原创 Ofbiz 入门教程

1.Ofbiz 介绍：

Ofbiz(http://www.ofbiz.org) 是 Open Source 的商务软件系统，充分利用了各优秀的的Open Source 项目，
像 Tomcat, Ant, BeanShell, Jboss 等，构建了一个强大的系统平台，Ofbiz 已经完成了大部分商务类软件系统
都需要的部件，像用户认证、工作流、商务规则处理等，Ofbiz 的核心技术在于 Entity Engine，其他的组件基本都
是基于它的。简单来说 Entity Engine 的主要功能是将数据库表创建、对象与数据表的映射、对象的查询等做了强大
封装，你可以在一个简单的 XML 文件中定义数据库表结构，Ofbiz 会自动帮你在数据库建表，并动态生成映射对象，
你在程序中可以只考虑对 Object 的处理,Ofbiz 会自动通过事务逻辑更新到数据库中。Ofbiz 宣称的优点之一是用
很少的 Code 完成复杂的处理。

2.Ofbiz 下载与安装

首先要安装 J2SDK1.4，到 http://java.sun.com 上下载，安装后设定 JAVA_HOME 环境变量为 J2SDK 的安装目录。

访问网站 http://www.ofbiz.org，上面有下载的连接，请选择 Complete 包，因为这个包中已经包含了运行 Ofbiz
的所有东西，下载下来解开后就可以运行了。

解开 Ofbiz 包到一个目录下，假设是 “C:\ofbiz”，该目录下将会有 catalina 和 ofbiz 两个目录， catalina
目录是 Tomcat 的目录，Ofbiz 对其配置做了修改，ofbiz 目录是 Ofbiz 的程序代码目录。在命令行状态下进入
“c:\ofbiz\catalina\bin” 目录，运行“ ofbiz run”命令，就可以启动 Ofbiz，启动后你可以用浏览器访问
“http://localhost:8080/ecommerce”，这可以访问 Ofbiz 的电子商务模块，通过页面上面的连接你可以访问到
其他模块。

3.Ofbiz Schema 的创建

Ofbiz 应用入门：

以一个实例说明，假设我们需要建一个客户资料表，起名为 StudyCustomer，各个段分别如下：

```
StudyCustomer {
  customerId      Integer,
  customerName    String,
  customerNote    String,
}
```

我们来实现基本的数据操作---增/删/改/查询，具体步骤如下：

1.在 XML 文件中定义数据 Schema：

需要用到三个文件，一个是我们要建的项目的 entitymodel_xxx.xml 和 entityengine.xml，还有

entitygroup.xml,

entitymodel_xxx.xml 是需要我们自己创建的，假设我们起名为 entitymodel_study.xml，放在

“c:\ofbiz\ofbiz\commonapp\entitydef”目录下，

entityengine.xml 是 Ofbiz 已经有的，放在“c:\ofbiz\commonapp\etc”目录下，用来包含我们定义的 entitymodel 文件。

entitygroup.xml 也是 Ofbiz 已经有的,跟 engityengine.xml 在同一目录下，我们需要把我们的 Schema 定义加入到该文件中

entitymodel_study.xml 文件的定义格式如下：

```
<!--============================================================================================--> 

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE entitymodel PUBLIC "-//OFBiz//DTD Entity Model//EN"

"http://www.ofbiz.org/dtds/entitymodel.dtd">

<entitymodel>
   <title>Entity of an Open For Business Project Component</title>
   <description>None</description>
   <copyright>Copyright (c) 2002 The Open For Business Project - www.ofbiz.org</copyright>
   <author>None</author>
   <version>1.0</version>

  <!-- ========================================================= -->
  <!-- ======================== Data Model ====================== -->
```

```
  <!-- The modules in this file are as follows:                    -->
  <!--  - org.ofbiz.commonapp.study -->
  <!-- ======================================================== -->


  <!-- ======================================================== -->
  <!-- org.ofbiz.commonapp.study -->
  <!-- ======================================================== -->

   <entity entity-name="StudyCustomer"
         package-name="org.ofbiz.commonapp.study"
   title="Study Customer Entity">
 <field name="customerId" type="id-ne"></field>
 <field name="customerName" type="long-varchar"></field>
 <field name="customerNote" type="long-varchar"></field>
 <prim-key field="customerId"/>
  </entity>
 </entitymodel>

<!--============================================================================
===============-->
```

