Poll

Plug-in Guide

trunk-SNAPSHOT

escenic

A CCI COMPANY

Table of Contents

1 Introduction

The Poll plug-in allows content creators to easily add **polls** to Escenic publications. A **poll** consists of a question and a set of alternative answers. Publication readers can take part in interactive polls by selecting one of the answers.

An Escenic poll is a type of content item that contains a title, a question, and a series of possible answers. This makes it very easy for content creators to add polls to publications. To add a poll to a publication, a Content Studio user simply needs to:

1. Create a new poll content item.
2. Fill in the title, question and answer fields.
3. Save and publish the new content item.
4. Add the poll content item to the required section page, or add it to some other content item as related content.

The Poll plug-in provides:

- A set of Java beans that provide the basic polling functionality
- A JSP tag library containing tags that provide access to the poll beans
- A set of Struts actions and forms that simplify the process of building poll templates

In order to make use of this functionality, a template developer must create a set of one or more content types and associated templates that present polls in the required ways. Template developers can control:

- The overall look and feel of the polls in a publication
- Where polls appear in the publication
- Whether or not poll results are displayed with the poll or are hidden

> The Poll plug-in is intended for lightweight, single-question polls. It is not suitable for purposes where secure, certain results are required, since it is not secure enough. Although it does contain some basic protection against multiple voting, this protection is cookie-based and easily circumvented. It is also not really suitable for multiple-question surveys, since it does not include any functionality for processing the results.

1.1 Getting Started

In order to get started with the Poll plug-in in you need to:

1. Install the plug-in. For instructions on how to do this, see chapter 2.
2. Verify that the plug-in is correctly installed. For instructions on how to do this, see section 2.2.
3. Verify that the plug-in is working correctly by installing and trying out the poll-demo publication. For instructions on how to do this, see section 2.3 and section 2.4.

Once you have carried out these basic tasks and are sure that you have a fully-functioning plug-in, you can start adding polling functionality to your own publications. For instructions on how to do this, see chapter 3.

2 Installation

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

- The Content Engine is installed and in working order.
- The Escenic assembly tool has been extracted and successfully used to set up a test EAR file as described in the **Escenic Content Engine Installation Guide**.
- You have the required plug-in distribution file **`poll-dist-trunk-SNAPSHOT.zip`**.

2.1 Install Poll

> In the following description, *escenic-home* refers to the server folder in which the Content Engine is installed.

Installing Poll on the server involves the following steps:

1. **Make sure there is a plug-in folder:** If the folder *escenic-home*`/plugins` does not already exist on your server, create it. If for some reason you need to create the plug-in folder in some other location, edit the *escenic-home*`/assemblytool/assemble.properties` file and set the **`plugins`** property accordingly. For example:

   ```
   plugins=escenic-home/my/plugin/folder
   ```

   This folder will be referred to as *plugin-home* in the rest of this manual.
2. **Unpack the Poll distribution:** Unpack the Poll distribution file to *plugin-home*. This will result in the creation of a *plugin-home*`/poll` folder.
3. **Configure the plug-in:** If you want to configure the plug-in now, you can do so; alternatively, you can leave it until later. If you are installing the plug-in for testing or development purposes, then you can skip this step. If you are setting up a production system, then you may as well do it now, since otherwise you will have to rebuild and redeploy the Content Engine later. See section 2.5 for detailed instructions.
4. **Rebuild the Content Engine:** Build the Escenic enterprise archive by entering the following commands:

   ```
   cd escenic-home/assemblytool
   ant ear
   ```

   The assembly tool will then add the Poll plug-in to the Content Engine's classpath, including default configuration files and any required web application components.
5. **Deploy the Content Engine:** Deploy the new EAR file. For general instructions on how to deploy the EAR file on different application servers, see the **Escenic Content Engine Installation Guide**.
6. **Verify the plug-in installation:** See section 2.2 for details of how to verify plug-in installations.

> If the application server does not support EAR-based deployment, then all the JAR files located in the *plugin-home*`/poll/lib` folder must be added to the application server's classpath. All the WAR files that have been rebuilt with the assembly tool should be redeployed.

