FAQ

{scrollbar}

Frequently Asked Questions

NOTE: THE DOCUMENTATION HAS BEEN MOVED TO http://myfaces.apache.org/wiki/core/faq.html

Note that in addition to this FAQ, there are a number of "how-to" guides on the MyFaces wiki home page that address common issues.

42

What is MyFaces?

MyFaces is a family of projects related to the JavaServer Faces (JSF) specification published as part of the Java Community Process. The "core" project is an implementation of that specification. Other MyFaces projects implement related specifications (eg the Portlet Bridge), or add features to any JSF implementation (not just the Myfaces Core).

JSF FAQ

Are there any other FAQ sites for JSF?

There is an excellent FAQ addressing general JSF issues at www.jsfcentral.com. There are a number of other places where such information is available; just use the web's search engines!

JSF SPEC

I don't like something about the JSF specification. How can I change it?

You can create issues on JAVASERVERFACES-SPEC-PUBLIC to send your suggestion for the next version of JSF. This will be received by the JSR spec leads, and if considered reasonable will be added to the project issue tracker.

Additionally, you can suscribe to MyFaces Users and Dev lists and send your suggestions there. In this way, a MyFaces PMC member could help to include that feature on the Expert Group.

What is the difference between javax.faces.STATE_SAVING_METHOD client and server?

In short, server-side-state stores information held by the UI components (ie the "widgets") in the HTTP Session, while client-side state stores it in hidden fields in the page sent back to the user.

Client-side-state scales much better for very large numbers of users, as no memory is required on the server for user state. However the disadvantage is that more data gets transferred across the network for each request.

Any session-scoped managed beans still remain in the http session, even with client-side-state selected; this flag only affects where the JSF implementation stores data that the UI components hold internally.

For the technical details, see:

- http://java.sun.com/javaee/5/docs/tutorial/doc/bnavu.html#bnavz
- http://java.sun.com/javaee/5/docs/tutorial/doc/bnaxj.html#bnaxn

The Browser always shows a link to the previous page. Why?

By default, JSF uses "forward" operations internally to navigate between pages.

So a user first views page A, and gets back some content. The content will be wrapped in a form that has page A as the address to submit to.

When the user then does something to cause that page to submit (eg clicks on a "save" button), then JSF receives that post request, restores the view for page A and performs postback processing. As part of the postback processing, the logic for page A may tell the JSF framework that page B should now be shown.

By default, JSF will then perform an internal forward to page B, causing that page to be rendered back to the user as the result of their form post.

The user now sees page B on their screen. However all the browser knows is that it sent some data to page A, so that is the url that it will show in the browser navigation bar even though the content on display is page B. Unfortunately HTTP/HTML provide no way to tell the browser that it should display the url for page B in the browser. Therefore with JSF default behaviour, the browser url is always "one page behind" the content actually being shown.

Not only does this look odd to users, but it makes bookmarking pages within a JSF application difficult. Note, however, that bookmarking arbitrary places within JSF applications (which are usually very interactive and stateful) is not always sensible anyway.

Another disadvantage of this "optimisation" is that it is not safe to use relative links anywhere within a JSF view (eg relative path to a stylesheet or icon). When a browser posts to page A, and gets back content for page B it will resolve any relative links in the returned page relative to the last url it knew about. So on the first view of page B, all relative links are relative to A; if A is in a different directory then all relative links in the view definition for page B are broken.

One solution is to use navigation rules marked with <redirect/>. Then, instead of performing an internal forward to the new page, an http redirect command is sent to the browser, causing it to then fetch the new page. This is less efficient, however, as the browser now has to make a second request to get its content, instead of just getting the content immediately as a response to its original form posting. Another serious issue is that because the fetch of page B is in a separate request, it is impossible for page A to pass any data to page B via request-scoped variables. Using request-scoped variables for this purpose is quite convenient and therefore quite common.

Myfaces FAQ

Why should I use MyFaces Core?

