Building Your First Process with Oracle BPM 11g

This tutorial contains the following sections:

Purpose
Time to Complete
Overview
Scenario
Software and Hardware Requirements
Prerequisites
E Creating the Basic Hello World Application
Enhancing the Basic Hello World Process
E Deploying and Testing the Application
Summary
Resources

Purpose

This tutorial shows you how to build a simple Hello World application using Oracle BPM Suite 11gR1. It also shows you how to deploy the process to the BPM engine and test it in the BPM Workspace.

Time to Complete

Approximately 2 hours

Overview

In this tutorial, you use Studio, the JDeveloper based IDE, to create a simple Hello World process. This process demonstrates the use of a file service, interactive tasks implemented by the human workflow engine, and by conditional branching. The conditions for the conditional branching are determined through the use of a business object and a business rule. You also use a script task to initialize a variable. After building the process, you deploy it to the BPM engine and test it in the runtime environment.

Scenario

There are two roles involved in the Hello World process, the sender of the message, acting in the Requester role, and a reviewer, acting in the Reviewer role. The requester is prompted, through the Request Hello activity, to enter a Hello message, greeting, and a date for the message. After the form is submitted, a business rule is applied to the message content to determine whether the message requires a review, based on the length of the greeting and message fields.

If the message does not require review, the process flows to a script task, which initializes a variable needed by the next task, then the message is sent to the Write Message activity to be written to the file system. If the message requires review, the reviewer is prompted to review the message and either accept or reject it. If the message is rejected, it returns to the Request Hello activity so that the requester can correct the message, otherwise, it goes directly to the Write Message activity for file processing.



Software and Hardware Requirements

In order to perform this tutorial, you must have previously installed Oracle BPM 11gR1 and JDeveloper 11.1.1.3 with both the SOA and BPM extensions. You will also need to have at least one user in the internal LDAP database of your WebLogic server in the OBPM installation in order to map this user to the roles that you define in your Hello World process. You can take care of both of these tasks (installation and seeding of the LDAP database) by performing the <u>Installing Oracle BPM 11g</u> OBE.

Prerequisites

If you have not yet installed OBPM 11gR1, perform the Installing OBPM 11g OBE. Performing this OBE will also seed your LDAP database.

If you already have an OBPM 11gR1 installation, but still wish to seed the Demo Community in the WebLogic server's internal LDAP database, download the <u>zip file</u> containing ANT files needed to perform this task. This is available from OTN as a **SOA 11g Human Workflow** sample code download. You will need to modify some parameters in the ANT build file to match your particular installation. A ReadMe file is included in the zip file to assist you.

Creating the Basic Hello World Application

In this section you create the basic starting point for the Hello World process using the JDeveloper Studio. You add two activities - an interactive activity and a service activity. The end user will be able to enter a Hello message, using the BPM Workspace. The message will be captured in a business object and passed to a file service, which, in turn will write the message to a disk file. Later, you expand upon this to add more complexity to the process.



You create several process elements throughout this section of the tutorial. The following naming convention will be used throughout this section:

Name	Description
HelloWorld_OBE	Application name
HelloWorldProject	Main project name
HelloWorld_UI	Project containing user task web form(s)

HelloWorldProcess Process name

Creating the Process Model Creating the Business Object Implementing the User Task Implementing the File Service

Creating the Process Model

1. Open JDeveloper Studio 11.1.1.3 from the Windows Start menu. When prompted to select a role, choose the Default Role. Click OK.

🕌 Select Role 🛛 🔀
Select the role that matches your requirements. You can also change roles using the Roles page in preferences.
<u>R</u> ole:
O Default Role
Enables all technologies.
Customization Developer Configures the product for customizing metadata.
O Database Edition
Includes only features for core database development.
Java EE Edition
Includes only features for core Java EE development.
🔘 Java Edition
Includes only features for core Java development.
Always prompt for role selection on startup
OK Cancel

Close the Daily Tips window.

2. Create a new application. Click the New Application bar in the left panel.



The BPM Application wizard opens. Name the application "HelloWorld_OBE" and accept the default directory for storing application files (C:\JDeveloper\mywork). Select **BPM Application** in the Application Template panel.

Name your applicatio	
Application Name Project Name Project SOA Settings	Applcation Name: HelloWorld_OBE Directory: C:_Developer\mywcrk\HelloWorld_OBE Applcation Package Prefiz:
	Application Template: Generic Application Creates an application which includes a single project. The project is not preconfigured with JDeveloper technologies, but can be customized to include any technologies.
	BPM Application Creates a BPM application. The application consists of one BPM project. This project has also SOA technology
	E Fusion Web Application (ADF) Creates a databound ADF web application. The application consists of one project for the view and controller components (ADF Faces and ADF Task Flows), and another project for the data model (ADF Business Components).
Help	< Back Next > Einish Cancel

Click Next.

3. In Step 2 of the Create BPM Application wizard, you create a project for the HelloWorld_OBE application. Enter HelloWorldProject as the Project Name. Notice that BPM and SOA are selected as project technologies by default. Click Finish.



In the upper left corner of the JDeveloper Studio window, you see the Navigator panel. This contains two tabs that will be important to you as you perform this tutorial: The **Application Navigator** tab and the **BPM Project Navigator** tab. Currently the Application Navigator tab is selected by default. You can see the HelloWorld_OBE application appearing in the drop-down list just above the panel and the HelloWorldProject appearing as the parent node within the panel. The fact that it appears in *italics* indicates that there are unsaved changes.



Click the Save All icon on the main toolbar.

4. To create a new process within this project, first click the BPM Project Navigator tab. Then right click on Processes and select New > Process .



In the BPM Process wizard, select the From Pattern radio button, and then the Manual Process pattern. Click Next.

BPMN Process:				
Default Process From Pattern Patterns Manual Process Asynchronous Service Synchronous Service	Preview	Start	JserTask	End
Help		< Back	Next > Fi	nish Cancel

In the next screen, name the process "HelloWorldProcess" and click Finish.

BPMN Process:				
General Advanced				
HelloWorldProcess				۲
Id: HelloWorldProcess Description				
				۲
Others				
Author: jmoritz				
Help	< Back	Next >	Finish	Cancel

The process model appears in the design editor panel in the middle of the JDeveloper window. The tab name will be same as the name of your new process.

Click the Save All icon again.

You may wish to close the other tabs, as you will not be using these. An \mathbf{X} will appear in the upper right corner of the tab when your cursor approaches it. The X will close the tabbed pane. All of these can be easily reopened later from either the menu or one of the navigator panels.



5. Change the name of the user task in the design model. Notice that the model begins and ends with two circular icons. The circle on the left is a Start activity and the circle on the right is an End activity. Connecting the two circles is a line that represents the flow of activities through the process. This is called the sequence flow and sometimes is referred to as the "transition line". Between the Start and End activities is a User Task type activity. Right click on this and select Properties.



When the **Properties** dialog box appears, on the **Basic** tab, change the name of the activity to "Request Hello". Click **OK**.

📥 Properti	ies - UserTask	×
Basic Imp	lementation	
Name:	Request Hello	۲
Description:		۲
🗄 Sampling	Point	

Don't worry about the warning message indicating that no implementation has been defined. You will do this later.

Role	Start Request Hello End

Click Save All.

 Add the Component Palette to the JDeveloper window by selecting View > Component Palette from the menu. The palette will appear in the right pane of the window.

Select **BPM** from the drop-down list at the top of the Component Palette, then expand the **Activities** accordion panel as shown below.



7. Add a service activity to the process. You'll need to first make room for another activity on the sequence flow.

Click on the **End** activity and drag it to the right, dropping it on the right side of the design panel, allowing enough room for another activity icon to fit between the **Request Hello** activity and the **End** activity.



Now click the **Service** activity icon in the Component Palette and drag it to the Sequence flow between **Request** Hello and End. Drop it there. Notice that the transition line turns blue when the drop target area approaches the line.

Important: The transition line *must be blue* when you drop the object in order for the transition line to be connected to the activity.



When you drop it, the **Properties** dialog box for the activity opens. On the **Basic** tab, change the activity name to "write Message".

Properties - ServiceTask	
Basic Implementation	
Name: Write Message	٢
Description:	٢

Click OK.

8. Click Save All. Your process model should now look similar to this.



Creating the Business Object

 Now you will create a business object capable of storing multiple pieces of data, related to the message that the user enters in the Request Hello activity. This object will be populated when the user enters the message. It will then be passed to the Write Message activity so that the message can be written to a file.

Business objects are stored in *modules* within the Business Catalog. In the **BPM Project Navigator**, expand the **HelloWorldProject** node. Right click on **Business Catalog** and select **New > Module**.



When prompted to name the new module, enter "HelloTypes" and click OK.

The HelloTypes module now appears beneath the Business Catalog node.



2. Right click the HelloTypes module and select New > Business Object.

Business Catalo	og	Role	
HumanTas HumanTas Reference	Sew New Delete ★ Delete Reload	Module Business Object Business Exception	art
Types Simulations Resources			_

In the Create Business Object window, enter "HelloObject" as the Name and accept HelloTypes as the Destination Module. Click OK.

Create Business Object		×
Business Object		
Name:	HelloObject	
Destination Module:	HelloTypes	
Based on External Schema		2
Help	OK	

The HelloObject editor now opens in the editor.

3. Add the following three attributes to HelloObject:

Attribute Name	Туре
date	Time
greeting	String
message	String

The following instructions describe how to create the date attribute.

Click the plus sign next to the Attributes section of the Business Object editor as shown below:

HelloWorldProces	s 🔀 HelloObject	
Business Object		<u></u>
Description:		
Documentation:	Edit	
🗄 Details		
Attributes		4

The **Create Attribute** popup appears. Enter "date" as the Name value. Select **Time** as the Type from the drop-down list. Click **OK**.

🍲 Create Attribute	
Name: date	
Type: Time	•
Help	OK Cancel

The date attribute now appears in the Attributes section of the Business Object editor.

Continue working in this way to create the other two attributes. When you finish, the Business Object editor should look like this:

HelloWorldProcess	HelloObject	
Business Object		
Description:		
Documentation: Edit		
🗉 Details		
Attributes		4
Time date Time da		× 🕆 🕹
String greeting		🗙 🔂 🦆
∃ String message		🗙 🔂 🤴

Click Save All and close the HelloObject tab in the editor panel.