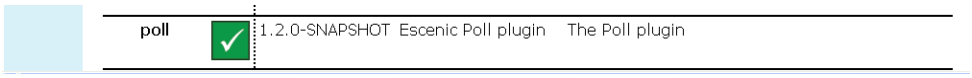
2.2 Verify The Installation

To verify the status of the Poll 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 Poll plug-in is correctly installed, you should see something like this in the displayed plug-in list:



You now know that the Poll plug-in is installed, but to really be sure that everything is working, you should install the demo publication **`poll-demo.war`** and check that it works correctly. For details, see section 2.3.

2.3 Install The Demo Publication

To install the demo publication:

1. Copy *escenic-home*`/plugins/poll/wars/poll-demo.war` to *escenic-home*`/assemblytool/publications/poll-demo.war`.
2. Create a `poll-demo.properties` file in the *escenic-home*`/assemblytool/publications` folder with the following contents:

   ```
   context-root=/poll-demo
   name=poll-demo
   source-war=poll-demo.war
   ```
3. Rebuild and deploy the Content Engine.
4. Restart the Content Engine.
5. Open the Escenic Admin web application (usually located at **`http://`***server*`/escenic-admin`).
6. Select **New Pubs** and use the displayed form to upload `poll-demo.war`.
7. Select **create a publication**.
8. Enter a name (`poll-demo`) and administrator password for the publication in the displayed form.
9. Select **Submit**.
10. A publication information page for the new publication should now be listed. To open the publication, click on the link under the heading **Browse the publication**.

The demo publication has no content, so all you will see when you open it is a W3C XHTML logo. For a description of how to test the demo publication, see section 2.4.

2.4 Test The Demo Publication

In order to be completely certain that the Poll plug-in is correctly installed and is working correctly, you can create a poll in the Poll demo publication as follows:

1. Start Content Studio and log in to the poll-demo publication using the administrator account you defined when creating the publication. If you called the publication **`poll-demo`**, then the administrator account name is **`poll-demo_admin`**. The password is whatever you entered when creating the publication.
2. Select **File** > **New** > **Poll** to create a poll content item.
3. Fill in the content item's **Title**, **Question** and **Alternative** fields.
4. Set **State** to **Published**.
5. Click on **Save**.
6. Click on **Properties** to display the **Properties** attribute ribbon.
7. Click on the **URL** link displayed in this attribute ribbon. This should display the poll you have created in your browser.
8. Click on one of the alternatives displayed in the poll. This should increment the result count displayed next to the alternative.

> The poll will not work properly if you display it using the Content Studio **Preview** function. You must use the **URL** link to display the content item.

2.5 Configure The Poll Plug-in

Configuring the Poll plug-in involves a couple of simple tasks, described in the following sections. In these instructions, the placeholder *escenic-config* is used to represent the path of your Escenic configuration, as defined with the **`com.escenic.config`** property in *escenic-home*`/assemblytool/assemble.properties`. If **`com.escenic.config`** is not defined, then *escenic-config* has a default definition of *escenic-home*`/localconfig`.

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

1. If the *escenic-config*`/com/escenic/poll` folder does not exist, create it.
2. If the *escenic-config*`/com/escenic/poll` folder does not contain **`MentometerManager.properties`** and **`PollSchedule.properties`** files, copy them from *escenic-home*`/engine/plugins/poll/misc/siteconfig/com/escenic/poll`.
3. Edit **`MentometerManager.properties`** and **`PollSchedule.properties`** to meet your requirements, as described in the following sections.
4. Use the assembly tool to reassemble the Content Engine.
5. Redeploy the Content Engine.

2.5.1 Set MentometerManager Properties

The configuration file *escenic-config*`/com/escenic/poll/MentometerManager.properties` can be used to set the following properties:

checkCookieEnabled

This property is set to **false** by default, which means that all poll visitors can vote an unlimited number of times. If you set **checkCookieEnabled** to **true**, then:

- Visitors are only allowed to vote if they have cookies enabled in their browser.
- Visitors are only allowed to vote once in any one poll (assuming that the template code checks the **mode** request attribute - see section 3.3 for details).

It may be useful to leave this property set to **false** during development, but for production purposes you will usually want to set it to **true**.