"... Every Java EE application server comes with a bundled JSF implementation. Often the reference implementation (RI), known as "Mojarra", is used for this. However, you can bundle an alternative JSF implementation to be used with your project. This way, you can use MyFaces Core as JSF implementation on every Java EE application server. One of the benefits of using MyFaces Core is that it generates more informative logging, which can be helpful while debugging. If you're looking for an application server that uses MyFaces Core by default, you should take a look at Apache Geronimo..." Th

anks to Bart Kummel for provide this text from its book Apache MyFaces 1.2 Web Application Development

Other reason why choose MyFaces core are:

- Community over code: MyFaces community counts with a lot of folks with outstanding knowledge on JSF. Suscribe to user and dev mailing lists are the best way to know what's going on, receive feedback and know other people interested in JSF. This is an important consideration, because sometimes you can find a bug or need to ask something to somebody about JSF.
- Innovation happens on MyFaces: MyFaces is not just a JSF implementation, it host other projects (Trinidad,Tobago, Tomahawk, ExtVal, CODI, Orchestra, PortletBridge RI, ...) that provides additional functionality not provided by JSF implementation itself.
- MyFaces Core generates code using myfaces-builder-plugin: these project is a maven plugin that with myfaces-builder-annotations provides a jsf development kit to create multiple artifacts (components, converters, validators, client behaviors) and handle all documentation related in an easier way, minimizing the number of files you need to handle. It uses java annotations to feed all related information, and each myfaces jar using this tool has a myfaces-metadata.xml which contains all meta-information required to build each bundle. This tool can read composite component definitions and build an integrated documentation with your normal components, converters or validators.
- MyFaces uses Maven, so edit the code or rebuild it is as simple as open any maven project with your IDE. Note that most IDEs has built-in support for maven.
- MyFaces Core is "OSGi" friendly: It provides some SPI interfaces to deal with special setups, when you need core control over classloading. A lot of good stuff has been put in this area.
- MyFaces Core is tested everyday against MyFaces UI component libraries and Add-ons/Extensions. Many of these projects are compatible with both MyFaces and Mojarra (because they are build on top of JSF standard), but you should note that those projects helps MyFaces Core to keep code in good shape, because all those projects test against MyFaces Core, and if there is a bug, it is handled more quickly (because everything is handled "in home").

For more technical reasons why choose MyFaces core take a look at this blog.

I have a problem. Should I open a JIRA issue on it or ask about it on the mailing list?

Please ask all questions on the !MyFaces users list.

As as guideline, only create a JIRA issue if

- The same code works in another implementation (eg Sun Mojarra) but not in !MyFaces.
- The same code works in another version of !MyFaces.
- You've examined the !MyFaces source code and have come up with a patch that fixes the behavior.
- You've asked on the mailing list, and someone has confirmed it's a bug and asked you to open a JIRA issue.

In all other cases, ask first on the users mailing list.

How can I donate new components?

There are various ways, you can send a note in the mailing list of !MyFaces for direct contact with the developers and guidance on what to do, you can put your component into Tomahawk issue tracker. There are no strict rules for the participation at the sandbox, all you need is some good components.

Where can I download source code jars?

Source for released versions of MyFaces Core can be found here:

- http://svn.apache.org/repos/asf/myfaces/core/tags
- http://svn.apache.org/repos/asf/myfaces/shared/tags

Note that MyFaces Core 1.1.x releases (ie implementations of JSF 1.1) include code from the shared project versions 2.0.x. MyFaces Core 1.2.x releases (ie implementations of JSF 1.2) include code from the shared project versions 3.0.x.

Since version 2.0.8 and 2.1.2, shared code was bundled as a module in core, so you only need to checkout its related tags folder.

Source for Tomahawk

- http://svn.apache.org/repos/asf/myfaces/tomahawk/tags/
- http://svn.apache.org/repos/asf/myfaces/shared/tags/ (choose 2.0.x tags)

You can also find builds at this location:
http://people.apache.org/builds/myfaces/

Sources jars usable in IDEs are available from the maven2 repository directories:

- http://www.ibiblio.org/maven2/org/apache/myfaces/core/
- http://www.ibiblio.org/maven2/org/apache/myfaces/shared/

Where can I download a sandbox jar?

Binaries of not-yet-released code can be grabbed from the maven build repositories.

