

J2EE Fusebox Whitepaper

Benjamin Edwards (ben.edwards@synthis.com)

June 06, 2002

J2EE Fusebox is a Java implementation of the Fusebox architecture framework. J2EE Fusebox is a powerful, yet easy to use, and brings the full power of Java and a proven architecture and development methodology to your web-applications.

Table of Contents

Introduction to J2EE Fusebox	3
J2EE Fusebox Components	4
Fusebox JSP Files	4
Fusebox API	7
Fusebox Classes.....	9
Fusebox Tag-Library	11
The Life of a Fusebox Request.....	15
J2EE Fusebox Resources	17
Appendix A: J2EE Fusebox and JSTL	18

Introduction to J2EE Fusebox

Fusebox is a methodology for designing web applications. Its smart design, ease of use, and openness have resulted in thousands of developers forming an innovative and dedicated community behind the growth of Fusebox. Consequently, documentation and training in the art of Fusebox development is abundant, and its adoption is ever farther-reaching.

While the Fusebox methodology was born and has grown on the ColdFusion platform, and the majority of the Fusebox community is comprised of ColdFusion developers, the strength and ideas of the methodology extend to most, if not all, page-centric web-architectures. There currently exist implementations on several platforms: PHP, Microsoft ASP, and of course Java (JSP) and ColdFusion.

The Fusebox architecture and methodology make a great combination in the Java enterprise world. J2EE Fusebox brings together the benefits of Fusebox and the industrial strength power of the Java 2 Enterprise Edition (J2EE) platform.

Two primary goals of the of J2EE Fusebox project are to create a Java implementation of Fusebox that aligns closely with the Fusebox standard and ColdFusion implementation, and to combine the benefits of Fusebox with the Java platform. JavaServer Pages (JSP) technology is very similar to ColdFusion. JSP allows developers to build dynamic Web applications using a combination of a powerful programming language (Java) and other standard Internet technologies, such as HTML and XML.

J2EE Fusebox provides benefits to Java and ColdFusion developers alike. J2EE Fusebox gives ColdFusion developers a great on ramp to the world of enterprise Java. And for experienced Java developers, J2EE Fusebox provides a powerful, flexible, and proven architecture and methodology to tame the complexity of J2EE.

J2EE Fusebox Components

The J2EE Fusebox framework consists of JSP files, a small tag-library, and compiled classes. The JSP files are very similar to their ColdFusion relatives; all that's changed is the syntax.

Fusebox JSP Files

Directory	File	Required	Description
/	fbx_fusebox301.jsp	Yes*	Provides core processing of a Fusebox request.
/	fbx_settings.jsp	No	This file is optional and is used to set variables needed by the application. Each circuit may have its own fbx_settings.jsp file. Each circuit's fbx_settings file will be included in top down order allowing you to set variables at high levels and override them.
/	fbx_circuits.jsp	Yes*	Required in the home circuit, this file is responsible for mapping circuit aliases to physical paths and setting them in the Fusebox.circuits structure.
/	fbx_switch.jsp	Yes	This file is placed in every circuit that will handle fuseactions. The file contains a switch statement with cases for every fuseaction the circuit should handle.
/	fbx_layouts.jsp	No	Required in any circuit that implements a separate layout file, this file is responsible for setting the variables Fusebox.layoutDir and Fusebox.layoutFile.
/WEB-INF	fbx_include.jsp	Yes**	A standard include that should be referencable with a url relative from the root (so it can be easily included in any page). This file includes maps the Fusebox tag-lib to the 'fbx' prefix and includes the Fusebox object in the page scope. It also includes some utility JSP methods that may aid in development (these methods are documented later in the document).
/WEB-INF	fusebox.tld	Yes**	The tag library descriptor for the Fusebox tag-lib.
/WEB-INF	web.xml	Yes**	A standard JSP web-application configuration file necessary for the tag-library.
/WEB-INF/lib	fusebox-j2ee.jar	Yes**	The standard jar containing the Fusebox classes.

* Only required in home circuit.

