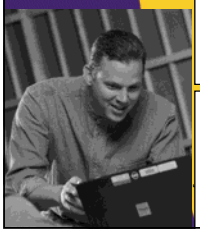
JSF: Installation, Setup, & Getting Started

JSF 2.2 Version

Originals of slides and source code for examples: http://www.coreservlets.com/JSF-Tutorial/jsf2/
Also see the PrimeFaces tutorial – http://www.coreservlets.com/JSF-Tutorial/primefaces/
and customized onsite JSF2 and PrimeFaces training courses – http://courses.coreservlets.com/jsf-training.html

For live training on JSF 2, PrimeFaces, or other Java EE topics, email hall@coreservlets.com
Marty is also available for consulting and development support



Taught by the author of *Core Servlets and JSP*, this tutorial, and JSF 2.2 version of *Core JSF*. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  - JSF 2, PrimeFaces, Ajax, jQuery, Spring MVC, JSP, Android, general Java, Java 8 lambdas/streams, GWT, custom topic mix
  - Courses available in any location worldwide. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  - Hadoop, Spring, Hibernate/JPA, RESTful Web Services

Contact hall@coreservlets.com for details

Topics in This Section

- **Getting required software**
  - Installing Java SE
  - Installing Eclipse (Java EE version)
  - Installing a server for JSF 2
- **Accessing documentation**
- **Testing projects**
  - Importing and testing an existing JSF 2.2 project
    - Deploying on servlet engine (e.g., Tomcat, Jetty) and on Java EE 7 server (e.g., Glassfish 4, JBoss 7)
- **Making your own JSF 2.2 project**
  - Using Eclipse wizard
  - By copying and renaming template project

Overview

Why Web Apps?

- **Downsides to browser-based apps**
  - GUI is poor
    - HTML is OK for static documents, but lousy for programs
  - Communication is inefficient
    - HTTP is poor protocol for the way we now use Web apps

Why Web Apps? (Continued)

- **So why does everyone want Web apps?**
  - Universal access
    - Everyone already has a browser installed
    - Any computer on the network can access content
  - Automatic "updates"
    - Content comes from server, so is never out of date


What is JSF?

- **A set of Web-based GUI controls and handlers?**
  - JSF provides many prebuilt HTML-oriented GUI controls, along with code to handle their events on the server.
- **A device-independent GUI control framework?**
  - JSF can be used to generate graphics in formats other than HTML, using protocols other than HTTP.
- **An MVC-based Web app framework?**
  - Like Apache Struts, JSF can be viewed as an Model-View-Controller framework for building HTML forms, validating their values, invoking business logic, and displaying results.
- **An Ajax library?**
  - JSF 2 provides very easy-to-use Ajax support. So, JSF 2 can be viewed as an alternative to jQuery or GWT.
- **But which is the proper way to view JSF?**
  - The answer depends on what you are going to use it for, but 1 & 3 are the most common ways of looking at JSF.

Overview of JSF 2

- **JSF is the official Java EE library for Web apps**
  - And JSF 2 (usually with a rich component library like PrimeFaces or RichFaces) is the most popular choice in practice.
- **JSF 2 adds many new features vs. JSF 1**
  - Smart defaults
  - Annotations as alternatives to most faces-config.xml entries
  - Integrated Ajax support
  - Facelets instead of JSP
  - Much simpler custom components
  - Ability to bookmark results pages (view parameters)
- **JSF 2.2 adds even more vs. JSF 2.1**
  - Flow scope
  - Stateless views
  - HTML 5 pass-through attributes
  - View actions

Summary: Requirements for Running JSF 2.2

- **Java**
  - Java 7 or Java 8 strongly preferred, but Java 6 technically legal
  - Java EE 7 servers run on top of Java 7, but Java 8 is great option
    - This tutorial uses Tomcat 7 and Java 7, but code will run on any Java server that supports servlets 3.0 or later and uses Java 7 or later. Some Java 8 examples included, but always with Java 7 alternatives.
- **A server**
  - Servlet engine supporting servlets 3.0 or later (by including JSF JAR file)
    - E.g., Tomcat 7, Jetty 9. Servlets 2.5 legal except for file upload component.
      - Using JSF on top of servlet engine instead of Java EE 7 server lets you use Java 8, which is a huge advantage
  - Any Java EE 7 server (no JSF JAR needed)
    - E.g., Glassfish 4, JBoss 8. WebSphere and WebLogic coming soon
- **An IDE**
  - Optional, but highly recommended
    - This tutorial uses Eclipse, which has explicit JSF 2.2 support.
    - NetBeans and IntelliJ IDEA also have good JSF 2 support.

