

OFBiz 开发快速入门

Hongs 收集整理

Contact: billhongs@gmail.com

软件环境

这里 OFBiz 使用内嵌的 Derby 数据库，正式环境一般不用，下面的开发采用 Mysql5.0 以上

JDK 安装

推荐使用 JDK6 最新版

假设安装在 C:\Java\jdk1.6.0

在环境变量中设置 JAVA_HOME= C:\Java\jdk1.6.0

OFBiz 下载

通过 SVN 下载 OFBiz Release4.0(推荐使用 TortoiseSVN 工具)

<http://svn.apache.org/repos/asf/ofbiz/branches/release4.0>

假设下载到 C:\Java\ofbiz-release4.0

OFBiz 运行

进入命令行，C:\Java\ofbiz-release4.0

运行 ant run-install 命令编译安装，这个估计得过一小段时间

运行 startofbiz.bat

查看结果

<http://localhost:8080/ecommerce>

OFBiz 概览

OFBiz 的宗旨，还是使用一系列自创的开源技术，开发大规模的企业级应用程序。它的目标，不是工作流，也不是 appfuse 这样一个整合其他开源技术的演示性的网站，而是一个综合的、企业级的电子商务网站。

总的目录

首先，进入 Ofbiz 目录：

C:\Java\ofbiz-release4.0

2008-12-21	04:13		10,856 .classpath
2008-12-21	04:13		1,127 .project
2008-12-21	04:13	<DIR>	.svn
2008-12-21	04:13		1,110 ant
2008-12-21	04:13		1,121 ant.bat
2008-12-21	04:13		5,981 APACHE2_HEADER
2008-12-21	04:12	<DIR>	applications
2008-12-21	04:13		10,006 build.xml
2008-12-21	04:13	<DIR>	framework
2008-12-21	04:10	<DIR>	hot-deploy
2008-12-21	04:13		1,233 ij.ofbiz
2008-12-21	04:13		1,997 KEYS
2008-12-21	04:13		102,937 LICENSE
2008-12-21	04:13		362 mergefromtrunk.bat
2008-12-21	04:13		161 mergefromtrunk.sh
2008-12-21	04:13		10,687 NOTICE
2008-12-21	04:13		56,336 ofbiz.jar
2008-12-21	04:13		8,588 OPTIONAL_LIBRARIES
2008-12-21	04:13		4,140 rc.ofbiz
2008-12-21	04:13		3,226 README
2008-12-21	04:10	<DIR>	runtime
2008-12-21	04:13	<DIR>	specialpurpose
2008-12-21	04:13		1,361 startofbiz.bat
2008-12-21	04:13		1,710 startofbiz.sh
2008-12-21	04:13		1,485 stopofbiz.sh

OFBIZ 采用了基于组件的架构。这就是说，每一个应用程序在一个组件内。这允许外部的和定制化的应用程序能够很容易的被包含进来。

applications

C:\java\ofbiz-release4.0\ applications

2008-12-21	04:13	<DIR>	accounting
2008-12-21	04:12		3,353 build.xml
2008-12-21	04:12		1,858 component-load.xml
2008-12-21	04:13	<DIR>	content
2008-12-21	04:13	<DIR>	ecommerce

2008-12-21	04:11	<DIR>	humanres
2008-12-21	04:13	<DIR>	manufacturing
2008-12-21	04:13	<DIR>	marketing
2008-12-21	04:13	<DIR>	order
2008-12-21	04:13	<DIR>	party
2008-12-21	04:13	<DIR>	product
2008-12-21	04:13	<DIR>	securityext
2008-12-21	04:13	<DIR>	workeffort

Applications 目录，包含了 OFBiz 核心的应用程序组件，如订单管理，电子商务存储等。

component-load.xml 文件配置需要载入哪几个应用程序组件。这里的每一个组件，都是一个基于 OFBIZ 构建的 Web 应用程序。

framework

C:\java\ofbiz-release4.0\framework

2008-12-21	04:13	<DIR>	appserver
2008-12-21	04:13	<DIR>	base
2008-12-21	04:13		6,868 build.xml
2008-12-21	04:13	<DIR>	catalina
2008-12-21	04:13	<DIR>	common
2008-12-21	04:13		2,501 component-load.xml
2008-12-21	04:13	<DIR>	datafile
2008-12-21	04:13	<DIR>	entity
2008-12-21	04:13	<DIR>	entityext
2008-12-21	04:13	<DIR>	example
2008-12-21	04:13	<DIR>	geronimo
2008-12-21	04:13	<DIR>	guiapp
2008-12-21	04:12	<DIR>	images
2008-12-21	04:13	<DIR>	jetty
2008-12-21	04:13	<DIR>	minilang
2008-12-21	04:13	<DIR>	security
2008-12-21	04:13	<DIR>	service
2008-12-21	04:12	<DIR>	shark
2008-12-21	04:13	<DIR>	testtools
2008-12-21	04:13	<DIR>	webapp
2008-12-21	04:13	<DIR>	webtools
2008-12-21	04:13	<DIR>	widget
2008-12-21	04:13	<DIR>	workflow

Framework 框架目录，包含 OFBIZ 框架的组件，例如实体引擎和服务引擎。这是 OFBIZ 框架的核心，其他应用程序都是基于它来构建的。component-load.xml 文件配置需要载入哪几个框架组件。

specialpurpose

C:\java\ofbiz-release4.0\specialpurpose

specialpurpose 专门目录，包含一些其他的应用程序，不是 OFBIZ 核心的一部分。

hot-deploy

C:\java\ofbiz-release4.0\hot-deploy

hot-deploy 热部署目录，是另一个目录。它的组件能够被删除和自动载入。这里没有 component-load.xml 这个文件。本目录中所有的组件，都会在 framework 和 application 目录下的组件被导入之后导入。

runtime

C:\java\ofbiz-release4.0\runtime

现在，让我们进入一个应用程序，看看里面有什么：

```
2008-12-21  04:13    <DIR>          build
2008-12-21  04:11                6,332 build.xml
2008-12-21  04:10    <DIR>          config
2008-12-21  04:11    <DIR>          data
2008-12-21  04:11    <DIR>          entitydef
2008-12-21  04:11                4,282 ofbiz-component.xml
2008-12-21  04:10    <DIR>          script
2008-12-21  04:10    <DIR>          servicedef
2008-12-21  04:11    <DIR>          src
2008-12-21  04:11    <DIR>          templates
2008-12-21  04:11    <DIR>          webapp
2008-12-21  04:10    <DIR>          widget
```

Build 目录是已编译的 java 代码和任何 Java 库。它们能够使这个应用程序运行。（但是不能仅仅靠它们运行。可以仅靠 webapp 目录下的一个或者多个 web 应用程序运行）

build.xml 文件是 ant 文件，用于测试和构建这个应用程序。

config 目录包含配置文件，例如多语言下的国际化 UI 标签的配置文件。

data 目录包含种子和演示数据，xml 格式。

entitydef 目录，包含这个应用程序的数据模型定义。

script 脚本目录，包含业务逻辑的脚本文件。

servicedef 目录，包含 services 服务，它们是细粒度的业务逻辑（类似于方法或函数）。

src 源文件目录，包含实现业务逻辑的 Java 类文件。

webapp 目录，是 web 接口，是应用程序的前端。一个 OFBIZ 应用程序能有多多个 webapp 应用程序。

Eclipse 开发环境的搭建

准备工作：

- 1、 安装 JDK6 或以上版本，设置好环境变量 JAVA_HOME
把%JAVA_HOME%\bin 加入 path
- 2、 安装 MySQL 5.0 或以上版本
- 3、 在 <http://www.eclipse.org/downloads/> 下载最新 Eclipse 版本，我下的是 3.4.2 版的 Eclipse IDE for Java EE Developers，并解压至工作目录（我解压至"C:\\"，即工作目录为"C:\\eclipse"）

OFBiz Eclipse

初步设定

如果你还没有把 ofbiz 导入 Eclipse，请先把 ofbiz 导入 Eclipse。操作如下 File > New > Java Project, 此时选择 Create Project from Existing Source. 注意项目名称随便填个(比如 ofbiz), 目录选择你解压后的 OFBiz 目录，c:\ofbiz-release4.0。

New Java Project

Create a Java project
Create a Java project in the workspace or in an external location.