** Required in the JSP/Fusebox environment.

fbx_fusebox.jsp – This file acts as the main controller of a Fusebox system providing the core processing of a Fusebox request. It is required in the home circuit of an application. Typically

this file will be suffixed with a version number. For instance, fbx_fusebox301.jsp denotes Fusebox version (3.01).

fbx_circuits.jsp – This file maps circuit aliases to directory paths and sets them in the Fusebox.circuits structure. It is required in the home circuit of a Fusebox application.

Example code from an fbx_circuits.jsp file:

```
<%
    fusebox.circuits.put( "home", "home" );
    fusebox.circuits.put( "Login", "home/Login" );
%>
```

fbx_switch.jsp – Each instance of this file acts as a circuit's controller, mapping each fuseaction in a circuit to specific fuse files to be executed. This file is required for each circuit.

Example code from an fbx_switch.jsp file:

```
<fbx:saveContent variable="fusebox.layout">
<fbx:switch value="<%= fusebox.fuseaction %>">
    <fbx:case value="editProfile">
        <%@ include file="qry_getProfile.jsp" %>
        <%@ include file="act_restoreProfile.jsp" %>
        <%@ include file="dsp_ProfileEdit.jsp" %>
    </fbx:case>
</fbx:switch>
</fbx:saveContent>
```

fbx_settings.jsp – Each instance of this file sets circuit configuration information. This file is optional for each circuit.

Example code from an fbx_settings.jsp file:

```
<%
    String defaultFuseaction = "home.main";
    fusebox.suppressErrors = false;

    if( fusebox.isHomeCircuit )
    {
        application.setAttribute( "name", "FuseboxApp" );
    }
%>
```

fbx_layouts.jsp – Each instance of this file sets a circuit's layout file and directory information. This file is optional for each circuit.

Example code from an fbx_layouts.jsp file:

```
<%
    fusebox.layoutDir = "";
    fusebox.layoutFile = "TypicalLayout.jsp";
%>
```

fbx_include.jsp – When included in a fuse (using the include directive), this file makes sure the request's Fusebox object is available in local scope and the Fusebox tag-library can be accessed. This file is not based on a ColdFusion relative, it is a utility file for the Fusebox/JSP environment. Typically this file will reside in an application's WEB-INF directory, where it can be easily referenced with a relative URI.

The fbx_include file has four main purposes:

1. Reference the fbx tag-library (and other commonly used taglibs).
2. Import the classes of the org.fusebox and java.util classes (and other common packages).
3. Retrieve the Fusebox instance from the request and put into the local scope.
4. Define methods that are used throughout the Fusebox system.

Pages that require fbx_include.jsp in each circuit:

1. fbx_circuits.jsp
2. fbx_layouts.jsp
3. fbx_settings.jsp
4. Any layout file (i.e. TypicalLayout.jsp)
5. Any file executed that is not included in a circuit switch with the include directive (<%@ include page="..." %>).

Example code from an fbx_include.jsp file:

```
<%@ taglib uri="/WEB-INF/fusebox.tld" prefix="fbx" %>
<%@ page import="org.fusebox.Fusebox, java.util.*" %>
<%
    Fusebox fusebox = (Fusebox) request.getAttribute(Fusebox.FUSEBOX);
%>
```

fusebox.tld – Describes the Fusebox Tag-Library (TLD stands for tag library descriptor). Refer to the J2EE specification for more information on TLD files.

web.xml – This standard JSP Web application configuration file is necessary for defining the Fusebox Tag-Library. Refer to the J2EE specification for more information on web.xml files.

fusebox-j2ee.jar – This file conveniently packages the compiled Fusebox classes and makes them available to the JSP environment.

Fusebox API

The core `fbx_fusebox.jsp` file makes use of the `org.fusebox.Fusebox` class. This class represents the Fusebox API. Some variables are public, but are not necessarily meant to be set by a user. The Fusebox class contains some public constants that can be used as parameters. An instance of Fusebox is created for each request and is set in the local scope of every page.