Software Needed: Summary (Details in Later Sections)

- **To run on Tomcat**
  - Install Java
    - Java 7 or 8
  - Install an IDE
    - I use Eclipse
  - Download Tomcat 7 or 8
    - Or any server supporting servlets 3.0 or later
  - Get JSF 2.2 JAR file
    - Download from Oracle (Mojarra) or Apache (MyFaces)
  - web.xml, faces-config.xml
    - Required entries shown later in tutorial
- **To run on Java EE 7**
  - Install Java
    - Java 7
  - Install an IDE
    - I use Eclipse
  - Download Glassfish 4
    - Or any server supporting Java EE 7
  - No extra JAR files needed
    - Java EE 7 has built-in support for JSF 2.2
  - web.xml, faces-config.xml
    - Required entries shown later in tutorial

Fast Start for Experts

- **If you already use Tomcat and Eclipse**
  - Grab jsf-blank.zip from online link
    - http://www.coreservlets.com/JSF-Tutorial/jsf2/
  - Import into Eclipse. Deploy to Tomcat
    - Eclipse Luna or later recommended
  - Run http://localhost/jsf-blank/
  - Build JSF 2.2 apps using the Eclipse wizard. Or, use jsf-blank as starting point. App has:
    - The JSF JAR file in WEB-INF/lib
    - The needed entries in WEB-INF/web.xml
    - A JSF 2.2 compliant WEB-INF/faces-config.xml file
  - Skip the rest of this tutorial section
    - And move on to sections on specific JSF 2 features.

Installing Java and Tomcat

For even more detailed step-by-step instructions, see tutorials on using Eclipse with Tomcat at http://www.coreservlets.com/Apache-Tomcat-Tutorial/

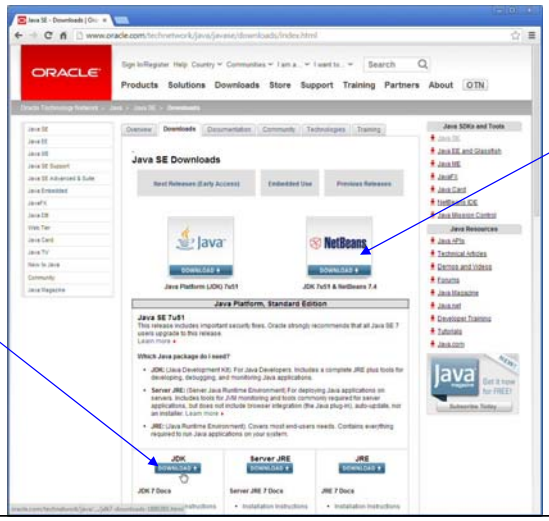
Installing Java SE

- **Minimum Java version**
  - Java 6 legal, but Java 7 or especially Java 8 much better
    - Java 8 *highly* recommended for Enterprise apps. Many new features for simpler code, dealing with large data sets, easier file IO, and automatic parallelization.
      - For details, see http://www.coreservlets.com/java-8-tutorial/
- **Downloading and installation**
  - Follow directions at Oracle site
    http://www.oracle.com/technetwork/java/javase/downloads/
  - Choose "JDK", not "JRE" or "Server JRE"
  - Install and accept all defaults
- **Bookmark the Java API ("JavaDocs")**
  - http://docs.oracle.com/javase/8/docs/api/ or
  - http://docs.oracle.com/javase/7/docs/api/

Installing Java SE

- **Install Java SE 7 or 8**
  - http://www.oracle.com/technetwork/java/javase/downloads/
    - Java 8 final available 3/2014, and is *huge* improvement. See Java 8 tutorial at http://www.coreservlets.com/java-8-tutorial/

Use this version. The "JDK – Java Development Kit" includes compiler for .java files, whereas the "JRE – Java Runtime Environment" is only for executing prebuilt .class files.