- https://repository.apache.org/content/repositories/snapshots/org/apache/myfaces/

For example, the sandbox jar is available beneath here:

- https://repository.apache.org/content/repositories/snapshots/org/apache/myfaces/tomahawk/tomahawk-sandbox20/
- https://repository.apache.org/content/repositories/snapshots/org/apache/myfaces/tomahawk/tomahawk-sandbox12/
- https://repository.apache.org/content/repositories/snapshots/org/apache/myfaces/tomahawk/tomahawk-sandbox/

What is the "shared" project?

If myfaces-core, tomahawk, tobago and trinidad were completely separate projects, then there would be no need for a "shared" code project; each project would maintain its own code in its own namespace. However there would be lots of code duplication and wasted effort. As these projects are all under the "myfaces" umbrella, code that is common to multiple projects can be placed into the "shared" project thus reducing development and maintenance effort, and making the projects more consistent with each other.

However these projects all have separate release cycles, and in addition tomahawk/tobago/trinidad should be runnable on JSF implementations other than myfaces-core. The solution currently in use is a compilation hack that renames the shared class package; the code is renamed to "org.apache.myfaces.shared_impl" for inclusion in the myfaces-impl jarfile, "org.apache.myfaces.shared_tomahawk" for inclusion in the tomahawk jarfile etc. In this manner, each project can be deployed with all the shared support classes it needs (with the right features and fixes) without needing a separate shared jarfile, and without worrying about conflicting definitions of the same class. An upgrade to one of these projects won't affect any other deployed in the same environment.

The shared code has release numbers that are separate from the release numbers of projects that use it. For example,
http://svn.apache.org/repos/asf/myfaces/tomahawk/tags/1_1_3/core/pom.xml
shows that tomahawk 1.1.3 has a dependency on version 2.0.2 of the shared project.

Note that file myfaces-impl-1.1.4.jar includes the .class files for the shared project, but myfaces-impl-1.1.4-sources.jar does not include the matching source files; myfaces-shared-impl-2.0.3.jar needs to be added to access the java source. Recent releases of MyFaces projects that use the shared lib now include the source from the shared project in their -sources jars so adding an extra sources jar is not necessary.

Since version 2.0.8 and 2.1.2, shared code was bundled as a module in core, to make easier work with MyFaces Core.

How can I get pretty-formatted HTML output from Myfaces?

The [http://jtidy.sourceforge.net] project provides a [http://jtidy.sourceforge.net/multiproject/jtidyservlet/filter.html] that reformats response messages before they are sent out.

The Mozilla Firefox web browser has an optional extension "[https://addons.mozilla.org/extensions/moreinfo.php?id=697]" which allows developers to see source correctly formatted without any changes at the server end.

In some versions of MyFaces there is a context param "org.apache.myfaces.PRETTY_HTML" that can be specified in the web.xml to enable pretty output. However this option was never well supported as it requires support from every renderer in order to work. This feature may be removed from future versions of MyFaces.

What do version numbers mean for the MyFaces Core and Tomahawk releases?

MyFaces Core uses 3-part version numbers, eg 1.1.1. However this value is different from the usual version numbering. The first two digits indicate which JSF specification the release is an implementation of. Because the binary api for a JSF specification does not change, two releases with the same first two digits are guaranteed to be binary compatible; all existing code using JSF-specified features will continue to work as previously (unless the previous behaviour was buggy).

The Tomahawk library also uses the same format of version number. However as the JSF 1.2 specification is backwards-compatible with the JSF 1.1 specification, Tomahawk releases with version numbers 1.1.x also work fine with JSF 1.2. Note however that Tomahawk releases are not guaranteed to be binary compatible with he previous Tomahawk release; an increment in the third digit is simply a release-count and *"not"* a "patch version". There are no releases of Tomahawk that are simply "bugfix" releases

Note in addition that Tomahawk releases are only tested with the latest versions of the MyFaces and Sun JSF implementations available at the time of release. success with other versions of these libraries, or with other libraries is likely but not guaranteed. In particular, when running early releases of Tomahawk with MyFaces Core, the versions used generally had to match exactly (due to the way the "shared" library was used). This has since been improved, and recent Tomahawk releases have a good chance of working with multiple versions of MyFaces Core.

Why isn't javax.faces.model.DataModel serializable?

The DataModel class (used with UIData components) doesn't have any state that needs to be preserved between render and restore-view phases. Therefore, there is no need for it to be serializable.

If you have a managed bean that you want to be serializable, and there is a member of type DataModel then just mark it transient.

If you want to serialize the list within the !DataModel along with the managed bean, then do this:

solidSomeManagedBean.javapublic class SomeManagedBean implements Serializable { private List myData; private transient DataModel myDataModel; public DataModel getDataModel() { if (myDataModel == null) { myDataModel = new ListDataModel(myData); } return myDataModel; } }

Why are my dates displaying the wrong day/time?

The JSF specification requires that date->string converters default to using the standard UTC (aka GMT) timezone.

Note that MyFaces releases 1.1.0 and earlier did not comply with the JSF spec; they defaulted to using the timezone of the server.

You can control the timezone used by the conversion by attaching an explicit converter:

xml <f:convertDateTime timeZone="Antarctica/South_Pole" .../>

or

xml <f:convertDateTime timeZone="#{bean.timeZone}" .../>

where #{bean.timeZone} returns either a string id, or a TimeZone instance.

The MyFaces commons converters project contains a custom converter tag which is like f:convertDateTime, but defaults to using the timezone of the server:

xml <mcc:convertDateTime/>

Alternatively you register your own converter to override the standard converter, causing your custom code to be applied by default to all date->string conversions.

How can I access one Managed Bean from another?

See Accessing one managed bean from another

Handling forward references to components in JSP pages

Note: this information applies only to projects using JSF 1.1 and JSP. Projects using either JSF 1.2 or Facelets can ignore this section.

In some cases, a component in a JSP page needs to reference another component by id. One common example is the tomahawk t:dataScroller component. This is fine when the referencing component is later in the page than the one it refers to. But when the components are in the wrong order, an error is reported.

xml <t:dataScroller for="someTable" .../> <t:dataTable id="someTable" .../>

When using JSP, creating and rendering of components is done in a single pass through the JSP page. This means that when the dataScroller is rendered, the dataTable it refers to has not yet been created, causing an error message about "cannot find UIData" to be displayed. Other components that refer to some associated component using a "for" attribute or similar mechanism will have this problem too.

This can be resolved by wrapping the components in a parent component that "renders its children". Such components cause their nested components to be processed in two passes (create then render):

xml <h:panelGroup> <t:dataScroller for="someTable" .../> <t:dataTable id="someTable" .../> </h:panelGroup>

This does have a few minor drawbacks. In JSF 1.1, components that render their children interact badly with nested "template text" and non-JSF JSP tags. Avoid non-JSF JSP tags within such components, and wrap any "template text" in f:verbatim tags to resolve this problem.

An alternative solution is to move to a presentation technology that uses separate "create" and "render" passes; Facelets and Clay both do this.

How do I know when a managed bean's properties have all been set?

A managed bean must have a default constructor (ie one with no parameters). Any managed-property declarations then cause the appropriate setter methods to be called. However it's common to want to perform some initialisation once "all" of the bean's properties have been defined.

Spring provides a post-initialisation callback; any bean implementing !InitializingBean gets its afterPropertiesSet method called. There isn't any exact equivalent in JSF, but something close can be achieved by:

- defining a setter method on the bean, eg "public void setInitialized(boolean state)"
- defining this as the last managed property for the bean:

xml <managed-bean> .... <managed-property> <property-name>initialized</property-name> <value>true</value> </managed-property> </managed-bean>

The JSF spec requires that managed properties are initialised in the order they are declared, so the setInitialized method will be called only after all other properties have been set.

In future versions of JSF, it may be possible to annotate managed bean methods with @!PostConstruct and @!PreDestroy to perform this functionality. See:

- https://javaserverfaces-spec-public.dev.java.net/issues/show_bug.cgi?id=119

"My PhaseListener is called twice"

The JSF specification requires any JSF implementation to automatically load /WEB-INF/faces-config.xml at startup. There is consequently no need for the following context parameter:

xml<context-param> <param-name>javax.faces.CONFIG_FILES</param-name> <param-value>/WEB-INF/faces-config.xml</param-value> </context-param>

Putting this context parameter in your deployment descriptor will force any JSF implementation to load the configuration twice, therefore registering each phase listener twice.

