Jivan and Enhydra

Table of Contents

Introduction

Besides XMLC, Enhydra supports Jivan, an additional way of DOM (Document Object Model) manipulation in presentation layer of Enhydra Applications. The Jivan project has its own way of DOM parsing, and DOM processing in manner to create dynamic HTML page used as user's HTTP response.

Jivan HTML Reparser is a HTML Parser based on the HTML Scanner nekoHTML. This tool is less robust then XMLC, but, on the other side, its usage is very simple, there is no need for Java DOM intermediate objects, and according to Jivan's documentation it is very fast. For more information about Jivan refer to http://www.jivan.net.

Enabling Jivan reparser usage in Enhydra

To enable Jivan usage in Enhydra application's presentation layer the corresponding parameter in configuration file of the application (or in web.xml file) must be correctly set. The Jivan must be declared in the list of all available presentation tools which can be used for dynamic manipulation of pages in presentation layer of the application. The line beyond should be set in case of if mixed toll usage within the application:

```
Application.PresentationTools[] = xmlc, jivan
```

In the case when only Jivan tool is going to be used within the application, the parameter can be written as:

```
Application.PresentationTools[] = jivan
```

If this parameter is missing the default value "xmlc" will be assumed which means that, in presentation layer, only XMLC can be used.

Note that for each tool, listed in above mentioned parameter, the not null factory object (for example

XMLCFactory or JivanFactory) will be created in the process of application initialization. So, if presentation tool is not defined in configuration file, and its factory object is used in presentation object, the NullPointerException will be thrown.

It is assumed that in case of Jivan usage, 'jivan.jar' is present in Enhydra library directory.

How to use Jivan reparser in Enhydra applications

Using of Jivan instead of XMLC in application presentation layer is very simple. There is no intermediate Java DOM classes which must be previously generated from HTML files. HTML file which have to be dynamically modified is read directly from the presentation page and is parsed on the fly. The most important object is DocumentManager which is used to obtain corresponding Document object (DOM representation of HTML file) which has to be modified.

To enable similarity with XMLC, Enhydra provides a class JivanFactory (package org.enhydra.util.jivan) which instance is placed in HttpPresentationComms object, and is available to each Enhydra presentation object. JivanFactory is used to invoke desired HTML resource, and to retrieve corresponding DocumentManager object. The following code line describes JivanFactory usage:

```
DocumentManager man = comms.jivanFactory.docManFor("file:///C:/projects/Welcome.html");
```

When DocumentManager object is available, the Document object (Jivan's implementation of org.w3c.dom.Document interface) can be found and manipulated by DOM manipulation methods. Also, Jivan posses DOMUtil class which has few methods for easier Node manipulation. As at the XMLC, the HTML tags which have to be accessible by DOMUtil methods, must have unique tag attribute 'id'. The following code lines describe DOMUtil usage:

```
DOMUtil.setTextChild(man.lookup("time"), datetime);
HTMLAnchorElement link = (HTMLAnchorElement)man.lookup("link");
link.setHref("RedirectPresentation.po");
```

When Document is dynamically changed it should be sent to user as HTTP response. Again, to enable similarity with XMLC usage, the already existing writeDOM() methods in tionReServletHttpPresentasponse object are used. Because writeDOM() methods as its argument took the XMLObject object, the one intermediate step must be done. The XMLObject object must be created with the Jivan's Document object. Enhydra enables two implementations for XMLObject abstract class in order to enable invoking writeDOM() methods with Jivan's Document object.

The first implementation is SimpleXMLObject (package org.enhydra.util.dom). Jivan's Document object passed via SimpleXMLObject object in its transformation into byte[] array, which will be sent as HTTP response, uses XMLC's DOMFormatter class. The following code lines describe SimpleXMLObject usage:

```
SimpleXMLObjectImpl xmlObject = new SimpleXMLObjectImpl();
xmlObject.setDocument(man.getDocument(), "text/html", "ISO-8859-1");
comms.response.writeDOM(xmlObject);
```

The second implementation is JivanSimpleXMLObject (package org.enhydra.util.jivan). Jivan's Document object passed via JivanSimpleXMLObject object in its transformation into byte[] array, which will be sent as HTTP response, uses Jivans's original DocumentManager serialize() method. It is recommended to use this implementation to exploit Jivans fastness, which is mentioned in its documentation. The following code lines describe SimpleXMLObject usage:

```
JivanSimpleXMLObjectImpl xmlObject = new JivanSimpleXMLObjectImpl();
xmlObject.setDocument(man, "UTF-8");
comms.response.writeDOM(xmlObject);
```

Note that there is few setDocument() methods in both implementations. Some of methods are using the default values for encoding or MIME type parameters. For more information refer to Java API documentation.

Example

The example of one Enhydra presentation page, named "WelcomePresentation", with Jivan DOM manipulation usage is shown below:

```
/*
 *  TestJivan
 *
 *  Enhydra super-servlet presentation object
 *
 */

package test.presentation;

import com.lutris.appserver.server.httpPresentation.*;
import java.util.GregorianCalendar;
import org.enhydra.util.jivan.JivanSimpleXMLObjectImpl;
import org.jivan.html.document.DocumentManager;
import org.jivan.html.util.DOMUtil;
import org.w3c.dom.html.HTMLAnchorElement;
import java.util.Date;

public class WelcomePresentation implements HttpPresentation {

public void run(HttpPresentationComms comms)
        throws HttpPresentationException {

  DocumentManager man = comms.jivanFactory.docManFor(
   "file:///C:/projects/TestJivan/src/test/resources/Welcome.html");

  //gets time to show that content is non-static
  GregorianCalendar c = new GregorianCalendar();
  Date d = c.getTime();
  String datetime = d.toString();

  DOMUtil.setTextChild(man.lookup("time"), datetime);

  HTMLAnchorElement link = (HTMLAnchorElement)man.lookup("link");
  link.setHref("RedirectPresentation.po");

  JivanSimpleXMLObjectImpl xmlObject = new JivanSimpleXMLObjectImpl();
  // invoking setDocument() method with default values for
  // encoding and MIME type
  xmlObject.setDocument(man);
  comms.response.writeDOM(xmlObject);

 }

}
```

Jivan AutoReloading

JivanFactory class supplied by Enhydra has implemented possibility of resources autoreloading. By default autoreload option is off. To turn on this option the configuration file parameter:

```
Server.Jivan.AutoReload = false
```

should be switched to 'true'.

This flag indicates to the application, which uses the Jivan DOM manipulation in its presentation layer, that resources (HTML files) will be reloaded if they were changed. This option can be useful in testing purpose when changing of resources can be visible without stopping the server.

Note that this option enable checking of HTML files timestamps, and according to this value changed HTML files will be reloaded and reparsed, or not. If the resource is passed as XMLInputSource object, which is created with character or byte streams, the timestamp checking is impossible and reloading is performed always if AutoReload option is set to true.