Jspx-bay

Subjects: Computer Science, Software Engineering
Contributor: HandWiki Li

jspx-bay, commonly referred to as jspx, is a free open source pure Java web RAD framework. Jspx should not be confused with other technologies using the same name like Oracle Application Framework and XML JSP. Jspx extends Java EE servlets to provide an Object-oriented programming model for HTML declarative code. Jspx can be compared to JSF as a web framework. There are many other Java Web frameworks like Apache Wicket that implement such ideas.

Keywords: servlets ; programming model ; open source

1. History

Jspx development initially started in February 2008 as a trial to provide an easier way to develop rich interactive web applications in Java. In July 2008, jspx was introduced to the Java community through java.net. Initially jspx had very limited features, including support for web form based development only. https://handwiki.org/wiki/index.php?curid=1172275

In December 2008, the project moved to SourceForge,[1] where it has been hosted since.

2. The Name

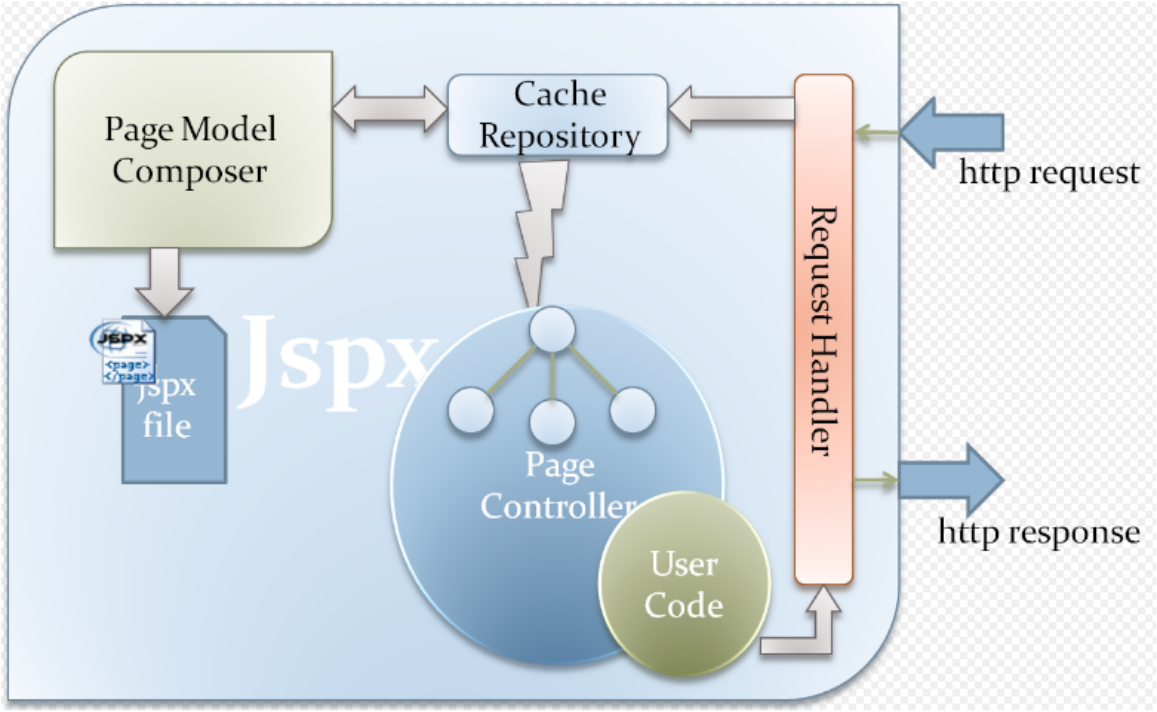
The name was chosen to indicate the next step of JSP technology, despite the fact that jspx is significantly different from JSP in that it does not directly embed Java code in HTML. The suffix **X** is analogous to ASP.NET pages, which have the extension aspx. The official jspx framework name on SourceForge is jspx-bay. The suffix **bay** distinguished the framework from an earlier inactive SourceForge project named jspx.[2]

On 16 November 2009, the jspx project on SourceForge was purged, and it provides the same content as jspx-bay.[3]

3. Framework Target

Jspx aims at providing easy to use, developer-friendly APIs. Based on the idea that web development is mostly about customizing the HTML that is presented based on user-input, jspx offers an object-oriented **view layer** interface to HTML. Jspx provides a means of implementing a stateful user interface over a stateless protocol (HTTP). JSF provides similar functionality, but requires that developers learn an entire new set of tags.

