# GlassFish ESB v 2.1 Creating a Healthcare Facility Web Service Provider

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.

In this document I will walk through the process of developing a SOA Composite Application, exposed as a Web Service, which will make available simple business functionality using a multi-operation service. We will use the GalssFish ESB v2.1 infrastructure. The service will use the SOAP/HTTP Binding Component, the Database Binding Component and the BPEL 2.0 Service Engine. This simple service will introduce the components and discuss how a multi-operation web service can be constructed using the GalssFish ESB.

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 will be made available as a multi-operation web service. This web service will be used elsewhere to construct a portlet that can be used in an enterprise portal.

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, is available for development and deployment of the web service discussed in this paper. 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 <a href="http://blogs.sun.com/javacapsfieldtech/entry/adding\_sun\_webspace\_server\_10">http://blogs.sun.com/javacapsfieldtech/entry/adding\_sun\_webspace\_server\_10</a>, 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 <a href="http://blogs.sun.com/javacapsfieldtech/entry/making\_web\_space\_server\_a\_nd">http://blogs.sun.com/javacapsfieldtech/entry/adding\_sun\_webspace\_server\_10</a>, 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 <a href="http://blogs.sun.com/javacapsfieldtech/entry/making\_web\_space\_server\_a\_nd">http://blogs.sun.com/javacapsfieldtech/entry/making\_web\_space\_server\_a\_nd</a>.

### **Create MySQL objects**

Our facility details will reside in a table in a relational database. Before the service can get access to that data we need to:

- 1. create the database
- 2. create the database table

- populate the database table
- 4. create a NetBeans connection to the database and exercise it
- 5. create a runtime connection pool at the GalssFish App Server
- 6. create JNDI Reference to the connection pool

If you don't have the MySQL database installed you can obtain and install it following instructions in "MySQL Community Server and GUI Tools - Getting, Installing and Configuring", at <u>http://blogs.sun.com/javacapsfieldtech/entry/mysql\_community\_server\_and\_gui</u>.

The following text walks through the process of preparing data and connection pools for this writeup, a step at a time.

Start mysql command line client as the root user and execute the following commands:

```
drop user pblog;
drop database pblog;
create database pblog default character set utf8;
use pblog;
create user pblog;
grant all privileges on *.* to 'pblog'@'localhost' identified by 'pblog';
set password for 'pblog'@'localhost' = password('pblog');
```

This set of commands drops the pblog database and user, if they exist, and creates a new database and user for this document. Dropping the database destroys all the objects it may contain. One assumes you don't have a database pblog used for a different purpose.

Execute the following commands to create the ui\_facility table:

Execute the following commands to add sample data to the table:

```
-- Disable foreign key checks
SET @OLD FOREIGN KEY CHECKS=@@FOREIGN KEY CHECKS, FOREIGN KEY CHECKS=0;
INSERT INTO `pblog`.`ui_facility`
    (`facility_code`
   description
  `status
  `addr1
   `addr2
  `city`
  `state`
   post_code
  `country`)
VALUES
    ('STC', 'SYDNEY TECHNICAL HOSPITAL', 'A', '404 HUNTINGTON DRIVE', NULL,
'MONROVIA', 'CA', '91016', 'USA'),
('D210', 'Nepean Hospital', 'A', NULL, NULL, NULL, NULL, NULL, NULL),
  ('D230', 'Tresillian', 'A', NULL, NULL, NULL, NULL, NULL, NULL),
  ('D204', 'Blue Mountains Hospital', 'A', NULL, NULL, NULL, NULL, NULL, NULL, NULL), ('D754', 'Governor Phillip', 'A', NULL, NULL, NULL, NULL, NULL, NULL),
```

```
('D214', 'Springwood Hospital', 'A', NULL, NULL, NULL, NULL, NULL, NULL, NULL),
('ICPMR', 'ICPMR', 'A', NULL, NULL, NULL, NULL, NULL, NULL, NULL),
('BIGH', 'A BIG HOSPITAL', 'A', NULL, NULL, NULL, 'NS', '2000', NULL),
('ARMC', 'A RED MEDICAL CENTRE', 'A', 'SOMEWHERE ST', NULL, 'SOMEWHERE CITY',
'NSW', '2060', 'AU'),
('TBIGH', 'THE BIG HOSPITAL', 'A', 'WHOKNOWES WHERE ST', NULL, 'ANYWHERE
CITY', 'NSW', '2132', 'AU'),
('NSMC', 'NORTH SYDNEY MEDICAL CENTRE', 'A', '100 WALKER ST', NULL, 'NORTH
SYDNEY', 'NW', '2000', NULL);
-- Re-enable foreign key checks
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
```

Execute the following select statement to make sure the data is there:

```
select * from ui_facility;
```

Now that we have the database objects in the database we need to "introduce" the database to NetBeans. This will make it simpler for us to use the database in the project.

Switch to the Services Tab, right-click Databases and choose "New Connection ...".



Populate fields of the connection wizard and click OK.

🗊 New Database Conne	ection X
Basic setting Advanced	
Data <u>I</u> nput Mode:	Ejeld Entry     O Direct URL Entry
Name:	MySQL (Connector/J driver)
Hos <u>t</u> :	localhost
Port:	3306
<u>D</u> atabase:	pblog
<u>U</u> ser Name:	pblog
Pass <u>w</u> ord:	*****
	Remember password (see help for information on security risks)
Additional Props:	
Show JDBC URL	jdbc:mysql://localhost:3306/pblog
	OK Cancel <u>H</u> elp

If all parameters are correct a new database connection entry will appear.



Right-click the name of the table and choose "View Data ..."

jdbc:mysql://loca	lhost:3306/pblog [pblo	og on Defa	ult schema]
E. Tables			
tiews	View Data		
Web Services     Web Services     Enterprise Beans (2.     Servers	Add Column Refresh <u>D</u> elete	ИS Delete	
	Grab Structure Recreate Table		
	Properties		

This will produce display similar to that shown below.

S. bpi	FacilitySvc_BM.bpel ×	g SQL Command 1 ×			
Connec	tion: jdbc:mysql://loca	alhost:3306/pblog [pblog on Default schema]	-	🛛 🔂 🐺 😰 🎼 -	- Q - 2
B	elect * from u	i_facility			
1;1	INS				
select	* from ui_facility ×				
		🍣 候 < 🗲 🗲 Page Size: 20	Total Rows	: 11 Page: 1 of 1	
#	facility_code	description	status	addr1	
1	ARMC	A RED MEDICAL CENTRE	A	SOMEWHERE ST	<null></null>
2	BIGH	A BIG HOSPITAL	A	<null></null>	<null></null>
3	D204	Blue Mountains Hospital	A	<null></null>	<null></null>
4	D210	Nepean Hospital	A	<null></null>	<null></null>
5	D214	Springwood Hospital	A	<null></null>	<null></null>
6	D230	Tresillian	A	<null></null>	<null></null>
7	D754	Governor Phillip	A	<null></null>	<null></null>
8	ICPMR	ICPMR	A	<null></null>	<null></null>
9	NSMC	NORTH SYDNEY MEDICAL CENTRE	A	100 WALKER ST	<null></null>
10	STC	SYDNEY TECHNICAL HOSPITAL	A	404 HUNTINGTON DRIVE	<null></null>
11	TBIGH	THE BIG HOSPITAL	A	WHOKNOWES WHERE ST	<null></null>

The database connection is available and correctly configured for NetBeans. Now we need to create a connection pool to be used at runtime.