Public API variable	Type	May be set	Description
<code>Fusebox.isCustomTag</code>	boolean	No	Is this Fusebox being called as a custom tag?
<code>Fusebox.isHomeCircuit</code>	boolean	No	Is the directory of the file currently being operated on by the <code>fbx_fusebox30</code> file the same as the home circuit app?
<code>Fusebox.isTargetCircuit</code>	boolean	No	Is the circuit context/directory being dealt with that of the target circuit?
<code>Fusebox.fuseaction</code>	String	No	The simple fuseaction extracted from the request.
<code>Fusebox.circuit</code>	String	No	The simple circuit extracted from the request.
<code>Fusebox.homeCircuit</code>	String	No	The circuit alias of the home circuit of the app.
<code>Fusebox.targetCircuit</code>	String	No	The circuit alias of the target circuit of the app. Usually this is the same as <code>fusebox.circuit</code> unless you've made a circuits definition error.
<code>Fusebox.thisCircuit</code>	String	No	The circuit alias of the directory of the file currently being operated on by the <code>fbx_fusebox301</code> file.
<code>Fusebox.thisLayoutFile</code>	String	Yes	The layout file to be applied to the current circuit after the fuseaction and its fuse(s) are done creating their content.
<code>Fusebox.thisLayoutDir</code>	String	Yes	The relative path from the target circuit where the layout file to be applied to this circuit is located. If no special layout directory has been created, this should be set to an empty string.
<code>Fusebox.currentPath</code>	String	No	The relative directory path of the file currently being operated on by <code>fbx_fusebox30</code> , relative from the root directory of the application (the home circuit). Example: <code>dir1/dir2/dir3</code>
<code>Fusebox.rootPath</code>	String	No	The relative directory path of the file currently being operated on by <code>fbx_fusebox30</code> , relative to the root directory of the application (the home circuit). Example: <code>../..</code>

Fusebox.layout	String	No	This is the variable used by the saveContent tag (by default) or its equivalent custom tag that captures the generated content to that point in preparation for wrapping a layout file around it (as defined by fusebox.layoutFile). This variable must be inside each layout file in order for content to be passed up to the next level of nested layouts.
Fusebox.suppressErrors	boolean	Yes	Defines whether or not fbx_fusebox30 should suppress native errors and instead give its best guess at what is wrong with your application (as it relates to Fusebox). Default value is always false, which therefore generates native compilation errors.
Fusebox.attributes	java.util. Properties	Yes	The structure to hold the attributes (query string and form variable attributes) of a request.
Fusebox.circuits	java.util. Properties	Yes	The structure to hold the circuit mappings as defined in the fbx_circuits file.
Fusebox.FUSEBOX	String	No	Constant string of the parameter defining the Fusebox object in the request scope.
Fusebox.FUSEACTION	String	No	Constant string of the parameter defining the fuseaction in a request (and in Fusebox.attributes).
Fusebox.FUSEBOX_LAYOUT	String	No	Constant string defining the default parameter identifying the layout content held in the request scope.

Fusebox Classes

The following classes are accessible from the j2ee-fusebox.jar file. For more information on the J2EE Fusebox classes, please review the org.fusebox package JavaDocs.

Class	Description
org.fusebox	
Fusebox	The <code>Fusebox</code> class is the base class of the Fusebox API. For each and every Fusebox request a <code>Fusebox</code> instance is created, stored as a <code>ServletRequest</code> attribute, and used in the process of constructing a request response.
Fusebox_	The <code>Fusebox_</code> class is a utility structure encompassing "private" variables used by the underlying Fusebox framework. A <code>Fusebox_</code> instance is created and used by each instance of a <code>Fusebox</code> class created during a request.
org.fusebox.taglib	
DispatchTag	The class behind the <code>fbx:dispatch</code> tag.
FuseboxBodyTagSupport	The base tag class to be extended by Fusebox tags that need body support.
FuseboxBodySupport	The base tag class to be extended by Fusebox tags that do not need body tag support.
FuseboxTagUtils	Utility class used by many of the Fusebox tag classes.
FusedocTag	The class behind the <code>fbx:secure</code> tag. A tag class used for containing and processing a page's XML Fusedoc information.
QuerySimResults	A simple implementation of <code>Result</code> that allows the user to populate the data by passing in a 2D object array.
QuerySimTag	The class behind the <code>fbx:querySim</code> tag.
SaveContentTag	The class behind the <code>fbx:saveContent</code> tag. A tag class used for surrounding output meant to be stored and potentially displayed to a Fusebox user.
SecureTag	The class behind the <code>fbx:secure</code> tag.
SwitchCaseTag	The class behind the <code>fbx:case</code> tag. A tag class for representing a case of switch-statement meant to be used in conjunction with a <code>SwitchTag</code> .
SwitchTag	The class behind the <code>fbx:switch</code> tag. A tag class for constructing a switch-statement. This class is used, in a JSP context, in conjunction with instances of <code>SwitchCaseTags</code> .