4. Design Goals



Jspx is relatively new, and combines many features and advantages from existing frameworks while eliminating what might be considered disadvantages. Jspx had the following design goals:[4] https://handwiki.org/wiki/index.php?curid=1486565

1. Business case Driven Framework

   Remove boilerplate code and tasks.
2. Zero configuration

   Unlike JSF, does not require any external configuration.
3. Declarative and Imperative code

   Attributes declared in HTML are accessible by fully object-oriented APIs.
4. Optional and Default Implementation

   No need to specify the value of every feature, because they have reasonable default values.
5. Integration with other frameworks

   Import existing jsp files into jspx HTML pages.
6. Portable framework

   Run on almost every Application Server without extra effort.

5. Comparison with Other Frameworks

5.1. ASP.NET

Jspx has the following similarities with ASP.NET:

1. Webform-based framework.
2. Page life cycle and events.
3. Page Templates through Master/Content pages.

Jspx is different from ASP.NET in the following ways:

1. Jspx does not require a particular extension for the html file (i.e. aspx).
2. Jspx uses standard HTML tags.
3. Jspx provides many extra features (such as Jspx Beans for View-Controller binding).

5.2. JSF

Jspx is similar to JSF in the following ways:

1. The same goal of providing an object-oriented interface to HTML.
2. Bean Binding.

3. Binding to Page properties.

Jspx differs from JSF in the following ways:

1. Uses standard HTML tags.
2. Does not rely on any external configuration.
3. Full control of the page life cycle.

Jspx can be also compared to many other frameworks, such as Apache Wicket, but jspx uses HTML tags and attributes without an additional XML namespace.

6. Web Development Using Jspx

As Jspx is a webform-based framework, for each business case there is one development unit. A development unit in a jspx application consists of two parts:[5]

1. Declarative HTML code (myPage.htm)
2. Imperative Java code (MyPageController.java)

In addition, the **web.xml** file must direct traffic to jspx main servlet controller (RequestHandler).

7. Example

A **hello world** example can be implemented in many different ways. The following example demonstrates one of them.

web.xml

```
<?xml version="#0" encoding="UTF-8"?> <web-app xmlns:xsi="http://www.w#org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_#xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_#xsd" id="WebApp_ID" version="#5"> <display-name>jspx-demo</display-name> <servlet> <servlet-name>JspxHandler</servlet-name> <servlet-class>eg.java.net.web.jspx.engine.RequestHandler</servlet-class> </servlet> <servlet> <servlet-name>ResourceHandler</servlet-name> <servlet-class>eg.java.net.web.jspx.engine.ResourceHandler</servlet-class> </servlet> <servlet-mapping> <servlet-name>JspxHandler</servlet-name> <url-pattern>*.html</url-pattern> </servlet-mapping> <servlet-mapping> <servlet-name>ResourceHandler</servlet-name> <url-pattern>/jspxEmbededResources/*</url-pattern> </servlet-mapping> </web-app>
```

HelloWorld.html

```
<page controller="HelloWorld"> <html> <body> <label id="message" /> </body> </html> </page>
```

HelloWorld.java

```
import eg.java.net.web.jspx.ui.controls.html.elements.Label; public class HelloWorld extends Page { Label message; protected void pageLoaded() { if (!isPostBack) message.setValue("Hello Jspx World"); } public void setMessage(Label message) { this.message = message; } public Label getMessage() { return message; } }
```

Note the following about the example above:

1. The only configurations required are to register two servlets of jspx; one for the normal requests on the pages, and the second for embedded resources (such as JavaScript and images).
2. The HTML must be contained within a root element `<page>`, and must be well-formed XHTML.
3. The Java code must extend the root Page controller from the jspx page.
4. The page has life cycle phases as with ASP.NET, with similar names like **pageLoaded**
5. The HTML page should point to the Java controller class through the **controller** attribute of the `<page>` element.

6. In the Java controller, in case it is needed to interact with an HTML control, a control of the same type should be declared as a java property with getter and setter, and the property name should be the same as the value of the attribute ID in the HTML control.

For the note number 6, the framework is considering using of Dependency Injection to eliminate such limitation and any coupling of HTML and java code.

8. Dependency Injection

Jspx build 1.0.9 provided new way of linking HTML controls to Java controls in the page controller. The above example showed the need to add getter and setter for each control. This methodology created the following cons:

1. Developer has to type a lot of code to link HTML control to Java control.
2. The Page controller is getting very crowded with irrelevant code.
3. Coupling between the HTML control and Java Control; as the link is created only if the Java control name is the same as the value of the attribute id in the HTML control.
4. Any change in the HTML requires changing the getter and setter in the page Java controller.