这个 XML 文件中的 Tag 基本是看得明白的，只是 field 的 type 是 Ofbiz 已经预定义好的，这
是为了保证数据库间的迁移。

在 entityengine.xml 加入我们刚才定义的文件，加入一行在合适的位置：

```
<resource loader="mainfile" location="entitymodel_study.xml"/>
```

具体放的位置我们可以通过查看 entityengine.xml 找到，里面已经有加好的其他文件。

在 entitygroup.xml 加入我们的 Schema 定义，在后面加入一行

```
<entity-group group="org.ofbiz.commonapp" entity="StudyCustomer" />
```

这样我们就定义好了 Schema，现在把 c:\ofbiz\commonapp\etc\entityengine.xml 拷贝到
c:\ofbiz\catalina\shared\classes 目录下，这点要切记，我以前就因为没有拷贝，最后 Schema
怎么也创建不了。

重新启动 Ofbiz，访问 URL: http://localhost:8080/webtools，点击右上方的 "Login" 链接，
用 admin/ofbiz 登录，登录进入后选择链接“Check/Update Database”，这时会出现 Check 的 Form，

该表单可以只检验 Schema 是否改变，默认的 GroupName 是“org.ofbiz.commonapp”，这个不需要变，

点击“Check Only”按钮，Ofbiz 会检验变动情况，显示出一个完整的列表，你可以查一下是否有我们刚建的

"StudyCustomer"，如果没有，可能是我们前面定义的有些问题，检查一下再重新做。

在检查到以后，可以再选择“Check and Add Missing”，这是 Ofbiz 很强大的一个功能，你在 XML 中新

增了表，或在某个表中新增了段，它会自动映射到数据库中，避免我们去直接操作数据库。

现在已经完成了 StudyCustomer Schema 的创建,如果想检验一下是否有表创建，我们可以用编辑器打开

c:\ofbiz\data\ofbiz.script ，在里面查询 CREATE TABLE StudyCustomer 的字样，如果前面没有

问题，我们可以找到的。

4.如何使用已经定义的 Schema

如何使用已经定义的 Schema

Ofbiz 遵循 MVC 的设计模式，在 View 端，即 JSP 端主要使用 Ofbiz 定义的 Tag 来显示或

提取数据，Control 是一个 Controller Servlet，我们在 Controller Servlet 的 URI mapping

配置文件中定义各 URL 应该指向什么程序，这样，通过这个 mapping 配置文件，可以保证我们各个页面

及具体处理程序之间的独立性，例我们可以通过修改这个配置文件就可以改变某个 Form 的 Post Action

的 URL，而不需要修改实际的 HTML 或 JSP 代码。

Ofbiz 中定义了 Regions 的概念，即将一个 HTML 页面分成几个区域，像 Top, Left, Right, Main

等，通过这些 Regions 我们可以方便的组合 UI 界面，并且可以方便改变各部分所处的位置，如我们可以

把菜单很容易的从上方移到下方，只需要改变一个配置文件。Regions 类似于 HTML 中的 Frame，但它是

通过一个页面来组合界面，Frame 是通过几个页面显示在不同的帧中，Frame 的控制比较复杂，而且需要

改变相关的程序。

在 Ofbiz 中，我们可以直接在 JSP 中操作 Schema 定义的 Object，即我们刚定义的 StudyCustomer，

示例如下：

```
<%@ taglib uri="ofbizTags" prefix="ofbiz" %>

<%@ page import="java.util.*" %>
<%@ page import="org.ofbiz.core.util.*, org.ofbiz.core.pseudotag.*" %>
<%@ page import="org.ofbiz.core.entity.*" %>

<jsp:useBean id="delegator"
type="org.ofbiz.core.entity.GenericDelegator" scope="request" />
<jsp:useBean id="security" type="org.ofbiz.core.security.Security"
scope="request" />

<%if(security.hasEntityPermission("PARTYMGR", "_VIEW", session)) {%>

<%
  try {
    delegator.create("StudyCustomer",

UtilMisc.toMap("customerId","1","customerName","Cust1","customerNote"
,"Customer Note 1"));

    Iterator custs =

UtilMisc.toIterator(delegator.findAll("StudyCustomer",UtilMisc.toList
("customerId","customerName","customerNote")));

    while(custs.hasNext())
    {
 GenericValue cust = (GenericValue)custs.next();
 out.println(cust.getString("customerId"));
 out.println(cust.getString("customerName"));
 out.println(cust.getString("customerNote"));
    }
  } catch(Exception e)
  {
    out.println(e.getMessage());
  }
%>
<%}else{%>
```