org.fusebox.util	
FuseboxDispatcher	Dispatches includes and forwards in a Fusebox safe manner.
FuseboxStack	A stack class implementation used by the core fbx_fusebox.jsp file.
org.fusebox.security	
ListSecurityModel	An implementation of SecurityModel used by the fbx:secure tag.
SecurityModel	SecurityModel is a generic security model interface to validate a user's roles against a required set of roles.

Fusebox Tag-Library

The J2EE Fusebox framework requires the use of some custom JSP tags for operation. These tags are also available for general development use. A brief overview of the standard tags is provided below.

Name	fbx:fusedoc
Class	org.fusebox.taglib.FusedocTag
Body	Tag-dependent
Description	Tag for surrounding Fusedoc XML. This tag now does little more than suppressing the output (not printing out the fusedoc to the user), but is defined to be extended for any of several possible purposes.
Attributes	none

Example usage of the fbx:fusedoc tag in a display-fuse:

```
<fbx:fusedoc>
<fusedoc fuse="dsp_Countries.jsp">
  <responsibilities>
    I display a list of countries.
  </responsibilities>
  <properties>
    <history author="Fuseboxer" date="2002 May 11" type="create" />
  </properties>
  <io>
    <in>
      <string name="self" optional="false" />
    </in>
    <out>
      <string name="fuseaction" scope="request" optional="true" />
    </out>
  </io>
</fusedoc>
</fbx:fusedoc>
```

Name	fbx:saveContent			
Class	org.fusebox.taglib.SaveContentTag			
Body	JSP			
Description	Save content tag familiar to ColdFusion developers. Saves contents, rather than immediately printing it to screen, for later use. The variable attribute determines where the content is stored.			
Attributes	Name	Optional	RTEExpr*	Description
	variable	true	true	The variable (request-scope) to store the

			content in.
append	true	true	Whether or not to append the content to the current content held in the same variable.

Example usage of the fbx:saveContent tag in a layout file:

```
<fbx:saveContent variable="fusebox.layout">
home/TypicalLayout.jsp
<hr>
<%= fusebox.layout %>
<hr>
</fbx:saveContent>
```

Name	fbx:switch			
Class	org.fusebox.taglib.SwitchTag			
Body	JSP (should contain one or more SwitchCaseTags)			
Description	General switch statement where one case (at most) is evaluated. If no 'value' attribute is specified, then the cases will be evaluated on the boolean value of their 'expr' attributes.			
Attributes	Name	Optional	RTEExpr*	Description
	value	true	true	String value to switch on.

Example usage of the fbx:switch tag in an fbx_switch file:

```
<fbx:switch value="<%= fusebox.fuseaction %>">
  <fbx:case expr="true">
    ...
  </fbx:case>
</fbx:switch>
```

Name	fbx:case			
Class	org.fusebox.taglib.SwitchCaseTag			
Body	JSP			
Description	A case for a switch tag. Relies on being a child of a switch tag. Depending on the evaluation mode of the parent switch tag (expression or value), either the 'value' or the 'expr' tag will be evaluated to determine executing. If another case of the same switch has been executed, another will not. Even if a switch is evaluating on value mode, if a true 'expr' attribute is defined while no 'value' attribute is, the case will be executed.			
Attributes	Name	Optional	RTEExpr*	Description
	value	true	true	
	expr	true	true	

	expr	true	true	
--	------	------	------	--

Example usage of the fbx:case tag in an fbx_switch file:

```
<fbx:switch value="<%= fusebox.fuseaction %>">
  <fbx:case value="doThis">
    ...
  </fbx:case>
  <fbx:case expr="true">
    ...
  </fbx:case>
</fbx:switch>
```

Name	fbx:dispatch			
Class	org.fusebox.taglib.DispatchTag			
Body	Empty			
Description	This tag dispatches includes and forwards in a Fusebox safe way. To avoid merging of request parameters when including and forwarding the dispatch tag will parse url attributes and store them in a place where the Fusebox core file knows where to look for them.			
Attributes	Name	Optional	RTEExpr*	Description
	action	true	true	Type of action to dispatch (forward or include).
	url	true	true	The Fusebox compliant url to dispatch to.
	fuseaction	true	true	The Fuseaction to dispatch to.
	xfa	true	true	The XFA to dispatch to. The attribute value given should be a JSP scope key for the Fuseaction.
	self	true	true	The Fusebox self value (usually index.jsp). If this is not provided, it will be searched for in the JSP scopes.

