XINS Front-end Framework

Anthony Goubard <`anthony.goubard@japplis.com`>

Table of Contents

Introduction

This document describe the XINS Front-end Framework. The XINS Front-end Framework is based on the XINS Framework and included in the releases of XINS (since version 1.5.0). This document supposes that you already know the basis of the XINS framework. If you want to know how to install XINS and create an API, read the primer or the user guide.

Presentation

The XINS Front-end Framework is a web application framework. As most of modern web application frameworks it separates the business logic layer which is using XINS and the presentation layer which is written in XSLT.

Here is a list of advantages compared to the others web application frameworks:

- You can run your application even if you don't have any XSLT yet. Just ask for the XML in the browser and you will see the values returned by your function. You can in some cases even use the generated XINS test forms.
- The presentation layer doesn't need to be included in the WAR file, so that deploy it when you want. Is there an error in a page? just copy the new XSLT file, no need to package and deploy again.
- No need to write components for common things, just write a XSL template, then use `xsl:include` to import the XSLT file and `xsl:call-template` or `xsl:apply-template` to call your XSL template with the parameters you want.
- You win at the same time all XINS features: statistics, security, runtime properties, Java Client API, HTML specifications. Your web application is also accessible using SOAP. The WSDL for your web app can be generated as well as the unit tests and stubs. And much more (logging, Open Source, performance, version management, ...).

A demo of the web application using the XINS Front-end Framework is available here [http://xins.users.mcs2.netarray.com/petstore/].

The framework

The calling convention

The XINS Front-end Framework is based on a calling convention. You will need to declare this calling convention in the impl.xml file to be able to use the framework:

```
<calling-convention name="xinsff" class="org.xins.server.frontend.FrontendCallingC
```

Templates

Templates are used for the presentation layer. The templates (also refered as XSLT) transform the XML returned by the function in HTML (or XHTML, WAP, iHTML).

The location of the templates is specified using the templates.<api name>.xinsff.source runtime property. The value must be a URL (file URL or http URL).

Examples:

- templates.petstore.xinsff.source=file:///${user.dir}/apis/petstore/xslt/
- templates.helloworld.xinsff.source=http://intranet.mycompany.com/templates/helloworld/

Special parameters

The requests can accept special parameters.

Table 1. Special parameters

Parameter with value	Description
mode=source	Returns the XML produced by the function before it is transformed using the template.
mode=template	Returns the XSLT source code used to transform the function return
command=Control	Shows information about the API such as the list of the functions, the value of the sessionproperties, the version of XINS and of the API.

Caching

XINS FF includes the possibility to cache the templates. Caching the templates will increase the response time as the XSLT file doesn't need to be parsed again and compiled (in case you use XSLTC). On the other side caching the templates will increase the memory usage of your web application.

I would advice to disable the cache during the development of the application and to enable it when deploying the application on production.

Enabling/disabling the cache is done by the templates.cache runtime property. The value can be true or false. If the value is not set, the cache is enabled by default.

If you execute the Control command, you have at the bottom of the page a few links to manage the cache. The flush link will clear the cache. The cache will be filled as the pages are used. The second link will clear the cache and reload all the templates in the cache.

Login

XINS Front-end Framework includes some settings to help you having a secure web application.

You first need to specify the login page. This is done be defining the xinsff.login.page bootstrap property.

Once this property set, all pages except the login page and the Control command will redirect to this page if the user is not logged in. If you have other pages that you would like to be accessible to everybody, you will need to specify them as a comma separated list in the value of the xinsff.unrestricted.pages bootstrap property.

There can be some case where you want the user to have certain privileges to access a specific page, so you want to check for the access to this page inside the function. For this purpose, if you create an error code named `NotLoggedIn` and you return this error code from the function, the framework will redirect it to the login page.

To indicate that the a customer is logged in is done by putting the session ID as session property with the value `Boolean.TRUE`. The session ID is put as a cookie in your browser using the `SessionId` cookie name.

Workflow

The workflow is the redirection between pages.

You can specify the default command by setting the xinsff.default.page bootstrap property. The browser will be redirected to this page when no command is specify in the request.

You can specify the redirection by returning a output parameter named *`redirect`* to your function. Note that the parameter will need to be declared in your function specification.

You can redirect a command to another page by setting the xinsff.redirect.<function name> bootstrap property. If the function exists, the function is executed and if the result of the function is successful, the browser is redirected to the value of the property. If the function does not exist, the browser is directly redirected to the value of the property.

