Paper on Integration of AJAX and Java Server Faces

Praveen Reddy Turpu
CIS 764
Kansas State University

Abstract:

Everybody is now talking about Web 2.0, a Web that is more dynamic, richer, more interactive, and, ultimately – much more exciting than anything we know now. It’s just human nature to look for unusual and new stuff. But, what does that mean for business applications?

Let’s look at Google Maps. Yes, we are all excited by Google Maps. It looks great. It is very interactive. And, most importantly, it behaves completely differently from what we expect to see in a “normal” browser. But, compared to any realistic business application – say something like trivial Internet banking – Google Maps is nothing. It supports just a few use cases compared to the hundreds or thousands of use cases for a typical business application.

How many input fields does it have? How many validation rules? How complex is the business process and page flow? Anybody could build Google Maps using any kind of technology, but business application developers face completely different challenges, challenges that are much more demanding!

What does Web 2.0 mean for the present world:

Web 2.0 is a phrase that is coined by O'Reilly Media in 2004, referring to the second generation Internet-based services. Second generation Internet services has become more and interactive and exiting to the present users. With out which it would have been impossible to construct such a small world. An enterprise want to improve there sales by providing users much user friendly and interactive websites.

Web 2.0 has been included with advanced technologies like AJAX, JSF, Visual Studio 2005, Web Parts Etc., these had made easy to build the things in its own manner. You can see a large number of examples in the present world which use these technologies for eg., Google Maps, Google Personalized Home, Shopping Websites with Drag and Drop feature and many more. Coming to the other side of building, which is security web 2.0 has improved a lot in providing a secure user interface, for eg., Yahoo has come up with a sign-in seal that Yahoo! will display on your computer. It has also provided several advanced encryption and decryption techniques.

Web components has become promising for the advancement Web 2.0. Building a web application has been made easy, instead of building it from the scratch using the components that are already built we can build a web application with all its interactive features in not time one of the examples of such technologies are Java Server Faces along with AJAX.

What is JavaServer Faces (JSF)?

Definition from sun: A server side user interface component framework for Java technology-based web applications.

- JavaServer Faces (JSF) is a new standard Java framework for building Web applications. It simplifies development by providing a component-centric approach to developing Java Web user interfaces.
- JavaServer Faces also appeals to a diverse audience of Java/Web developers.
- "Corporate developers" and Web designers will find that JSF development can be as simple as dragging and dropping user interface (UI) components onto a page.
- "systems developers" will find that the rich and robust JSF API offers them unsurpassed power and programming flexibility.
- JSF also ensures that applications are well designed with greater maintainability by integrating the well established Model-View-Controller (MVC) design pattern into its architecture.

Java Server Faces Implementation of MVC

One of the key advantages of JSF is that it is both a Java Web user-interface standard as well as a framework that firmly follows the Model-View-Controller (MVC) design pattern. This makes JSF applications much more manageable because the user-interface code (View) is cleanly separated from the application data and logic (Model).
What is AJAX?

Ajax, shorthand for Asynchronous JavaScript and XML, is a web development technique for creating interactive web applications. The intent is to make web pages feel more responsive by exchanging small amounts of data with the server behind the scenes.

Ajax isn’t a technology. It’s really several technologies, each flourishing in its own right, coming together in powerful new ways. Ajax incorporates:

- standards-based presentation using XHTML and CSS;
- dynamic display and interaction using the Document Object Model;
- data interchange and manipulation using XML and XSLT;
- asynchronous data retrieval using XMLHttpRequest;
- and JavaScript binding everything together.

An Ajax application eliminates the start-stop-start-stop nature of interaction on the Web by introducing an intermediary an Ajax engine between the user and the server.

It seems like adding a layer to the application would make it less responsive, but the opposite is true.

Instead of loading a webpage, at the start of the session, the browser loads an Ajax engine written in JavaScript and usually tucked away in a hidden frame. This engine is responsible for both rendering the interface the user sees and communicating with the server on the user’s behalf. The Ajax engine allows the user’s interaction with the application to happen asynchronously independent of communication with the server. So the user is never staring at a blank browser window and an hourglass icon, waiting around for the server to do something.

Integrating Java Server Faces and Ajax:

Rich and dynamic should not equate to being less functional and too simple. When you deal with a real-world application, you must be ready for complicated page flows, data model, and business logic. To handle this complexity, JSF puts all of the needed tools and capabilities into the hands of developers. What JSF still needs is a little bit of “richness” to complete it. And, that is exactly what Ajax4jsf does.
To embrace openness and the true spirit of JSF to achieve the widest possible compatibility with different JSF implementations and third-party component libraries
To add native support to great technologies complement JSF, like Facelets and JSF Extensions, to create a frameworks ecosystem around the JSF core

Introduction to Ajax4JsF Framework:

The framework is implemented by using a component library which adds AJAX capability in existing pages needlessly to write any JavaScript code or to replace existing components with new AJAX widgets. Ajax4jsf enables page-wide AJAX support instead of the traditional component-wide support. So, you can define the event on the page that invokes an AJAX request and the areas of the page that should be synchronized with the JSF Component Tree after the AJAX request changes the data on the server according to the events fired on the client.