Make sure that the GlassFish Application Server has access to the appropriate MySQL JDBC Driver JAR. The MySQL JDBC driver, mysql-connector-java-5.1.6bin.jar, is distributed as part of the GlassFishESB installation and is located in {GlassFishESBv21\_install\_rtoot}/netbeans/ide10/modules/ext. Copy the driver JAR file to {GlassFishESBv21\_install\_rtoot}/glassfish/domains/domain1/lib/ext and restart the application server before continuing.

Start the GlassFish Admin Console (<u>http://localhost:4848</u> by default – use your own Admin Port if you didn't use defaults at install time) and log in as user admin.



Navigate to Resources -> JDBC -> Connection Pools

Name the new pool "cp\_pblog\_XA", select the javax.sql.XADateSource type and MySQL Database Vendor, then click Next.

Resources > JDBC >	Connection Pools		
New JDBC Co	nnection Pool (Step 1 of 2) tings for the connection pool.	Next	Cancel
General Settings			
Name: *	cp_pblog_XA		
Resource Type:	javax.sql.XADataSource Must be specified if the datasource class implements more than 1 of the interface.		
Database Vendor:	MySQL		

Scroll to Additional Properties and configure the following properties and click Finish:

DatabaseName	pblog
Password	pblog
ServerName	localhost
URL	jdbc:mysql://localhost:3306/pblog
Url	jdbc:mysql://localhost:3306/pblog
User	pblog

Click the name of the new Data Source in the list and click Ping.



Resources >	JDBC > Connec	tion Pools > cp_pblog_XA	
General	Advanced	Additional Properties	
Edit Con	nection Po	ol	
Modify existing	g JDBC connectior ts Ping	n pools. A JDBC connection poo	ol is a grou

If all is configured correctly a success message should appear.

h



Modify existing JDBC connection pools. A JDBC connection pool is a group of reusable connection Defaults

Finally, let's create the JNDI reference to use in referring to this connection pool.

Expand Resources -> JDBC -> JDBC Resources and click New

Application Server	JDBC Resources	
Applications     Enterprise Applications	JUBC resources provide applications with a means to	connect to a da
► 📄 Web Applications	Resources (6)	
EJB Modules	B   New Delete Enable	Disable
Connector Modules	JNDI Name	t Enabled
Lifecycle Modules	jdbc/iepseDerbyXA	true
Application Client Modules	jdbc/iepseDerbyNonXA	true
👷 Web Services	jdbc/TimerPool	true
🛅 ЈВІ	jdbc/defaultDS	true
Service Assemblies	jdbc/default	true
Components	jdbc/CallFlowPool	true
Shared Libraries  ESB Custom MBeans Custom MBeans DDBC DDBC DDBC DDBC DDBC DDBC DDBC DDB		

Name the resource reference "jdbc/ cp\_pblog\_XA" and choose the correct pool, cp\_pblog\_XA, as the pool name, then click OK

Resources > JDE	3C > JDBC Resources
New JDBC	Resource
Specify a unique J underscore, dash,	NDI name that identifies the JDBC resource you want to create. Name or dot characters.
JNDI Name: *	jdbc/cp_pblog_XA
Pool Name: *	cp_pblog_XA  Use the JDBC Connection Pools page to create new pools
Description:	
Status:	Enabled

This completes database configuration.

The connection pool created above will be good for anything that needs a XA connection pool for the MySQL database pblog.

# Implementing the service

#### **Create a Project Group**

Let's start by creating a NetBeans Project Group, FacilitySvcProjGrp, in as a Folder of Projects in a convenient location in the file system.



Create New Group
Name: FacilitySvcProjGrp
C Free Crew
Contains any projects you like. Can be updated manually or automatically.
✓ Use Currently Open Projects
Automatically Save Project List
C Project and All <u>R</u> equired Projects
Contains a master project and all projects it requires, recursively,
Master Project: Browse
Folder of Projects
Contains any projects found beneath a given folder on disk.
Folder: G:\GlassFishESBv21Projects\FacilitySvcProjGrp
Create Group Cancel Help

#### **Create BPEL-based Service Implementation**

The process will be invoked as a web service, using the WSDL FacilitySvc we will create shortly, and will invoke a Database service encapsulating access to the ui\_facility table in the MySQL database.

To complete this section we will:

- 1. create XML Schema definition for input and output messages
- 2. create a web service interface WSDL definition
- 3. create a Database BC WSDL for a service which will select a code and description for all facilities
- 4. create a Database BC WSDL for a service that will select all information about a specific facility
- 5. implement two operations, each of which will orchestrate one of the two Database BC services
- 6. create a composite application and deploy it
- 7. create a test case and test the application

Let's create a new BPEL Module Project, FacilitySvc\_BM. Once done, a skeleton BPEL module, facilitySvc\_BM, will have been created. Rename it to bpFacilitySvc\_BM. You don't really have to. I just don't like names other people picked for stuff so I tend to rename things more to my liking.

J Proj ◀0 × J Files J Services	S facilitySvc_BM.bpel ×
E FacilitySvc_BM	Source Design Mapper Logging
🛱 🔂 Process Files	
facilitySvc_BM.bpel	
🗄 🗂 Referenced Resources	
	facilitySvc_BM
	Proces <mark>s</mark> Start

The service implementation will provide a list of all facilities and, given a facility code, all details available for the specific facility. A web service with two operations will be implemented.

The List operation will accept a dummy string, which it will ignore, and will return a list of facility\_code and description pairs.

The Details operation will accept a facility\_code and will return all facility details for the nominated facility.

Let's create a New -> XML Schema, Facility, with target namespace "urn:Sun:Michael.Czapski:XSDs:/Facility".

hoose File Type roject: R FacilitySvc_BM	
roject: 📴 FacilitySvc_BM	
ategories:	Eile Types:
ESB 🔺	XML Document
SOA	XML Schema
	DTD Entity
XML	WSDL Document     External XML Schema Document(s)
escription:	
	ESB SOA WebSpace/Liferay Plugins XML escription:

Steps	Name and Location	
. Choose File Type 2. Name and Location	File Name: Facility	
	Project: FacilitySvc_BM	
	Created File: :\GlassFishESBv21Projects\FacilitySvc\FacilitySvc_BMTest\	FacilitySvc_BM\src\Facility.xs
	Target Namespace: urn:Sun:Michael.Czapski:XSDs:/Facility	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~		

Add 4 elements at the root level: FacListReq, FacListRes, FacDetailsReq and FacDetailsRes.



To FacListReq add a leaf element, dummyString of "existing, built-in" type string and minOccurence of 0.



🇊 Add Element	×
Name: dummyString	
Type:	
O Inline Complex Type	
C Inline Simple Type	
C No Type	
Use Existing Type	
short         string         time         token         unsignedByte         unsignedInt         Current Selection: string (Global Simple Type)         Set of any allowed characters in XML.         Preview:	
<pre><xsd:element name="dummyString" type="string"> </xsd:element></pre>	▲ ▼
OK Cancel	Help

Use properties pane to change min occurrence for dummyString to 0.

