AIM

MD.070 APPLICATION EXTENSION TECHNICAL DESIGN

Oracle Application Framework Extension

Create

Author:	Arone.zhang
Creation Date:	April 24, 1999
Last Updated:	March 5, 2009
Document Ref:	MD070_OAExtension _TB006
Version:	1.0

Approvals:

<Approver 1> Arone.zhang

<Approver 2> _____

Copy No. _____

Document Control

Change Record

Date	Author	Version	Change Reference
24-Apr-99	Arone.zhang	1.0	No Previous Document

Reviewers

Name	Position
Arone.zhang	Project Leader

Distribution

Copy No.	Name	Location
1	Library Master	Project Library
2		Project Manager
3		
4		

Note To Holders:

If you receive an electronic copy of this document and print it out, please write your name on the equivalent of the cover page, for document control purposes.

If you receive a hard copy of this document, please write your name on the front cover, for document control purposes.

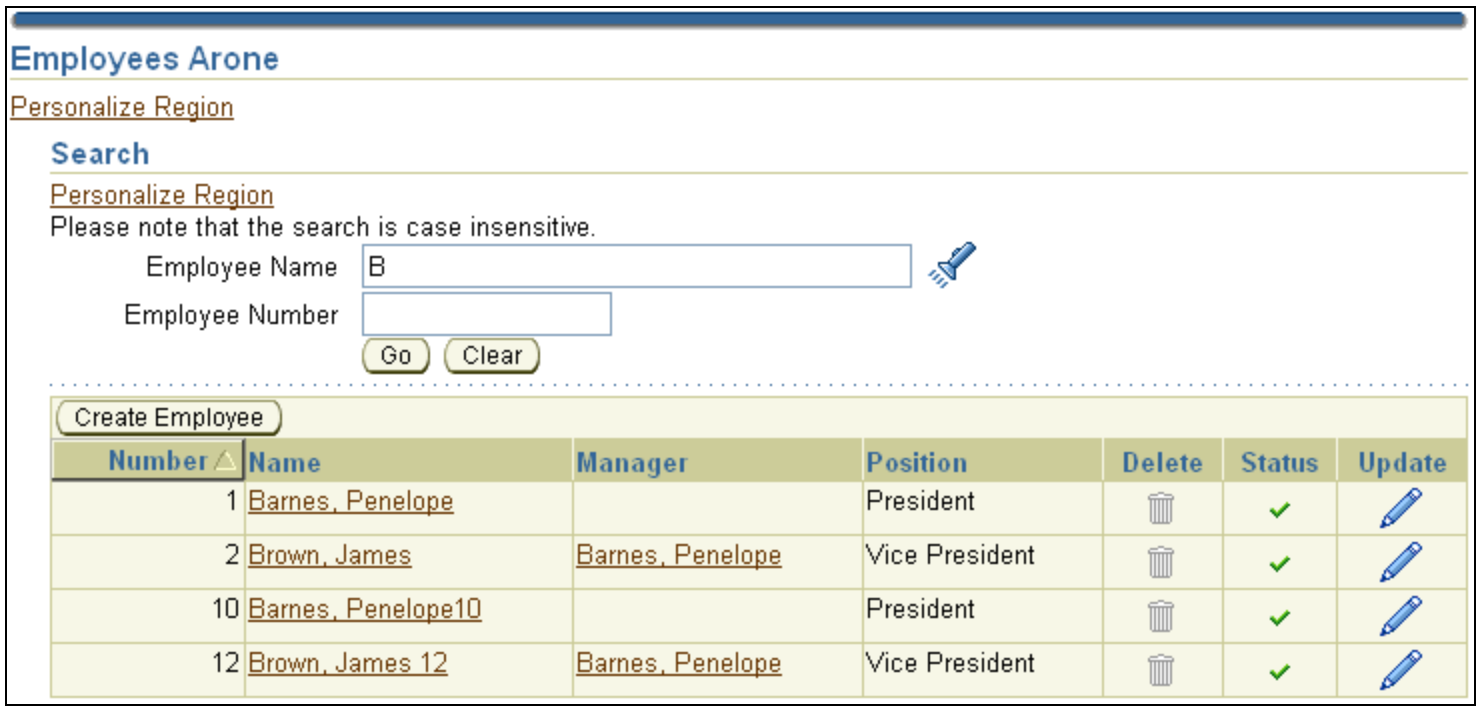
Contents

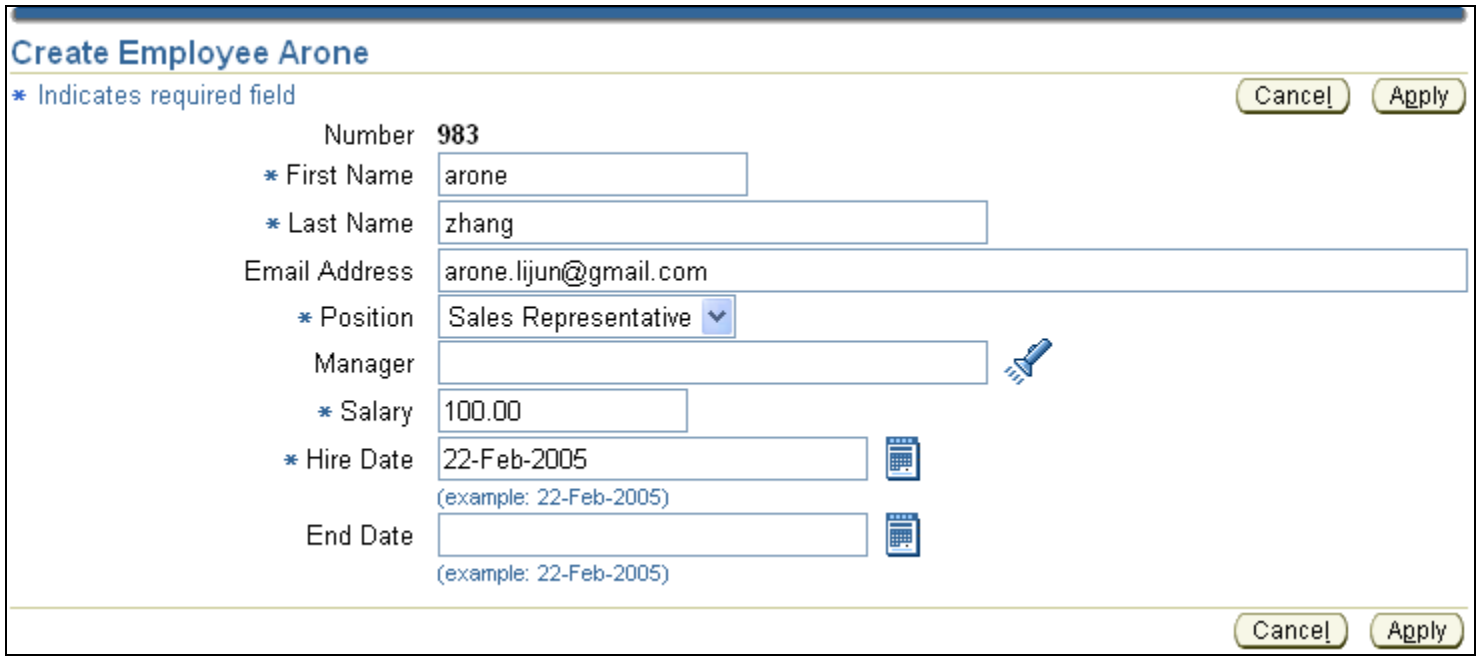
课程介绍

概述

本课程讲述了如何在 Employees 查询页中添加一个 Create Employee 按钮，并实现一个 Create 页面

前提：已经完成了 Search 和 Drilldown to Detail 的课程，如果还没有进行相关的学习，请参照之前的相关工作





课程的目标

完成此课程之后，需要掌握如下的课题：

- 添加一个全局表按钮
- 添加一个 "Indicates required field" 区域到页中，并指定必要的项目为必填项
- 创建一个下拉列表
- 添加页级别的动作/导航按钮
- 通过编写必要的程序实现页面直接的 JSP Forward