Project name: ofbiz-release4.0

Contents

☐ Create new project in workspace

☒ Create project from existing source

Directory: C:\Java\ofbiz-release4.0

Browse...

JRE

☒ Use default JRE (Currently 'jdk1.6.0')

Configure default...

☐ Use a project specific JRE:

jdk1.6.0

☐ Use an execution environment JRE:

JavaSE-1.6

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files

Configure default...

Working sets

☐ Add project to working sets

Working sets:

Select...

The wizard will automatically configure the JRE and the project layout based on the specified existing source.

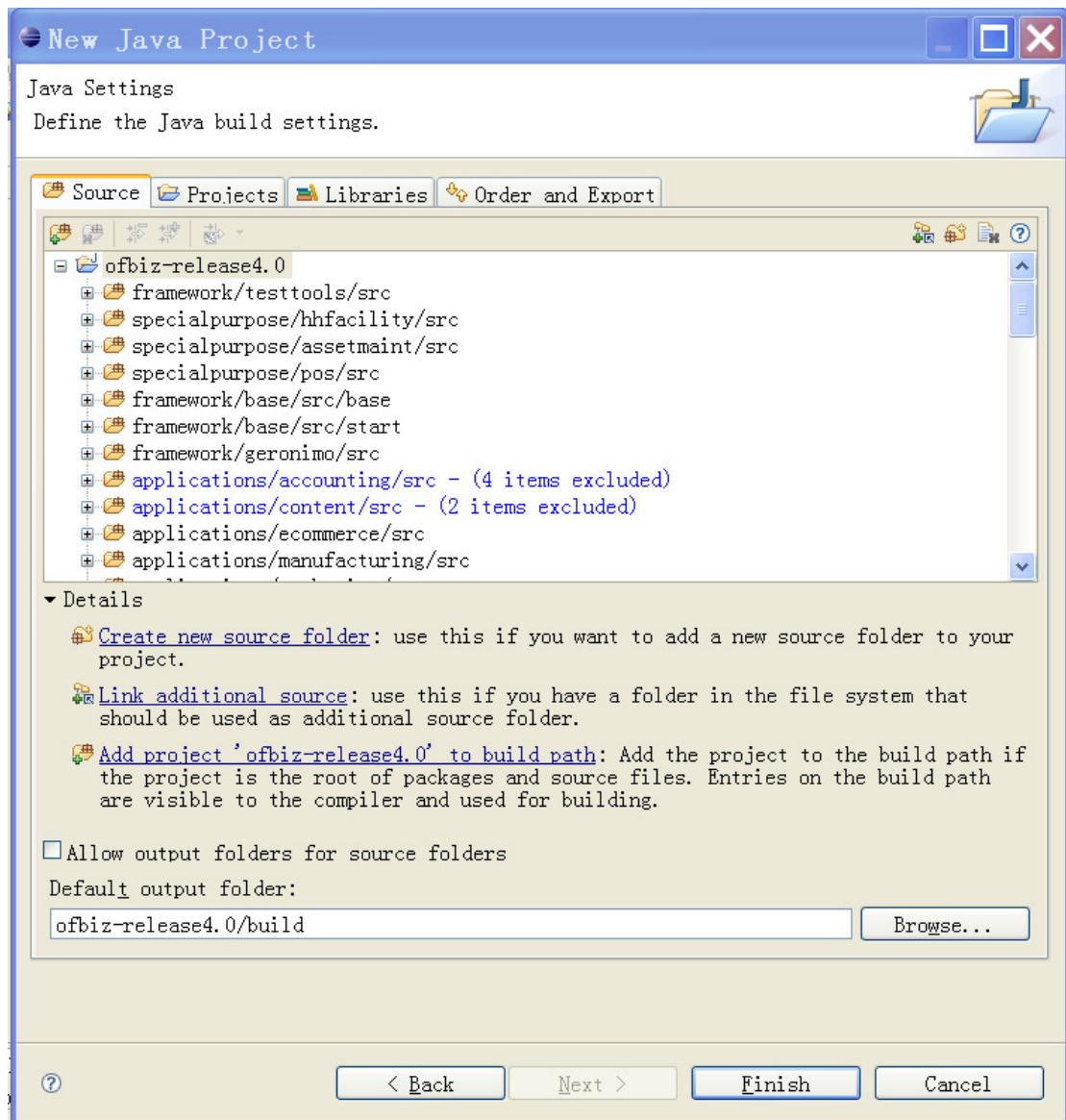
?

< Back

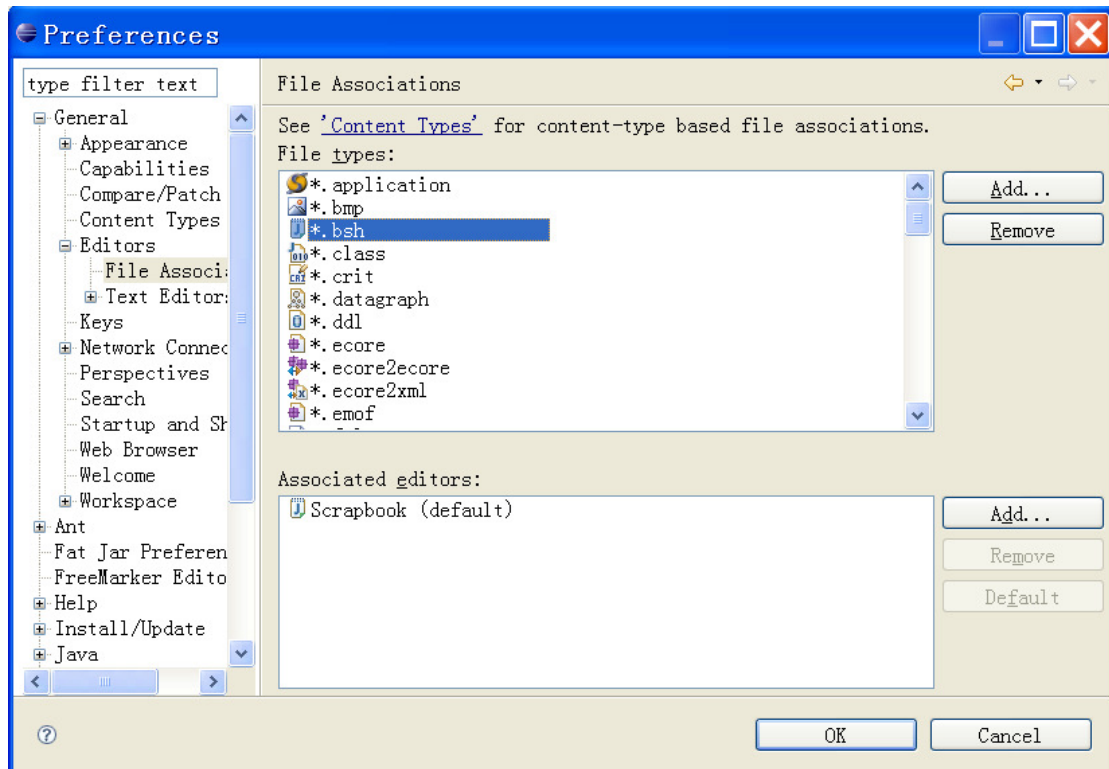
Next >

Finish

Cancel



下一步是使 Eclipse 能正确识别你的.xml 和.ftl , 以及.bsh 等文件。首先, 安装 Freemarker 的 Eclipse 插件。最后, 到 Windows > Preferences > General > Editors > File Associations 点击 add "*.bsh" 并且在 Associate editor. 添加他使用 Scrapbook。