For all these reasons build 1.0.9 is using Java Annotation to link Java objects to HTML. The following example is an alternative to the above one.

```
import eg.java.net.web.jspx.ui.controls.html.elements.Label; public class HelloWorld extends Page { @JspxWebControl(name="message") Label msg; protected void pageLoaded() { if (!isPostBack) msg.setValue("Hello Jspx World"); } }
```

As noted the size of the code is very much reduced. Also it is noted that the annotation @JspxWebControl is using an attribute *name* which is an optional attribute. The value of this attribute should be the same as the id of the HTML control. This technique removes the coupling between HTML and Java code.[6]

Some consider this new feature an insignificant addition to the framework. They use the JspxBean to link the HTML form to Java Model Object, and do not declare server side controls linked to the HTML controls.

JspxBean in build 1.0.9 also has been improved, there is no need to create getter and setter for every JspxBean. Forgetting to add getter and setter was a very common mistake causing the developer to loose the link with the HTML. Dependency Injection is used to inject the JspxBeans without the need for getters and setters.

9. Expression Language Support

Jspx does not allow Java code inside HTML page. However, in order to satisfy the need of injecting data from the controller; Jspx fully supports Expression Language.

Some examples on EL as following:

- **${this.customerName}**. The keyword *this* refers to the current instant of the page controller. *customerName* is the name of the Java property in the controller class. The property should have a getter method as
  ```
  public String getCustomerName()
  ```
- **${request.parameters.id}**. The keyword *request* refers to the HTTP Servlet Request. *parameters* refers to the Request parameters. *id* refers to the parameter name. HTTP Servlet Request attributes can be also accessed as *${request.attributes.id}*.
- **${session.id}**.The keyword *session* is used to access parameters in the HTTP Servlet Session.
- **${var.age}**. In Data Item Controls like DataTable and List Table, custom rendering of Data Column can be done by using ItemTemplate. in order to access the current Database record a variable should be declared in the DataTable with name *var* for example. Then inside the ItemTemplate the values inside the *var* variable can be accessed as shown in the EL.
- **${size(this.username)}** JEXL provides static methods to be invoked .
- **${user.age>21?'You are OK':'You are minor'}** *user* must be a jspx-bean defined in the Page controller and the expression is checking on the value of the age (member property of the user bean).

10. Limitations and Disadvantages

Any framework is evaluated according to different criteria aspects, the following which jspx needs more improvements.

1. **IDE support**. Jspx can be developed using any IDE, however there is no direct support for jspx. Developer has to write the event handling manually.
2. **Documentation**. Tutorials and examples are found on the official website. There is a section on the website for documentation, however not sufficient.
3. **EL Support**. Jspx did not fully support EL language until version 2.0, JEXL is used starting from jspx version 2.0 to fully support EL.

11. Features

Jspx offers some powerful features like:

1. Native support for Ajax through AjaxPanel[7][8][9]
2. Data Controls like DataTable and ListTable.
3. Integration with JAAS and adding more features.[10]
4. Data Binding through JspxBean.
5. HTML Template through Master/Content Pages.
6. Client/Server side validation.[11]
7. Embedded JQuery, JQuery UI, Bootstrap style and many other open source libraries inside.
8. Embedded JEXL to support EL expressions.