- 启用 "Warn About Changes" 功能，帮助用户减少不必要的数据丢失工作
- 显示一个确认对话框消息
- 实现实体对象、实体属性和行级的业务逻辑(包括抛出属性和行级的验证异常)
- 在业务逻辑中使用视图对象验证和应用模块验证
- 在基于实体对象的视图对象中添加一个新的行
- 处理 Apply 按钮的动作(实现提交)
- 确保应用程序在使用浏览器中的 Back 按钮是安全的

步骤1：修改查询页面，使其包括一个 Create Employee 按钮

任务1.1: 添加一个 “Create Employee” 按钮到查询页

首先，需要修改 **EmpSearchPG**，使结果表中包括一个 Create Employee 按钮。因此需要在结果表中添加一个 **tableAction** 组件

- 在结构窗口中 **ResultsTable** 区域，右键从上下文菜单中选择 **New > tableActions**
- JDeveloper 自动创建一个 **flowLayout** 区域，修改它的 ID 属性值为 **ButtonLayout**
- 选择 **ButtonLayout** 区域，右键选择 **New > Item** ，并设置如下的属性值：

属性	值
ID	Create

属性	值
Item Style Attribute Set	submitButton /oracle/apps/fnd/framework/toolbox/attributesets/FwkTbxEmployees/CreateEmployee

- 选择 **EmpSearchPG**，右键选择 **Run EmpSearchPG.xml** ，页面运行结果如下所示：

任务1.2: 添加 “Create Employee” 按钮处理逻辑

当用户按下 Create Employee submitButton 按钮，需要编写 processFormRequest() 方法来处理按钮被按下，然后导航到员工创建页面中。

首先，创建一个和 **ResultsTable** 区域关联的控制器

- 在 **EmpSearchPG** 中选择 **ResultsTable** ，右键从上下文菜单中选择 **Set New Controller...**
- 指定 **Package Name** 的值为 **oracle.apps.ak.employee.webui**，**Class Name** 的值为 **EmployeeResultsCO**
- 确定创建控制器

然后添加如下的代码到 EmployeeResultsCO.processFormRequest() 方法中

```
import oracle.apps.fnd.framework.webui.OAWebBeanConstants;
...

 public void processFormRequest(OAPageContext pageContext, OAWebBean webBean)
 {
  super.processFormRequest(pageContext, webBean);
```

```
  if (pageContext.getParameter("Create") != null)
  {

   // Navigate to the "Create Employee" page while retaining the AM.
   // Note the use of KEEP_MENU_CONTEXT as opposed to GUESS_MENU_CONTEXT
   // since we know the current tab should remain highlighted.

   pageContext.setForwardURL("OA.jsp?page=/oracle/apps/ak/employee/webui/EmployeePG",
                        null,
                        OAWebBeanConstants.KEEP_MENU_CONTEXT,
                        null,
                        null,
                        true, // Retain AM
                        OAWebBeanConstants.ADD_BREAD_CRUMB_YES,
                        OAWebBeanConstants.IGNORE_MESSAGES);
  }
 }
```

任务1.3: 保存编译

步骤2：为 Positions Poplist 创建一个视图对象

Create Employee 页面需要为选择员工的职位创建一个 Poplist 。在创建页面之前，需要为 Poplist 的数据创建一个视图对象

任务2.1: 为 Poplist 视图对象创建 BC4J包

按照 OA Framework File 标准，为了 Poplist 而显示创建的视图对象必须在特殊的产品级的目录中

- 在 **SearchOAProject.jpr** 项目中创建一个包名为 **oracle.apps.ak.poplist.server** 的 **business components package**

任务2.2: 在 Poplist BC4J包中创建视图对象 PositionsVO

- 在 **oracle.apps.ak.poplist.server** 包中创建一个新的视图对象 **PositionsVO**
- 一个 poplist 视图对象需要包括两个属性列：一个为显示值，一个为选择的内部值。使用下列查询语句(meaning为显示值，lookup_code 为开发内部使用的值)

```
SELECT meaning, lookup_code
FROM fwk_tbx_lookup_codes_vl
WHERE lookup_type = 'FWK_TBX_POSITIONS'
```

- 不要产生 *VOImpl 文件，产生 *VORowImpl 文件

任务2.3: 添加 PositionsVO 视图对象到应用模块 EmployeeAM

- 添加 **PositionsVO** 到 **EmployeeAM**

步骤3：创建 Create 页

此步骤的任务是创建一个如下的员工 Create 页面。

任务3.1: 创建 EmployeePG 页

- 在包 **oracle.apps.ak.employee.webui** 中创建一个新的 OA Components 页面 EmployeePG

任务3.2: 更改页面布局区域

- 设置和验证如下的属性值

属性	值
ID	PageLayoutRN
Region Style	pageLayout
AM Definition	oracle.apps.ak.employee.server.EmployeeAM
Window Title	Framework ToolBox Tutorial: Labs Arone
Title	Create Employee Arone
Warn About Changes	True
AutoFooter	True

任务3.3: 添加产品商标图片

每个 Oracle 应用页面都要求有一个商标图片

- 在结构窗口中选择 **PageLayoutRN** 区域，右键从上下文菜单中选择 **New…productBranding**
- JDeveloper 会自动创建一个包括了**productBranding**(名为 **Item1**) 图形项目的 **pageLayoutComponents** 文件夹，选择其中的项目进行下面的属性设置

属性	值
ID	ProdBrand
Image URI	FNDTAPPBRAND.gif
Additional Text	OA Framework ToolBox Tutorial Arone

任务3.4: 添加页级别的 Apply 和 Cancel 按钮

按照 BLAF UI 的规则，所有页级按钮都会被显示两次：页标题下面和 ski 下面，然而开发中只需要使用特殊的 **pageButtonBar** 区域添加一次。

- 在结构窗口中选择 **PageLayoutRN** 区域，右键从上下文菜单中选择 **New > Region**
- 设置如下的属性值：

属性	值
ID	PageButtons
Item Style	pageButtonBar

- 增加 Cancel 按钮，在结构窗口中选择 **PageButtons**，右键从上下文菜单中选择 **New > Item**，设置和验证如下的属性值：

属性	值
ID	Cancel
Item Style	submitButton
Attribute Set	/oracle/apps/fnd/attributesets/Buttons/Cancel
Disable Server Side Validation	True (用户可以选择此按钮来退出页面，而不会出现任何的服务端验证错误)
Disable Client Side Validation	True (用户可以选择此按钮来退出页面，而不会出现任何的客户端验证错误)
Prompt	Cancel (指定属性集的时候自动指定)

属性	值
Warn About Changes	False (用户可以选择此按钮来退出页面，而不会出现任何的未决变化警告信息)
Additional Text	Select to cancel this transaction.

- 增加 Apply 按钮，在结构窗口中选择 **PageButtons**，右键从上下文菜单中选择 **New > Item**，设置和验证如下的属性值：

属性	值
ID	Apply
Item Style	submitButton
Attribute Set	/oracle/apps/fnd/attributesets/Buttons/Apply
Prompt	Apply (属性集自动指定)
Additional Text	Select to save this employee.

