NetBeans 6.5.1 and GlassFish v 2.1
Creating a Healthcare Facility Visual Web Application

Michael.Czapski@sun.com
June 2009

Introduction

In some views SOA is represented as a series of 4 layers: Presentation Layer (SOA 1), Business Process Layer (SOA 2), Business Service Layer (SOA 3) and Technical Layer (SOA 4). Typically each layer higher up in the hierarchy consumes services exposed by the layer under it. So the Presentation Layer would consume services provided by the Business Process or Business Service Layers. Service interfaces are described using Web Services Description Language (WSDL), sheltering service consumers from details of service implementation. Web Services are seen as the technical means to implement the decoupled functional layers in a SOA development. Decoupling allows implementations of business functionality at different layers to be swapped in and out without disturbing other layers in the stack.

The business idea is that patients are looked after in various healthcare facilities. Frequently applications need to allow selection of a facility and to access facility details for display to human operators. A relational database is used to hold the details of facilities which are a part of the healthcare enterprise. To shelter application developers from the details of the data store facility list and details are made available as a multi-operation web service. This web service will be used to construct the web application that provides a user view into the facilities and facility details.

The previous document in this series, "GlassFish ESB v 2.1 - Creating a Healthcare F acility Web Service Provider", at http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v_2_1, walked the reader through the process of implementing a GlassFish ESB v2.1-based, multi-operation web service which returns facility list and facility details. In this document I will walk through the process of developing a Visual Web Application which will use the Web Service as a data provider. We will use the NetBeans 6.5.1 IDE, which comes as part of the GlassFish ESB v2.1 installation. The application will be implemented as a Visual Web JavaServer Faces Application using JSF component provided by Project Woodstock. This application will introduce the technology in a practical manner and show how a multi-operation web service can be used as a data provider, decoupling the web application from the data stores and specifics of data provision.

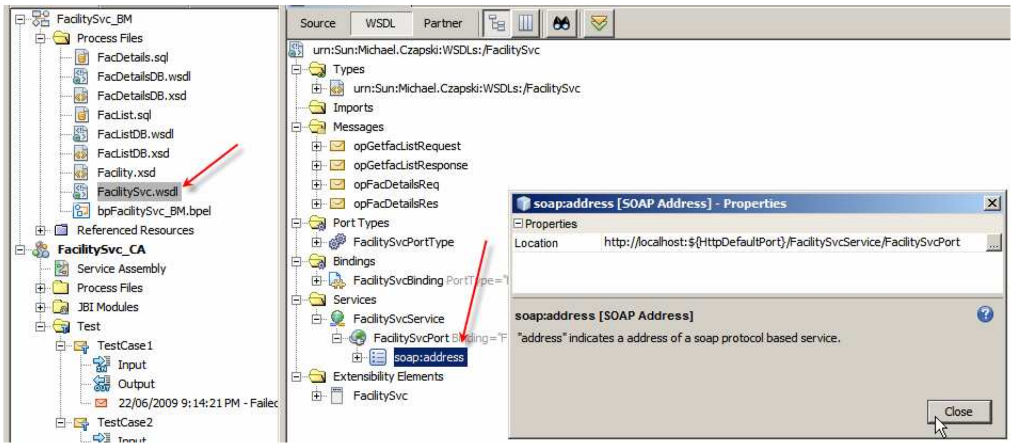
Note that this document is not a tutorial on JavaServer Faces, Visual Web JSF, Project Woodstock components or Web Application development. Not also that all the components and technologies used are either distributed as part of the NetBeans 6.5, as part of the GalssFish ESB v2.1 or are readily pluggable into the NetBeans IDE. All are free and open source.

It is assumed that a GlassFish ESB v2.1-based infrastructure, supplemented by the Sun WebSpace Server 10 Portal functionality and a MySQL RDBMS instance, are available for development and deployment of the web application discussed in this paper. It is further assumed that the web service, developed using instructions in "GlassFish ESB v 2.1 - Creating a Healthcare Facility Web Service Provider", at http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v_2_1, is available and deployed to the infrastructure. The instructions necessary to install this infrastructure are discussed in the blog entry "Adding Sun WebSpace Server 10 Portal Server functionality to the GlassFish ESB v2.1 Installation" at http://blogs.sun.com/javacapsfieldtech/entry/adding_sun_webspace_server_10, supplemented by the material in blog entry "Making Web Space Server And Web Services Play Nicely In A Single Instance Of The Glassfish Application Server", at http://blogs.sun.com/javacapsfieldtech/entry/making_web_space_server_and.

Determining Service Endpoint and WSDL Location

This document assumes that a web application will use the web service developed elsewhere as a data provider. To make it possible we need to know the endpoint location of the service and the location of the WSDL. This information is available if one knows where to look.

Let's open the FacilitySvc.wsdl document in project FacilitySvc_BM and inspect the properties of the soap:address node under the FacilitySvcService node.



soap:address [SOAP Address] - Properties

Properties

Location http://localhost:${HttpDefaultPort}/FacilitySvcService/FacilitySvcPort

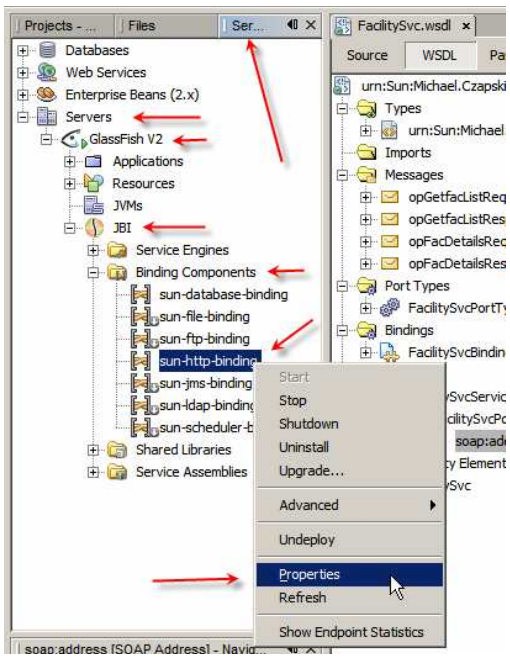
soap:address [SOAP Address]

"address" indicates a address of a soap protocol based service.

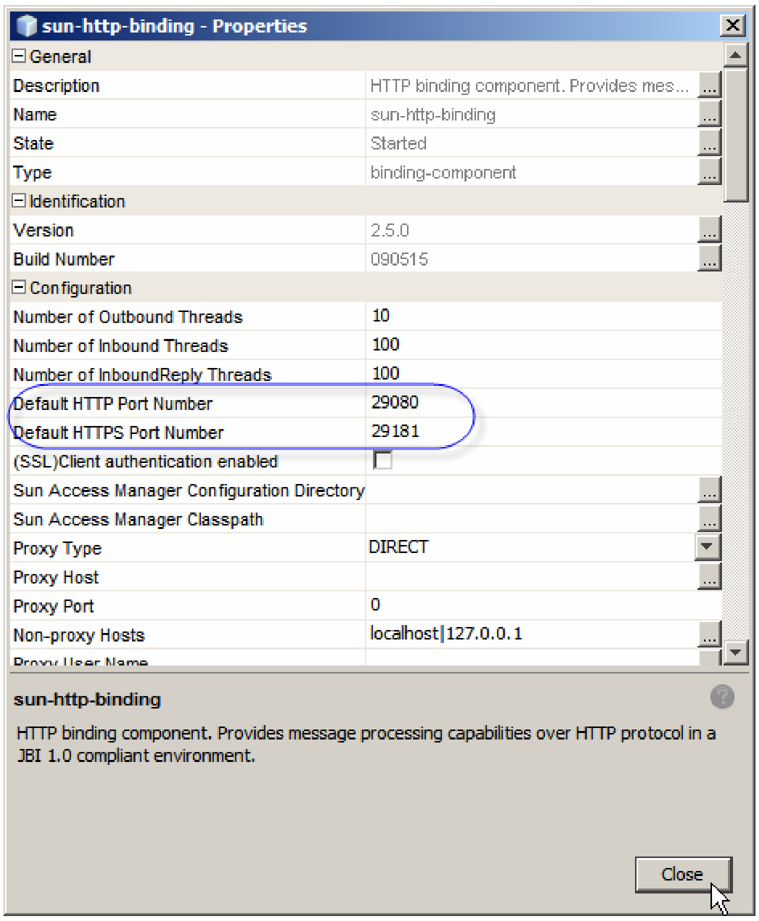
Note the Location property value:

```
http://localhost:${HttpDefaultPort}/FacilitySvcService/FacilitySvcPort
```

The HttpDefaultPort is the port which SOAP/HTTP BCs use. At CA deployment time this variable gets replaced with the actual port. To find out what this port is let's switch to the Services tab in Netbeans, expand Servers, expand JBI, expand Binding Components, right-click sun-http-binding and choose Properties.



Observe the Default HTTP Port Number property value. For my installation this will be 29080. For a default installation it will be 9080. It can be changed.



sun-http-binding - Properties

General	
Description	HTTP binding component. Provides mes...
Name	sun-http-binding
State	Started
Type	binding-component
Identification	
Version	2.5.0
Build Number	090515
Configuration	
Number of Outbound Threads	10
Number of Inbound Threads	100
Number of InboundReply Threads	100
Default HTTP Port Number	29080
Default HTTPS Port Number	29181
(SSL)Client authentication enabled	☐
Sun Access Manager Configuration Directory	
Sun Access Manager Classpath	
Proxy Type	DIRECT
Proxy Host	
Proxy Port	0
Non-proxy Hosts	localhost|127.0.0.1
Proxy User Name	

sun-http-binding

HTTP binding component. Provides message processing capabilities over HTTP protocol in a JBI 1.0 compliant environment.

So, the final service endpoint URL, from the soap:address Location property earlier, will be:

```
http://localhost:29080/FacilitySvcService/FacilitySvcPort
```

This URL is the service location.

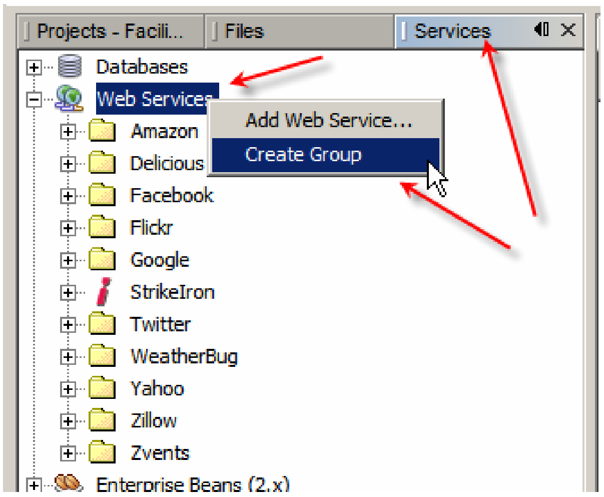
The WSDL for this service can be accessed, using the regular convention, at:

`http://localhost:29080/FacilitySvcService/FacilitySvcPort?WSDL`

With this knowledge we can use the service in related projects.

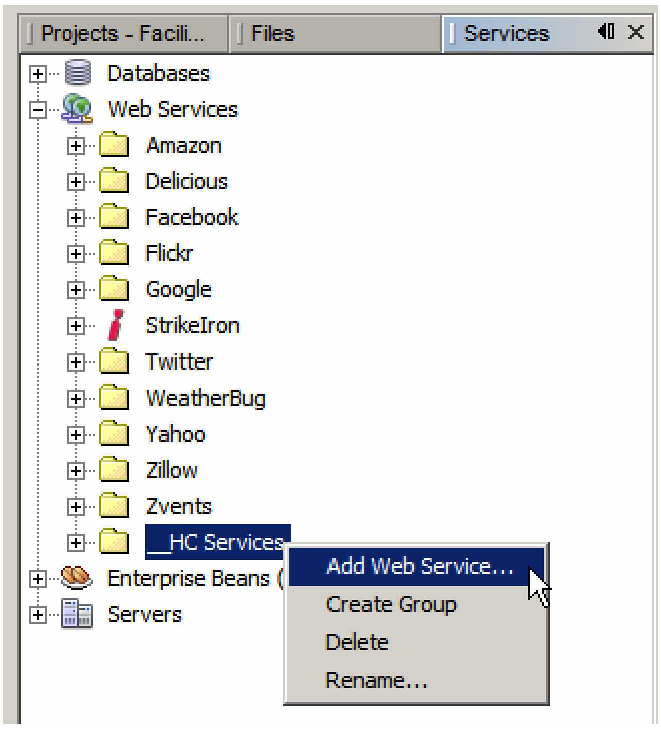
Making Service Available for use in a Web Application

To make a web service available for use as a data provider in a web application we must "introduce" it to the NetBeans IDE. Switch to the Services Tab, right-click on the Web Services node and choose Create Group.