Example usages of the fbx:dispatch tag in an action-fuse:

```
<fbx:dispatch url="/index.jsp?fuseaction=home.showWelcome" action="include"/>
<fbx:dispatch fuseaction="home.showWelcome" action="forward" />
<fbx:dispatch xfa="XFA_welcome" self="/index.jsp" action="forward" />
```

Name	fbx:secure
Class	org.fusebox.taglib.SecureTag
Body	JSP (or Empty)

Description	Secures a section of code. Authroize execution with tag attributes to verify against a provided security model (org.fusebox.security.SecurityModel).																							
Attributes	<table border="1"> <thead> <tr> <th>Name</th> <th>Optional</th> <th>RTEExpr*</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>userPermissions</td> <td>false</td> <td>true</td> <td>A comma-delimited String list or java.util.List of the user's permissions.</td> </tr> <tr> <td>requiredPermissions</td> <td>false</td> <td>true</td> <td>A comma-delimited String list or java.util.List of the required permissions.</td> </tr> <tr> <td>model</td> <td>false</td> <td>true</td> <td>The org.fusebox.security.SecurityModel implementation to use.</td> </tr> <tr> <td>failureXFA</td> <td>true</td> <td>true</td> <td>The url to dispatch to in the event of a security failure.</td> </tr> </tbody> </table>				Name	Optional	RTEExpr*	Description	userPermissions	false	true	A comma-delimited String list or java.util.List of the user's permissions.	requiredPermissions	false	true	A comma-delimited String list or java.util.List of the required permissions.	model	false	true	The org.fusebox.security.SecurityModel implementation to use.	failureXFA	true	true	The url to dispatch to in the event of a security failure.
Name	Optional	RTEExpr*	Description																					
userPermissions	false	true	A comma-delimited String list or java.util.List of the user's permissions.																					
requiredPermissions	false	true	A comma-delimited String list or java.util.List of the required permissions.																					
model	false	true	The org.fusebox.security.SecurityModel implementation to use.																					
failureXFA	true	true	The url to dispatch to in the event of a security failure.																					

Example usage of the fbx:secure tag in an fbx_switch file:

```
<fbx:saveContent variable="fusebox.layout">
<hr>
<%= fusebox.layout %>
<hr>
</fbx:saveContent>
```

Name	fbx:querySim																			
Class	org.fusebox.taglib.QuerySimTag																			
Body	JSP (or Empty)																			
Description	Simulates a JDBC query by creating java.sql.Recordset with the supplied data.																			
Attributes	<table border="1"> <thead> <tr> <th>Name</th> <th>Optional</th> <th>RTEExpr*</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>var</td> <td>false</td> <td>true</td> <td>The name to store the created data in.</td> </tr> <tr> <td>scope</td> <td>true</td> <td>true</td> <td>The scope to store the created data in.</td> </tr> <tr> <td>data</td> <td>false</td> <td>true</td> <td>The data to use.</td> </tr> </tbody> </table>				Name	Optional	RTEExpr*	Description	var	false	true	The name to store the created data in.	scope	true	true	The scope to store the created data in.	data	false	true	The data to use.
Name	Optional	RTEExpr*	Description																	
var	false	true	The name to store the created data in.																	
scope	true	true	The scope to store the created data in.																	
data	false	true	The data to use.																	

Example usages of the fbx:querySim tag in a query-fuse:

```
<% Object[][] queryData = { {"name", "password"}, {"abc", "123"} }; %>
<fbx:querySim var="userData" scope="request" data="<%=queryData%>" />
```

*RTEExpr – Request-Time Expression Value allowed

J2EE Fusebox in Action

The Life of a Fusebox Request

The processing of a Fusebox request is handled primarily by the fbx_fusebox.jsp file. This file is divided into eleven sections, described below.