<h3>You do not have permission to view this page. ("PARTYMGR_VIEW" or "PARTYMGR_ADMIN" needed)</h3>
<%}%>

这段程序挺容易理解，先是通过 delegator 创建一个 Object，该 Object 将会由 Ofbiz 自动同步到
数据库中。然后通过 delegator 的 findAll 取到所有已保存的 Object，最后通过一个 Iterator 对象
显示出来。

这个程序起名为 testofbiz.jsp，为简单起见，我们放到 Ofbiz 已有的一个 Webapp 的目录下，放到
c:\ofbiz\ofbiz\partymgr\webapp\party 目录下。然后我们需要修改两个配置文件：controller.xml
和 regions.xml,这两个文件就是我们上面提到的 mapping 和 regions 配置文件。

这两个文件都在：c:\ofbiz\ofbiz\partymgr\webapp\WEB-INF 下，在 controller.xml 中加入下面

```
<request-map uri="testofbiz">
   <description>Test Ofbiz</description>
   <security https="false" auth="false"/>
   <response name="success" type="view" value="testofbiz"/>
</request-map>
```

和

```
<view-map name="testofbiz" type="region"/>
```

加入位置请参照 controller.xml 中已经有的配置。在 regions.xml 中加入：

```
<define id='testofbiz' region='MAIN_REGION'>
    <put section='title'>Test Ofbiz</put>
    <put section='content' content='/party/testofbiz.jsp'/>
</define>
```

具体加入位置请参考已有的配置。

配置完后，重新启动 ofbiz，然后访问 URL:
http://localhost:8080/partymgr/control/testofbiz

由于我们在 testofbiz.jsp 程序中使用了 Ofbiz 的安全控制机制,系统会提示现在没有访问
权限，需要登录，点击右边的“Login” 用 admin/ofbiz 登录后会看到我们程序 testofbiz.jsp
的运行结果。如果需要增加新记录，请修改

UtilMisc.toMap("customerId","1","customerName","Cust1","customerNote","Customer Note 1"));

中的各个段的值，然后再访问
http://localhost:8080/partymgr/control/testofbiz，如果不修改
而直接访问那个 URL 时，系统会提示 Primary key 冲突。

5.按照显示与逻辑分离的原则使用 Schema:

上篇讲了如何在 JSP 中使用创建的 Schema 对象，这次我们来讲述一下如何把程序
逻辑放到 JavaBeans 中，把显示处理放到 JSP 中，并使用 controller.xml 将两
部分整合起来。