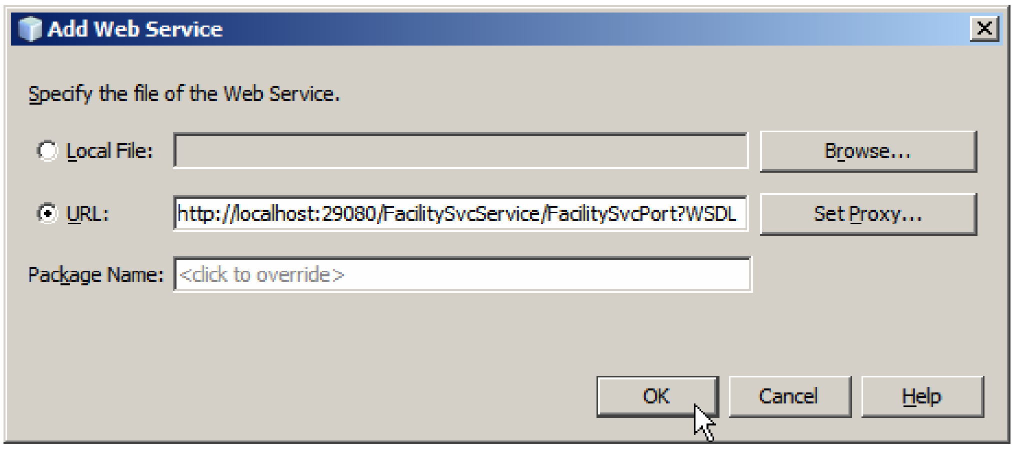


Name this group "__HC Services".

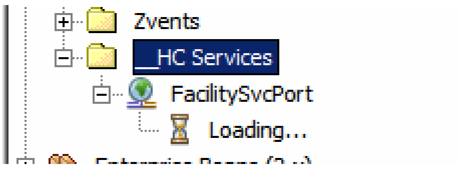
Right-click on the __HC Services node and choose Add Web Service...



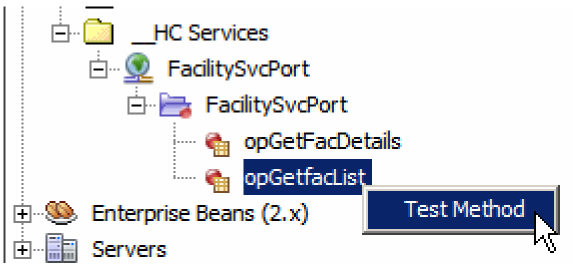
Enter service WSDL URL and click OK. The service WSDL URL for me will be `http://localhost:29080/FacilitySvcService/FacilitySvcPort?WSDL`


Add Web Service
Specify the file of the Web Service.
Local File:
Browse...
URL:
http://localhost:29080/FacilitySvcService/FacilitySvcPort?WSDL
Set Proxy...
Package Name:
<click to override>
OK
Cancel
Help

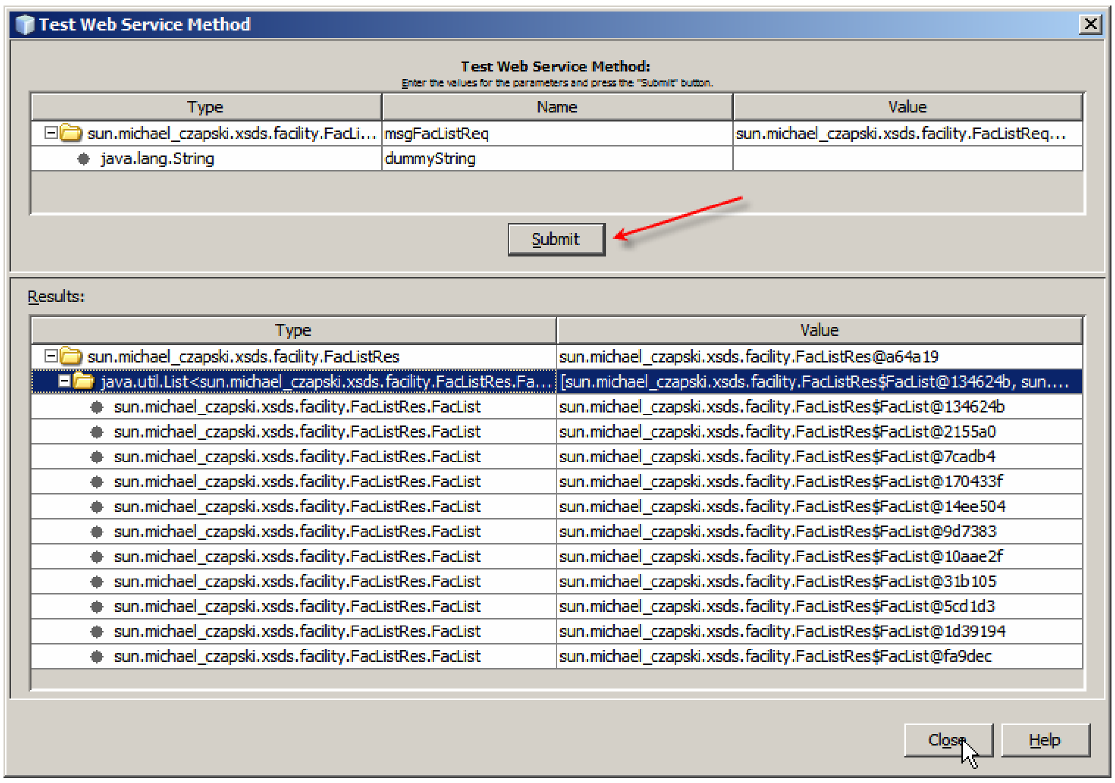
Expand __HC Services -> FacilitySvcPort, wait for the WSDL to be loaded and appropriate classes to be generated, compiled and packaged.



Expand the node tree all the way to the operations, right-click on opFacList and choose Test Method.



Click Submit button and, when execution is completed, inspect the results.



Test Web Service Method:

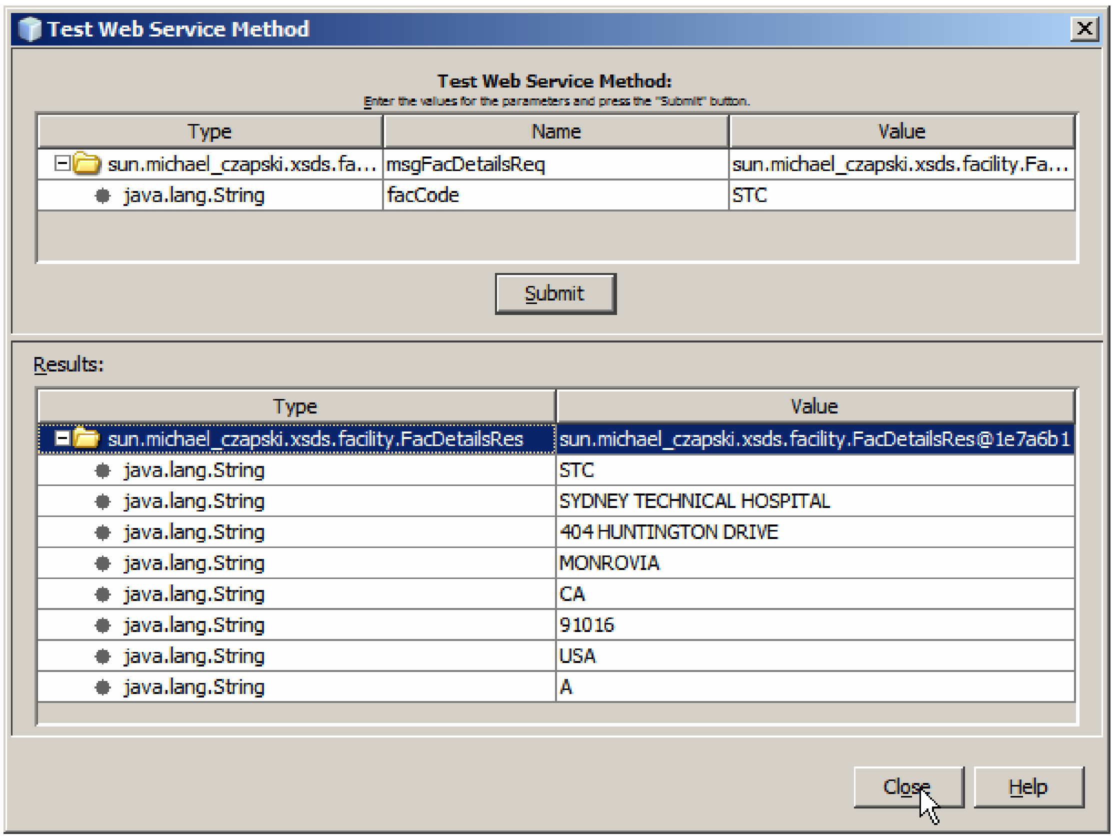
Enter the values for the parameters and press the "Submit" button.

Type	Name	Value
⊟ sun.michael_czapski.xsds.facility.FacLi…	msgFacListReq	sun.michael_czapski.xsds.facility.FacListReq…
java.lang.String	dummyString	

Results:

Type	Value
⊟ sun.michael_czapski.xsds.facility.FacListRes	sun.michael_czapski.xsds.facility.FacListRes@a64a19
⊟ java.util.List<sun.michael_czapski.xsds.facility.FacListRes.Fa…	[sun.michael_czapski.xsds.facility.FacListRes$FacList@134624b, sun.…
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@134624b
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@2155a0
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@7cadb4
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@170433f
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@14ee504
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@9d7383
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@10aae2f
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@31b105
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@5cd1d3
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@1d39194
sun.michael_czapski.xsds.facility.FacListRes.FacList	sun.michael_czapski.xsds.facility.FacListRes$FacList@fa9dec

Now execute the test for the opGetFacDetails, providing facility code of STC.



Test Web Service Method:

Enter the values for the parameters and press the "Submit" button.

Type	Name	Value
⊟ sun.michael_czapski.xsds.fa…	msgFacDetailsReq	sun.michael_czapski.xsds.facility.Fa…
java.lang.String	facCode	STC

Results:

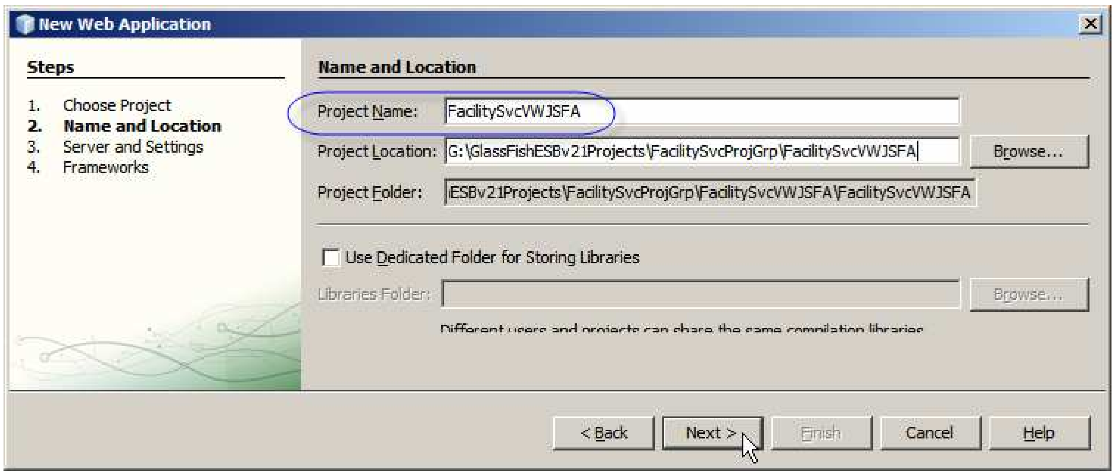
Type	Value
⊟ sun.michael_czapski.xsds.facility.FacDetailsRes	sun.michael_czapski.xsds.facility.FacDetailsRes@1e7a6b1
java.lang.String	STC
java.lang.String	SYDNEY TECHNICAL HOSPITAL
java.lang.String	404 HUNTINGTON DRIVE
java.lang.String	MONROVIA
java.lang.String	CA
java.lang.String	91016
java.lang.String	USA
java.lang.String	A

This is yet another method of testing web services in NetBeans. We have valid references to service operations, ready to be used in web applications and portlets.

