NetBeans 6.5.1, GlassFish v 2.1, Web Space Server 10
Patient Lookup Visual Web JSF Portlet with a basic Google Map

Michael.Czapski@sun.com
July 2009

Table of Contents

Abstract

In this walkthrough I will add a basic Google Map panel to the Visual Web JSF Portlet, developed in the previous writeup, “GlassFish ESB v2.1, Web Space Server 10 – Creating a Patient Lookup Visual Web JSF Portlet”, at http://blogs.sun.com/javacapsfieldtech/entry/creating_a_patient_lookup_visual, which uses Facility-related and Patient-related Web Service as data providers to implement search for Patient Details.

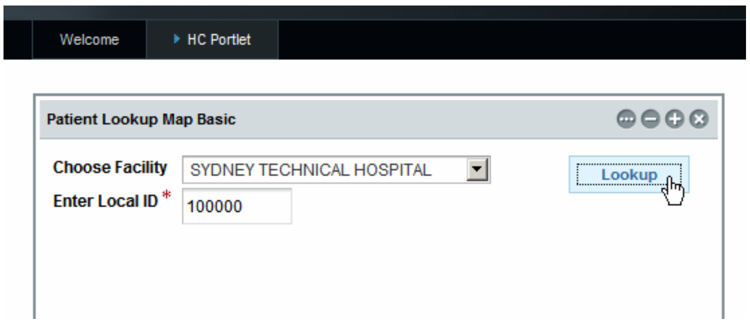
Introduction

The business idea behind the functionality developed in this walkthrough is that patients are looked after in various healthcare facilities. Healthcare workers need to lookup patient details such as their identifier, gender, birth date or address. A relational database holds patient details as well as other information of relevance such as descriptions of various coded values. Patient details are available through a web service. Facility list and details, used to narrow down the search for patients to a specific facility, are available through a web service. These web services will be used to construct the Portlet that will allow patient search and a display of patient details with display a Google Map, centered at patient’s address, if one is available and is valid for the purpose of mapping. This Portlet will be deployed to the Sun FOSS Web Space Server 10 Portal.

The previous document [9], walked through development and deployment of the basic Patient Lookup Portlet. In this document I will add a basic Google Map to the Patient Lookup Portlet.

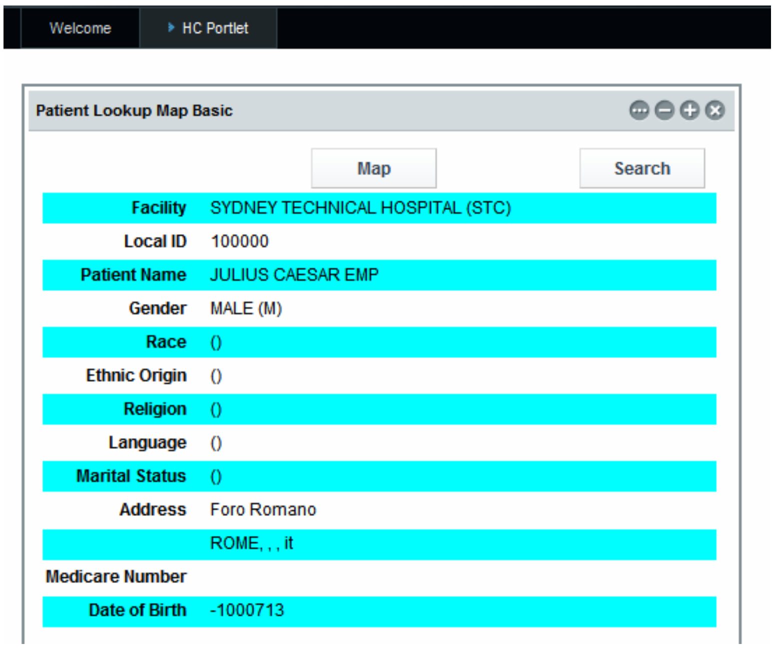
Other documents in this series, see pre-requisites, walked the reader through the process of implementing GlassFish ESB v2.1-based web services which return facility list and facility details as well as patient details.

To give you some idea of what we will get at the end of the process here are screenshots of the completed portlet running out of the Web Space Server Portal.



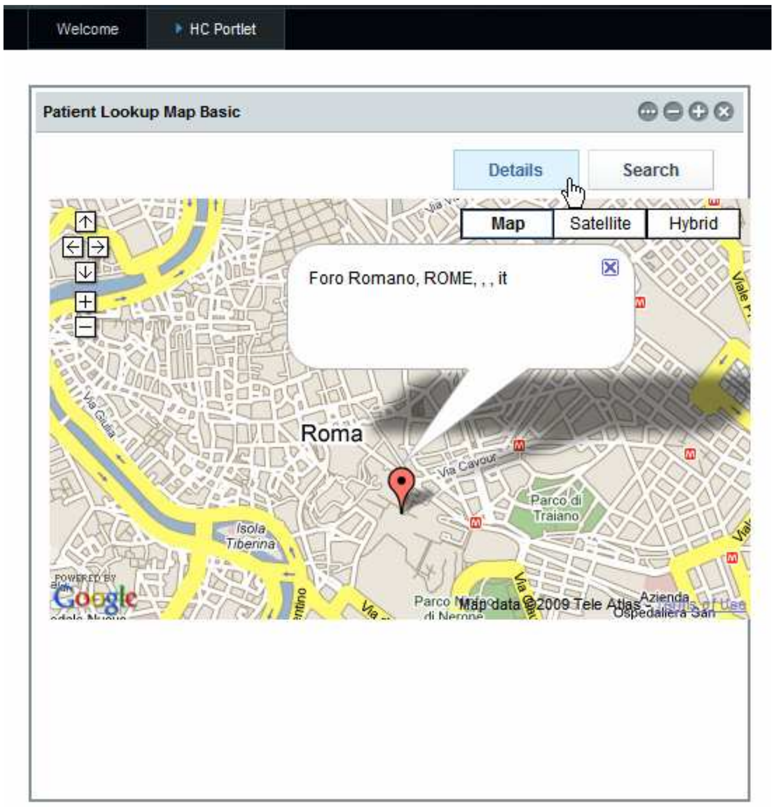
Patient Lookup Map Basic

Choose Facility	SYDNEY TECHNICAL HOSPITAL
Enter Local ID *	100000



Patient Lookup Map Basic

Facility	SYDNEY TECHNICAL HOSPITAL (STC)
Local ID	100000
Patient Name	JULIUS CAESAR EMP
Gender	MALE (M)
Race	()
Ethnic Origin	()
Religion	()
Language	()
Marital Status	()
Address	Foro Romano
	ROME, , , it
Medicare Number	
Date of Birth	-1000713



Patient Lookup Map Basic

Foro Romano, ROME, , , it

Note that this walkthrough builds on the Patient Lookup Portlet, built previously, but deals exclusively with Visual Web JSF portlet-related technologies, Java Script and Google Maps API.

Prerequisites

To work through this material certain pre-requisites have to be met.

It is assumed that:

- MySQL RDBMS is installed and available, as discussed in [1]
- GlassFish ESB v2.1 is installed, as discussed in [2]
- Sun Web Space Server Portal is installed, as discussed in [3]
- Web Space Server is configured as discussed in [4]
- Facility Service Web Service is implemented and deployed, as discussed in [5]
- Patient Service Web Service is implemented and deployed, as discussed in [6]
- Patient Lookup Portlet has been developed and tested [9]

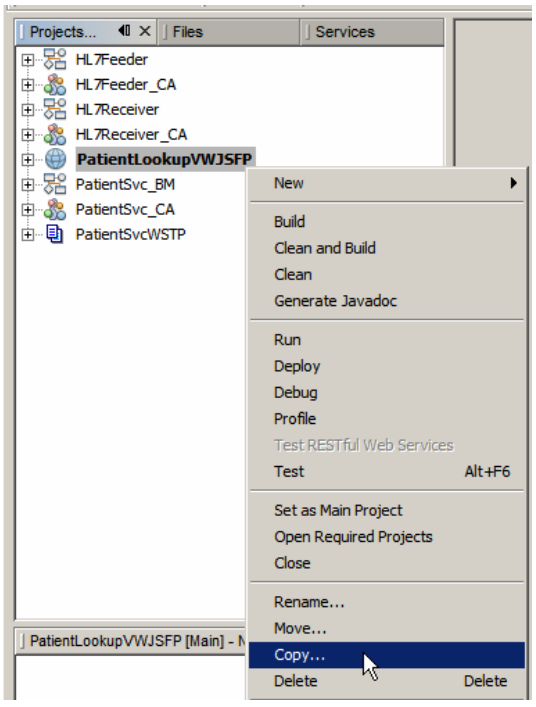
Unless these pre-requisites are met, you will not be able to complete this walkthrough.

Copy Patient Lookup Project

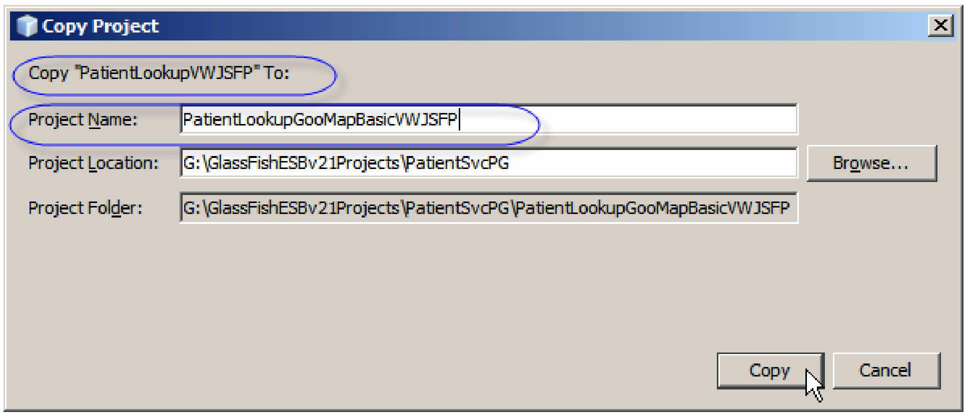
This document assumes that the Patient Lookup Portlet, developed in [9], is available for cloning.

To save the time and trouble we will copy the project PatientLookupVWJSFP [9] and use it as the basis for elaboration.

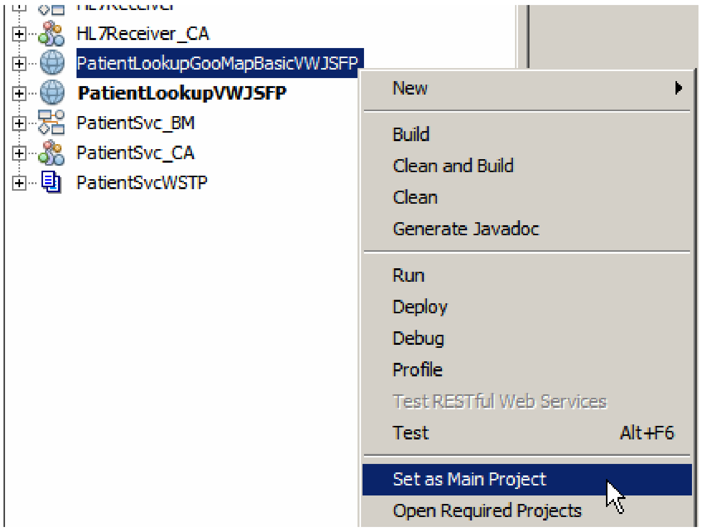
Right-click the name of the project and choose Copy.