If you are using portlets, see [http://issues.apache.org/jira/browse/MYFACES-1338]

"Action listeners and actions for my commands on dataTables do not fire"

Action listeners and actions are not invoked when the action source (h:commandLink, h:commandButton) is not rendered. When our action sources are on a dataTable, and the value attribute of the dataTable points to a request scoped data source, the action source just isn't rendered on a subsequent request.

xml<h:dataTable value="#{requestScopedBean.dataModel.wrappedData}" /> <h:column> <h:commandLink value="click here" action="#{backingBean.willNotFire}" /> </h:column> </h:dataTable>

The action source (h:commandLink, h:commandButton), is not rendered because the data source does not exist during a subsequent request (it was garbage collected after the first response was completed).

To solve this problem, use t:saveState or put the request scoped backing bean in session scope.

xml<t:saveState value="#{myRequestScopedBean.dataModel.wrappedData}" />

t:saveState is preffered over a session scoped solution.

Calendar, Tree, etc wont work or javascript errors

You have to configure the MyFacesExtensionsFilter - see: [http://myfaces.apache.org/tomahawk/extensionsFilter.html]

Error "ExtensionsFilter not correctly configured."

If you are getting the error "java.lang.!IllegalStateException: !ExtensionsFilter not correctly configured. JSF mapping missing. JSF pages not covered", and everything is well configured:

1. Make sure that you have the correct filters as explained in [http://myfaces.apache.org/tomahawk/extensionsFilter.html]
2. If you are using servlets 2.4, you cannot do a jsp:forward or request.getDispatcher().forward to any page, because the extensions filter is not executed. Instead you may want to try with a response.sendRedirect()

The Tomahawk ExtensionsFilter is not working on Websphere

You may need to set the special websphere property com.ibm.ws.webcontainer.invokefilterscompatibility to true.

It looks like Websphere does not run filters when a request url does not map to a servlet *and* does not map to a file on disk.
As Tomahawk !ExtensionFilter urls do neither of these, Websphere simply doesn't run the filter. Setting the above property to true causes Websphere to act like every other servlet engine on planet earth, and run the filters in all cases.

Using the tomahawk :popup tag causes java.lang.NullPointerException in HtmlPopupRenderer.encodeEnd

The facet name needs to be hard coded "popup".

ClassCastException in com.sun.facelets.tag.jsf.ActionSourceRule when using the tomahawk dataScroller

Update to the latest Tomahawk jar from http://myfaces.apache.org/download.html

Using the tomahawk you get java.lang.NoClassDefFoundError: org/apache/commons/lang/builder /HashCodeBuilder

Download the latest jakarta commons jars (inside the MyFaces Core 1.1.3 or later distribution) from http://myfaces.apache.org/download.html

WARNING: The AdfFacesFilter has not been installed. ADF Faces requires this filter for proper execution.

Use a servlet-name in web.xml: adfFaces Faces Servlet ... where you've assigned "Faces Servlet" as the name of the FacesServlet as below Faces Servlet javax.faces.webapp.FacesServlet 1

The ADF component doesn't popup anything or submit (no javascript working). Error: submitForm is not defined

You need to define how the ADF resources (such as .js files and images) are provided in web.xml resources oracle.adf.view.faces.webapp. ResourceServlet resources /adf/*

The MyFaces component doesn't popup anything or submit (no javascript working).

You need to define how the MyFaces resources (such as .js files and images) are provided in web.xml with a MyFacesExtensionsFilter entry.

NullPointerException in adf faces at the start of the render phase OR rendering warnings