Create Visual Web JSF Application

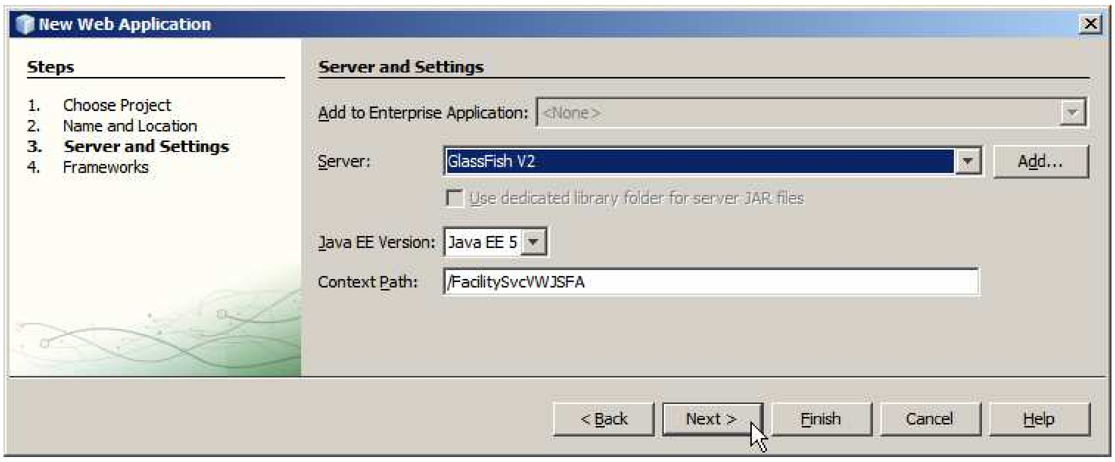
Of the variety of methods available in NetBeans to develop web applications I have chosen the Visual Web JSF method. The reason is that it is visual, easy and quick, if one knows what one is doing. I know enough to be dangerous but not enough to help you out if you get into trouble with JSF or Visual Web JSF. This is a practical cookbook for the specific web applications I built. Feel free to learn the technology and ad-lib.

Create a New Project -> Java Web -> Web Application, FacilitySvcVWJSFA.



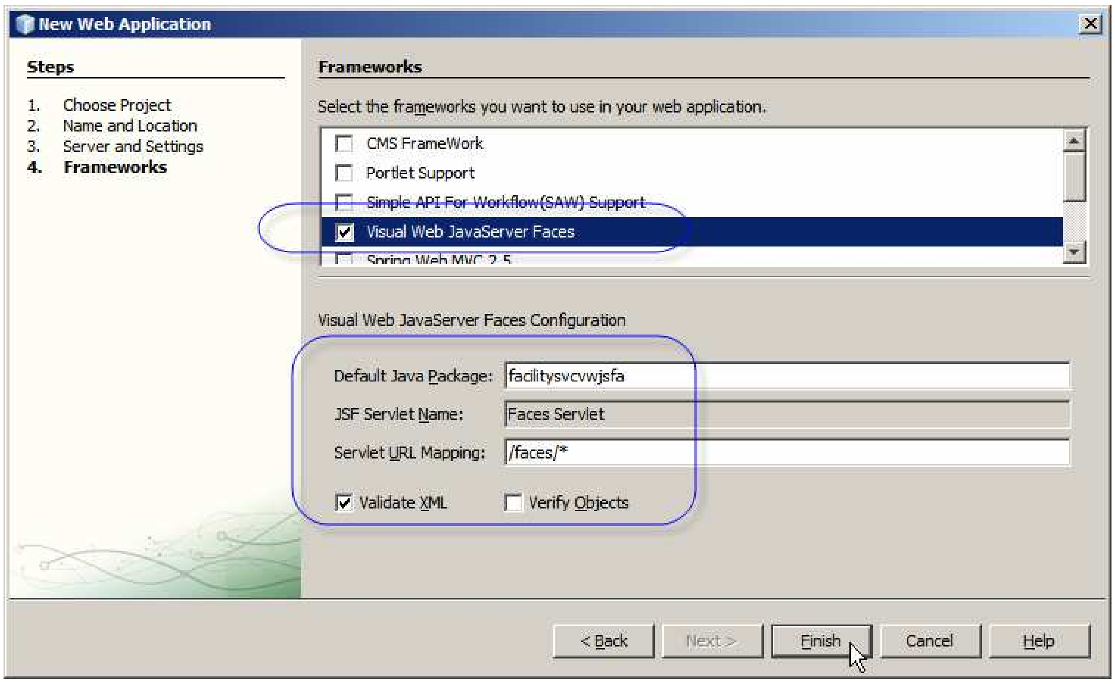
Project Name: FacilitySvcVWJSFA
Project Location: G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvcVWJSFA
Project Folder: ESBv21Projects\FacilitySvcProjGrp\FacilitySvcVWJSFA\FacilitySvcVWJSFA
Use Dedicated Folder for Storing Libraries
Libraries Folder:
Different users and projects can share the same compilation libraries

Accept defaults for Server Settings.



Add to Enterprise Application: <None>
Server: GlassFish V2
Use dedicated library folder for server JAR files
Java EE Version: Java EE 5
Context Path: /FacilitySvcVWJSFA

Check the checkbox next to Visual JavaServer Faces item, accept defaults for other fields and click Finish.



New Web Application

Steps

1. Choose Project
2. Name and Location
3. Server and Settings
4. **Frameworks**

Frameworks

Select the frameworks you want to use in your web application.

- [ ] CMS FrameWork
- [ ] Portlet Support
- [ ] Simple API For Workflow(SAW) Support
- [x] Visual Web JavaServer Faces
- [ ] Spring Web MVC 2.5

Visual Web JavaServer Faces Configuration

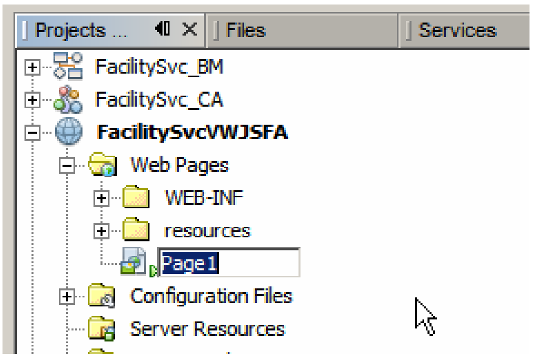
Default Java Package: facilitysvcvwjsfa

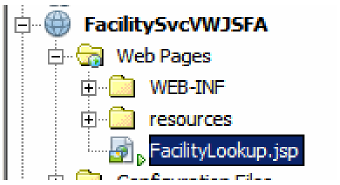
JSF Servlet Name: Faces Servlet

Servlet URL Mapping: /faces/*

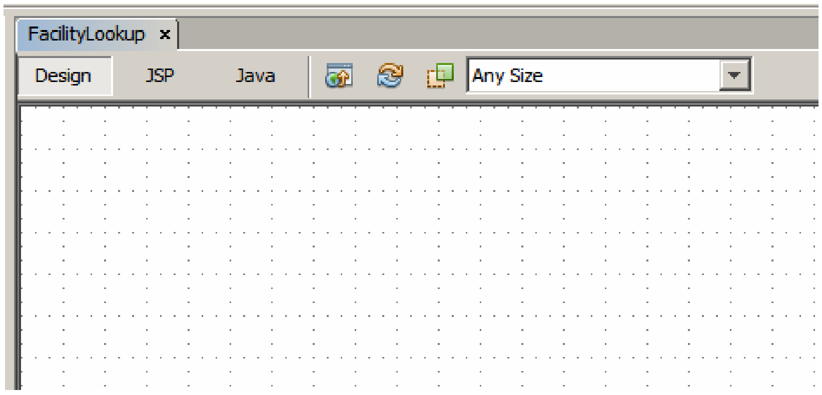
[x] Validate XML   [ ] Verify Objects

Rename FacilitySvcVWJSFA -> Web Pages -> Page1 to FacilityLookup.

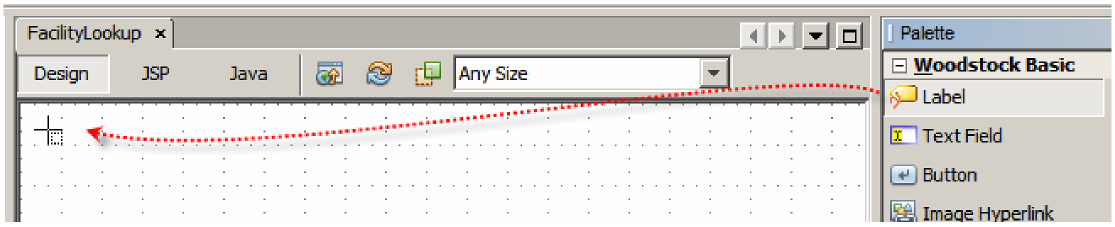


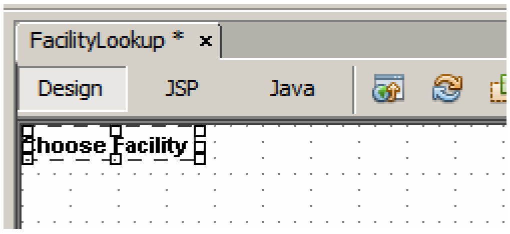


Open FacilityLookup.jsf in Design mode.

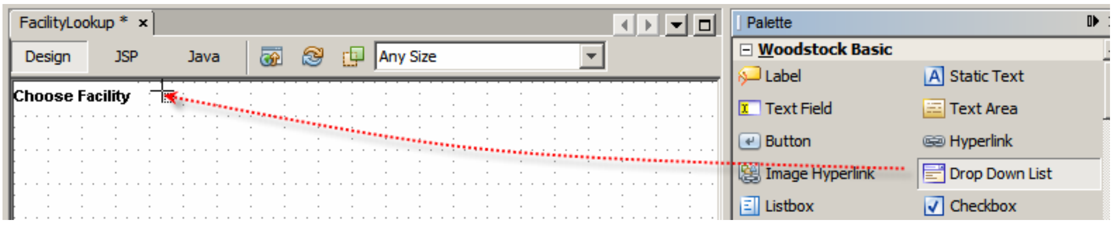


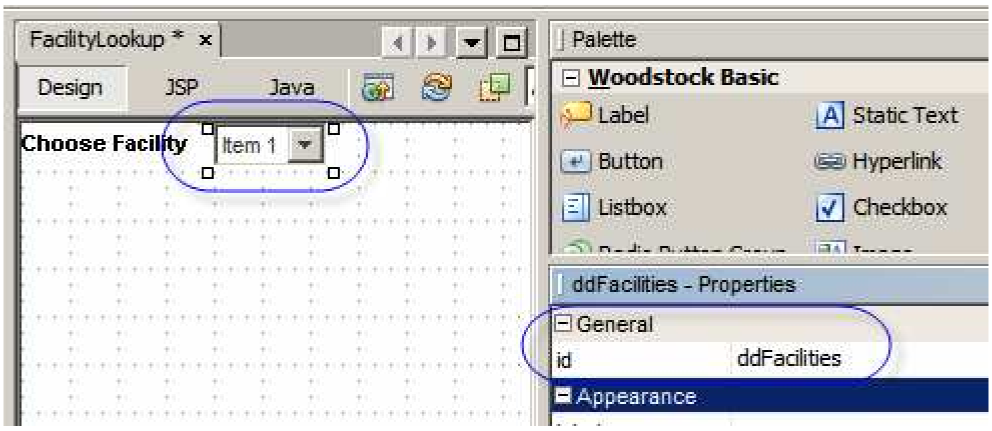
Drag the Woodstock Basic -> Label component onto the canvas and change its text property to read “Choose Facility”.





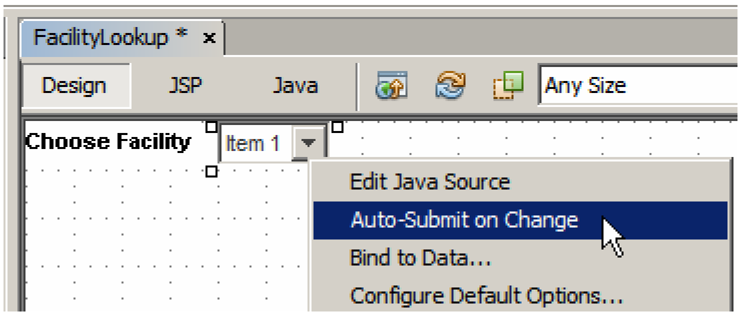
Drag Woodstock Basic -> Drop Down List component to the canvas to the right of the label. Change the General -> Id property of the component to ddFacilities.