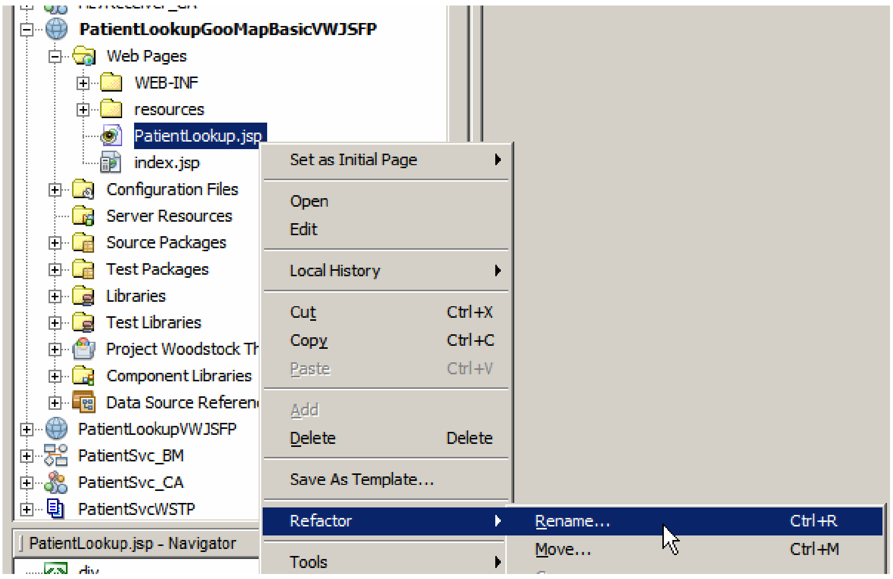
Name the new project PatientLookupGooMapBasicVWJSFP and click the Copy button.


Copy Project
Copy "PatientLookupVWJSFP" To:
Project Name: PatientLookupGooMapBasicVWJSFP
Project Location: G:\GlassFishESBv21Projects\PatientSvcPG
Browse...
Project Folder: G:\GlassFishESBv21Projects\PatientSvcPG\PatientLookupGooMapBasicVWJSFP
Copy
Cancel

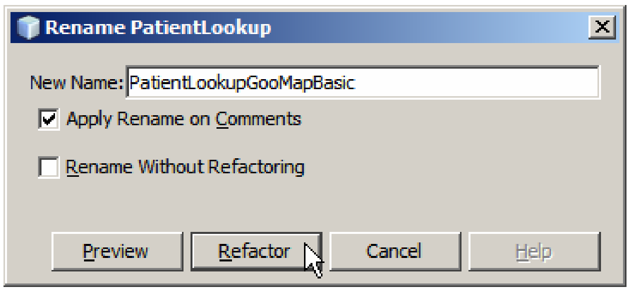
Right-click the name of the new project and choose "Set as Main Project".



Expand the project's Web Pages folder, right-click on the PatientLookup.jsp page and choose Refactor -> Rename.



Change the name to PatientLookupGooMapBasic, check the “Apply Rename on Comments” and click the “Refactor” button.

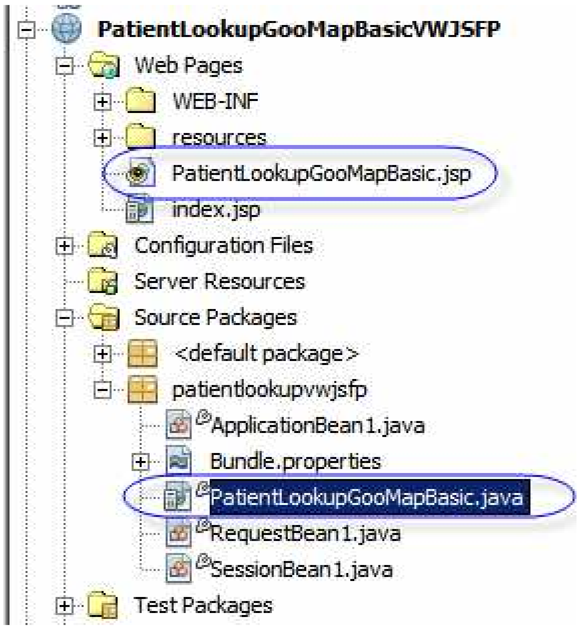

Rename PatientLookup

New Name: PatientLookupGooMapBasic

[✓] Apply Rename on Comments

[ ] Rename Without Refactoring

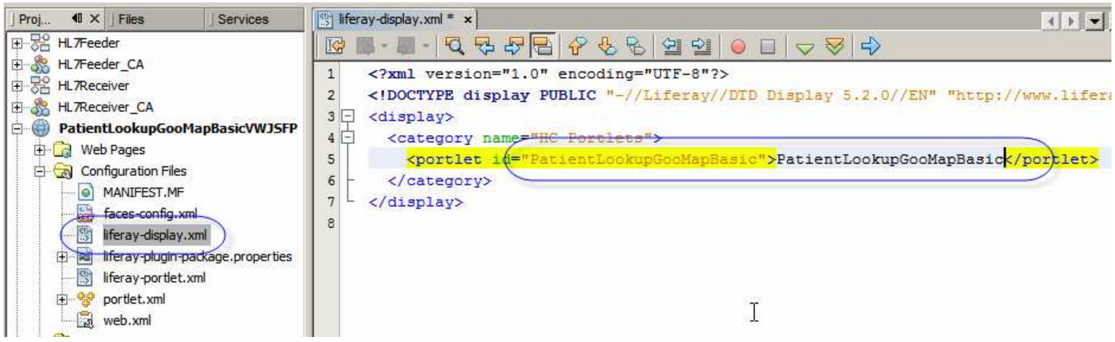
Note that the portlet backing class was also renamed.


```
PatientLookupGooMapBasicVWJSFP
  Web Pages
    WEB-INF
    resources
    PatientLookupGooMapBasic.jsp
    index.jsp
  Configuration Files
  Server Resources
  Source Packages
    <default package>
    patientlookupvwjsfp
      ApplicationBean1.java
      Bundle.properties
      PatientLookupGooMapBasic.java
      RequestBean1.java
      SessionBean1.java
  Test Packages
```

Alas, there are a few configuration files which must be manually modified.

Expand the “Configuration Files” folder.

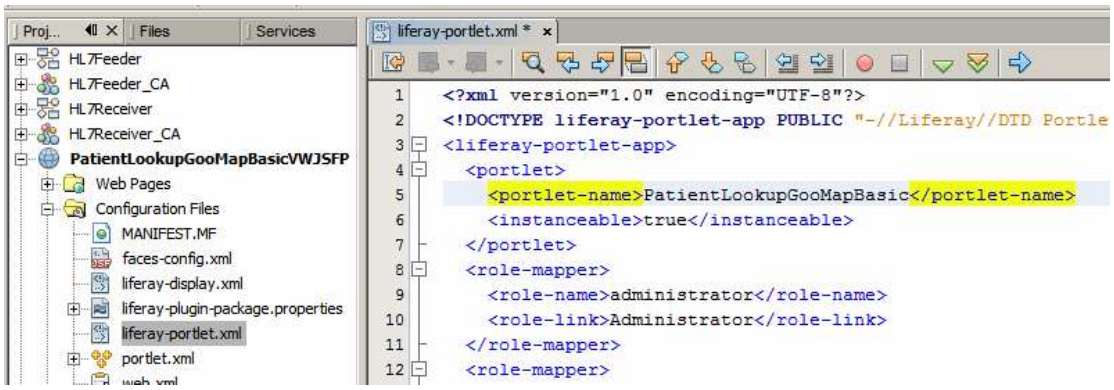
Open the liferay-display.xml and update portlet name and id.



```
HL7Feeder
HL7Feeder_CA
HL7Receiver
HL7Receiver_CA
PatientLookupGooMapBasicVWJSFP
  Web Pages
  Configuration Files
    MANIFEST.MF
    faces-config.xml
    liferay-display.xml
    liferay-plugin-package.properties
    liferay-portlet.xml
    portlet.xml
    web.xml
```

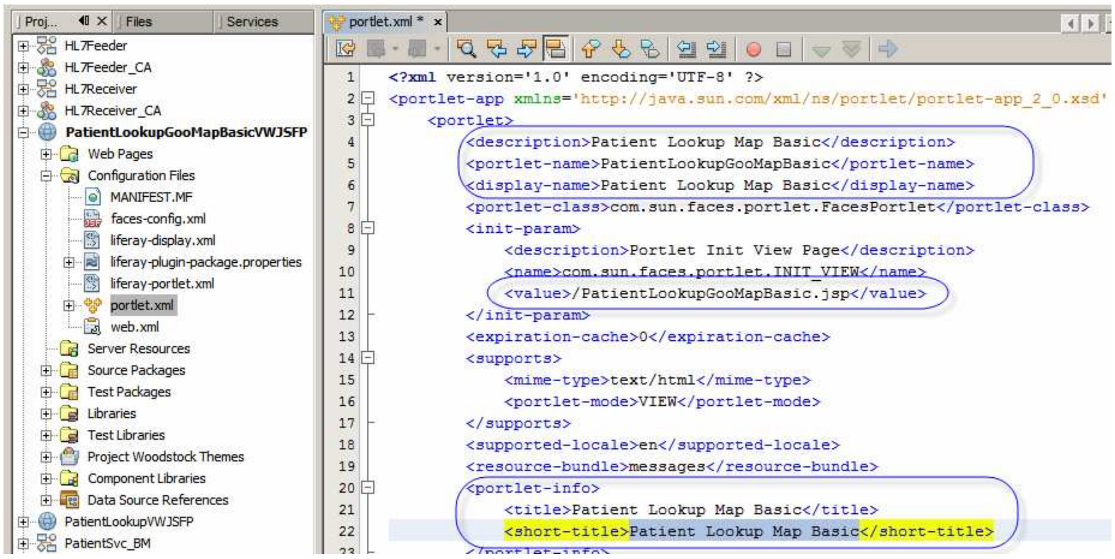
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE display PUBLIC "-//Liferay//DTD Display 5.2.0//EN" "http://www.lifera
<display>
  <category name="HC Portlets">
    <portlet id="PatientLookupGooMapBasic">PatientLookupGooMapBasic</portlet>
  </category>
</display>
```

Open liferay-portlet.xml and update portlet-name.



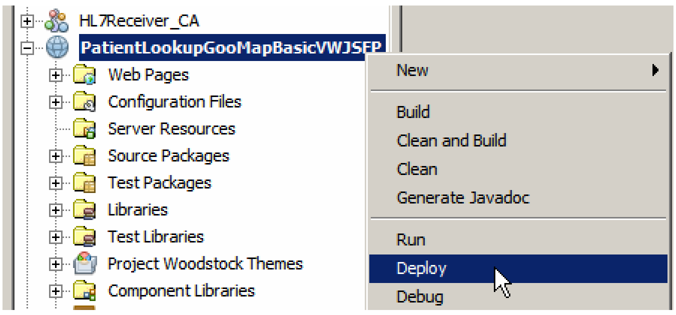
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE liferay-portlet-app PUBLIC "-//Liferay//DTD Portle
<liferay-portlet-app>
  <portlet>
    <portlet-name>PatientLookupGooMapBasic</portlet-name>
    <instanceable>true</instanceable>
  </portlet>
  <role-mapper>
    <role-name>administrator</role-name>
    <role-link>Administrator</role-link>
  </role-mapper>
  <role-mapper>
```

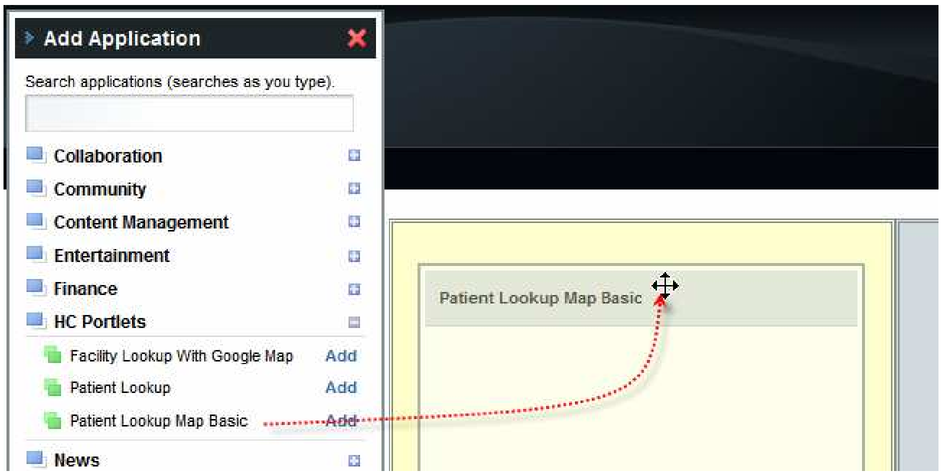
Open portlet.xml and update description, portlet-name, display-name, title and short-title. Of these only the portlet-name and init-param -> value are critical.