Ajax Filter:
To get all benefits of using Ajax4jsf register an Ajax Filter in web.xml file of your application. This filter is responsible the first case the whole JSF tree will be encoded. In the second option it depends on the “size” of the AJAX region (you can define AJAX region by using the <a4j:region> tag). As you for a few tasks. The Ajax Filter recognizes multiple request types. The sequence diagram on Figure 3 shows the difference in processing of a “regular” JSF request and an AJAX request. In can see, in the second case the filter parses the content of an AJAX response before sending it to the client side.
In both cases, the information about required static or dynamic resources your application requested will be registered in the Resource Builder class. When a request for a resource comes, the AJAX filter checks the Resource Cache for this resource and if it is there, the resource will be sent to the client. Otherwise, the filter will search for the resource among those that are registered by the Resource Builder. If the resource is registered, the AJAX filter will send a request to the ResourceBuilder to create (deliver) the resource.

Ajax Action Components:
There are three Ajax Action Components: AjaxCommandButton, AjaxCommandLink and AjaxSupport. You can use them to send Ajax requests from the client side.

Ajax Containers:
AjaxContainer is an interface that describes an area on your JSF page that should be decoded during an Ajax request. AJAX-style interactions between the client and server sides of an application. Here is an example

AjaxViewRoot and AjaxRegion are implementations of this interface.

JavaScript Engine:
Ajax4jsf JavaScript Engine runs on the client-side. It knows how to update different areas on your JSF page based on the information from the Ajax response. Application developers do not use this JavaScript code directly. It’s available automatically.

Use Ajax4jsf Components:
In addition to all of these great features, Ajax4jsf contains a set of JSF components that are essential for creating modern Web 2.0 applications. These components work right out of the box and allows dynamic
Steps in Sending Ajax Requests from JSF:

Send an AJAX Request

There are different ways to send Ajax requests from your JSF page. You may use `<a4j:commandButton>`, `<a4j:commandLink>` or `<a4j:support>` tags. All these tags hide the usual JavaScript activities needed to build an XMLHttpRequest object and send an Ajax request. Also, they allow you to decide which components of your JSF page are to be re-rendered as a result of the Ajax response. `<a4j:commandButton>` and `<a4j:commandLink>` tags are used to send an Ajax request on “onclick” JavaScript event. The `<a4j:support>` tag allows you to add Ajax functionality to standard JSF components and send Ajax request on the JavaScript event of your choice: “onkeyup”, “onmouseover”, etc.

Most important attributes of components which provide AJAX request calling features:

- **reRender attribute**: as it was mentioned before specifies components to be reRendered after AJAX response. Can be specified using EL expression and formed dynamically on the server side.
- **RequestDelay attribute**: used for regulate frequency of the requests.
  ```html
  <h:inputText size="50" value="#{bean.text}">
    <a4j:support event="onkeyup" RequestDelay="3"/>
  </h:inputText>
  
  So every next request from frequent keyboard events will be delayed on 3 ms to reduce number of requests.
- **EventsQueue**: name of the queue which stores the next request. The queue helps to block next requests before completion of the current one.
- **LimitToList attribute**: used to regulate updatable regions. Setting it to true – will limit the updatable areas only to specified in reRender list, in other case all Output Panels of the region will be updated.
- **ajaxSingle attributes**: specify regions to be sent with request if false – it’ll be full region in other case - only control caused event.

Decide What to Send

You may describe a region on the page you wish to send to the server. This way you can control what part of the JSF View will be decoded on the server side when you send an AJAX request. The easiest way to describe an Ajax region on your JSF page is to do nothing. That’s because the content between the `<f:view>` and `</f:view>` tags is considered the default Ajax region. You may define multiple Ajax regions on the JSF page (they can even be nested) by using the `<a4j:region>` tag. If you wish to render the content of Ajax response outside of the active region then the value of the “renderRegionOnly” attribute should be set to “false”. Otherwise your Ajax updates are limited to elements of the active region.

Decide What to Change

Using IDs in the “reRender” attribute to define “AJAX zones” for update works fine in many use cases. But you can not use this approach if your page contains, for example, a `<f:verbatim>` tag and you wish to update its content on AJAX response. The problem with the `<f:verbatim>` tag that we described above is related to the value of the transientFlag of JSF components. If the value of this flag is true, the component must not participate in state saving or restoring of process. In order to provide a solution to this kind of problems, Ajax4jsf uses the concept of an output panel that is defined by the `<a4j:outputPanel>` tag. If you put a `<f:verbatim>` tag inside of the output panel, then the content of the `<f:verbatim>` tag and
content of other panel’s child tags could be updated on AJAX response. There are two ways to control this:

- By setting the “ajaxRendered” attribute’s value to “true”.
- By setting the “reRender” attribute’s value of an Action Component to the output panel’s ID.

Limitation and Rules:

For properly creating AJAX Java Server Faces applications, developers must keep the following points in mind:

- The AJAX framework should not append or delete, but only replaces elements on the page. For successful updates, an element with the same ID as in the response must exist in the page. If you want append any code to a page, put in a placeholder for it (any empty element). For the same reason, we recommend placing messages in the AjaxOutput component (since no messages is also a message).
- AJAX requests are made by XMLHttpRequest functions in XML format, but this XML bypasses most validations and the corrections that might be made in a browser. So, create only strict standards-compliant code for HTML and XHTML, not skipping any required elements or attributes. Any needed XML corrections are automatically made by the XML filter on the server, but many unexpected effects can be produced by incorrect HTML.

Conclusion:

A components-based approach using JSF and Ajax4jsf is the best way for enterprises to address the Web 2.0 challenge. Ajax4jsf component library is a perfect match for JSF component library because this component library will not jump outside the JSF component model. It provides cool, rich, fast and interactive internet experience which is exactly what the present world needs.

References:

