Simple Forms (2.1 legacy document)

Table of contents

Table of contents

Warning:
This document was copied as is from the Cocoon 2.1 documentation, but has not yet been fully reviewed or moved to its new home.

1. Introduction

This chapter describes the (simple) form handling approach of the session framework.

Cocoon provides several approaches for form handling. This chapter explains one of them.

2. Form Handling

To get feedback or information from a user, forms are commonly used to present input field in the browser. The usual approach for form handling in web application consists of two steps. The first request presents the form to the user. This form initiates a second request that processes the form values.

Cocoon supports this two step process, in addition Cocoon offers a single step approach.

2.1. The common approach

The common approach consists of two steps or of creating two resources. The first resource defines the form: All input fields are declared, each gets a unique name. This form invokes the second resource.

This resource uses the session transformer to get the values provided by the user. The values are added by the browser to the parameters of the request. So using the request context and *getxml*, the values can be fetched.

If you want to create a form with two values - forename and surname of the user, you could generate a base xml file with the information about this form:

```
<page> <form> <action>form-handling-page</action> <input name="forename" type="text"/>
<input name="surname" type="text"/> </form> </page>
```

A stylesheet can transform this into valid html. The action tag indicates that the "form-handling-page" should be invoked by submitting the values.

The "form-handling-page" is a pipeline which is declared in the sitemap and uses the session transformer. It could also read the following xml:

```
<page xmlns:session="http://apache.org/cocoon/session/1.0"> <forename> <session:getxml
context="request" path="/parameter/forename"/> </forename> <surname> <session:getxml
context="request" path="/parameter/surname"/> </surname> </page>
```

As the form values are appended to the request, *getxml* with specifying the path (which is the parameter name used for the input field) inserts the value submitted by the user into the xml stream.

If you want to write the information in a session context, you must wrap the whole xml inside a setxml:

```
<page xmlns:session="http://apache.org/cocoon/session/1.0"> <session:setxml
context="userdata" path="/user"> <forename> <session:getxml context="request"
path="/parameter/forename"/> </forename> <surname> <session:getxml context="request"
path="/parameter/surname"/> </surname> </session:setxml> </page>
```

The user data is now stored inside the session context "userdata", so the context has the following content:

<user> <forename>Walter</forename> <surname>Walterson</surname> </user>

2.2. The Session Framework approach

The previous chapter showed the common approach for handling form values. It forces the user to create two resources for a single form handling.

Cocoon offers an advanced approach. Only one single resource is created. This resources contains the information about the input fields used and in addition the information about where the submitted values should be stored inside the session.

The example from the previous chapter could look like this:

<page xmlns:session="http://apache.org/cocoon/session/1.0"> <session:form name="myform"> <session:action>the-next-page</session:action> <session:content> <session:inputxml name="forename" type="text" context="userdata" path="/user/forename"/> <session:inputxml name="surname" type="text" context="userdata" path="/user/surname"/> </session:content> </session:form> </page>

The form tag starts the form definition. The name attribute is required to distinct between different forms on the same page. The action tag defines the url invoked by the form and the content tag describes the content of the form: its input fields.

The *inputxml* tag tells Cocoon that the following request contains form values which should be stored in the specified context under the given path. The session transformer transforms by removing the namespace and the context attribute:

<page xmlns:session="http://apache.org/cocoon/session/1.0"> <form action="the-next-page"> <inputxml name="forename" type="text"/> <inputxml name="surname" type="text"/> </form> </page>

A stylesheet can now generate the appropriate html (or any other format). The main difference is, that the resource invoked by submitting the values has not to care about the form as Cocoon maintains the form handling. The only prerequisit is that a session for the current user and a session context to store the information exists.

The Cocoon approach allows a very easy way of form handling where the resource which creates the form also handles the form.

For editing values - if the context already contains information about the user - *inputxml* inserts the current value inside the tag. So the xml streamed would after a second run would look like this:

<page xmlns:session="http://apache.org/cocoon/session/1.0"> <form action="the-next-page"> <inputxml name="forename" type="text">Walter</inputxml> <inputxml name="surname" type="text">Walterson</inputxml> </form> </page>

Like *getxml* it is also possible to provide default values for the input field, if the context does not contain any information:

<session:inputxml name="forename" context="userdata" path="/user/forename"> Defaultname </session:inputxml>

But as always there is one drawback with this approach as well, you have to put the *session-form-manager* action somewhere so that it is called when the form values are submitted. As this action does no harm, it can simply be put as the first action in your main sitemap:

<map:act type="session-form-manager"/>

1. Comments

add your comments