Oracle Forms / Java tutorial

How to build a new JavaBean

1. Purpose

The purpose of this tutorial is to demonstrate how to build and integrate a new JavaBean in an Oracle Forms application.

2. The goal

In this tutorial we are going to develop a JavaBean that retrieved the background color of a Forms canvas.

But, why not simply use the `Get_Canvas_Property( ‘canvas_name’, BACKGROUND_COLOR )` built-in ?

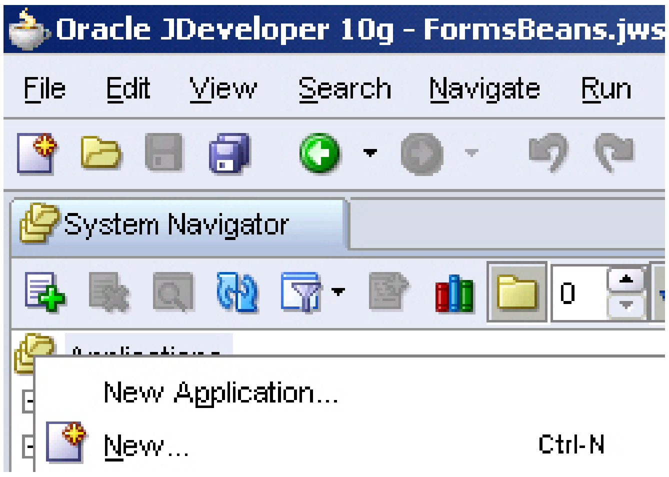
The reason why we cannot use the Forms GET_Canvas_Property built-in is that it returns **null** if the background color is not explicitly set.

The trick to solve this problem is thus to use a Forms bean area item and locate in on the Forms canvas. The background color property is then read from JavaBean container’s parent container.

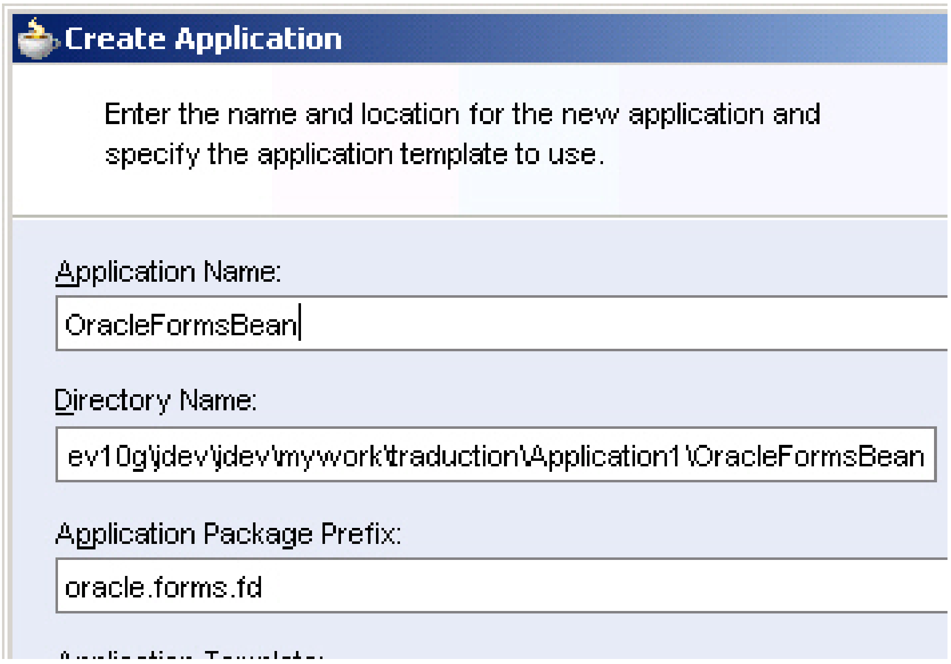
3. Build the bean

- Open Jdeveloper
- Create a new application

  Right click on the **Applications** node then click the **New Application** option



- Enter the name of the new application



Create Application

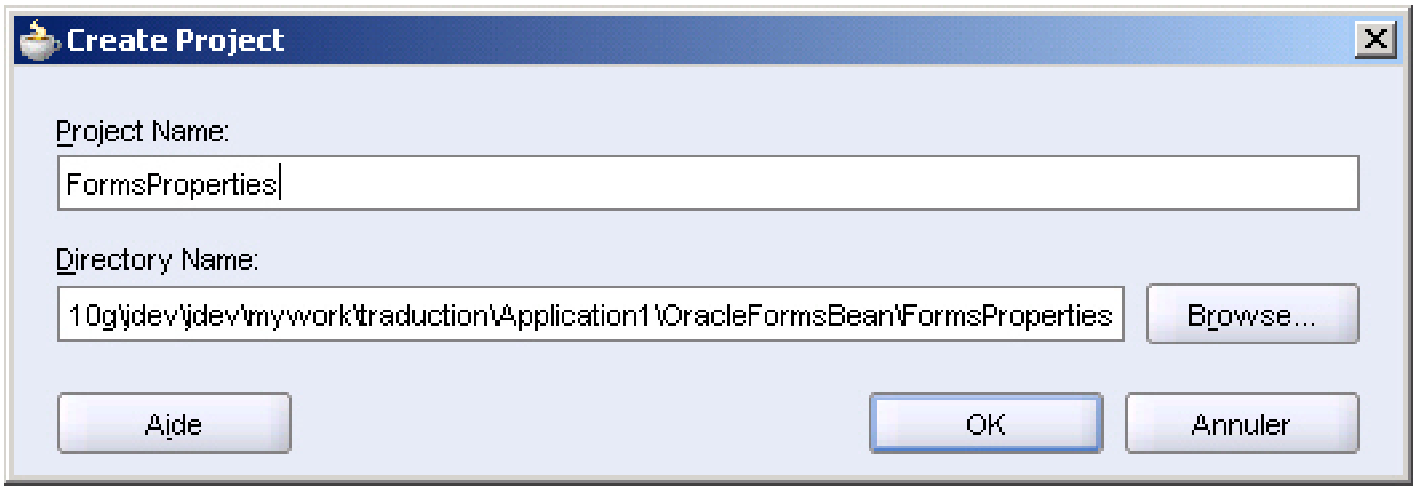
Enter the name and location for the new application and specify the application template to use.