12. External Jar Dependency

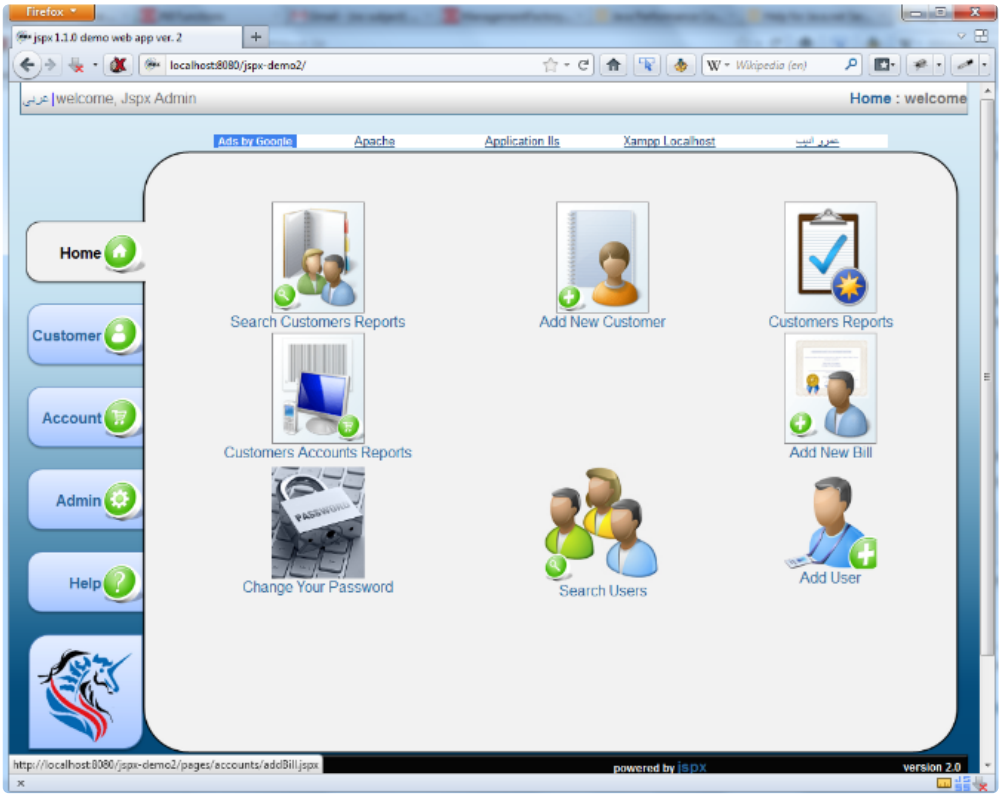
Jspx tries to minimize external jar dependencies, however there are some essential jars: [12]

1. commons-fileupload-1.2.1.jar (when using File Upload)
2. slf4j-api-1.7.2.jar (Mandatory for logging)
3. poi-3.1-FINAL-20080629.jar (when using Export To Excel)
4. servlet-api.jar (only during compilation)
5. commons-io-1.3.2.jar (when using File Upload)
6. commons-jexl-2.1.1.jar
7. jcl-over-slf4j-1.7.5.jar slf4j bridge for Commons logging (As JEXL is using it)

13. JDC 2010

Jspx had made its first public appearance at Java Developer Conference 2010 (JDC). Java Developer Conference is the largest Java event in MENA, organized by EGJUG.[13]

14. Jspx Demo Projects



Jspx distribution package includes several files beside the binary jar and the source code. Since the first release on SourceForge the distribution included a demo project as a binary war file and a zipped source code. That project was a simple discrete pages showing different features of Jspx. There was no common business module wrapping these pages, beside it had a poor GUI design. Jspx build 1.1.0 add new demo project that facilitates the easy use of jspx to develop common business case of interacting with DB. The demo is using MySQL. Sql script file is separately downloadable. https://handwiki.org/wiki/index.php?curid=1342358

Some of jspx features are used like Master/Content pages, Ajax, DataTable, Validators and web forms.

With Jspx 1.2 there were two new demo projects added. One offline Demo was jspx-demo3, that is demonstrating the use of jspx to create simple asset tracking application. The other demo was an online jspx live demo.

References

1. http://jspx-bay.sourceforge.net http://jspx-bay.sourceforge.net
2. "Jspx Bay Name". http://jspx-press.blogspot.com/2009/09/jspx-bay.html.
3. "Jspx Name on SourceForge". http://sourceforge.net/projects/jspx/.
4. "Jspx Design Aspects". http://jspx-bay.sourceforge.net/index.html?l=pages/tout/design.html.
5. "Jspx Hello Web". http://jspx-bay.sourceforge.net/index.html?l=pages/tout/helloweb.html.
6. "jspx dependency injection". http://jspx-press.blogspot.com/2009/10/jspx-dependency-injection.html.
7. "Jspx Ajax". http://jspx-bay.sourceforge.net/index.html?l=pages/tout/ajax.html.
8. "Ajax in Jspx". http://jspx-press.blogspot.com/2009/09/ajax-in-jspx.html.
9. "Jspx AjaxPanel Limitation". http://jspx-press.blogspot.com/2009/09/ajax-panel-limitations.html.
10. "JAAS on the webcontrol level". http://jspx-press.blogspot.com/2009/09/jaas-on-web-control-level.html. .
11. "jspx features". http://jspx-bay.sourceforge.net/index.html?l=pages/jspx/features.html.
12. "jspx jar dependency". http://jspx-bay.sourceforge.net/pages/tout/start.html.
13. "jspx in JDC 2010". http://jdc2010.egjug.org/node/24.

Retrieved from https://encyclopedia.pub/entry/history/show/73524