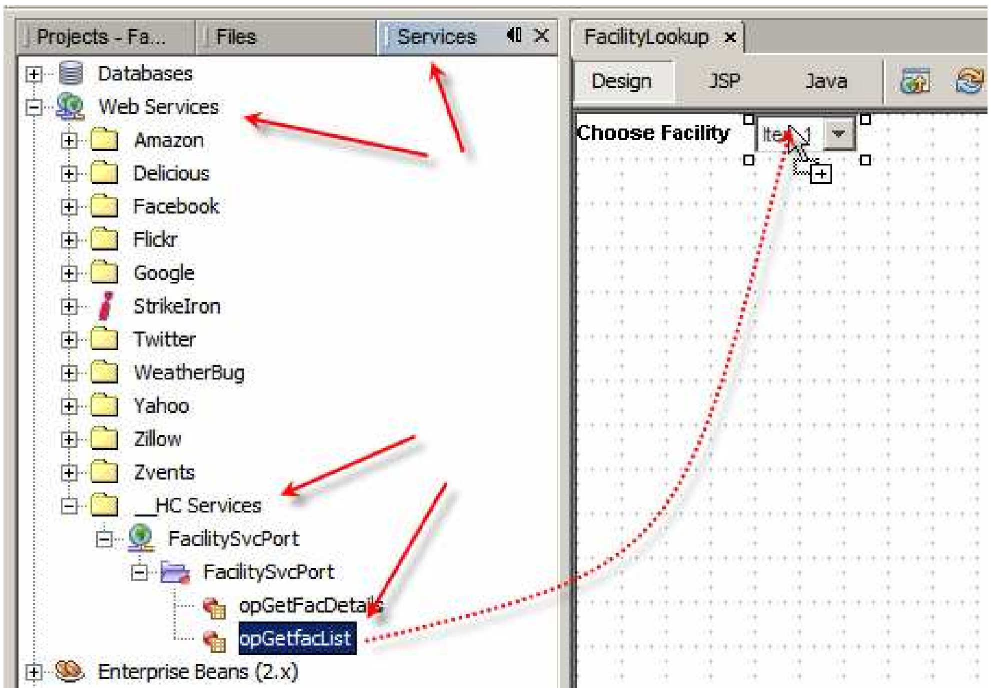
id	ddFacilities

Right-click on the drop down component and choose Auto-Submit On Change.

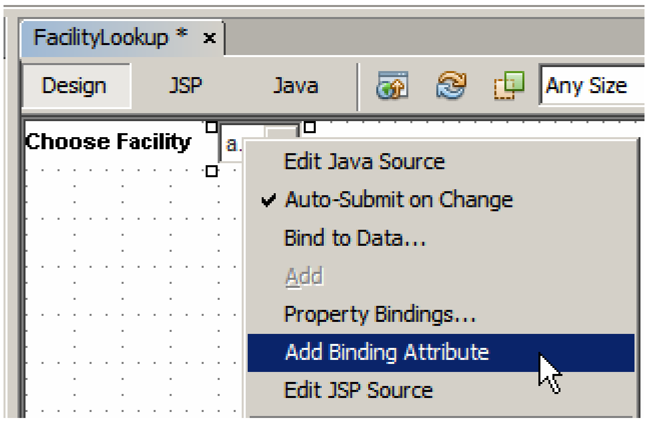


Save the project.

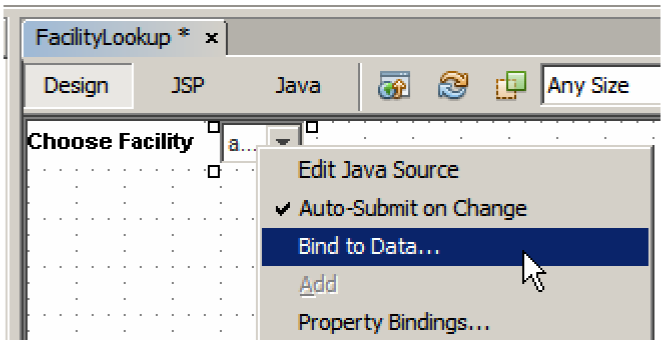
Switch from the Project view to Services view, locate the opGetFacList operation on the web service we added to NetBeans view of web services, drag it onto the FacilityLookup JSF page canvas and drop it right over the top of the drop down component.



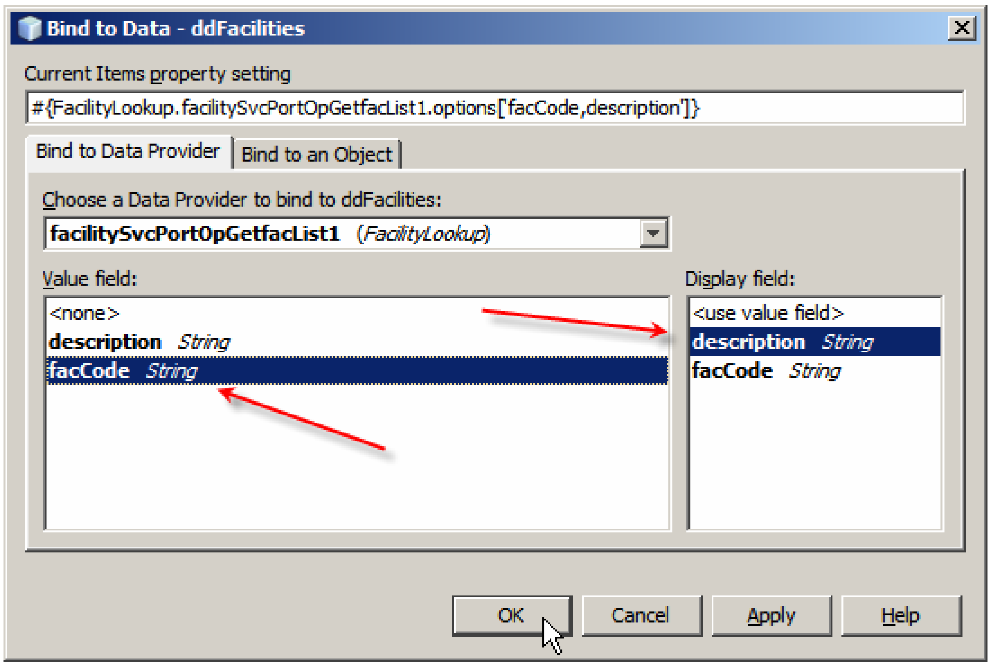
Right-click the drop down component and choose Add Binding Attribute.



Right-click the drop down component and choose Bind to Data.



Make sure facCode is selected in the Value field list and description is selected in Display field list, the click OK.

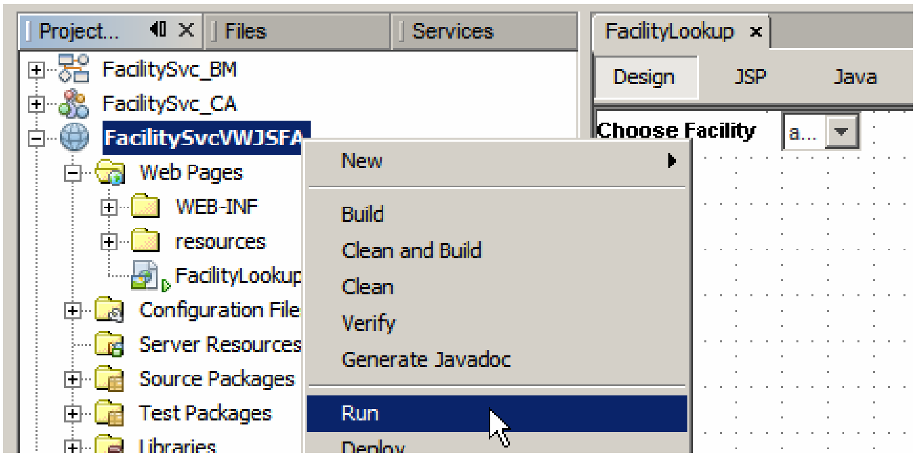


Bind to Data - ddFacilities

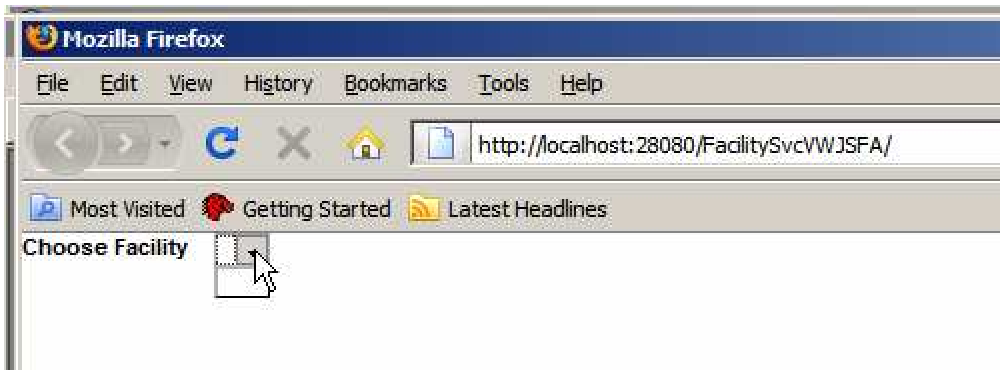
Current Items property setting

#{FacilityLookup.facilitySvcPortOpGetfacList1.options['facCode,description']}

Switch to the project view, right-click the name of the project and choose Run.

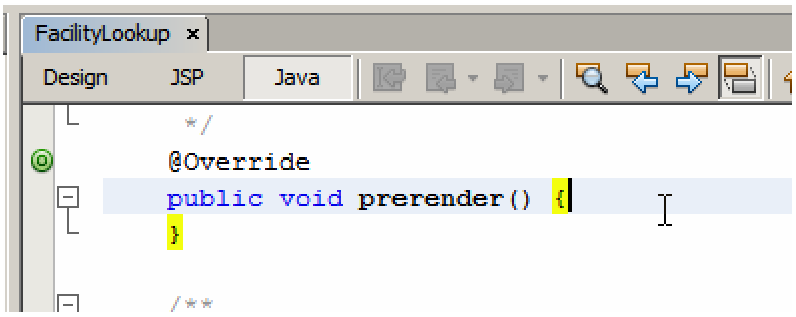


Your default web browser will be open with the web application as it stands now.



Note that the drop-down is empty. The server.log shows exceptions as well. We need to initialize the call to the web service by providing dummy string value.

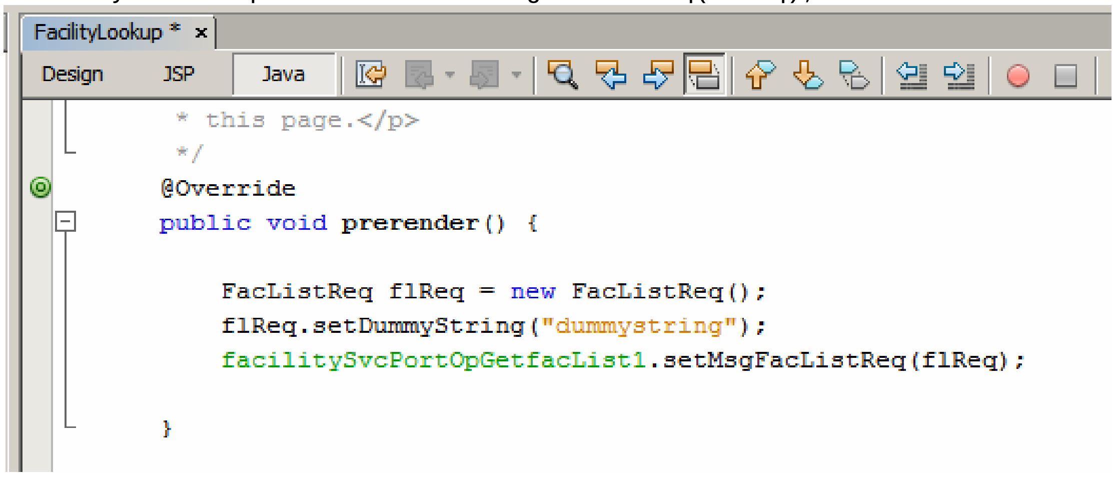
Switch to Java Tab and scroll down to the prerender() method.



```
    */
    @Override
    public void prerender() {
    }

    /**
```

Into the prerender method insert the following code.

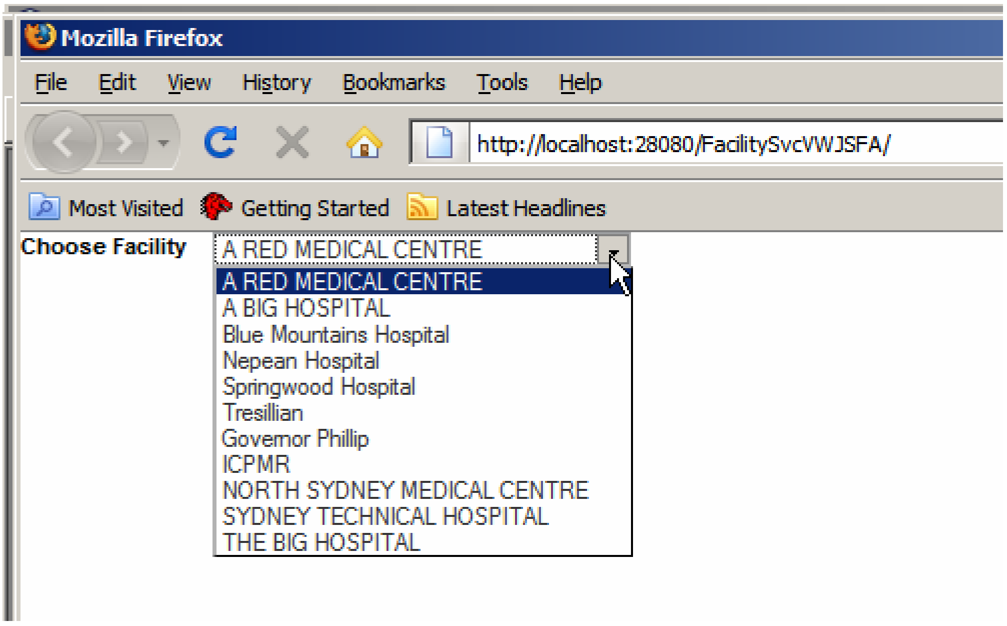
```
FacListReq flReq = new FacListReq();
flReq.setDummyString("dummystring");
facilitySvcPortOpGetfacList1.setMsgFacListReq(flReq);
```



```
     * this page.</p>
     */
    @Override
    public void prerender() {

        FacListReq flReq = new FacListReq();
        flReq.setDummyString("dummystring");
        facilitySvcPortOpGetfacList1.setMsgFacListReq(flReq);

    }
```

Run the application again.