用 OFBiz 创建一个完整的应用 hello

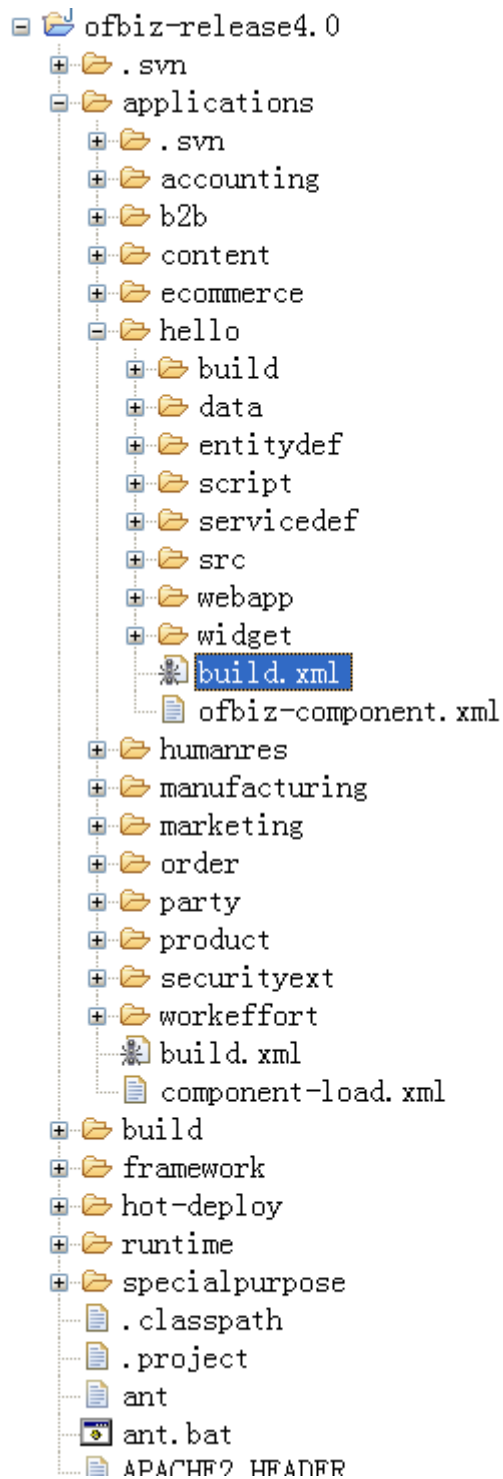
创建一个应用

目录结构

首先创建如下所示的目录结构

applications 下的 hello

其中 ofbiz-component.xml、build.xml、web.xml、controller.xml 几个文件可以从任意一个模块中拷过来，再作些修改。



几个重要文件

增加 ofbiz-component.xml

```
ofbiz-component.xml x build.xml
* the rights to use, copy, modify, merge, publish, distribute, sublicense,
* and/or sell copies of the Software, and to permit persons to whom the
* Software is furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included
* in all copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
* OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
* IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
* CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
* OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
* THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*
*
-->

<ofbiz-component name="hello"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/ofbiz-component.xsd">
  <resource-loader name="main" type="component"/>
  <classpath type="jar" location="build/lib/*"/>
  <classpath type="dir" location="script"/>

  <entity-resource type="model" reader-name="main" loader="main" location="entitydef/entitymodel.xml"/>
  <entity-resource type="group" reader-name="main" loader="main" location="entitydef/entitygroup.xml"/>
  <entity-resource type="data" reader-name="seed" loader="main" location="data/HobbiesData.xml"/>

  <service-resource type="model" loader="main" location="servicedef/services.xml"/>

  <webapp name="hello"
    title="My Hello OFBiz Application"
    server="default-server"
    location="webapp/hello"
    mount-point="/hello"
    app-bar-display="false"/>
</ofbiz-component>
```

增加 build.xml，这个是 ant 用的。

```
ofbiz-component.xml build.xml
<?xml version="1.0"?>
<!--
 * Copyright (c) 2005 The Open For Business Project and respected authors.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
 * OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
 * THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
 * This is the Open for Business Party Build Script.
 * $Id: build.xml 4537 2005-02-21 23:55:27Z jaz $
-->

<project name="Hello Sample Application" default="jar" basedir=".">

    <!-- ===== -->
    <!-- Initialization of all property settings -->
    <!-- ===== -->

    <target name="init">
        <property environment="env"/>
        <property name="desc" value="Hello Application"/>
        <property name="name" value="ofbiz-hello"/>
        <property name="src.dir" value="src"/>
        <property name="dtd.dir" value="dtd"/>
        <property name="lib.dir" value="lib"/>
        <property name="build.dir" value="build"/>
    </target>
</project>
```

在 hello/webapp/hello/WEB-INF 下增加 web.xml

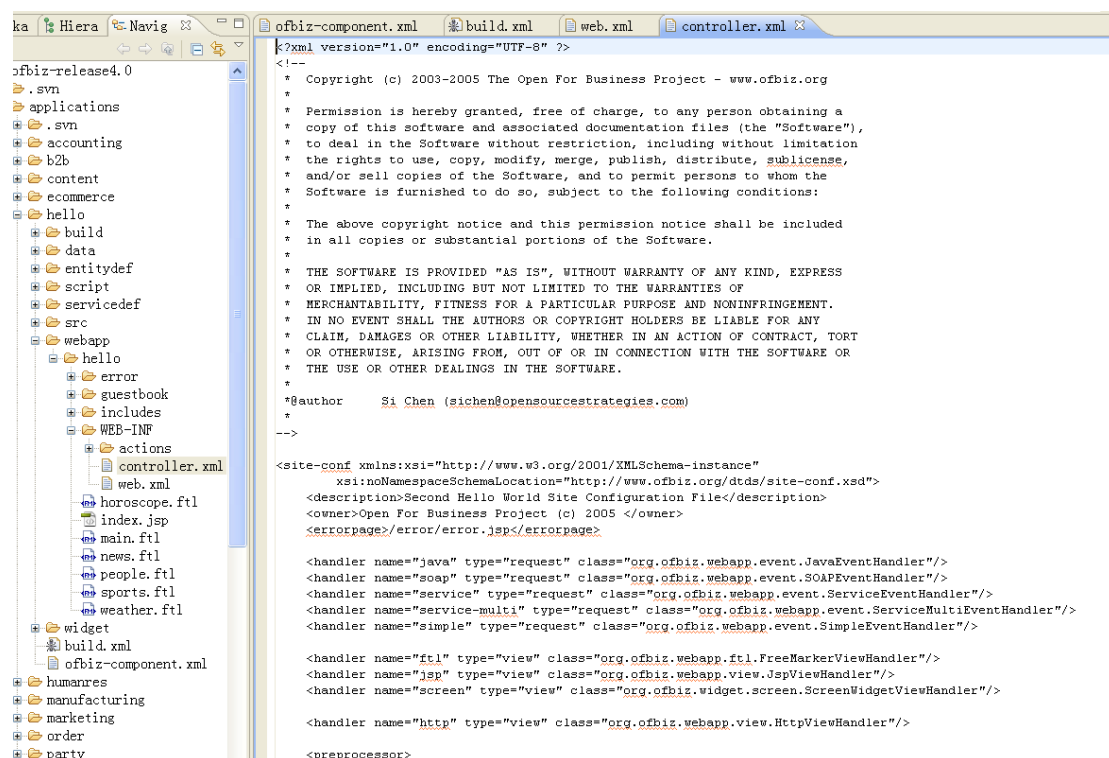
```
ofbiz-component.xml build.xml web.xml
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN" "http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
    <display-name>Hello</display-name>
    <description>The First Hello World Application</description>

    <context-param>
        <param-name>entityDelegatorName</param-name>
        <param-value>default</param-value>
        <description>The Name of the Entity Delegator to use, defined in entityengine.xml</description>
    </context-param>
    <context-param>
        <param-name>localDispatcherName</param-name>
        <param-value>hello</param-value>
        <description>A unique name used to identify/recognize the local dispatcher for the Service Engine</description>
    </context-param>
    <context-param>
        <param-name>serviceReaderUrls</param-name>
        <param-value>/WEB-INF/services.xml</param-value>
        <description>Configuration File(s) For The Service Dispatcher</description>
    </context-param>

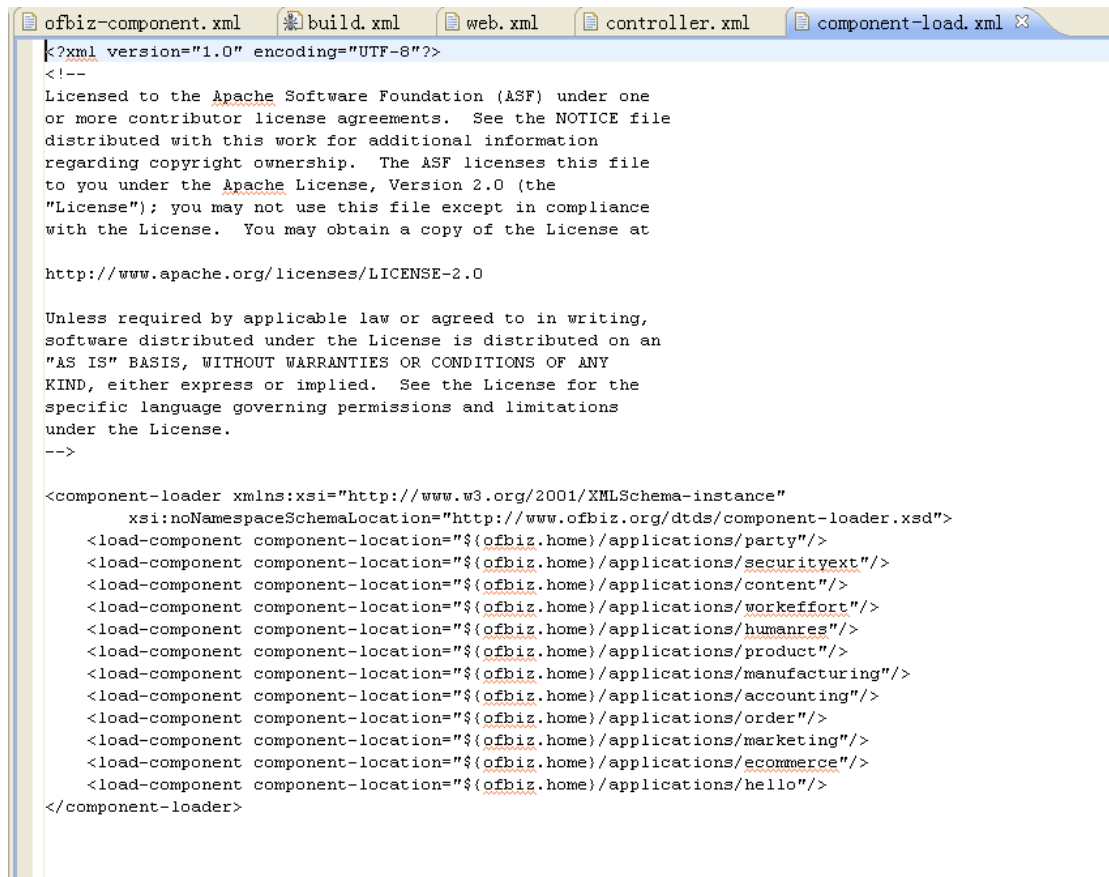
    <filter>
        <filter-name>ContextFilter</filter-name>
        <display-name>ContextFilter</display-name>
        <filter-class>org.ofbiz.webapp.control.ContextFilter</filter-class>
        <init-param>
            <param-name>disableContextSecurity</param-name>
            <param-value>N</param-value>
        </init-param>
        <init-param>
            <param-name>allowedPaths</param-name>
            <param-value>/control:/select:/index.html:/index.jsp:/default.html:/default.jsp:/images:/includes/maincss.css</param-value>
        </init-param>
        <init-param>
            <param-name>errorCode</param-name>
            <param-value>403</param-value>
        </init-param>
    </filter>
</web-app>
```