To set this property to true, add the following line to **MentometerManager.properties**:

```
checkCookieEnabled=true
```

fieldNameAnswer

You can use this property to set the prefix to be used for poll alternative field names. By default it is set to **svar**, so that the alternative fields in your poll content types must be called **svar1**, **svar2**, **svar3** etc. (see section 3.2). If you want to set the prefix to something that is more meaningful to you (**alt**, for example), add the following line to **MentometerManager.properties**:

```
fieldNameAnswer=alt
```

fieldNameSum

You can use this property to set the name of the article field for saving the sum of the answers. By default it is set to **sum**. If you want to set the field name to something that is more meaningful to you (**total**, for example), add the following line to **MentometerManager.properties**:

```
fieldNameSum=total
```

fieldNameCorrectAnswer

You can use this property to set the name of the article field for holding the correct answer for use in a competition. By default it is set to **correctAnswer**. If you want to set the field name to something that is more meaningful to you (**rightAnswer**, for example), add the following line to **MentometerManager.properties**:

```
fieldNameCorrectAnswer=rightAnswer
```

howManyAlternatives

You can use this property to set the maximum number of poll alternative field names. The prefix of this field names should be **fieldNameAnswer** and the suffix for those should start with a number from 1 to the value of **howManyAlternatives**. By default it is set to **20**. If you want to set another value (**10**, for example), add the following line to **MentometerManager.properties**:

```
howManyAlternatives=10
```

If *escenic-config*/com/escenic/poll/MentometerManager.properties does not already exist, then copy **MentometerManager.properties** from *escenic-home*/engine/plugins/poll/misc/siteconfig/com/escenic/poll to *escenic-config*/com/escenic/poll/ and set the required properties.

2.5.2 Set PollSchedule Properties

The configuration file *escenic-config*/com/escenic/poll/PollSchedule.properties can be used to set the following properties:

serviceEnabled
This property is set to **false** by default, which means that poll-related data is not stored in the database. This means that all poll-related data will be lost whenever the Escenic server is restarted. For production purposes, therefore, you must set this property to **true**. Note , however, that In a multi-server environment it must only be set to true on **one** server.

To set this property to true, add the following line to **PollSchedule.properties**:

```
serviceEnabled=true
```

If *escenic-config*/com/escenic/poll/PollSchedule.properties does not already exist, then copy **PollSchedule.properties** from *escenic-home*/engine/plugins/poll/misc/siteconfig/com/escenic/poll to *escenic-config*/com/escenic/poll/ and set the required properties.

2.5.3 Set PollRedirectUrlValidator and SubmitActionHelper Properties

The configuration files **PollRedirectUrlValidator.properties** and **SubmitActionHelper.properties** are used to configure a whitelist of valid redirection targets for the Poll plug-in, in order to protect end-users from phishing attacks in which they are redirected to a malicious web site.

The configuration file *escenic-config*/com/escenic/poll/PollRedirectUrlValidator.properties has the following property:

validRedirectUrlHosts
A comma-separated list of host names that the Poll plug-in is allowed to redirect to. For example:

```
validRedirectUrlHosts=escenic.com,ccieurope.com
```

If *escenic-config*/com/escenic/poll/PollRedirectUrlValidator.properties does not already exist, then copy **PollRedirectUrlValidator.properties** from *escenic-home*/engine/plugins/poll/misc/siteconfig/com/escenic/poll to *escenic-config*/com/escenic/poll/ and set the above property.

The configuration file *escenic-config*/com/escenic/poll/SubmitActionHelper.properties can contain one or more properties like this:

redirectUrlValidator.*publication-name*
where *publication-name* is the name of one of your publications. Create one copy of this property for each publication that makes use of the Poll plug-in and set it to point to your redirect URL validator component. For example:

```
redirectUrlValidator.dailynews=./PollRedirectUrlValidator
redirectUrlValidator.sportsweekly=./PollRedirectUrlValidator
```

If *escenic-config*/com/escenic/poll/SubmitActionHelper.properties does not already exist, then copy **SubmitActionHelper.properties** from *escenic-home*/engine/plugins/poll/misc/siteconfig/com/escenic/poll to *escenic-config*/com/escenic/poll/ and set the above properties.