There are no exceptions in the log and the drop-down contains descriptions of all facilities.



Choose Facility

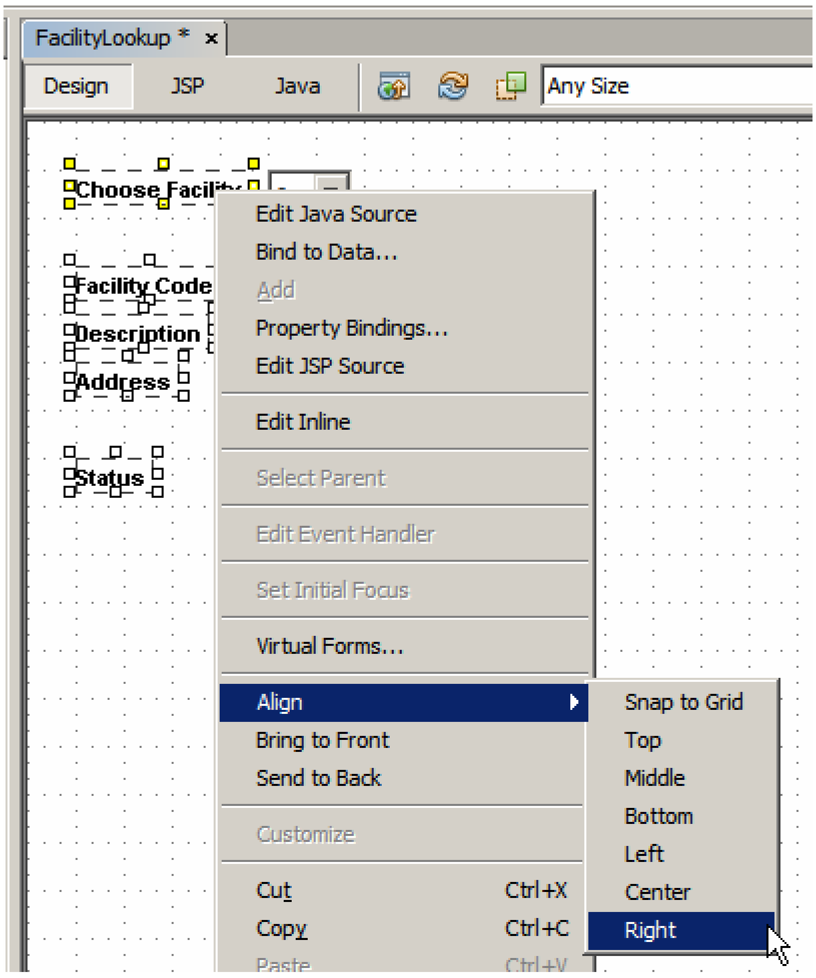
A RED MEDICAL CENTRE

A RED MEDICAL CENTRE
A BIG HOSPITAL
Blue Mountains Hospital
Nepean Hospital
Springwood Hospital
Tresillian
Governor Phillip
ICPMR
NORTH SYDNEY MEDICAL CENTRE
SYDNEY TECHNICAL HOSPITAL
THE BIG HOSPITAL

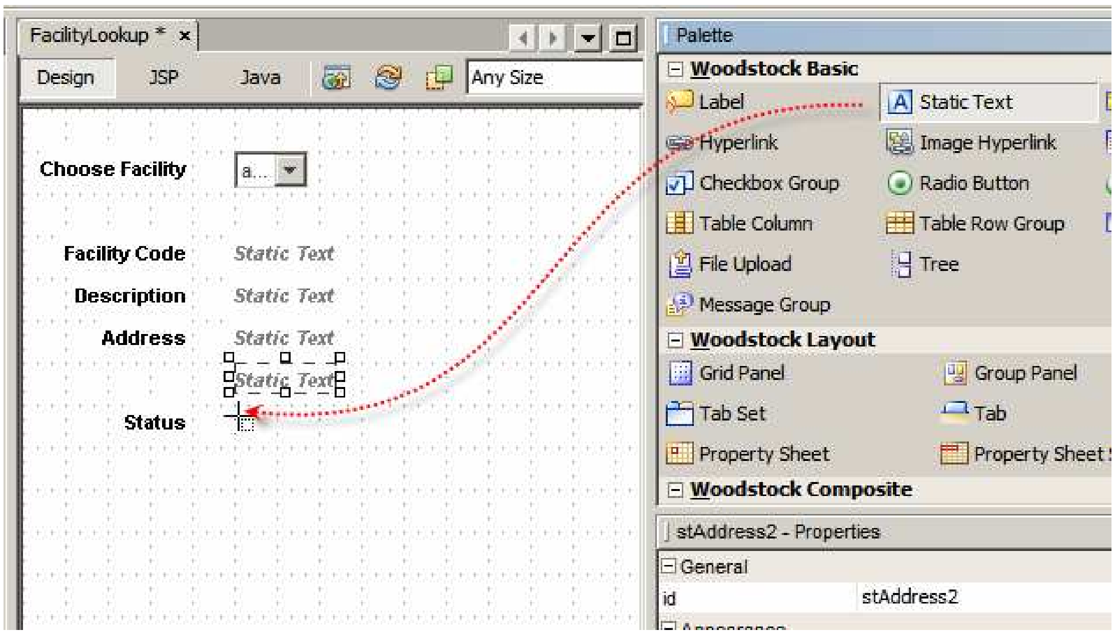
Now that we have the drop down being populated we would like to display details of the selected facility each time facility changes in the dropdown.

Let's add 4 labels and 5 static fields to contain descriptions and values of the various facility properties.

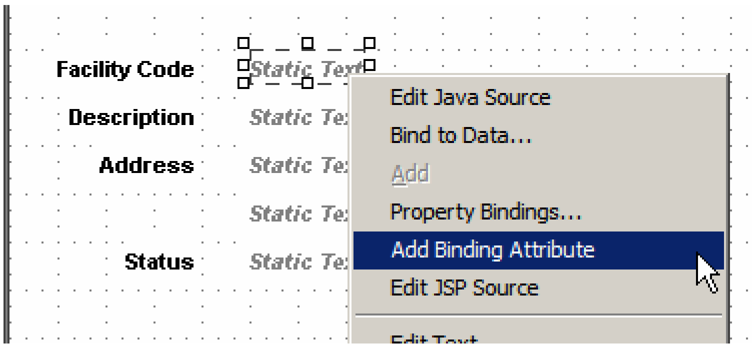
Let's add labels for "Facility Code", "Description", "Address" and "Status", select all labels, right-click and choose Align and choose Right.



Let's now drag 5 static fields and give them ID of stFacCode, stDescription, stAddress1, stAddress2 and stStatus respectively.

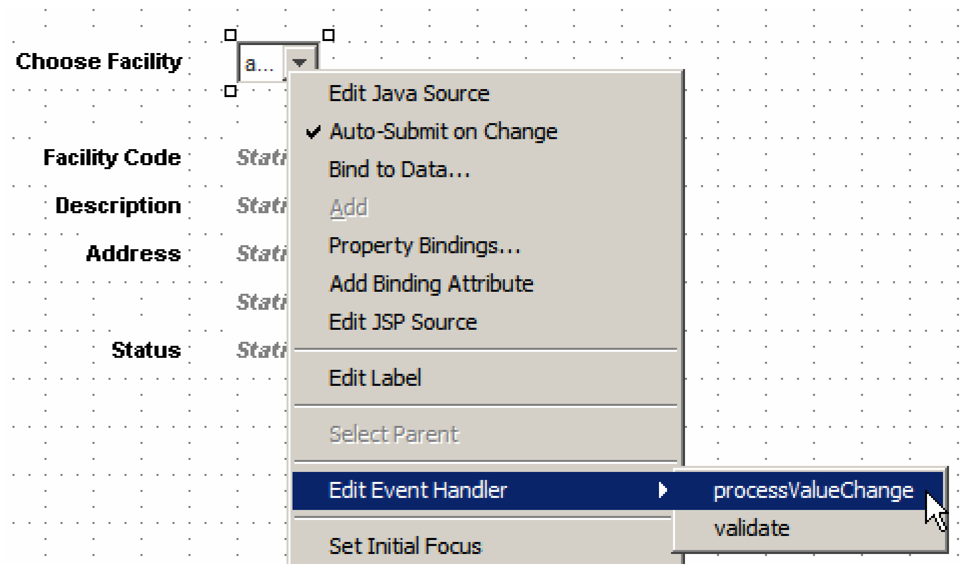


Now right-click on each static field in turn and choose Add Binding Attribute.

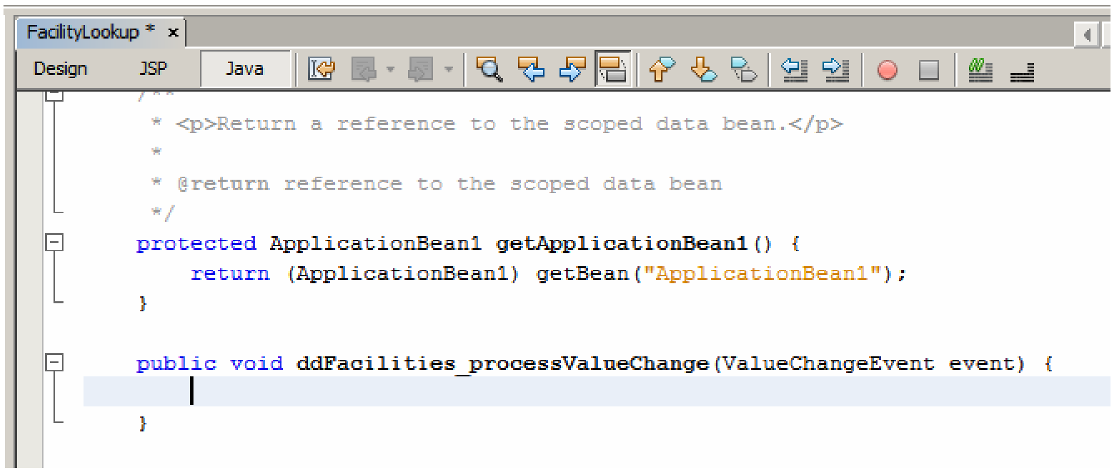


This will allow us to manipulate contents of the fields at runtime.

So far nothing happens if we change the value of the ddFacilities drop-down. Let’s add a handler to be executed when the value changes. Right-click the drop dpwn component, choose Edit Event Handler and choose processValueChange.



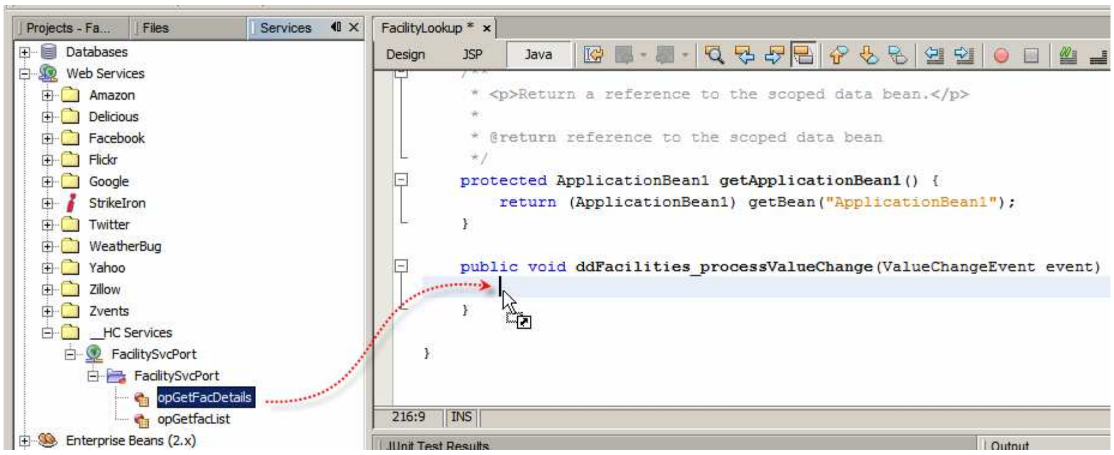
This switches display to the Java mode, inserts a skeleton ddFacilities_processValueChange() method and allows us to add custome Java code.