在 hello/webapp/hello/WEB-INF 下增加 controller.xml



模块加载文件

在 applications 目录下



```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Licensed to the Apache Software Foundation (ASF) under one
or more contributor license agreements. See the NOTICE file
distributed with this work for additional information
regarding copyright ownership. The ASF licenses this file
to you under the Apache License, Version 2.0 (the
"License"); you may not use this file except in compliance
with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing,
software distributed under the License is distributed on an
"AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
KIND, either express or implied. See the License for the
specific language governing permissions and limitations
under the License.
-->

<component-loader xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/component-loader.xsd">
  <load-component component-location="{ofbiz.home}/applications/party"/>
  <load-component component-location="{ofbiz.home}/applications/securitytext"/>
  <load-component component-location="{ofbiz.home}/applications/content"/>
  <load-component component-location="{ofbiz.home}/applications/workeffort"/>
  <load-component component-location="{ofbiz.home}/applications/humanres"/>
  <load-component component-location="{ofbiz.home}/applications/product"/>
  <load-component component-location="{ofbiz.home}/applications/manufacturing"/>
  <load-component component-location="{ofbiz.home}/applications/accounting"/>
  <load-component component-location="{ofbiz.home}/applications/order"/>
  <load-component component-location="{ofbiz.home}/applications/marketing"/>
  <load-component component-location="{ofbiz.home}/applications/ecommerce"/>
  <load-component component-location="{ofbiz.home}/applications/hello"/>
</component-loader>
```

定义一个数据模型

第一步是定义数据模型。我们想用户信息和他们的爱好并且查看每个人的所有爱好（或者，所有的人分享一种爱好）。数据模型是定义人员，爱好，和人员-爱好的联系。

在关系型数据库中,你将定义两个 table,其中一个 是人员,另一个是爱好。 还有定义他们之间联系的第三个表. 第三个表允许一个人员关联许多爱好, 反之亦然. 你将定义人员和爱好的外键来约束第三个表。

OFBiz 工作原理也是类似. 你可以定义两个实体,我们将定义 HelloPerson 和 HelloHobby , 和一个连接实体, HelloPersonHobby , 并建立了它们之间的关系. 他们之间关系主要是外键约束,而且不需要你自己去定义。(当外键改变, 需要改变你的代码)。

注意:我们正在创造一个应用完全独立的数据模型来作说明. 如果我们是在建立一个真正完整的应用, 最好的做法是尽可能利用许多现有的 OFBiz 框架实体来做,而不是创造自己 HelloPerson ,我们将利用现有的框架 OFBiz Party/Person/PartyGroup 这三个已有实体。

定义数据模型, 在 hello 下 创建 entitydef 文件夹在 entitydef /目录内 ,创建 entitymodel.xml 和 entitygroup.xml

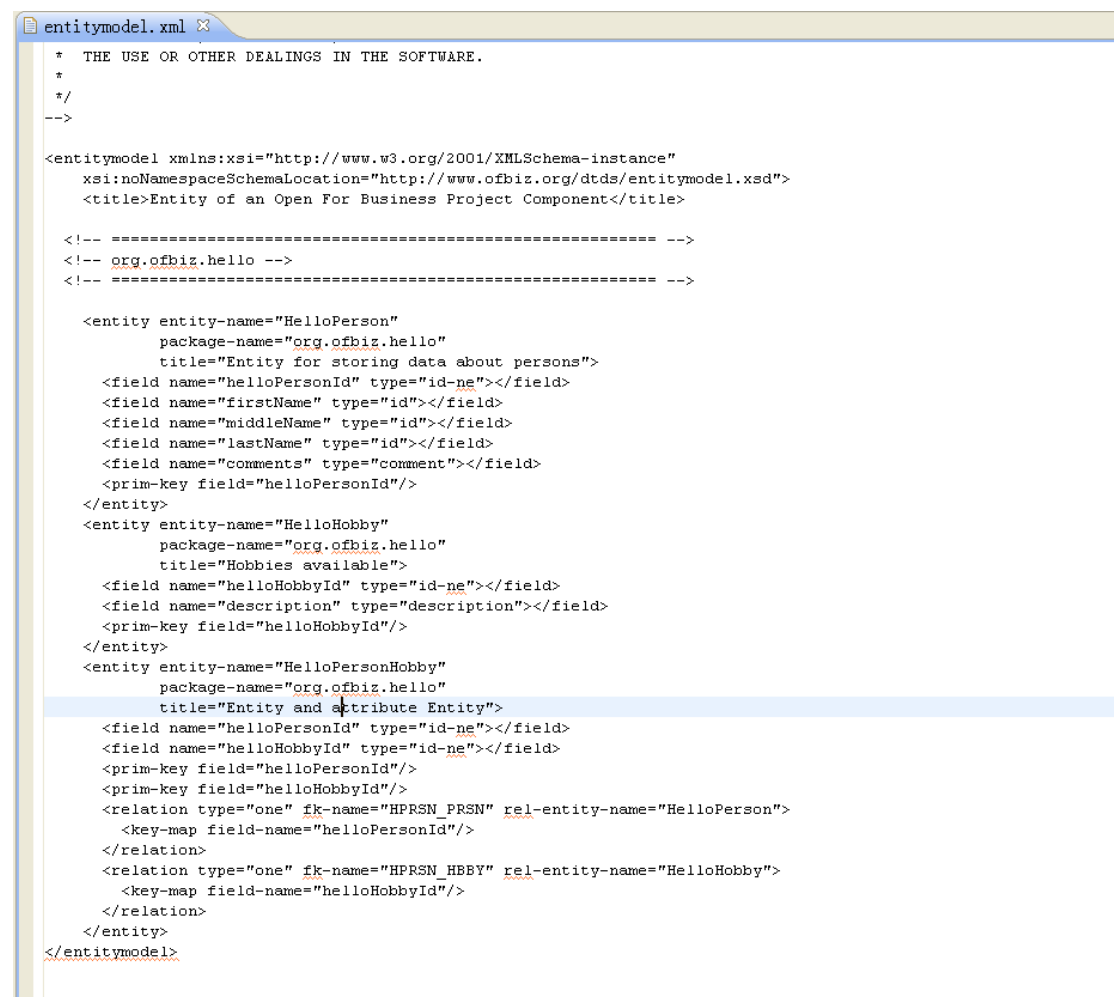
HelloPerson 和 HelloHobby 各有一个主键, HelloPersonHobby 有两个主键, 它用来链接 HelloPerson 和 HelloHobby . 这是好的做法,让你外键关联,使调试容易,避免发生意外. 外键的名称 OFBiz 由产生.