4. Now create a process data object of type HelloObject so that you can use it in your process.

If the **HelloWorldProcess** tab is still open in the editor panel, click anywhere in the design editor to put the focus on the HelloWorldProcess. (If it is not open, select it within the BPM Project navigator by expanding **HelloWorldProject** > **Processes** to find it)

When a process has been given focus, a detailed outline of its structure appears in the **Structure** pane in the lower left corner of the JDeveloper window.

HelloWorldProcess - Structure	
Activities Activities Business Indicators Measurements Tim Process Data Objects Project Data Objects	

Right click on Process Data Objects in the Structure pane and select New.



In the **Create Data Object** popup, enter "hello" as the Name and click the ellipses button to open another window to search for complex data types.

🍲 Cre	ate Data Object	
Name:	hello	
Type:	String	
🛃 Auto	o initialize	
Help		OK Cancel

In the Browse Types window, select <Component> as the Type and then select HelloObject from the list of components appearing below. Click OK.

े Browse Types		X
Type: <component></component>	•	
Find:		[₽
👼 HelloObject		
HelloTypes.HelloObject:	2	
Help	OK Cance	el

Back in the Create Data Object window, click OK again. The hello data object now appears in the Structure pane.



Click Save All.

Implementing the User Task

1. Every interactive activity must be bound to a task service to provide its implementation. In the case of a User Task (such as Request Hello), it must be bound to a *Human Task* type task service. You will create the Human Task in this step.

Right click the Request Hello activity in the design editor and select Properties to open the Properties window.

Click the Implementation tab. Next to the Human Task field, click the plus sign button as shown below.

🖕 Properties - Request Hello			×
Basic Implementation			
Implementation Type: [User task			-
Human Task:	4	م 🂊	
Human Task Attributes			

In the Create Human Task dialog box, enter or select the following values for fields in the top portion of the dialog:

Field	Value	
Name	SayHello	
Pattern	Initiator	
Value	Please Enter a Hello Message	
Outcomes	Submit <this auto="" choose="" for="" is="" pattern="" selected="" the="" when="" you=""></this>	

Add a parameter by clicking the plus icon above and to the right of the Parameters panel.

🍲 Create H	luman T	ask		
General				
Name:	SayHello	1		Priority: 3 (normal) 💌
Pattern:	😰 Initi	ator		•
Title:	Please E	nter a Hello Message		
Outcomes:	SUBMIT			۹.
Parameters:				+ ×
Parameter		Name	Туре	Editable
Outcome targ	get:			+ 🧳
Help				OK Cancel

This opens the Data Object window displaying available data objects that you can drag into the Parameters panel. Click on the **hello** data object and drag it into the parameters panel. Select the **Editable** checkbox for the new parameter.

े Create H	luman Task			🎐 Data Object 🛛 🔀
General				Drag Data Objects to Parameters table and Outcome Tarnet field
Name:	SayHello	F	Priority: 3 (normal) 🔻	
Pattern:	🔁 Initiator		-	
Title:	Please Enter a Hello Mes	sage		
Outcomes:	SUBMIT		Q	HelloWorldProject
Parameters:			× + ×	
Parameter	Name	Түре	Editable 📕	
HelloObject	hello	HelpTypes.HeloObj	. 🔽	
Outcome targ	jet:			
Help			CK Cancel	
				Help

Close the Data Object window and then click OK in the Create Human Task window.

You are returned to the Properties window for the Request Hello activity. Click $\ensuremath{\text{OK}}$.

🖕 Properties - Request Hello	
Basic Implementation	
Implementation Type: [🔄 User task	
Human Task: 🔯 SayHello	
Human Task Attributes	
Title: Plain Text 🔻	
Priority: 3 (normal) 💌	
Re initiate	
Advanced	
-Data Associations	
Use Associations Type: Simple V	
Use Transformations	
Help	OK Cancel

Click Save All.

2. Now you must provide a form for the user to enter the Hello message, ensuring that the form is linked to the hello data object.

In the BPM Project Navigator, expand **Business Catalog > Human Tasks**. Here you see the **SayHello.task** object. This is the human task that you just defined.



Double click it to open it in the editor. When it opens, click the Create Form drop-down list in the editor toolbar on the left and select Auto-Generate Task Form ...

😴 HelloWorldProcess	ayHello.task	
🧏 🔮 Create Form 🔻	R	
Auto-Generat	e Task Form	
Launch Task F	Task little	Plain Text Please Enter a Hello Message
酇 Assignment	Description:	
🅮 Presentation		
🚳 Deadlines	Outcomes:	SUBMIT
🦫 Notification	Priority:	3 (Normal) 💌
🐍 Access	Category:	By expression 👻
🗲 Events	Owner	Application Role HelloWorldProject.ProcessOwner Static
	Application Context:	OracleBPMProcessRolesApp

The **Create Project** window opens. It is necessary to create a separate project to contain UI elements (ADF forms). Enter HelloWorld_UI as the Project Name and accept the default directory. Click **OK** when finished.

💩 Create Project					
Enter the name and directory of the new project. Use projects to organize your files.					
Project Name: HelloWorld_UI					
Directory:					
Developer\mywork\Hello	oWorld_OBE\HelloWorld_UI	Browse			
Help	ОК	Cancel			

It will take several seconds (even up to a minute, depending on resources) to create the form and open the necessary editor. Eventually, you will see the editor shown below (partial view). The highlighted section is the portion that will be visible to the end user. Notice the **Date**, **Greeting**, and **Message** fields.

2 ???DETAILS??? 🕕		toolba
<pre>??? #(displayName) ASSIGNEES??? #(displayName) ???CREATOR??? #(displayName) ??? #(displayName) ??? #(displayName) ??? #(displayName) ??? #(displayName) UPDATE_DATE??? inputValue)</pre>	<pre>??? #(.expiratiorDate. EXPIRATION_DATE??? inputValue) ???ACQUIRED_BY?? #(acquiredEy. inputValue) ???DUE_DATE??? #(dueDete. inputValue) #(???OUTCOME??? actonDisplayName. inputValue)</pre>	? TASK_NUMBER? ???PRIORITY? ???STATE?
???CONTENTS???		

Select the **Message** field as shown above. When you do so, the properties for this field appear in a panel in the lower right corner of the JDeveloper window. Expand the **Appearance** accordion panel within this panel and change the value of the **Rows** property to **5**.

😋 Input Tex	t - #	{binding	s.message.hints.la		
🖳 I 🏓 😫	10	/ 🖪 🤇	🎁 Find 🕹 🖓	D	
🗆 Common					
• Id:	• Id: it2]~	
Rendered:	<de< td=""><td>efault>(tri</td><td>ue) 🗸</td><td>]~</td></de<>	efault>(tri	ue) 🗸]~	
Label:	Label: #{b		ndings.message.hints.label}		
🗊 Value:	#{bindings.message.inputValue}				
🖃 Appearanc	e				
Columns:		#{binding	s.message.hints.display]~	
Rows:		5]~	
Secret:		<default>(false)</default>			
Wrap:		<default></default>]~	
ShowRequir	ed:	<default> (false)</default>]~	
Channed:		cdefault	> (false) 🔹 🔻	l~	

Tab out of the field so that the change will take effect in the form.

Date	#{date.inputValue}	
Greeting	#{greeting.inputValue}	
Message	#{message.inputValue}	^
		×

 $\label{eq:click} \mbox{Save All. Close all tabs in the editor except for the $\mbox{HelloWorldProcess}$ tab. $\mbox{}$

Implementing the File Service

 Create the implementation for the Write Message service activity using the SOA Composite editor. Click the Goto Composite Editor button on the Design Editor toolbar as shown below.

Goto	Composite Editor	
STAR A	0	

The SOA Composite editor opens. To view the design editor, rather than the XML source, click the **Design** tab at the bottom left margin of the Composite Editor panel. The HelloWorldProcess BPMN component and the SayHello human task component are shown in the composite editor. These are considered components of the SOA composite.

	olo composite.xml	•
4 4 5 12 12 12	(1) 🛛 🖻 🏦 🏷 👘 🥠	Composite: SOAComposite1
	HelloWorldP	SayHello
ए		~
Design Source History		*

2. Click the File Adapter service adapter from the Component Palette. Notice that the Component Palette is now showing SOA components, by default.



Drag and drop the File Adapter into the External Reference column of the editor.



The Adapter Configuration Wizard opens when you drop it.

3. Click Next on the Welcome page of the Adapter Configuration Wizard.

On the Service Name page of the wizard, name the service MessageWriter. Click Next.

Service Name	· · · · · · · · · · · · · · · · · · ·
Enter a Service Name.	
Service Type: File Adapter	
Service Name: MessageWriter	

On the Adapter Interface page, select Define from operation and schema (specified later) . Click Next.

Adapter Interface		5
The adapter interface is this wizard. Optionally, I	defined by a wsdl that is generated using the operation name and schema(s) specified the adapter interface may be defined by importing an existing WSDL.	d later in
Interface 💿 Define fro	om operation and schema (specified later)	
◯ <u>I</u> mport an	n existing WSDL	
<u>W</u> SDL URL	L:	1
<u>P</u> ort Type	81	-
<u>O</u> peration	n:	•

On the Operation page, select Write File. The Operation Name will be pre-populated with the name write. Accept this value and click Next.

Operation								*	5
The File Adapt system, a Writ contents of a f Operation Nam	er supports l e File operat ile, and a Lis ne. Only one	four operations, ition that creates ist Files operation e operation per A	There is a outgoing fi that lists fi dapter Ser	Read File iles, a Syr ile names vice may	operation achronous in specified be defined	that polls fo Read File op I locations. I using this v	or incoming fil peration that Specify the (vizard,	es in your lo reads the cu Operation ty	cal file irrent pe and
Operation Typ	e: <u>R</u> ead	d File e File thronous Read Fil Files	e						
Operation Nam	ne: Write								

On the File Configuration page, select Physical Path as the Directory specified. Enter a dot ('.') for the Directory for Outgoing Files (physical path) field. Also enter a File Naming Convention of:

Hello_%SEQ%.xml

File Configuration							
Specify the parameters for the Write File operation.							
Directory specified as <u>Physical Path</u> <u>Logical N</u> Directory for Outgoing Files (physical path):	ame Browse						
Eile Naming Convention (po_%SEQ%.txt): Hello_%SEQ%.xml							
Append to existing file							
Write to output file when any of these conditions are met-							
Number of Messages Equals: 1							
Elapsed Time Exceeds:	minutes 🔻						
File Size Exceeds: 1000	🗧 kilobytes 💌						

Click Next.