This tutorial uses Eclipse, but if you prefer the NetBeans IDE, it is very easy to adapt the instructions to that development environment. So, if you prefer NetBeans or your organization has standardized on it, use this download instead of (not in addition to) the one on the left.

Download and Unzip Tomcat

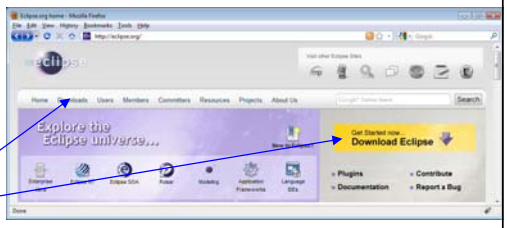
- **Start at http://tomcat.apache.org**
  - Choose download link on left, then ZIP version
    - Tomcat 7 or 8 (recommended)
    - Tomcat 6 (if you need compatibility with older servers, but will not support JSF 2.2 file upload component)
- **Or, go to http://www.coreservlets.com/**
  - Choose Tomcat tutorial from top left
  - This is preconfigured version
    - Set for development, not deployment mode
      - Port changed to 80, servlet reloading enabled, directory listings turned on, etc.
    - Otherwise unchanged
- **Either way, just unzip the file**
  - E.g., resulting in C:\apache-tomcat-7.0.34

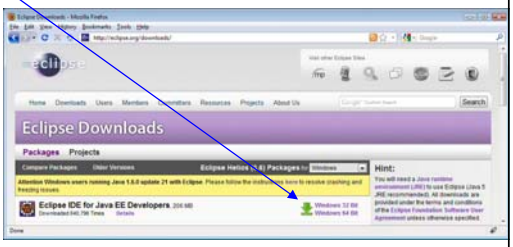
Installing Eclipse

For even more detailed step-by-step instructions, see tutorials on using Eclipse with Tomcat at http://www.coreservlets.com/Apache-Tomcat-Tutorial/

Installing Eclipse

- **Overview**
  - Eclipse is a free open source IDE. Support for Java, Android, HTML, CSS, JavaScript, C++, PHP, JSF, servlets, and more.
    - http://eclipse.org/downloads/
    - Choose "Eclipse IDE for Java EE Developers"



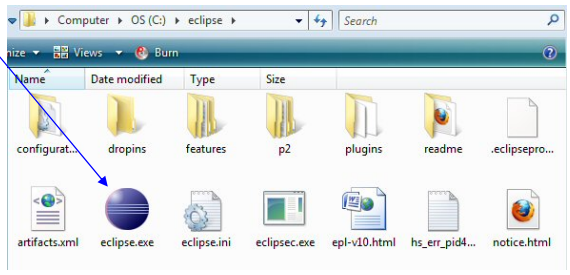


- **Features**
  - Checks your syntax as you type
  - Automatically compiles every time you save file
  - Many tools: refactoring, debugging, server integration, templates for common tasks, etc.
    - Low learning curve: beginners can use Eclipse without knowing these tools

Reminder: step-by-step guide at http://www.coreservlets.com/ (click "Apache Tomcat 7" in top left).

Running Eclipse

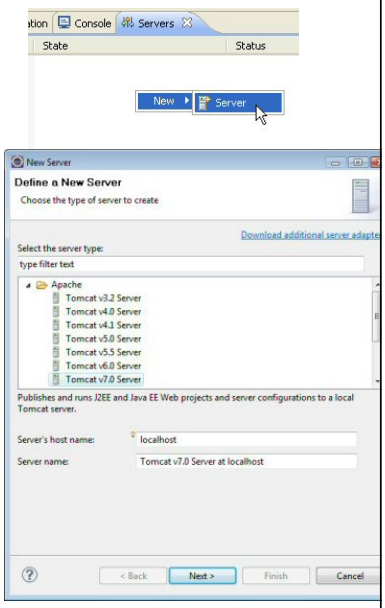
- **Unzip the downloaded file (no installer!)**
  - Call the folder you unzip into "installDir"
- **Double click eclipse.exe**
  - From *installDir*/bin
- **Click on "Workbench" icon**
  - Next time you bring up Eclipse, it will come up in workbench automatically



- **Shortcut**
  - Many developers put Eclipse link on their desktop
    - R-click eclipse.exe, Copy, then go to desktop, R-click, and Paste Shortcut (not just Paste!)

