JSF 2: Installation, Setup, and Getting Started

Originals of Slides and Source Code for Examples:
http://www.coreservlets.com/JSF-Tutorial/jsf2/

Topics in This Section

- **Getting required software**
  - Installing Java SE 6
  - Installing Eclipse (Java EE version)
  - Installing a server for JSF 2.0
    - Tomcat 6 or 7 (also needs jsf-api.jar and jsf-impl.jar) or
    - Any Java EE 6 server (e.g., Glassfish 3)
- **Testing projects**
  - Importing and testing an existing JSF 2.0 project
    - Deploying on Tomcat and Glassfish
  - Making your own JSF 2.0 project
- **Sneak preview of basic features**
  - Summary of code in jsf-test project

Overview

Overview of JSF 2

- **JSF 2 adds many new features vs. JSF 1.*x***
  - Smart defaults
  - Annotations to replace many faces-config.xml entries
  - Ajax support
  - Integrated support for facelets
  - Simpler custom components
  - More components and validators
  - Support for Groovy
  - Ability to bookmark results pages
  - Lots more
- **But, as of early 2012, JSF 2.0 was hard to test**
  - Simple installation and testing instructions hard to find
    - Rectifying this is the main point of this section
    - Later sections give detailed tutorial on JSF 2.0 features

Summary: Requirements for Running JSF 2.0

- **Java**
  - To run with Tomcat 6, Java 5 or later needed
  - To run with Tomcat 7 or Glassfish 3, Java 6 or later needed
    - This tutorial uses Tomcat and Java 6
- **A server**
  - Servers that just support servlets 2.5 or later (e.g., Tomcat 6 or 7) need two JAR files (jsf-api.jar and jsf-impl.jar)
    - In addition, JSTL 1.2 JAR files needed if you use ui:repeat tag
    - JSF 2.0 also runs on the Google cloud server (which uses Jetty)
  - Servers that support Java EE 6 (e.g., Glassfish 3, JBoss 6, WebLogic 11g) have built-in support for JSF 2.0 & JSTL 1.2
    - All tutorial examples run on Tomcat 6, Tomcat 7, & Glassfish 3
- **An IDE**
  - Optional, but highly recommended.
    - This tutorial uses Eclipse 3.6, which has explicit JSF 2 support.

Software Needed: Summary (Details in Later Sections)

- **To run on Tomcat**
  - Install Java
    - Java 5 or later
  - Install an IDE
    - I use Eclipse 3.6
  - Download Tomcat 6 or 7
    - Or any server supporting servlets 2.5
  - Get JSF 2.0 JAR files
    - jsf-api.jar, jsf-impl.jar
    - (JSTL 1.2 JAR files)
    - Download from Oracle Mojarra or Apache MyFaces
  - web.xml, faces-config.xml
    - Required entries shown later in tutorial
- **To run on Glassfish**
  - Install Java
    - Java 6 or later
  - Install an IDE
    - I use Eclipse 3.6
  - Download Glassfish 3
    - Or any server supporting Java EE 6
  - No extra JAR files needed
    - Java EE 6 has built-in support for JSF 2.0
  - 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 3.6 added JSF 2 support, so 3.6+ is recommended.
  - Run http://localhost/jsf-blank/
  - Use jsf-blank as starting point for your own JSF 2.0 apps. App already has:
    - The two needed JAR files in WEB-INF/lib
      - Also the two optional but recommended JSTL 1.2 JAR files
    - The needed entries in WEB-INF/web.xml
    - A JSF 2.0 compliant WEB-INF/faces-config.xml file
  - Skip the rest of this tutorial
    - 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 6 or Tomcat 7 at http://www.coreservlets.com/Apache-Tomcat-Tutorial/

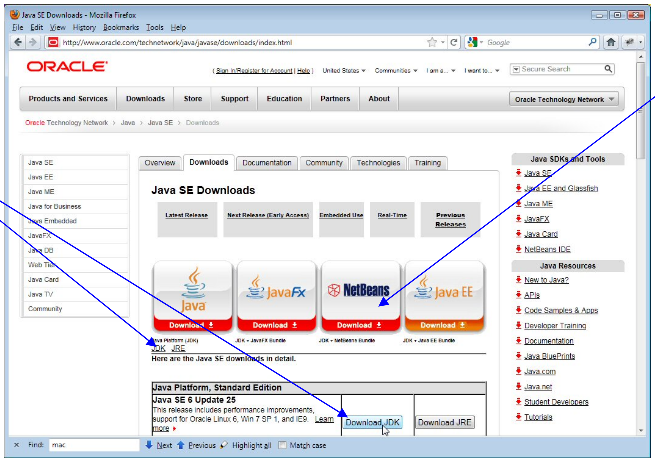
Installing Java SE 6

- **Minimum Java version**
  - Tomcat 7 (servlets 3.0) requires Java 6
  - Tomcat 6 and other servlet 2.5 containers require Java 5+
    - But Java 6 recommended for performance and features
- **Downloading and installation**
  - Follow directions at Oracle site
    http://www.oracle.com/technetwork/java/javase/downloads/
  - Get Java SE; choose "JDK", not "JRE"
    - Not "with Java EE", "with JavaFX", or "with NetBeans"
- **Bookmark the Java API ("JavaDocs")**
  - http://download.oracle.com/javase/6/docs/api/
    - This is the most important Java reference for developers. Eclipse integrates this API, but a separate link is still good

Installing Java SE 6

- **Install Java 6**
  - http://www.oracle.com/technetwork/java/javase/downloads/

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.