Application Name:
OracleFormsBean

Directory Name:
ev10g\jdev\jdev\mywork\traduction\Application1\OracleFormsBean

Application Package Prefix:
oracle.forms.fd

- Enter the name of the new project

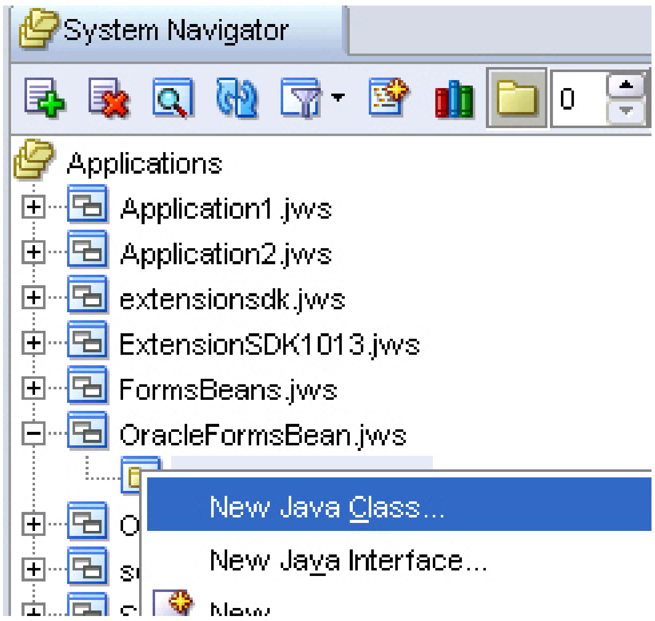


Create Project

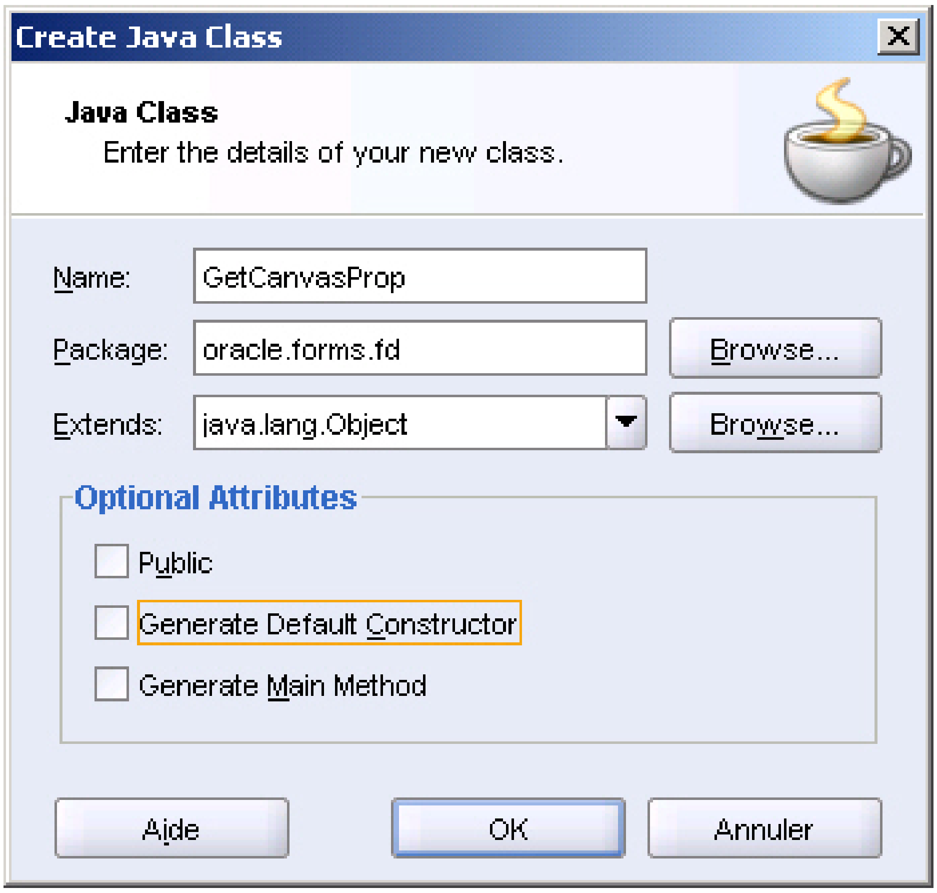
Project Name:
FormsProperties

Directory Name:
10g\jdev\jdev\mywork\traduction\Application1\OracleFormsBean\FormsProperties

- Add a new class to the project
  Right click on the **OracleFormsBean** node and click the **New Java Class** option