To FacListRes add an Element FacList with maxOccurs of "unbounded".

E- C FacListRes	e e	
	ist [1*]	
	FacList [Local Element] - Properties	
	Properties	
Jimple Types	Kind	Local Element
	ID	Not specified
	Name	FacList
	Structure	Click to customize
	Nillable	False (not set)
	Fixed Value	Not specified
	Default Value	Not specified
	Max Occurs	unbounded
	Min Occurs	1
	Form	Default for schema (not set)

Under FacList add two leaf elements, facCode and description, both of string type.



To FacDetailsReq add a string element facCode.



To FacDetailsRes add the following elements, each of string type and of nominated optionality:

Name	Туре	Min Occurence	Max Occurrence
facCode	string	1	1
description	string	1	1
addressLine1	string	0	1
suburbTown	string	0	1
state	string	0	1
postCode	string	0	1
country	string	0	1
status	string	0	1



FacilityListReq and FacilityListRes are messages that will be associated with the operation opGetFacilityList, which will return the cafility code and description pairs for all facilities in the database table.

FacilityDetailReq and FacilityDetailRes are messages that will be associated with the operation opGetFacilityDetail, which will return all available details for a specific facility.

Note that the opGetFacilityList input message consists of a single optional string. The content of the message, if any, will be ignored since the service implementation will return the complete list of facilities every time the operation is invoked.

Let's now create a New -> WSDL Document, of WSDL Type Concrete, Binding: SOAP and Type Document/Literal, named FacilitySvc, with target namespace of "urn:Sun:Michael.Czapski:WSDLs:/FacilitySvc", which uses the schema elements.

iteps	Name and Loca	tion	
. Choose File Type . Name and Location . Abstract Configuration	File <u>N</u> ame: Facilit	tySvc	
. Concrete Configuration	Project: Fac	alitySvc_BM	
	Folder: src		Browse
	Created File: ss	FishESBv21Projects\FacilitySvc\FacilitySvc_BMTest\FacilitySvc_B	M\src\FacilitySvc.w
	Target Namespac	e: urn:Sun:Michael.Czapski:WSDLs:/FacilitySvc	
	W <u>S</u> DL Type:	<u>A</u> bstract WSDL Document <u>Concrete WSDL Document</u>	
	Binding:	SOAP	
	Туре:	Document Literal	
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~			

Rename operation to opGetFacList and start naming and typing messages.

eps	Abstract Configura	tion		
Choose File Type	Port Type Name: F	acilitySvcPortType		
Abstract Configuration	Operation <u>N</u> ame:	pGetFacList		
Concrete Configuration	Operation Type:	equest-Response Operation		-
	Input:	Message Part Name	Element Or Type	
		nsgFacListReq	xsd:string	
		Select Ele 	ement Or Type e adiitySvc_BM	
	Output:	Add Rer H	j src/Facility.xsd └み Complex Types └み Elements ⊕-≪≫ FacDetailsReq	
			Some and the second secon	
	- Fault:	Add Ref G Built-I Messag ar	n Schema Types nyType nyURI (restricts :anySimpleType)	
		∎ ba ⊡ bo	ase64Binary (restricts :anySimpleType) bolean (restricts :anySimpleType) vte (restricts :short)	
		Add Rer da	ate (restricts :anySimpleType) ateTime (restricts :anySimpleType)	
1.2	_ Generate partn	erlinktype automa 🖬 de 🖬 de	comai (restricts :anySimple1ype) puble (restricts :anySimpleType) uration (restricts :anySimpleType)	
200			NTITIES NTITY (restricts :NCName)	

Input: msgFacListReq of type FacListReq.

Output: msgFacListRes of type FacListRes.



Finish the wizard.

This gives us a WSDL interface definition for a service with a single operation, opGetFacList. We now need to add another operation, opGetFacDetails, which uses FacDetailsReq and FacDetailsRes messages for input and output respectively.

Let's add two new messages, opFacDetailsReq and opFacDetailsRes.



Let's now rename parts to msgFacDetailsReq and msgFacDetailsRes.

🔁 🔁 Messages
🖨 🖂 opGetFacListRequest
🗄 📴 msgFacListReq ns:FacListReq
🖨 🗹 opGetFacListResponse
😟 🕒 🕒 msgFacListRes ns:FacListRes
F 🗹 opFacDetailsReq
🖳 🗄 📴 msgFacDetailsReq
- g opfacDetailsRes
😟 🖻 🕒 📴 msgFacDetailsRes
🖻 🧑 Port Types
the state comparities

Let's now set the data types for these parts: FacDetaisIReq and facDetailsRes respectively.





🛱 🖓 Messages
🗇 🖂 opGetFacListRequest
🗄 🖻 msgFacListReq ns:FacListReq
🖻 🗹 opGetFacListResponse
🕀 🖙 🖻 msgFacListRes ns:FacListRes
🔁 🗹 opFacDetailsReg
🖻 msgFacDetailsReq ns:FacDetailsReq
🖻 🗹 opfacDetailsRes
msgFacDetailsRes ns:FacDetailsRes
Port Types

This gives us two new messages for the new operation.

Let's right-click the FacilitySvcPortType node, choose Add and choose Operation.



Rename operation to opGetFacDetails and choose opFacDetailsReq message type for Input.

🇊 Create New Ope	eration		×
Operation <u>Name</u> :	opGetFacDetails		
Operation <u>Type</u> :	Request-Response Operation	•	
Input:	FacilitySvcOperationRequest	<b>•</b>	[
	FacilitySvcOperationRequest tns:opGetFacListRequest		
$\longrightarrow$	tns:opGetFacListResponse tns:opFacDetailsReq tns:opfacDetailsRes		
	<u>A</u> dd <u>R</u> emove		7
Output:	FacilitySvcOperationResponse	<b>•</b>	
	Message Part Name	Element Or Type	
	part1	xsd:string	
	<u>A</u> dd <u>R</u> emove		
Fa <u>u</u> lt:		•	
	Message Part Name	Element Or Type	
	<u>A</u> dd <u>R</u> emove		

Choose opFacDetailsRes message type for Output.

Operation Name:	: JopGetFacDetails		
Operation Type:	Request-Response Operation		-
nput:	tns:opFacDetailsReq		*
	Message Part Name	Element Or Type	
	parti	xsd:string	
jutput:	FacilitySvcOperationResponse		*
Output:	FacilitySvcOperationResponse		
Qutput:	FacilitySvcOperationResponse FacilitySvcOperationResponse tns:opGetFacListRequest		•
Output:	FacilitySvcOperationResponse FacilitySvcOperationResponse tns:opGetFacListRequest tns:opGetFacListResponse		
Dutput:	FacilitySvcOperationResponse FacilitySvcOperationResponse tns:opGetFacListRequest tns:opGetFacListResponse tns:opFacDetailsReq		
Output:	FacilitySvcOperationResponse FacilitySvcOperationResponse tns:opGetFacListRequest tns:opGetFacListResponse tns:opFacDetailsReq tns:opFacDetailsRes		
Output:	FacilitySvcOperationResponse FacilitySvcOperationResponse tns:opGetFacListRequest tns:opGetFacListResponse tns:opFacDetailsReg tns:opFacDetailsRes Add Remove		<u>*</u>

Complete the wizard.