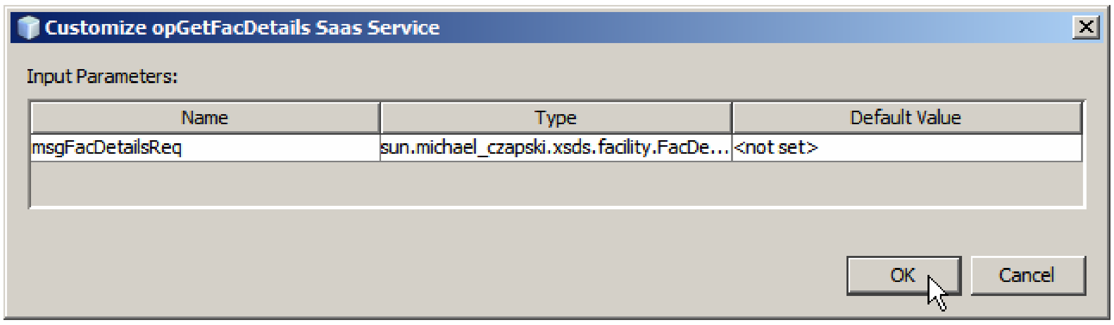
```
     * <p>Return a reference to the scoped data bean.</p>
     *
     * @return reference to the scoped data bean
     */
    protected ApplicationBean1 getApplicationBean1() {
        return (ApplicationBean1) getBean("ApplicationBean1");
    }

    public void ddFacilities_processValueChange(ValueChangeEvent event) {
        
    }
```

What we need to do here is to look at the new value of the ddFacilty (facility code), use this code to look up facility details and populate facility details-related static fields with the property values for the designated facility.

Let's switch to the Services Tab, locate the opGetFacDetails web service operation and drag it onto the Java canvas inside the ddFacility_processValueChange method.



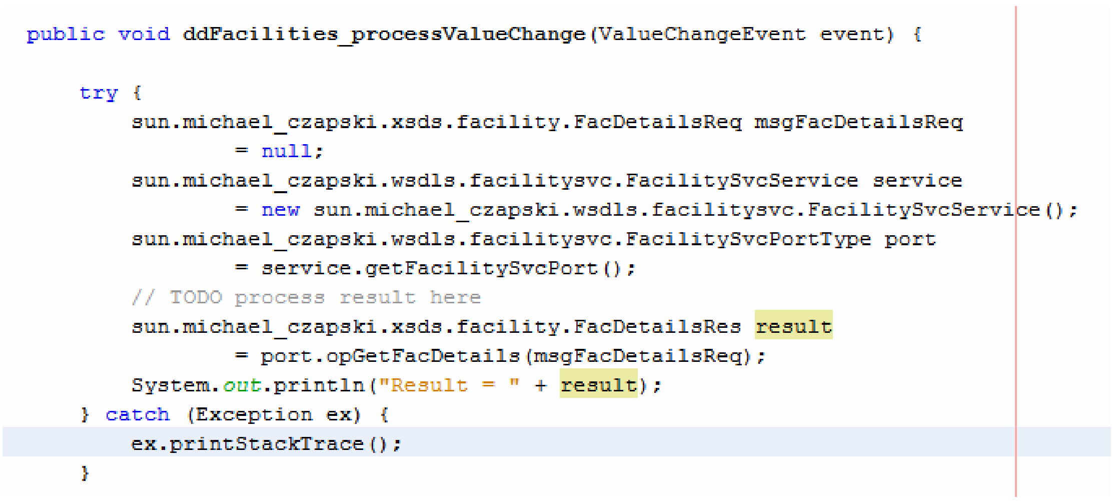
Accept defaults in the dialog box that results.



Input Parameters:

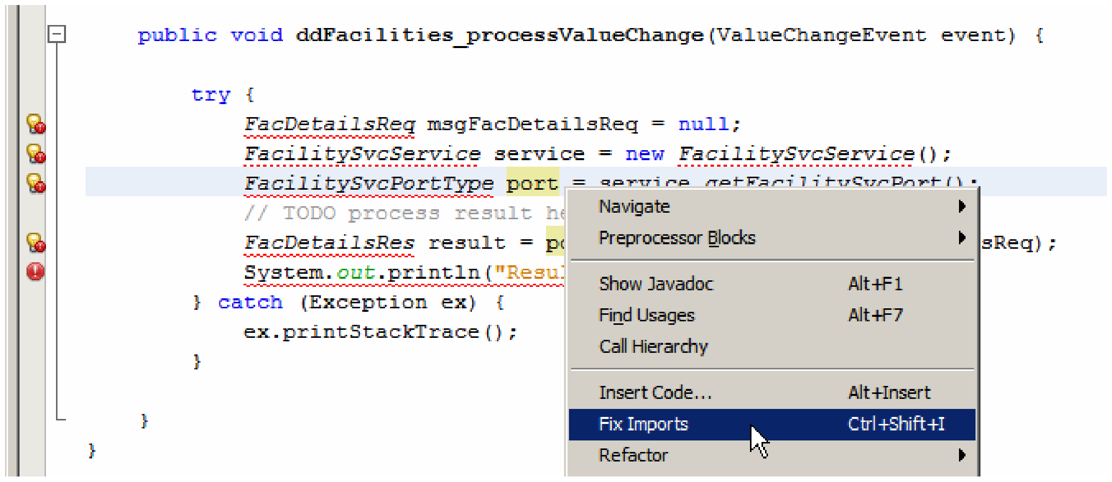
Name	Type	Default Value
msgFacDetailsReq	sun.michael_czapski.xsds.facility.FacDe...	<not set>

A slab of boilerplate code will be added. I re-formatted it for better visibility.



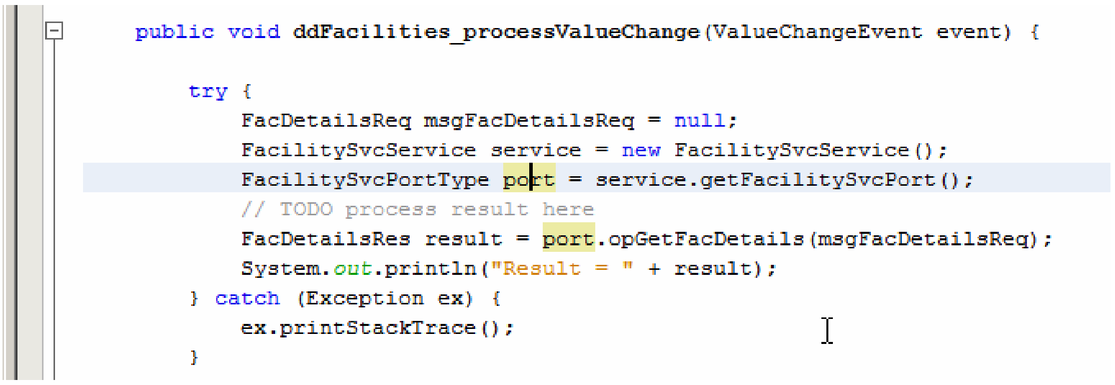
```
public void ddFacilities_processValueChange(ValueChangeEvent event) {

    try {
        sun.michael_czapski.xsds.facility.FacDetailsReq msgFacDetailsReq
                = null;
        sun.michael_czapski.wsdls.facilitysvc.FacilitySvcService service
                = new sun.michael_czapski.wsdls.facilitysvc.FacilitySvcService();
        sun.michael_czapski.wsdls.facilitysvc.FacilitySvcPortType port
                = service.getFacilitySvcPort();
        // TODO process result here
        sun.michael_czapski.xsds.facility.FacDetailsRes result
                = port.opGetFacDetails(msgFacDetailsReq);
        System.out.println("Result = " + result);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
```

Let’s remove sun.michael_czapski.xsds.facility and sun.michael_czapski.wsdld.facility svc strings to minimize clutter and use NetBeans IDE facilities to resolve references through the Java import mechanism. Right-click anywhere inside the Java source window and choose Fix Imports.



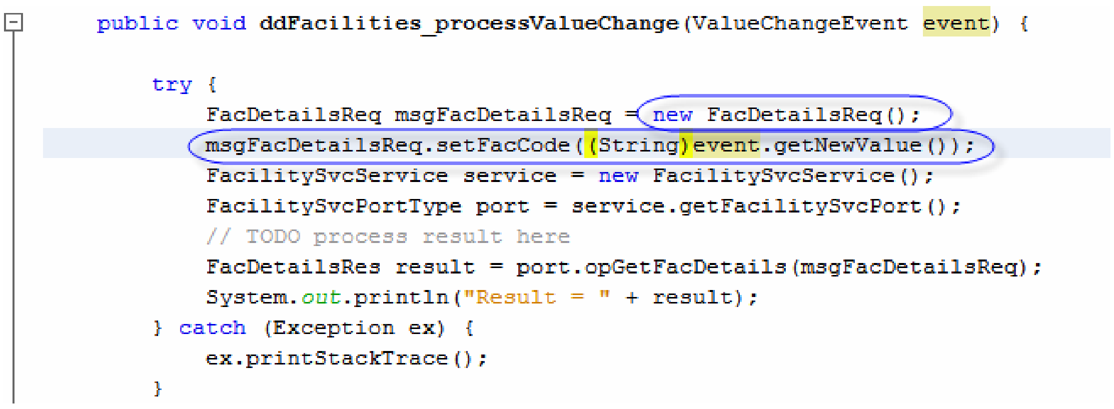
```
    public void ddFacilities_processValueChange(ValueChangeEvent event) {

        try {
            FacDetailsReq msgFacDetailsReq = null;
            FacilitySvcService service = new FacilitySvcService();
            FacilitySvcPortType port = service.getFacilitySvcPort();
            // TODO process result he
            FacDetailsRes result = pc                                  sReq);
            System.out.println("Resu
        } catch (Exception ex) {
            ex.printStackTrace();
        }

    }
}
```

The brokenness will be resolved and appropriate import statements will be added.



```
    public void ddFacilities_processValueChange(ValueChangeEvent event) {

        try {
            FacDetailsReq msgFacDetailsReq = null;
            FacilitySvcService service = new FacilitySvcService();
            FacilitySvcPortType port = service.getFacilitySvcPort();
            // TODO process result here
            FacDetailsRes result = port.opGetFacDetails(msgFacDetailsReq);
            System.out.println("Result = " + result);
        } catch (Exception ex) {
            ex.printStackTrace();
        }
```

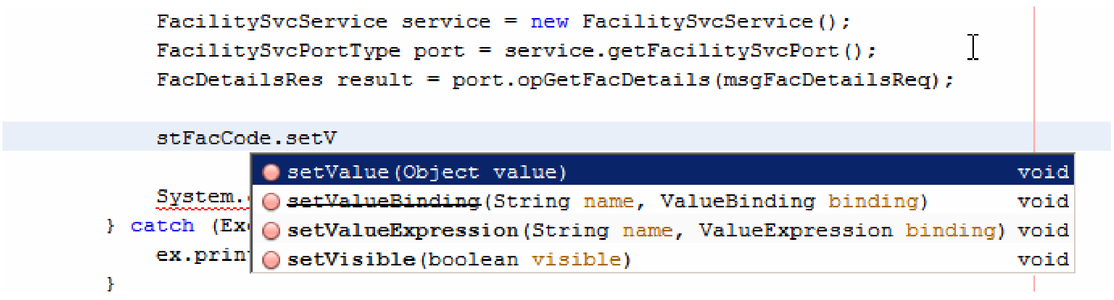
Looking at the statement `FacDetailsReq msgFacDetailsReq = null;` we deduce that we have an opportunity to populate the request message with the facility code and that we will get a response containing facility details 4 statements later. Let’s populate the request message with the new facility code chosen in the drop down.



```
public void ddFacilities_processValueChange(ValueChangeEvent event) {

    try {
        FacDetailsReq msgFacDetailsReq = new FacDetailsReq();
        msgFacDetailsReq.setFacCode((String)event.getNewValue());
        FacilitySvcService service = new FacilitySvcService();
        FacilitySvcPortType port = service.getFacilitySvcPort();
        // TODO process result here
        FacDetailsRes result = port.opGetFacDetails(msgFacDetailsReq);
        System.out.println("Result = " + result);
    } catch (Exception ex) {
        ex.printStackTrace();
    }
```

Now the service execution (port.opGetFacDetails(…)) will give us details for the chosen facility. We can use the response message to populate all the static fields we created earlier.