首先我们来创建一个 JavaBeans，来完成 Add/Get/Delete/Update Schema 对象
的操作，程序文件名为 TestOfbiz.java，放置在
c:\ofbiz\ofbiz\testOfbiz\com\geeyo\ofbiz 目录下，具体程序如下：

```
>================================================================
package com.geeyo.ofbiz;

import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.net.*;
import org.ofbiz.core.util.*;
import org.ofbiz.core.entity.*;
import org.ofbiz.core.service.*;
import org.ofbiz.core.security.*;
import org.ofbiz.core.stats.*;

public class TestOfbiz
{
  public static void main(String[] args)
  throws Exception
  {
     GenericDelegator delegator =
GenericDelegator.getGenericDelegator("default");
```

```
delegator.create("StudyCustomer",UtilMisc.toMap("customerId","3","cus
tomerName","Kane3","customerNote","This is test customer.3"));

     Iterator custs =
UtilMisc.toIterator(delegator.findAll("StudyCustomer",UtilMisc.toList
("customerId","customerName","customerNote")));

     while(custs.hasNext())
     {
        GenericValue cust = (GenericValue)custs.next();
        System.out.println(cust.getString("customerId"));
        System.out.println(cust.getString("customerName"));
        System.out.println(cust.getString("customerNote"));
     }
  }

  public static String createNewRecord(HttpServletRequest request,
HttpServletResponse response)
  throws Exception
  {
     Map paras = UtilMisc.getParameterMap(request);

     GenericDelegator delegator =
GenericDelegator.getGenericDelegator("default");
     delegator.create("StudyCustomer",paras);

     return "success";
  }

  public static String lookAllRecords(HttpServletRequest request,
HttpServletResponse response)
  throws Exception
  {
     GenericDelegator delegator =
GenericDelegator.getGenericDelegator("default");
     Iterator custs =
UtilMisc.toIterator(delegator.findAll("StudyCustomer",UtilMisc.toList
("customerId","customerName","customerNote")));

     Collection col = new ArrayList();

     while(custs.hasNext())
     {
```

```
            GenericValue cust = (GenericValue)custs.next();
            col.add(cust);

        }

        request.getSession().setAttribute("search_results",col);

        return "success";
    }

    public static String findRecord(HttpServletRequest request,
HttpServletResponse response)
    throws Exception
    {
        String id = (String)request.getParameter("customerId");

        GenericDelegator delegator =
GenericDelegator.getGenericDelegator("default");

        try {
            GenericValue cust =
delegator.findByPrimaryKey("StudyCustomer",UtilMisc.toMap("customerId
",id));

            request.getSession().setAttribute("edit_cust",cust);
        } catch (GenericEntityException gee) {
            Debug.logWarning(gee);
        }

        return "success";
    }

    public static String updateRecord(HttpServletRequest request,
HttpServletResponse response)
    throws Exception
    {
        Map paras = UtilMisc.getParameterMap(request);

        GenericDelegator delegator =
GenericDelegator.getGenericDelegator("default");
        GenericValue cust =
delegator.findByPrimaryKey("StudyCustomer",UtilMisc.toMap("customerId
",paras.get("customerId")));
        cust.setNonPKFields(paras);
```

```
        cust.store();

        request.getSession().setAttribute("edit_cust",cust);

        return "success";
    }

    public static String removeRecord(HttpServletRequest request,
HttpServletResponse response)
    throws Exception
    {
        String strId = request.getParameter("id");
        GenericDelegator delegator =
GenericDelegator.getGenericDelegator("default");
        GenericValue cust =
delegator.findByPrimaryKey("StudyCustomer",UtilMisc.toMap("customerId
",strId));
        cust.remove();

        return "success";
    }

}
```

>==========

程序中的处理大部分可以看懂的，其中有个功能，是
Map paras = UtilMisc.getParameterMap(request);
这是 Ofbiz 的一个有趣但非常有用的功能,它是把 request 中各段的名字和值映射到一个 Map
对象中，然后使用
cust.setNonPKFields(paras);
就可以赋给 Object cust 的各个段，免了我们使用
request.getParameter("name")来取各个
值，在值很多的时候这个功能可以大大减少冗余代码量。

基本程序的逻辑是这样的，
1.从 request 读取传来的值
2.使用 delegator 来处理，Add/Update/Delete/Query
3.将返回结果放到 Session 中传给 JSP

我做了个 Ant build.xml 文件可以帮助编译，把这个文件放在:

c:\ofbiz\ofbiz\testOfbiz\ 目录下，然后在命令行窗口下进入该目录，敲入 ant
来编译（需要保证已经安装 Ant），编译后的 .class 会放在
c:\ofbiz\ofbiz\testOfbiz\com\geeyo\ofbiz 下，
拷贝 c:\ofbiz\ofbiz\testofbiz\com 目录到
c:\ofbiz\ofbiz\partymgr\webapp\WEB-INF\classes
目录下。