Configuring Eclipse

- **Tell Eclipse about Java version**
  - Window → Preferences → Java → Installed JREs → Press "Add", choose "Standard VM", navigate to JDK folder (not "bin" subdirectory)
    - E.g., C:\Program Files\Java\jdk1.7.0_51
- **Tell Eclipse about Tomcat**
  - Click on Servers tab at bottom. R-click in window.
  - New, Server, Apache, Tomcat v7.0, Next, navigate to folder, Finish.
- **JSF 2.2 support in Eclipse**
  - Eclipse Luna and later support JSF 2.2
    - Autocompletion in .xhtml and .java
    - Visual editor for faces-config.xml
    - Visual previews when editing .xhtml files



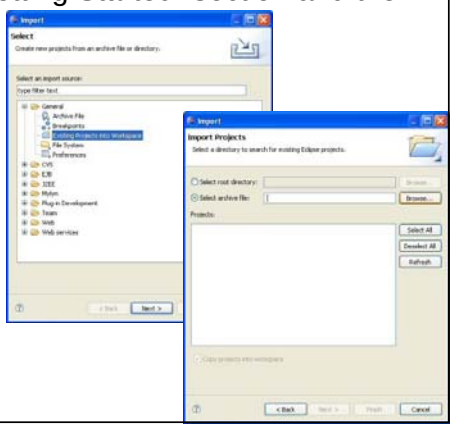
Most recent version (Luna as of 1/2015) is best choice. Old versions have JSF 2 support, but not explicit support for 2.2.

If you lose the "Servers" tab at the bottom of Eclipse, use Window menu, Show View, and hunt for "Servers".

Deploying Apps from Eclipse

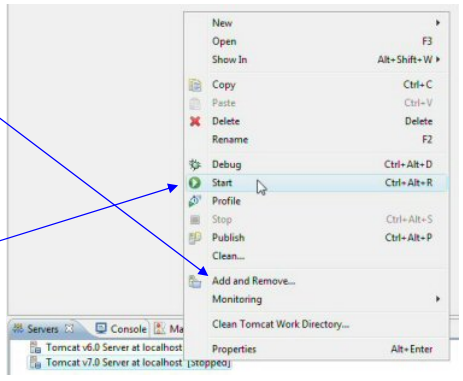
Download and Import Sample Project

- **Get jsf-blank.zip from coreservlets.com**
  - Start at JSF 2 tutorial
    - Go to http://www.coreservlets.com/JSF-Tutorial/jsf2/ or just go to http://www.coreservlets.com/ and click on "JSF 2" in tutorials section in upper left
      - Expand "Installation, Setup, and Getting Started" section and then click on jsf-blank.zip
- **Then, import into Eclipse**
  - After downloading zip file:
    - File, Import, General, Existing Projects, Select archive file. Then click Browse and navigate to jsf-blank.zip.



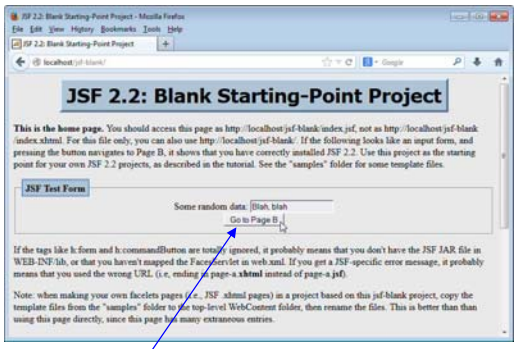
Deploying App in Eclipse

- **Deploy project**
  - Select "Servers" tab at bottom
  - R-click on Tomcat
  - Choose "Add and Remove"
  - Choose project
  - Press "Add"
  - Click "Finish"
- **Start Server**
  - R-click Tomcat at bottom
  - Start (use "Restart" if Tomcat already running)
- **Test URL**
  - http://localhost/jsf-blank/ in any Web browser



Testing Deployed App in Eclipse

- **Start a browser**
  - Eclipse also has builtin browser, but I prefer to use Firefox, Chrome, or IE separately
- **Test base URL**
  - http://localhost/jsf-blank/
    - If you see a form with fields and a button, then app was deployed properly.