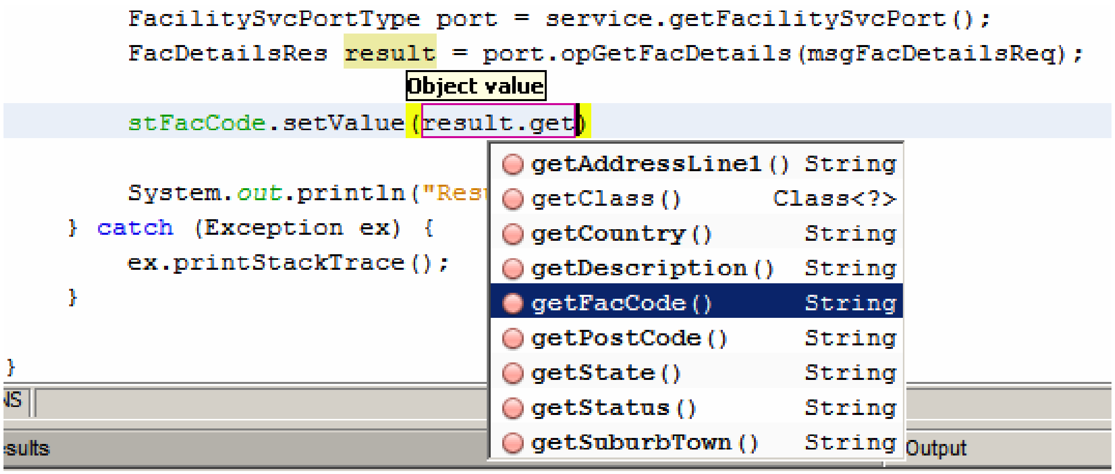
As you enter this code note how NetBeans helps through code completion and other clever tricks.



```
        FacilitySvcService service = new FacilitySvcService();
        FacilitySvcPortType port = service.getFacilitySvcPort();
        FacDetailsRes result = port.opGetFacDetails(msgFacDetailsReq);

        stFacCode.setV

        System.
    } catch (Ex
        ex.prin
    }
```

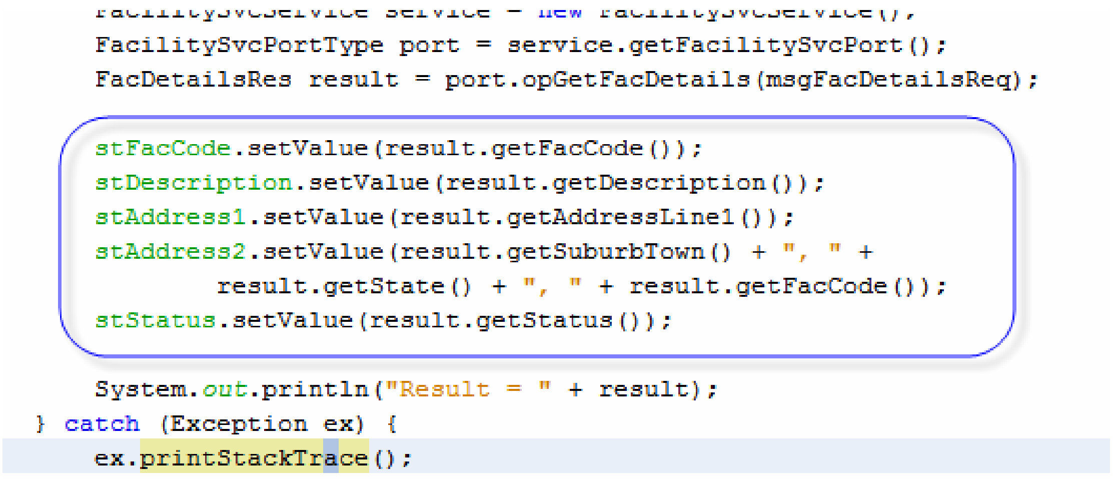
```
setValue(Object value)                                       void
~~setValueBinding~~(String name, ValueBinding binding)           void
setValueExpression(String name, ValueExpression binding)     void
setVisible(boolean visible)                                  void
```

.



```
        FacilitySvcPortType port = service.getFacilitySvcPort();
        FacDetailsRes result = port.opGetFacDetails(msgFacDetailsReq);
                                 Object value
        stFacCode.setValue(result.get)

        System.out.println("Res
    } catch (Exception ex) {
        ex.printStackTrace();
    }

}
```

```
getAddressLine1()  String
getClass()         Class<?>
getCountry()       String
getDescription()   String
getFacCode()       String
getPostCode()      String
getState()         String
getStatus()        String
getSuburbTown()    String
```

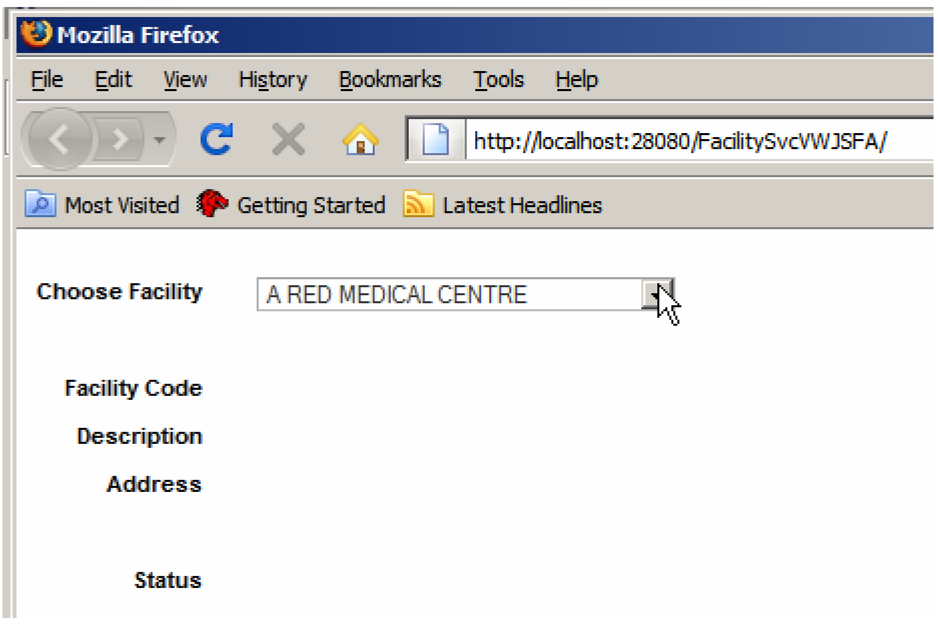


```
        FacilitySvcService service = new FacilitySvcService();
        FacilitySvcPortType port = service.getFacilitySvcPort();
        FacDetailsRes result = port.opGetFacDetails(msgFacDetailsReq);

        stFacCode.setValue(result.getFacCode());
        stDescription.setValue(result.getDescription());
        stAddress1.setValue(result.getAddressLine1());
        stAddress2.setValue(result.getSuburbTown() + ", " +
                result.getState() + ", " + result.getFacCode());
        stStatus.setValue(result.getStatus());

        System.out.println("Result = " + result);
    } catch (Exception ex) {
        ex.printStackTrace();
```

Our ddFacility_processValueChange() method body is completed. Each time we change the value in the drop down we will get the details of the facility displayed.

Let's run the application, as it stands now, to see what it looks like and how it behaves.

Notice that we have the populated drop down, as we did before, but no facility details for the selected facility.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:28080/FacilitySvcVWJSFA/
Most Visited Getting Started Latest Headlines

Choose Facility A RED MEDICAL CENTRE

Facility Code

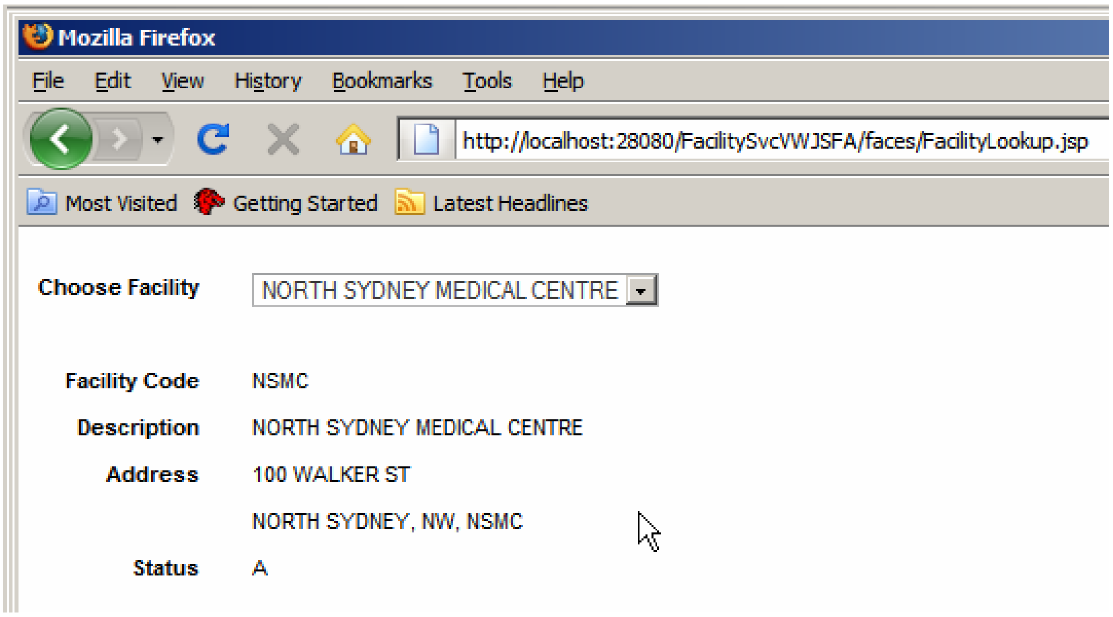
Description

Address

Status

Let's choose a different facility and see what happens.

This time details for the new facility got displayed.


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:28080/FacilitySvcVWJSFA/faces/FacilityLookup.jsp
Most Visited Getting Started Latest Headlines

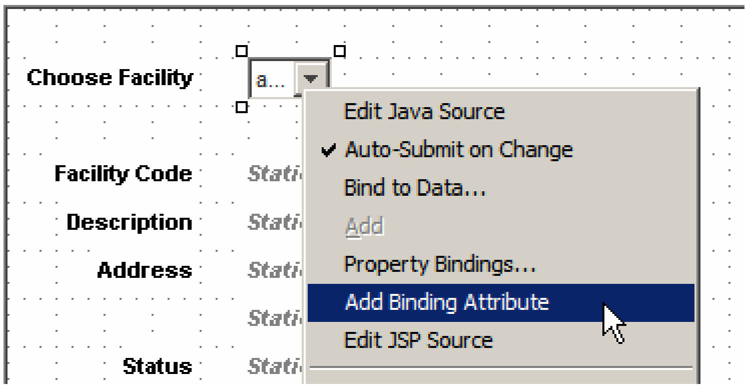
Choose Facility NORTH SYDNEY MEDICAL CENTRE

Facility Code NSMC
Description NORTH SYDNEY MEDICAL CENTRE
Address 100 WALKER ST
NORTH SYDNEY, NW, NSMC
Status A

The reason we did not see the details when the web page first appeared was because the ddFacility_provessValueChange was not executed the first time around. To make it execute we need to add some code to the prerender method to first establish whether this is the first time that the page is displayed, then to work out what the facility code is the first time around and finally to force ddFacility_processValueChange method to get executed so the facility details get displayed.

Let's do these things a bit at a time.

Let's switch to the Design mode, right-click on the drop down and choose Add Binding Attribute.



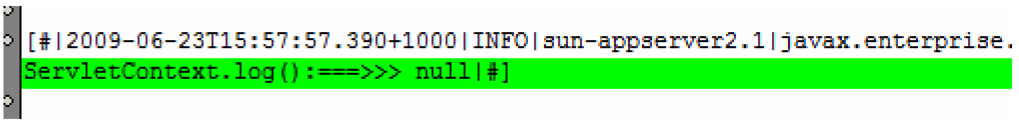
Let's switch to Java source mode and scroll down to the prerender() method.

At the end of the code already in the pre-render method let's add the following:

```
log("===>>> " + ddFacilities.getValue());
```

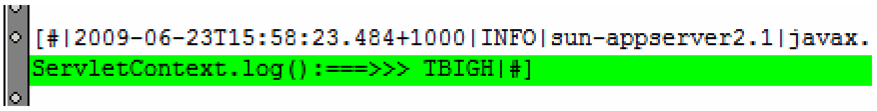
Run the application again. Look at the server.log to see what this statement diaplays the first time the application is run, the change facility in the drop down and see what this statement displays the second time around.