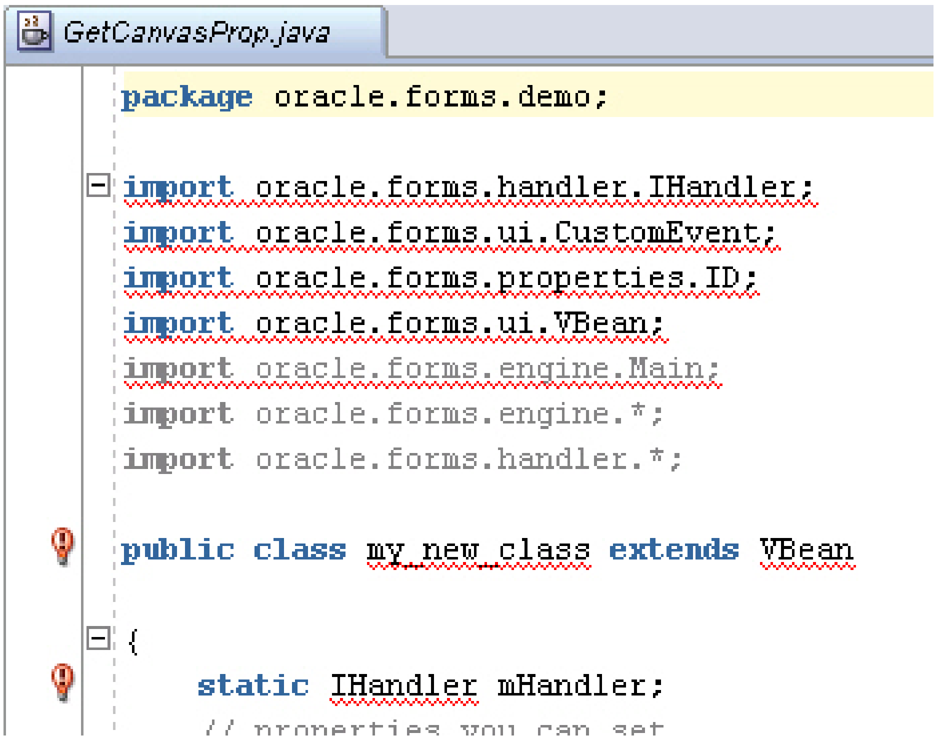
- Set the name of the class and the name of the package
  In this sample, the class name is **GetCanvasProp** and the package name is oracle.forms.fd



Name: GetCanvasProp
Package: oracle.forms.fd
Extends: java.lang.Object
Public [ ]
Generate Default Constructor [ ]
Generate Main Method [ ]

A Javabean source code 's skeleton (bean_template.txt) is provided with the samples to help you to quickly start a new PJC/Javabean class.

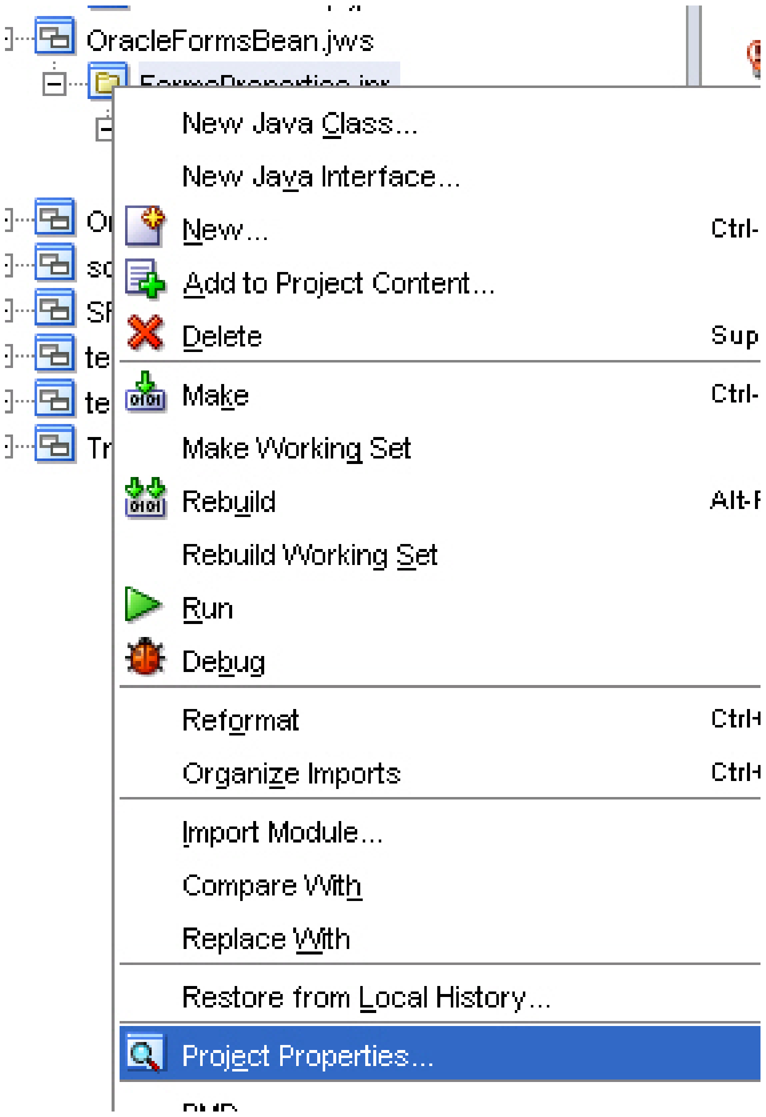
- Open the **bean_template.txt** file provided with the samples, then copy the content in the new GetCanvasProp.java window