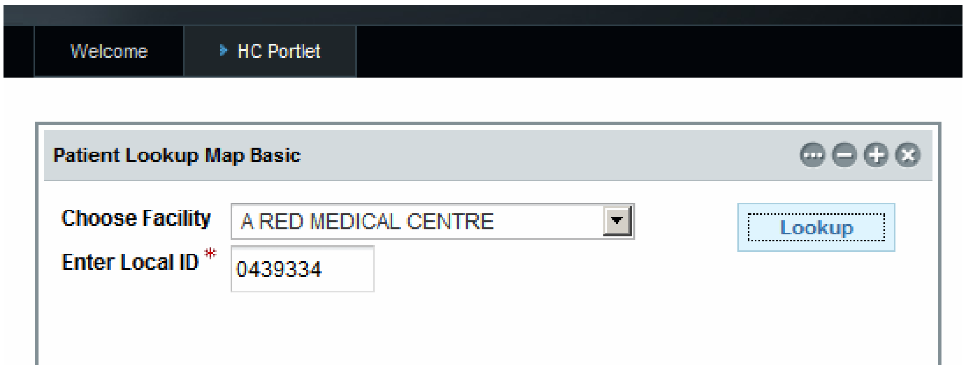


```
<?xml version='1.0' encoding='UTF-8' ?>
<portlet-app xmlns='http://java.sun.com/xml/ns/portlet/portlet-app_2_0.xsd'
    <portlet>
        <description>Patient Lookup Map Basic</description>
        <portlet-name>PatientLookupGooMapBasic</portlet-name>
        <display-name>Patient Lookup Map Basic</display-name>
        <portlet-class>com.sun.faces.portlet.FacesPortlet</portlet-class>
        <init-param>
            <description>Portlet Init View Page</description>
            <name>com.sun.faces.portlet.INIT_VIEW</name>
            <value>/PatientLookupGooMapBasic.jsp</value>
        </init-param>
        <expiration-cache>0</expiration-cache>
        <supports>
            <mime-type>text/html</mime-type>
            <portlet-mode>VIEW</portlet-mode>
        </supports>
        <supported-locale>en</supported-locale>
        <resource-bundle>messages</resource-bundle>
        <portlet-info>
            <title>Patient Lookup Map Basic</title>
            <short-title>Patient Lookup Map Basic</short-title>
```

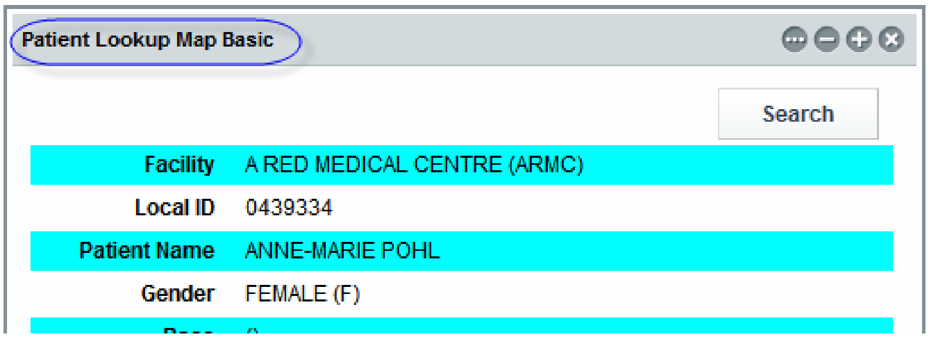
Once done, deploy the portlet and exercise it in the browser to make sure it still functions.







Choose Facility: A RED MEDICAL CENTRE
Enter Local ID *: 0439334



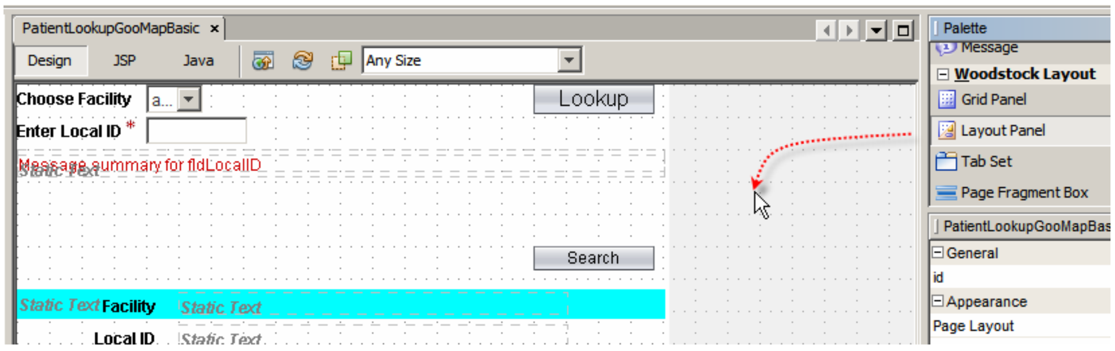
Facility	A RED MEDICAL CENTRE (ARMC)
Local ID	0439334
Patient Name	ANNE-MARIE POHL
Gender	FEMALE (F)

It works for me.

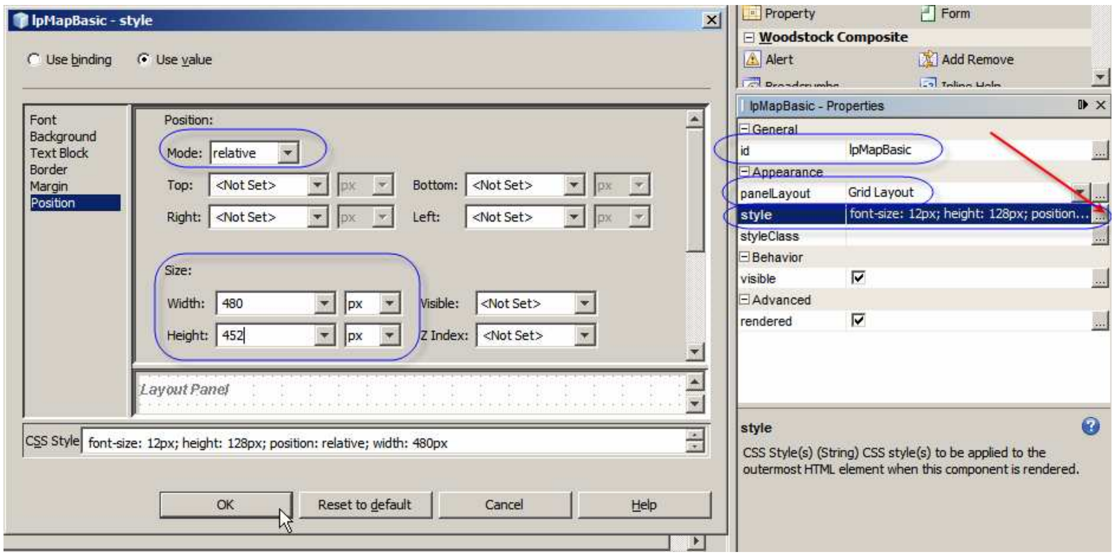
Add Google Map components

If you implemented the Facility Portlet with Google Map [10] some of the material here may be familiar.

Open the PatientLookupGooMapBasic.jsp in Design mode. Drag the Woodstock Layout Panel to the canvas anywhere outside the existing layout panels.



Change the id property to lpMapBasic, panelLayout property to “Grid Layout” and style property attributes to: font-size:12px, position: relative, height: 452px, width: 480px.


lpMapBasic - style

Use binding [✓ Use value]

Position:
Mode: relative
Top: <Not Set> px
Bottom: <Not Set> px
Right: <Not Set> px
Left: <Not Set> px

Size:
Width: 480 px
Height: 452 px
Visible: <Not Set>
Z Index: <Not Set>

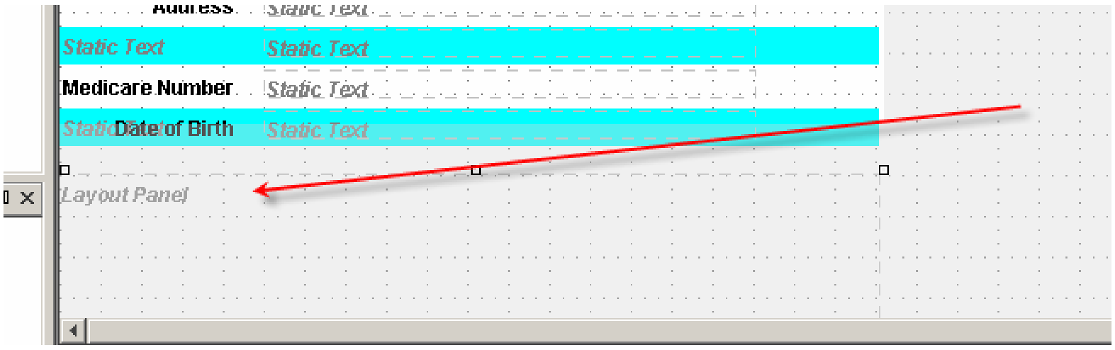
CSS Style: font-size: 12px; height: 128px; position: relative; width: 480px

lpMapBasic - Properties

id	lpMapBasic
panelLayout	Grid Layout
style	font-size: 12px; height: 128px; position...
styleClass	
visible	✓
rendered	✓

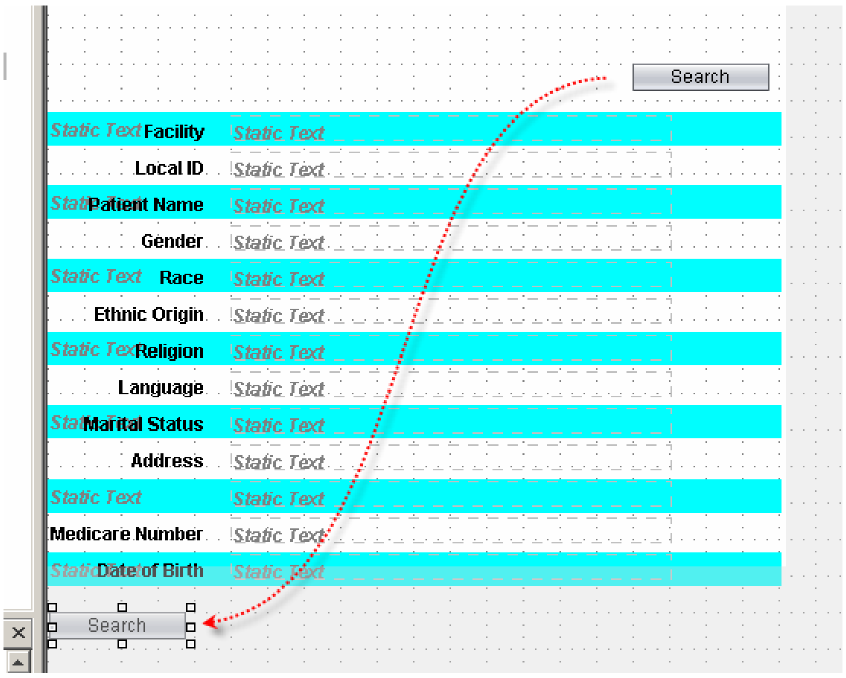
style
CSS Style(s) (String) CSS style(s) to be applied to the outermost HTML element when this component is rendered.