The first time around I see:



```
[#|2009-06-23T15:57:57.390+1000|INFO|sun-appserver2.1|javax.enterprise.
ServletContext.log():===>>> null|#]
```

When I choose “The Big Hospital” I see:

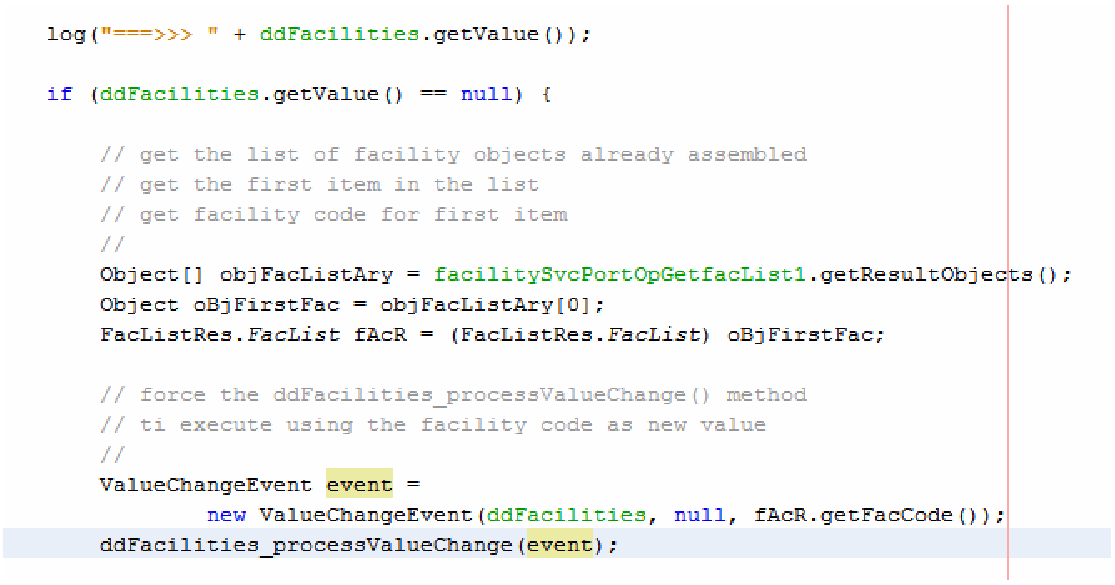


```
[#|2009-06-23T15:58:23.484+1000|INFO|sun-appserver2.1|javax.
ServletContext.log():===>>> TBIGH|#]
```

Knowing that this is happening allows us to determine whether we are displaying the first time or the subsequent time.

First time around, as we noticed, we have no idea what the “selected” code is. We may or may not be aware that at this point in the code the opGetFacList operation was executed and the list of facilities is available. We do know that the first time around the first facility will be the one “selected” in the drop down. We can exploit both of these to work out what the facility code is.

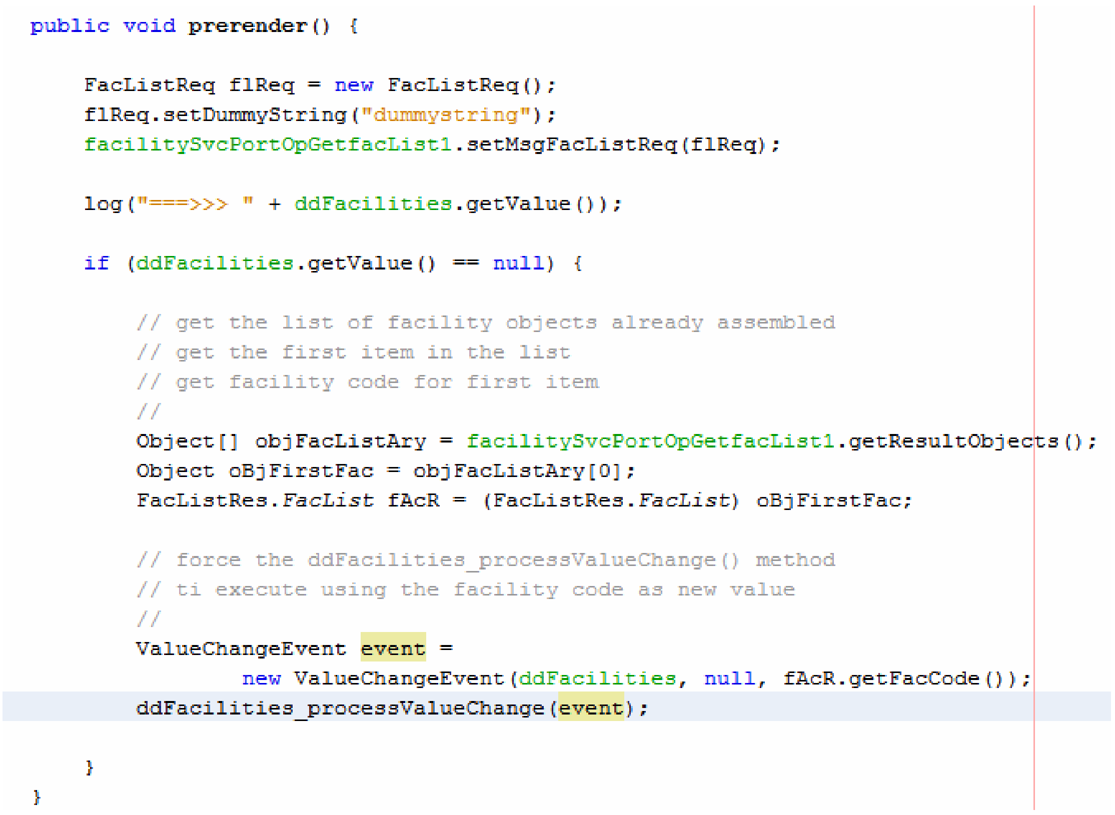
Let’s add the following slab of code to the end of the prerender() method.



```
log("===>>> " + ddFacilities.getValue());

if (ddFacilities.getValue() == null) {

    // get the list of facility objects already assembled
    // get the first item in the list
    // get facility code for first item
    //
    Object[] objFacListAry = facilitySvcPortOpGetfacList1.getResultObjects();
    Object oBjFirstFac = objFacListAry[0];
    FacListRes.FacList fAcR = (FacListRes.FacList) oBjFirstFac;

    // force the ddFacilities_processValueChange() method
    // ti execute using the facility code as new value
    //
    ValueChangeEvent event =
            new ValueChangeEvent(ddFacilities, null, fAcR.getFacCode());
    ddFacilities_processValueChange(event);
```

I will not explain what this does. Suffice it to say that facilitySvcPortOpGetfacList1.getResultObjects() is a reference to the web service-returned list of facility objects. The first item in that list will give us access to the facility code. We create a new event in which we are setting the new value to that code and explicitly executing ddFacility_processValueChange method. This causes the static fields to be populated.

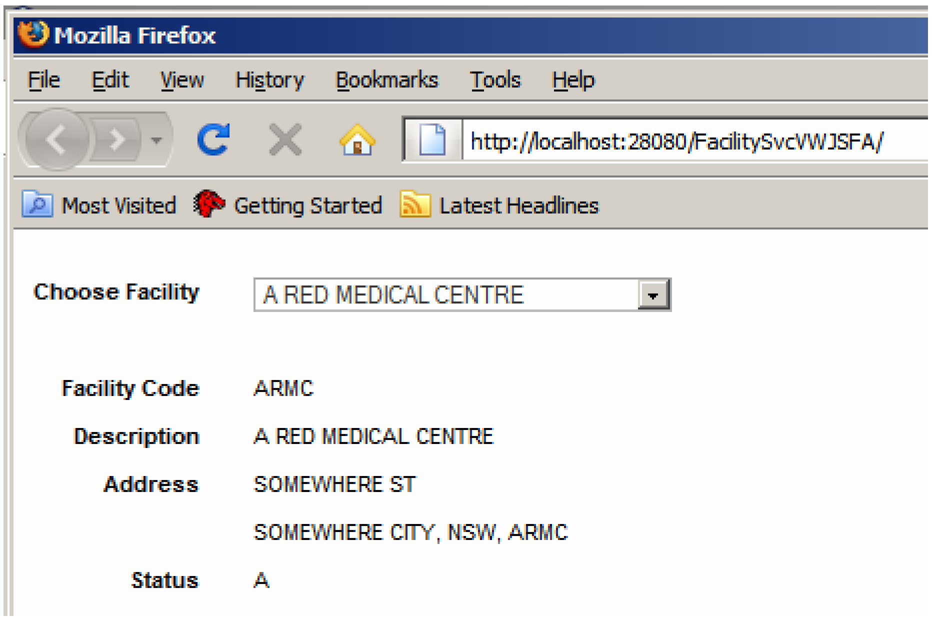
Here is the complete prerender() method.



```java
public void prerender() {

    FacListReq flReq = new FacListReq();
    flReq.setDummyString("dummystring");
    facilitySvcPortOpGetfacList1.setMsgFacListReq(flReq);

    log("===>>> " + ddFacilities.getValue());

    if (ddFacilities.getValue() == null) {

        // get the list of facility objects already assembled
        // get the first item in the list
        // get facility code for first item
        //
        Object[] objFacListAry = facilitySvcPortOpGetfacList1.getResultObjects();
        Object oBjFirstFac = objFacListAry[0];
        FacListRes.FacList fAcR = (FacListRes.FacList) oBjFirstFac;

        // force the ddFacilities_processValueChange() method
        // ti execute using the facility code as new value
        //
        ValueChangeEvent event =
                new ValueChangeEvent(ddFacilities, null, fAcR.getFacCode());
        ddFacilities_processValueChange(event);

    }
}
```

Let’s run the application again, see what we get the first time around and what we get when we change the facility selection.

First time around:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:28080/FacilitySvcVWJSFA/
Most Visited Getting Started Latest Headlines

Choose Facility A RED MEDICAL CENTRE

Facility Code ARMC

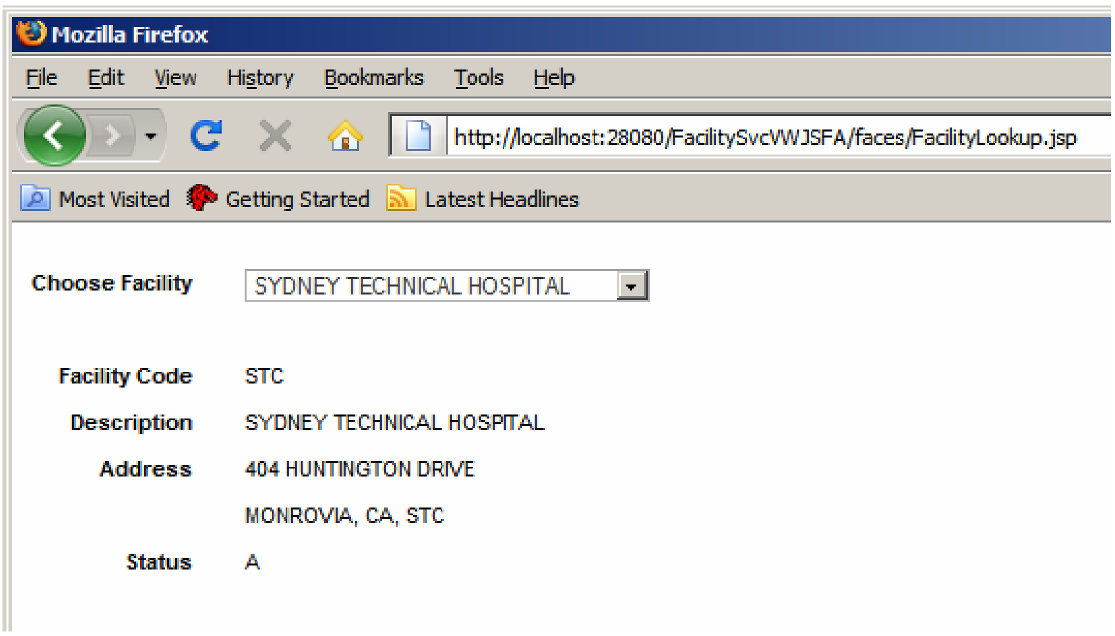
Description A RED MEDICAL CENTRE

Address SOMEWHERE ST

SOMEWHERE CITY, NSW, ARMC

Status A

Choosing Sydney Technical Hospital:


Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://localhost:28080/FacilitySvcVWJSFA/faces/FacilityLookup.jsp
Most Visited Getting Started Latest Headlines

Choose Facility: SYDNEY TECHNICAL HOSPITAL

Facility Code: STC

Description: SYDNEY TECHNICAL HOSPITAL

Address: 404 HUNTINGTON DRIVE
MONROVIA, CA, STC

Status: A

This is it. That's what it took to develop a web application that used the Facilities web service as a data provider.

Summary

In this document we created and exercised a web application that provided a list of Healthcare Facilities as a drop down and details of a specific Facility when chosen.

We used the NetBenas 6.5.1 IDE, included with the GlassFish ESB v 2.1 infrastructure. We used the Visual Web JavaServer Faces technologies, Project Woodstock JSF components and JBI-based multi-operation web service. The NetBeans IDE tooling assisted in rapidly developing the web application with minimum of custom Java code. This web application, a component in SOA 1, Presentation Layer, consumes SOA 3, Business Service service in a loosely coupled manner.