The abstract operation, opGetFacDetails is now configured.

🖻 🖃 o	pGetFacList
	input1 Message="opGetFacListRequest"
S	output1 Message="opGetFacListRespor
	pGetFacDetails
±-5	input2 Message="opFacDetailsReq"
( 🗄 🏀	output2 Message="opfacDetailsRes"

We need to add a new binding.

Right-click the FacilitySvcBinding, choose Add and choose Binding Operation.



New operation gets added. We must now configure it.



Right-click opGetFacDetails and choose Add -> SOAP Operation.

⊡ in output1 ⊡ in soap:b pr	ody		
input2	Cu <u>t</u> Cop <u>y</u>	Ctrl+X Ctrl+C	
	Paste	Ctrl+V	
Evtensibility Elemente	<u>A</u> dd	۰.	Documentation [E
	<u>D</u> elete	Delete	SOAP Operation
	Change Or	der	<u>√</u>

Right-click in Input2 and Output2 in turn and choose Add -> SOAP Body



This completes the WSDL definition. Switch to Source mode and Validate and Check XML.



Now we need to prepare interfaces definitions for Database-related services.

Each of the two service operations will execute a different SQL statement to get different data from the database.

With GlassFish ESB and the Database Binding Components there are a couple of ways in which a database service can be configured. I chose to use the SQL File method. I create a SQL File containing the prepared statement then use this SQL File to construct a WSDL for the Database BC.

The first operation, opGetFaclList, will return a list of facility code and description pairs. The SQL statement will be:

select facility\_code, description from ui\_facility;

The second operation, opGetFacDetails, will return all there is to know about a specific facility. The SQL statement will be:

```
select * from ui_facility where facility_code = ?;
```

Let's construct the SQL Files and Database BC WSDLs one at a time.

Browse
_BM\src\FacList.

Let's create a New -> Other -> Empty File, named FacListDB.sql.

Let's enter "select facility\_code, description from ui\_facility;" into the editor window and choose the MySQL connection we created earlier as the connection to use for this SQL statement.

8	bpFacility	ySvc_BM.bpel 🗙 🛃 Facility.xsd 🗙 🎇 FacilitySvc.wsdl 🗙 📑 FacList.sql * 🗙
Cg	nnection:	🔽 🔁 🖓 - 💀 - 💀 🐨 🚱 - 💀 - 🕄 -
	seled	jdbc:derby://localhost:15 jdbc:derby://localhost:15 jdbc:derby://localhost:15 jdbc:mysql://localhost:73( jdbc:mysql://localhost:3306/pblog [pblog on Default schema]

To verify that the connection and the statement work let's click the "Run SQL" button.

🕞 br	pFacilitySvc_BM.bpel ×	🚯 Facility.xsd 🗙 🔛 FacilitySvc.wsdl 🗴 📄	FacList.sql ×
Conne	ection: jdbc:mysql://local.	. 🔽 🛐 🐺 🚱 🐺 💽 - 题	- 🔍 😓 4
	select facility_	code, de Run SQL (Ctrl+Shift+E) 11_fac	ility
1:4	40   INS		
selec	t facility_code, des ×		
	📰 📰 📰 🗷 L 🍣	9 K < ≽ 为   Page Size: 20	Total Rows: 11
#	facility_code	description	
1	ARMC	A RED MEDICAL CENTRE	
2	BIGH	A BIG HOSPITAL	
3	D204	Blue Mountains Hospital	
4	D210	Nepean Hospital	
5	D214	Springwood Hospital	

The connection and the statement work as expected.

Let's now create a new WSDL Document, FacListDB.wsdl, a Concrete WSDL, using Database Binding of type SQL File.

teps	Name and Locati	on
Choose File Type Name and Location Select one of the following : Please select a SQL File Enter A Prepared Statement Enter JND Name	File <u>N</u> ame: FacList	DB tySvc_BM
Enter JNUI Name	Folder:  src	Browse
	Created File: G:\G	assFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\src\FacListDB.wsdl
	Target Namespace:	http://j2ee.netbeans.org/wsdl/FacilitySvc_BM/FacListDB
	WSDL Type:	C Abstract WSDL Document
		Concrete WSDL Document
	Binding:	DATABASE
	Туре:	SQL File
X		

Choose the MySQL connection we created earlier.

Steps	Select o	ne of the following :
Choose File Type     Name and Location     Select one of the following	URL:	<select an="" from="" item="" list="" the=""></select>
: 4. Please select a SQL File 5. Enter A Prepared Statement 5. Enter JNDI Name		<select an="" from="" item="" list="" the=""> jdbc:derby://localhost:1527/sample [app on APP] jdbc:derby://localhost:1527/travel [travel on TRAVEL] jdbc:derby://localhost:1527/vir [vir on VIR]</select>
<u>~~</u>		jdbc:mysql://localhost:3306/pblog [pblog on Default schema]

Choose the FacList.sql SQL file we created earlier.

Steps	Please select a SQL	File
1. Choose File Type 2. Name and Location 3. Select one of the following : 4. Please select a SOL File	COL File Location ·	G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\src\FacList.sql
·		

Click the "Discover Parameters" button (there will be non in this case) and the "Discover Resultset" button, then click Next.

teps	Enter A Prepared Statement			
Choose File Type Name and Location Select one of the following : Please select a SQL File Enter A Prepared Conternent	Enter a valid prepared statement	Prepared Statement Edito	P <u>a</u> rameters	
Enter JNDI Name	select facility_code, description from ui_facility	Discover Paramet	Parameter Name	Parameter Type
	Status:	Agd Parameters Remove Parameters	Resultset	Column Turne
			Column Name	Column Type
		Discover Resultset	description	java.lang.String
~				

Specify the name of the pool we created earlier, "jdbc/cp\_pblog\_XA", and click Finish.

Choose File Type     A     JNDI Name     Jdbc/cp_pblog_XA	
Please select a SQL File     Enter A Prepared Statement	

Rename the operation under Port Types -> jdbcPortType from newUntitled\_Operation ot opGetFacList.

FacListDB.wsdl ×							
Source WSDL Partner 📔 🛄 🏍 😽							
http://j2ee.netbeans.org/wsdl/FacListDB							
types							
🗄 🐻 http://schemas.sun.com/jbi/wsdl-extensions/jdbc/							
Imports							
🛱 🕀 Messages							
🗄 🖂 inputMsg							
🗄 🖂 outputMsg							
Der Types							
i jdbcPortType							
hewuntitled_Operation							
🖻 💀 binding PortType="jdbcPortType"							
🗓 🖳 🤕 jdbc:binding							
emer in the term of							
🛱 🖓 Services							
🖻 👰 service							
🖻 💮 port Binding="binding"							
jdbc:address							
🖻 🖳 Extensibility Elements							
i jdbcpartner							

Note that Bindings -> Binding Operation gets renamed as well.



This completes the Database BC WSDL for the SQL statement that will return the list of facility codes and descriptions.

Let's now create the SQL File and the Database BC WSDL for the statement "select \* from ui\_facility where facility\_code = ?".