In the Messages page, you determine *what* should be written to the file. Click the magnifying glass icon next to the URL field to open the Type Chooser popup. Expand **Project Schema Files** and **HelloObject.xsd** to find and select **HelloObject**. as shown below. Click **OK** to accept the selection and return to the Messages window of the wizard.



Click Next in the Messages window.

Messages	01010101010101010	
Define the message defines the messag 'Schema is Opaque',	e for the Write File operation. Specify the Schema File Location and ges in the outgoing files. Use the Browse button to find an existing s ', then you do not need to specify a Schema.	select the Schema Element that chema definition. If you check
-Message Schema-		
Native <u>f</u> ormat t	translation is not required (Schema is Opaque)	
URL	businessCatalog/HelloTypes/HelloObject.xsd	Q 🔅
Schema Element	HelloObject	-

Click Finish in the final page of the wizard. The service is created and appears in the SOA Composite editor.

Click Save All and close the Composite editor tab.

4. Now you must wire the service implementation you just created to the Write Message activity in the BPM process. You need to be in the BPM Process design editor for this.

Note: If necessary, open the HelloWorldProcess in the design editor by double clicking on it in the BPM Project Navigator.

Right click on the Write Message activity and select Properties.



In the Properties window, click the Implementation tab.

Select Service Call as the Implementation. Click the magnifying glass icon next to the Name field to browse for a service. The Type popup window appears, displaying your MessageWriter service. Select it and click OK.

😂 Properties - Write Message	X
Basic Implementation	
Implementation Type: 🔯 Service task	-
Service task	-
Implementation: 🔯 Service Call	
Name:	
Operation:	
Data Associ 👌 Turco	
Use As Search:	
/Services/Externals/MessageWriter:	
Help	

Back in the Properties window, notice that the Operation field has now been automatically set to write.

In the **Data Associations** panel, click the **Use Associations** checkbox, then click the pencil icon next to it. This opens an editor that allows you map data into the activity. In this case, you want to map the *hello* object into the activity so that the MessageWriter service can write the message.

In the **Data Associations** window that opens next, drag the *hello* data object from the right column over to the **Inputs** field on the left side of the window. Notice that this maps the *hello* data object to the *helloObject* expected as input to the activity's service implementation.

🚔 Data Associations	X
	📲 🖓 🔚 🛛 📼 🚥
Drag variables from the right panel into fields on the left.	HelloWorldProcess
Help	OK Cancel

Click OK in the Data Associations window to save the mapping and return to the Properties window.

Click **OK** in the Properties window.

Click Save All.

Enhancing the Basic Hello World Process

In this section, you enhance the basic process you just built by adding a review capability for the message entered by the user. You add another business object to hold review-

related data and use this business object in conjunction with a business rule that tests the length of the greeting and message. You also add another human interaction that allows the user with the Reviewer role to review the entered message and accept or reject the message. Then you change the process model, itself, so that it conditionally branches to the Review Message activity and potentially back to the Request Hello activity if the message was rejected. A script task will also be needed in order to initialize a variable used in the conditional branching logic.

You begin the enhancement by changing the name of the role assigned to the user who executes the Request Hello activity to something more meaningful. Then you add a new role for the user who reviews the message.

Adding the ReviewNeeded Business Object Adding a Business Rule Defining a Decision Table for the Business Rule Adding the Review Message Human Interaction Adding Conditional Branching Adding a Script Task

Adding the ReviewNeeded Business Object

1. Change the name of the Role role. In the BPM Navigator, right click on the Organization node beneath HelloWorldProject and select Open.

Application	🕲 BPM Projec	t	6	_
HelloWorld HelloWorld Hereit Proces Activit Hereit Activit Hereit Activit	Project sses y Guide ization isss Ca A Open tions rces	- Co		

The Organization editor opens in the center panel. Select the **Role** role and click the pencil icon to edit it.

B	Drganization	
	Roles	î
	Name	4
	Process Owner	
	Role	
		Edit

The Role popup window appears. Enter the name Requester and click OK.

Role	×
Name:	Requester
Help	OK Cancel

2. With the Organization editor still open, click the green plus sign button to add a new role.

-Roles	
Name	ا چ ۲
Process Owner	
Requester	Add

Enter the name Reviewer in the Name field of the Role popup window and click OK.

Your Organization editor should now look like this:

1 -

-Koles		
Name		4
Process Owner	4	1
Requester	•	se la compañía de la
Reviewer	~	~

- 3. Click Save All and close the Organization editor.
- 4. Add a new business object in the same manner that you did in the Creating the Business Object section above. Follow Steps 2 and 3 of that section, placing the new business object in the HelloTypes module and substituting the name and attributes mentioned below.

Name the object ReviewObject and add the following attributes to it:

Attribute Name	Туре
review	String
reason	String

	ReviewObject			
Business Object		1		
Description:				
Documentation:	Edit			
🗉 Details				
Attributes				
			×	Ŷ
				~
 String reason 			×	Τ

Click Save All and close the object editor. The ReviewObject should now appear in the BPM Navigator beneath the HelloTypes folder.

 Declare a process data object of type ReviewObject in the same manner that you did in the Creating the Business Object section above. Follow the instructions in Step 4, substituting review as the data object name.

6. Add one more process data object of type String, called reviewOutcome. Follow the same procedure as the previous step, except choose String as the type.

🌢 Create Data Object	×
Name: reviewOutcome	
Type: String	▼
🖌 Auto initialize	
Help	OK Cancel

Adding a Business Rule

1. Add a business rule to the design model.

Begin by making more room on the sequence flow between the Request Hello and Write Message activities.Move the End activity and the Write Message activity further to the right.

Expand the Activities accordion panel in the Component palette, then click and drag a Business Rule into the design editor, dropping it onto the sequence flow between Request Hello and Write Message.

The Properties dialog for the business rule appears when you drop the object. Enter **ReviewNeeded** in the **Name** field. Click **OK**.

🍲 Properti	es - BusinessRuleTask	×				
Basic Implementation						
Name:	ReviewNeeded	۲				
Description:		۲				
± Sampling	Point					
<	<i>i</i>	>				
Help	ОК Сал	el				

Click Save All

2. In the next step, you will define the implementation of the ReviewNeeded business rule using the Composite editor. In order to make the two business objects available to the Composite editor, you must copy the xsd files that were generated for each object when you created them from the $\texttt{businessCatalog} \\ \texttt{HelloTypes}$ folder to the xsdfolder within the directory structure of your project. [This is a workaround to a known bug in this release of the product

Open Windows File Explorer and navigate to:

C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorldProject\businessCatalog\HelloTypes

In this folder, find and copy the following two xsd files:

HelloObject.xsd ReviewObject.xsd

🖽 🚞 src

표 🚞 Testing

Paste them into the following folder:

C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorldProject\xsd

3. Open the Composite editor by clicking the Goto Composite Editor button on the Design editor toolbar.

NelloObject.xsd

4. Add a Business Rule component to the Composite Editor. From the Component palette panel on the right, click and drag a Business Rule from the Service Components section of the palette and drop it in the Components column of the Composite Editor.

When you drop the component, the Properties dialog appears. Enter HelloRules in the Name field. Then click the green plus button above the Inputs/Outputs section and select Input...

Create	Business Rules			×
Business R	tule			
A busine: assert bu	ss rule defines or cor Jsiness structure or i	nstrains one aspect o nfluence the behavio	of your business that is i or of your business.	intended to
General	Advanced			1
⊙ <u>C</u> reate	e Dictionary 🔘 Imp	ort Dictionary		
Specify th	ne name and package	e for the dictionary t	hat will be created.	
<u>N</u> ame:	HelloRules			
<u>P</u> ackage:	helloworldproject			
Project:	C:\JDeveloper\myw	vork\HelloWorld_OBE	\HelloWorldProject\Hell	oWorldProject.jpr
Inputs/Ou	utputs:			 ×
Direction	1	Name	Туре	Input
				🗢 Output.!.
Help				OK Cancel

The **Type Chooser** popup window appears. Expand the second **HelloObject.xsd** entry, then select the **HelloObject** beneath it and click **OK**.

Click the green plus button once more and this time select **Output...** When the Type Choose popup window appears, expand the second **ReviewObject.xsd** entry and select **ReviewObject**. Click **OK**.