build.xml
>===
=========

```
<project name="TestOfbiz" default="dist" basedir=".">
    <description>
        Test ofbiz
    </description>

  <!--test cvs-->
  <!-- set global properties for this build -->

  <property name="src" location="."/>
  <property name="build" location="."/>

  <property name="lib_dir"  location="c:/ofbiz/catalina/shared/lib"/>
  <property name="lib1_dir"
location="c:/ofbiz/catalina/common/lib"/>

  <path id="project.class.path">
    <fileset dir="${lib_dir}">
       <include name="*.jar"/>
    </fileset>
    <fileset dir="${lib1_dir}">
       <include name="*.jar"/>
    </fileset>
  </path>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
  </target>

  <target name="compile" depends="init"
```

```
        description="compile the source " >
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}">
      <classpath refid="project.class.path"/>
    </javac>
  </target>

  <target name="dist" depends="compile"
        description="generate the distribution" >
    <!-- Create the distribution directory -->
  </target>

  <target name="clean"
        description="clean up" >
    <!-- Delete the ${build} and ${dist} directory trees -->
  </target>
</project>
```

>==========

然后我们来创建 JSP 程序，JSP 程序全部放在
c:\ofbiz\ofbiz\partymgr\webapp\party 下面

1.listofbiz.jsp

>==========

```
<%@ taglib uri="ofbizTags" prefix="ofbiz" %>

<%@ page import="java.util.*, org.ofbiz.core.service.ModelService" %>
<%@ page import="org.ofbiz.core.util.*, org.ofbiz.core.pseudotag.*" %>
<%@ page import="org.ofbiz.core.entity.*" %>
<jsp:useBean id="security" type="org.ofbiz.core.security.Security"
scope="request" />
<jsp:useBean id="delegator"
type="org.ofbiz.core.entity.GenericDelegator" scope="request" />

<script language="JavaScript">
  function confirmDelete()
  {
     return confirm("Are your sure to delete?");
  }
```

```
</script>


<%if(security.hasEntityPermission("PARTYMGR", "_VIEW", session)) {%>

<table width="600" align="center">
  <ofbiz:if name="search_results">
   <tr><th>Id</th><th>Name</th><th>Note</th><th></th></tr>
    <ofbiz:iterator name="cust" property="search_results">
      <tr>
        <td><ofbiz:entityfield attribute="cust"
field="customerId"/></td>
        <td><ofbiz:entityfield attribute="cust"
field="customerName"/></td>
        <td><ofbiz:entityfield attribute="cust"
field="customerNote"/></td>
        <td>
          <a href='<ofbiz:url>/showtest?customerId=<ofbiz:entityfield
attribute="cust" field="customerId"/></ofbiz:url>'
class="buttontext">[Edit]</a>
          <a
href='<ofbiz:url>/removetest?customerId=<ofbiz:entityfield
attribute="cust" field="customerId"/></ofbiz:url>' class="buttontext"
onclick="return confirmDelete()">[Remove]</a>
        </td>
      </tr>
     </ofbiz:iterator>
    </ofbiz:if>
</table>
<table width="200" align="center">
  <tr>
  <td><a href='<ofbiz:url>/createTestForm</ofbiz:url>'>Create
customer</a></td>
  </tr>
</table>

<%}else{%>
  <h3>You do not have permission to view this page. ("PARTYMGR_VIEW" or
"PARTYMGR_ADMIN" needed)</h3>
<%}%>

>=====================================================================
=========
```

上面程序中需要说明的是
<ofbiz:if name="search_results">
和
<ofbiz:iterator name="cust" property="search_results">,

<ofbiz:if name="search_results"> 是用来检验在 session 或 pageContext 对象
中是否包含 search_results 对象,该对象是由我们的程序放到 session 中的。
<ofbiz:iterator name="cust" property="search_results"> 是用来循环读取对象
search_results（是个 Collection 对象）中存储的各对象，并赋给 cust，然后在循环体
中，我们就可以用 cust 对象来读取各个段的值了。