GetCanvasProp.java

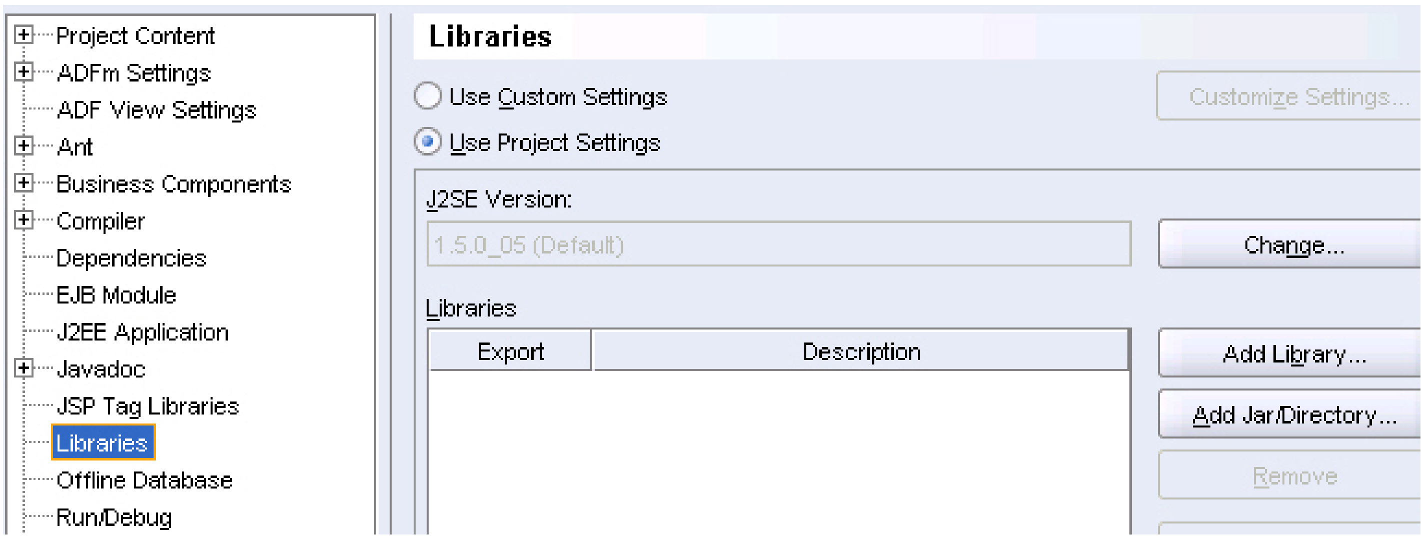
```
package oracle.forms.demo;

import oracle.forms.handler.IHandler;
import oracle.forms.ui.CustomEvent;
import oracle.forms.properties.ID;
import oracle.forms.ui.VBean;
import oracle.forms.engine.Main;
import oracle.forms.engine.*;
import oracle.forms.handler.*;

public class my_new_class extends VBean

{
    static IHandler mHandler;
    // properties you can set
```

Some of he imported classes are Forms specific and are not available in JDeveloper by default. You need to add these classes by creating a custom library in JDeveloper that points to the f90all.far file located in the <ORACLE_HOME>/forms90/java/ directory (versions 9.0.2 and 9.0.4) or in the <ORACLE_HOME>/forms/java/ directory (version 10.1.2), where It is named to frmall.jar.

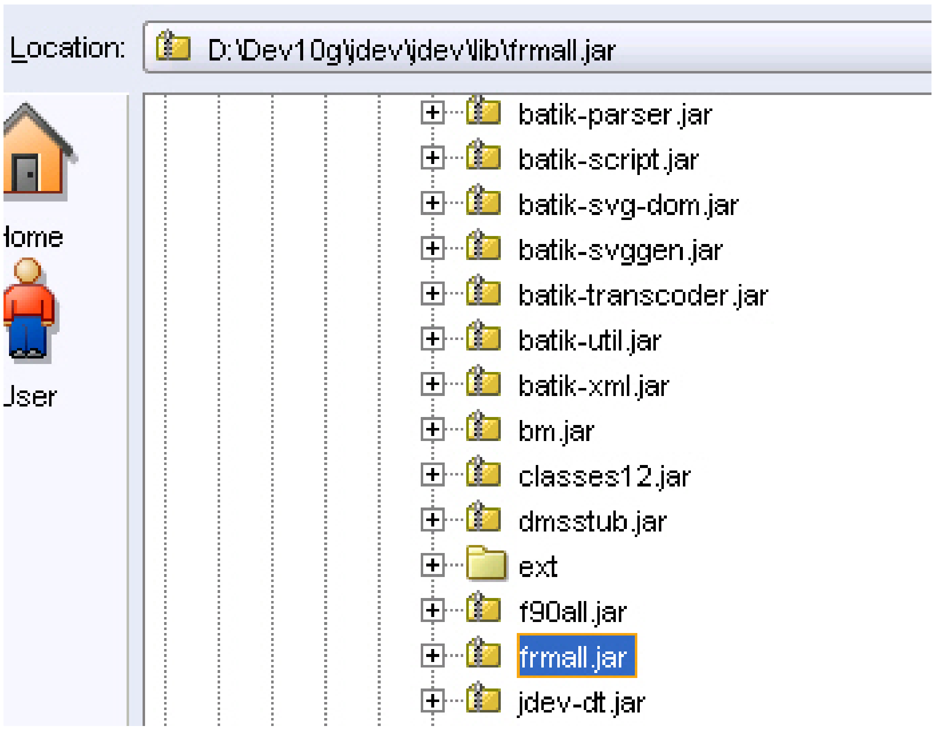
- Copy this file and paste it in the **/lib** sub-directory of your Jdeveloper installation path.
- Right-click on the **OracleFormsBean** project node then click the **Project Properties** option.



- In the **Libraries** tab, click the **Add Jar/Directory** button.