3

Type CI					
					- 2 0
	plorer ject Schem HelloObje ReviewOb ReviewOb Plocument HelloObje ReviewOb SayHelloP SayHelloP TaskState	a Files tt.xsd ject.xsd Package.xsd tt.xsd ject.xsd ewObject ewObject wObjectType ayload.xsd VorkflowTask.xsd Machine.xsd	2	Δ.	
vpe: {http	://xmlns.o	racle.com/bpm/b	pmobject/HelloTyp	es/ReviewObject}Rev	viewObject
Show De	tailed Node	e Information	window, click OI	OK .	Cancel
Show De Help ck in the E usiness R A busines assert bu	tailed Node Susiness F ule is rule defii siness stru	e Information Rule properties nes or constrains icture or influence	window, click O one aspect of you e the behavior of	OK ur business that is inter your business.	Cancel
Show De Help ck in the E usiness R A busines assert bu	tailed Node Business F ule is rule defii siness stru Advance	e Information Rule properties nes or constrains icture or influence	window, click Oł one aspect of you e the behavior of	OK ur business that is inte your business.	Cancel
Show De Help ck in the E usiness R A busines assert bu General	tailed Node Business F ule Siness stru Advance	e Information Rule properties nes or constrains icture or influence d	window, click O one aspect of you e the behavior of	OK ur business that is inter your business.	Cancel
Specify th	tailed Node Business F ule Is rule defi siness stru Advance Dictionary e name an	e Information Rule properties nes or constrains icture or influence d /Import Dicti d package for th	window, click O one aspect of you e the behavior of ionary e dictionary that v	OK C. ur business that is inter your business.	Cancel
Show De Help ck in the E usiness R A busines assert bu General © Greate Specify th Name:	tailed Node Business F ule :s rule defii siness stru Advance : Dictionary e name an HelloRule:	e Information Rule properties nes or constrains icture or influence id /Import Dicti g package for th ;	window, click O one aspect of you e the behavior of ionary e dictionary that v	OK ur business that is inter your business. will be created,	Cancel
Show De Help ck in the E usiness R A busines assert bu General O Greate Specify th Name: Parkane:	tailed Node Business F ule :s rule defii siness stru Advance : Dictionary e name an HelloRules	e Information Rule properties nes or constrains cture or influence d / _ Import Dicti d package for th ; Inroject	window, click O one aspect of you e the behavior of ionary e dictionary that v	OK ur business that is inter your business. vill be created.	Cancel
Show De Help ck in the E asiness R A busines assert bu General O Create Specify th Name: Package:	tailed Node Business F ule is rule defii siness stru Advance > Dictionary e name an HelloRules hellowork	e Information Rule properties nes or constrains icture or influence id / _ Import Dict d package for th ; project	window, click O one aspect of you e the behavior of ionary e dictionary that v	OK ur business that is inter your business. will be created.	Cancel
Spow De Help ck in the E usiness R A busines assert bu General O greate Specify th Name: Package: Project:	tailed Node Business F ule is rule defii siness stru Advance > Dictionary e name an HelloRules helloworlc C:\JDevel	e Information Rule properties nes or constrains icture or influence id project project oper\mywork\He	window, click O one aspect of you e the behavior of ionary e dictionary that v	OK ur business that is inter your business. vill be created. oWorldProject\HelloW	Cancel nded to
Spour De Help ck in the E usiness R A busines assert bu General O greate Specify th Name: Package: Project: Inputs/OU	tailed Node Business F ule :s rule defii siness stru Advance : Dictionary e name an HelloRule: helloworlc C:\JDevel tputs:	e Information Rule properties nes or constrains icture or influence id project project oper\mywork\He	window, click O one aspect of you e the behavior of ionary e dictionary that v	OK A ur business that is inter your business. will be created. oWorldProject\HelloW	Cancel
Spour De Help ck in the E usiness R A busines assert bu General O greate Specify th Name: Package: Project: Inputs/Ou Direction	tailed Node Business F ule :s rule defii siness stru Advance : Dictionary e name an HelloRule: hellowork C:(JDevel tputs:	e Information Rule properties nes or constrains icture or influence id project project Name	window, click O one aspect of you e the behavior of ionary e dictionary that v lloWorld_OBE\Hell	OK A ur business that is inter your business. will be created. oWorldProject\HelloW	Cancel
Spour De Help ck in the E usiness R A busines assert bu General © Greate Specify th Name: Project: Inputs/Ou Direction Input Output	tailed Node Business F ule is rule defii siness stru Advance Dictionary e name an HelloRules helloworld C:\JDevel tputs:	e Information Rule properties nes or constrains icture or influence id / _ Import Dict d package for th s lproject oper\mywork\He Name HelloObject ReviewObject	window, click O one aspect of you e the behavior of ionary e dictionary that v lloWorld_OBE\Hell Type {http://xmlns {http://xmlns	OK A A A A A A A A A A A A A	Cancel
Specify th Name: Project: Inputs/OU Direction Input Output	alied Node Business F ule srule defii siness stru Advance Dictionary e name an HelloRules hellowork C:\JDevel itputs: e as Comp	e Information Rule properties nes or constrains cture or influence d / _ Import Dicti d package for th f Iproject oper\mywork\He HelloObject ReviewObject osite Service	window, click O one aspect of you e the behavior of ionary e dictionary that v lloWorld_OBE\Hell Type {http://xmlns {http://xmlns	OK A ur business that is inter your business. will be created. will be created. oWorldProject\HelloW a coracle.com/bpm/bpm/ pm/bpm/bpm/	Cancel
Spow De Help ck in the E usiness R A busines assert bu General © Greate Specify th Name: Project: Inputs/Ou Direction Input Output	tailed Node Business F ule :s rule defii siness stru Advance : Dictionary e name an HelloRules hellowork C:\JDevel itputs: e as Comp	e Information Rule properties Rule properties nes or constrains icture or influence id / _ Import Dict d package for th s Iproject oper\mywork\He HelloObject ReviewObject osite Service	window, click O one aspect of you e the behavior of ionary e dictionary that v lloWorld_OBE\Hell Type {http://xmlns {http://xmlns	OK A A A A A A A A A A A A A	Cancel

Click Save All.

 You now need to wire the implementation that you just created (the HelloRules business rule) to the ReviewNeeded business rule activity in the BPM process. Click the HelloWorldProcess tab to bring the focus back to the BPM Design Editor.

Right click the **ReviewNeeded** activity to open its properties window. Click the browse button to the right of the **Business Rule** field.

Properties - ReviewNeeded						
Basic Implementation						
Implementation Type: 🔣 Business Rule task						
Business Rule: 🔶 🔶						
Decision Function:						
Data Associations						
🗌 Use Associations Type: Simple 💌 🥒						
Use Transformations						

When the Browse Business Rules window opens, you should now see your ${\rm HelloRules}$ business rule implementation. Select it and click ${\rm OK}.$

🕹 Browse Business Rules	
Search:	
HelloRules	
Help	OK Cancel

 In the Properties window, click the checkbox next to Use Associations. Keep the default setting for Type, which is Simple associations and click the Edit Data Associations button (the pencil icon).

鏲 Properties - Rev	iewNee de d	×
Basic Implementatio	n	
Implementation Type:	쬃 Business Rule task	-
Business Rule:	HelloRules 🕂 🗣	4
Decision Function:	HelloRules_DecisionService_1	-
Data Associations		
Use Associations	Type: Simple 🗸 🥢	
Use Transforma	tions Edit Data Associations	

The Data Associations window opens. In this window, you map process data objects (hello and review) to the Inputs and Outputs that you defined for the HelloRules business rule implementation. From the right column, click and drag the review data object, dropping it into the **Outputs** field as shown below.

Now click and drag the hello data object from the right column, dropping it on the Inputs field. Click OK to save this mapping and close the Data Associations window.

🕹 Data Associations		
× 1		
Drag variables from the right panel into field	s on the left.	
Inputs	ReviewNeeded	Outputs
		····· > review
		R

Click **OK** in the Properties window. Click **Save All**.

Defining a Decision Table for the Business Rule

In this section you define a set of rules that will be applied when the user enters a hello message and greeting. The following table illustrates what those rules are and how they will be applied.

IF	Length is						
greeting:	Short	Medium	Medium	Long			
message:	(anything)	Short	Medium or Long	(anything)			
ReviewObject:	ReviewObject:		Ļ	ļ			
reason: "Greeting is too short"		"Length is too short"	null	null			
review:	"true"	"true"	"false"	"false"			

The lengths of the HelloObject.greeting and HelloObject.message strings will be evaluated. You will define four rules that determine the combination of greeting and message lengths and then set the appropriate values to the ReviewObject properties: reason and review. For example, if the greeting length is Medium and message length is Short, the ReviewObject's reason property would be set to "Length is too short" and its review property would be set to "true". Consequently, the process would flow to a ReviewNeeded activity and could be accepted or rejected by the reviewer.

A decision table can be designed in many different ways. In our scenario, the decision table consists of four elements as described here:

A set of Conditions - In this case, each condition is the length of either the greeting or the message string variable

A Range of Values, called a bucketset - For example a variable might be < 5 characters in length, or between 5 and 50 characters in length, and so on. These are ranges within a bucketset.

An Action - This is the act that should occur when a rule is evaluated. For example, the review property might be set to "true".

A Rule - This is a specific mapping of condition > range > action.

The Decision table is created in five high level steps:

- 1. Define the conditions (Step 2 below)
- Define a bucketset (Step 3)
 Assign that bucketset to each condition (Step 4)
- 4. Define the action (Step 5) 5. Define the rules (Steps 6 - 9)
- 1. Open the Rules Editor .

In the BPM Navigator, expand Business Catalog > Rules > Helloworldproject. Double click on HelloRules.rules.

The Rules Editor opens and, by default, the Ruleset 1 node in the left panel of the editor is selected.

2. Create a decision table for the rule.

Click the green plus sign button on the toolbar and select Create Decision Table from the drop-down menu.

9	Ruleset_1 🛛 🛛 Eilter On View: 🍑 IF/	/THEN 🕶 💠 🔽 🔀 🖶 🖶 60 😞	\otimes
т	be ruleset contains no IE/THEN rules. You can creat	te Rules or De	
0	orresponding buttons below or clicking the plus sign	above in the	
Γ		Paste Decision Table	
	IF/THEN Rule	Decis 🛅 Paste Rule	
	You create business rules to process facts and to obtain intermediate conclusions that	A Decision Table displays multiple related rules in a single spreadsheet-style view. In	

The Ruleset_1 editor changes to provide a table for you to define the decision table. Change the default name of the decision table from DecisionTable_1 to ReviewDecisionTable. Click on the field containing the default name. A text field appears below it. Enter the new name and hit Enter to accept the value.

± Ru	leset_1 View: 🔯 DecisionTable_1	+ + × ×
* ®	DecisionTable_1 <enter description=""></enter>	
	ReviewDecisionTable	🔜 🛛 🔂 🔛 🔜 🖏 🖏
	Press the Enter key to save changes	.::

Add two conditions to be checked for the table:

HelloObject.greeting.length() HelloObject.message.length()

Add the first condition by clicking on the **<insert_condition>** field in the table as shown below. This will automatically add a condition row, with a link that says **<edit_condition>**.

Right click on the **<edit_condition>** field and select **Edit Condition**. A drop-down list appears, displaying the defined data object types within the project. Expand **HelloObjectType** and select **greeting**.

Right click on HelloObjectType.greeting and select Edit Condition again.

This time, the drop-down list that appears presents more options. Expand HelloObjectType > greeting and select the length() function.