- **Test buttons**
  - The fact that you see textfields and a button means that JSF is working, since the fields and button were created with JSF-specific tags. But, you can click on the button to verify that doing so takes you to another page.

Installing JSF 2.2

Main JSF 2.2 Implementations

- **Oracle Mojarra**
  - Main page: https://javaserverfaces.java.net/
  - Runs in any server supporting servlets 3.0 or later
    - Or servlets 2.5 if you do not use file upload component
  - Also integrated into Glassfish 4
- **Apache MyFaces**
  - Main page: http://myfaces.apache.org/core22/
  - Runs in any server supporting servlets 3.0 or later
    - Or servlets 2.5 if you do not use file upload component
- **Any Java EE 7 server**
  - JSF 2.2 is an official builtin part of Java EE 7
    - JBoss 8, Glassfish 4, WildFly 8, JEUS 8 available now (4/2015)
    - WebLogic, WebSphere, etc. coming soon.
- **JSF 2.0 runs in any Java EE 6 server**
  - JBoss 6, Glassfish 3, WebLogic 11, WebSphere 8, Geronimo 3, etc.

Making a JSF 2 Eclipse Project: Alternatives

- **Use Eclipse wizard to build project. Needed:**
  - JAR file
    - Put JSF 2.2 JAR into WebContent/WEB-INF/lib
  - web.xml entries
    - Two required and one recommended settings TODO
  - WEB-INF/faces-config.xml
    - Not required, but a blank one is good idea for later use
  - JSF 2.2 project facet
    - Not needed for execution, but useful for development. Tells Eclipse what editors to use for .xhtml files, faces-config, etc.
- **Copy/rename jsf-blank project**
  - Eclipse project with all required pieces already included
    - Also has the Eclipse JSF 2.2 facet already added
  - Use as starting point for your JSF 2 projects
- **Details on both approaches**
  - Given later in this tutorial

Downloading JSF 2.2 JAR File

- **Oracle Mojarra**
  - https://javaserverfaces.java.net/nonav/2.2/download.html
    - Download a single JAR file (javax.faces-2.2.*x*.jar). Put it in WEB-INF/lib or register it with the Eclipse wizard that will add it for you later.
    - Note that this JAR file is already inside the jsf-blank sample project, so you can also copy it from there
- **Apache MyFaces**
  - http://myfaces.apache.org/download.html
    - Download ZIP file (myfaces-core-assembly-2.2.*x*-bin.zip), then grab JAR file (myfaces-bundle-2.2.*x*.jar) from the lib folder inside it.
- **Java EE 7 servers**
  - No extra JAR files needed. JSF 2.2 built in.

JSF Documentation

- **JSF 2.2 Docs home page**
  - https://javaserverfaces.java.net/nonav/docs/2.2/
- **JSF 2.2 Tags API**
  - https://javaserverfaces.java.net/nonav/docs/2.2/vdldocs/facelets/
    - This is the reference you probably use the most
- **JSF 2.2 Java API**
  - https://javaserverfaces.java.net/nonav/docs/2.2/javadocs/
- **Java 7 API**
  - http://docs.oracle.com/javase/7/docs/api/
- **Java 8 API**
  - http://docs.oracle.com/javase/8/docs/api/

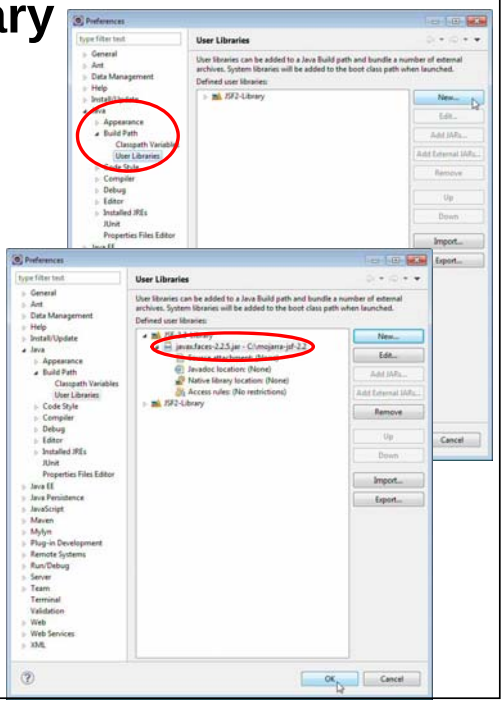
Making JSF 2.2 Project with Eclipse Wizard