- Select the **f90all.jar** file (version 9.0.2, 9.0.4) or the **frmall.jar** file (version 10.1.2).


Location: D:\Dev10g\jdev\jdev\lib\frmall.jar

batik-parser.jar
batik-script.jar
batik-svg-dom.jar
batik-svggen.jar
batik-transcoder.jar
batik-util.jar
batik-xml.jar
bm.jar
classes12.jar
dmsstub.jar
ext
f90all.jar
frmall.jar
jdev-dt.jar

Ok, so this is the java code, now:

```
package oracle.forms.demo;

import oracle.forms.handler.IHandler;
import oracle.forms.ui.CustomEvent;
import oracle.forms.properties.ID;
import oracle.forms.ui.VBean;

public class my_new_class extends VBean

{
    static IHandler mHandler;
    // properties you can set
    protected static final ID SET_01   = ID.registerProperty("SET_01");
    protected static final ID SET_02   = ID.registerProperty("SET_02");
    // properties you can be get
    protected static final ID GET_01   = ID.registerProperty("GET_01");
    // events you can raise
    protected static final ID EVT_01   = ID.registerProperty("EVT_01");

    // default constructor
    public my_new_class()
    {
      super();
    }

    public void init(IHandler handler)
    {
      super.init(handler);
      mHandler = handler;
      // put your initialisation instructions here
    }

 /**
   * Set the properties to the bean
   **/
  public boolean setProperty(ID property, Object value)
  {
    if(property == SET_01)
    {
       System.out.println("Set property 01=" + (String)value) ;
       // add your code here
       return true;
      }
    else if (property == SET_01)
    {
        // add your code here
        return true;
    }
    else // default behaviour
       {
          return super.setProperty(property, value);
       }
  }

 /**
   * Get the properties of the bean
   **/
  public Object getProperty(ID property)
  {
    if (property == GET_01)
    {
      // return the corresponding value
      return "the property needed" ;
    }
    else // default behaviour
    {
      return super.getProperty(property);
    }
  }

 /**
   * Send a message to the Forms module
   **/
  public void dispatch_event( ID id )
  {
      CustomEvent ce = new CustomEvent(mHandler, id);
      dispatchCustomEvent(ce);
  }

}
```

The parts displayed with a **bold red font** are those you have to adapt.

- Adapt the correct name of the bean (replace the `my_new_class` occurences with the name of your real class.
- Adapt the properties IDs to give them a more comprehensive sense.

Get a particular property

In our sample we want to get the background color of the canvas, so a good name for the corresponding property ID could be : **GETBGCOLOR**

You get the bean property from the Forms module through the **getProperty()** method wich is part of the Vbean class.

All what we need is to get the background color of the parent's component

This is the code of the **getProperty()** function:

```
    if (property == GETBGCOLOR)
    {
      // get the color of the parent container
      String sColor=""  ;
      Color color = this.getParent().getBackground() ;
      sColor = "r"+ color.getRed() +"g"+color.getGreen()+"b"+ color.getBlue()  ;
      // return the value
      return sColor ;
    }
```

Set a particular property

We also want to hide/show the Java Bean component, so we need to set its “visible” property.
This is done in the **setProperty()** method, trough the SETHIDDEN property ID.

The setProperty() method takes 2 arguments :
The property ID and an object that contains the value. (this value comes from the 4th argument of the Forms built-in Set_Custom_Property).

This is the code of the **setProperty()** function:

```
    if(property == SETHIDDEN) // Hide/Show the Java Bean component
    {
       // get the parameter string
       String sParam = (String)value ;
       if( sParam.equals("true") )
          setVisible(false);
       else
          setVisible(true);
       return true;
      }
```

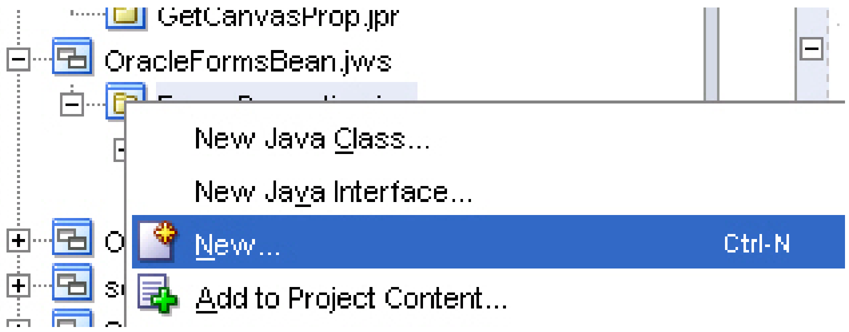
Then, the java code should, now, look like this:
(modified parts appear in a **bold font**)

```
package oracle.forms.demo;

import java.awt.Color;

import oracle.forms.handler.IHandler;
import oracle.forms.ui.CustomEvent;
import oracle.forms.properties.ID;
import oracle.forms.ui.VBean;

public class GetCanvasProp extends VBean

{
    static IHandler mHandler;
    // properties you can set
    protected static final ID SETHIDDEN    = ID.registerProperty("SETHIDDEN");
    // properties you can be get
    protected static final ID GETBGCOLOR   = ID.registerProperty("GETBGCOLOR");
    // events you can raise
    protected static final ID EVT_01   = ID.registerProperty("EVT_01");

    // default constructor
    public GetCanvasProp()
    {
      super();
    }

    public void init(IHandler handler)
    {
      super.init(handler);
      mHandler = handler;
      // put your initialisation instructions here
    }

 /**
   * Set the properties to the bean
   **/
  public boolean setProperty(ID property, Object value)
  {
    if(property == SETHIDDEN) // Hide/Show the Java Bean component
    {
       // get the parameter string
       String sParam = (String)value ;
```