Add the second condition by clicking the green plus sign button directly above the Conditions table and selecting **Condition** from the drop-down menu.

Right click on the newly added condition as you did for the first condition and select Edit Condition. This time, you

can go directly to adding the length() function as part of the condition. Expand HelloObjectType > message and select length().

•		
UI	Heliocojec(Type.greeting.jen)	deller
C2	<edit condition=""></edit>	
		F x
4	Value Options	<u>^</u>
Þ	HelloObjectType	
	🗄 📲 date	
	🛓 🚥 greeting	
	🛓 📲 message	
	elength()	
	toL(^h prCase()	
	toUpperCase()	
	□ trim()	

Click Save All.

3. The rule will compare each condition against a range of values, called a "bucketset". Click the **Bucketset** node in the left panel of the Rules editor to create the bucketset.

Click the green plus sign button and select List of Ranges from the drop-down menu.

A new bucketset row is inserted. With this row selected, click the pencil icon to edit the list of ranges.

🐼 Bucketsets										
B <u>u</u> ck	etsets:			🔞 i 🕂 - 🏹 🗙						
	Name	Datatype	Form	Description						
13	Bucketset_1	int	Range							

In the Edit Bucketset window that appears, change the value of the **Name** field to **Lengths**. Leave the Data Type field set to int.

The Range Bucket Values table has an initial default row in which the **Endpoint** is set to **-Infinity**. Click the green plus sign button twice more to add two more rows.

ا 🔶	Edit Bucketset - Bucketset_1								
N	Name: Lengths Data Type: int								
R	Include D	visallowed Buckets in	n Tests			* ×			
F	Endpoint	Included Endpoin!	Allowed in Actions	Range	Alias	Description			
	50 ✓ ✓ >=50 >=50								
	0	 Image: A set of the set of the	~	[050)	[050)				
	Infinity	✓	✓	<0	<0				

Change the 0 (zero) **Endpoint** value to 5 by clicking on the value and typing the new value. Notice that this changes the Range for this row to [5..50] and also changes the Range of the row whose endpoint is **'Infinity** to <**5**.

Change the Alias for each row to short, Medium, and Long as shown below. Double clicking on the default Alias value makes it editable.

I	Range	+ ×					
		Endpoint	Included Endpoin	Allowed in Action	Range	Alias	Description
		50	~	Image: A start of the start	>=50	Long	
		5	~	>	[550)	Medium	
		-Infinity	~	~	<5	Short	

Click \mathbf{OK} to close the Edit Bucketset window.

Click Save

Assign the Lengths range of values bucketset to each of the conditions in your conditions table. Click the Ruleset_1
node in the Ruleset editor to bring back the decision table editor.

Click the C1 condition to select it and then click the Local List of Ranges drop-down list above the Conditions table and select Lengths.

Do the same thing for the C2 condition.

ReviewDecisionTable <enter description=""></enter>		
C2 R1: ?		
• <u>C</u> onditions		
C1 HelloObjectType.greeting.length()		
C2 HelloObjectType.message.length()		

 You are almost ready to start defining the rules to be applied to each combination of range of values and condition. First you must define an Action to be performed when a rule is implemented.

Create a new Action. In the Actions panel, click the <insert_action> field and select Assert New.

Once the assert new () line appears, right click it and select Edit.

The Action Editor window opens. Select ReviewObjectType from the Facts panel. The properties of this object type now appear in the Properties panel in the lower portion of the window. Check the Parmeterized and Constant checkboxes for both the reason and the review properties.

Action Edit	pr			
Form: Assert	New			-
Value: Assert	New ReviewObject	Type (reason:?, rev	view:?)	
Eacts:				
Properties:				
Property	Туре	Value	Parameterized	Constant
reason	String		Image: A start of the start	~
review	String		Image:	 Image: A set of the set of the
Al <u>w</u> ays Sele	ected			
Help				OK Cancel

Notice that the effect of this is that an **assert new(**) statement is being constructed in the Value field. It instantiates a ReviewObject with values for the **reason** property (a String), and the **review** property (a String) You can see this assert statement in the screenshot above. Each rule that you define in the next step will provide specific values for these parameters.

Click OK, then Save

 Define the first rule. Click on the C1 row beneath the R1 column. A drop-down list appears representing the bucketset you defined for this ruleset. Check Short and click OK.

-	<u>C</u> onditions	R1
C1	HelloObjectType.greeting.length()	ζ
C2	HelloObjectType.message.length()	Short
		Medium Long
×	Conflict <u>R</u> esolution	OK Cancel

Click the R1 column for the C2 row. This is the length for the message property. In this case, it doesn't matter what the length is, so enter a dash (-) and click OK.

•	<u>C</u> onditions	R1
C1	HelloObjectType.greeting.length()	Short
C2	HelloObjectType.message.length()	2
×	Conflict <u>R</u> esolution	Short Medium Long OK Cancel

In the **Actions** panel, click the checkbox in the column to the right of the **assert new** statement, indicating that you do want the ReviewObject asserted for this rule. Then click the row beneath the checkbox, next to the **reason** property. When the text field appears, type: "Greeting is too short". Hit Enter when finished.

•	Acti <u>o</u> ns			
Al	assert new ReviewObiectType(
	reason:String			
	review:String)	"Greeting is too short"	Ι	_f_
		No SDK Option Matches Found!		

Click the column next to the review property and enter the value "true" into the text field. Hit Enter when finished.

•	<u>C</u> onditions	R1	
C1	HelloObjectType.greeting.length()	Short	
C2	HelloObjectType.message.length()	-	
×	Conflict Resolution		
-	Acti <u>o</u> ns		
Al	assert new ReviewObjectType(✓	
	reason:String	"Greeting is too short"	
	review:String)	"true"	

You have just defined **Rule 1**, indicating that if the *greeting* is "Short" (<5 characters), the message is subject to review (*review* = "true") and the *ReviewObject.reason* should be set to "Greeting is too short".

7. Define the second rule. Click the green plus sign button above the Rules column and select Rule.

+ * 🗶 A 🛇	苗・甘・艮 60 忠 🖬 🎟・ 🕲
Rule	RI
Condition	Short
Action 🕨	7.

Notice that when the new rule column is added, it is added with the heading R1 and the rule you just defined is moved over to the right, under the heading R2.

1

* 6	ReviewDecisionTable <enter< th=""><th>description></th><th></th></enter<>	description>	
A1 R	1:	╋╸ ※ ◇ ◇ 苗・	ti - 🗈 i 60 🐮 🔜 🖽 - 🔞
•	<u>C</u> onditions	R1	R2
C1	HelloObjectType.greeting.length()	ζ	Short
C2	HelloObjectType.message.length()	Z	-
×	Conflict Resolution		
-	Connect <u>R</u> esolution		
-	Acti <u>o</u> ns		
Al	assert new ReviewObjectType(✓
	reason:String		"Greeting is too short"
	review:String)		"true"

Beware: When you make the first edit to the new rule, it will switch back to its original location and original heading name. Always be sure that you are editing the rule you intended to edit.

Make edits to this rule, just as you did the previous rule, using the following table to guide you:

Section of Decision Table	Property	Value to Set
Condition	HelloObject.greeting.length() Medium	
Condition	HelloObject.message.length()	Short
Action	ReviewObject.reason	"Length is too short"
Action	ReviewObject.review	"true"

Define the last rule that appears in the table as shown at the beginning of this section. Define it in the same way you
defined the rule in the preceding step. The table below provides the values you will use.

Section of Decision Table	Property	Value to Set
Condition HelloObject.greeting.length()		Long
Condition	HelloObject.message.length()	-
Action	ReviewObject.reason	null
Action	ReviewObject.review	"false"

Note that you must select the null from the drop-down list, rather than type it into the text field.

•	<u>C</u> onditions	R1	R2	R3
C1	HelloObjectType.greeting.length()	Short	Medium	Long
C2	HelloObjectType.message.length()	-	Short	-
×	Conflict <u>R</u> esolution			
-	Acti <u>o</u> ns			
Al	assert new ReviewObjectType(✓	✓	 Image: A set of the set of the
	reason:String	"Greeting is too short"	"Length is too short"	null
	review:String)	"true"	"true"	"false"

9. Run the **Gap Analysis** tool to automatically create the final rule. It will determine which conditions have not been covered by the existing rules. Click the Gap Analysis button on the Decision Table editor toolbar.

+ - ≫	ा⇔ ङ∣ॉच•ॉ∎•ा	💫 । 60 🏦 🔛 🖽 - 🔞
R1	R2	R3
Short	Medium	Gap Analysis
-	Short	-

The Gap Analysis window opens showing the rule that it determined was missing. Click the checkbox above the rule to allow the rule to be included in the decision table. Click OK.

💩 Gap .	Analysis	×				
	There is 1 missing rule in the decision table.					
	You can add the missing rul the table header column.	e to the decision table by selecting the checkbox in				
	Conditions					
HelloOb)jectType.greeting.length()	Medium				
HelloOb)jectType.message.length()	Long, Medium				
💽 <u>F</u> it Co	olumns To Width					
Help		OK Cancel				

The rule now appears in the decision table as Rule 3 and the rule that you defined in the last step moves to the Rule 4 position. Assert the action for Rule 3 (Column **R3**) and add the following action values:

reason=null review= "fal	.se"		
R1	R2	R3	R4
Short	Med	lium	Long
-	Short	Medium,Long	-
~	~	 Image: A start of the start of	✓
"Greeting is too short"	"Length is too short"	null	null
"true"	"true"	"false"	"false"

 $\label{eq:click} Click \ \textbf{Save} \ and \ close \ the \ \textbf{HelloRules.Rules} \ tab.$

Adding the Review Message Human Interaction

1. Add a new human interaction activity to the HelloWorldProcess. If necessary, click the HelloWorldProcess tab in the Design Editor to bring it to the front.

Expand the Activities accordion panel in the Component Palette and, from the Interactive section, click and drag a User activity, dropping it *outside* of the Requester lane and directly below the ReviewNeeded activity.

When the Properties window opens, name the activity Review Message and click OK.