任务3.5: 创建主内容区域

为了使项目显示出合适的缩进，在 **PageLayoutRN** 区域中添加 **defaultSingleColumn** 区域，由于此区域中所有的项目都和 **EmployeeFullVO1** 视图实例相对应，使用区域向导快速创建。

- 在结构窗口中选择 **PageLayoutRN** 区域，右键从上下文菜单中选择 **New > Region Using Wizard**
- 跳过欢迎页
- 在 **BC4J Objects** 页中选择 **EmployeeAM(oracle.apps.ak.employee.server.EmployeeAM)**，并从 **Available View Objects** 列表中选择 **EmployeeFullVO1** 视图对象实例
- 选择 下一步 到 **Region Properties** 页，设置 **Region ID** 的值为 **MainRN**；**Region Style** 的值为 **defaultSingleColumn**

- 下一步到 **View Attributes** 页中，从 **Available View Attributes** 列表中选择如下属性列到 **Selected View Attributes** 列表中

  EmployeeId
  FirstName
  LastName
  EmployeeEmail
  PositionCode
  ManagerName
  Salary
  StartDate
  EndDate
  ManagerID

- 选择 下一步 到 **Region Items** 页，设置如下的项目属性：

ID	Prompt	Style	Data Type	Attribute Set
EmpNum	Number	messageStyledText	NUMBER	/**< base path>**/EmployeeId_Number
FirstName	First Name	messageTextInput	VARCHAR2	/**< base path>**/FirstName
LastName	Last Name	messageTextInput	VARCHAR2	/**< base path>**/LastName
EmailAddress	Email Address	messageTextInput	VARCHAR2	/**< base path>**/EmailAddress
Position	Position	messageChoice	VARCHAR2	/**< base path>**/Position
MgrName	Manager	messageLovInput	VARCHAR2	/**< base path>**/FullName_Manager
MgrNum	Manager Number	messageStyledText	NUMBER	
Salary	Salary	messageTextInput	NUMBER	/**< base path>**/Salary
StartDate	Start Date	messageTextInput	DATE	/**< base path>**/StartDate
EndDate	End Date	messageTextInput	DATE	/**< base path>**/EndDate

<base patch> 为 **/oracle/apps/fnd/framework/toolbox/attributesets/FwkTbxEmployees**

- 确定创建数据表

任务3.6: 设置 MainRN 区域的属性值

大部分 **MainRN** 区域的属性在创建 **defaultSingleColumn** 的时候就已经设置好了，现在需要进一步设置属性值

打开 **MainRN** 区域下的项目进行如下的属性值设置：

EmpNum

此字段值通过使用数据库序列自动生成一个新的员工号码，因此这个值为只读

属性	值
CSS Class	OraDataText

FirstName

对于文本域来说，Maximum Length 属性值对应了数据库列的长度。Length 属性值为页面上显示出来的长度。

如果将 Required 属性的值设置为 **yes**，一个必填指示符会出现在提示前面。当页面被提交的时候，在 POST 提交服务器之前 UIX 进行非空值验证。

属性	值
Required	yes
Maximum Length	20
CSS Style	OraFieldText
Length	20

LastName

属性	值

属性	值
Required	yes
Maximum Length	40
CSS Style	OraFieldText
Length	40

EmailAddress

属性	值
Required	no
Maximum Length	240
CSS Style	OraFieldText
Length	80

Position

Position 项目是 poplist，上面已经为项目指定了数据源(视图对象中的 PositionCode)

属性	值
Required	yes
Picklist View Definition	oracle.apps.ak.poplist.server.PositionsVO
Picklist Display Attribute	Meaning
Picklist Value Attribute	LookupCode
CSS Class	OraFieldText

MgrName

主管的域需要一个值列表(LOV)，可以复用在 Search 课程中创建的 EmployeesLovRN ，这个字段没有标示为必填，它是有条件的必填项，在后面可以看到，所以需要在实体对象层次实现验证。

属性	值
Required	no
External LOV	/oracle/apps/ak/lov/webui/EmployeesLovRN
Maximum Length	240
CSS Class	OraFieldText
Length	40
LOV Mappings	创建如下的 LOV 映射： LOV Map1 ID：**ToFromMgrName** Lov Region Item: **EmpName** Return Item: **MgrName** Criteria Item: **MgrName** LOV Map2 ID：**ToMgrNum** Lov Region Item: **EmpNumber** Return Item: **MgrNum**

Salary

属性	值
Required	yes
Maximum Length	15
Initial Value	68320.99
CSS Style	OraFieldText
Length	15

StartDate

属性	值
Required	yes
Maximum Length	30
CSS Style	OraFieldText
Length	30
Tip type	dateFormat

EndDate

属性	值
Required	no
Maximum Length	30
CSS Style	OraFieldText
Length	30
Tip type	dateFormat

任务3.7: 改变区域 MainRN样式 defaultSingleColumn 到 messageComponentLayout

- 选择 **MainRN defaultSingleColumn** 区域，改变它的 Region Style 属性值为 **messageComponentLayout** 。JDeveloper 将显示如下的警告信息，选择 **是** 继续，区域中的项目不会受到影响。
- 选择 **messageComponentLayout** 区域，右键选择 **New > messageLayout**
- 设置 **messageLayout** ID 属性值为 **MgrNumLayout**
- 选择 **MgrNum** 项目到 **messageLayout** 下面
- 改变 **MgrNum** Item Style 属性值为 **formValue**

为了LOV配置，隐藏开发员需要的内部关键值 **formValue** 项目，因此为MgrNum 项目创建了一个而外的区域，只有 **message*** 项目才能够直接加入到 **messageComponentLayout** 区域中

为了学习更多的 **messageComponentLayout** 区域知识，请查看 OA Framework 开发员手册中的页布局(如何放置内容)

任务3.8: 添加 "标示必填" 区域

“标示必填” 区域和页级别的按钮显示在同一行。为了达到此目的，添加一个页级别的关键组件 **pageStatus** 。

- 选择 **PageLayoutRN** 区域，右键从选择 **New > pageStatus**
- 选择 **pageStatus** 下 JDeveloper 创建的 **flowLayout** 区域，设置 ID 的属性值为 PageStatusRN
- 选择 **PageStatusRN** ，右键选择 **New > Region** 。为新建的区域设置如下的属性值(任何时候需要一个 BLAF 标准的”标示必填”，只要简单的扩展如下所示的通用 OA Framework 区域即可)：

属性	值
ID	RequiredKey
Extends	/oracle/apps/fnd/framework/webui/OAReqFieldDescRG
Width	100%

任务3.9: 保存测试

步骤4：实现浏览器后退按钮安全页初始化和创建/回滚方法

任务4.1: 添加 createEmployee() 方法到 EmployeeAMImpl 类中

添加如下的方法到 **EmployeeAMImpl** 类中实现在 **EmployeeFullVO** 视图对象下创建一个新员工

```
import oracle.jbo.Row;
import oracle.apps.fnd.framework.OAViewObject;
...
```

```
/*
 ********************************************************************************
 * Creates a new employee.
 ********************************************************************************
*/
public void createEmployee()
{
 OAViewObject vo = (OAViewObject)getEmployeeFullVO1();

 // Per the coding standards, this is the proper way to initialize a
 // VO that is used for both inserts and queries.  See View Objects
 // in Detail in the Developer's Guide for additional information.
  if (!vo.isPreparedForExecution())
 {      vo.executeQuery();    }

 Row row = vo.createRow();    vo.insertRow(row);
// Required per OA Framework Model Coding Standard M69
row.setNewRowState(Row.STATUS_INITIALIZED);

} // end createEmployee()
```

任务4.2: 添加 rollbackEmployee() 方法到 EmployeeAMImpl 类中

添加如下的方法到 **EmployeeAMImpl** 类中实现回滚数据库和中间层。

调用 **oracle.jbo.Transaction** 中的 **getTransaction()** 方法，取代调用 **oracle.apps.fnd.framework.server.OADBTransaction** 中的 **getOADBTransaction()** 方法。

```
import oracle.jbo.Transaction;
...
```

```
/*
********************************************************************************
* Executes a rollback including the database and the middle tier.
********************************************************************************
*/
public void rollbackEmployee()
{
  Transaction txn = getTransaction();
  // This small optimization ensures that we don't perform a rollback
  // if we don't have to.
  if (txn.isDirty())
  {
    txn.rollback();
  }
} // end rollbackEmployee()
```