Here are the Java SE downloads in detail.

Java Platform, Standard Edition

Java SE 6 Update 25

This release includes performance improvements, support for Oracle Linux 6, Win 7 SP 1, and IE9. Learn more

This tutorial uses Eclipse, but if you prefer the NetBeans environment, 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

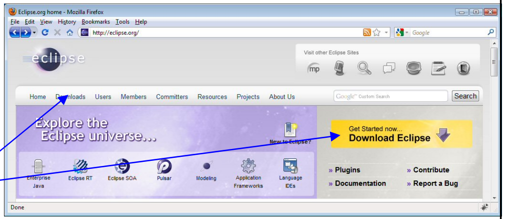
- **Start at http://tomcat.apache.org**
  - Choose download link on left, then ZIP version
    - Tomcat 7 (recommended)
    - Tomcat 6 (if you need compatibility with older servers)
- **Or, go to http://www.coreservlets.com/**
  - Choose Tomcat tutorial from top left
  - This is preconfigured version
    - 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-7.0.8

Installing Eclipse

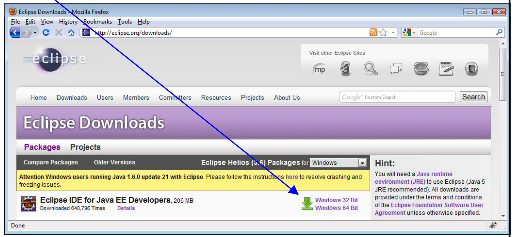
For even more detailed step-by-step instructions, see tutorials on using Eclipse with Tomcat 6 or Tomcat 7 at http://www.coreservlets.com/Apache-Tomcat-Tutorial/

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"
      - Need version 3.6 or later for JSF 2.0 and Tomcat 7 support


Eclipse.org home - Mozilla Firefox
http://eclipse.org/
eclipse
Home Downloads Users Members Committers Resources Projects About Us
Search
Explore the Eclipse universe...
Get Started now... Download Eclipse
Plugins
Documentation
Contribute
Report a Bug

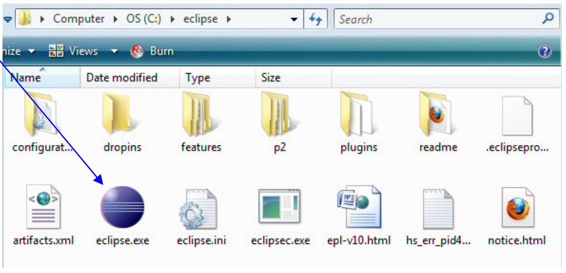
- **Features**
  - Checks your syntax as you type
  - Automatically compiles every time you save file
  - Many tools: refactoring, debugging, server integration, templates for common tasks, etc.
    - Low learning curve: beginners can use Eclipse without knowing these tools


Eclipse Downloads - Mozilla Firefox
http://eclipse.org/downloads/
eclipse
Eclipse Downloads
Packages Projects
Compare Packages Older Versions
Eclipse Helios (3.6) Packages for Windows
Eclipse IDE for Java EE Developers, 206 MB
Windows 32 Bit
Windows 64 Bit
Hint:

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

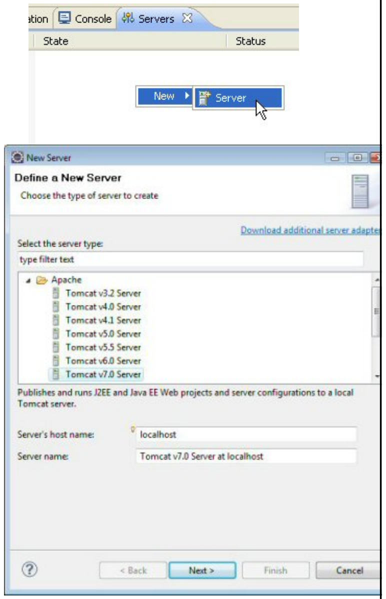
Running Eclipse

- **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\jdk1.6.0_21
- **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 2.0 support**
  - Eclipse 3.6 has support for JSF 2.
    - R-click and add Project Facet for JSF 2
    - R-click .xhtml files and Open With, Web Page Editor
    - Double-click faces-config.xml



Define a New Server

Choose the type of server to create

Download additional server adapte

Select the server type:

type filter text

Apache
- Tomcat v3.2 Server
- Tomcat v4.0 Server
- Tomcat v4.1 Server
- Tomcat v5.0 Server
- Tomcat v5.5 Server
- Tomcat v6.0 Server
- Tomcat v7.0 Server

Publishes and runs J2EE and Java EE Web projects and server configurations to a local Tomcat server.

Server's host name: localhost

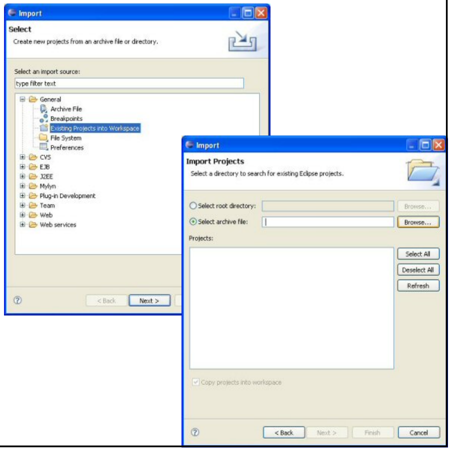
Server name: Tomcat v7.0 Server at localhost

Tomcat v7.0 is choice only in Eclipse 3.6 (Helios). If you prefer Tomcat 6, choose Tomcat v6.0 above instead. If you lose the “Servers” tab at the bottom of Eclipse, use Window, Show View, and hunt for “Servers”.

