qa forum has a related forum. If **`false`**, then it does not.

Type: **`boolean`**

Example usage

```
${myPresentationForum.hasRelatedForum}
```

12.1.9 homeSection

This forum's home section.

Type: `neo.xredsys.api.Section`

Example usage

```
${myPresentationForum.homeSection}
```

12.1.10 moderated

This property is deprecated: This method is deprecated because Forum no longer uses Moderator webapp for moderation. Use `PresentationForum.fields` (see section 12.1.7) instead and retrieve the content of the `moderation-scheme` field.

If `true`, then this forum is **moderated**, meaning that new postings must be approved by a moderator before they can be published. If `false`, then the forum is not moderated and submitted postings are published immediately.

Type: `boolean`

Example usage

```
${myPresentationForum.moderated}
```

12.1.11 name

The name of the forum.

Type: `java.lang.String`

Example usage

```
${myPresentationForum.name}
```

12.1.12 postingCount

The total number of postings in this forum.

Type: `int`

Example usage

```
${myPresentationForum.postingCount}
```

12.1.13 postings

A list of all postings in this forum. The list is sorted by date, with the newest first.

Type: `java.util.List<PresentationPosting>`

Example usage

To get all the postings:

```
${myPresentationForum.postings}
```

To get one particular posting:

```
${myPresentationForum.postings[index]}
```

where *index* is the index of the posting you want.

12.1.14 relatedForum

This forum's **related forum** (if it has one). Only **qa** forums can have a related forum. A related forum is a forum in which discussion can continue after a qa forum has become inactive.

Type: `com.escenic.forum.presentation.PresentationForum`

Example usage

```
${myPresentationForum.relatedForum}
```

12.1.15 relativeUrl

The relative URL of this forum.

Type: `java.lang.String`

Example usage

```
${myPresentationForum.relativeUrl}
```

12.1.16 sections

All the sections this forum belongs to.

Type: `neo.xredsys.api.Section`

Example usage

```
${myPresentationForum.sections}
```

12.1.17 threadCount

The total number of threads in this forum.

Type: `int`

Example usage

```
${myPresentationForum.threadCount}
```

12.1.18 threads

A list of all threads in this forum. The list is sorted by date, with the newest first.

Type: `java.util.Collection<PresentationThread>`

Example usage

To get all the threads:

```
${myPresentationForum.threads}
```

To get one particular thread:

```
${myPresentationForum.threads[index]}
```

where *index* is the index of the thread you want.

12.1.19 typeName

The type of this forum. Valid types are **`normal`** or **`qa`**.

Type: **`java.lang.String`**

Example usage

```
${myPresentationForum.typeName}
```

12.1.20 url

The absolute URL of this forum.

Type: **`java.lang.String`**

Example usage

```
${myPresentationForum.url}
```

12.2 PresentationThread

Represents a thread in an Escenic forum. A thread can contain postings (represented by **`PresentationPosting`** beans).

`PresentationThread` has the properties described in the following sections.

12.2.1 forum

The forum to which this thread belongs.

Type: **`com.escenic.forum.presentation.PresentationForum`**

Example usage

```
${myPresentationThread.forum}
```

12.2.2 postingCount

The total number of postings in this thread.

Type: **`int`**

Example usage

```
${myPresentationThread.postingCount}
```

12.2.3 postings

A list of all postings in this thread. The list is sorted by date, with the newest first.

Type: `java.util.List<PresentationPosting>`

Example usage

To get all the postings:

```
${myPresentationThread.postings}
```

To get one particular posting:

```
${myPresentationThread.postings[index]}
```

where *index* is the index of the posting you want.

12.2.4 root

The root posting of this thread.

Type: `com.escenic.forum.presentation.PresentationPosting`

Example usage

```
${myPresentationThread.root}
```

12.2.5 title

The title of this thread.

Type: `java.lang.String`

Example usage

```
${myPresentationThread.title}
```

12.3 PresentationPosting

Represents a posting in an Escenic forum.

`PresentationPosting` has the properties described in the following sections.

12.3.1 articleTypeName

The name of the content type on which this posting is based.

Type: `java.lang.String`

Example usage

```
${myPresentationPosting.articleTypeName}
```

12.3.2 authors

All the authors of this posting.

Type: `java.util.List<neo.xredsys.api.Person>`

Example usage

To get all the authors:

```
${myPresentationPosting.authors}
```

To get one particular author:

```
${myPresentationPosting.authors[index]}
```

where *index* is the index of the author you want.

12.3.3 body

The body of the posting.