Options for Making JSF Project

- **Use Eclipse wizard**
  - Advantage
    - Moderately easy (after initial setup)
  - Disadvantages
    - Makes non-standard web.xml
    - Does not copy existing Java, HTML, CSS code
- **Copy and rename jsf-blank sample project**
  - Advantage
    - Copies Java, HTML, CSS code
      - So better if you have existing project with lots of code
    - Makes standard web.xml
  - Disadvantage
    - Requires tedious process to rename .component file due to Eclipse (not JSF!) bug in renaming projects

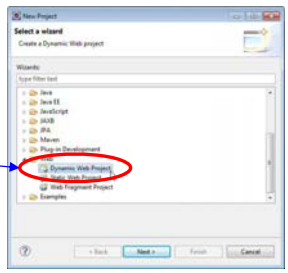
One Time Only: Register JSF 2.2 JAR

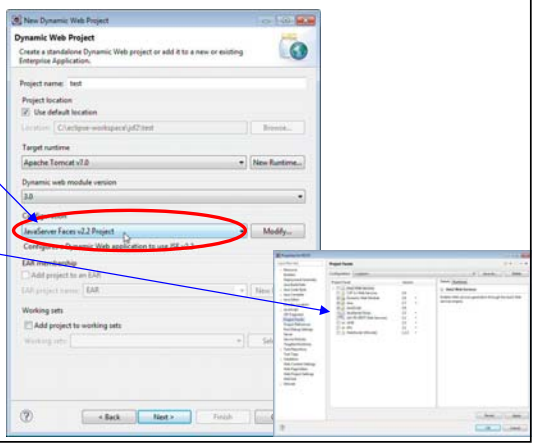
- **Create JSF 2.2 user library**
  - Window → Preferences → Java → Build Path → User Libraries → New
- **Select new library and point at JSF 2.2 JAR file**
  - Click on your new library
  - Click "Add External JARs"
  - Point to JAR file from wherever you saved it from previous section



Each Time: Make Dynamic Web Project with JSF 2.2

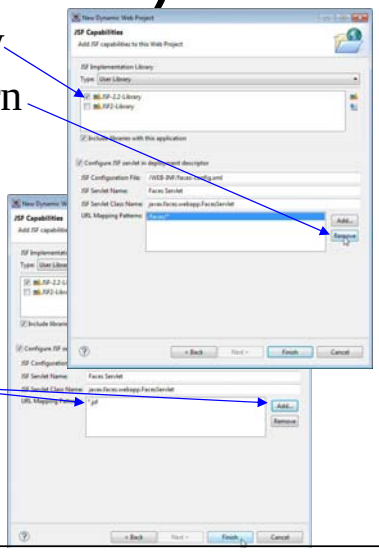
- **Create project**
  - File → New → Project → Web → Dynamic Web Project
    - Next time, you can do File → New → Dynamic Web Project
- **Specify JSF 2.2 for configuration**
  - You can also create vanilla Dynamic Web Project, then add JSF 2.2 later by R-clicking project, selecting Properties, and going to Project Facets





Make Dynamic Web Project with JSF 2.2 (Continued)

- **Use defaults for src location and web.xml**
  - (1st and 2nd screens after last slide)
- **For "JSF Capabilities" (3rd screen)**
  - Add your new JSF 2.2 user library
  - Remove *.faces as the URL pattern
  - Add *.jsf as the URL pattern



Make Dynamic Web Project with JSF 2.2 (Continued)

- **Clean up auto-generated web.xml**
  - The one made by Eclipse has some spurious entries (localizationContext and ConfigureListener), lacks the very valuable PROJECT_STAGE setting, and fails to set any welcome-page URLs
- **Solution: copy web.xml from jsf-blank**
  - In principle, you could also edit by hand, but this is tedious and error prone.
    - Format of web.xml entries is discussed in "Programming Basics" tutorial section
    - You can download the more-standard web.xml from "Getting Started" section of JSF 2.2 tutorial at coreservlets.com, but easier to download all of jsf-blank, then copy from there

web.xml: Required Entries

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app ... version="3.0">
 <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

Behind-the-scenes servlet that runs JSF

This means that all URLs ending in .jsf will be handled by JSF. There are other options besides *.jsf, but *.jsf is the most common.