Deploying Apps from Eclipse

Download and Import Sample Project

- **Get test-app.zip from coreservlets.com**
  – Start at Apache Tomcat tutorial
    - http://www.coreservlets.com/Apache-Tomcat-Tutorial/
      – Choose Tomcat 7 (recommended) or Tomcat 6 version
- **Then, download test-app.zip**
  – Then, import into Eclipse.
    - File, Import, General, Existing Projects, Select archive file. Then click Browse and navigate to test-app.zip.

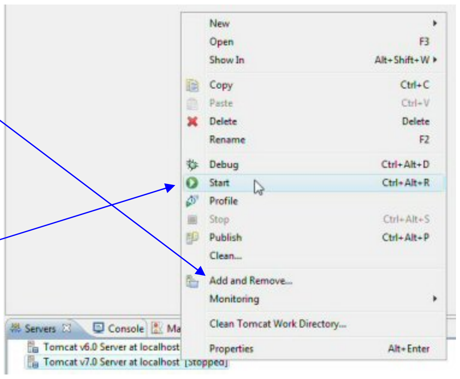


Select
Create new projects from an archive file or directory.
Select an import source:

Import Projects
Select a directory to search for existing Eclipse projects.
Select root directory:
Select archive file:
Projects:
Copy projects into workspace

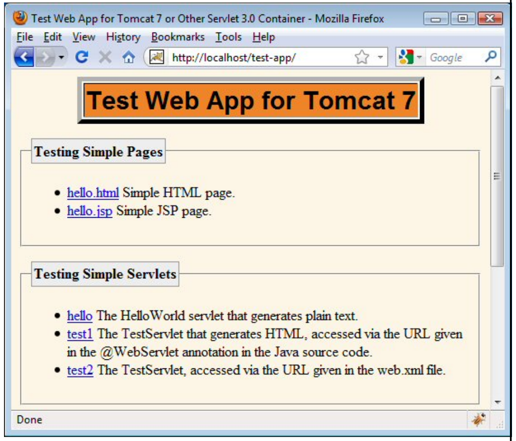
Deploying App in Eclipse

- **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/test-app/ in any Web browser



Testing Deployed App in Eclipse

- **Start a browser**
  - Eclipse also has builtin browser, but I prefer to use Firefox, IE, or Chrome separately
- **Test base URL**
  - http://localhost/test-app/
- **Test Web content**
  - http://localhost/test-app/hello.html
  - http://localhost/test-app/hello.jsp
- **Test servlets**
  - http://localhost/test-app/hello
  - http://localhost/test-app/test1
  - http://localhost/test-app/test2


Test Web App for Tomcat 7 or Other Servlet 3.0 Container - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost/test-app/
Google
Done

Test Web App for Tomcat 7

Testing Simple Pages

- hello.html Simple HTML page.
- hello.jsp Simple JSP page.

Testing Simple Servlets

- hello The HelloWorld servlet that generates plain text.
- test1 The TestServlet that generates HTML, accessed via the URL given in the @WebServlet annotation in the Java source code.
- test2 The TestServlet, accessed via the URL given in the web.xml file.

Installing JSF 2.0

Main JSF 2.0 Implementations

- **Sun/Oracle Mojarra**
  - Main page: http://javaserverfaces.java.net/
  - Runs in any server supporting servlets 2.5 or later
  - Also integrated into Glassfish 3
- **Apache MyFaces**
  - Main page: http://myfaces.apache.org/core20/
  - Runs in any server supporting servlets 2.5 or later
  - Also integrated into Apache Geronimo 3
- **Any Java EE 6 server**
  - JSF 2.0 is an official part of Java EE 6
    - JBoss 6, Glassfish 3, WebLogic 11, WebSphere 8, etc.

Making a JSF 2.0 Eclipse Project: Alternatives

- **Copy/rename jsf-blank**
  - Eclipse project with all required pieces already included
    - Also has the Eclipse 3.6 JSF 2 facet already added
  - Use as starting point for your JSF 2.0 projects.
- **Or, build project from scratch**
  - JAR files
    - Put two required and two recommended JAR files into WebContent/WEB-INF/lib
  - web.xml entries
    - Two required and one recommended settings
  - WEB-INF/faces-config.xml
    - Body can be empty, but legal start/end tags required
- **Details on both approaches**
  - Given later in this tutorial

Downloading JSF 2.0 From Scratch (Mojarra)

- **Required JAR files: jsf-api.jar, jsf-impl.jar**
  - Go in the WEB-INF/lib folder of your projects.
  - Download
    - http://javaserverfaces.java.net/download.html
    - Click on latest 2.*x*.*y* binary bundle
    - Download and grab the two JAR files from lib folder
- **Suggested: jstl-1.2-api.jar, jstl-1.2-impl.jar**
  - Although the Mojarra Web site states that only the jsf-*blah*.jar files are needed, the standard ui:repeat tags use JSTL 1.2 internally. So, the JSTL JARs are highly recommended.
  - Download
    - http://jstl.java.net/download.html
    - Click on both "API" and "Implementation" links

If you download the jsf-blank Eclipse project, you can skip this entire slide, since the jsf-blank project already includes all of the required pieces.