Let's create New -> Other -> Empty File, named FacDetails.sql, using the MySQL connection created earlier.



Let's now create a New -> WSDL Document, named FacDetailsDB.wsdl, a Concrete WSDL using Database Binding of type SQL File.

Steps	Name and Location	
. Choose File Type 2. Name and Location 5. Select one of the following :	File Name: FacDetailsDB	
<ul> <li>Please select a SQL File</li> <li>Enter A Prepared Statement</li> </ul>	Project: FadilitySvc_BM	
Enter JNDI Name	Folder: src	Browse
	Iarget Namespace:       http://j2ee.netbeans.org/wsdl/FacilitySv         WSDL Type:       C Abstract WSDL Document         © Concrete WSDL Document	rc_BM/FacDetailsD8
	Target Namespace:       http://j2ee.netbeans.org/wsdl/FacilitySv         WSDL Type:       Cabstract WSDL Document         © Concrete WSDL Document         Binding:       DATABASE	rc_BM/FacDetailsDB

Let it use the MySQL connection we created earlier.

Steps	Select one of the following :
<ol> <li>Choose File Type</li> <li>Name and Location</li> <li>Select one of the follow</li> <li>Please select a SQL File</li> </ol>	Aing URL : jdbc:mysql://localhost:3306/pblog [pblog on Default schema]
5. Enter A Prepared Statemer	it 🔽

Let it use the SQL File, FacDetails.sql, we created earlier.

eps	Please select a SQL	. File	
Choose File Type Name and Location Select one of the following : Please select a SQL File	SQL File Location :	G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\src\FacDetails.sql	Browse

Click the "Discover Parameters" button and change the name of the parameter to "pFacCode".

Steps	Enter A Prepared Statement	
Choose File Type     Name and Location     Select one of the following :     Select as QL File     Enter A Prepared     Statement     Enter JNDI Name	Enter a valid prepared statement select * from ui_facility where facility_code = ?	Prepared Statement Editor Parameters Parameter Name Parameter Ty PFacCode java.lang.Strin Discover Parameter
		Add Parameters Remove Parameters

Click the "Discover Resultset" button and click Next.

cps	Enter A Prepared Statement			
Choose File Type Name and Location Select one of the following : Please select a SQL File Enter A Prepared	Enter a valid prepared statement	Prepared Statement Edi	tor P <u>a</u> rameters	
Statement Enter JNDI Name	select * from ui_facility where facility_code = ?	-	Parameter Name	Parameter Type
			pFacCode	java.lang.String
		Discover Paramet		
		Add Parameters		
		Remove Parameters	,	
	S <u>t</u> atus:	/	Resultset	
		- 🖌	Column Name	Column Type
		Discover Resultset	facility_code	inun lang String
				Java.lang.Sunny
		- biddvici (debulact	description	java.lang.String
			description status	java.lang.String java.lang.String java.lang.String
			description status addr1	java.lang.String java.lang.String java.lang.String java.lang.String
			description status addr1 addr2	java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String
			description status addr1 addr2 city	java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String
		- DECOT LESSUEL	description status addr1 addr2 city state	java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String java.lang.String

Specify the connection pool to use, "jdbc/cp\_pblog\_XA" and Finish.

Steps	Enter JNDI Name
<ol> <li>Choose File Type</li> <li>Name and Location</li> <li>Select one of the following :</li> <li>Please select a SQL File</li> <li>Enter A Prepared Statement</li> <li>Enter JNDI Name</li> </ol>	JNDI Name jdbc/cp_pblog_XA

Rename "newuntitled\_Operation" to opGetFacDetails.



Both Database BC WSDLs are now ready. The service interface, FacilitySvc WSDL is also ready. We can now construct the business process to orchestrate the two Database BC services.

Let's drag the FacilitySvc WSDL onto the left-hand swim line of the bpFacilitySvc\_BM process and release over the target marker.



Double-click the object to open the editor panel, change the name for FacilityRR and click OK.

Name: Fa	IcilityRR	
WSDL File:	acilitySvc_BM] FacilitySvc.wsdl	
Use Existin	ig Partner Link Type	
Partner Lin	k <u>Type</u> : FacilitySvc	
My <u>R</u> ole:	FacilitySvcPortTypeRole	
Partner Ro	le: [	
	Swap Roles	
C Use a New	ly <u>C</u> reated Partner Link Type	
Wrapper N	ame: FadlitySvcWrapper	
Partner Lin	k Type Name:	
Process	s will implement (My Role)	
Role Na	ame:	
Port Ty	pe: FacilitySvcPortType	ŧ.
Lore IT		
F Partner	service will implement (Partner Role)	
Partner Role Na	service will implement (Partner Role) Im <u>e</u> :	

Let's now drag the FacListDB WSDL and drop it onto the target marker at the righthand swim line.

E-Se FacilitySvc_BM	Source Des	sign Mapper	Logging		<u> </u>
FacDetails.sql					
FacDetailsDB.wsdl					
FacDetailsDB.xsd					
FacList.sql					
FacListDB.wsd					
FacListDB.xsd		******			
Facility.xsd		fa	cility Swc_BM		0
FacilitySvc.wsd		2-		And a state of the	
bpFacilitySvc_BM.bpel				and the second	
Referenced Resources		Pi	ocess Start	-	
	- 2 2			100	N
					No.
	FacilotyRR	<b>4</b>			N.
					N.
			000		* -
	opGetfac		6		<u></u>
			Y		iller i
			$\sim$		

Rename the partner link to DBList.

Let's drag the FacDetailsDB WSDL onto the right-hand swim line and release. Rename the partner link to DBDetails.



The FacilitySvc WSDL (FacilityRR partner link) has two operations – opGetFacilityList and opGetFacilityDetails. We will implement each as a separate stream of activities starting with the Pick activity.

Pick Activity is a 'either or' selection. The service will be invoked using either the opGetFacList operation or the opGetFacDetails operation. Implementation of the logic required by each operation is different. The mechanism that allows the BPEL Process to distinguish between the operations required for the particular invocation is the Pick Activity.

Let's drag the Pick activity from the palette to the canvas and drop it at the target marker in the process scope.



Connect the "OnMessage" Activity to the first operation of the FacilityRR partner link.



The service is a request/reply service. OnMessage acts as a receive activity for the specific operation. The Reply Activity will provide the reply on completion. Let's drag the Reply Activity onto the target marker and release.



Between receiving a trigger message and replying, the service must access the database and receive the list of facilities. Let's drag the Invoke Activity, which will cause invocation of the DBList service, onto the canvas and connect it to the DBList service's operation.

M.bpel * ×			] Palette	0	×
n Mappe			🗆 Web Service		
			🖸 Invoke	Receive	
			Reply	Partner Link	
	and the second se	20	Basic Activities		
	1		Assign	🕞 JavaScript	
-			Validate	Empty	
-	facilitySvc_BM	_ <b>~</b>	🕒 Wait	🛞 Throw	
	$\cap$		🐼 ReThrow	🔘 Exit	
	Process Start		( Compensate	OcompensateScope	
			Structured Acti	ivities	
	Pick1		🛞 If	😭 While	
			🔚 RepeatUntil	💮 ForEach	
	OnMess @	DBList	Pick	I Flow	-
			J Invoke1 [Invoke] - P	roperties D	×
			🖂 Main		
		opGet	Property Editor		
	Invoke1		Name	Invoke1	
			Documentation		- 14

Let's create an input variable for the FacilityRR trigger message - FacListReq.