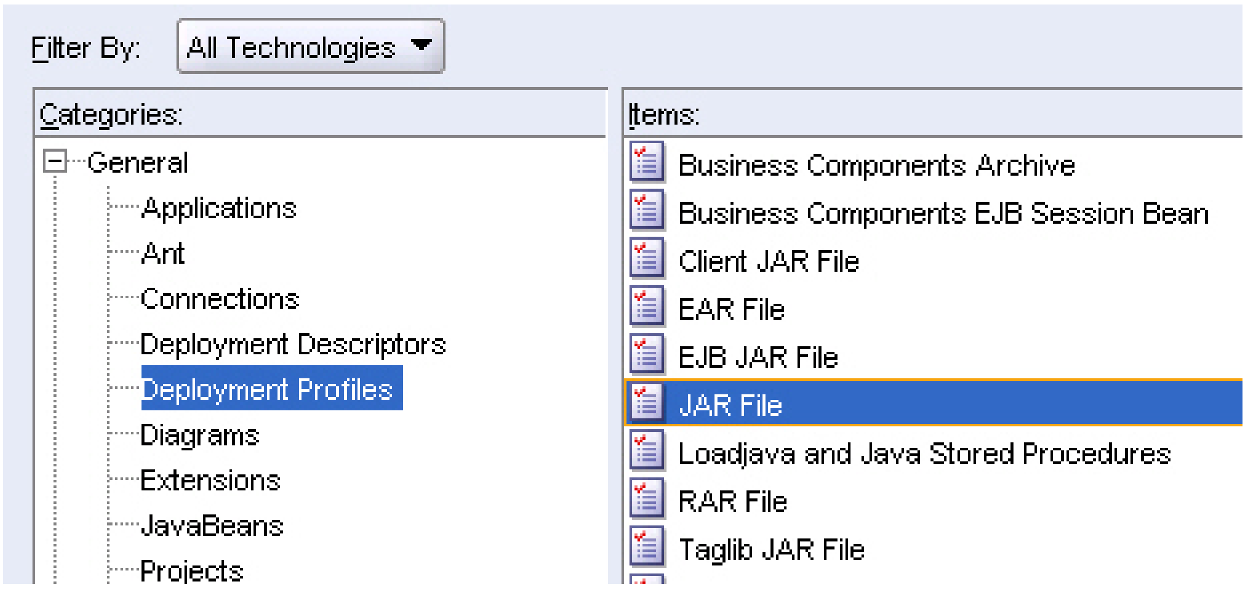
```
        if( sParam.equals("true") )
           setVisible(false);
        else
           setVisible(true);
        return true;
      }
     else // default behaviour
        {
           return super.setProperty(property, value);
        }
  }

 /**
   * Get the properties of the bean
   **/
  public Object getProperty(ID property)
  {
    if (property == GETBGCOLOR)
    {
      // get the color of the parent container
      String sColor=""  ;
      Color color = this.getParent().getBackground() ;
      sColor = "r"+ color.getRed() +"g"+color.getGreen()+"b"+ color.getBlue()  ;
      // return the value
      return sColor ;
    }
    else // default behaviour
    {
      return super.getProperty(property);
    }
  }

 /**
   * Send a message to the Forms module
   **/
  public void dispatch_event( ID id )
  {
      CustomEvent ce = new CustomEvent(mHandler, id);
      dispatchCustomEvent(ce);
  }

}
```

So, at this moment we need to deploy the .jar file to integrate it with the Forms application.

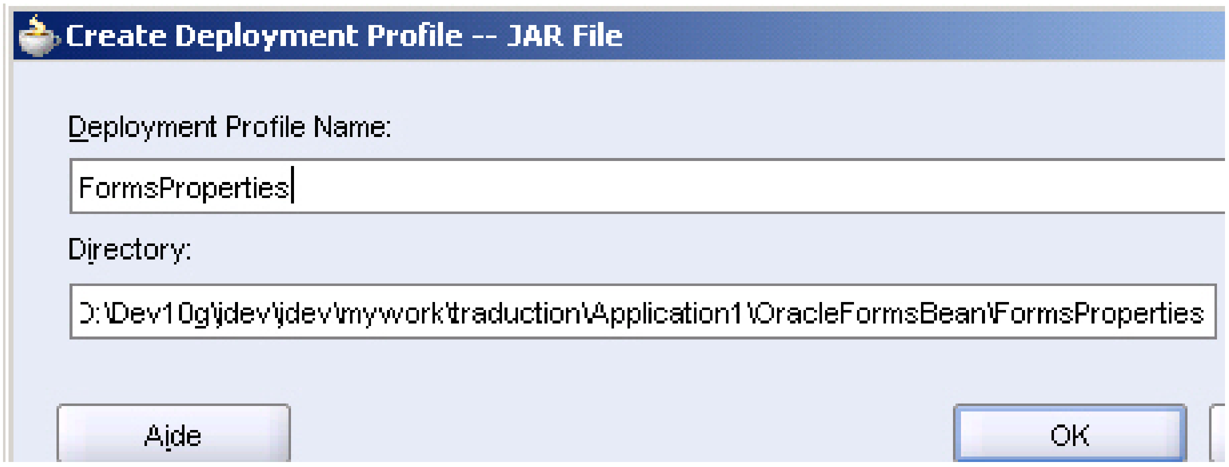
- Add a new Deployement Profile to the project that allows to generate and deploy the corresponding .jar file of the new javabean.
  Right-click the **OracleFormsBean** project node then click the **New** option