3 Adding Polling To a Publication

To add polling functionality to a publication, you need to:

- Add Struts configuration information to the publication's **WEB-INF** folder.
- Create a poll content type (or add polling functionality to an existing content type).
- Create templates for displaying the poll questions and results in the publication.

Once you have done this you should be able to create poll content items

3.1 Struts Configuration

The Poll plug-in uses the Apache Struts framework to manage the polling 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:

```
<!-- 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</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:

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
The path of the Struts configuration file, described below.

Several Escenic plug-ins use Struts, so the publication **web.xml** file may already contain 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>
  <form-beans>
    <form-bean
      name="pollForm"
      type="com.escenic.poll.struts.SubmitForm"
    />
  </form-beans>
  <action-mappings>
    <action
      path="/poll/vote"
      type="com.escenic.poll.struts.HandleSubmitAction"
      name="pollForm"
      scope="request"
    >
      <forward
        name="success"
        path="success.jsp"
        redirect="true"
      />
      <forward
        name="error"
        path="error.jsp"
        redirect="true"
      />
      <forward
        name="invalid-redirect"
        path="confirm_redirect.jsp"
        redirect="true"
      />
    </action>
  </action-mappings>
</struts-config>
```

If the **WEB-INF** folder **does** already contain a **struts-config.xml** file, then add the highlighted sections of the above content to it, in the appropriate locations. Insert the **form-bean** element as a

child of the existing **form-beans** element, and insert the **action** element as a child of the existing **action-mappings** element.

3.2 Creating a Poll Content Type

To create a poll content type, you must edit your publication's **content-type** resource, which is located in the publication's **META-INF/escenic/publication-resources** folder. The following example shows a simple poll content type similar to the one used in the **poll-demo** publication. It defines a field group containing a title field, a question field and three alternative answer fields. It then defines a content type which places this group of fields in the default panel.

```
<field-group name="poll">
  <field mime-type="text/plain" type="basic" name="title">
    <ui:label>Title</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="question">
    <ui:label>Question</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="svar1">
    <ui:label>Alternative 1</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="svar2">
    <ui:label>Alternative 2</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="svar3">
    <ui:label>Alternative 3</ui:label>
  </field>
</field-group>

<content-type name="poll">
  <panel name="default">
    <ref-field-group name="poll"/>
  </panel>
</content-type>
```

The result of adding this code to the **content-type** resource will be that a new content type will be added to the publication and made available to Content Studio users. Note the following:

- The above code is an example and deliberately kept short. A real polling content type would be likely to have a much larger number of alternative fields, in order to give Content Studio users the possibility of creating polls with many alternatives when necessary.
- You can give the field containing the poll question any name you like.
- The alternative fields, on the other hand, must (at least by default) have names starting with the prefix **svar**. You can change this prefix to something else by editing the configuration file **MentometerManager.properties**. For further information about this, see section 2.5.1.

The above content type definition will produce content items that looks like this in Content Studio:

3.2.1 Displaying Poll Results in Content Studio

You can very easily get Content Studio to display the results of polls by adding a single field to your **poll** content type. The field you add must have the following characteristic:

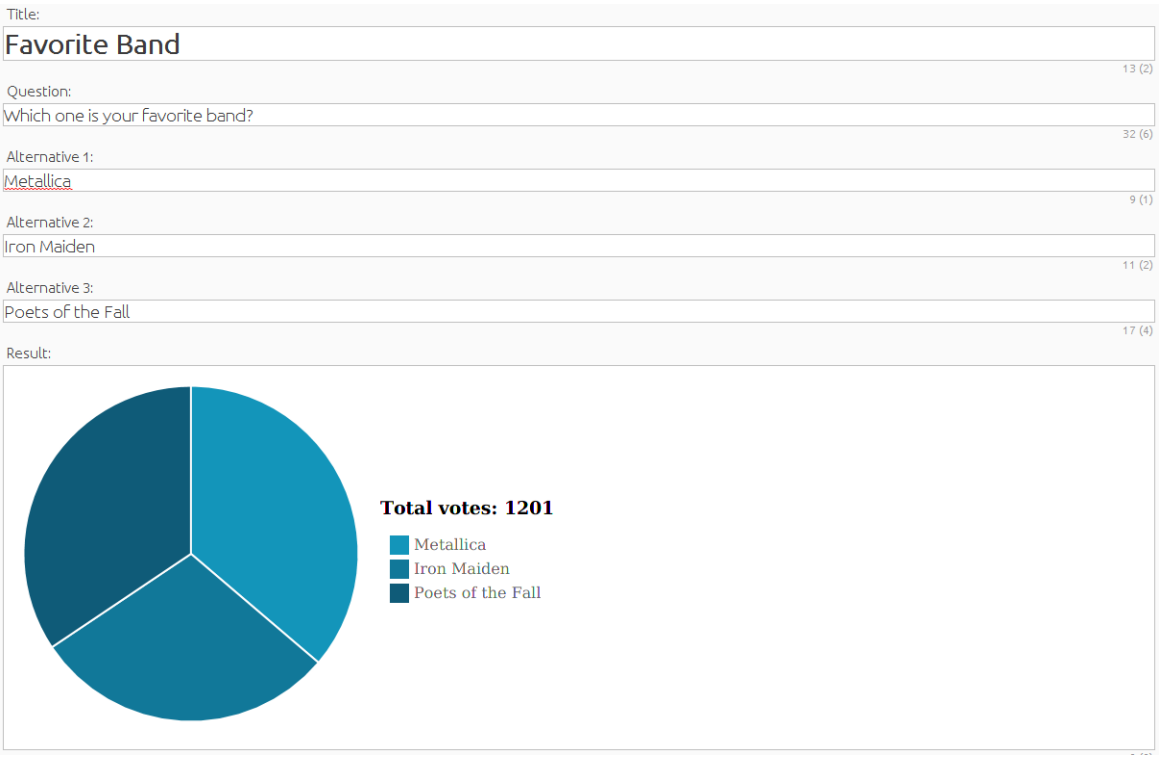
- It must be a **basic** field
- It must have a **mime-type** of **application/xhmtl+xml**
- It must have a **result** sub-element with an **enabled** attribute value set to **true**. This element must belong to the namespace **http://xmlns.escenic.com/2014/poll**.

The following example shows such a **poll** with a result field (highlighted):

```
<field-group name="poll">
  <field mime-type="text/plain" type="basic" name="title">
    <ui:label>Title</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="question">
    <ui:label>Question</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="svar1">
    <ui:label>Alternative 1</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="svar2">
    <ui:label>Alternative 2</ui:label>
  </field>
  <field mime-type="text/plain" type="basic" name="svar3">
    <ui:label>Alternative 3</ui:label>
  </field>
  <field name="pollresult" type="basic" mime-type="application/xhmtl+xml">
    <poll:result enabled="true" />
  </field>
</field-group>

<content-type name="poll">
  <panel name="default">
    <ref-field-group name="poll"/>
  </panel>
</content-type>
```

This is what the resulting content items look like in Content Studio:



3.3 Creating Poll Templates

The following very simple examples are based on the templates in the **poll-demo** publication.

Here is the main content item template, called **art_default.jsp**. All it does is check the type (**articleTypeName**) of the current content item, and if it is **poll**, transfers control to **render-poll.jsp**. In a real publication, of course, this template would also check for other content types and transfer control to other templates as well.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<html xmlns="http://www.w3.org/1999/xhtml" lang="en" xml:lang="en">
 <head>
  <title>${publication.name}</title>
 </head>

 <body>
  <h1>${article.title}</h1>
  <c:if test="${article.articleTypeName == 'poll'}">
   <jsp:include page="render-poll.jsp"/>
  </c:if>
 </body>
</html>
```

Here is the content of **render-poll.jsp**, which actually displays the polls:

