Session tracking with Cocoon (2.1 legacy document)

Table of contents

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

Table of contents

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

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

1.1. Goal

Maintaining state is a common problem for web server frameworks because HTTP is a stateless protocol. There are many solutions known to obtain stateful information. Client-side storage of state information like the usage of cookies will not be discussed here, since this depends heavily on the client's browser. Since Cocoon is a server-side framework, storing visitor information at the server side will give full access to the list of all visitors, to what they have done, and what they are doing.

Please always think a little while if you really want to set up session management. Less scalability and performance is the dark side of keeping user sessions at the server-side. Each user session consumes memory, disk, and CPU, and it is always recommended that you be careful to system resources before wasting it.

If you decided to set up session tracking, Cocoon offers you:

- creation of new session IDs
- full session control by the underlying Servlet API 2.2 servlet engine
- cookie- and URI-based session management
- automatic link rewrite if you like your XSP pages to be URI-session-aware

1.2. The xsp-session:encode-url template

To enable Cocoon for URI-based session IDs, an XSP template with the name xsp-session:encode-url will do this for you. It uses the encodeURL method from the Servlet API which encodes an URL in a way that a session ID is being attached. Consult your servlet engine documentation for information about what the encodeURL method returns. For example, the Tomcat engine adds a string ;jsession= followed by an MD5 hash to the URL, but only if the client's browser does not accept cookies.

Here is the fragment for the xsp-session:encode-url from session.xsl:

```
<!-- encode an URL with the session ID --> <xsl:template match="xsp-session:encode-url">
<xsl:variable name="href"> "<xsl:value-of select="@href"/>" </xsl:variable> <xsp:element
name="a"> <xsp:attribute name="href"> <xsp:expr>
response.encodeURL(String.valueOf(<xsl:copy-of select="$href"/>)) </xsp:expr>
</xsp:attribute> <xsl:value-of select="."/> </xsp:element> </xsl:template>
```

As you might wonder, the XSP template constructs a HTML tag <a> with an attribute href which is enough for most of the cases. Other methods, like XLink, are planned to be supported at a later time when final W3C recommendations are out.

1.3. Creating new sessions

The best place of a web site where new sessions should be created is the entry point where all or most of the visitors step in. After creating the session, or retrieving an old session, the visitor is redirected to a start page. In Cocoon, you must edit your sitemap in order to specify this interesting point of session creation. The map-redirect-to has an extra attribute session, which can be set to true or false. The former will generate a new session ID if needed by invoking the Servlet API method session =

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

request.getSession(true), while the latter ignores session ID handling.

How can Cocoon recognize URIs with appended session IDs? The answer is: Cocoon can match a wildcard against your sessionized pages and keeps happy. So please do not forget to append an asterisk '*' to your patterns in the pipelines.

This fragment from sitemap.xsl shows how you can add a map:redirect-to to your Cocoon framework with session handling at the root URL for your web application:

```
<map:pipelines> <map:pipeline> <map:match pattern=""> <map:redirect-to session="true"
uri="welcome"/> </map:match> <map:match pattern="welcome*"> <map:generate type="file"
src="site/welcome.xml"/> <map:transform src="stylesheets/welcome.xsl"/> <map:serialize/>
</map:match> <map:match pattern="**.xsp*"> <map:generate type="serverpages"
src="site/{1}.xsp"/> <map:transform src="stylesheets/dynamic-page2html.xsl"/>
<map:serialize/> </map:match>
```

2. Example

2.1. A simple XSP page with session ID

Here you can see the source of an XSP example of how the session feature can be used. The example is located in a file named sessionpage.xsp and it displays the received session ID together with a rewritten link to the page itself. Depending on your browser settings, you will see nothing (because your browser prefers crunching cookies) or a session ID is encoded into the URL. After clicking on the link named "Follow me!", the session ID is taken into the URL, and the session tracking is established.

```
<?xml version="1.0" encoding="iso-8859-1"?> <xsp:page language="java"
xmlns:xsp="http://apache.org/xsp" xmlns:xsp-session="http://apache.org/xsp/session/2.0"
xmlns:xsp-request="http://apache.org/xsp/request/2.0" > <!-- a simple session page by
J&ouml;rg Prante <joerg@7val.com> --> <page> <title>A Simple URI-based Session
Example</title> <content> <para> <xsp-request:get-uri as="xml"/> </para> <para> Session
ID = <xsp-session:get-id as="xml"/> </para> <para> Encode URL Test =
<xsp-session:encode-url href="sessionpage.xsp">Follow me!</xsp-session:encode-url>
</para> </content> </page> </xsp:page>
```

If you have been successful with installing the session feature and the example file, the following HTML output will be generated by Cocoon from the above sessionpage.xsp example, which shows how the rewritten link looks like. Please don't ask why the session ID in the generated link is different from yours.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN"
"http://www.w3.org/TR/WD-html-in-xml/DTD/xhtml1-strict.dtd"> <html><head><title> A
Simple URI Session Example </title></head> <body vlink="blue" link="blue" alink="red"
bgcolor="white"> <h2 style="color: navy; text-align: center"> A Simple URI Session Example
</h2> <content> <p align="left"><i> <b
xmlns:xsp-response="http://apache.org/xsp/response/2.0"
xmlns:xsp-request="http://apache.org/xsp/request/2.0"> sessionpage.xsp</b> </i></p> <p
align="left"><i> Session ID =
<xsp-session:id>F3E9575442D1899760A0B231D0042281</xsp-session:id> </i></p> <p
align="left"><i> Encode URL Test = <a
href="sessionpage.xsp;jsessionid=F3E9575442D1899760A0B231D0042281"> Follow
me!</a> </i></p> </content> </body></html>
```


Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.

3. Log analysis of sessions

To be done.

1. Comments

add your comments

Copyright © 1999-2005 The Apache Software Foundation. All rights reserved.