JSF Documentation

- **JSF 2 Java API**
  - http://javaserverfaces.java.net/nonav/docs/2.0/javadocs/
- **JSF 2 Tags API**
  - http://javaserverfaces.java.net/nonav/docs/2.0/pdldocs/facelets/
- **Java 6 API**
  - http://download.oracle.com/javase/6/docs/api/

Using JSF 2.0 with Glassfish 3

Installing Glassfish 3

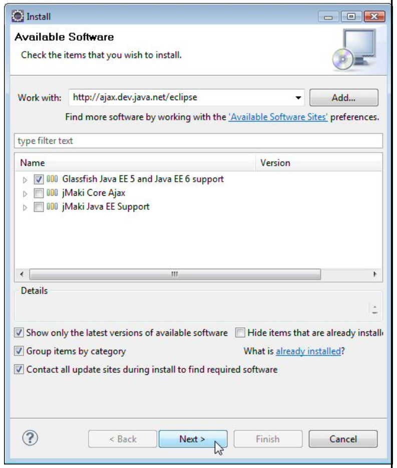
- **Download**
  - Start at https://glassfish.dev.java.net/, follow link to "Downloads".
  - Choose the latest released 3.x version
    - There are both completely open source and commercially-supported versions. The completely open source version is sufficient for everything in JSF 2.0
- **Install**
  - Run installer
    - I installed in C:\glassfishv3
    - I chose anonymous admin login and changed HTTP port from 8080 to 80

Install Java 6 and Eclipse

- **Java 6 required**
  - Java EE 6 will not work with JDK 1.5
- **Eclipse or another IDE strongly recommended**
  - I use Eclipse 3.6 (Java EE Edition) in this tutorial
- **Details**
  - See slides in previous section

Installing Eclipse Glassfish Adapter

- **Eclipse 3.5**
  - Help → Install New Software
  - Enter http://ajax.dev.java.net/eclipse
  - Choose Glassfish Java EE 5, Java EE 6 support
- **Eclipse 3.6 (better!)**
  - Click on Servers tab at bottom. R-click in window.
  - New, Server
  - Click "Download additional server adapters" in top right
  - Choose Oracle, Glassfish 3


Install

Available Software

Check the items that you wish to install.

Work with: http://ajax.dev.java.net/eclipse

Find more software by working with the 'Available Software Sites' preferences.

type filter text

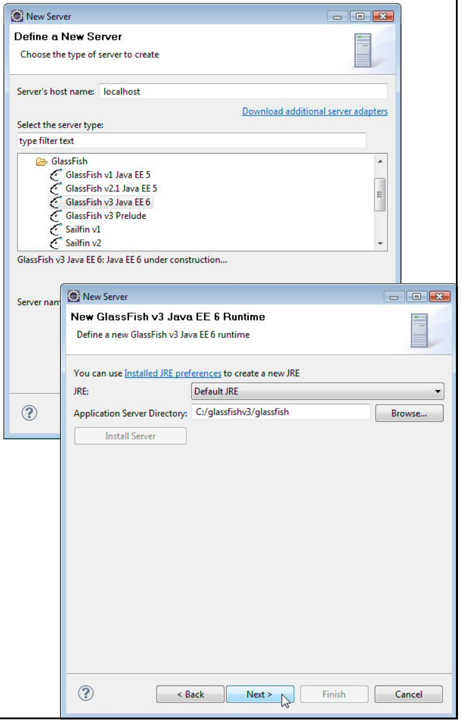
Name	Version
✓ Glassfish Java EE 5 and Java EE 6 support	
jMaki Core Ajax	
jMaki Java EE Support	

Details

✓ Show only the latest versions of available software
Hide items that are already install
✓ Group items by category
What is already installed?
✓ Contact all update sites during install to find required software

Registering Glassfish 3 with Eclipse

- **New server entry**
  - Click on Servers tab at bottom
  - New → Server
  - Choose Glassfish v3 Java EE 6
- **Specify folder**
  - Choose "glassfish" *subfolder* in location where you installed Glassfish. For example, I installed in C:\glassfishv3, so I navigate to C:\glassfishv3\glassfish


New Server

Define a New Server

Choose the type of server to create

Server's host name: localhost

Download additional server adapters

Select the server type:

type filter text

GlassFish
- GlassFish v1 Java EE 5
- GlassFish v2.1 Java EE 5
- GlassFish v3 Java EE 6
- GlassFish v3 Prelude
- Sailfin v1
- Sailfin v2

GlassFish v3 Java EE 6: Java EE 6 under construction...

Server nam

New Server

New GlassFish v3 Java EE 6 Runtime

Define a new GlassFish v3 Java EE 6 runtime

You can use Installed JRE preferences to create a new JRE

JRE: Default JRE

Application Server Directory: C:/glassfishv3/glassfish

Using JBoss 6, WebLogic 11, WebSphere 8, etc.