Note that the new panel now appears below both existing panels in the Deign mode.



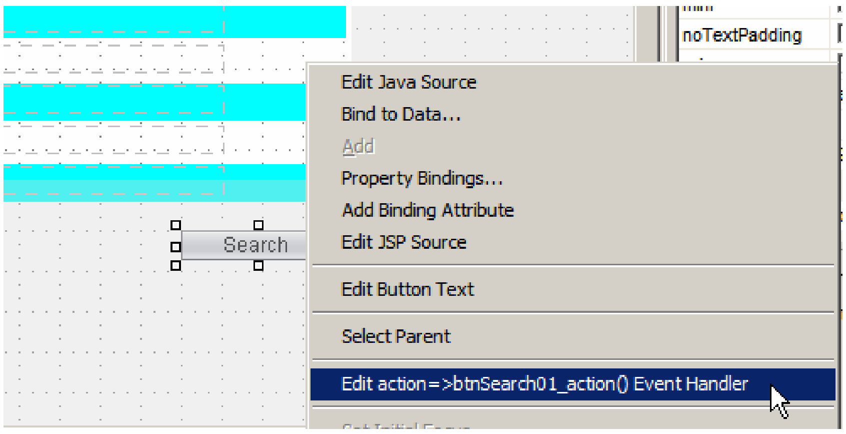
Right-click the panel and choose “Add Binding Attribute” so that we can manipulate visible property of the panel in the Java class.

Let’s copy the search button from the lpView panel and paste it into the lpMapBasic panel.



Let’s change properties of this new button to id: btnSearch02, style property attributes: left: 383px, top: 0px, position: absolute, width: 90px.

Right-click on the button and notice that the Edit Action option has the “btnSearch01_action() Event Handler” specified. We will leave it as is since all of the search buttons should do the same thing – return the user to the Lookup panel.

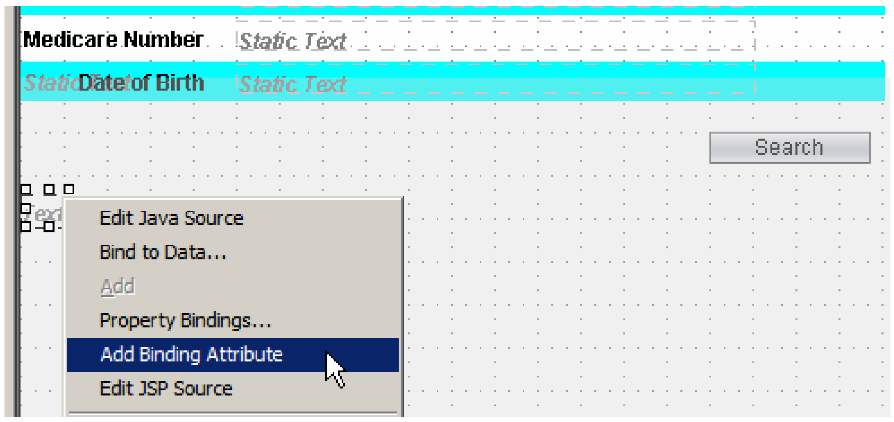


In the not too distant future we will invoke a Google Map REST service, pass to it an address of the patient, and get back a HTML fragment which includes a bunch of Java Script scripts. We will need to inject this HTML fragment into the existing HTML so that it gets rendered by the browser and so that the browser executes the Java Script scripts embedded inside it.

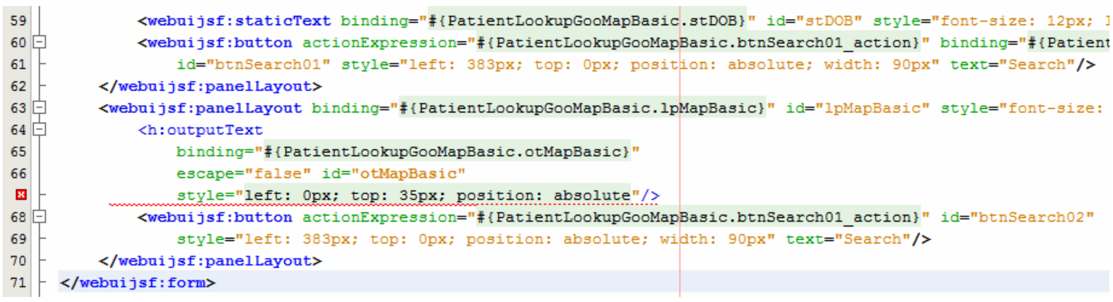
To inject the HTML we need a non-Woodstock outputText container. Switch to the JSP mode, scroll down to the webuijsf:layoutPanel element with id: lpMapBasic and insert the following code on a new line.

```
<h:outputText escape="false" id="otMapBasic" style="left: 0px; top:
35px; position: absolute"/>
```

This creates a SPAN element with id of otMapBasic, places somewhat lower then the top of the panel. Switch to the Design mode, right-clik on the new outputText element and choose “Add Binding Attribute”. We will need to access this component in the Java code later.



The result should look something like this (where I re-formatted the text somewhat).

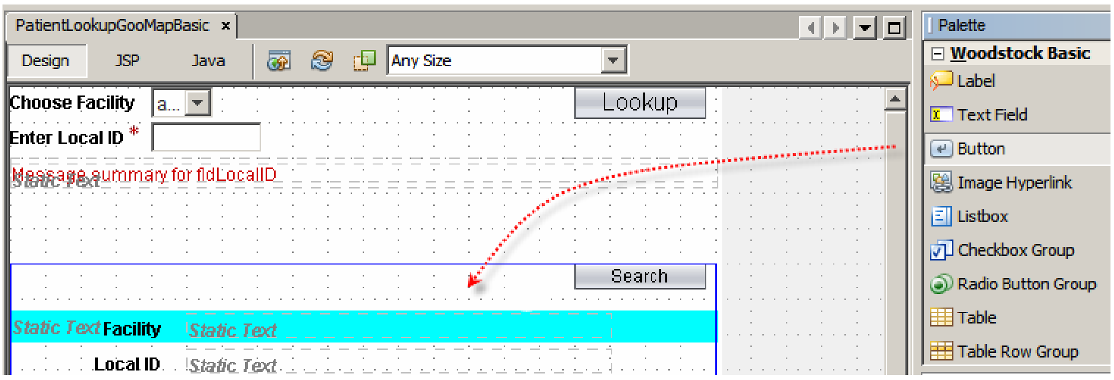


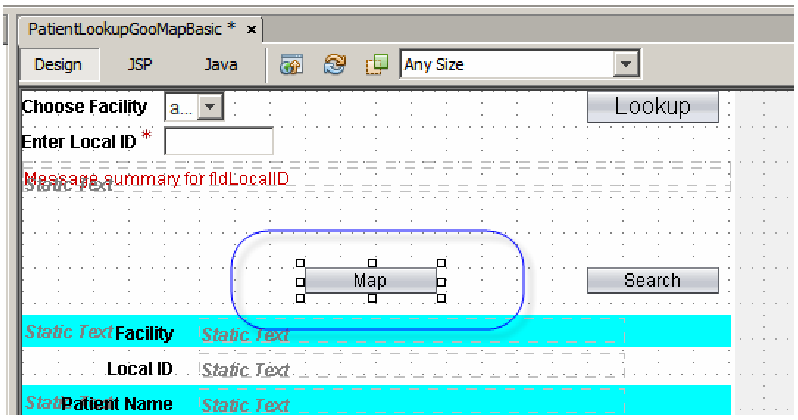
```
59              <webuijsf:staticText binding="#{PatientLookupGooMapBasic.stDOB}" id="stDOB" style="font-size: 12px;
60              <webuijsf:button actionExpression="#{PatientLookupGooMapBasic.btnSearch01_action}" binding="#{Patient
61                  id="btnSearch01" style="left: 383px; top: 0px; position: absolute; width: 90px" text="Search"/>
62          </webuijsf:panelLayout>
63          <webuijsf:panelLayout binding="#{PatientLookupGooMapBasic.lpMapBasic}" id="lpMapBasic" style="font-size:
64              <h:outputText
65                  binding="#{PatientLookupGooMapBasic.otMapBasic}"
66                  escape="false" id="otMapBasic"
                    style="left: 0px; top: 35px; position: absolute"/>
68              <webuijsf:button actionExpression="#{PatientLookupGooMapBasic.btnSearch01_action}" id="btnSearch02"
69                  style="left: 383px; top: 0px; position: absolute; width: 90px" text="Search"/>
70          </webuijsf:panelLayout>
71      </webuijsf:form>
```

Ignore the warning shown in my picture on line 67.

So far we have just one button on each of the lpView and lpMapBasic panels. We would like to be able to switch between the Details view and the Map view. To do this we need to add two buttons, one on the lpView panel which will switch to the lpMapBasic panel and one on the lpMapBasic panel which will switch to the lpView panel. Let’s call these buttons btnMap01 (for the lpView panel) and btnView02 (for the lpMapBasic panel).

Drag a Woodstock Button component onto the lpView panel. Change its properties as follows: id: btnMap01, text: Map, style attributes: left: 191px, width: 90px, top: 0px.

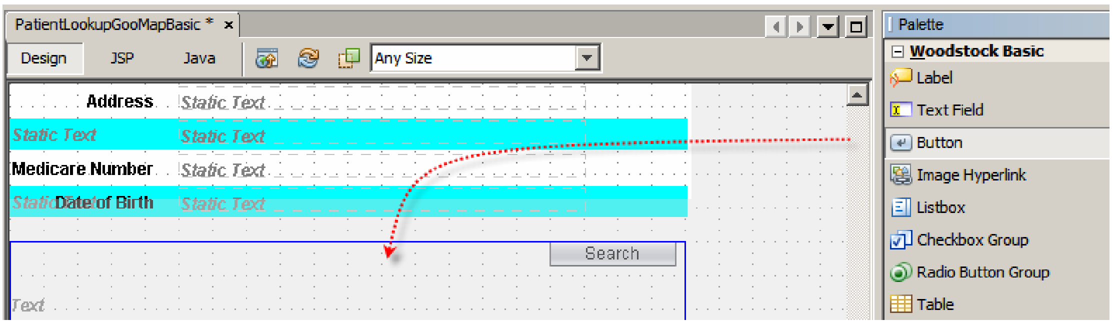


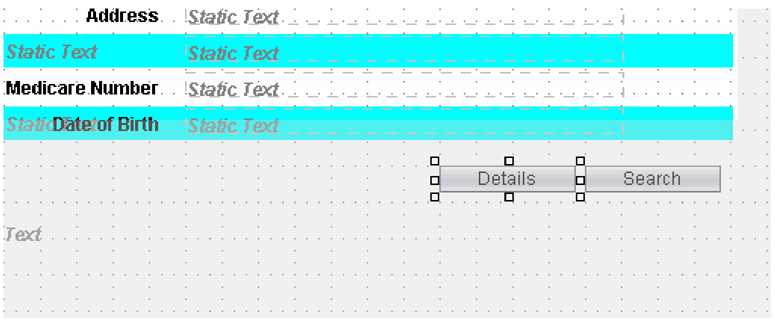


Right-click on the btnMap01 button and choose “Add Binding Attribute”. We will need to show and hide this button in Java depending on whether a mappable address is available.