```
<%@ taglib prefix="poll" uri="http://www.escenic.com/taglib/escenic-poll" %>
<%@ taglib prefix="html" uri="http://jakarta.apache.org/struts/tags-html" %>
```

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"  %>

<poll:use id="poll">
  <span>${article.fields.question}</span>
  <c:choose>
    <c:when test="${mode eq 'vote'}">
      <html:xhtml />
      <html:form action="/poll/vote">
        <ul>
          <html:hidden property="mentometerId" value="${article.id}"/>
          <html:hidden property="publicationId" value="${publication.id}"/>
          <html:hidden property="redirectTo" value="${article.url}"/>
          <c:forEach items="${poll.mentometerOption}" var="option">
            <li>
              <html:radio property="vote" value="${option.articleElement}"/>
                ${option.title} - ${option.votes}
            </li>
          </c:forEach>
          <html:submit/>
        </ul>
      </html:form>
    </c:when>
    <c:when test="${mode eq 'voted'}">
      <ul>
        <c:forEach items="${poll.mentometerOption}" var="option">
          <li>${option.title} - ${option.votes}</li>
        </c:forEach>
      </ul>
    </c:when>
  </c:choose>
</poll:use>
```

If you want to submit poll using ajax then you have to add another hidden property named **forward** in the poll form and have to set its value to **false**. This parameter can be set as following :

```
<html:hidden property="forward" value="false" />
```

If this parameter is present and value is set to **false** then poll action will not forward to any URI or path. If vote cast is done poll action will response **200** status code. On the other hand if user is already voted on that poll then poll action will response **403** status code.

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

article.fields.question
This simply displays the **question** field of the poll content item.

poll:use
This tag is provided with the Poll plug-in, in the **escenic-poll** tag library. For detailed information, see chapter 4. It creates a **Mentometer** bean (see section 5.1) based on the current content item and makes it the current bean. The **Mentometer** bean is also provided with the Poll plug-in, and represents a poll. It contains an array of **MentometerOption** beans (see section 5.2), one for each alternative (**svar**) field in the content item. The name of the **Mentometer** bean is set with the **poll:use** tag's **id** attribute, so in this case the bean is called **poll**.

mode eq 'vote'
The request attribute **mode** can be used to prevent poll visitors voting more than once: it may be set either to **vote** (the visitor is allowed to vote) or **voted** (the visitor has already voted

and may not vote again). This template allows visitors who are not allowed to vote to see the current results. The **mode** attribute will only ever be set to **voted** if you configure the Poll plug-in to check for multiple voting. To do this you must set the **checkCookieEnabled** property in **MentometerManager.properties** to **true** (it is set to false by **default**). For instructions on how to do this, see section 2.5.1.

html:form
This is a Struts tag used to handle the display of action forms. The fields in the form are automatically linked to current bean (in this case the **Mentometer** bean created with the **poll:use** tag). The fields in the form can then be linked to the bean's properties by means of the Struts tags' **property** attributes (see **memtometerId** below). The action to be executed when the form is submitted is set to **/poll/vote** - the Struts action defined in **struts-config.xml** (see section 3.1.2 for details).

mentometerId, publicationId, redirectTo
These action form properties are set by defining hidden form fields and using the **html:hidden** tag's **property** attribute to associate them with the correct properties. The **mentometerId** and **publicationId** properties must be set to the current content item and publication IDs. **redirectTo** determines what page is displayed after the visitor has submitted a vote. In this case it is set to the URL of the current content item (meaning the same page is redisplayed).

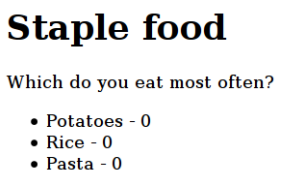
poll.mentometerOption
The **Mentometer** bean's **mentometerOption** property is a list of **MentometerOption** beans, one for each alternative answer. The **forEach** element here cycles through all of these beans, assigning each one in turn to a variable called **option**.

html:radio
This Struts tag displays a radio button and associates it with the action form's **vote** property, so that when the user selects a radio button and submits the form, a vote is cast for the selected alternative.

mode eq 'voted'
If the user is not allowed to vote, then the alternatives and current votes are displayed, but no radio buttons or submit button.

This template will produce output that looks like this for users who are allowed to vote:

or like this, for users who have already voted:



3.4 Poll REST service for vote submission

Using poll REST voting service you can submit a poll. The vote should be submitted to the following URI.

http://**your-host**/poll-ws/poll/**poll-id**/vote

In above URI pattern you should use your host name and a valid poll id. This service requires only one form parameter named **`option`** which is the voting option. If poll is found and voting option is valid then vote will be submitted successfully and REST service will return HTTP status code 200. If poll is not found or voting option is invalid then it will return 404 status code.