- **Similar instructions to above**
  - Download server
  - Integrate with Eclipse (download server adapter if needed)
  - R-click on server, Add and Remove, etc.
- **Main difference**
  - Remove JAR files from WEB-INF/lib before deploying
    - All of the sample projects at coreservlets.com have the JSF 2.0 and JSTL 1.2 JAR files in WebContent/WEB-INF/lib. Delete them before deploying to Java EE 6 server.
  - Java EE 6 already supports JSF 2.0 and JSTL 1.2, so it is illegal to supply those JAR files.
    - Glassfish 3 ignores them, but even with Glassfish 3, it is better to delete them.

Using the jsf-blank Project

Big Idea

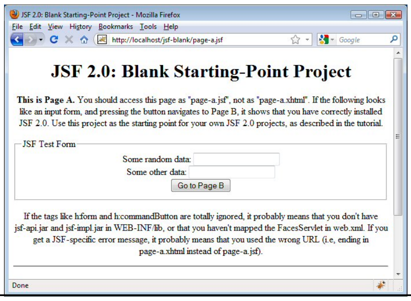
- **Start with a pre-made Eclipse project**
  - JAR files:
    - jsf-api.jar, jsf-impl.jar, jstl-1.2-api.jar, jstl-1.2-impl.jar
    - You can delete these if you use a Java EE 6 server
  - web.xml
    - servlet, servlet-mapping for FacesServlet
    - PROJECT_STAGE set to Development
  - faces-config.xml
    - Legal start/end tags, empty body
  - Super-simple test pages
- **Making your own project**
  - Make your own Dynamic Web project and copy top three pieces above to the new project.
  - Or, copy/rename jsf-blank
    - But due to an Eclipse bug, you must also manually edit an Eclipse file on the file system

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**
  - Tomcat 6 or 7
    - Click on Servers tab at bottom. R-click on Tomcat v6.0, choose “Add and Remove …”. Choose jsf-blank. R-click Tomcat again and choose Start.
  - Glassfish
    - Click on Servers tab at bottom. R-click on Glassfish v3 Java EE 6, choose “Add and Remove …”. Choose jsf-blank. R-click Glassfish again and choose Start.
- **Test**
  - Start browser and enter URL
    - http://localhost/jsf-blank/
      - It should redirect to http://localhost/jsf-blank/page-a.jsf
  - Try the pushbutton


JSF 2.0: Blank Starting-Point Project - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Google

http://localhost/jsf-blank/page-a.jsf

JSF 2.0: Blank Starting-Point Project

This is Page A. You should access this page as "page-a.jsf", not as "page-a.xhtml". If the following looks like an input form, and pressing the button navigates to Page B, it shows that you have correctly installed JSF 2.0. Use this project as the starting point for your own JSF 2.0 projects, as described in the tutorial.

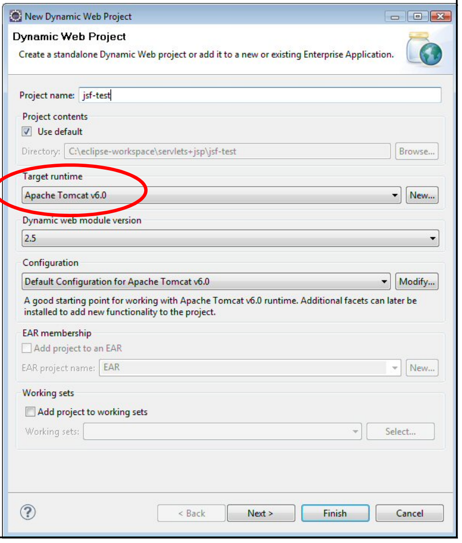
JSF Test Form
Some random data:
Some other data:
Go to Page B

If the tags like h:form and h:commandButton are totally ignored, it probably means that you don't have jsf-api.jar and jsf-impl.jar in WEB-INF/lib, or that you haven't mapped the FacesServlet in web.xml. If you get a JSF-specific error message, it probably means that you used the wrong URL (i.e, ending in page-a.xhtml instead of page-a.jsf).

Done

Making Your Own JSF 2.0 Project (Option 1: Copy Files)

- **Make Dynamic Web Project**
  - File → New → Project → Web → Dynamic Web Project
  - Or (if done before) File → New → Dynamic Web Project
  - Pick a name
    - E.g., jsf-test
  - Specify Apache Tomcat v6.0 or v7.0 as Target Runtime
  - Copy files from jsf-blank to same location in new project
    - Four JAR files in WebContent/WEB-INF/lib
    - WebContent/WEB-INF/web.xml
    - WebContent/WEB-INF/faces-config.xml



Dynamic Web Project
Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

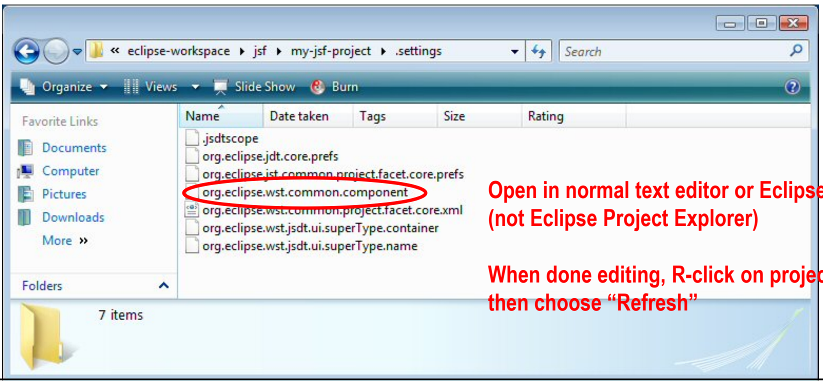
Project name: jsf-test
Project contents
[✓] Use default
Directory: C:\eclipse-workspace\servlets+jsp\jsf-test
Target runtime: Apache Tomcat v6.0
Dynamic web module version: 2.5
Configuration: Default Configuration for Apache Tomcat v6.0
A good starting point for working with Apache Tomcat v6.0 runtime. Additional facets can later be installed to add new functionality to the project.
EAR membership
[ ] Add project to an EAR
EAR project name: EAR
Working sets
[ ] Add project to working sets
Working sets:

Making Your Own JSF 2.0 Project (Option 2: Copy Project)

- **Copy the jsf-blank project**
  - R-click on jsf-blank at left. Copy. R-click again. Paste.
    - Using jsf-blank is better than making your own project because jsf-blank already has JSF 2 Eclipse 3.6 facet
- **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 has to be changed manually, so you might as well replace both 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

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



.jsdtscope
org.eclipse.jdt.core.prefs
org.eclipse.jst.common.project.facet.core.prefs
org.eclipse.wst.common.component
org.eclipse.wst.common.project.facet.core.xml
org.eclipse.wst.jsdt.ui.superType.container
org.eclipse.wst.jsdt.ui.superType.name

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

When done editing, R-click on project in Eclipse, then choose "Refresh"

This issue is already in the Eclipse known bugs list, so hopefully it will be fixed soon. But as of early 2011, the bug still exists in all Eclipse versions, including 3.6 (Helios).

Copying jsf-blank: Example (Continued)

- .component file: before

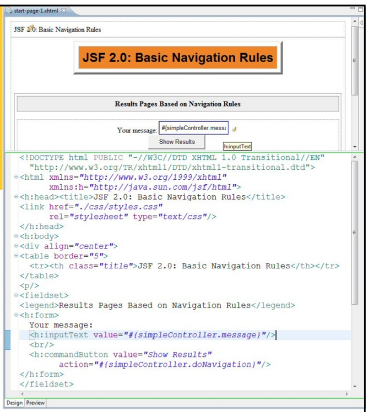
```
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="jsf-blank">
        <wb-resource deploy-path="/" source-path="/WebContent"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="/src"/>
        <property name="context-root"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

- .component file: after

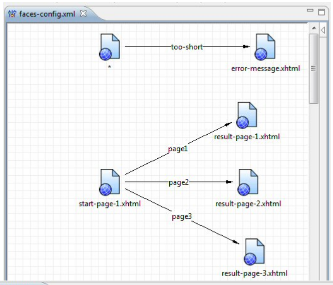
```
<?xml version="1.0" encoding="UTF-8"?>
<project-modules id="moduleCoreId" project-version="1.5.0">
    <wb-module deploy-name="my-jsf-project">
        <wb-resource deploy-path="/" source-path="/WebContent"/>
        <wb-resource deploy-path="/WEB-INF/classes" source-path="/src"/>
        <property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

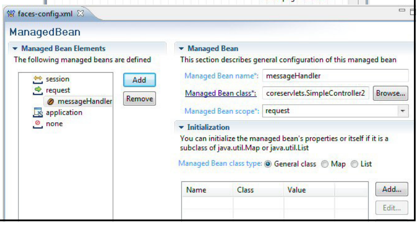
Eclipse Support

- **jsf-blank project already has JSF 2.0 facet**
  – You can add this to any project by R-clicking, going to Properties, Project Facets, and JavaServer Faces 2
- **Eclipse 3.6 and 3.7 have JSF 2 support**
  – To edit .xhtml files: R-click, Open With, Web Page Editor
    - Or, you can make it automatic by going to Window, Preferences, General, Editors, File Associations, *.xhtml, make Web Page Editor the default
  – To edit faces.config.xml: double click it


start-page-1.xhtml
JSF 2.0: Basic Navigation Rules
JSF 2.0: Basic Navigation Rules
Results Pages Based on Navigation Rules
Your message:
#{simpleController.messa
Show Results
h:inputText
Design
Preview