Type: `java.lang.String`

Example usage

```
${myPresentationPosting.body}
```

12.3.4 customFields

The custom fields of this posting.

Type: `java.util.Map<String, String>`

Example usage

To get all the customFields:

```
${myPresentationPosting.customFields}
```

To get one particular customField:

```
${myPresentationPosting.customFields.customField}
```

where *customField* is the key of the customField you want.

12.3.5 email

The email address of the person who submitted the posting.

Type: `java.lang.String`

Example usage

```
${myPresentationPosting.email}
```

12.3.6 fields

The fields of this posting.

Type: `java.util.Map<String,PresentationProperty>`

Example usage

To get all the fields:

```
${myPresentationPosting.fields}
```

To get one particular field:

```
${myPresentationPosting.fields.field}
```

where *field* is the key of the field you want.

12.3.7 forum

The forum to which this posting belongs.

Type: `com.escenic.forum.presentation.PresentationForum`

Example usage

```
${myPresentationPosting.forum}
```

12.3.8 modified

If `true`, then the moderator has modified this posting. If `false`, then the posting is unmodified.

Type: `boolean`

Example usage

```
${myPresentationPosting.modified}
```

12.3.9 parent

This posting's parent posting: `null` if this posting has no parent.

Type: `com.escenic.forum.presentation.PresentationPosting`

Example usage

```
${myPresentationPosting.parent}
```

12.3.10 relativeUrl

The relative URL of this posting.

Type: `java.lang.String`

Example usage

```
${myPresentationPosting.relativeUrl}
```

12.3.11 replies

All the replies to this posting. The list is sorted by date, with the oldest first.

Type: `java.util.List<PresentationPosting>`

Example usage

To get all the postings:

```
${myPresentationPosting.replies}
```

To get one particular posting:

```
${myPresentationPosting.replies[index]}
```

where *index* is the index of the posting you want.

12.3.12 repliesCount

The number of direct replies made to this posting.

Type: `int`

Example usage

```
${myPresentationPosting.repliesCount}
```

12.3.13 root

If **`true`**, then this posting is the root of a thread. If **`false`**, then it is not.

Type: `boolean`

Example usage

```
${myPresentationPosting.root}
```

12.3.14 thread

The thread to which this posting belongs.

Type: `com.escenic.forum.presentation.PresentationThread`

Example usage

```
${myPresentationPosting.thread}
```

12.3.15 title

The title of the posting.

Type: `java.lang.String`

Example usage

```
${myPresentationPosting.title}
```

12.3.16 type

The type of this posting. Valid types are **`normal`**, **`question`**, **`answer`** or **`complaint`**.

Type: `PostingType`

Example usage

```
${myPresentationPosting.type}
```

12.3.17 url

The absolute URL of this posting.

Type: `java.lang.String`

Example usage

```
${myPresentationPosting.url}
```

13 escenic-forum-syndication

The **escenic-forum-syndication** schema defines a set of additional elements that can be used to include Forum postings in an Escenic Content Engine syndication file, thus enabling Forum postings to be imported and exported in the same way as other publication content. **escenic-forum-syndication** schema elements are identified within a syndication file by the namespace to which they belong (**http://xmlns.escenic.com/2010/forum-import**).

The following example shows a syndication file containing one forum content item (the forum itself) and one very simple posting in that forum:

```
<escenic version="2.0"
    xmlns="http://xmlns.escenic.com/2009/import"
    xmlns:forum="http://xmlns.escenic.com/2010/forum-import"
    >
  <content type="forum" state="published" source="MyphpBBForum" sourceid="1">
    <field name="title">My Forum</field>
  </content>
  <forum:posting source="MyphpBBForum" sourceId="2">
    <forum:body>My first posting.</forum:body>
    <forum:forum-ref source="MyphpBBForum" sourceId="1"/>
  </forum:posting>
</escenic>
```

This example shows a syndication file containing a more complex posting element that makes use of most of the **escenic-forum-syndication** schema elements:

```
<escenic version="2.0"
  xmlns="http://xmlns.escenic.com/2009/import"
  xmlns:forum="http://xmlns.escenic.com/2010/forum-import"
  >
  <forum:posting
      creationDate="2010-04-01 13:01:02"
      email="user123@domain.com"
      lastModified="2010-04-01 13:01:02"
      moderationState="approved"
      publicationId="1"
      type="normal"
      ip="221.120.101.130"
      source="ece-forum"
      sourceId="2345"
      userId="2"
      pingbackURI="http://www.some-host.com"
      >
    <forum:thread-ref source="ece-forum" sourceId="12"/>
    <forum:parent-ref source="ece-forum" sourceId="15"/>
    <forum:forum-ref source="ece-content" sourceId="123"/>
    <forum:commented-article-ref source="ece-content" sourceId='11'/>
    <forum:title>This is posting Title</title>
    <forum:body><![CDATA[This is posting Body]]></body>
    <forum:custom-fields>
      <forum:custom-field name="posting-user-karma-level">
        Gold
      </forum:custom-field>
    </forum:custom-fields>
  </forum:posting>
```

```
</escenic>
```

For a description of the Escenic Content Engine syndication file format, see the **Escenic Content Engine Syndication Reference**.