任务4.3: 添加回退按钮处理到 EmployeeResultsCO processRequest()

为了解决 Create Employee 和 Search 页之间的各种回退按钮导航操作，添加如下的处理逻辑到 **EmployeeResultsCO** 控制器中的 **processRequest()** 方法中，这样保证了当用户使用回退按钮到 Search 页中的时候，任何未完成的员工对象都将从中间服务器层缓冲清除，然后再重新创建一个新的员工。

提示：当开始开发应用程序时，可以通过 OA Framework开发员手册学习更多关于浏览器回退按钮的相关内容。

```
import oracle.apps.fnd.framework.webui.TransactionUnitHelper;
import oracle.apps.fnd.framework.OAApplicationModule;
...
public void processRequest(OAPageContext pageContext, OAWebBean webBean)
{
  super.processRequest(pageContext, webBean);
  OAApplicationModule am = pageContext.getApplicationModule(webBean);
```

```
  // The following checks to see if the user navigated back to this page
  // without taking an action that cleared an "in transaction" indicator.
  // If so, we want to rollback any changes that she abandoned to ensure they
  // aren't left lingering in the BC4J cache to cause problems with
  // subsequent transactions.  For example, if the user navigates to the
  // Create Employee page where you start a "create" transaction unit,
  // then navigates back to this page using the browser Back button and
  // selects the Create Employee button again, the OA Framework detects this
  // Back button navigation and steps through processRequest() so this code
  // is executed before you try to create another new employee.

  if (TransactionUnitHelper.isTransactionUnitInProgress(pageContext, "empCreateTxn", false))
  {
   am.invokeMethod("rollbackEmployee");
   TransactionUnitHelper.endTransactionUnit(pageContext, "empCreateTxn");
  }
}
```

任务4.4: 添加 Create 页初始化

本页包括了使用户使用浏览器的回退按钮过程中不会遇到错误信息的处理逻辑。例如，它保证了在创建新员工的过程中，在中间层的缓冲中不会存在已创建部分 **EmployeeEO** 的对象。

首先，在 **EmployeePG** 中创建一个和 **pageLayoutRN** 区域相关联的控制器

- 选择 **EmployeePG** 中的 **pageLayoutRN** 区域，右键从上下文菜单中选择 **Set New Controller**
- 指定 **Package Name** 值为 **oracle.apps.ak.employee.webui**
- 指定 **Name** 值为 **EmployeeCreateCO**
- 选择 **确定** 按钮创建控制器

然后，添加如下的代码到 processRequest() 方法中供页面初始化的时候调用。

```
import oracle.apps.fnd.framework.OAApplicationModule;
import oracle.apps.fnd.framework.webui.OADialogPage;
import oracle.apps.fnd.framework.webui.TransactionUnitHelper;
...

public void processRequest(OAPageContext pageContext, OAWebBean webBean)
{
  // Always call this first.   super.processRequest(pageContext, webBean);
  // If isBackNavigationFired = false, we're here after a valid navigation
  // (the user selected the Create Empoyee button) and we should proceed
  // normally and initialize a new employee.

  if (!pageContext.isBackNavigationFired(false))
  {
   // We indicate that we are starting the create transaction (this
   // is used to ensure correct Back button behavior).

    TransactionUnitHelper.startTransactionUnit(pageContext, "empCreateTxn");

   // This test ensures that we don't try to create a new employee if
   // we had a JVM failover, or if a recyled application module
   // is activated after passivation. If this things happen, BC4J will
   // be able to find the row that you created so the user can resume
   // work.

   if (!pageContext.isFormSubmission())
   {
    OAApplicationModule am = pageContext.getApplicationModule(webBean);
    am.invokeMethod("createEmployee", null);

    // Iniitalize the ApplicationPropertiesVO for PPR.
    am.invokeMethod("init");
   }
  }
  else
  {
   if (!TransactionUnitHelper.isTransactionUnitInProgress(pageContext, "empCreateTxn",    true))
   {
    // We got here through some use of the browser "Back" button, so we
```

```
      // want to display a stale data error and disallow access to the page.
      // If this were a real application, we would probably display a more
      // context-specific message telling the user she can't use the browser
      // "Back" button and the "Create" page. Instead, we wanted to illustrate
      // how to display the Applications standard NAVIGATION ERROR message.

      OADialogPage dialogPage = new OADialogPage(NAVIGATION_ERROR);
      pageContext.redirectToDialogPage(dialogPage);
    }
  }
} // end processRequest()
```

任务4.5: 测试代码

- 选择 **SearchOAProject.jpr** 项目，右键选择 **Rebuild SearchOAProject.jpr**
- 添加如下的断点到：
  - **EmployeeAMImpl** 类中 **createEmployee**() 方法的第一执行行
  - **EmployeeEOImpl** 类中 **create**() 方法的第一执行行
  - **EmployeeAMImpl** 类中 **rollbackEmployee**() 方法的第一执行行
  - **EmployeeAMImpl** 类中 **rollbackEmployee**() 方法的第一执行行
  - **EmployeeCreateCO** 类中 **processRequest**() 方法的第一执行行
  - **EmployeeResultsCO** 类中 **processRequest**() 方法的第一执行行
  - **EmployeeResultsCO** 类中 **processFormRequest**()方法的第一执行行
- 选择 **EmpSearchPG**，右键选择 **Debug EmpSearchPG.xml**

1. 当 Search 页显示的时候，选择 Create Employee 按钮，逐步执行 EmployeeCreateCO 类中的 **processRequest**()
2. 使用浏览器中的回退按钮回到 Search 页面
3. 从新选择 Create Employee 按钮

- 试着使用回退按钮回到Search页面，验证页面显示的时候没有任何错误发生

步骤5：实现员工业务逻辑(声明式验证和初始化)

下表汇总处理在本系列课程中实现的员工业务逻辑规则。黑体的部分可以通过声明式验证来实现

属性对象	业务规则
EmployeeId	当创建员工实体对象时使用FWK_TBX_EMPLOYEES_S序列自动生成一个唯一关键字 在整个系统中保持唯一性 **必须，不能为空** **提交后能够被更新**
FirstName	**必须，不能为空**
LastName	**必须，不能为空**
FullName	通过程序来组合 LAST_NAME 和 FIRST_NAME 只要 LAST_NAME 或 FIRST_NAME 发生变化都必须更新
ManagerId	除 positionCode 为 PRESIDENT 之外的员工都是必须 必须指向一个有效的员工(有效意味着员工在表中已创建，且

属性对象	业务规则
	end_date 为空)
PositionCode	必须为有效的职位(有效意味着在表 FWK_TBX_LOOKUP_CODES_B 中存在)
StartDate	当员工实体对象创建的时候，默认值为 SYSDATE 必须 >= SYSDATE **必须，不能为空** **已经提交的行不能被更新**
EndDate	如果指定值，必须 >= START_DATE 必须 >= SYSDATE
Salary	**必须，不能为空** 必须 > 0

任务5.1: 设置声明式验证

为了实现业务规则，从声明式验证开始设置。

- 在导航窗口中选择 **EmployeeEO**，右键从上下文菜单中选择 **Edit EmployeeEO ...**
- 选择 **Attributes**，选择 **<attribute name>** 设置如下的属性值：

属性对象	业务规则
EmployeeId	选中 **Mandatory** 和 **Updateable While New**
FirstName	选中 **Mandatory**
LastName	选中 **Mandatory**
StartDate	选中 **Mandatory** 和 **Updateable While New**
Salary	选中 **Mandatory**