Double-click the OnMessage activity, click the Create button next to the Input message, change the name and complete.

	Process Start	( Compensate	😶 Co
		□ <u>Structured Activ</u>	vities
	Pick1	🛞 If	😭 Wł
FacilutyRR		RepeatUntil	📀 Fo
		ew Input Variable	×
onGetFar	Nam (Nam	e: FacListReq	
		e: tns:opGetfacListRequest	
	Message Handler] - Proper	e: FacilitySvc BM	-
opGetFa	Main Correlations	indencysve_ord	
	Coreadoria		
	Partner Link: FacilutyRR		ncel
	Operation: opGetfacl ist	-	
	operation		
	Input Variable:	Create Browse	10
	· · · · · · · · · · · · · · · · · · ·		
		Ok Cancel <u>H</u> elp	
			8

Let's connect the Reply1 activity to the FacRR opGetFacList operation and create the output variable for the Reply1 activity – FacListRes.

FacilutyRR opGetfac opGetFa	Message Sequen	DBList DB
	Reply1 [Reply] - Property E         Main       Correlations         Name:       Reply1         Partner Link:       FacilutyRR         Ogeration:       opGetfacList	New Output Variable     X       Name:     FacListRes       Iype:     Ins:opGetfacListResponse       Scope:     facilitySvc_BM       Ok     Cancel
Output - XML che XML valic 0 Error(s XML valic	Normal Response     Qutput Variable:     Eault Response     Fault Name:     Fault Variable:     Fault Variable:	Create Browse Choose Create Browse

The input and output variables will contain the request and response messages. Invocation of the DBList partner will result in a set of records being returned. We need to connect the Invoke1 activity to the DBList partner and provide the input and output variables for this service. Double-click the Invoke1 activity and create input variable, DBListReq, and output variable, DBListRes, to hold the messages.

FacilutyRR	0	nMessage		DBList	_	RepeatUntil	<ul> <li>(*)</li> <li>(*)</li></ul>
opGetFac			New	Output Variable		   Invoke1 [Invoke] -    — Main	Propertie
	Main Correlation	roke] - Property 15	<u>N</u> ame: <u>Type</u> :	ths:outputMsg			
	<u>N</u> ame: Partner Link: Operation:	Invoke1 DBList opGetFacList	<u>S</u> cope:	facilitySvc_BM	<b>→</b> [	Ok Canc	el
	Input Variable: Output Variable:	DBListReq		_	→ Crea	te   Browse   te   Browse	
Mut - XML chec				Ok	Can	cel <u>H</u> elp	

The output message is expected to contain the list of facility codes and descriptions. To allow our BPEL service to return this list to the caller we must map the list from the output message of the DBList service to the response message of the BPEL process. Since the structure is a repeating structure we need to add a ForEach loop in order to map each iteration from the DBListRes to FacListRes. Let's drag the ForEach activity to the target marker between Invoke1 and Reply1.



Let's configure the loop conditional to iterate over all elements of the list starting at 1 and continuing for as long as there are elements in the list. Double-click the ForEach icon.



Select "Start Value", drop down the Number functiod list and choose Numeric Literal.

Source Design Mapper	Logging		
🚰 Operator 🛛 🚆 Boolean 🛛 🗛 S	ring 🖳 Node 📃 Number 🔞 D	ate & Time 🔀 BPEL	
All Output	🕅 Number		Input All 🖻
Variables	[1] Numeric Liter		For Each 🔇 Start Value 🗇 –
FacListRes	Sum		Final Value Completion Condition

Change the value to 1, which is the BPEL's base index, and connect to the Start Value node.



The Final Value node will be the count of items in the list. There is a Count functoid that will be used to count the number of items. Lets select Final Value, drop down the Node functiod list and click the Count functoid.

🚰 Operator 🛛 📑 Boolean 🗼 String	Number	Date & Time 🔠 BPEL	
All Output	Local Name		Input All
Variables  DBDetailsRes  DBDetailsReq  Control Contro	Namespace URI		 For Each ⇒ Start Value ⇒ Final Value ⇒
	Count		Completion Condition 🗇

Now connect the FacListDB\_Record node to the Count and the Count to the Final Value.

bpFacilitySvc_BM.bpel * ×	N		
Source Design Mapper Logging			
Boolean A String	E Node 🔜 Number Yo Date & Time 🔂 BPEL	Inpu	t All 🖅 *
Variables  DBDetailsRes  DBDetailsReq  FacDetailsReq  DatailsReq  DatailsReq  DatailsReq  DatailsReq  DatailsReq  DatailsReq  DatailstRes  DatailstR	Count Node Set Number	Comple	For Each ← > Start Value

Let's now add the Assign Activity to the ForEach loop to perform the actual mapping.



Double-click Assign1 to switch to the Mapper editor and map the nodes of the DBListRes structure to the corresponding nodes of the FacListRes structure. This will be done in a few stages, for clarity.

Let's first add a predicate to the DBListRes at the FacListDB\_Record node level.



The predicate will be based on the current value of the ForEach1Counter node.



Let's add a predicate for the target, FacListRes at the FacList repeating node.



Set the predicate to the current value of the ForEach1Counter node as well. This way the assignment, which comes next, will use corresponding entries in the source and target lists.

bpFadlitySvc_BM.bpel * x Source Design Mapper Logging	
	Input All
Image: Second	Variables DBListRes FacListRes FacListRes FacList & FacList &

Switch to Design view, click the Warning icon next to the Assign1 Activity and note the warnings.



The BPEL editor complaints that the nodes don't have the same data types. Ignore the warning since datatype in both cases are actually compatible.

Click on the Pick symbol and make sure to check the "Create Instance" property.



While the DBList service does not require input the BPEL process will not be able to be built if the input message to this service is not initialized and if the input message to the process itself is not used. Let's add an Assign activity and map the FacListReq - >dummyString message node to the DBLisgReq message node.



bpFacilitySvc_BM.bpel* x 3 Actual_20090622203804_F.xml x 3 Actual_200906222     Source Design Magner Logging	205658_S.xml ×
Coperator 📑 Boolean 🔬 String 📲 Node 🧮 Number 📸 Date & Time 🔂 BP	E
All Output	Input All 🗗 +
□       Yariables         □       DBDetailsRes         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □         □       □	Variables DBDetailsRes DBDetailsReg FacDetailsReg FacDetailsReg DBListReg DBListReg DBListReg
Properties     Protection	Properties 👝 🕁 FacListRes 🐏 – 🕀 FacListReg ங – 🕀 Partner Links 🖶 – 🕀

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

The process will be built successfully with warnings. Note the warnings – review them and ignore them for this process.