Namespace URI

The namespace URI of the **escenic-forum-syndication** schema is **http://xmlns.escenic.com/2010/forum-import**.

13.1 body

The **body** field of this body.

Syntax

```
<body>
  text
</body>
```

13.2 commented-article-ref

This element references an article to which a comment **posting** is to be added.

Syntax

```
<commented-article-ref
    dbid="positiveInteger"?
    (source="text" sourceId="text")?
  />
```

Attributes

dbid="*positiveInteger*" (optional)
The **dbid** of the referenced article (content item) on which a comment has been made. If this attribute is specified, then one of the following two conditions must be satisfied:

- The target publication must already contain a content item with a **dbid** attribute that matches this attribute, **or**
- A **content** element with a **dbid** attribute that matches this attribute must appear somewhere **before** this **commented-article-ref** element in the syndication file.

source="*text*" (optional)
The **source** of the referenced article (content item) on which a comment has been made. If this attribute is specified, then **sourceid** must also be specified. One of the following two conditions must be satisfied:

- The target publication must already contain a content item with **source** and **sourceid** attributes that match this element's **source** and **sourceid**, **or**

- A **content** element with **source** and **sourceid** attributes that match **source** and **sourceid** must appear somewhere **before** this **commented-article-ref** element in the syndication file.

If **dbid** is specified, then **source** and **sourceid** are ignored.

sourceId="*text*" (optional)
The **sourceid** of the referenced article (content item) on which a comment has been made. If this attribute is specified, then **source** must also be specified. One of the following two conditions must be satisfied:

- The target publication must already contain a content item with **source** and **sourceid** attributes that match this element's **source** and **sourceid**, **or**
- A **content** element with **source** and **sourceid** attributes that match **source** and **sourceid** must appear somewhere **before** this **commented-article-ref** element in the syndication file.

If **dbid** is specified, then **source** and **sourceid** are ignored.

13.3 custom-field

This element defines a custom custom-field field.

Syntax

```
<custom-field
    name="text"
  >
  text
</custom-field>
```

Attributes

name="*text*"
The name of the custom custom-field field. It must correspond to the name of an actual field defined in the target forum database.

13.4 custom-fields

This element is a container for a sequence of 1 or more **custom-field** elements, each of which defines a custom custom-fields field.

Syntax

```
<custom-fields>
  <custom-field>...</custom-field>*
</custom-fields>
```

13.5 forum-ref

This element references the forum content item to which a **posting** is to be added.

Syntax

```
<forum-ref
    dbid="positiveInteger"?
    (source="text" sourceId="text")?
  />
```

Attributes

dbid="*positiveInteger*" (optional)
The **dbid** of the referenced forum content item. If this attribute is specified, then one of the following two conditions must be satisfied:

- The target publication must already contain a forum content item with a **dbid** attribute that matches this attribute, **or**
- A **content** element with a **dbid** attribute that matches this attribute must appear somewhere **before** this **forum-ref** element in the syndication file.

source="*text*" (optional)
The **source** of the referenced forum content item. If this attribute is specified, then **sourceid** must also be specified. One of the following two conditions must be satisfied:

- The target publication must already contain a forum content item with **source** and **sourceid** attributes that match this element's **source** and **sourceid**, **or**
- A **content** element with **source** and **sourceid** attributes that match **source** and **sourceid** must appear somewhere **before** this **forum-ref** element in the syndication file.

If **dbid** is specified, then **source** and **sourceid** are ignored.

sourceId="*text*" (optional)
The **sourceid** of the referenced forum content item. If this attribute is specified, then **source** must also be specified. One of the following two conditions must be satisfied:

- The target publication must already contain a forum content item with **source** and **sourceid** attributes that match this element's **source** and **sourceid**, **or**
- A **content** element with **source** and **sourceid** attributes that match **source** and **sourceid** must appear somewhere **before** this **forum-ref** element in the syndication file.