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
<h:head><title>JSF 2.0: Basic Navigation Rules</title>
<link href="./css/styles.css"
      rel="stylesheet" type="text/css"/>
</h:head>
<h:body>
<div align="center">
<table border="5">
  <tr><th class="title">JSF 2.0: Basic Navigation Rules</th></tr>
</table>
<p/>
<fieldset>
<legend>Results Pages Based on Navigation Rules</legend>
<h:form>
  Your message:
  <h:inputText value="#{simpleController.message}"/>
  <br/>
  <h:commandButton value="Show Results"
      action="#{simpleController.doNavigation}"/>
</h:form>
</fieldset>
```





The following managed beans are defined

This section describes general configuration of this managed bean

Managed Bean name*: messageHandler
Managed Bean class*: coreservlets.SimpleController2
Managed Bean scope*: request

You can initialize the managed bean's properties or itself if it is a subclass of java.util.Map or java.util.List

Managed Bean class type: [✓ General class] Map List

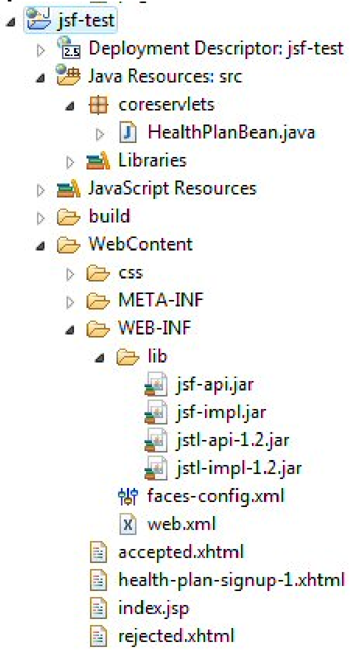
A Small Sample Project

The jsf-test Project

- **A very tiny project**
  - With the bare minimum code needed to do anything in JSF 2.0
- **Web site gives two options**
  - Download entire project to test and examine the code
  - Download individual files, to practice putting them in the right place needed in real projects
  - Individual files and complete project online
    - http://www.coreservlets.com/JSF-Tutorial/jsf2/
- **Deploying and testing**
    - http://localhost/jsf-test/ or (if default port)
    - http://localhost:8080/jsf-test/

Project Layout

- Download files and drag/drop into proper locations
  - src/coreservlets
    - HealthPlanBean.java (R-click on src and make package first)
  - WebContent
    - All .xhtml files, index.jsp
  - WebContent/WEB-INF
    - web.xml and faces-config.xml
  - WebContent/css
    - styles.css (R-click WebContent to make folder)
  - WebContent/WEB-INF/lib
    - The four .jar files

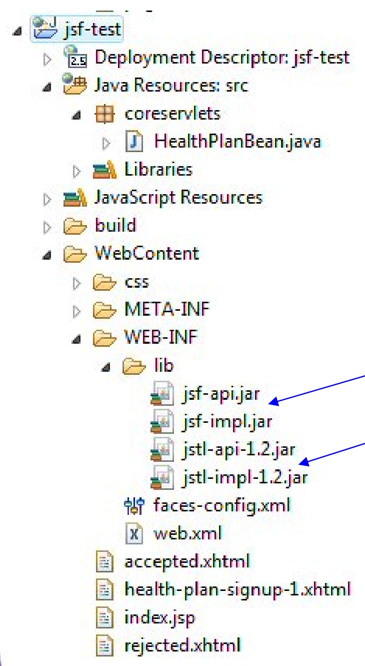


```
jsf-test
  Deployment Descriptor: jsf-test
  Java Resources: src
    coreservlets
      HealthPlanBean.java
    Libraries
  JavaScript Resources
  build
  WebContent
    css
    META-INF
    WEB-INF
      lib
        jsf-api.jar
        jsf-impl.jar
        jstl-api-1.2.jar
        jstl-impl-1.2.jar
      faces-config.xml
      web.xml
    accepted.xhtml
    health-plan-signup-1.xhtml
    index.jsp
    rejected.xhtml
```

Overview of Code

- **Quick summary given here**
  - *Very* simple app
    - Form data ignored
    - Simplest possible action controller
    - Simplest possible results pages
- **Technical details in later sections**
  - More details on each construct
  - More types of apps
- **Reminder**
  - Individual files and complete project can be downloaded from tutorial home page
    - http://www.coreservlets.com/JSF-Tutorial/jsf2/

JAR Files



These two can be downloaded from http://javaserverfaces.java.net/download.html

These two can be downloaded from http://jstl.java.net/download.html

Or, more simply, grab jsf-blank from coreservlets.com and copy from there.

If you run on Tomcat 6 or another server that supports servlets 2.5 but not Java EE 6, the first two JARs are always needed, and the second two JARs are needed if you use any of the ui:repeat tags. If you run on Glassfish 3 or another server that supports Java EE 6, you can delete all four of these JARs.

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app …  version="2.5">
  <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>
  <context-param>
    <param-name>javax.faces.PROJECT_STAGE</param-name>
    <param-value>Development</param-value>
  </context-param>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

Must be version 2.5 or later. Glassfish supports servlets version 3.0. This is an updated requirement from JSF 1.x.

Leave unchanged. This is the same as in JSF 1.x.

Use JSF for URLs that end in blah.jsf. Other popular options are .faces and /faces/*. This is the same as in JSF 1.x.

Give more and more detailed error messages. For example, unknown outcomes are flagged this way (vs. silently redisplaying input form when in deployment mode). Optional. This is new in JSF 2.0.

faces-config.xml

```
<?xml version="1.0"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
   version="2.0">

</faces-config>
```

File is mostly empty in this example. It uses default bean names (derived from the class name), default bean scopes (request), and default results pages (derived from the action controller's return values).

But you are still required to have a faces-config.xml file with legal start and end tags.

health-plan-signup-1.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
…
<h:body>
…
<h:form>
<fieldset>
  <legend>Health Insurance Plan Signup</legend>
  First name: <h:inputText/><br/>
  Last name: <h:inputText/><br/>
  SSN: <h:inputText/><br/>
  Complete medical history since the day you were born:<br/>
  <h:inputTextarea/><br/>
  <h:commandButton value="Sign Me Up!"
                   action="#{healthPlanBean.signup}"/>
</fieldset>
</h:form>
…
</h:body></html>
```

Same header as with facelets in JSF 1.x. But in JSF 2.0, facelets, not JSP, is the standard way of making JSF pages. Note that file is blah.xhtml, but URL is blah.jsf (assuming url-pattern of *.jsf in web.xml).