2.createofbiz.jsp
>==========

```
<%@ taglib uri="ofbizTags" prefix="ofbiz" %>

<%@ page import="java.util.*, org.ofbiz.core.service.ModelService" %>
<%@ page import="org.ofbiz.core.util.*, org.ofbiz.core.pseudotag.*" %>
<%@ page import="org.ofbiz.core.entity.*" %>
<jsp:useBean id="security" type="org.ofbiz.core.security.Security"
scope="request" />
<jsp:useBean id="delegator"
type="org.ofbiz.core.entity.GenericDelegator" scope="request" />
<%if(security.hasEntityPermission("PARTYMGR", "_VIEW", session)) {%>

<form method="post" action="<ofbiz:url>/createTest</ofbiz:url>"
name="createofbiz">
<table width="300" align="center">
  <tr>
   <td>Id</td><td><input type="text" name="customerId" size="20"></td>
  </tr>
  <tr>
   <td>Name</td><td><input type="text" name="customerName"
size="20"></td>
  </tr>
  <tr>
   <td>Note</td><td><input type="text" name="customerNote"
size="30"></td>
  </tr>
```

```
  <tr>
   <td></td>
  <td><input type="submit"></td>
  </tr>
</table>
</form>

<%}else{%>
  <h3>You do not have permission to view this page. ("PARTYMGR_VIEW" or
"PARTYMGR_ADMIN" needed)</h3>
<%}%>
>=====================================================================
=========
```

这个程序很容易理解，需要注意的是每个文本框的名字，要跟 Schema StudyCustomer 的各
个段一致，以使程序中跟容易处理。

3.showofbiz.jsp

```
>=====================================================================
=========

<%@ taglib uri="ofbizTags" prefix="ofbiz" %>

<%@ page import="java.util.*, org.ofbiz.core.service.ModelService" %>
<%@ page import="org.ofbiz.core.util.*, org.ofbiz.core.pseudotag.*" %>
<%@ page import="org.ofbiz.core.entity.*" %>
<jsp:useBean id="security" type="org.ofbiz.core.security.Security"
scope="request" />
<jsp:useBean id="delegator"
type="org.ofbiz.core.entity.GenericDelegator" scope="request" />
<%if(security.hasEntityPermission("PARTYMGR", "_VIEW", session)) {%>

<form method="post" action="<ofbiz:url>/updateTest</ofbiz:url>"
name="updateofbiz">
<table width="300" align="center">
  <tr>
   <td>Id</td><td><input type="text" name="customerId" size="20"
value="<ofbiz:entityfield attribute="edit_cust"
field="customerId"/>"></td>
  </tr>
  <tr>
```

```
   <td>Name</td><td><input type="text" name="customerName" size="20"
value="<ofbiz:entityfield attribute="edit_cust"
field="customerName"/>"></td>
  </tr>
  <tr>
   <td>Note</td><td><input type="text" name="customerNote" size="30"
value="<ofbiz:entityfield attribute="edit_cust"
field="customerNote"/>"></td>
  </tr>
  <tr>
   <td></td>
   <td><input type="submit"></td>
  </tr>
</table>
</form>

<%}else{%>
  <h3>You do not have permission to view this page. ("PARTYMGR_VIEW" or
"PARTYMGR_ADMIN" needed)</h3>
<%}%>
```

>==========

这个程序中，主要是通过
<ofbiz:entityfield attribute="edit_cust" field="customerId"/>
把取到的对象的段显示出来，对象 edit_cust 是我们在程序中取到并放到 session 中的。

下面我们来配置 controller.xml 和 regions.xml，在 controller.xml 中加入：

>==========

```
   <request-map uri="createTestForm">
      <description>Show the create form</description>
      <security https="false" auth="false"/>
      <response name="success" type="view" value="createTestForm"/>
   </request-map>

   <request-map uri="testofbiz">
      <description>Test Ofbiz</description>
      <security https="false" auth="false"/>
      <response name="success" type="view" value="testofbiz"/>
```

```
   </request-map>

   <request-map uri="listtest">
      <description>List all records</description>
      <security https="false" auth="false"/>
      <event type="java" path="com.geeyo.ofbiz.TestOfbiz"
invoke="lookAllRecords" />
      <response name="success" type="view" value="listAllTest"/>
   </request-map>

   <request-map uri="showtest">
      <description>Show records</description>
      <security https="false" auth="false"/>
      <event type="java" path="com.geeyo.ofbiz.TestOfbiz"
invoke="findRecord" />
      <response name="success" type="view" value="showTest"/>
   </request-map>

   <request-map uri="createTest">
       <security https="true" auth="true"/>
       <event type="java" path="com.geeyo.ofbiz.TestOfbiz"
invoke="createNewRecord"/>
       <response name="success" type="request" value="listtest"/>
       <response name="error" type="view" value="createTestForm"/>
   </request-map>

   <request-map uri="updateTest">
      <description>update a record</description>
      <security https="false" auth="false"/>
      <event type="java" path="com.geeyo.ofbiz.TestOfbiz"
invoke="updateRecord" />
      <response name="success" type="request" value="listtest"/>
   </request-map>

   <request-map uri="removetest">
      <description>remove a record</description>
      <security https="false" auth="false"/>
      <event type="java" path="com.geeyo.ofbiz.TestOfbiz"
invoke="removeRecord" />
      <response name="success" type="request" value="listtest"/>
   </request-map>

   <view-map name="listAllTest" type="region"/>
   <view-map name="createTestForm" type="region"/>
```

```
   <view-map name="showTest" type="region"/>
>=============================================================================
=========
```