> Here voting restriction is not handled. Your REST client should handle that.

3.5 Poll REST service to fetch poll result

A REST service is available with poll plugin that will provide poll result details as **`JSON`** format. With poll plugin a web application named **poll-ws.war** is provided that contains the REST services. To get result for a specific poll you have to send request to the following URI.

http://**your-host**/poll-ws/poll/**poll-id**/status

In the above example you must use your host name where poll web service application is deployed and also must use a valid poll Id. For malformed Id like **55Y** REST service will return 400 status code. If poll is not found then 404 status will be returned. If poll Id is valid and content is found then it will return poll title, other fields and options in the json format. An example of **JSON** response is given bellow -

```
{
  "id": "2742",
  "title": "Euro 2012 quiz",
   "fields": {
      "description": "Count your vote",
      "question": "Who will win Euro 2012 ?"
    },
  "options": [
    {
      "option": svar1,
      "value": "Spain",
      "vote": 2,
      "percentage": "25"
    },
    {
      "option": svar2,
      "value": "Germany",
      "vote": 4,
      "percentage": "50"
    },
    {
      "option": svar3,
      "value": "Portugal",
      "vote": 1,
```

```
            "percentage": "12"
          },
          {
            "option": svar4,
            "value": "Italy",
            "vote": 1,
            "percentage": "12"
          }
        ]
      }
```

4 Tag Library Reference

The Poll plug-in includes a **poll** tag library that provides easy access to the poll beans **Mentometer** and **MentometerOption**.

4.1 poll Tag Library

This library contains tags that operate on **Mentometer** and **MentometerOption** beans. A **Mentometer** bean represents a poll, and a **MentometerOption** bean represents one of the alternative answers in a poll.

In order to use any **poll** tags in a JSP file you must include the following prefix declaration in the file:

```
<%@ taglib uri="/WEB-INF/escenic-poll.tld" prefix="poll" %>
```

article is the prefix normally used for this tag library.

4.1.1 poll:use

Creates a poll (**Mentometer** bean) and makes it available in the body of this tag. It can be accessed within the body of the tag via the name specified with the **id** attribute. The **Mentometer** bean is based on the current content item. You can, however, reset the current content item within the body of the tag by setting one of the following attributes or attribute combinations:

- **article**
- **articleId**
- **name**
- **name** and **property**
- **source** and **sourceId**

Syntax

```
<poll:use
   article="..."?
   articleId="..."?
   id="..."
   name="..."?
   property="..."?
   source="..."?
   sourceId="..."?>
   ...
</poll:use>
```

Attributes

id, mandatory, no runtime expressions
Name of the **Mentometer** (poll) bean created.

article
The content item to be set as current content item in the body of this tag. The supplied bean must either be a **neo.xredsys.api.Article** or a **neo.xredsys.presentation.PresentationArticle** bean.

Using this attribute excludes the use of the attributes **articleId**, **name**/**property** and **source**/**sourceId**.

articleId
An article id identifying the content item to be set as current content item in the body of this tag. This content item will be loaded from the current publication. The value supplied can be an **int**, **Integer** or **String**.

Using this attribute excludes the use of the attributes **article**, **name**/**property** and **source**/**sourceId**.

name
The name (key) of either:

Using this attribute excludes the use of the attributes **article**, **articleId** and **source**/**sourceId**.

property
The name of a bean property. This attribute is used together with the **name** attribute to locate a content item that will be used as current content item in the body of this tag.

This attribute cannot be used without the **name** attribute. Using this attribute excludes the use of the attributes **article**, **articleId** and **source**/**sourceId**.

source
The source of a content item to be used as current content item in the body of this tag.

This attribute must be used together with **sourceId**. Using this attribute excludes the use of the attributes **article**, **articleId** and **name**/**property**.

sourceId
The source ID of a content item to be used as current content item in the body of this tag.

This attribute must be used together with **source**. Using this attribute excludes the use of the attributes **article**, **articleId** and **name**/**property**.