It is also possible to specify conditional redirection. The conditions are defined using XPath using the generated XML source as XML to evaluate the XPath. The bootstrap property looks like xinsff.redirect.<function name>[xpath]. If none of the condition matches, the page is redirected to the xinsff.redirect.<function name> bootstrap property value. And if this property is not set the page is not redirected.

The function name is the concatenation of the command name with the action name. For example if the command is `Login` and the action is `okay`, the front-end framework will try to execute the `LoginOkay` function. The show action is the default action, so the Login command with the show action will try to execute the Login function.

Session management

The session is used to stored information about the user or about the state of the application for this user. The session needs to be declared as a shared object so that the session can be accessible from every function.

```
<instance name="_session" getter="getSessionManager" class="org.xins.server.fronte
```

Then you can use the session by calling `_session.setProperty(propertyName, propertyValue)` or `_session.getProperty(propertyName)`. Session property names should be in lowercases. All session properties are put in the XML using `<param name="session.propertyName">propertyValue</param>`. Also all input parameters are stored automatically in the session. The session also contains a _inputs property which contains a `Map` with the input parameters and their values of the request.

If you put in the session an object which is of type `org.xins.common.xml.Element` or a `java.util.List` of `Element` objects, then this (or these) object(s) will be put in the data section of the generated XML.

Note

The session manager uses `ThreadLocal` to keep the `HttpSession`. This means that it assumes 1 thread = 1 request. The Servlet container should be configured for it. Tomcat for example needs the property `protocol=org.apache.coyote.http11.Http11Protocol`.

Note

The session manager can be overridden so that you can use your own session system if you want to.

XML and XSLT

Let's have a look of what the XML produce by the front-end calling convention is.

```
<commandresult command="SearchPet">
  <parameter name="session.petName">test</parameter>
  <parameter name="input.command">SearchPet</parameter>
  <parameter name="input.mode">source</parameter>
  <parameter name="input.action">Okay</parameter>
  <parameter name="input.petName">test</parameter>
</commandresult>
```

Here is an example of the of result when the request is invalid:

```
<commandresult command="Login">
  <parameter name="input.password"/>
  <parameter name="input.command">Login</parameter>
  <parameter name="input.mode">source</parameter>
  <parameter name="input.email">testtest.com</parameter>
  <parameter name="input.action">Okay</parameter>
  <parameter name="error.type">FieldError</parameter>
  <data>
    <errorlist>
      <fielderror type="mand" field="password"/>
      <fielderror type="format" field="email"/>
    </errorlist>
  </data>
</commandresult>
```

From it, the XML can be translated to anything (WAP, iHTML, text) using XSLT. We will more focus in this part in transforming the XML to XHTML and using cascading style sheets (css).

For an example, I suggest you have a look at the Pet store demo XSLT files located in the demo\xins-project\apis\petstore\xslt directory.

A. Properties

Table A.1. XINS FF bootstrap properties

Property name	Description	Example
xinsff.login.page	The login page of the web application. If this parameter is not set, this mean that the application doesn't require a login.	Login
xinsff.unrestricted.pages	The list of the pages that don't need to be redirected to the login page if the user is not logged in.	RegisterCustomer,ThankYouRegistration
xinsff.error.page	The page to redirect to if an XSLT error occurs.	Error
xinsff.default.command	The default command to redirect the user when no command is specify in the request.	SearchPet
xinsff.error.page	The command the user should be redirected if an XSLT error occurs.	Failure
xinsff.redirect.<function name>	If the function is successful, redirect the browser to another page. If the value of this property is "-", no direction will be done. If the value of this property is "/", the browser will be redirected to the default command.	<bootstrap-property name="xinsff.redirect.LoginOkay">SearchPet</ bootstrap-property>
xinsff.redirect.<function name>[xpath expression]	If the function is successful, redirect the browser to another page if the XML result fullfill the xpath expression.	<bootstrap-property name="xinsff.redirect.LoginOkay[param[@name=" = 'true']>Console</bootstrap-property>

Table A.2. XINS FF runtime properties

Property name	Description	Example
templates.<api name>.xinsff.source	This property indicates the location of the XSLT pages. The value must be a URL (file URL or http URL). You can also have system properties in the value by using ${system property}. Don't forget to end the URL with a '/'.	templates.petstore.xinsff.source=file:/// ${user.dir}/apis/petstore/xslt/
templates.cache	Specify if the XSLT should be cached. The value can be true or false. If the value is not set, the cache is enabled by default.	templates.cache=false