Mar 21, 2006 12:10:15 AM org.apache.myfaces.renderkit.html.HtmlRenderKitImpl getRenderer WARNING: Unsupported component-family/renderer-type: oracle.adf.Panel/oracle.adf.Group Mar 21, 2006 12:10:15 AM oracle.adf.view.faces.component.UIXComponentBase _getRendererImpl WARNING: Could not find renderer for CorePanelGroup[UIXFacesBeanImpl, id=_id38], rendererType = oracle.adf.Group Mar 21, 2006 12:10:15 AM org.apache.myfaces. renderkit.html.HtmlRenderKitImpl getRenderer WARNING: Unsupported component-family/renderer-type: oracle.adf.Command/oracle.adf.Link Mar 21, 2006 12:10:15 AM oracle.adf.view.faces.component.UIXComponentBase _getRendererImpl WARNING: Could not find renderer for CoreCommandLink [CommandLinkFacesBean, id=_id41], rendererType = oracle.adf.Link Mar 21, 2006 12:10:15 AM org.apache.myfaces.renderkit.html.HtmlRenderKitImpl getRenderer WARNING: Unsupported component-family/renderer-type: oracle.adf.Output/oracle.adf.Formatted SEVERE: Error Rendering View[/main. xhtml] java.lang.NullPointerException at oracle.adfinternal.view.faces<WBR>.uinode.FacesRenderingContext<WBR>.setRenderingProperty (FacesRenderingContext.java:157) at oracle.adfinternal.view.faces<WBR>.ui.RenderedNodeRenderingContex<WBR>t.setProperty (RenderedNodeRenderingContext<WBR>.java:162) at oracle.adfinternal.view.faces<WBR>.ui.RootRenderingContext.init(RootRenderingContext.java:81) at oracle.adfinternal.view.faces<WBR>.uinode.FacesRenderingContext. (FacesRenderingContext.java:106) at oracle.adfinternal.view.faces<WBR>.uinode. FacesRenderingContext<WBR>.createRenderingContext(FacesRenderingContext.java:79) at oracle.adfinternal.view.faces<WBR>.uinode. UINodeRendererBase<WBR>.getRenderingContext(UINodeRendererBase.java:89) at oracle.adfinternal.view.faces<WBR>.uinode. FacesRenderingContext<WBR>.getRenderingContext(FacesRenderingContext.java:66) at oracle.adfinternal.view.faces<WBR>.uinode. FacesRenderingContext<WBR>.getRenderingContext(FacesRenderingContext.java:52) at oracle.adfinternal.view.faces<WBR>.renderkit.htmlBasic. UINodeRend<WBR>erer.getRenderingContext(UINodeRenderer.java:79) at oracle.adfinternal.view.faces<WBR>.renderkit.htmlBasic. UINodeRend<WBR>erer.encodeBegin(UINodeRenderer.java:38) at javax.faces.component.UICompone<WBR>ntBase.encodeBegin(UIComponentBase. java:512) at com.sun.facelets.FaceletViewHan<WBR>dler.encodeRecursive(FaceletViewHandler.java:555) at com.sun.facelets. FaceletViewHan<WBR>dler.encodeRecursive(FaceletViewHandler.java:562) at com.sun.facelets.FaceletViewHan<WBR>dler.encodeRecursive (FaceletViewHandler.java:562) at com.sun.facelets.FaceletViewHan<WBR>dler.renderView(FaceletViewHandler.java:457) at org.apache.myfaces. lifecycle<WBR>.LifecycleImpl.render(LifecycleImpl.java:367) at javax.faces.webapp.FacesServlet<WBR>.service(FacesServlet.java:138) at org.apache. catalina.core<WBR>.ApplicationFilterChain<WBR>.internalDoFilter(ApplicationFilterChain.java:252) at org.apache.catalina.core<WBR>. ApplicationFilterChain<WBR>.doFilter(ApplicationFilterChain.java:173) at org.apache.myfaces.webapp<WBR>.filter.ExtensionsFilter<WBR>.doFilter (ExtensionsFilter.java:130) at org.apache.catalina.core<WBR>.ApplicationFilterChain<WBR>.internalDoFilter(ApplicationFilterChain.java:202) at org. apache.catalina.core<WBR>.ApplicationFilterChain<WBR>.doFilter(ApplicationFilterChain.java:173) at org.apache.catalina.core<WBR>. StandardWrapperValve.invoke(StandardWrapperValve.java:213) at org.apache.catalina.core<WBR>.StandardContextValve.invoke(StandardContextValve. java:178) at org.apache.catalina.authenticat<WBR>or.AuthenticatorBase.invoke(AuthenticatorBase.java:432) at org.apache.catalina.core<WBR>. StandardHostValve.invoke(StandardHostValve.java:126) at org.apache.catalina.valves<WBR>.ErrorReportValve.invoke(ErrorReportValve.java:105) at org. apache.catalina.core<WBR>.StandardEngineValve.invoke(StandardEngineValve.java:107) at org.apache.catalina.connector<WBR>.CoyoteAdapter. service(CoyoteAdapter.java:148) at org.apache.coyote.http11<WBR>.Http11Processor.process(Http11Processor.java:869) at org.apache.coyote. http11<WBR>.Http11BaseProtocol$Http11Conne<WBR>ctionHandler.processConnection(Http11BaseProtocol.java:667) at org.apache.tomcat.util. net<WBR>.PoolTcpEndpoint.processSocket(PoolTcpEndpoint.java:527) at org.apache.tomcat.util.net<WBR>.LeaderFollowerWorkerThread<WBR>.runIt (LeaderFollowerWorkerThread.java:80) at org.apache.tomcat.util.threads<WBR>.ThreadPool$ControlRunnable.run(ThreadPool.java:684) at java.lang. Thread.run(Thread.java:595)