定义这些实体在另一文件里, entitygroup.xml, 这也是很重要的(但很容易忘记) 。 在 entitydef /目录,这样 OFBiz 就会知道哪些数据写入数据库中,以供使用人):否则,将实体存在

OFBiz 的,但是当你真正尝试使用他们,你会得到这个错误:

org.ofbiz.entity.GenericEntityException: Helper name not found for entity HelloPerson

最后,你必须增加这一行到你的 ofbiz-component.xml 中的,这样让 OFBiz 知道这个实体是应用程序的一个组件



```
entitymodel.xml
* THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*
*/
-->

<entitymodel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/entitymodel.xsd">
  <title>Entity of an Open For Business Project Component</title>

  <!-- ===== -->
  <!-- org.ofbiz.hello -->
  <!-- ===== -->

  <entity entity-name="HelloPerson"
    package-name="org.ofbiz.hello"
    title="Entity for storing data about persons">
    <field name="helloPersonId" type="id-ne"></field>
    <field name="firstName" type="id"></field>
    <field name="middleName" type="id"></field>
    <field name="lastName" type="id"></field>
    <field name="comments" type="comment"></field>
    <prim-key field="helloPersonId"/>
  </entity>
  <entity entity-name="HelloHobby"
    package-name="org.ofbiz.hello"
    title="Hobbies available">
    <field name="helloHobbyId" type="id-ne"></field>
    <field name="description" type="description"></field>
    <prim-key field="helloHobbyId"/>
  </entity>
  <entity entity-name="HelloPersonHobby"
    package-name="org.ofbiz.hello"
    title="Entity and attribute Entity">
    <field name="helloPersonId" type="id-ne"></field>
    <field name="helloHobbyId" type="id-ne"></field>
    <prim-key field="helloPersonId"/>
    <prim-key field="helloHobbyId"/>
    <relation type="one" fk-name="HPRSN_PRSN" rel-entity-name="HelloPerson">
      <key-map field-name="helloPersonId"/>
    </relation>
    <relation type="one" fk-name="HPRSN_HBBY" rel-entity-name="HelloHobby">
      <key-map field-name="helloHobbyId"/>
    </relation>
  </entity>
</entitymodel>
```

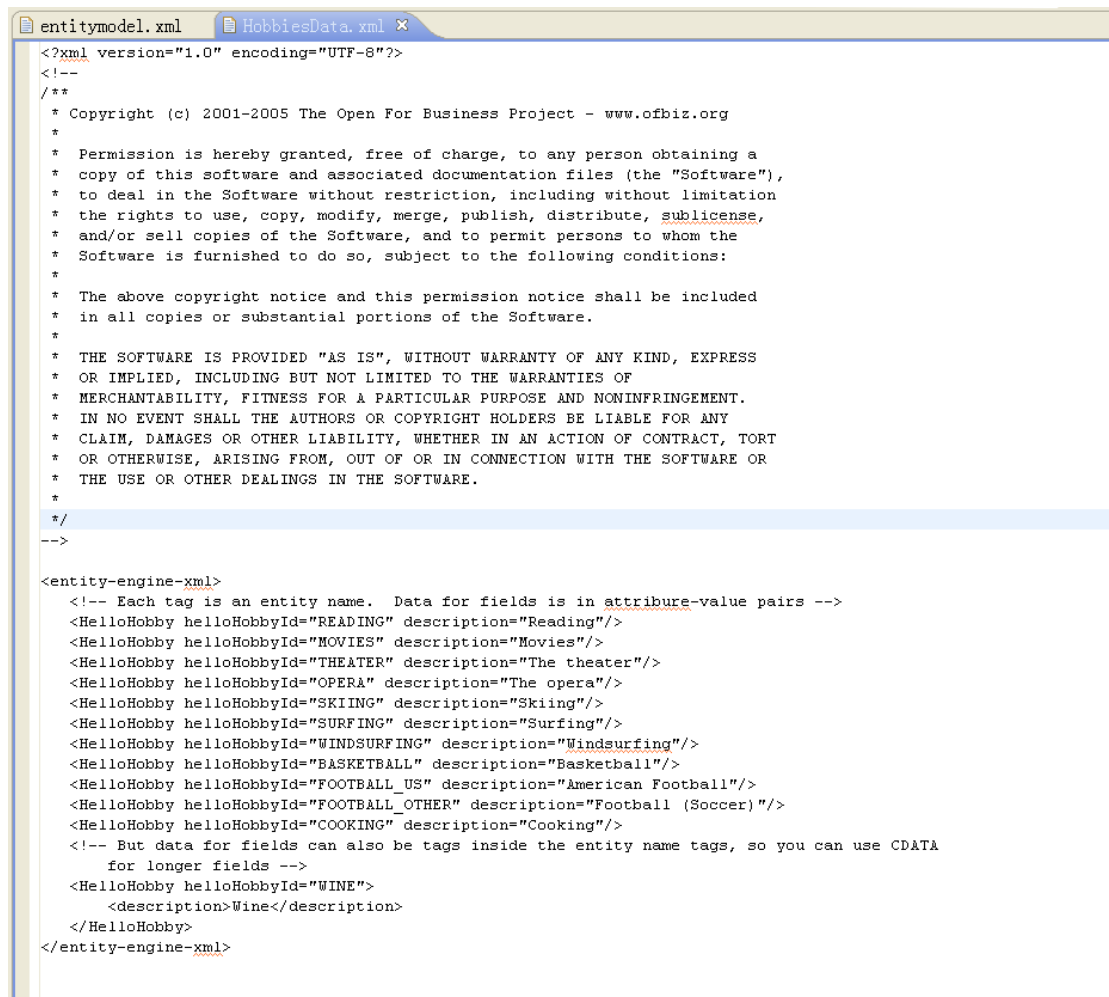
创建例子数据

现在让我们一起创建一些例子数据,为我们的 hobbies. 在大多数 OFBiz 的应用程序中,我们都将创建一个 data/ 目录内的应用,并创建一个 XML 文件的例子数据. 让我们看看 HobbiesData.xml :

这个实体引擎 XML 文件, 是一个标准 OFBiz 的 XML 格式的输入及输出数据. 实在是相当简单. 使用名称是实体(" HelloHobby "在我们的情况)的标签名称, 使用名称的领域要么属性或作为内标签(见上次值为"葡萄酒"

你的属性可以是属性的数值或内部的标签的值, 如果你有很长的数值在 CDATA 段落里面是最好的. 另外, 主键不须大写或隔着_, 但在其他 OFBiz 的应用程序这是一个规范.