1

🔶 Properties - UserTask	
Basic Implementation	
Name: Review Message	۲
Description	۲
Sampling Point	
*	>
Нер ОК Са	ncel

You will then be prompted to assign a role to this activity. Select the Reviewer role from the drop-down list. Click OK.

🔷 Role properties		
Name		
Reviewer	-	New
Use variable value as parameter content		
review@utcome		-
Help	ОК	Cancel

2. Define the implementation for the Review Message activity. In the Design Editor, right click Review Message and select Properties.

Requester	Start	Request Hello ReviewNeeded Write Message	End
Reviewer		Node 'Review Message' does not have an ourgoing penautic Node 'Review Message' has no implementation defined	ce flow

In the Properties window, click the green plus sign button next to the Human Task field to define a new human task implementation.

🔷 Properties - Review Message	
Basic Implementation	
Implementation Type: [Iser task	•
Human Task:	🔦 🎤
Human Task Attributes	Add

In the Create Human Task window, define the following properties. Accept the other default values.

Name: ReviewMessage Title: Review the Message

.

🕹 Create Human Task 🛛 🛛 🔀							
General							
Name:	ReviewM	lessage			Pr	iority:	3 (normal) 🔻
Pattern:	📳 Simp	ble					•
Title:	Review t	he Messaç	je				
Outcomes:	APPROVE	E,REJECT					Q
Parameters:							+ ×
Parameter		Name		Туре		Editabl	e
Outcome targ	get:						
Help						ОК	Cancel

Click the green plus sign button next to the Parameters panel as shown above to add parameters.

The **Data Object** popup window appears next to the Create Human Task window. Drag the hello and review process data objects into the **Parameters** panel. Select the **Editable** checkbox for the hello data object only. Also drag the reviewOutcome process data object into the **Outcome Target** field.

🕹 Create Hur	nan Task		×	
General				
Name: Revi	ewMessage	Priori	ty: 3 (normal) 🔻	
Pattern: 👩	Simple		-	🖕 Data Object 🛛 📐 🛛 🔀
Title: Revi	ew the Message			Drag Data Objects to Parameters tab Target field.
Outcomes: APPF	ROVE,REJECT		Q	A9 A3 🗉 🕞 🛶
Parameters:			🕂 🗙	E
Parameter	Name	Туре	Editable	hello
HelloObject	hello	HelloTypes.Hello	 ✓ 	review
ReviewObject	review	HelloTypes.Revi 🗲		HelloworldProject
Outcome target	reviewOutcome	6		
Help			K Cancel	Help Close

Click Close in the Data Object popup window and click OK in the Create Human Task window.

Click \mathbf{OK} in the Properties window.

Click Save All.

3. Create a new task flow, based on the ReviewMessage human task that you created in the previous step.

Click on the Application Navigator tab and then right click on the HelloWorld_UI project node and select New...

Application	🚱 в 🚱 💶 🗌				
🔁 HelloWorld_OBE 🛛 👻 🔁 🗸					
Projects	◙ ಔ ♥ - № -				
🛅 HelloWorld_UI	<u>^</u>				
🛅 HelloWorldPrc 🇳	New.				
😟 🛅 Processes	Edit Project Source Paths				
🖻 🛅 SOA Cont 💥	Delete Project				
🗄 📄 classe	Version Project				
🗄 📄 testsi 👝					
😟 🕀 💼 😟 💷	Find Proje <u>c</u> t Files				
🕀 🧰 🕀	<u>S</u> how Overview				

The New Gallery wizard opens. Click the **All Technologies** tab. Expand the **Web Tier** category and select **JSF**. From the Items panel, select **ADF Task Flow Based on Human Task**. By using this approach to creating the taskflow, you can store the taskflow in the same project that is storing your earlier human taskflow, rather than have separate projects for each taskflow.

All Technologies Current Pro	oject Technologies	
Search All Technologies		
<u>⊂</u> ategories:	Items:	Show All Descriptio
General General General	🔛 ADF Task Flow	
Business Tier Client Tier Database Tier SoA Tier Sorvice Components Transformations Web Tier Applet	ADF Task Flow Based on Human Task Creates data controls and an AUF task flow b First you will be asked to select a human task This will generate a set of data controls, and wizard. To enable this option, you must select a proje Navigator.	ased on an existing SOA human task. definition from the SOA Resource Chooser. then launch the ADF task flow creation sct or a file within a project in the Application
Facelets	ADF Task Flow Template	
- JSF	🖺 JSF Declarative Component	
- JSP Serviets	🔢 JSF Page	
Struts	35F Page Flow and Configuration (faces-confi	g.xml)
All Items	🚰 JSF Page Fragment	
	🚳 15E Page Template	

Click OK.

You are now prompted to identify the XML file that was generated when you defined the ReviewMessage human task. The **SOA Resource Browser** window opens.

It is currently looking at the <code>HelloWorld_UI</code> folder. Navigate up one level to the <code>HelloWorld_OBE</code> folder and then open the <code>HelloWorldProject</code> folder. Select the file, <code>ReviewMessage.task</code>.

🖕 SOA Resource Browser 🛛 🔀 🔀
ၽ File System
Location: AlloworldProject
Image: Second
File Name: ReviewMessage.task File Iype: Task Definition Files (*.task) Help OK

Click OK.

In the $\ensuremath{\textit{Create Task Flow}}$ window, accept all the default values and click $\ensuremath{\textit{OK}}.$

Create Task Flow	
Create a task flow source file whose contents define either a bounded task flow or part of the web application's unbounded task flow.	
A bounded task flow can refer specifically to JSP pages or page fragments, but not both. You can also designate the bounded task flow to be a train at this time.	
<u>Fi</u> le Name:	
ReviewMessage_TaskFlow.xml	
Directory:	
C:\JDeveloper\mywcrk\HelloWorld_OBE\HelloWorld_UI\public_html\WEB-INF	vse
Create as Bounded Task Flow	
Task Flow ID: ReviewMessage_TaskFlow	
Create with Page Fragments	
Create Train	
Base on Template:	-
✓ Update the Task Flow when the Template Changes	
Нер ОК Са	ncel

4. The bounded task flow that you just created, using the wizard, now appears in the Task Flow editor (Diagram tab). It does not yet have a web page associated with the taskDetails2_jspx. This is the reason that you see an error indication. Recall that in the basic Hello World process, you allowed Studio to automatically generate a form for the Request Hello task flow. For the Review Message task flow, you create your own.

Double click the **taskDetails2_jspx** icon in the Task Flow editor. The Create JSF Page window opens. Accept all defaults and click **OK**.

े Create JSF Page 🛛 🔀
Enter the name, directory, and choose a type for the JSF Page. Optionally reference a <u>Page Template</u> to include its content in this page, or apply a <u>Quick Start Layout</u> to add and configure an initial set of layout components.
Eile Name: kaskDetails2.jspx
Directory: C:\JDeveloper\mywork\HelloWorld_OBE\HelloWorld_UI\public_html Browse
☑ ⊆reate as XML Document (*.jspx)
Render in Mobile Device
Initial Page Layout and Content
Ilank Page
O Page Template Oracle Three Column Layout -
O Quick Start Layout
One Column (Stretched)
Browse
Page Implementation (UI components are not exposed in managed bean)
Help OK Cancel

It will take several seconds for the JSF page designer to become initialized.

 You should resize some of the accordion panels in the left side of the Studio window now in order to make the Data Controls accordion panel larger, since you will be selecting components from it in this step.

Expand HelloWorld_UI_ReviewMessage > getTaskDetails > Return. Click Task and drag it into the Taskflow Design editor. When you drop it, a menu appears. Select Human Task > Complete Task with Payload.

✓ Data Controls		
🖮 🔚 getTaskDetails(String, Stri	Create	
🗄 💼 Parameters	😐 Caro <u>u</u> sel	
🖻 🖓 Return	Eorm 🕨	
	Gantt 🕨	
	Gauge	
	Geographic Map 🕨	
	Graph	
	Hierarchy Viewer	Type, paste, or
	Human Task 🔹 🕨 🔛 Complete Task	with Payload
	Master-Detail 🔹 🕨 🚵 Complete Task	without Payload HS
	Multiple Selection 🔹 🚵 Task details for	r email
	Navigation 🕨 🚵 Task Header	
	Single Selection 🔹 🗎 Task Action	
	Iable 🛛 🕨 🚔 Task History	
	Tr <u>e</u> e 🕨 🔠 Task Comment	and Attachment

6. The Edit Action Binding window opens next. If you are unable to expand the HelloWorld_UI_ReviewMessage node, click OK in this window.

A second Edit Action Binding window will appear. Expand HelloWorld_UI_ReviewMessage > getTaskDetails(...) > Return. Select the Task object. Click OK.

춸 Edit Action Bi	inding		
Select a data collect on the data objects Data <u>C</u> ollection:	tion and the action y of the selected coll	you want your control ection.	l to initiate. The control initiates the action
□ 🛃 HelloWorld □ getTas □ Re □ Re	_UI_ReviewMessag kDetails(String, Strii turn] <mark>Task</mark>	e ng, String)	Jr.
Select an <u>I</u> terator:	taskIterator		▼ <u>N</u> ew
Operation:	UpdateAction(Strin	ng) 💌	
	Apply to all ite	rators in page defintio	on
Parameters :			
Name	Туре	Value	Option
action	java.lang.String		
Help			OK Cancel

After a few seconds, the JSF page appears in the design editor (in design mode). Notice that it contains two boxes, the upper one to accommodate the contents of the message entered by the user, and the lower one to accommodate the review comments by the reviewer.

Date	#{date.inputValue}	
Greeting	#{greeting.inputValue}	\leftarrow
Message	#{message.inputValue}	
Review Ob	ject	
Review	#{review.inputValue}	
Reason	#{reason.inputValue}	

You are now finished with the taskflow and JSF page. Click **Save All**. You can close all tabs except for the HelloWorldProcess tab.

Adding Conditional Branching

1. Add an Exclusive Gateway to the process model.

You will be moving design elements around quite a bit in this section and, by default, the Design Editor has Automatic Layout turned ON. You may wish to turn it off so that it does not undo your moves. Click the **Layout** button on the toolbar.

On the Automatic Layout menu that appears, click the **ON** button, which toggles it to an **OFF** button. Note that you can come back to this menu and get a one time auto-layout at any time or choose to turn automatic layout back on.