Right-click the btnMap01 button and choose “Edit action Event Handler”. This will switch the view to Java mode and create boilerplate code for the action handler. We will attend to this later. Switch back to the Deign view.

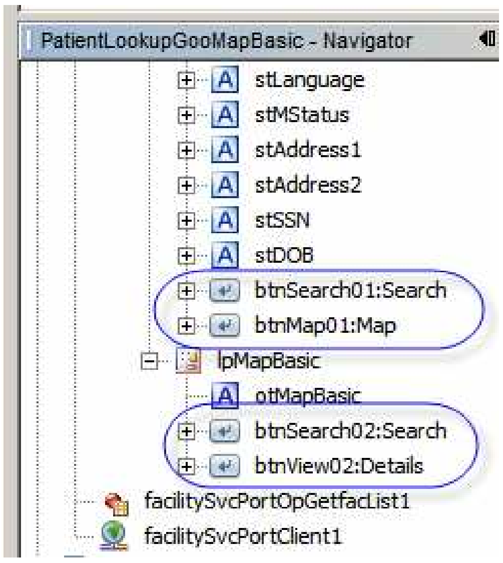
Drag a Woodstock Button component onto the lpMapBasic panel and set its properties as follows: id: btnView02, text: Details, style attributes: top: 0px, left: 287px, width: 90px.



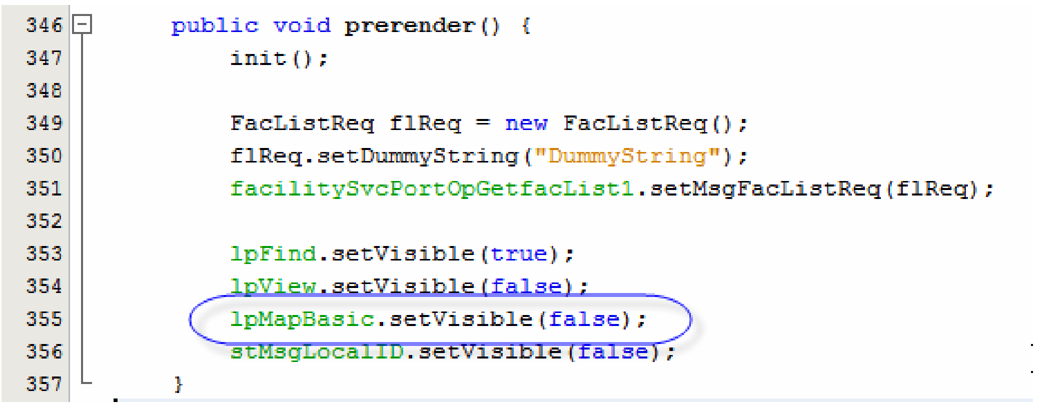


Right-click on the btnView02 button and choose “Edit action Event Handler”. This will switch the view to Java mode and create boilerplate code for the action handler. We will attend to this later. Switch back to the Deign view.

Switch to the Design mode and inspect the hierarchy in the Navigator panel to make sure all components are ordered and nested correctly.

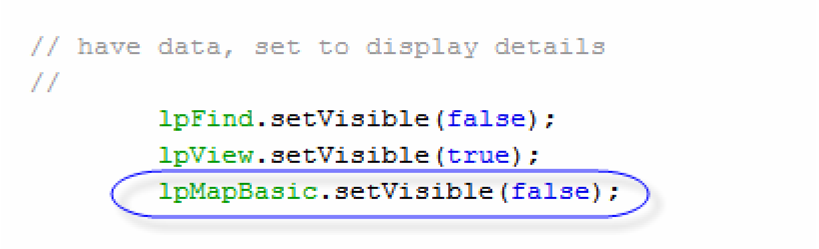


Now we are ready to add the Java code to get the Google Map and manipulate the components we added. Switch to Java mode and scroll to the prerender() method. Ass a statement that sets the visible property of the lpMapBasic panel to false, hiding it.



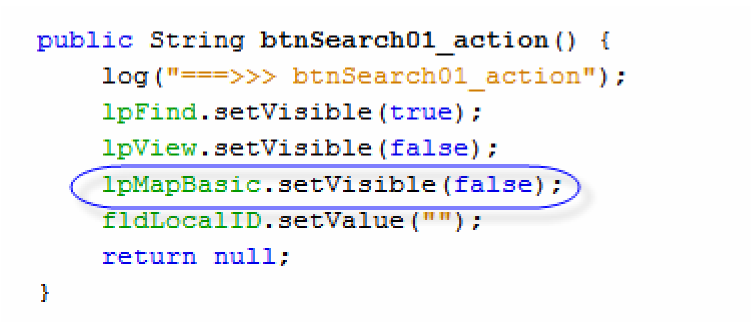
```
    public void prerender() {
        init();

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

        lpFind.setVisible(true);
        lpView.setVisible(false);
        lpMapBasic.setVisible(false);
        stMsgLocalID.setVisible(false);
    }
```

Scroll to the end of the btnLookup_action() method and set the visibility of the panel lpMapBasic to false. When the lookup button is clicked we will either get redirected to the View panel or will remain on the Lookup panel.



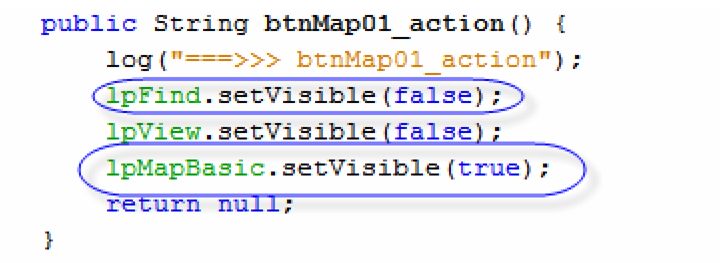
```
// have data, set to display details
//
            lpFind.setVisible(false);
            lpView.setVisible(true);
            lpMapBasic.setVisible(false);
```

Scroll down to the btnSearch01_action() method and add the change to visibility of the lpMpaBasic there as well.



```
public String btnSearch01_action() {
     log("===>>> btnSearch01_action");
     lpFind.setVisible(true);
     lpView.setVisible(false);
     lpMapBasic.setVisible(false);
     fldLocalID.setValue("");
     return null;
}
```

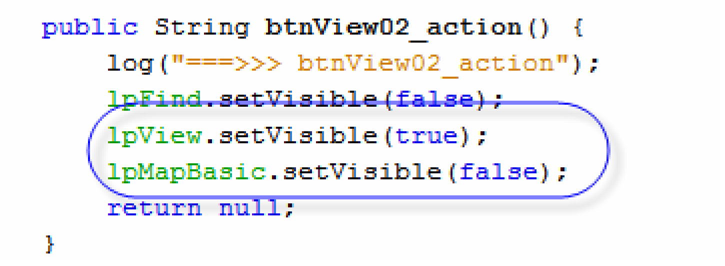
Copy the 4 lines of code shown in the picture above and paste them into the body of the btnMap01_action() method, replacing the two lines of comment.



```
public String btnMap01_action() {
     log("===>>> btnMap01_action");
     lpFind.setVisible(false);
     lpView.setVisible(false);
     lpMapBasic.setVisible(true);
     return null;
}
```

Change the visible setting of the lpFind panel to false and lpMapBasic to true – the Map button was clicked so the Map panel must be displayed and the Lookup panel must be hidden.

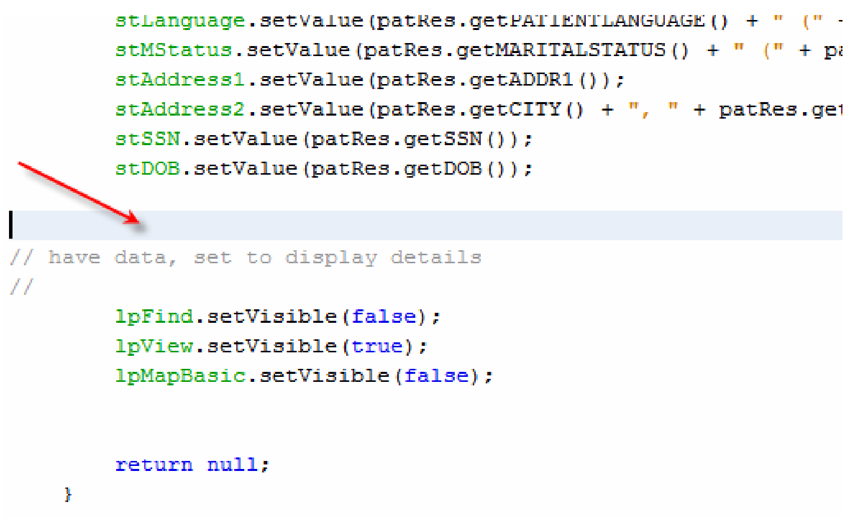
Copy the 4 lines of code shown in the picture above and paste them into the body of the btnView02_action() method, replacing the two lines of code.



```
public String btnView02_action() {
     log("===>>> btnView02_action");
     lpFind.setVisible(false);
     lpView.setVisible(true);
     lpMapBasic.setVisible(false);
     return null;
}
```

Change the visible setting of the lpView panel to true and lpMapBasic panel to false. Button with the caption Details was clicked. We need to show the view panel and hide the Map panel.

Scroll back to the bottom of the btnLookup_action() method. We will be adding more code between the end of the details setting statements and the beginning of the visibility setting statements.



```
        stLanguage.setValue(patRes.getPATIENTLANGUAGE() + " (" -
        stMStatus.setValue(patRes.getMARITALSTATUS() + " (" + pa
        stAddress1.setValue(patRes.getADDR1());
        stAddress2.setValue(patRes.getCITY() + ", " + patRes.get
        stSSN.setValue(patRes.getSSN());
        stDOB.setValue(patRes.getDOB());


// have data, set to display details
//
        lpFind.setVisible(false);
        lpView.setVisible(true);
        lpMapBasic.setVisible(false);


        return null;
    }
```

The Google Map REST Service, which we will be invoking shortly, requires a reasonable address. A reasonable address is one which has a street name and a city. If the address has a street number, a state/province and a postal code - so much the better. Some example valid addresses are:

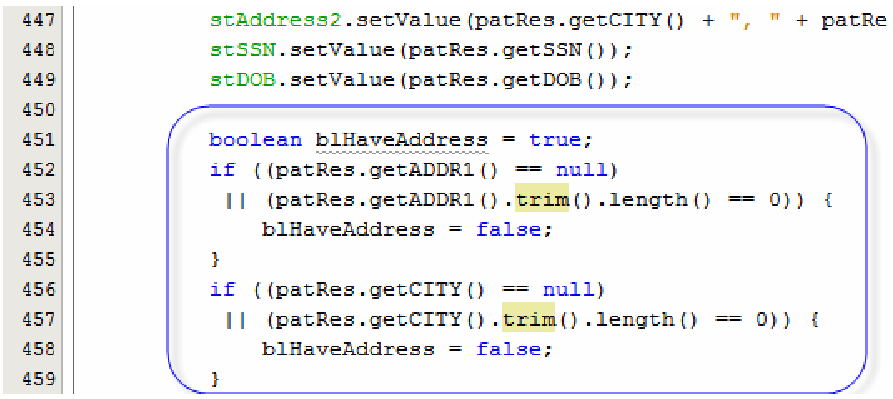
foro romano, rome
edwin street, croydon
edwin street, croydon, uk

In the example above "edwin street, croydon" would be for the Edwin Street, Croydon, NSW, AU because I live in Sydney and the Google Maps works out from what it thinks my systems location is, which country and geographical region to place the partial address like this. As soon as I add the country code, UK, Croydon gets to be in Greater London. Some addresses, which might otherwise appear valid, may not be mappable because they do not exist.