- Select the **Deployment Profiles** tab then the **JAR File** option

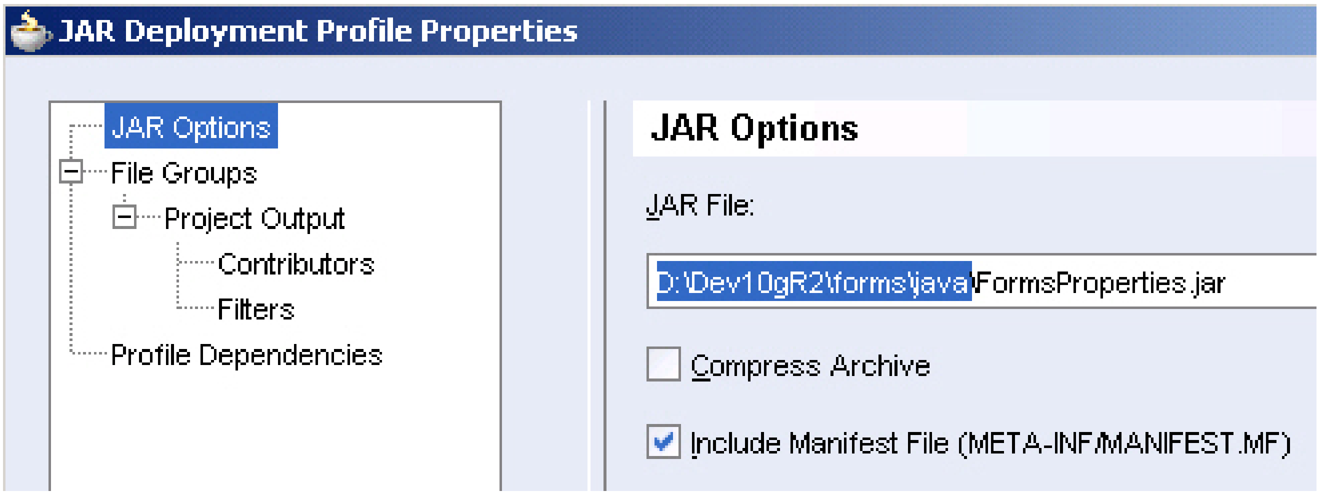


- Choose a name for this Deployment profile

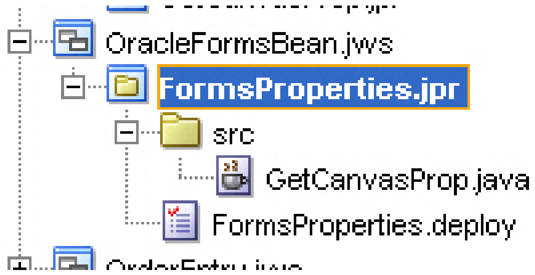


Deployment Profile Name:
FormsProperties
Directory:
D:\Dev10g\jdev\jdev\mywork\traduction\Application1\OracleFormsBean\FormsProperties

- Set the path where the .jar file will be copied at each deployment (I use to indicate the /forms/java directory)

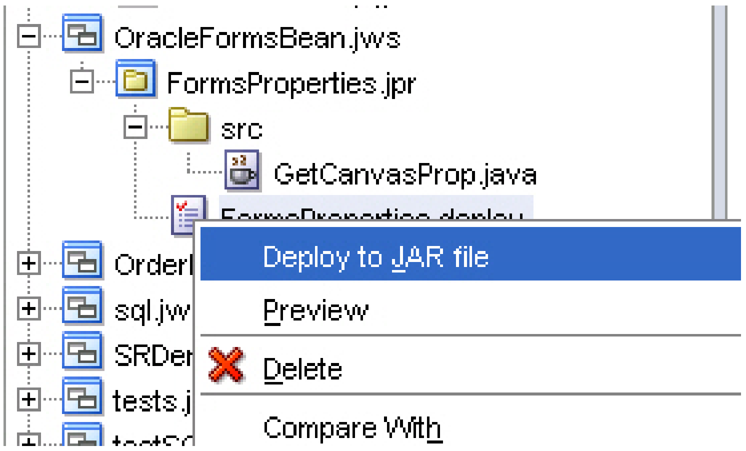


JAR File:
D:\Dev10gR2\forms\java\FormsProperties.jar
☐ Compress Archive
☑ Include Manifest File (META-INF/MANIFEST.MF)

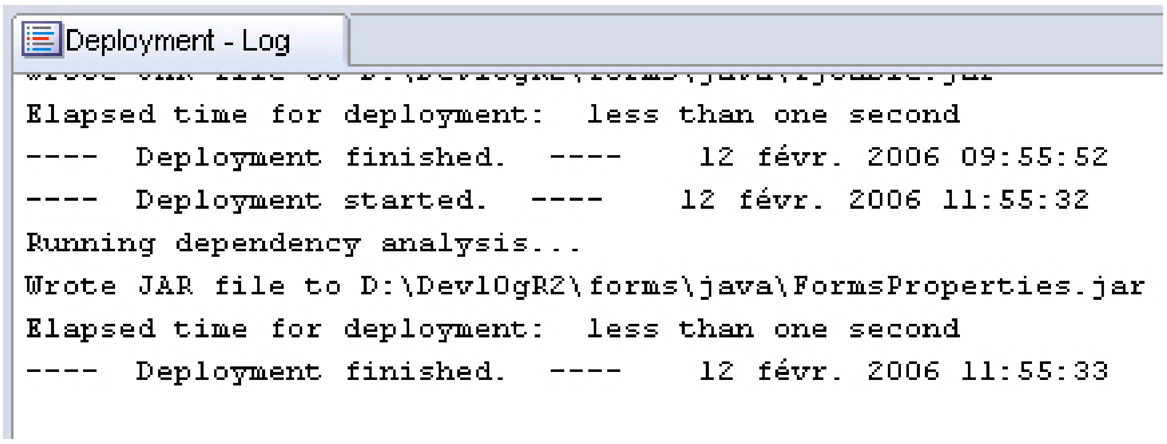


Then deploy the .jar file.