Make room for changes to the process model by moving the End and Write Message activities further to the right.

From the Gateway accordion panel of the Component Palette, click and drag an Exclusive Gateway, dropping it on the sequence flow between ReviewNeeded and Write Message.

Important: The line must appear **blue** when you drop the gateway in order for the transition line to connect to it properly. You will probably have to drop it near the center of the space between activities in order to pick up the blue line. You can move the gateway further to the left after you have finished placing it.

When you drop it, the Properties window opens. Change the name of the gateway to g1 and click OK.

🔷 Properties - ExclusiveGateway	×
Basic Outflows Order	
Name: g1	6
Description:	6
Sampling Point	
Help OK Cance	Ż

Move the g1 gateway over to the left to make room for another exclusive gateway that you will add in the next step.

 Add a second exclusive gateway to the transition line between g1 and Write Message. Follow the same procedure you did in the last step. Name this gateway g2.

Start Request Hello	Request Helb Reviewveeded		al		Write Message		
			Review Message				

3. Add a conditional sequence flow from g1 to Review Message. Right click on g1 and select Add conditional sequence flow.

\sim	\sim	
\sim	Properties	ī
g1	Add conditional sequence flow	
	¥ Delete ₩	

Connect the transition line by clicking on **Review Message**. Do not worry about the warning message regarding lack of default sequence flow. You will fix that in the next step.

Reshape the transition line as shown below by clicking and dragging it into the desired shape.

4. Add a default sequence flow from Review Message to g2. Right click on Review Message and select Add default sequence flow.

Connect the other end of the sequence flow by clicking on g2. Reshape the transition line as shown below, as you did in the previous step.

 Add a conditional sequence flow from g2 back to Request Hello. Reshape the transition line as you have the other two transition lines. (It will be obscured by the existing main sequence flow) It should look like this when you've finished.

6. Define the condition for the sequence flow from g1 to Review Message. Double click on the transition line between the two objects. The Transition properties window opens.

On the **Description** tab, name the transition Review Needed.

🔶 Transition from Activity: 'g	1' to Activity: 'Review Message' 🛛 🛛 🔀
Description Properties	
Name: Review Needed	

Click the Properties tab. On this tab, you define an expression whose outcome determines whether the process flows down this transition to the Review Message activity. In other words, you define the condition for the conditional sequence flow. Click the Expression Builder button on the right side of the window.

📤 Transition from Activity: 'g1' to Activity: 'Review Message'	×
Description Properties	
Туре	
Condition	-
Expression:	
O Simple ○ XPath	
Transition from Activity: 'g1' to Activity: 'Review Message' Description Properties Type Condition Expression: Simple \ XPath	^ 🖪
	~

The Expression Builder opens. Build the expression in 3 steps:

- In the Variables panel, expand review and select the review attribute.
 Click the Insert Into Expression button. This puts review.review into the Expression panel at the top. 3. In the Expression panel, add:
- - = "true" to the existing variable reference.

🕌 Expression Builder	X
Expression:	S (2)
review.review = "true"	
2 -> A Insert I	into Expression
¥ar	iables
HelloWorldProcess	
Content Preview:	Description:
review.review	Data Declaration
Help	OK Cancel

Click **OK** when finished to close the Expression Builder.

Click OK to close the Transition properties window.

7. Define the condition for the sequence flow from g2 to Request Hello. Double click on the transition line between the two objects. The Transition properties window opens.

On the **Description** tab, name the transition Message Rejected.

🕹 Transition from /	Activity: 'g2' to Activity: 'Request Hello'	
Description 📄 Prop	erties	
	1	
Name:		
Message Rejected		
Description:		

Click the Properties tab. Click the **Expression Builder** button to open the Expression Builder and define the following expression, using the same procedure as you did in the last step:

reviewOutcome = "REJECT"

Expression Builder				
Expression:		6	0	~
reviewOutcome = "REJECT"				
				_
📣 Insert I	nto Expression			
Yari	ables			
👀 👀 📃 I 📼 🚥				
HelloWorldProject				
Content Preview:	Description:			
			-0	-
Help	OK		Cancel	

Click $\ensuremath{\textbf{OK}}$ to save the expression.

Click **OK** in the Properties window.

8. Your process should now look like this:

Adding a Script Task

When the process flow reaches the second gateway (g2), it checks the value of the reviewOutcome variable. There
must be a value in the reviewOutcome variable in order for the process to move on to the Write Message activity. If
the value is REJECT, the process returns to Request Hello. When a message is flagged for review, the reviewer
clicks either REJECT or ACCEPT, thereby populating the reviewOutcome variable.

If the process skips the review branch and moves from the g1 gateway directly to the g2 gateway, the reviewOutcome variable has no value.

Add a script task to initialize reviewOutcome between g1 and g2. From the Activities accordion panel of the Components Palette, click and drag a Script task, dropping it on the transition line between g1 and g2.

When the Properties window appears, name the activity InitApproval on the Basic tab and click OK.

2. Define the implementation for the InitApproval task. Right click on InitApproval and select Properties.

When the Properties window opens, click the **Implementation** tab. Then select the **Use Associations** checkbox. Click the pencil icon to open the Data Associations window.

Properties - InitApproval			
Basic Implementation			
Implementation Type: E Script task	•		
Use Associations Type: Simple			
Use Transformations			

In the Data Associations window, drag the **reviewOutcome** process data object from the right panel into the gray box in the left panel labeled "**Custom Assignments**". In the text field to the left of the gray box, provide an initial value for the *reviewOutcome* variable of "ACCEPT".

🖕 Data Associations	
	📲 🕷 📃 📼 —
Drag variables from the right panel into fields on the left.	HelloWorldProcess
Нер	OK Cancel

Click OK in the Data Associations window.

Click OK in the Properties window.

Click Save All. The Hello World process is now complete!

Deploying and Testing the Application

In this section, you deploy the Hello World application to the BPM engine running in the WebLogic server that is part of your SOA installation. This tutorial assumes that your server is running on a remote Linux machine. You need to know:

The server hostname and port

The WebLogic domain in which SOA is running

The username and password for the WebLogic administrative user

The name of the specific WebLogic server to which you will deploy. This tutorial assumes a single server (i.e. not managed instances) configuration and will deploy to the Admin Server.

Before deploying the application, you connect to the internal LDAP realm within the WebLogic server and map the Reviewer and Requester roles to a user in the LDAP.

After deployment, you run the Oracle BPM Workspace web application to test the Hello World application.

Mapping the Studio Role to an LDAP Role
Deploying the Process
Testing the Process in Workspace

Mapping the Studio Role to an LDAP Role

1. In the BPM Project Navigator, expand the HelloWorldProject and right click on the Organization node, selecting Open.

It may take several seconds to open the Organization editor.

2. Add user members to the Requester role. In the Organization editor, select the Requester role. Click the green plus sign button to the right of the Members panel.

Organization		
Roles		
Name		
Process Owner		î /
Requester		i 🕺
Reviewer		V
Role		
Name: Requester		
Members		
		Type: User 💌
Name	Туре	•
		×

The **Identity Lookup** window opens. Studio is not yet aware of your remote server, so the only application server that appears is the integrated weblogic server that was installed with JDeveloper.

Create a new profile for your remote server. Click the green plus sign icon next to the **Application Server** field to launch the **Create Application Server Connection** wizard.

🕌 Identity Look	up	×
Application Server:	IntegratedWebLogicServer (Resource Palette Connection)	
Realm:	•	
Search <u>P</u> attern:	* User Name	1
Search User		
Select	Hierar Repor Detail << >>	

3. In the Name and Type page of the wizard, enter Remote_WLServer as the Connection Name. Make sure that the Connection Type is WebLogic 10.3, and click Next.

In the Authentication page of the wizard, enter weblogic as the **Username** and welcome1 as the **Password** (or substitute your weblogic password). Click **Next**.

💩 Create Application S	Server Connection - Step 2 of 5	X
Authentication		
Authentication Configuration Test Finish	Specify a username and password to autherticate the connection. Username: weblogi: Password:	
	< <u>B</u> adk <u>N</u> ext > Enish Cance	3

In the Configuration page of the wizard, enter your **Weblogic Hostname**, accept the default values for Port and SSL Port, and enter domain1 in the **Weblogic Domain** field. Click **Next**.

In the Test page of the wizard, click the **Test Connection** button. You should see results similar to those shown below. If not, click the **Back** button, correct any errors and test again.

Test		
Authentication Configuration Test	Clok Test Connection to determine if the inf connection with the application server. <u>Test Connection</u> <u>Status:</u>	ormation specified successfully establishes a
Einish	Testing JSR-160 Runtime Testing JSR-60 Domainkuntime Testing JSR-68 Testing JSR-88-10CAL Testing JSR-160 Edit Testing HTTP Testing HTTP Testing HTTP Testing HTTP Authentication S of S tests successful.	512C455. 512C455. 512C455. 512C455. 512C455. 512C455. 512C455. 512C455.

Click Finish.

You are now returned to the Identity Lookup window.

 Select the Remote_WLServer profile from the Application Server list. After a few seconds, Studio will connect to the LDAP server in the remote server and you see jazn.* appearing in the Realm field.

Click the browse icon next to the **User Name** field to bring up a list of all users in the Demo Community from the LDAP server. Select **jcooper** from the list and click the **Select** button.

🛓 Identity Look	up			
Application Server:	Remote_WLServer (Re	esource Palette (Connection)	- +
Realm: —>	jazn.com			-
Search <u>P</u> attern:	*	User Name		-
Search User				
eheming fkafka istone jausten				^
jcooper				 _
Select	Hier Rep	<u>D</u> etail		>>
Selected User				
jcooper 🗲				
			Remove	Detail
Help			<u>o</u> k	Cancel

jcooper will appear in the Selected Users panel of this window. Click **OK**. You are returned to the Organization editor.