任务5.2: 增加导入声明

在 **EmployeeEOImpl** 类中添加如下的导入声明：

```
import oracle.jbo.server.EntityDefImpl;
import oracle.jbo.server.AttributeDefImpl;
import oracle.jbo.AttributeList;
import oracle.jbo.domain.Number;
import oracle.jbo.domain.Date;import oracle.jbo.Key;
import oracle.apps.fnd.framework.OAAttrValException;
import oracle.apps.fnd.framework.OARowValException;
import oracle.apps.fnd.framework.server.OADBTransaction;
import oracle.apps.fnd.framework.OAException;
import oracle.apps.fnd.framework.server.OAEntityImpl;
import oracle.jbo.RowIterator;
import oracle.jbo.server.EntityImpl;
```

在实体对象的上下文中使用 **OADBTransaction**， 因为超类 **OAEntityImpl** 提供了一个方便的方法 **getOADBTransaction()** 供我们使用

任务5.3: 覆盖 setEmployeeId() 方法

在 **EmployeeEOImpl** 类中添加如下的验证逻辑到 **setEmployeeId()** 方法中。验证确保员工的ID是唯一的(检查实体缓冲和数据库)，如果违反规则抛出属性级别验证的错误信息

```
/*
  ********************************************************************************
  * Sets <code>value</code> as the attribute value for EmployeeId
```

```
 ********************************************************************************
*/
public void setEmployeeId(Number value)
{
  // Because of the declarative validation that you specified for this attribute,
  // BC4J validates that this can be updated only on a new line, and that this
  // mandatory attribute is not null. This code adds the additional check
  // of only allowing an update if the value is null to prevent changes while
  // the object is in memory.

  if (getEmployeeId() != null)
  {
    throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
        getEntityDef().getFullName(), // EO name
        getPrimaryKey(), // EO PK
        "EmployeeId", // Attribute Name
        value, // Attribute value
        "AK", // Message product short name
        "FWK_TBX_T_EMP_ID_NO_UPDATE"); // Message name
  }

  if (value != null)
  {
    // Employee id must be unique. To verify this, you must check both the
    // entity cache and the database. In this case, it's appropriate
    // to use findByPrimaryKey() because you're unlikely to get a match, and
    // and are therefore unlikely to pull a bunch of large objects into memory.
    // Note that findByPrimaryKey() is guaranteed to check all employees.
    // First it checks the entity cache, then it checks the database.

    OADBTransaction transaction = getOADBTransaction();
    Object[] employeeKey = {value};
    EntityDefImpl empDefinition = EmployeeEOImpl.getDefinitionObject();
    EmployeeEOImpl employee =
      (EmployeeEOImpl)empDefinition.findByPrimaryKey(transaction, new Key(employeeKey));

    if (employee != null)
    {
      throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
```

```
          getEntityDef().getFullName(), // EO name
          getPrimaryKey(), // EO PK
          "EmployeeId", // Attribute Name
          value, // Attribute value
          "AK", // Message product short name
          "FWK_TBX_T_EMP_ID_UNIQUE"); // Message name
    }
  }

  // Note that this is the point at which the value is actually set on the EO cache
  // (during the scope of setAttributeInternal processing).  If you don't make this
  // call after you perform your validation, your value will not be set correctly.
  // Also, any declarative validation that you define for this attribute is executed
  // within this method.

  setAttributeInternal(EMPLOYEEID, value);

 } // end setEmployeeId()
```

任务5.4: 添加验证逻辑到 setStartDate() 方法；添加 validateStartDate() 方法

在 **EmployeeEOImpl** 类中添加如下的验证逻辑到 **setStartDate()** 方法，在此我们委派 **validateStartDate()** 方法来将验证逻辑放到实体层次，因此我们将属性值的设置和验证逻辑分开。

```
/*
  ********************************************************************************
  * Sets <code>value</code> as the attribute value for StartDate
  ********************************************************************************
 */
 public void setStartDate(Date value)
 {
   validateStartDate(value);
   setAttributeInternal(STARTDATE, value);
```

```
} // end setStartDate()
```

由于需要保证 **EmployeeEOImpl** 的子类能调用 **validateStartDate()** 方法，因此将其设置为 **protected** 。此方法演示了使用 oracle.jbo.domain.Date 正确进行对比处理

```
/*
 ***************************************************************************
 * Verifies that the employee's Start Date is valid.
 *
 * Business Rules:
 * Start Date is required.
 * Cannot be earlier than sysdate.
 ***************************************************************************
*/
protected void validateStartDate(Date value)
{

  // BC4J ensures that this mandatory attribute has a non-null value.
  if (value != null)
  {

    OADBTransaction transaction = getOADBTransaction();

    // Note that we want to truncate these values to allow for the possibility
    // that we're trying to set them to be the same day. Calling
    // dateValue( ) does not include time. Were we to want the time element,
    // we would call timestampValue(). Finally, you cannot compare
    // oracle.jbo.domain.Date objects directly. Instead, convert the value to
    // a long as shown.
    long sysdate = transaction.getCurrentDBDate().dateValue().getTime();
    long startDate = value.dateValue().getTime();

    if (startDate < sysdate)
    {
      throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
```

```
        getEntityDef().getFullName(), // EO name
        getPrimaryKey(), // EO PK
        "StartDate", // Attribute Name
        value, // Attribute value
        "AK", // Message product short name
        "FWK_TBX_T_START_DATE_PAST"); // Message name
    }

  }

 } // end validateStartDate()
```

任务5.5: 覆盖 create() 方法

当 **EmployeeEOImpl** 类被初始化的时候 **create()** 方法被调用。任何编写默认的逻辑都应该包括此方法。在此实例中，基于数据库序列设置员工的 ID； 设置开始日期为数据库的 SYSDATE

可以调用 set<AttributeName>() 方法并传入想更改的值。

```
/*
 ********************************************************************************
 * Add attribute defaulting logic here.
 ********************************************************************************
 */
 public void create(AttributeList attributeList)
 {
```

```
    super.create(attributeList);

    OADBTransaction transaction = getOADBTransaction();
    Number employeeId = transaction.getSequenceValue("FWK_TBX_EMPLOYEES_S");
    setEmployeeId(employeeId);

    // Start date should be set to sysdate
    setStartDate(transaction.getCurrentDBDate());

  } // end create()
```

任务5.6: 测试页面

步骤6：实现员工业务逻辑(员工名称处理)

现在我们继续来实现 **FirstName** 和 **LastName** 属性变化的业务规则。在每个方法中，检查新的值是否和旧的值相等，如果不相等则组合 **FirstName** 和 **LastName** 的值成为新的 **FullName** 的值，然后调用 **FullName** 设置器。

任务6.1: 添加验证逻辑到 setFirstName() 方法中

添加如下的代码到 **EmployeeEOImpl** 类中的 **setFirstName()** 方法中

```
/*
 ********************************************************************************
 * Sets <code>value</code> as the attribute value for FirstName
 ********************************************************************************
 */
 public void setFirstName(String value)
 {

  // BC4J will handle making sure this value is not null. We still need to
  // check before trying to set the full name, however.
  if ((value != null) || (!("".equals(value.trim()))))
  {
   String oldFirstName = getFirstName();
   if (oldFirstName == null)
   {
    oldFirstName = "";
   }

   // If we're dealing with a new last name value, update the full name.
   if (value.compareTo(oldFirstName) != 0)
   {
    String lastName = getLastName();
    if (lastName == null)
    {
     lastName = "";
    }
    setFullName(lastName.concat(", ").concat(value));
   }

  }

  setAttributeInternal(FIRSTNAME, value);

 } // end setFirstName()
```

任务6.2: 添加验证逻辑到 setLastName() 方法中

添加如下的代码到 **EmployeeEOImpl** 类中的 **setLastName()** 方法中

```
/*
 ***************************************************************************
 * Sets <code>value</code> as the attribute value for LastName
 ***************************************************************************
 */
 public void setLastName(String value)
 {
  // BC4J will handle making sure this value is not null. We still need to
  // check before trying to set the full name, however.
  if ((value != null) || (!("".equals(value.trim()))))
  {
   String oldLastName = getLastName();
   if (oldLastName == null)
   {
    oldLastName = "";
   }

   // If we're dealing with a new last name value, update the full name.
   if (value.compareTo(oldLastName) != 0)
   {
    String firstName = getFirstName();
    if (firstName == null)
    {
     firstName = "";
    }
    setFullName(value.concat(", ").concat(firstName));
   }

  }

  setAttributeInternal(LASTNAME, value);
```

```
} // end setLastName()
```