在 regions.xml 中加入：

```
>=============================================================================
=========
   <define id='createTestForm' region='MAIN_REGION'>
       <put section='title'>Create Ofbiz</put>
       <put section='content' content='/party/createofbiz.jsp'/>
   </define>

   <define id='listAllTest' region='MAIN_REGION'>
       <put section='title'>List Ofbiz</put>
       <put section='content' content='/party/listofbiz.jsp'/>
   </define>

   <define id='showTest' region='MAIN_REGION'>
       <put section='title'>Show Ofbiz</put>
       <put section='content' content='/party/showofbiz.jsp'/>
   </define>

>=============================================================================
=========
```

现在就完成了，我们重新启动 Ofbiz，然后用 IE 访问：
http://localhost:8080/partymgr/control/listtest，用 admin/ofbiz 登录后就可以
看到我们刚才的工作成果了，你现在可以增加/删除/修改记录。

6.Ofbiz 通过 XML 来完成数据库操作（非常强大的功能）

这是 Ofbiz 的一个非常强大的功能，可能通过简单的 XML 文件来完成数据增/删/改的处理，
这些处理在数据库应用中是非常多的，因为很多需要维护的数据，所以写程序也是最花时间的，
Ofbiz 把这些操作通过 XML 来完成，不能不说是一大革命---使我们不用写程序就可以完成大
部分处理，这是每个程序员都向往的终极目标。

我们下面举例来讲述一下，处理的数据还是利用我们前面创建的 StudyCustomer，使用 XML
配置文件来完成前面程序 TestOfbiz.java 的大部分操作。

在 c:\ofbiz\ofbiz\testOfbiz\com\geeyo\ofbiz 目录下创建文件 TestOfbizServices.xml，
该文件的内容如下：

>==========

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE simple-methods PUBLIC "-//OFBiz//DTD Simple Methods//EN"
"http://www.ofbiz.org/dtds/simple-methods.dtd">

<simple-methods>

   <!-- TestOfbiz methods -->
   <simple-method method-name="createNewRecord"
short-description="Create a new record">
       <check-permission permission="STUDYCUSTOMER"
action="_CREATE"><fail-message message="Security Error: to run
createRecord you must have the STUDYCUSTOMER_CREATE
permission"/></check-permission>        <check-errors/>

       <make-value entity-name="StudyCustomer"
value-name="newEntity"/>
       <set-pk-fields map-name="parameters" value-name="newEntity"/>
       <set-nonpk-fields map-name="parameters"
value-name="newEntity"/>

       <create-value value-name="newEntity"/>
   </simple-method>
   <simple-method method-name="updateRecord" short-description="Update
a record">
       <check-permission permission="STUDYCUSTOMER"
action="_UPDATE"><fail-message message="Security Error: to run
updateRecord you must have the STUDYCUSTOMER_UPDATE
permission"/></check-permission>

       <check-errors/>

       <make-value entity-name="StudyCustomer"
value-name="lookupPKMap"/>
       <set-pk-fields map-name="parameters" value-name="lookupPKMap"/>
       <find-by-primary-key entity-name="StudyCustomer"
map-name="lookupPKMap" value-name="lookedUpValue"/>
```

```
       <set-nonpk-fields map-name="parameters"
value-name="lookedUpValue"/>

       <store-value value-name="lookedUpValue"/>
   </simple-method>

   <simple-method method-name="findRecord" short-description="lookup a
record">
       <check-errors/>