- Right-click the **FormsProperties.deploy** node then click the **Deploy to JAR file** option.



Be sure the .jar file is well deployed in the /forms/java directory.
If not, copy manually the .jar file into the forms/java directory.



```
Elapsed time for deployment:  less than one second
----  Deployment finished.  ----    12 févr. 2006 09:55:52
----  Deployment started.  ----    12 févr. 2006 11:55:32
Running dependency analysis...
Wrote JAR file to D:\Dev10gR2\forms\java\FormsProperties.jar
Elapsed time for deployment:  less than one second
----  Deployment finished.  ----    12 févr. 2006 11:55:33
```

4. Setup the Forms configuration file : formsweb.cfg

This file is located in your ***<ORACLE_HOME>/forms/server*** directory.

- Open this file and search for the **archive_jini** tag
- Add your .jar file to this tag.
- archive_jini=frmall_jinit.jar,...,**FormsProperties.jar**
- save the formsweb.cfg file.

5. Build the Forms test dialog

- Open the Forms Builder
- Create a new canvas
- Add a javabean item on the new canvas
- Set its Implementation Class property to : **oracle.forms.demo.GetCanvasProp** (remember the package and class name of your bean)

From now on you are able to set and get the properties of your javabean.

Set a property

To set a property of your javabean, use the **Set_Custom_Property()** built-in

```
Set_Custom_Property( ‘item_name’, record_number, ‘property_name’, ‘property_value’
) ;
```

On the When-Button-Pressed trigger, we set this property with the following code:

```
Set_Custom_Property( 'BL.BEAN_AREA', 1, 'SETHIDDEN', 'true' ) ;
```
or
```
Set_Custom_Property( 'BL.BEAN_AREA', 1, 'SETHIDDEN', 'false ) ;
```

Get a property

To get a property from your javabean, use the **Get_Custom_Property()** built-in

```
Varchar2 := Get_Custom_Property( ‘item_name’, record_number, ‘property_name’ ) ;
```

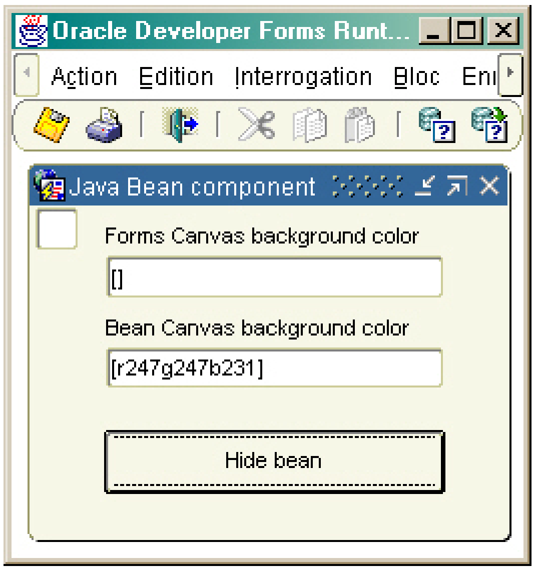
`item_name` is the name of the bean area item that support the Java Implementation Class.

In our sample, we want to get the background color through the GETBGCOLOR property ID, so the call to the built-in is the following:

```
BColor := Get_Custom_Property( 'BL1.BEAN_AREA', 1, 'GETBGCOLOR' ) ;
```

6. The sample dialog

- Download the first_bean.zip file
- Unzip the file
- Copy the FormsProperties.jar file in your /forms/java/ directory
- Edit the /forms/server/formsweb.cfg file to add the jar file to the **archive_jini** variable **archive_jini=**f90all_jinit.jar,......,**FormsProperties.jar**
- Save the formsweb.cfg file
- Open the **BEAN.fmb** dialog provided with the zip file.
- Compile and run it.


Oracle Developer Forms Runt...
Action Edition Interrogation Bloc En
Java Bean component

Forms Canvas background color

[]

Bean Canvas background color

[r247g247b231]

This is the code executed in the ***When-New-Form-Instance*** trigger:

```
Declare
      FColor  Varchar2(30) ;
      BColor  Varchar2(30) ;
Begin

      -- Get the Forms property --
      FColor := Get_Canvas_Property( 'CV1', BACKGROUND_COLOR ) ;
      :BL.EL1 := '[' || FColor || ']' ;

      -- Get the Javabean property --
      BColor := Get_Custom_Property( 'BL.BEAN_AREA', 1, 'GETBGCOLOR' ) ;
      :BL.EL2 := '[' || BColor || ']' ;
      If BColor is not null Then
        Set_Item_Property( 'BL.PUSH_BUTTON2', BACKGROUND_COLOR, BColor ) ;
      End if ;

End;
```

As you can see, the **Get_Canvas_Property()** built-in returns a NULL value, whereas the **Get_Custom_Property()** returns the correct value.
(my current colorscheme is BLAF at this very moment)

The Hide bean push button allows to hide/show the bean item.

You may get, of course, another value if you use a different colorscheme.

7. Forms and Java version

In order to run your javabean in the Forms application, you have to verify that the Compiler option of the Jdeveloper product is setted to **1.3**