现在你准备加载例子数据. 到 Web Tools 主界面,你会看到链接的" XML Import ". 点击" XML Import ",并于下一个画面,它会提示你文件夹的名称, 在你的 OFBiz 中的路径. 我通常不点击任何复选框和仅仅是" Import ". 如果你是成功的, 同一个介面底部会告诉你,有多少数据添加:



```
<?xml version="1.0" encoding="UTF-8"?>
<!--
/**
 * Copyright (c) 2001-2005 The Open For Business Project - www.ofbiz.org
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
 * OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
 * THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
-->

<entity-engine-xml>
  <!-- Each tag is an entity name.  Data for fields is in attribute-value pairs -->
  <HelloHobby helloHobbyId="READING" description="Reading"/>
  <HelloHobby helloHobbyId="MOVIES" description="Movies"/>
  <HelloHobby helloHobbyId="THEATER" description="The theater"/>
  <HelloHobby helloHobbyId="OPERA" description="The opera"/>
  <HelloHobby helloHobbyId="SKIING" description="Skiing"/>
  <HelloHobby helloHobbyId="SURFING" description="Surfing"/>
  <HelloHobby helloHobbyId="WINDSURFING" description="Windsurfing"/>
  <HelloHobby helloHobbyId="BASKETBALL" description="Basketball"/>
  <HelloHobby helloHobbyId="FOOTBALL_US" description="American Football"/>
  <HelloHobby helloHobbyId="FOOTBALL_OTHER" description="Football (Soccer)"/>
  <HelloHobby helloHobbyId="COOKING" description="Cooking"/>
  <!-- But data for fields can also be tags inside the entity name tags, so you can use CDATA
       for longer fields -->
  <HelloHobby helloHobbyId="WINE">
    <description>Wine</description>
  </HelloHobby>
</entity-engine-xml>
```

除了增加你的属性和数据之外, 当你的数据被创建并且被更新时, OFBiz 自动地创建了时间戳, 这是为了横跨 OFBiz 多个事例同步数据。

最后, 如果你增加这个命令到你的 ofbiz-component.xml 文件, 在安装过程期间 OFBiz 可以自动地安装你的例子数据:

OFBiz 实际上让你定义数据是否有“例子数据”, 这样为你的 app 可以展示, 例子数据仅仅是展示用的。

创建业务逻辑

既然我们已经定义了数据模型, 我们可以写一种简单的应用以 delegator 直接地访问实体。 这是 OFBiz 应用程序的标准操作, 但是, 做为业务逻辑的分层: 新建, 更新和删除。 delegator 直接地为查找数据使用, 如更加复杂的查寻。

创建业务逻辑的过程是有二步。首先, 你定义了通用的业务逻辑在 XML 文件(使用 class 和 方法 或者 脚本), 来告诉 OFBiz 业务逻辑引擎你的参数和位置, 第二, 你实现服务可以用 java, OFBiz minilang, 或者另一种脚本语言。

业务逻辑通常定义在一个 `servicedef/` 目录里面，并且包括一个或更多 `services.xml` 文件。这我们的 `services.xml` 文件：

注意 `services.xml` 是参考实体执行。同时直接地与 标签执行。这些自动属性标签保存你时间并且使你的应用更加容易维护。

你在你的 `ofbiz-component.xml` 也会需要参考 `service` 资源。另外，你必须创建 并告诉它在哪里的 `ofbiz-component.xml` 装载 apps。这就是为什么我们的 `ofbiz-component` 文件看起来相似，在增加 `classpaths` 以后，实体体定义、`service` 定义和例子数据：

现在它是否看起来和其他 `ofbiz-component.xml` 很相似？

现在创建 `services`。A Java `service` 流行的写法是在你的应用的 `src/` 目录里面被写的：一个公共的 `class` 有两个参数的公开静态方法，`DispatchContext` 为得到一个对象是你输入参数的 `delegators`, `dispatchers`, `locale`, and `security`, 和称上下文的 `map` 并且退回结果 `map`：

Java `services` 也将需要编译，需要适当的 `classpaths` 的知识为其他 OFBiz apps。这可以使用 `ant` 并且 `build.xml` 构建脚本，当然你可以从另一种应用通常复制。它是在所有 OFBiz 应用和框架中一个相当长的标准文件，因此我这里不会包括它。

当写你的 `service` 和创造 `build.xml` 脚本，你能建立你的象这样的 `Javaservice`：

构建过程基本上采取了在 `src/` 目录的所有您的文件，编译了他们并且把他们放入构建或 `lib` 目录。

`Minilang` 比较起来是简单的。简单的 `minilang service` 在 `script/` 目录里面并且是一个 XML 文件。由于它是专门为共同的 OFBiz 应用任务设计，例如 查寻数据，存放数据，检查 `premissions`，并且与现有的实体一起使用，并且执行业务逻辑，它使那些任务工作非常容易：

最后，测试它，重新启动 OFBiz 装载所有新的定义在 `ofbiz-component.xml` 和 `services.xml`。然后，打开 `beanshell` 窗口并且测试我们的 `service` (注意：您首先应该 下载 `bshcontainer.bsh` 并且把它放入在你的 `ofbiz/` 目录)

这里，`beanshell` 称执行 `service` 的服务调度员，他们是成功的，因此正确的数据被创建。(当然，它实际上采取了大约六次尝试，但我不会和你谈论细节使你不耐烦。)

那么现在你创造了数据模型和 `service` 为他们。下一个步骤将放他们入 Web 应用程序。

```
services.xml x
<?xml version="1.0" encoding="UTF-8"?>
<!--
/**
 * Copyright (c) 2001-2005 The Open For Business Project - www.ofbiz.org
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
 * OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
 * THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
-->

<services xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/services.xsd">
  <description>Hello Services</description>

  <!-- this will be implemented in Java -->
  <service name="createHelloPerson" engine="java"
          location="org.ofbiz.hello.HelloServices" invoke="createHelloPerson">
    <description>Create a HelloPerson</description>
    <auto-attributes mode="IN" entity-name="HelloPerson" include="nonpk" optional="true"/>
    <attribute name="helloPersonId" mode="OUT" type="String" optional="false"/>
  </service>

  <!-- this will be implemented in OFBiz minilang -->
  <service name="createHelloPersonHobby" engine="simple"
          location="org.ofbiz.hello.HelloServices.xml" invoke="createHelloPersonHobby">
    <description>Create a HelloPersonHobby which links a person and a hobby</description>
    <auto-attributes mode="IN" entity-name="HelloPersonHobby" include="pk" optional="false"/>
  </service>
</services>
```

扩展你的 web app 以 screen-widget, decorator, and actions

创建屏幕

OFBiz screen-widget 使用汇集复杂页与显示元件许多更小的片断。(如果您使用了 OFBiz 的更早的版本, 它替换 JPublish 和 regions 框架。)

第一步将在您的 webapp 并且创建 includes/目录和文件 header.ftl、 footer.ftl。 如果您想要使用通用的 stylesheet 和 graphics, 这是使用它最好的地方。 您也可以想要从 main.ftl 去除同一个代码:

第二步使用 `screen-widget` 并把他们拼接起来。在 `hello2/` 里面的 `webapp/` 旁边创建一个 `widget/` 目录。在 `widget/` 里面，创建一个 XML 文件定义你的页面：