This leads us to the following:

1. If the address has a street name and a city / suburb then it is superficially valid and can be used for mapping
2. If the address is mappable then invoke the Google Map service and show the btnMap02 button
3. If either street name or city / suburb is missing the address is invalid and can not be mapped
4. If address is not mappable do not invoke the Google Map service and hide the btnMap02 button/

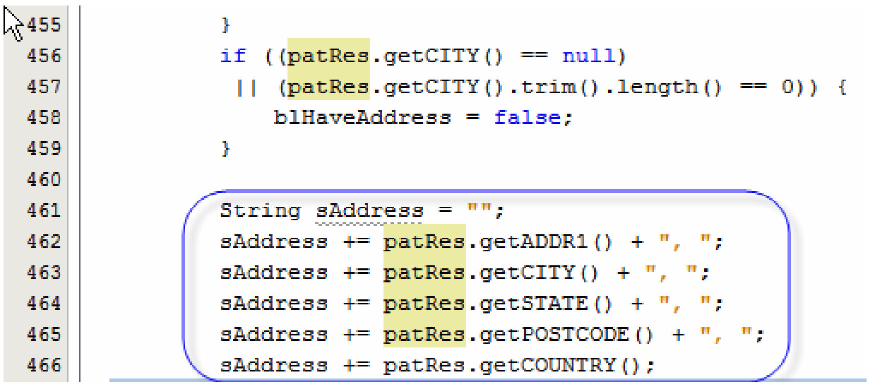
Let's now add the slab of Java code which works out whether the address is mappable according to the rules set out above.



```
        stAddress2.setValue(patRes.getCITY() + ", " + patRe
        stSSN.setValue(patRes.getSSN());
        stDOB.setValue(patRes.getDOB());

        boolean blHaveAddress = true;
        if ((patRes.getADDR1() == null)
         || (patRes.getADDR1().trim().length() == 0)) {
            blHaveAddress = false;
        }
        if ((patRes.getCITY() == null)
         || (patRes.getCITY().trim().length() == 0)) {
            blHaveAddress = false;
        }
```

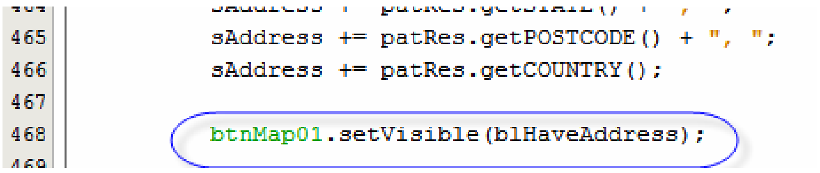
If the address is mappable let’s assemble a single address string which to pass to the Google Map service. It will be of the form “street address, city, state, post code, country”, where “foro romano, rome,,,” is valid so we will not be fussy about multiple commas.

Add the following code to assemble the address string.



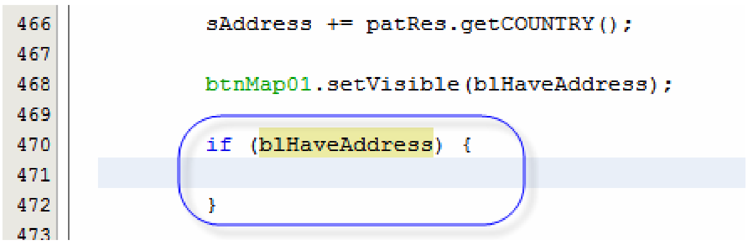
```
        }
        if ((patRes.getCITY() == null)
         || (patRes.getCITY().trim().length() == 0)) {
            blHaveAddress = false;
        }

        String sAddress = "";
        sAddress += patRes.getADDR1() + ", ";
        sAddress += patRes.getCITY() + ", ";
        sAddress += patRes.getSTATE() + ", ";
        sAddress += patRes.getPOSTCODE() + ", ";
        sAddress += patRes.getCOUNTRY();
```

The Map button should only be visible if there is a mpappable address. Set the visibility property of that button to the value of the blHave Address Boolean.



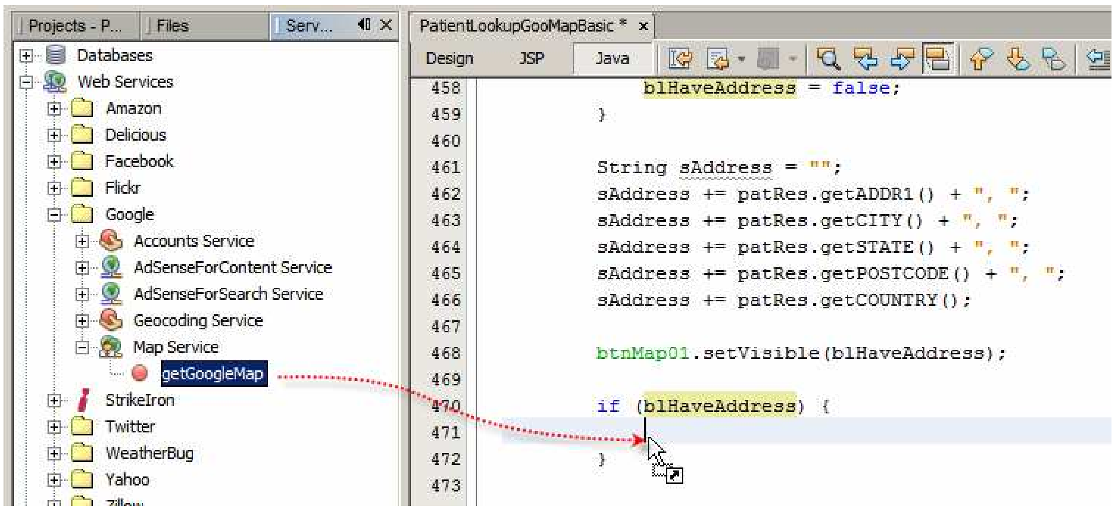
```
        sAddress += patRes.getPOSTCODE() + ", ";
        sAddress += patRes.getCOUNTRY();

        btnMap01.setVisible(blHaveAddress);
```

Create a conditional expression, based on the blHaveAddress Boolean, with which to surround the Google Map REST service invocation.



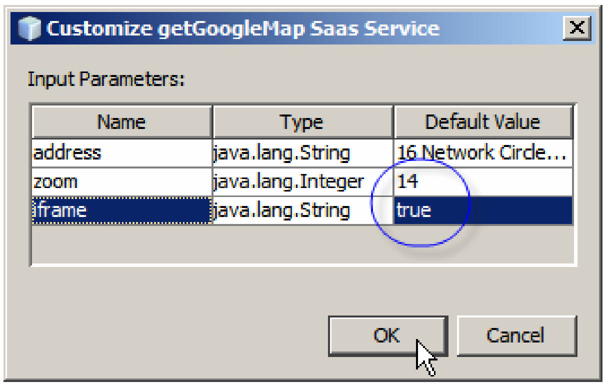
```
        sAddress += patRes.getCOUNTRY();

        btnMap01.setVisible(blHaveAddress);

        if (blHaveAddress) {

        }
```

In the left panel switch from the Project Explorer view to the Services view, expand web services through Google -> Map Service and drag the getGoogleMap operation onto the canvas inside the conditional expression just created.



```
            blHaveAddress = false;
        }

        String sAddress = "";
        sAddress += patRes.getADDR1() + ", ";
        sAddress += patRes.getCITY() + ", ";
        sAddress += patRes.getSTATE() + ", ";
        sAddress += patRes.getPOSTCODE() + ", ";
        sAddress += patRes.getCOUNTRY();

        btnMap01.setVisible(blHaveAddress);

        if (blHaveAddress) {

        }
```

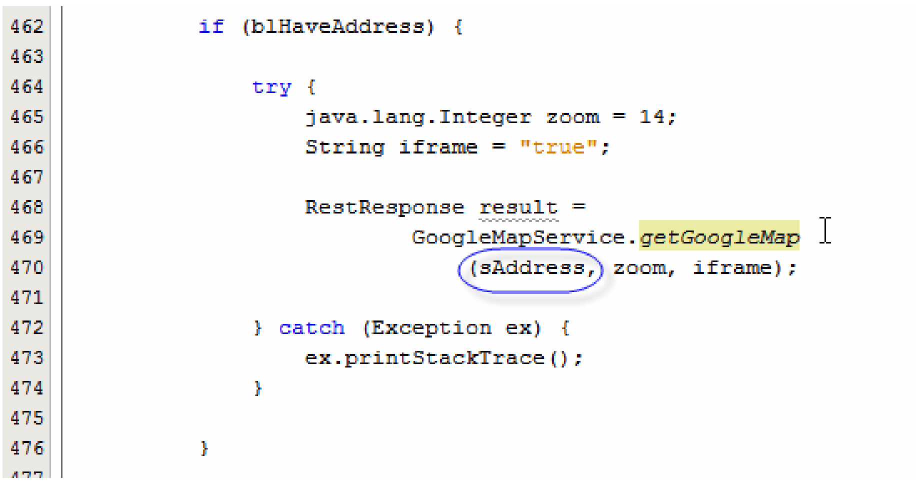
Customize the service invocation in the dialogue box by setting zoom to 14 and iframe to true.



Input Parameters:

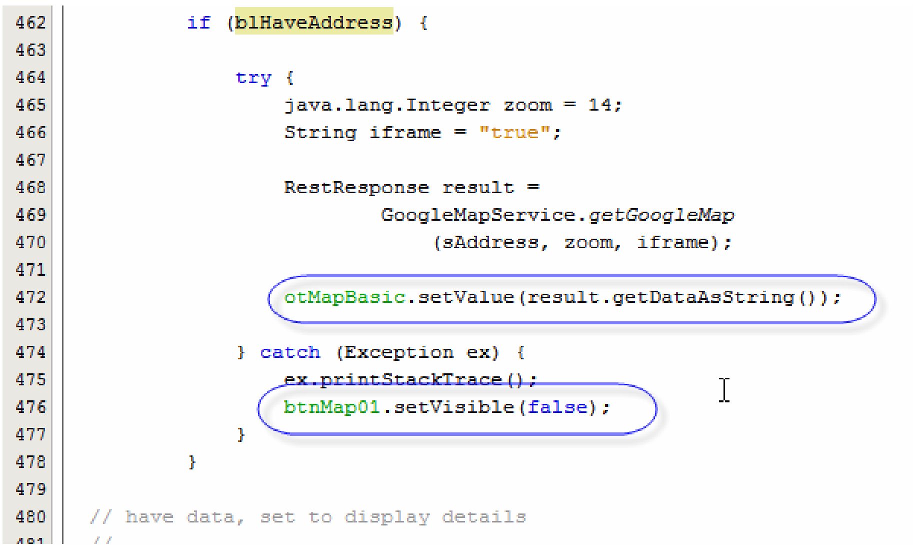
Name	Type	Default Value
address	java.lang.String	16 Network Circle...
zoom	java.lang.Integer	14
iframe	java.lang.String	true

Modify the generated code so that it uses the sAddress string, which we assembled before, as the address the service will use.