任务6.3: 保存测试

在实现提交之前，按 Apply 按钮之后员工的数据并不会被保存到数据库中

一个被更改值的页面被提交之后，OA Framework 依次调用实体对象下的设置器值回写视图对象，在按了没有提交的 Apply 按钮后，数据被存储在中间层的实体对象缓存中，而非数据库中。

步骤7：实现员工业务逻辑(实体专家，VAM和VVO验证)

在此步骤中，将创建 ManagerId 和 PositionCode 验证使用的一个实体专家，一个应用模块验证和几个视图对象验证。

任务7.1: 创建员工验证应用模块

创建一个应用模块来组织 EmployeeEO 和它的实体专家所需要的视图对象验证逻辑。UI 不需要使用验证应用模块，因此可以和实体对象创建在同一个包中。

- 在 **oracle.apps.ak.schema.server** BC4J 包中创建一个新的应用模块 **EmployeeVAM**
- 在 **Java** 页中，反选 **Generate Java File** 选项框

任务7.2: 注册 EmployeeEO 实体对象到员工验证应用模块中

在 **OAEntityExpert** 类中可以使用方便的方法来关联 **EmployeeVAM** 和 **EmployeeEO**

- 在导航栏中选择 **EmployeeEO**，右键从上下文菜单中选择 **Edit EmployeeEO...**
- 导航到 **Properties** 页
- 创建如下的属性
  - Name： VAMDef
  - Value：oracle.apps.ak.schema.server.EmployeeVAM
- 选择 确定 创建新的属性

任务7.3: 创建职位验证视图对象

验证视图对象包装了简单的验证查询，而不是直接写到实体对象或实体专家中。如 PositionVVO 检查提供的职位是否存在于数据库中

- 在 **oracle.apps.ak.schema.server** BC4J 包中创建一个新的视图对象 **PositionVVO**
- 使用如下的查询语句

```
SELECT lookup_code
FROM fwk_tbx_lookup_codes_b
WHERE lookup_code = :1
and lookup_type = 'FWK_TBX_POSITIONS'
```

- 为视图对象产生 *VOImpl 和 *VORowImpl

任务7.4: 添加 PositionVVO 到 EmployeeVAM 验证应用模块

- 添加 **PositionVVO** 到 **EmployeeVAM** 中

任务7.5: 添加 initQuery 方法到 PositionVVO 验证视图对象中

添加如下的 **initQuery()** 方法到 **PositionVVOImpl** 类中

```
public void initQuery(String positionCode)
 {
  setWhereClauseParams(null); // Always reset
  setWhereClauseParam(0, positionCode);
  executeQuery();
 }
```

任务7.6: 创建员工验证视图对象

此验证视图对象查找有效的员工。

- 在 **oracle.apps.ak.schema.server** BC4J 包中创建一个新的视图对象 **EmployeeVVO**
- 使用如下的查询语句

  SELECT employee_id

```
FROM fwk_tbx_employees
WHERE end_date is null
and employee_id = :1
```

- 为视图对象产生 *VOImpl 和 *VORowImpl

任务7.7: 添加 EmployeeVVO 到 EmployeeVAM 验证应用模块

- 添加 **EmployeeVVO** 到 **EmployeeVAM** 中

任务7.8: 添加 initQuery 方法到 EmployeeVVO 验证视图对象中

添加如下的 **initQuery()** 方法到 **EmployeeVVOImpl** 类中

```
import oracle.jbo.domain.Number;
...


 public void initQuery(Number employeeId)
 {
  setWhereClauseParams(null); // Always reset
  setWhereClauseParam(0, employeeId);
  executeQuery();
 }
```

任务7.9: 创建实体专家类

实体专家是一个和实体对象紧密关联的 singletons(整个应用系统只存在一个实例供所有的用户使用)，他可以用来执行实体对象的内部操作。

- 在导航栏中选择 **SearchOAProject.jpr**，右键从上下文菜单中选择 **New > Class**
- 在 **New Class** 对话框中：
- 设置 **Name** 的值为 **EmployeeEntityExpert**
- 设置 **Package** 的值为 **oracle.apps.ak.schema.server**
- 设置 **Extends** 的值为 **oracle.apps.fnd.framework.server.OAEntityExpert**
- 在 **Optional Attributes** 区域，只选择 **Public** 选项框

任务7.10: 关联实体专家和 EmployeeEO 实体对象

实体专家类被注册到它自己的实体对象上

- 在导航栏中选择 **EmployeeEO** ，右键从上下文菜单中选择 **Edit EmployeeEO ...**
- 导航到 **Properties** 页
- 添加如下的属性
  - Name：ExpertClass
  - Value：oracle.apps.ak.schema.server.EmployeeEntityExpert

任务7.11: 在 EmployeeEOImpl 类中为专家存取添加一个简便的方法

添加如下的静态方法到 EmployeeEOImpl 类中(可以在 EmployeeEOImpl 类没有初始化的时候调用)

```
import oracle.apps.fnd.framework.server.OADBTransaction;

...


 /*
 * Convenience method returns the EmployeeEntityExpert.
 */
 public static EmployeeEntityExpert getEmployeeEntityExpert (OADBTransaction txn)
 {
   return (EmployeeEntityExpert)txn.getExpert(EmployeeEOImpl.getDefinitionObject());

 } // end getEmployeeEntityExpert()
```

任务7.12: 添加 isEmployeeActive( ) 方法到实体专家

添加如下的方法到 **EmployeeEntityExpert** 类中

提示：如果没有完成 7.2 的步骤：注册 EmployeeEO 到 EmployeeVAM ， findValidationViewObject() 方法将返回空

```
import oracle.jbo.domain.Number;...
 /*
  ********************************************************************************
  * Returns true if the given employee is currently active and false if not.
```

```
 ********************************************************************************
*/


public boolean isEmployeeActive(Number employeeId)
{
  boolean isActive = false;

  // Note that we want to use a cached, declaratively defined VO instead of creating
  // one from a SQL statement which is far less performant.


  EmployeeVVOImpl empVO =
    (EmployeeVVOImpl)findValidationViewObject("EmployeeVVO1");
  empVO.initQuery(employeeId);

  // We're just doing a simple existence check. If we don't find a match, return false.
  if (empVO.hasNext())
  {
    isActive = true;
  }
  return isActive;
} // end isEmployeeActive()
```

任务7.13: 添加 isPositionValid ( ) 方法到实体专家

添加如下的方法到 **EmployeeEntityExpert** 类中

提示：如果没有完成 7.2 的步骤：注册 EmployeeEO 到 EmployeeVAM ， findValidationViewObject() 方法将返回空

```
/*
```

```
 ********************************************************************************
 * Returns true if the given position is valid.
 ********************************************************************************
*/
public boolean isPositionValid(String position)
{
  boolean isActive = false;

  // Note that we want to use a cached, declaratively defined VO instead of creating
  // one from a SQL statement which is far less performant.
  PositionVVOImpl positionVO =
    (PositionVVOImpl)findValidationViewObject("PositionVVO1");
  positionVO.initQuery(position);

  // We're just doing a simple existence check. If we don't find a match, return false.
  if (positionVO.hasNext())
  {
    isActive = true;
  }
  return isActive;
} // end isPositionValid()
```