] C	0utput - build.xml (dist_se)
	deps-jar-dist: do-dist:
	Created dir: G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\build
12	<u>G:/GlassFishESBv21Projects/FacilitySvcProjGrp/FacilitySvc_BM/src/bpFacilitySvc_BM.bpel:60: 32</u>
1.00	WARNING: The types of "From" and "To" activities are different: "description" and "string".
	G:/GlassFishESBv21Projects/FacilitySvcProjGrp/FacilitySvc_BM/src/bpFacilitySvc_BM.bpel:37: 24
	WARNING: The types of "From" and "To" activities are different: "FacListReq" and "FacListDB_Request".
	<u>G:/GlassFishESBv21Projects/FacilitySvcProjGrp/FacilitySvc_BM/src/bpFacilitySvc_BM.bpel:48: 32</u>
	WARNING: The types of "From" and "To" activities are different: "facility_code" and "string".
	Copying 7 files to G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\build
	Building jar: G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\build\SEDeployment.jar
	post-dist:
	dist_se:
	BUILD SUCCESSFUL (total time: 2 seconds)

This completes the opGetFacList operation implementation.

Now we will implement the opGetFacDetails operation. Here the input message will contain the facility code used to look up the facility whose details are to be returned.



Click the Pick symbol and the Add OnMessage icon.

Add Assign, Invoke, Assign and Reply activities.



Connect the new OnMessage activity to the opGetFacDetails operation.



Connect Reply2 to the opGetFacDetails operation.



Drag the DBDetails.wsdl onto the target market at the right hand swim line and rename it to DBDetails.



Connect Invoke2 to the DBDetails partner operation.

	OnMessage	
1		
1	Sequen	opGetFa
		DBDetails
5		
		opGetF8

Double-click the new OnMesage activity and create a new Input Variable FacDetailsReq.

Message Sequence1	<b>A</b>	nMessage	New In	DBList put Variable		Compensate	ritie X
ForEach1	andler] - Property	Message Sequen Editor	Name: Fa	cDetailsReq s:opFacDetailsF clitySvc_BM	Req	ot Court	
Partner Link: Operation:	FacilutyRR opGetFacDetails					in mentation	
Input Variable:	ļ	/	/	Create	Browse	ssage Handler]	
			Ok	Cancel	Help	J	

Double-click the Reply2 Activity and create a new Output Variable FacDetailsRes.

Double-click the Invoke2 Activity and create two new variables, Input DBDetailsReq and Output DBDetailsRes.

Invoke1 ForEach1 Scope1	Sequen	Rev	DBDetails	able	ain perty Editor ne	Invoke
Invoke2 [Invoke] - Proper         Main       Correlations         Name:       Invoke2         Partner Link:       DBDetails	ty Editor	<u>Type:</u>	DBDetailsRes tns:outputMs facilitySvc_B	) Ig M		
Ogeration: opGetFacDetail Input Variable: DBDetailsReq Output Variable:			Create Create	Browse		zel
<		Ok	Cancel	Help	DetailsReg-	

Double-click Assign3 and add mapping from FacDetailsReq->msgFacDetailsReq->facCode to DBDetailsReq->part->pFacCode.

DepFacilitySvc_BM.bpel * ×					
Source Design Mapper Lo	gging				
😵 Operator 📑 Boolean 🔬 Strin	g 🖳 🖁 🖓 Node	Number	🛅 Date & Time	🔂 BPEL	
tter → All Output				Input	All 🕂 🕶
🖃 🕂 🕂 Variables					Variables 🖶 – 📮
🕀 🎦 DBDetailsRes				DBDeta	ilsRes 🌯 🕂
🕀 🔤 DBDetailsReq				DBDetail	sReg 🖳 🗕
Hard FacDetailsRes				DDDCtu	
				pa	rt ≌
l ⊡ is in the second seco				—⊳pFacCode ≪	≫
				Propertie	es 🚞- 🗄 🔰
🕀 🎦 DBListRes				FacDetail	sRes 🌯 🕂
庄 🖓 DBListReq				FacDetai	ilsReg 🖳 🕂

Switch back to Design view.

Double-click Assign4 activity and map nodes of the DBDetailsRes structure to the corresponding nodes of the FacDetailsRes structure.

bpFacilitySvc_BM.bpel * ×	< Þ 🗸 🗖
Source         Design         Mapper         Logging                @ Operator               Boolean               A String               Boolean	er 🖓 Date & Time 🔂 BPEL
All Output	Input All
Variables  Variables  DBDetailsRes  Spart  Spart Spart Spart  Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Spart Sp	Variables DBDetailsRes DBDetailsReg FacDetailsRes msgFacDetailsRes + facCode <> + facCode <> + addressLine 1 <\$ + suburbTown <\$ + state <\$  + state <\$ 

Switch to Design view, right-click project name and choose Build. The build should complete successfully with warnings.

Out	put - build.xml (dist_se)									
	<u>G:/GlassFishESBv21Projects/FacilitySvcProjGrp/FacilitySvc_BM/src/bpFacilitySvc_BM.bpel:118: 24</u>									
	WARNING: The types of "From" and "To" activities are different: "post_code" and "string".									
23	G:/GIASSFIShESBV2IPFOJECts/FacilitySvCPFOJGrp/FacilitySvC_BM/src/DpFacilitySvC_BM.bpe1:54: 32									
20	WARNING: The types of "From" and "To" activities are different: "facility_code" and "string".									
	<u>G:/GlassFishESBv21Projects/FacilitySvcProjGrp/FacilitySvc_BM/src/bpFacilitySvc_BM.bpel:102: 24</u>									
	WARNING: The types of "From" and "To" activities are different: "status" and "string".									
	<u>G:/GlassFishESBv21Projects/FacilitySvcProjGrp/FacilitySvc_BM/src/bpFacilitySvc_BM.bpel:94: 24</u>									
	WARNING: The types of "From" and "To" activities are different: "facility code" and "string".									
	Copying 1 file to G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc BM\build									
	Building jar: G:\GlassFishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BM\build\SEDeployment.jar									
	post-dist:									
	dist_se:									
	BUILD SUCCESSFUL (total time: 1 second)									

The process is complete.

To deploy this piece of logic to the runtime we need to create a Composite Application Project. Let's call it FacilitySvc\_CA.

je ¶ × ∣Fi	es 🗍	Services		
FacilitySvc_	BM			
Process F	iles			
📑 FacDe	tails.sql			
FacDe	tailsDB.wsdl			
FacDe	tailsDB.xsd			
FacLis	it.sql			
FacLis	tDB.wsdl			
New Compos	ite Applicatio	n Project		
Steps		Name and Locat	ion	
	(16.0/42)			
<ol> <li>Choose Project</li> <li>Name and Loca</li> </ol>	Ject	Project Name	FaditySyc CA	
	10000	rioject <u>H</u> amer	posityovc_on	
		Project Location:	G:\GlassFishESBv21Projects\FacilitySvcProjGrp	Browse
		Project Folder:	G:\GlassEishESBv21Projects\FacilitySvcProjGro\FacilitySvc_CA	1
		rigeer au	Tor forest and over the operation of the product of	
		Set as Main P	roject	
0		-		
				1

Drag the project FacilitySvc\_BM, by project name node, onto the CASA Editor canvas.



Click Build, and when completed, Deploy buttons.

🛜 bpFaci	itySvc_BM.	bpel ×	FacilitySvc_CA.ca	sa * ×
Source	Design		R 10 V	
L Ports		1	Build Project	JBI Modules
		1	🛛 🕐 (BPEL) Facilit	hySvc_BM

The complete CASA map will look similar to that shown below.