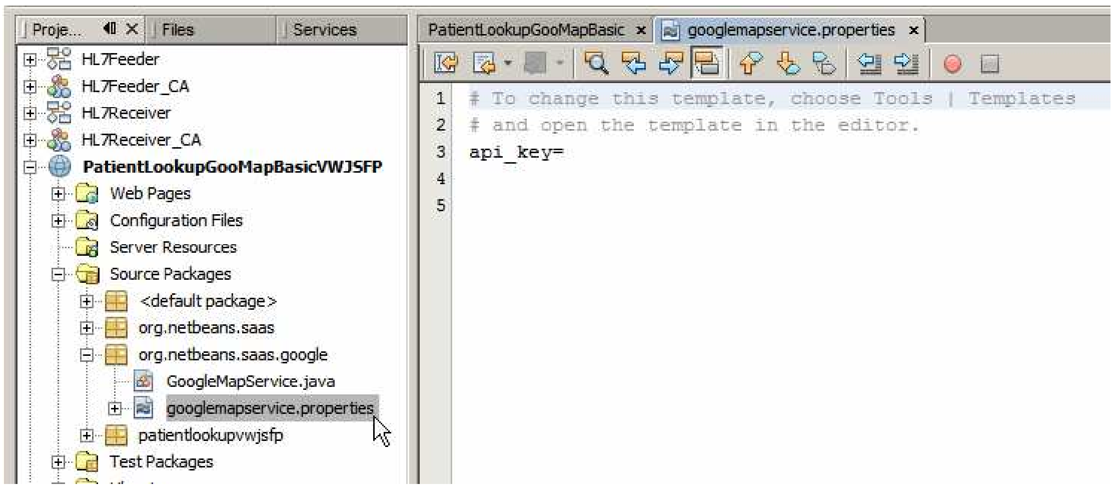
```
        if (blHaveAddress) {

            try {
                java.lang.Integer zoom = 14;
                String iframe = "true";

                RestResponse result =
                        GoogleMapService.getGoogleMap
                           (sAddress, zoom, iframe);

            } catch (Exception ex) {
                ex.printStackTrace();
            }

        }
```

Assign the result to the outputText component on success and set the btnMap01 visibility to false on exception.



```
            if (blHaveAddress) {

                try {
                    java.lang.Integer zoom = 14;
                    String iframe = "true";

                    RestResponse result =
                            GoogleMapService.getGoogleMap
                                (sAddress, zoom, iframe);

                    otMapBasic.setValue(result.getDataAsString());

                } catch (Exception ex) {
                    ex.printStackTrace();
                    btnMap01.setVisible(false);
                }
            }

        // have data, set to display details
```

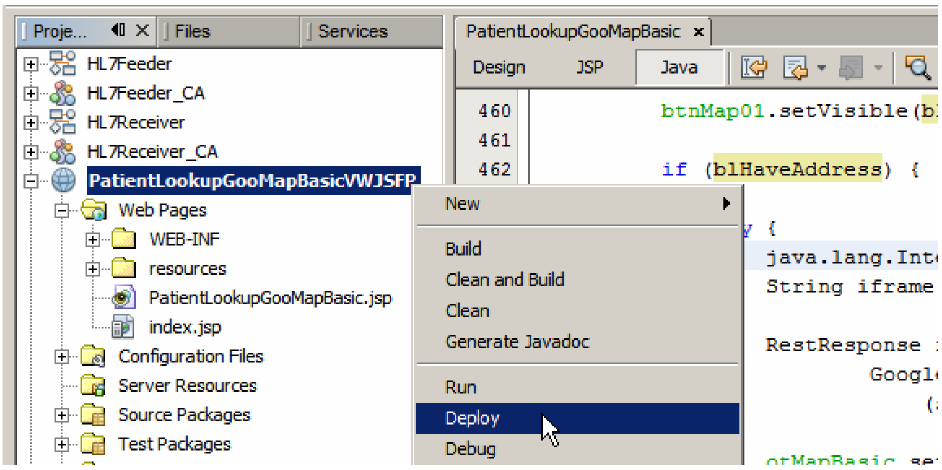
Switch back to the Project Explorer view in the left hand pane, expand the Source Packages folder through to org.netbeans.google.saas, open googlemapservice.porperties and fill in the value of the API Key.



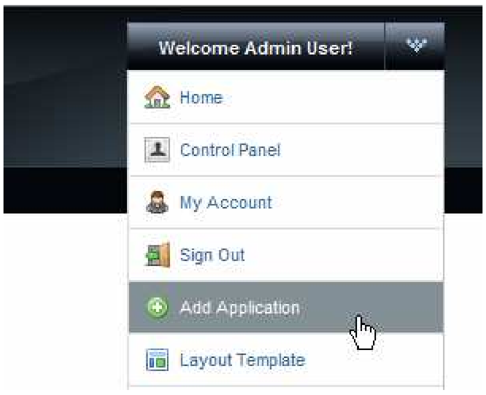
```
# To change this template, choose Tools | Templates
# and open the template in the editor.
api_key=
```

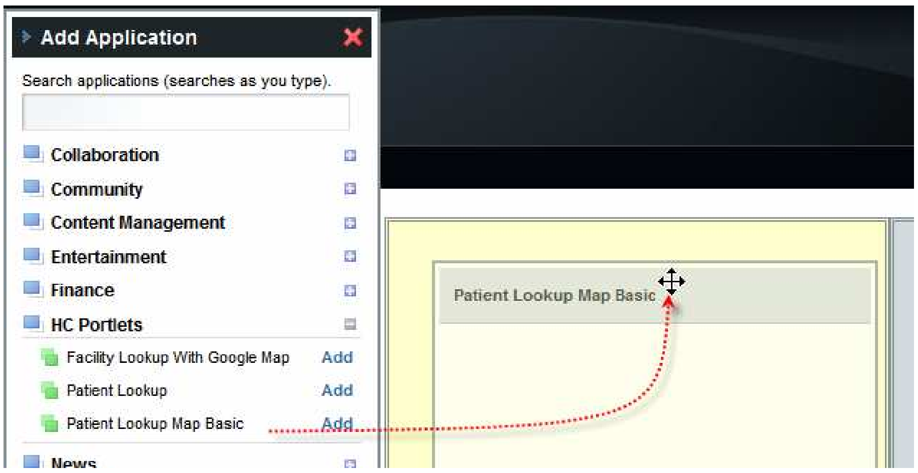
If you don't have the Google Map Service AP Key then get it from Google - see [10], pages 47-50 for details.

Right-click on the project name and choose Deploy.



Now that the portlet is deployed we need to add it to the portal page. If the early version of the portlet is on the portal page it needs to be removed.



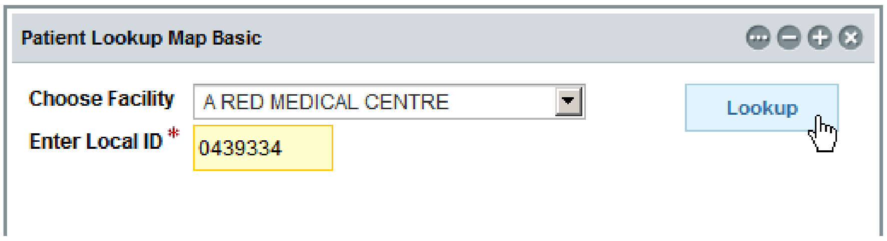


When creating the Patient Service web service we created and populated a database table with seed data. Some of the facility/local id combinations are:

facility	local_id	last_name	first_name	middle_initial	addr1	city	state	post_code	country
ARMC	0101018	WHITEHURST	JULIA	(null)	1/97 FARRINGDON VILLAGE	WERRINGTON DOWNS	NSW	2747	AU
ARMC	0439334	POHL	ANNE-MARIE	(null)	164 Edwin Street North	Croydon	NSW	2132	AU
ARMC	0461040	WESTWOOD	JOKA	(null)	1/97 FARRINGDON VILLAGE	PENRITH	NSW	2750	AU
ARMC	0468815	HARIHARAN	JOSE EVASCO	(null)	100 SMITH ST	ST CLAIR	NSW	2753	AU
ARMC	0498727	DEGRENIS	DAGWOOD	(null)	1/97 FARRINGDON VILLAGE	NORTH SYDNEY	NSW	2759	AU
ARMC	0532821	SANDERS	GRACE	(null)	(null)	WERRINGTON	NSW	2750	AU
ARMC	0533086	KRYSIAK	BABY OF DENISE	(null)	100 STRANGE ST	JAMISONTOWN	NSW	2782	AU
ARMC	0533805	ZAHRA	LEANNE GAI	(null)	100 FRUITBOWEL CLOSE	WERRINGTON		2745	
ARMC	0534005	STYZINSKI	JOSEPHINE	(null)	1/97 FARRINGDON VILLAGE	Dianella	NSW	2747	AU

ARMC	0534006	DUNSTAN	VIOLET MADGE	(null)	(null)	LONDONDERRY	NSW	2148	AU
STC	100000	CAESAR	JULIUS	(null)	Foro Romano	ROME	(null)	(null)	it

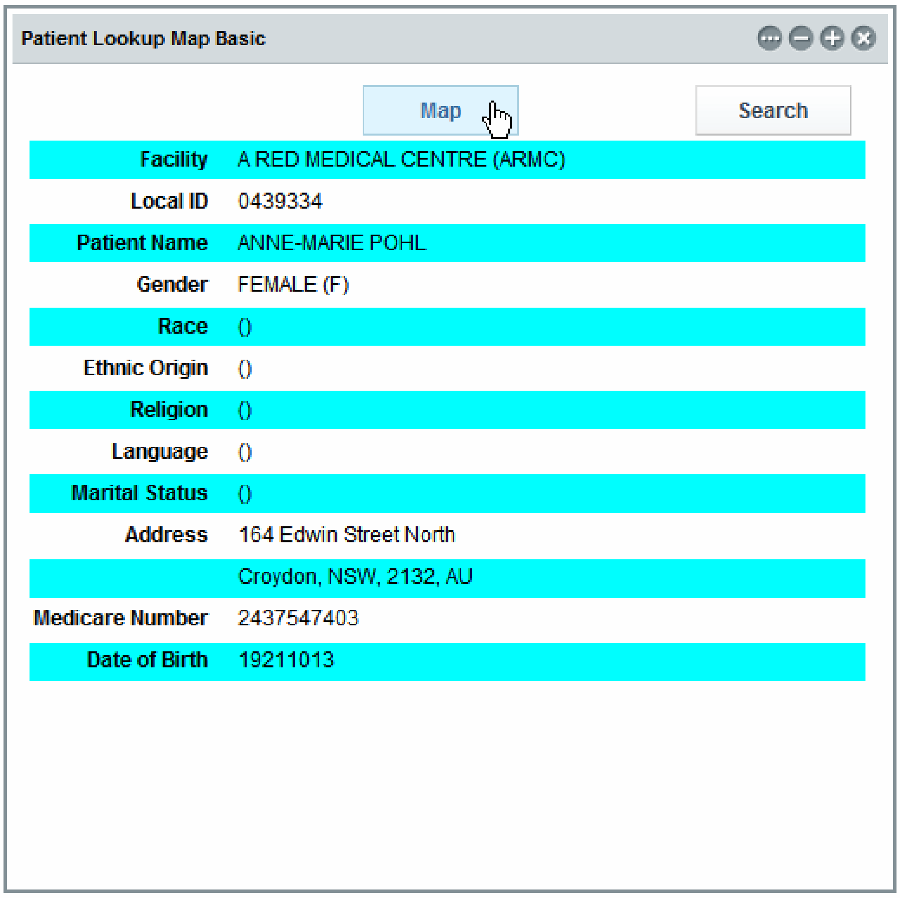
Choose A RED MEDICAL CENTRE from the list of facilities and enter 0439334 as Local ID. Click the Lookup button.