1. Fusedoc

Here the Fusedoc of the fbx_fusebox301 file is specified.

2. Initialization of Fusebox

The Fusebox class encompasses the public Fusebox API. An instance of Fusebox created for each Fusebox request and placed in the request scope. It is recommend that you make no direct changes to the variables of the request instance. After creation the Fusebox instance is pushed onto the Fusebox stack.

3. Initialization of inner Fusebox

The Fusebox_ class encompasses "private" variables used by the underlying Fusebox framework. A Fusebox_ instance is created for each instance of Fusebox. Make no changes to it without a full understanding of the ramifications of those changes.

4. Initializations of Fusebox attributes

In this section all the request parameters are set in the Fusebox instance attributes structure. If the dispatch attributes parameter is present in the request, the stored attributes will be used rather than being read from the request parameters.

5. Inclusion of fbx_circuits

In this section the home circuit's fbx_circuits file is included with the <jsp:include> JSP tag. The fbx_circuits JSP populates the circuit structure of the fusebox instance in the request.

6. Create reverse mapping of Circuits

This section creates a reverse path look-up of the fusebox circuits structure which can be used later to conveniently look up the circuit alias of whichever circuit is being accessed at that moment, particularly when determining fusebox.thisCircuit. The reverse paths are set in the fusebox.FB_.reverseLookupPath structure.

7. Inclusion of fbx_settings

In this section the home circuit's fbx_settings file is included, if present. Executed fbx_settings JSPs set variables in the fusebox instance held in the request.

8. Dissection and interpreting of fuseaction, begin the aliased lookup process

Using fusebox.attributes, this section determines if the fuseaction parameter is a compound fuseaction (i.e. it only includes the circuit in the form of "?fuseaction=circuit."). If the fuseaction is not compounded, then set the fuseaction is set as blank, meaning the target circuit's default case should execute. The circuit alias is then used to lookup the target circuit in the fusebox.circuit structure for the full path to the circuit.

9. Inclusion of fbx_settings in nested (top-to-bottom) order

This section attempts to include any nested fbx_settings files, in top-to-bottom order, so that variables set in children fbx_settings files can overwrite variables set in higher fbx_settings files. If any fbx_settings.cfm file or directory alias cannot be found, processing continues on.

10. Inclusion of the target fbx_switch file, processing of the fuseaction

This section "reaches down" and includes the fbx_switch in the target circuit, executing switch-case matching based on the fusebox.fuseaction variable. The output of executed fuses are typically stored for later output using the <fbx:saveContent> tag.

11. Inclusion of layout files in nested (top-to-bottom) order, final output of page

Now the layouts are handled, resolving them in bottom-to-top order to nest the circuits, if needed. Also set is fusebox.thisCircuit, equal to the circuit name of the current circuit the code is passing through, which will let any layout files in circuits know where they are. If attempting to include any layout file throws an error, then processing continues on. This entire section and functionality of nesting layouts and controlling layouts via layout files is optional. If specific layout files are not specified some or all circuits, including the home, everything should still work. After processing is complete, the Fusebox stack is popped.

J2EE Fusebox Resources

There are a growing number of tools and resources available for J2EE Fusebox developers.

Fusebox.org – www.fusebox.org

Fusebox.org is the home for the Fusebox community.

SourceForge – www.sourceforge.net/projects/j2ee-fusebox

SourceForge.net is the world's largest open source development website and current home the J2EE Fusebox development project. The J2EE Fusebox project web site provides several useful resources for developers, such as documentation, sample applications, and the latest code releases. It's also the best place to learn how to get involved in the J2EE Fusebox development effort.

Synthis.com – www.synthis.com

Synthis Corporation has developed a new class of software development tool that helps automate the design, development and integration of Internet-based software applications across a broad range of development platforms and technologies. Synthis' newest product, Adalon, is an analysis, modeling and design tool that helps unify key software development functions. It is the first tool built from the ground up to support business process design for Internet-based software applications, and the only commercial tool capable of generating best-practice based code frameworks for specific software architectures and programming languages, including ColdFusion Fusebox and J2EE Fusebox.

Appendix A: J2EE Fusebox and JSTL

Coming soon...