Reminder: do not type this in by hand. Copy from jsf-blank instead.

web.xml: Optional-but-Recommended Entries

```
<?xml version="1.0" encoding="UTF-8"?>
  ...
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <context-param>
    <param-name>javax.faces.STATE_SAVING_METHOD</param-name>
    <param-value>client</param-value>
  </context-param>
  <welcome-file-list>
    <welcome-file>index.jsf</welcome-file>
    <welcome-file>welcome.jsf</welcome-file>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

Means that you are in the development (not deployment) phase, so JSF should give extra error messages.

How the Java objects that represent the form elements are stored. Best choice is debatable, but client is slightly preferred. JSF 2.2 also has stateless views, where this entry is irrelevant.

If you go to http://host/project/ (with no file name), it will try index.jsf first, welcome.jsf next, and so forth.

Reminder: do not type this in by hand. Copy from jsf-blank instead.

Making JSF 2.2 Project by Copying jsf-blank

Options for Making JSF Project

- **Use Eclipse wizard**
  - Advantage
    - Moderately easy (after initial setup)
  - Disadvantages
    - Makes non-standard web.xml
    - Does not copy existing Java, HTML, CSS code
- **Copy and rename jsf-blank sample project**
  - Advantage
    - Copies Java, HTML, CSS code
      - So better if you have existing project with lots of code
    - Makes standard web.xml
  - Disadvantage
    - Requires tedious process to rename .component file due to Eclipse (not JSF!) bug in renaming Web projects

Big Idea

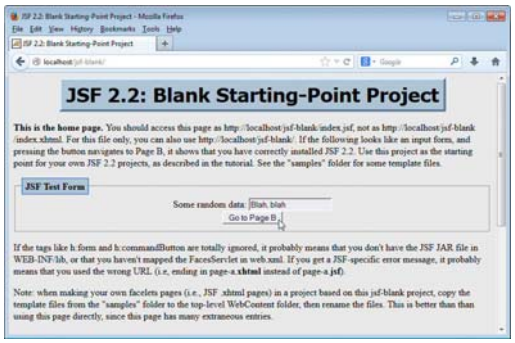
- **Start with a pre-made Eclipse project**
  - JAR file
    - javax.faces-2.2.*x*.jar (not needed if using Java EE 7)
  - web.xml
    - All entries from previous section
  - faces-config.xml
    - Legal start/end tags, empty body
  - "samples" folder with templates for .xhtml files
    - Copy these to WebContent, rename them to fit your app, and edit for your own code
  - Sample CSS style sheet
    - Copy or use as a reference if you are rusty on CSS
  - Super-simple test code
    - You will remove this for your own apps

Importing the jsf-blank Project

- **Grab jsf-blank.zip from tutorial site**
  - http://www.coreservlets.com/JSF-Tutorial/jsf2/
- **Import into Eclipse**
  - Start Eclipse and go to Workbench
  - Use File → Import → General → Existing Projects into Workspace → Next → Select archive file
  - Then click Browse, navigate to jsf-blank.zip, and continue
  - You should now see jsf-blank in project list at left
  - Can run as is on Tomcat 6, Tomcat 7, or Glassfish 3
    - Test locally (next page), or build WAR file in normal manner to send to deployment server (R-click project, Export → WAR file).

Testing the jsf-blank Project

- **Deploy to server**
  - Click on Servers tab at bottom.
  - R-click on Tomcat v7.0
  - Choose "Add and Remove and select jsf-blank.
  - R-click Tomcat again and choose Start.
- **Test**
  - Start browser and enter URL http://localhost/jsf-blank/
  - Try the pushbutton

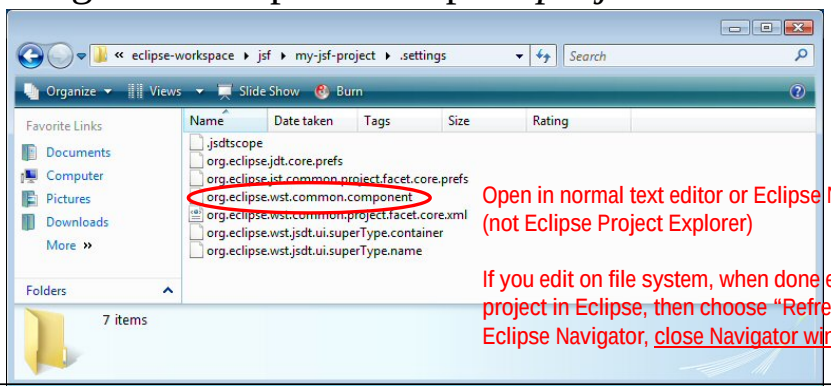


Making Your Own JSF 2 Project by Copying jsf-blank

- **Copy the jsf-blank project**
  - R-click on jsf-blank at left. Copy. R-click again. Paste.
    - Use same process later if you have a JSF project you want to copy, and you want all the code to be copied with it.
- **Problem: Eclipse bug**
  - Eclipse leaves references to old name in new project.
    - One can be found by R-clicking project, then Properties → Web Project Settings. But the other references have to be changed manually, so you might as well replace all of them manually.
- **Solution**
  - Go to file system, edit eclipse-workspace/*projName*/ .settings/org.eclipse.wst.common.component
    - You could also use Eclipse "Navigator" (*not* Proj Explorer)
  - Change *all* instances of old project name to new one
  - R-click on project and choose Refresh
  - Close the Navigator when done. Do not forget this step!

Copying jsf-blank: Example

- **First, copy project**
  - R-click on jsf-blank, choose "Copy"
  - R-click in Project Explorer window, choose "Paste"
    - E.g., name it my-jsf-project
- **Next, edit .component file**
  - Navigate to Eclipse workspace/*projectName*/.settings



This issue is already in the Eclipse known bugs list, so hopefully it will be fixed soon. But as of Jan. 2014, the bug still exists in all Eclipse versions, including Kepler & Luna.

Open in normal text editor or Eclipse Navigator (not Eclipse Project Explorer)

If you edit on file system, when done editing, R-click on project in Eclipse, then choose "Refresh". If you edit with Eclipse Navigator, close Navigator window when finished.

Copying jsf-blank: Example (Continued)

- **.component file: before**