HelloScreens.xml

最简单的 `screen` 定义是把几个 `ftl` 页面汇集到一起：

最后一步，是在你的控制器中关联 `screen` 的定义而不是直接使用 `Freemarker` 模板

```
services.xml news.ftl controller.xml x HelloScreens.xml

</request-map>
<request-map uri="sports">
  <response name="success" type="view" value="sports"/>
</request-map>
<request-map uri="horoscope">
  <response name="success" type="view" value="horoscope"/>
</request-map>
<request-map uri="people">
  <response name="success" type="view" value="people"/>
</request-map>
<request-map uri="guestbook">
  <response name="success" type="view" value="guestbook"/>
</request-map>
<request-map uri="hobbies">
  <response name="success" type="view" value="hobbies"/>
</request-map>

<!-- these requests correspond to POSTs of forms and call services to do their work.
      OFFBiz automatically parses form fields to service inputs.
      User can be re-directed to other requests or views after the service is called.
      The re-direct can depend on if the service succeeded or failed. -->
<request-map uri="createPerson">
  <event type="service" invoke="createHelloPerson"/>
  <response name="success" type="view" value="guestbook"/>
  <response name="error" type="view" value="guestbook"/>
</request-map>
<request-map uri="createPersonHobby">
  <event type="service" invoke="createHelloPersonHobby"/>
  <response name="success" type="view" value="hobbies"/>
  <response name="error" type="view" value="hobbies"/>
</request-map>

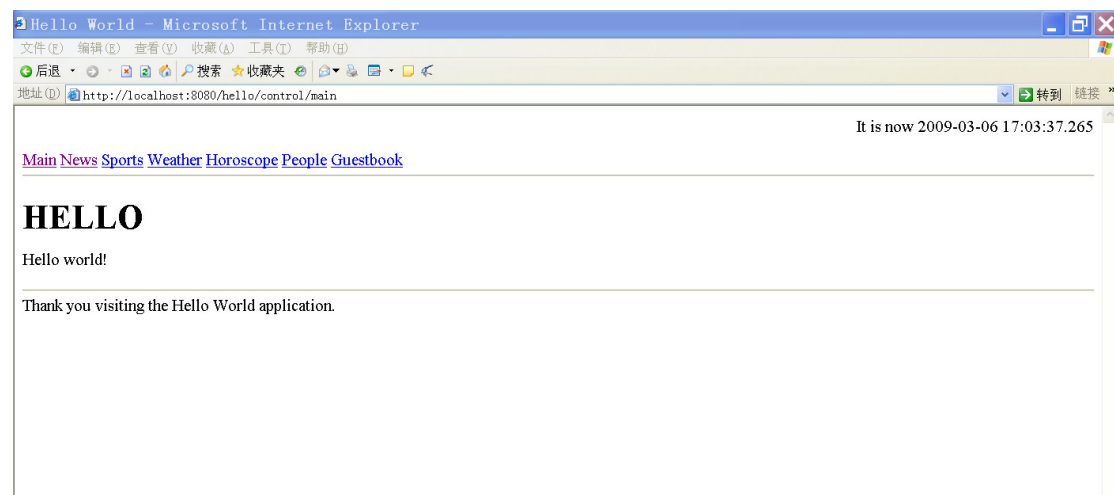
<!-- end of request mappings -->

<!-- View Mappings -->
<view-map name="error" type="jsp" page="/error/error.jsp"/>
<view-map name="main" type="screen" page="component://hello/widget/HelloScreens.xml#main"/>
<view-map name="news" type="screen" page="component://hello/widget/HelloScreens.xml#news"/>
<view-map name="weather" type="screen" page="component://hello/widget/HelloScreens.xml#weather"/>
<view-map name="sports" type="screen" page="component://hello/widget/HelloScreens.xml#sports"/>
<view-map name="horoscope" type="screen" page="component://hello/widget/HelloScreens.xml#horoscope"/>
<view-map name="people" type="screen" page="component://hello/widget/HelloScreens.xml#people"/>
<view-map name="guestbook" type="screen" page="component://hello/widget/HelloScreens.xml#guestbook"/>
<view-map name="hobbies" type="screen" page="component://hello/widget/HelloScreens.xml#hobbies"/>
<!-- end of view mappings -->
</site-conf>
```

现在我们开始，当你键入

<http://localhost:8080/hello2/control/main>

在你的浏览器中，你将得到这个



通过多个片段装饰页面，screen-widget 帮助你建立更加复杂的应用程序。

首先在 controller.xml 增加请求的 URL

```
controller.xml
<!-- Request Mappings -->
<request-map uri="main">
  <security https="false" auth="false"/>
  <response name="success" type="view" value="main"/>
</request-map>
<request-map uri="news">
  <response name="success" type="view" value="news"/>
</request-map>
<request-map uri="weather">
  <response name="success" type="view" value="weather"/>
</request-map>
<request-map uri="sports">
  <response name="success" type="view" value="sports"/>
</request-map>
<request-map uri="horoscope">
  <response name="success" type="view" value="horoscope"/>
</request-map>
<request-map uri="people">
  <response name="success" type="view" value="people"/>
</request-map>
<request-map uri="guestbook">
  <response name="success" type="view" value="guestbook"/>
</request-map>
<request-map uri="hobbies">
  <response name="success" type="view" value="hobbies"/>
</request-map>

<!-- these requests correspond to POSTs of forms and call services to do their work.
    OFBiz automatically parses form fields to service inputs.
    User can be re-directed to other requests or views after the service is called.
    The re-direct can depend on if the service succeeded or failed. -->
<request-map uri="createPerson">
  <event type="service" invoke="createHelloPerson"/>
  <response name="success" type="view" value="guestbook"/>
  <response name="error" type="view" value="guestbook"/>
</request-map>
<request-map uri="createPersonHobby">
  <event type="service" invoke="createHelloPersonHobby"/>
  <response name="success" type="view" value="hobbies"/>
  <response name="error" type="view" value="hobbies"/>
</request-map>

<!-- end of request mappings -->

<!-- View Mappings -->
<view-map name="error" type="jsp" page="/error/error.jsp"/>
<view-map name="main" type="screen" page="component://hello/widget/HelloScreens.xml#main"/>
```

其中 request-map 和 view-map 对应
再创建一个 screen 文件 HelloScreens.xml

装饰你的页面

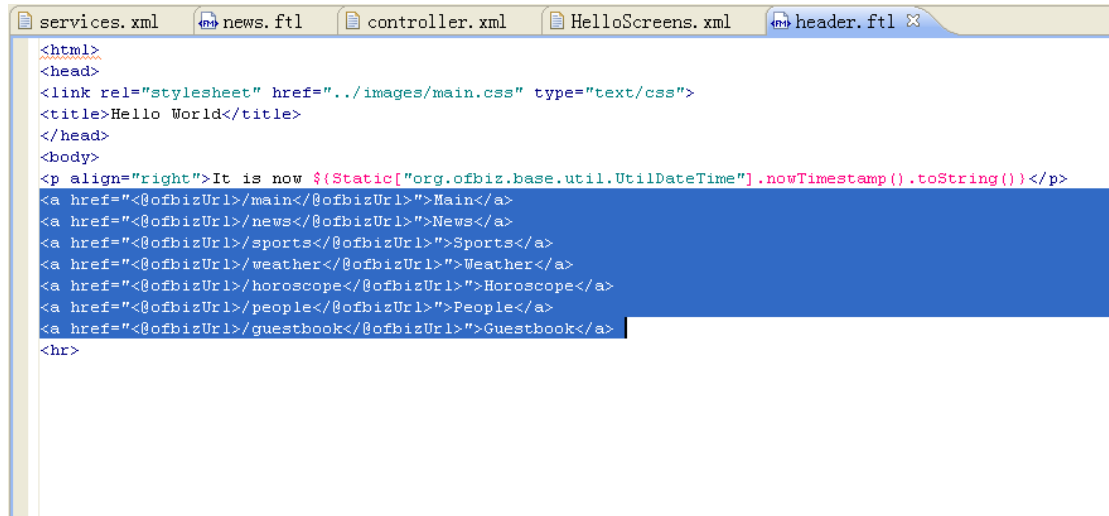
装饰器模式允许你使用许多不同的部分来组成页面,也允许你在多个页面中重复利用那些部分。例如,你能有表头、导航条、新闻区、提议和底部信息等多个片段。

例如,假设您想创建几页内容不同使用同一个 header 和 footer。screen-widget 允许你通过创建多个页面重复利用同一个显示元件。开始,我们首先在 controller.xml 创造多个请求和视图:

下一步,我们创建其他的页面和 screens。在 screen-widget XML 文件中,你能复制每个 screen 定义和修改 main.ftl 到你的页面。

在页面的式样,您不必再添加新的元素。

注意 <@ofbizUrl>标签在 header.ftl (底部。)这是用于产生您的 URL。 这些页面类似这样。



```
<html>
<head>
<link rel="stylesheet" href="../images/main.css" type="text/css">
<title>Hello World</title>
</head>
<body>
<p align="right">It is now ${Static["org.ofbiz.base.util.UtilDateTime"].nowTimestamp().toString()}</p>
<a href="<@ofbizUrl>/main</@ofbizUrl>">Main</a>
<a href="<@ofbizUrl>/news</@ofbizUrl>">News</a>
<a href="<@ofbizUrl>/sports</@ofbizUrl>">Sports</a>
<a href="<@ofbizUrl>/weather</@ofbizUrl>">Weather</a>
<a href="<@ofbizUrl>/horoscope</@ofbizUrl>">Horoscope</a>
<a href="<@ofbizUrl>/people</@ofbizUrl>">People</a>
<a href="<@ofbizUrl>/guestbook</@ofbizUrl>">Guestbook</a>
<hr>
</body>
```