       <make-value entity-name="StudyCustomer"
value-name="lookupPKMap"/>
       <set-pk-fields map-name="parameters" value-name="lookupPKMap"/>
       <find-by-primary-key entity-name="StudyCustomer"
map-name="lookupPKMap" value-name="edit_cust"/>
             <field-to-session field-name="edit_cust"/>
   </simple-method>

   <simple-method method-name="removeRecord" short-description="Delete
a record">
       <check-permission permission="STUDYCUSTOMER"
action="_DELETE"><fail-message message="Security Error: to run
deleteRecord you must have the STUDYCUSTOMER_DELETE
permission"/></check-permission>
       <check-errors/>

       <make-value entity-name="StudyCustomer"
value-name="lookupPKMap"/>
       <set-pk-fields map-name="parameters" value-name="lookupPKMap"/>
       <find-by-primary-key entity-name="StudyCustomer"
map-name="lookupPKMap" value-name="lookedUpValue"/>
       <remove-value value-name="lookedUpValue"/>
   </simple-method>

   <simple-method method-name="lookAllRecords"
short-description="lookup suitable records">
       <check-errors/>
       <find-by-and entity-name="StudyCustomer"
list-name="search_results"/>
             <field-to-session field-name="search_results"/>
   </simple-method>

</simple-methods>
```

>==========

上面的 XML 基本是不用解释的，定义了

createNewRecord
updateRecord
lookAllRecords
removeRecord
findRecord

这几个方法，而且都有对用户权限的检查，这几个方法对应于前面 TestOfbiz.java 中的几个方法，
这样来做数据库操作显然比用 Java 程序写要简单得多，

下面还需要在 controller.xml（具体文件得位置请参照前面的教程）更改一下 mapping 的设置，
更改如下，以前使用 TestOfbiz.java 时的配置我以注释的方式保留着以做参照：

>==========

```
   <request-map uri="createTestForm">
      <description>Show the create form</description>
      <security https="false" auth="false"/>
      <response name="success" type="view" value="createTestForm"/>
   </request-map>

   <request-map uri="listtest">
      <description>List all records</description>
      <security https="false" auth="false"/>
         <event type="simple"
path="com/geeyo/ofbiz/TestOfbizServices.xml" invoke="lookAllRecords"
/>
      <response name="success" type="view" value="listAllTest"/>
   </request-map>

   <request-map uri="showtest">
      <description>Show records</description>
      <security https="false" auth="false"/>
         <event type="simple"
path="com/geeyo/ofbiz/TestOfbizServices.xml" invoke="findRecord" />
      <response name="success" type="view" value="showTest"/>
   </request-map>
```

```
   <request-map uri="createTest">
       <security https="true" auth="true"/>
       <event type="simple"
path="com/geeyo/ofbiz/TestOfbizServices.xml"
invoke="createNewRecord"/>
       <response name="success" type="request" value="listtest"/>
       <response name="error" type="view" value="createTestForm"/>
   </request-map>

   <request-map uri="updateTest">
      <description>update a record</description>
      <security https="false" auth="false"/>
      <event type="simple"
path="com/geeyo/ofbiz/TestOfbizServices.xml" invoke="updateRecord" />
      <response name="success" type="request" value="listtest"/>
   </request-map>

   <request-map uri="removetest">
      <description>remove a record</description>
      <security https="false" auth="false"/>
      <event type="simple"
path="com/geeyo/ofbiz/TestOfbizServices.xml" invoke="removeRecord" />
      <response name="success" type="request" value="listtest"/>
   </request-map>

   <view-map name="listAllTest" type="region"/>
   <view-map name="createTestForm" type="region"/>
   <view-map name="testofbiz" type="region"/>
   <view-map name="showTest" type="region"/>

>====================================================================
```

配置该文件的方法请参照前面的教程，regions.xml 不需改动。

配置完后请用前面讲过的方法访问 URL:
http://localhost:8080/partymgr/control/listtest

现在我们可以看到，Ofbiz 在 MVC 方面做得非常好，我们可以把后端的处理程序从 java 改
成用 XMl 控制，而其他部分（像 JSP）不需任何改动，这可以保证我们系统各部分的独立性。