4.1.2 poll:option

Syntax

```
<poll:option
   id="..."/>
```

Attributes

id, mandatory, no runtime expressions

5 Bean Reference

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

5.1 Mentometer

Represents one poll, with a set of **`MentometerOption`** (see section 5.2) . **`Mentometer`** beans are automatically created when required.

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

5.1.1 articleID

The id of the content item that contains the poll.

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

Example usage

```
${myMentometer.articleID}
```

5.1.2 checkCookieEnabled

Determines whether or not the Poll plug-in is to check whether cookies are enabled in visitors' browsers. If this property is set to **`true`**, then voting is only allowed when cookies are enabled in the visitor's browser. The default setting is **`true`**.

Type: **`boolean`**

Example usage

```
${myMentometer.checkCookieEnabled}
```

5.1.3 description

The description of the poll, set in the content item. It could contain something like: "Choose your movie of the week".

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

Example usage

```
${myMentometer.description}
```

5.1.4 mentometerOption

A list of alternatives from which the visitor can choose.

Type: **`com.escenic.poll.MentometerOption<mentometerOption>`**

Example usage

To get all the mentometer options:

```
${myMentometer.mentometerOption}
```

To get one particular mentometer option:

```
${myMentometer.mentometerOption[index]}
```

where *index* is the index of the mentometer option you want.

5.1.5 publicationID

The id of the publication to which the poll belongs.

Type: `java.lang.String`

Example usage

```
${myMentometer.publicationID}
```

5.1.6 title

The title of the poll ("Movie of the week", for example), used when displaying the poll. By default, this property takes its value from the poll content item's **`title`** property. (The **`title`** property of a content item is the field defined as the title field in the content-type resource.)

Type: `java.lang.String`

Example usage

```
${myMentometer.title}
```

5.1.7 totalVotes

The total number of votes cast in this poll (read-only).

Type: `int`

Example usage

```
${myMentometer.totalVotes}
```

5.2 MentometerOption

Represents one of the alternatives presented in a **`Mentometer`** (see section 5.1) (poll).

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

5.2.1 articleElement

The name of the element in the underlying article. This is typically used when you want to display images with the option.

Type: `java.lang.String`

Example usage

```
${myMentometerOption.articleElement}
```

5.2.2 mentometer

This option's parent **`Mentometer`** (see section 5.1) .

Type: `com.escenic.poll.Mentometer`

Example usage

```
${myMentometerOption.mentometer}
```

5.2.3 percentage

The percentage of votes cast for this alternative.

Type: `java.lang.String`

Example usage

```
${myMentometerOption.percentage}
```

5.2.4 title

The title of this option.

Type: `java.lang.String`

Example usage

```
${myMentometerOption.title}
```

5.2.5 votes

The number of votes cast for this option.

Type: `int`

Example usage

```
${myMentometerOption.votes}
```

6 Struts Component Reference

The Poll plug-in includes the following Struts components. They are configured in the **`/WEB-INF/struts-config.xml`** file (see section 3.1). For a proper introduction to Struts, see http://struts.apache.org/primer.html.

6.1 SubmitForm

`SubmitForm` inherits properties from:

- **`ActionForm`**

It has no properties of its own.

6.2 HandleSubmitAction

Accepts a vote, and adds it to the correct poll and option.

Struts config

```
<action path="/poll/vote"
 type="com.escenic.poll.struts.HandleSubmitAction"
 name="submitForm"
 input="/poll/vote-input.jsp"
 validate="true">

 <forward name="success"
       path="/poll/vote-success.jsp"/>

 <forward name="error"
       path="/poll/vote-error.jsp"/>

 <forward name="invalid-redirect"
        path="/poll/vote-confirm-redirect.jsp"/>
</action>
```

Action Form

This action can be used with the following form:

`com.escenic.poll.struts.SubmitForm`
Gathers visitors' votes.

Action Forwards

The forwarding actions defined in this action.

`success`
Performed if the vote is successful. A vote can be successful even if the visitor has voted before. This forwarding action can be overridden from the HTML form by setting the 'redirectTo' property.

`error`
Performed if there is a configuration error.

`invalid-redirect`
Performed if the **`redirectTo`** url is not in the whitelist. The redirect URL contains a **`redirectTo`** query parameter.