If **dbid** is specified, then **source** and **sourceid** are ignored.

13.6 parent-ref

This element references the parent-ref to which a parent-ref is a response (if any). This element is only required for parent-refs that are in fact responses.

Syntax

```
<parent-ref
    dbid="positiveInteger"?
```

```
    (source="text" sourceId="text")?
  />
```

Attributes

`dbid="`*positiveInteger*`"` (optional)
The **`dbid`** of the referenced parent parent-ref. If this attribute is specified, then one of the following two conditions must be satisfied:

- The target publication must already contain a parent-ref with a **`dbid`** attribute that matches this attribute, **or**
- A **`posting`** element with a **`dbid`** attribute that matches this attribute must appear somewhere **before** this **`parent-ref`** element in the syndication file.

`source="`*text*`"` (optional)
The **`source`** of the referenced parent parent-ref. If this attribute is specified, then **`sourceid`** must also be specified. One of the following two conditions must be satisfied:

- The target publication must already contain a parent parent-ref with **`source`** and **`sourceid`** attributes that match this element's **`source`** and **`sourceid`**, **or**
- A **`posting`** element with **`source`** and **`sourceid`** attributes that match **`source`** and **`sourceid`** must appear somewhere **before** this **`parent-ref`** element in the syndication file.

If **`dbid`** is specified, then **`source`** and **`sourceid`** are ignored.

`sourceId="`*text*`"` (optional)
The **`source`** of the referenced parent parent-ref. If this attribute is specified, then **`sourceid`** must also be specified. One of the following two conditions must be satisfied:

- The target publication must already contain a parent parent-ref with **`source`** and **`sourceid`** attributes that match this element's **`source`** and **`sourceid`**, **or**
- A **`posting`** element with **`source`** and **`sourceid`** attributes that match **`source`** and **`sourceid`** must appear somewhere **before** this **`parent-ref`** element in the syndication file.

If **`dbid`** is specified, then **`source`** and **`sourceid`** are ignored.

13.7 posting

Represents a Forum posting.

Syntax

```
<posting
    type="(normal|complaint|answer|question|pingback)"?
    email="text"?
    ip="text"?
    moderationState="(not-reviewed|reviewed|approved|pending|rejected)"?
    userId="text"?
    dbid="positiveInteger"?
    (source="text" sourceId="text")?
    publicationId="positiveInteger"?
    creationDate="text"?
```

```
    lastModified="text"?
    pingbackURI="anyURI"?
  >
  <forum-ref/>

    <commented-article-ref/>?
    <thread-ref/>?
    <parent-ref/>?
    (<title>...</title>|<body>...</body>|<title>...</title> <body>...</body>)

    <custom-fields>...</custom-fields>?
</posting>
```

Attributes

`type="(normal|complaint|answer|question|pingback)"` (optional)
The type of this posting.

Allowed values are:

`normal` (default)
The posting is an ordinary posting of no special type.

`complaint`
The posting is a complaint about another posting.

`answer`
The posting is a web meeting answer (not currently supported).

`question`
The posting is a web meeting question (not currently supported).

`pingback`
The posting is a pingback.

`email="text"` (optional)
The email address of the author of this posting.

`ip="text"` (optional)
The IP address from which the posting originated.

`moderationState="(not-reviewed|reviewed|approved|pending|rejected)"` (optional)
The moderation state of this posting.

Allowed values are:

`not-reviewed` (default)
The posting has not been reviewed.

`reviewed`
The posting has been reviewed.

`approved`
The posting has been accepted by the moderator.

`pending`
The posting is pending.

`rejected`
The posting has been rejected by the moderator.

`userId="text"` (optional)
The user ID of the user who created the posting.

`dbid="`*positiveInteger*`"` (optional)
The internal database ID of this posting, which can be used when importing updated versions of existing postings. It can also be used for establishing relationships between elements in the syndication file. Other elements in the file have **`dbid`** attributes that can be used for this purpose. If a **`posting`** element does not have a **`dbid`** attribute then it must have a **`source`** and **`sourceid`** attribute. A **`posting`** element may have both a **`dbid`** attribute and a **`source`**/**`sourceid`** attribute pair, in which case either of them can be used for establishing relationships.

You should only use the **`dbid`** attribute when importing updated versions of **existing** postings.