任务7.14: 添加 验证逻辑方法 setPositionCode() 到 EmployeeEOImpl 类中

添加如下的逻辑到 EmployeeEOImpl 类的setPositionCode() 方法中。它调用实体专家的 isPositionValid() 方法，如果返回 false抛出属性级别的例外。

```
/*
 ********************************************************************************
 * Sets <code>value</code> as the attribute value for PositionCode
 ********************************************************************************
 */
public void setPositionCode(String value)
```

```
{
  // BC4J ensures that this mandatory attribute is non-null
  if ((value != null) || (!("".equals(value.trim()))))
  {
    EmployeeEntityExpert expert = getEmployeeEntityExpert(getOADBTransaction());
    if (!(expert.isPositionValid(value)))
    {
      throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
        getEntityDef().getFullName(), // EO name
        getPrimaryKey(), // EO PK
        "PositionCode", // Attribute Name
        value, // Attribute value
        "AK", // Message product short name
        "FWK_TBX_T_EMP_POSITION_INVALID"); // Message name
    }
  }

  setAttributeInternal(POSITIONCODE, value);

} // end setPositionCode()
```

任务7.15: 添加 验证逻辑方法 setManagerId () 到 EmployeeEOImpl 类中

添加如下的逻辑到 EmployeeEOImpl 类的 setManagerId () 方法中。它调用实体专家的 isEmployeeActive() 方法，如果返回 false 抛出属性级别的例外。

```
/*
 ***************************************************************************
 * Sets <code>value</code> as the attribute value for ManagerId
 ***************************************************************************
 */
 public void setManagerId(Number value)
 {
```

```
  if (value != null)
  {
   EmployeeEntityExpert expert = getEmployeeEntityExpert(getOADBTransaction());
   if (!(expert.isEmployeeActive(value)))
   {
    throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
      getEntityDef().getFullName(), // EO name
      getPrimaryKey(), // EO PK
      "ManagerId", // Attribute Name
      value, // Attribute value
      "AK", // Message product short name
      "FWK_TBX_T_EMP_MGR_INACTIVE"); // Message name
   }
  }

  setAttributeInternal(MANAGERID, value);

 } // end setManagerId()
```

任务7.16: 编译测试

步骤8(可选)：实现员工业务逻辑(杂项属性)

任务8.1: 添加逻辑到 setEndDate() 方法，添加一个 validateEndDate() 方法

这两个方法实际上和前面所创建的 **setStartDate()** 和 **validateStartDate()** 方式是一样的

添加如下的逻辑到 **EmployeeEOImpl** 类中的 **setEndDate()** 方法

```
/*
 ***************************************************************************
 * Sets <code>value</code> as the attribute value for EndDate
 ***************************************************************************
 */
public void setEndDate(Date value)
{
  validateEndDate(value);
  setAttributeInternal(ENDDATE, value);
} // end setEndDate()
```

添加如下的方法到**EmployeeEOImpl** 类中

```
/*
 ***************************************************************************
 * Verifies that the employee's End Date is valid.
 *
 * Business Rules:
 * This is an optional value which can be updated at any time.
 * Cannot be earlier than sysdate.
 ***************************************************************************
 */
protected void validateEndDate(Date value)
{
  // If a value has been set, validate it.
  if (value != null)
  {

    OADBTransaction transaction = getOADBTransaction();
```

```
    // Note that we want to truncate these values to allow for the possibility
    // that we're trying to set them to be the same day. Calling      // dateValue( ) does not include time. Were we to want the
time element,
    // we would call timestampValue(). Finally, you cannot compare
    // oracle.jbo.domain.Date objects directly. Instead, convert the value to
    // a long as shown.
    long sysdate = transaction.getCurrentDBDate().dateValue().getTime();
    long endDate = value.dateValue().getTime();

    if (endDate < sysdate)
    {
      throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
        getEntityDef().getFullName(), // EO name
        getPrimaryKey(), // EO PK
        "EndDate", // Attribute Name
        value, // Attribute value
        "AK", // Message product short name
        "FWK_TBX_T_END_DATE_PAST"); // Message name
    }

   }

 } // end validateEndDate()
```

任务8.2: 添加逻辑到 setSalary () 方法

此方法简单的验证了 salary 的值大于 0 。它描述了如何正确对比 **oracle.jbo.domain.Number** 值

添加如下的逻辑到 **EmployeeEOImpl** 类中的 **setSalary ()** 方法

```
/*
```

```
 ********************************************************************************
 * Sets <code>value</code> as the attribute value for Salary
 ********************************************************************************
*/
public void setSalary(Number value)
{
  if (value != null)
  {
    // Verify value is > 0
    if (value.compareTo(0) <= 0)
    {
      throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
        getEntityDef().getFullName(), // EO name
        getPrimaryKey(), // EO PK
        "Salary", // Attribute Name
        value, // Attribute value
        "AK", // Message product short name
        "FWK_TBX_T_EMP_SALARY_REQUIRED"); // Message name
    }

    setAttributeInternal(SALARY, value);

  }

} // end setSalary()
```

任务8.3: 保存测试

步骤9：实现员工业务逻辑(实体级验证)

在实体对象中，所有的交叉属性验证都必须添加到实体级别的 validateEntity() 方法中。当属性值被传回中间层时，BC4J 调用属性设置器，然后为"脏"(改变的)实体调用 validateEntity() 方法。所以此逻辑总是在所有的属性设置器调用后执行。

添加如下的逻辑到 **EmployeeEOImpl** 类中的 **validateEntity ()** 方法

```
/*
 ***************************************************************************
 * Add entity-level and cross-attribute validation here.
 ***************************************************************************
 */
protected void validateEntity()
{
  // Always call this first.
  super.validateEntity();

  // Check to see that manager id is specified unless the employee
  // is the "President," in which case it's OK for this to be null.
  if (getManagerId() == null)
  {
    if (!("PRESIDENT".equals(getPositionCode())))
    {
      throw new OAAttrValException(OAException.TYP_ENTITY_OBJECT,
        getEntityDef().getFullName(), // EO name
        getPrimaryKey(), // EO PK
        "ManagerId", // Attribute Name
        getManagerId(), // Attribute value
       "AK", // Message product short name
       "FWK_TBX_T_EMP_MANAGER_REQUIRED"); // Message name
    }
  }
```

```
/*
   // Validate the start and end dates.
   Date startDate = getStartDate();
   Date endDate = getEndDate();

   // The start date can be entered only when the object is new.  You don't    // want to call this validation when updating a row since it will, in
   // all likelihood, fail.

   if (getEntityState() == STATUS_NEW)
   {
    validateStartDate(startDate);
   }

   validateEndDate(endDate);

   // We validate the following here instead of from within an attribute because
   // we need to make sure that both values are set before we perform the test.
   if (endDate != null)
   {
    // Note that we want to truncate these values to allow for the possibility
    // that we're trying to set them to be the same day. Calling
    // dateValue( ) does not include time. Were we to want the time element,
    // we would call timestampValue(). Finally, whenever comparing jbo
    // Date objects, we have to convert to long values.
    long endDateLong = endDate.dateValue().getTime();

    // We can assume we have a Start Date or the validation of this
    // value would have thrown an exception.
    if (endDateLong < startDate.dateValue().getTime())
    {
     throw new OARowValException(OARowValException.TYP_ENTITY_OBJECT,
       getEntityDef().getFullName(),
       getPrimaryKey(),
       "AK", // Message product short name
       "FWK_TBX_T_START_END_BAD"); // Message name
    }
   }
*/
```

```
} // end validateEntity()
```

步骤10：实现 Apply 和 Cancel 按钮处理

当用户按下 Apply 按钮后，需要提交数据并返回 Search 页面，并显示创建成功的消息。当用户按下 Cancel 按钮后，需要回滚事务处理并返回 Search 页面。

任务10.1: 在 EmployeeAMImpl 类中创建 apply() 方法

除了 **OAApplicationModule** 接口外，控制器不直接存取服务器端的对象。需要在 EmployeeAMImpl 类中创建 apply()方法来提交事务处理。然后在 AM 中增加控制器代码来调用此方法。