Source Design	
WSDL Ports	JBI Modules
FacilitySvcPort	(BPEL) FacilitySvc_BM
port	DBList DBList OBDetails OBDetails
DATABA	

Let's now test the service using the built-in testing facility.

Right-click the Test node and choose New Test Case.



Accept default name.

Steps	Enter the Test Case Name	_
<ol> <li>Enter the Test Case Nam</li> <li>Select the WSDL Document</li> <li>Select the Operation to Test</li> </ol>	Test Case Name: TestCase 1	1
	< Back Next > Finish Cancel Help	1

Choose SOAP WSDL.

🗊 New Test Case	×
Steps	Select the WSDL Document
1. Enter the Test Case Name	WSDL Documents:
2. Select the WSDL	E FacilitySyc CA - Process Files
3. Select the Operation to Test	E FacilitySvc_BM - Process Files
	FacDetailsDB.wsdl
	FacilitySvc.wsd
	FacListDB.wsd
	Selected WSDL: 21Projects/FacilitySvcProjGro/FacilitySvc_BM\src/FacilitySvc.wsd
	< Back Next > Einish Cancel Help

Choose opGetFacList operation and click Finish.

🗊 New Test Case	×
Steps	Select the Operation to Test
<ol> <li>Enter the Test Case Name</li> <li>Select the WSDL Document</li> <li>Select the Operation to Test</li> </ol>	Binding Operations: FacilitySvcPortTypeBindingPort (Binding="FacilitySvcPortTypeBinding") opGetFacilityDetail opGetFacilityList
	Selected Operation: GetFacilityList(msgGetFacilityListReq): msgGetFacilityListRes
	<back next=""> Einish Cancel Help</back>

It may be that the test case will not appear. There appears to be a bit of an issue with NetBeans at this point. Let's go ahead and create TestCase2 for testing the other operation. Right-click the Test node, choose New test case, accept the default name testCase2, choose FacilitySvc.wsdl in the FacilitySvc\_BM project's process files, choose opGetFacDetails operation and click Finish.

Note the two test cases now appear in NetBeans.



Expand the TestCase1 and look at the input message.

Leave the request as is. Recall that the List operation requires no parameters.



Right-click TestCase1 and choose Run.



Clcik Yes to create a new output file.



Double-click the output document and inspect it.

E StacilitySvc_CA
Service Assembly
🕀 💼 Process Files
🗄 🕞 JBI Modules
🗄 🕞 Test
E
🔚 Output
🖂 16/06/2009 3:00:59 PM - Failed
16/06/2000 2:00:50 DM Eniled Newigeter
1 10/00/2009 3.00.59 PM - Failed - Navigator XML Document
whe bocalier

The test output shows a XML instance document with a list of faciliuty and description pairs – what we expect to see.



Let's now run TestCase2. Before we do. Lets modify the input focument to provide a facility code STC.



As before, let's accept the output file and look at it.

Indeed, the XML instance document contains information about facility whose code was STC.



## Test with Soap UI

The service is implemented and, for all appearances, works.

To use the service in a composite application or a web application we need to be abgle to get at the service WSDL and at the service at runtime. That information is available if one knows where to look. Let's use Soap UI plugin to emulate a client invoking the service. This will require us to deal with the location of the WSDL and the location of the service.

Let's open the facilitySvc.wsdl document and inspect the properties of the soap:address node under the FacilitySvcService node.



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
Туре	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
Drovy/Hear Nama	
sun-http-binding	0
HTTP binding component. Provides message p IBL 1.0 compliant environment.	processing capabilities over HTTP protocol in a
bit the compliant criviterinient.	
	Chara

So, the final service 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 let's create a Web Services testing Project.

Create a New -> Java EE -> Web Services testing Project (this assumes you installed the Soap UI Plugin – if not this project type will not be available).

Steps	Choose Project	Choose Project		
1. Choose Project	<u>Categories:</u>	Projects:		
ζα, της	Java Java Web Java EE Java EE Java ME Groovy NetBeans Modules SOA DA Samples	<ul> <li>Enterprise Application</li> <li>Enterprise Application with Existing Sources</li> <li>EJB Module</li> <li>EJB Module with Existing Sources</li> <li>Enterprise Application Client</li> <li>Enterprise Application Client with Existing Sources</li> <li>Packaged Archive</li> <li>Web Service Testing Project</li> </ul>		
	Description:			
a a	Creates a soapUI-based Web Serv Inspect, Call and Test an Create Functional, Comp Create fully functional si	Creates a soapUI-based Web Service Testing Project that allows you to: Inspect, Call and Test any Web Service Create Functional, Compliance and Load-Tests for any Web Service Create fully functional simulations of any Web Service		

Name the project FacilitySvcWSTP and provide the WSDL WRL which you derived a little while ago :

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

🗊 New Project		-	
Steps	Name and Location		
<ol> <li>Choose Project</li> <li>Name and Location</li> </ol>	Project Name: FacilitySvcWSTP		
	Project Location: G:\GlassFishESBv21Projects\FadlitySvcProjGrp\FadlitySvc_BMTest3	Browse	
	Project Eolder: FishESBv21Projects\FacilitySvcProjGrp\FacilitySvc_BMTest3\FacilitySvcProjErp\FacilitySvc_BMTest3\FacilitySvcBMTest3\FacilitySvcProjErp\FacilitySvcBMTest3\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcBMTest3\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcBMTest3\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcBMTest3\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcProjErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPropErp\FacilitySvcPro	SvcWSTP_	
	Initial WSDL (URL/file): http://localhost:29080/FacilitySvcService/FacilitySvcPort?WSDL	Browse	
	Generate TestSuite?		
and a	Generate MockService?		
	<back next=""> Einish</back>	Cancel <u>H</u> elp	

Right-click on the opGetfacList and choose New Request.



Accept default name. Submit the request and observe the result.



Right-click on opGetfacDetails, modify the request to use facility code of "ARMC", submit the request and observe the result.



This is it. The service has been built, deployed and exercised.

#### Summary

In this document we created and exercised a multi-operation web service that provided a list of Healthcare Facilities and details of a specific Facility.

We used the GlassFish ESB v 2.1 infrastructure. In particular, the BPEL 2.0 Service Engine was used to orchestrate access to the relational database and to expose the logic as a SOAP over HTTP Web Service. We used the SOAP/HTTP Binding Component to expose the process as a service. We used the NetBeans tooling to create SQL Files containing prepared statements and to create Database BC service interfaces for these statements. We used the built-in JUnit testing facility to test both service operatins and, independently, a Soap UI plugin-provided Web Service testing Project type to test the service.

Building a multi-operation web service that accesses a relational database is fairly straight-forward when using good tooling like that provided by the GalssFish ESB v2.1.

The service developed in this document will be used as a data provider for the Visual Web JSF Web Application and a Visual Web JSP Portlet in subsequent documents. At the end of the process we will have a SOA 1, Presentation Layer, as well as SOA 3, Business Service (this service) and SOA 4, Technical Layer (the Database BC services) artifacts that are the basis for a part of a healthcare SOA solution.