Name		
Process Owner		<u>^</u>
Requester		
Reviewer		~
Role		
Name: Requester		
Members		
		Type: User 🗸 🗸
Name	Туре	
Invente	· r = -	

5. Follow the same procedure as in the previous step to add jcooper to the **Reviewer** role as well.

Roles		
Name		
Process Owner		<u>^</u>
Requester		· · · · · · · · · · · · · · · · · · ·
Reviewer		
Role		
Name: Reviewer		
Members		
		Type: User
Name	Туре	
jcooper	User	*

6. Click Save All and close the Organization editor.

Deploying the Process

1. Deploy the HelloWorldProject.

In the Application Navigator, right click HelloWorldProject and select Deploy > HelloWorldProject...

Applicatio	BPM Pro	6 -	RelloWorldProcess	🇞 SayHello.task	
🔁 HelloWorld	_OBE		8 (+) (-)		
Projects	S & 2	∀ - 1 5 - I			
	orld_UI				
	orldProject				
<u> 1</u>	<u>l</u> ew		Ctrl-N		
E	dit Project Source P <u>a</u> th	s			505K
🔰 🗶 C)elete Project			-	- · · · · · · ·
7	ersion Project				DeviewNeed
👘 F	ind Project Files			.equest Hello	Reviewiveed
	how Overview				
a 1	1ake HelloWorldProject.	jpr	Ctrl-F9		
ස්ථි F	Re <u>b</u> uild HelloWorldProje	ct.jpr	Alt-F9		
[Deploy			HelloWorldP	roject
► <u>F</u>	lun	NG.			
🕨 Applica 🏶 🛙	<u>e</u> bug				

The Deploy HelloWorldProject wizard opens.

2. In the Deployment Action page of the wizard, select Deploy to Application Server and click Next.

Deployment Action	
Deployment Action	Select a deployment action from the list below.
Epolog Configuration Summary	Deploy to Application Server Deploy to SAR Deploy to SAR Deploy this archive to SOA configured Application server(s)
Help	< Back User > Einish Cancel

In the Deploy Configuration page, click the **Overwrite any existing composites with the same revision ID** checkbox and click **Next**.

Deploy Configuration	1	
Deployment Action Deploy Configuration Task flow deployment Summary	Project Project: HelloWorldProject Current Revision ID: 1.0 New Revision ID: 1.0	SOA Configuration Plan Do not attach Select a configuration plan from the list.
teip >	Mark composite revision Overwrite any existing Use the following SOA o	as default. composites with the same revisit D. configuration plan for all composites: Browse < Back Next > Einish Cancel

In the Task flow deployment page, select the checkbox next to **Projects**. This selects all taskflow projects in the HelloWorldProject, so you will see the HelloWorld_UI project selected now also.

Task	flow deploymen	t			0101010	010101010	149898340	
八 人 人 (人) (人) (人) (人) (人) (人) (人) (人) (人)	Deployment Action Deploy Configuration T ask flow deploymer	Ear Profile Name:	HelloWor	rld_UI nd composit generated p	e revision to rofiles to app	name lication	V	
	Select Server Summary	Optional: Select W/ Deployable Taskflor	Over AR profiles w Projects	write EAR 5. Uncheck p 5	orojacts ta ex	clude from	deployment	
	R	Projects Composite: He HelloWorld_UI	elloWorldP . jpr	WAR Yroject Hello	Profiles Vorld UI		App Context Ro /workflow/Hello\	ot World_U1
H	elp				< <u>B</u> ack	<u>N</u> ext >	Einish	Cancel

Click Next.

In the Select Server page, select Remote_WLServer and click Finish. Deployment will begin.

Select Server	and constrong of the second	
Deployment Action Deploy Configuration Task flow deployment Select Server SOA Servers Summary	Application Servers: IntegratedWebLogicServer (domain unconfigured) Remote_WLServer	ີ ເຊ
Help	< Back Next >	Einish Cancel

Check the **Deployment** tab in the **Log** panel in the lower central portion of the Studio window to watch the progress and determine when deployment has finished. This first deployment of the SOA Composite (shown as sca_HelloWorldProject_rev1.0.jar in the log viewer screenshot below) will be very quick. The deployment of the WAR file containing the taskflows (HelloWorld_UI.war) will take much longer. Total deployment time will be around 15 minutes, depending upon your environment.

_				
	[11:58:38 AM]	Successfully deployed archive sca_HelloWorldProject_rev1.0.jar to partition "de		
	[11:59:03 AM]	Retrieving existing application information		
	[12:07:41 PM]	Deploying Application		
	[12:09:41 DM]	[Deployer:149192]Operation 'deploy' on application 'HelloWorld_UI' is in progression		
	[12:12:20 DM]	[Deployer:149194]Operation 'deploy' on application 'HelloWorld_UI' has succeede		
	[12:12:36 PM]	Application Deployed Successfully.		
	[12:12:49 PM]	The following URL context root(s) were defined and can be used as a starting po		
	[12:12:49 PM]	HeiloWorld_UI.var		
	[12:12:49 PM]	/workflow/HelloWorld_UI		
-	[1Z:1Z:49 PM]	HelloWorld_UI.war		
12	[12:12:49 PM]	/workflow/HelloWorld_UI		
÷	[12:12:50 PM]	Elapsed time for deployment: 15 minutes, 18 seconds		
E E	[12:12:50 PM]	Deployment finished		
- All	2	~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~ ~		
12				
l în	Messages E	ktensians Deployment 📑 SOA 🗐 🔍		

Testing the Process in Workspace

1. Open a browser (either on the server or from your windows machine) and enter the following URL:

http://<your server hostname>:7001/bpm/workspace

When the Welcome page of the Workspace appears, enter jcooper in the Username field and welcome1 in the Password field. Click Log In.

The main Workspace window opens.

2. Instantiate the process by clicking the HelloWorldProcess v1.0 link under Applications in the Tasks tab on the left side of the window.

Note: If the HelloWorldProcess v1.0 link does not appear in the Applications panel on the Tasks tab, click the **Process Tracking** tab. It should appear there. You can instantiate the process from that tab.

This action executes the first task of the process, Request Hello.

3. First enter a message that will *not* require review. In the **Please Enter a Hello Message** popup window that appears, enter a value in the **Date** field using the format MMM d, yyyy (example: Jun 2, 2010) Also enter a value in both the **Greeting** and **Message** fields (Greeting field length > 5, Message field length > 5). Click **Submit**.

ease Ente	ra Hello Message	Actions 🕶	SUBMIT	Claim
2 🔏 De	etails ()			
Contents				
Date	Jun 2, 2010			
Greeting	Hello World			
Message	Have a wonderful day!			

After submitting the message, the process goes to the ReviewNeeded business rule. Since both the Greeting and the Message are considered "Medium" in length, the message does not require a review. Hence the process goes through the g1 gateway, the *reviewOutcome* variable is initialized in the script task, and the process flows on through the g2 gateway to the **Write Message** activity.

The Write Message activity is a service type activity, not an interactive activity., therefore there will be no indication of its action within the Tasks panel of the Workspace.

4. Open the file that was created by the Write Message implementation. Recall that you configured the file adaptor to write the file to the file path "." (dot). This is relative to the domain1 directory within the WebLogic file structure.

On the server, using either a terminal window or an Explorer style window as shown here, navigate to:

<path to your mwhome>/mwhome/user_projects/domains/domain1

Find the file Hello_1.xml. Open it in a text editor to view the outcome from your process.

👂 📁 sysman		
👂 📁 tmp		
D 📁 UMSJMSFileSt	ore	
门 edit.lok		
🗍 fileRealm.pro	perties	
Hello_1.xml	Dpen with "Firefox Web Browser"	
🔊 Hondp.Gut	Open with "Emacs Text Editor"	
👂 📁 utils	Open with "Text Editor"	
▷ iserver_10.3	Open with Other <u>Application</u>	
domain-registry.xml	Make Link	
🚺 ocm.rsp		
C registry dat		

It should look something like this:

Hello_1.xml ×
<pre><?xml version="1.0" encoding="UTF-8" ?><helloobject< pre=""></helloobject<></pre>
<pre>xmlns="http://xmlns.oracle.com/bpm/bpmobject/HelloTypes/</pre>
HelloObject">
<pre><date xmlns="">2010-06-02T00:00:00Z</date></pre>
<pre><greeting xmlns="">Hello World</greeting></pre>
<message xmlns="">Have a wonderful day!</message>

 Create another instance of the HelloWorld process by once more clicking the HelloWorldProcess v1.0 link. This time, enter a message that will be routed to the Review Message activity.

When the **Please enter a hello message** window appears, enter a date and some text that has a *length* < 5 in the **Greeting** field. Leave the **Message** field blank. Click **Submit**.

Please enter a Hello message				
Please enter a Hello message	Actions - SUMMIT Claim			
≥ 🤮 Details 🕕	. т. Г			
Contents				
Date Jul 12, 2010				
Greeting Hi				
Message				

Summary

In this tutorial, you have created a simple process that illustrates some of the basic features of Oracle BPM's modeling and implementation capability. This is just a start. However, it has given you hands on exposure to some of the key benefits of the product such as:

Modeling complex business logic with the Studio IDE Creating business rules with complex decision tables Creating auto-generated and custom forms for human interaction Moving and persisting data throughout a business process Integrating a business process with external systems

Resources

Oracle Fusion Middleware Modeling and Implementation Guide for Oracle BPM 11g Tutorial: Getting Started with Oracle SOA Suite 11g Tutorial: Installing Oracle BPM 11g To learn more about , refer to additional OBEs in the OLL Web site

Credits

Lead Curriculum Developer: Jill Moritz Other Contributors: Heidi Buelow

User Comments

Title:

Post as miguel.pardal@ist.utl.pt
Post anonymously

By submitting a comment, you confirm that you have read and agreed to the terms and conditions.

This feedback form is for tutorial corrections and suggestions. Because of the volume of suggestions, we cannot reply to every comment. In particular:

If you have general questions about this Oracle software product, <u>consult the OTN forums</u> instead. If you are encountering a software problem and need to request support or consulting, <u>file a service request on My Oracle Support</u>.

If you want to order hardcopy documentation, go to the Oracle Store.

Submit a new comment (Comments are moderated and will not appear immediately.)

Hardware and Software **Engineered to Work Together**

About Oracle |Oracle and Sun | 🔝 | Careers | Contact Us | Site Maps | Legal Notices | Terms of Use | Your Privacy Rights