`source="`*text*`"` (optional)
The name of the system from which this posting originates. Together with the **`sourceid`** attribute it forms a globally unique external identifier for the posting that can be used for establishing relationships between elements in the syndication file. Other elements in the file have **`source`** and **`sourceid`** attributes that can be used for this purpose. If this attribute is specified then a **`sourceid`** attribute must also be specified. If a **`posting`** element does not have a **`source`** and **`sourceid`** attribute then it must have a **`dbid`** attribute. A **`posting`** element may have both a **`source`**/**`sourceid`** attribute pair and a **`dbid`** attribute, in which case either of them can be used for establishing relationships.

If **`source`** and **`sourceid`** are supplied and **`dbid`** is not supplied, then they are used to lookup an existing posting. If a posting with matching **`source`** and **`sourceid`** is found, then this posting is updated; otherwise a new posting is created.

If supplied, **`source`** and **`sourceid`** are imported and stored when creating new postings, but not when updating existing postings.

`sourceId="`*text*`"` (optional)
The id of this posting in the system from which it originates. Together with the **`source`** attribute it forms a globally unique external identifier for the **`posting`** that can be used for establishing relationships between elements in the syndication file. Other elements in the file have **`source`** and **`sourceid`** attributes that can be used for this purpose. If this attribute is specified then a **`source`** attribute must also be specified. If a **`posting`** element does not have a **`source`** and **`sourceid`** attribute then it must have a **`dbid`**. A **`posting`** element may have both a **`source`**/**`sourceid`** attribute pair and a **`dbid`** attribute, in which case either of them can be used for establishing relationships.

If **`source`** and **`sourceid`** are supplied and **`dbid`** is not supplied, then they are used to lookup an existing posting. If a posting with matching **`source`** and **`sourceid`** is found, then this posting is updated; otherwise a new posting is created.

If supplied, **`source`** and **`sourceid`** are imported and stored when creating new postings, but not when updating existing postings.

`publicationId="`*positiveInteger*`"` (optional)
The ID of the Content Engine publication to which this posting belongs.

`creationDate="`*text*`"` (optional)
The date/time this posting item was created, specified in the format:

```
yyyy-mm-dd hh:mm:ss.ffffffff
```

If specified, this attribute is only used when importing a new posting that does not already exist in the database. It is ignored when updating a posting that already exists. If it is omitted when importing a new posting, then the new posting's creation date is set to the current date.

`lastModified="`*text*`"` (optional)
The date/time this posting was last modified, specified in the format:

```
yyyy-mm-dd hh:mm:ss.ffffffff
```

If this attribute is omitted when importing a posting, then the posting's last modified date is set to the current date.

pingbackURI="*anyURI*" (optional)
The pingback URI of this posting.

13.8 thread-ref

This element references the thread to which a **posting** is to be added. In most cases this element is not required and is not used during import. In the case of pingback thread-refs, however, it is required, and must reference a special pingback thread.

Syntax

```
<thread-ref
    dbid="positiveInteger"?
    (source="text" sourceId="text")?
  />
```

Attributes

dbid="*positiveInteger*" (optional)
The **dbid** of the referenced thread. If this attribute is specified, then the target forum database must already contain a thread with a **dbid** attribute that matches this attribute.

source="*text*" (optional)
The **source** of the referenced thread. If this attribute is specified, then **sourceid** must also be specified. In addition, the target forum database must already contain a thread with with **source** and **sourceid** attributes that match this element's **source** and **sourceid**.

If **dbid** is specified, then **source** and **sourceid** are ignored.

sourceId="*text*" (optional)
The **sourceid** of the referenced thread. If this attribute is specified, then **source** must also be specified. In addition, the target forum database must already contain a thread with with **source** and **sourceid** attributes that match this element's **source** and **sourceid**.

If **dbid** is specified, then **source** and **sourceid** are ignored.

13.9 title

The **title** field of this title.

Syntax

```
<title>
  text
</title>
```

14 Exporting Postings

You can export Forum postings from a publication using the Content Engine's administration web application, **escenic-admin**. You can access **escenic-admin** by starting a browser and pointing it at:

http://*your-server***:8080/escenic-admin/**

where *your-server* is the domain name or IP address of the server on which the Content Engine is running. To export postings from a publication, select the **Export publication content** option, which displays the **Export from publication** form. You must fill in the following fields in this form:

Publication ID
: Enter the ID of the publication from which you want to export postings.

Export content items
: Check this option.

Content types
: Enter **forum** in this field.

You may also fill in other fields if required (you might, for example, fill in the **Section IDs** field in order to limit the export to certain sections). For a full description of the **Export publication content**, see the **Escenic Content Engine Server Administration Guide**.

> This is the only way to export Forum postings. You cannot export postings using the Content Engine's export service.