Make sure your FaceletViewHandler is commented out and the default render kit of ADF faces defined in faces-config.xml oracle.adf.core (Note that the ADF faces implementation ensures non-ADF components still get rendered in the normal way so it can safely co-exist with other JSF components) ..and your web.xml contains these lines for ADF faces oracle.adf.view.faces.ALTERNATE_VIEW_HANDLER com.sun.facelets.FaceletViewHandler AdfFacesFilter oracle.adf.view.faces.webapp.AdfFacesFilter AdfFacesFilter FacesServlet

Thanks to Murray Brandon for the initial contribution.

Select Items stop working in MyFaces 1.1.5

MyFaces 1.1.4 and earlier did not correctly implement the select components. Although the behavior was nice, the components were not supposed to convert the values from the UISelectItems. For example, in 1.1.4 this would have worked:

xml<f:selectOneMenu value="#{bean.intValue}"><f:selectItem itemValue="1" /></selectOneMenu>

The reason this used to work is that the code in the past used the converter on the select one menu component to convert the value from the select item. According to the JSF specification, this is not supposed to be done. Therefore, it is important that the converted value from the selectOneMenu is identical (passes the equals function test) to the itemValue from one of the select items. Therefore, for the example above, itemValue must be an integer value, not a string with a number in it.

If you want the old behavior, I suggest creating a custom selectItem component that converts the value using the converter from the input control. Here is the code:

solidUISelectItem.javapublic class UISelectItem extends javax.faces.component.UISelectItem { public final static String COMPONENT_TYPE = "org. apache.myfaces.wiki.SelectItem"; private Boolean convertValue; /** * @return the convertValue */ public boolean getConvertValue() { if (this.convertValue ! = null) return this.convertValue; ValueBinding vb = getValueBinding("convertValue"); return (vb == null) ? true : (Boolean) vb.getValue(getFacesContext()); } /** * @param convertValue the convertValue to set */ public void setConvertValue(boolean convertValue) { this.convertValue = convertValue; } /** * @see javax.faces.component.UISelectItem#getItemValue() */ @Override public Object getItemValue() { Object value = super.getItemValue(); if (getConvertValue()) { UIInput parent = null; for (UIComponent comp = getParent(); comp != null; comp = comp.getParent()) { if (comp instanceof UIInput) { parent = (UIInput)comp; break; } } if (parent != null) value = getConvertedValue(getFacesContext(), parent, value); } return value; } /** * @see javax.faces. component.UISelectItem#saveState(javax.faces.context.FacesContext) */ @Override public Object saveState(FacesContext context) { return new Object[] { super.saveState(context), convertValue, }; } /** * @see javax.faces.component.UISelectItem#restoreState(javax.faces.context.FacesContext, java.lang. Object) */ @Override public void restoreState(FacesContext context, Object state) { Object[] arr = (Object[]) state; int index = -1; super.restoreState (context, arr[++index]); this.convertValue = (Boolean) arr[++index]; } private Object getConvertedValue(FacesContext context, UIInput input, Object value) throws ConverterException { Renderer renderer = getRenderer(context); if (renderer != null) return renderer.getConvertedValue(context, this, value); else if (value instanceof String) { Converter converter = RendererUtils.findUIOutputConverter(context, input); if (converter != null) return converter.getAsObject (context, this, (String)value); } return value; } }

Then just register this new component in the standard JSF way.

{scrollbar}