Patient Lookup Map Basic

Choose Facility: A RED MEDICAL CENTRE

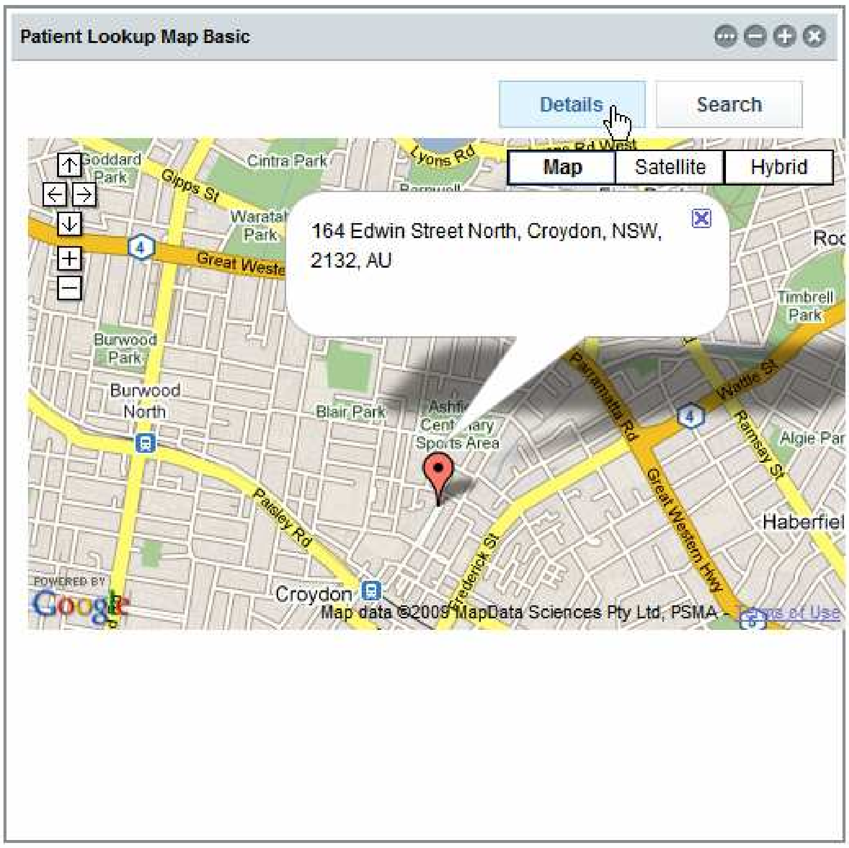
Enter Local ID *: 0439334

Click the Map button.

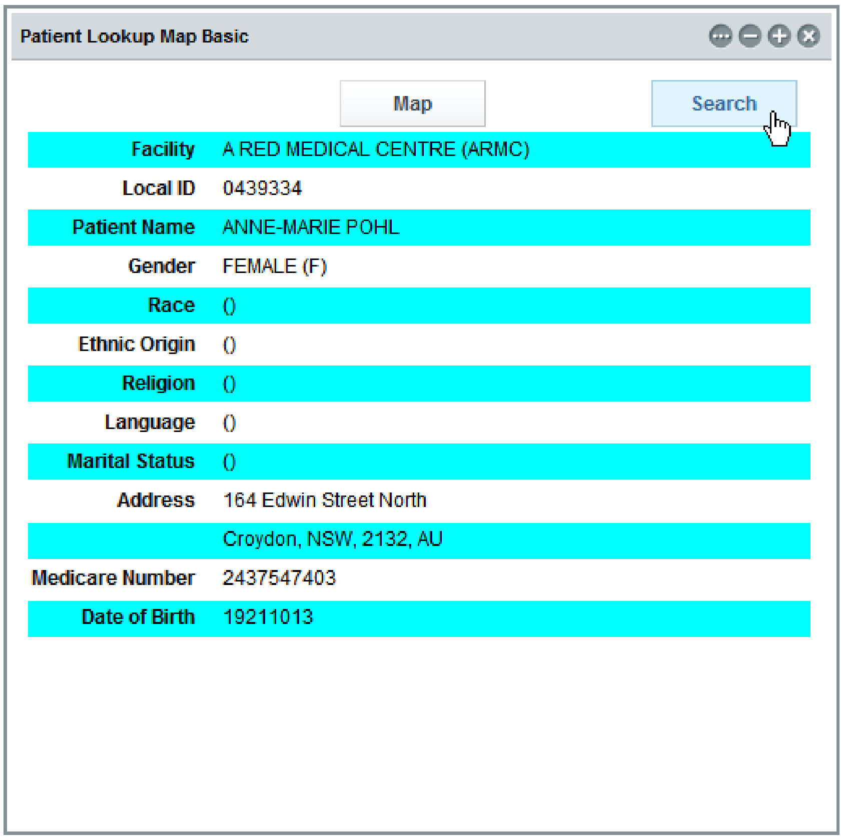

Patient Lookup Map Basic

Facility	A RED MEDICAL CENTRE (ARMC)
Local ID	0439334
Patient Name	ANNE-MARIE POHL
Gender	FEMALE (F)
Race	()
Ethnic Origin	()
Religion	()
Language	()
Marital Status	()
Address	164 Edwin Street North
	Croydon, NSW, 2132, AU
Medicare Number	2437547403
Date of Birth	19211013

Click the Details button.

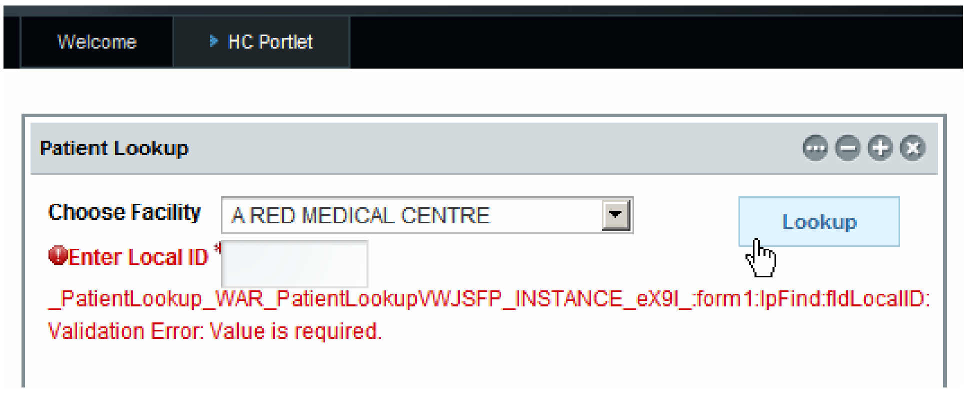

Patient Lookup Map Basic
Details
Search
Map
Satellite
Hybrid
164 Edwin Street North, Croydon, NSW, 2132, AU
Map data ©2009 MapData Sciences Pty Ltd, PSMA - Terms of Use

Click the Search button.



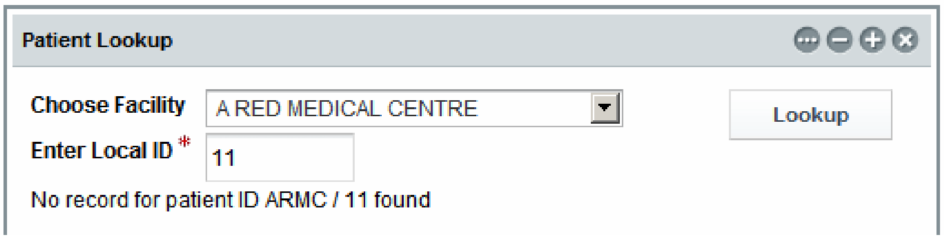
Facility	A RED MEDICAL CENTRE (ARMC)
Local ID	0439334
Patient Name	ANNE-MARIE POHL
Gender	FEMALE (F)
Race	()
Ethnic Origin	()
Religion	()
Language	()
Marital Status	()
Address	164 Edwin Street North
	Croydon, NSW, 2132, AU
Medicare Number	2437547403
Date of Birth	19211013

Click the Search button, noting that the Local ID field is cleared. Without entering anything into the Local ID field click the Lookup button. Notice the error message.



_PatientLookup_WAR_PatientLookupVWJSFP_INSTANCE_eX9I_:form1:lpFind:fldLocalID: Validation Error: Value is required.

Enter a local id which is not in the database, for example 11. Click the Lookup button and notice the error message.



Choose Facility: A RED MEDICAL CENTRE

Enter Local ID *: 11

No record for patient ID ARMC / 11 found

This error message was explicitly set in the btnLookup_action() method when the web service invocation returned with no record.

We are done. This is what it took to develop a portlet, which used the Facilities Service web service and the Patient Service web service as a data providers, and deploy it to the Web Space Server 10 Portal.

Summary

In this document we walked through the process of developing a JSR-286-compliant Visual Web JSF Portlet, deployed to the Sun Web Space Server 10 Portal, which used the Facility Service and the Patient Service Web Service as a data providers. We used the NetBeans 6.5.1 IDE, which comes as part of the GlassFish ESB v2.1 installation, the Portal Pack 3.0.1 NetBeans Plugin and the JSF Portal Bridge infrastructure provided by the Web Space Server 10. The Portlet was implemented as a Visual Web JavaServer Faces Portlet using JSF components provided by Project Woodstock.

Because the portlet used the web service data providers the interfaces between it and the data stores was defined in the appropriate WSDL definitions. The enterprise, where this portlet is used, can change the implementation of both web services and, as long as the interface does not change, the protlet will not need to change. This is how loose coupling is achieved in Service Oriented Architectures. The portlet is a component in the SOA Layer 1, Presentation Layer.

References

[1] MySQL Community Server and GUI Tools - Getting, Installing and Configuring, at http://blogs.sun.com/javacapsfieldtech/entry/mysql_community_server_and_gui.

[2] GlassFish ESB v2.1 download and installation, https://open-esb.dev.java.net/Downloads.html

[3] Adding Sun WebSpace Server 10 Portal Server functionality to the GlassFish ESB v2.1 Installation, http://blogs.sun.com/javacapsfieldtech/entry/adding_sun_webspace_server_10

[4] Making Web Space Server And Web Services Play Nicely In A Single Instance Of The Glassfish Application Server, http://blogs.sun.com/javacapsfieldtech/entry/making_web_space_server_and.

[5] GlassFish ESB v 2.1 - Creating a Healthcare Facility Web Service Provider, http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v_2_1

[6] GlassFish ESB v2.1, MySQL v5.1 - Creating a Patient Service Web Service Provider, http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v2_1_mysql1

[7] GlassFish ESB v2.1, MySQL v5.1 - Make HL7 v2.3.1 Delimited Messages from Custom Delimited Records with HL7 Encoder and HL7 BC, http://blogs.sun.com/javacapsfieldtech/entry/glassfish_esb_v2_1_mysql

[8] Healthcare Facility Mashup Portlet with Google Map - GlassFish v 2.1, Web Space 10, Web Service and REST Service, http://blogs.sun.com/javacapsfieldtech/entry/healthcare_facility_mashup_portlet_with

[9] GlassFish ESB v2.1, Web Space Server 10 - Creating a Patient Lookup Visual Web JSF Portlet, http://blogs.sun.com/javacapsfieldtech/entry/creating_a_patient_lookup_visual

[10] Healthcare Facility Mashup Portlet with Google Map - GlassFish v 2.1, Web Space 10, Web Service and REST Service, http://blogs.sun.com/javacapsfieldtech/entry/healthcare_facility_mashup_portlet_with