添加如下的方法到 **EmployeeAMImpl** 类中

```
/*
*******************************************************************************
* Commits the transaction.
*******************************************************************************
*/
public void apply()
{
  getTransaction().commit();
} // end apply()
```

任务10.2: 添加 processFormRequest( ) 逻辑到 EmployeeCreateCO 类

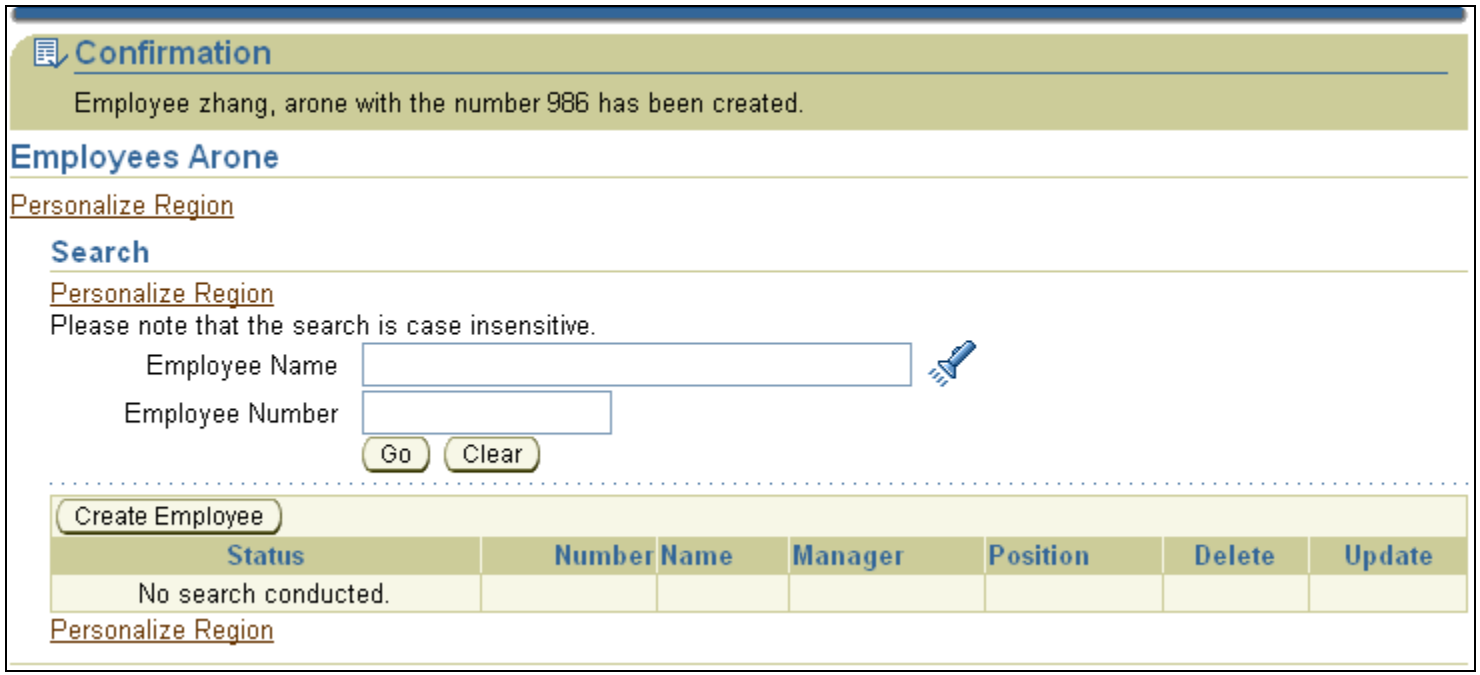
添加如下的代码到 EmployeeCreateCO 类来处理 Apply 和 Cancel 按钮的选择

```
import oracle.jbo.domain.Number;
import oracle.apps.fnd.common.MessageToken;
import oracle.apps.fnd.framework.OAApplicationModule;
import oracle.apps.fnd.framework.OAException;
import oracle.apps.fnd.framework.OAViewObject;
import oracle.apps.fnd.framework.webui.OAWebBeanConstants;

...

 // Always call this first.
 super.processFormRequest(pageContext, webBean);
 OAApplicationModule am = pageContext.getApplicationModule(webBean);

 // Pressing the "Apply" button means the transaction should be validated
 // and committed.
 if (pageContext.getParameter("Apply") != null)
 {
  // Generally in the tutorial application and the labs, we've illustrated
  // all BC4J interaction on the server (except for the AMs, of course). Here,
  // we're dealing with the VO directly so the comments about the reasons
  // why we're obtaining values from the VO and not the request make sense
  // in context.
  OAViewObject vo = (OAViewObject)am.findViewObject("EmployeeFullVO1");

  // Note that we have to get this value from the VO because the EO will
  // assemble it during its validation cycle.
  // For performance reasons, we should generally be calling getEmployeeName()
  // on the EmployeeFullVORowImpl object, but we don't want to do this
  // on the client so we're illustrating the interface-appropriate call. If
  // we implemented this code in the AM where it belongs, we would use the
  // other approach.
  String employeeName = (String)vo.getCurrentRow().getAttribute("EmployeeName");
  // We need to get a String so we can pass it to the MessageToken array below. Note
  // that we are getting this value from the VO (we could also get it from
  // the Bean as shown in the Drilldwon to Details lab) because the item style is messageStyledText,
```

```
 // so the value isn't put on the request like a messaqeTextInput value is.
 Number employeeNumber = (Number)vo.getCurrentRow().getAttribute("EmployeeId");
 String employeeNum = String.valueOf(employeeNumber.intValue());
// Simply telling the transaction to commit will cause all the Entity Object validation
// to fire.
//
// Note: there's no reason for a developer to perform a rollback. This is handled    by
// the framework if errors are encountered.

am.invokeMethod("apply");
// Indicate that the Create transaction is complete.
TransactionUnitHelper.endTransactionUnit(pageContext, "empCreateTxn");

// Assuming the "commit" succeeds, navigate back to the "Search" page with
// the user's search criteria intact and display a "Confirmation" message
// at the top of the page.

MessageToken[] tokens = { new MessageToken("EMP_NAME", employeeName),
new MessageToken("EMP_NUMBER", employeeNum) };
OAException confirmMessage = new OAException("AK", "FWK_TBX_T_EMP_CREATE_CONFIRM",    tokens,
OAException.CONFIRMATION, null);
// Per the UI guidelines, we want to add the confirmation message at the
// top of the search/results page and we want the old search criteria and
// results to display.

pageContext.putDialogMessage(confirmMessage);
pageContext.forwardImmediately("OA.jsp?page=/oracle/apps/ak/employee/server/webui/EmpSearchPG",
                    null,
                    OAWebBeanConstants.KEEP_MENU_CONTEXT,
                    null,
                    null,
                    true, // retain AM
                    OAWebBeanConstants.ADD_BREAD_CRUMB_NO);
}
else if (pageContext.getParameter("Cancel") != null)
{
  am.invokeMethod("rollbackEmployee");
  // Indicate that the Create transaction is complete.
  TransactionUnitHelper.endTransactionUnit(pageContext, "empCreateTxn");
```

```
    pageContext.forwardImmediately("OA.jsp?page=/oracle/apps/ak/employee/server/webui/EmpSearchPG",
                                   null,
                                   OAWebBeanConstants.KEEP_MENU_CONTEXT,
                                   null,
                                   null,
                                   true, // retain AM
                                   OAWebBeanConstants.ADD_BREAD_CRUMB_NO);
   }
  } // end processFormRequest()
```

任务10.3: 测试



Open and Closed Issues for this Deliverable

Open Issues

ID	Issue	Resolution	Responsibility	Target Date	Impact Date

Closed Issues

ID	Issue	Resolution	Responsibility	Target Date	Impact Date