```
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="jsf-blank">
        <wb-resource deploy-path="/" source-path="/WebContent"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="/src"/>
        <property name="context-root" value="jsf-blank"/>
        <property name="java-output-path"
                  value="/jsf-blank/build/classes"/>
    </wb-module>
</project-modules>
```

- **.component file: after**

```
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="my-jsf-project">
        <wb-resource deploy-path="/" source-path="/WebContent"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="/src"/>
        <property name="context-root" value="my-jsf-project"/>
        <property name="java-output-path"
                  value="/my-jsf-project/build/classes"/>
    </wb-module>
</project-modules>
```

Wrap-Up

Summary

- **Setup**
  - Install Java 8 (or 7) and Eclipse
  - Install recent Tomcat version (7 or 8) or Java EE 7 server
  - Test by downloading and deploying jsf-blank.zip
    - jsf-blank includes the JSF 2.2 JAR file
- **Try your own JSF 2 project**
  - Use Eclipse wizard to make JSF 2.2 project
    - But, when done, overwrite web.xml to get standard setup
  - Or, better (?), copy/rename jsf-blank
    - Due to Eclipse bug, you must then edit workspace/*projectName*/.settings/…component and change all occurrences of "jsf-blank" to new name
      - Close Navigator window when done!

Questions?

More info:
http://www.coreservlets.com/JSF-Tutorial/jsf2/ – JSF 2.2 tutorial
http://www.coreservlets.com/JSF-Tutorial/primefaces/ – PrimeFaces tutorial
http://courses.coreservlets.com/jsf-training.html – Customized JSF and PrimeFaces training courses
http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training