It is not necessary to use h:body and h:head in this example (regular <body> and <head> are fine). However, when you use h:outputScript and especially f:ajax, you need those tags. So, you might as well plan ahead and use them routinely.

The input elements are ignored in this simplistic example. The next tutorial section will give 'value' attributes corresponding to bean properties.

Same format as in JSF 1.x. But name of bean is automatically derived from Java class name.

HealthPlanBean.java

```
package coreservlets;

import javax.faces.bean.*;

@ManagedBean
public class HealthPlanBean {
  public String signup() {
    if (Math.random() < 0.2) {
      return("accepted");
    } else {
      return("rejected");
    }
  }
}
```

Declares this as managed bean, without requiring entry in faces-config.xml.

Since no name given, name is the class name with the first letter changed to lower case (i.e., healthPlanBean).

Since no scope given, it is request scope.

Since there are no explicit navigation rules in faces-config, these return values correspond to accepted.xhtml and rejected.xhtml (in same folder as page that has the form).

accepted.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
<h:head>
…
</h:head>
<h:body>
…
<table border="5">
  <tr><th class="title">Accepted (Version 1)</th></tr>
</table>
<p/>

<h2>You are accepted into our health plan.</h2>
<p>Congratulations.</p>
…
</h:body></html>
```

I don't actually have any dynamic code in this simplistic example, but it is a good idea to plan ahead and always include these.

rejected.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html">
<h:head>
…
</h:head>
<h:body>
…
<table border="5">
  <tr><th class="title">Rejected (Version 1)</th></tr>
</table>
<p/>

<h2>You are rejected from our health plan.</h2>
<p>Get lost.</p>
…
</h:body></html>
```

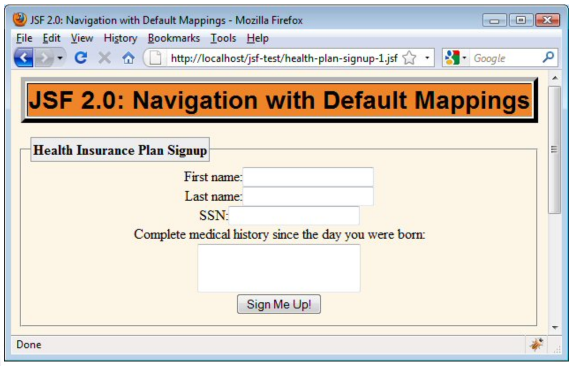
Again, this part plus h:head and h:body are not strictly necessary in this simple example, but it is recommended practice to include them routinely.

index.jsp

```
<% response.sendRedirect("health-plan-signup-1.jsf"); %>
```

So that http://localhost/jsf-test/ redirects to http://localhost/jsf-test/health-plan-signup-1.jsf

Results


JSF 2.0: Navigation with Default Mappings - Mozilla Firefox

http://localhost/jsf-test/health-plan-signup-1.jsf

JSF 2.0: Navigation with Default Mappings

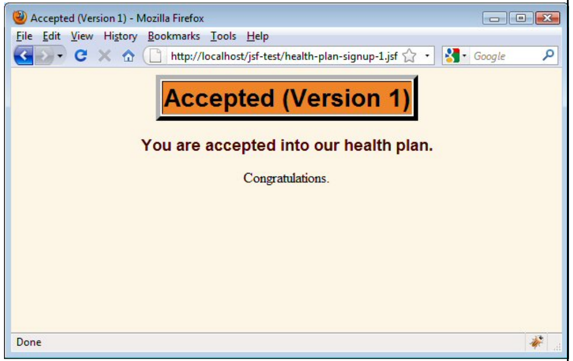
Health Insurance Plan Signup

First name:

Last name:

SSN:

Complete medical history since the day you were born:

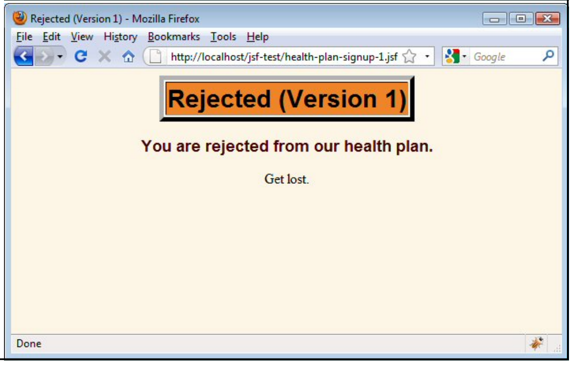

Accepted (Version 1) - Mozilla Firefox

http://localhost/jsf-test/health-plan-signup-1.jsf

Accepted (Version 1)

You are accepted into our health plan.

Congratulations.


Rejected (Version 1) - Mozilla Firefox

http://localhost/jsf-test/health-plan-signup-1.jsf

Rejected (Version 1)

You are rejected from our health plan.

Get lost.

Wrap-Up

Summary

- **Setup**
  - Install Java 6 or 7 and Eclipse 3.6 or 3.7
  - Install recent Tomcat version (6 or 7) or Java EE 6 server
  - Test by downloading and deploying jsf-blank.zip
- **Try your own JSF 2 project**
  - Or, make a new Dynamic Web Project and copy three required pieces from jsf-blank
    - Four JAR files in WEB-INF/lib
    - WEB-INF/web.xml
    - WEB-INF/faces-config.xml
  - Or, 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

Questions?