当你构建大一些的站点,然而,重复布局命令是一个维护问题。幸运地, OFBiz screen-widget 允许你定义可以被多个屏幕重复利用的模板。 您通过宣称 a 做此 在模板屏幕里面(通常称 “CommonDecorator”)。 然后, 在您的其他屏幕, 您参考模板屏幕与 a 并且宣称他们自己的 内容与标记。 这是什么它看似:

引起的网页看同以前一样。 如果, 然而, 您在所有页得到了请求投入一个法律声明, 它是一样简单的象 modifying CommonDecorator 的布局。

增加 Actions

现在你可以看到使用 OFBiz 构建静态站点。 下一步, 将添加动态的内容。 一般情况下, 大多数 web 脚本允许你直接在页面内写代码。 Freemarker 也允许你这样作, 把一个日期放在 header 里。 但是 Freemarker 的作者不建议你这样作, 因为他们感到它将导致过度复杂的页面。

OFBiz 通过分离布局避免这个问题从行动或代码, 会集并且准备数据。 由网页的个别行动和介绍, 您能允许用不同的技能的分开的人(编程对设计)在网页工作。 您能也重复利用同一个代码为引起多个看法, 例如网页或同一个内容的 PDF。

要做到这点, 你首先要创建一个 actions/目录在你的 WEB-INF 目录里面。 然后你将进入 action 写 beanshell 脚本。 这里 beanshell 脚本是很象 Java servlet, 只有它被输入并且动态地被解释(而不是被编译。), 如果您想要把对象返回显示到 Freemarker 页面, 您可以将他们放入“上下文”的 Map。 最后, 你会需要展示这个 action 在你的 screen, 我这里创建了一个新的:

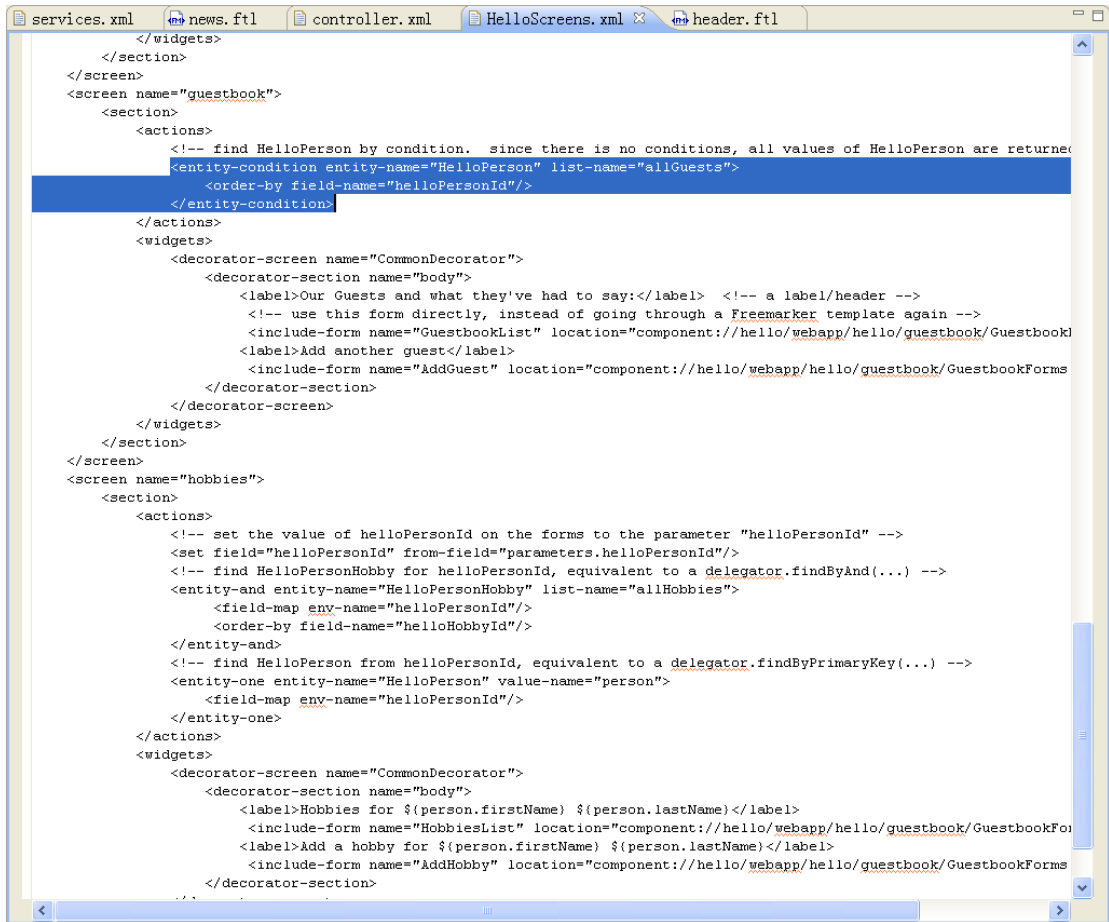
我在 controller.xml 里创建一个模型为他。

这里有很多。 在 HelloScreens.xml, 你创建一个 screen 使用 people.ftl 和 people.bsh, 这是一个数据展示的 screen。 people.bsh 是一个简单的得到 delegator 的脚本, 查找所有人员, 然后把他们放入 map。 (OFBiz 把 delegator 放入请求。)最后, 在底部, people.ftl 显示所有名字。 (这是一个 Freemarker 句法的好例子。)这些你都将看见。

```
services.xml | news.ftl | controller.xml | HelloScreens.xml x | header.ftl
<decorator-screen name="CommonDecorator">
  <decorator-section name="body">
    <platform-specific>
      <html><html-template location="component://hello/webapp/hello/sports.ftl"/></html>
    </platform-specific>
  </decorator-section>
</decorator-screen>
</widgets>
</section>
</screen>
<screen name="horoscope">
  <section>
    <widgets>
      <decorator-screen name="CommonDecorator">
        <decorator-section name="body">
          <platform-specific>
            <html><html-template location="component://hello/webapp/hello/horoscope.ftl"/></html>
          </platform-specific>
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
<screen name="people">
  <section>
    <actions>
      <script location="component://hello/webapp/hello/WEB-INF/actions/people.bsh"/>
    </actions>
    <widgets>
      <decorator-screen name="CommonDecorator">
        <decorator-section name="body">
          <platform-specific>
            <html><html-template location="component://hello/webapp/hello/people.ftl"/></html>
          </platform-specific>
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

也有一种直接在 screen-widget 使用<entity-one>方式,而不需要写一些分散的脚本和 Java 代码。 这点我就讲这些,希望你可以深入了解。

是否因为有 delegator 对象,你可以直接读写数据库并且把你所有的业务逻辑写在.bsh 脚本中? 答案是可以,但是不建议这样做,为了更大的应用。实际上,从来没有这样的 OFBiz 应用程序。

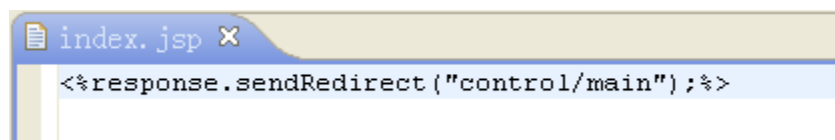


```
</widgets>
</section>
</screen>
<screen name="guestbook">
  <section>
    <actions>
      <!-- find HelloPerson by condition. since there is no conditions, all values of HelloPerson are returned -->
      <entity-condition entity-name="HelloPerson" list-name="allGuests">
        <order-by field-name="helloPersonId"/>
      </entity-condition>
    </actions>
    <widgets>
      <decorator-screen name="CommonDecorator">
        <decorator-section name="body">
          <label>Our Guests and what they've had to say:</label> <!-- a label/header -->
          <!-- use this form directly, instead of going through a Freemarker template again -->
          <include-form name="GuestbookList" location="component://hello/webapp/hello/guestbook/GuestbookList.ftl"/>
          <label>Add another guest</label>
          <include-form name="AddGuest" location="component://hello/webapp/hello/guestbook/GuestbookForms.ftl"/>
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
<screen name="hobbies">
  <section>
    <actions>
      <!-- set the value of helloPersonId on the forms to the parameter "helloPersonId" -->
      <set field="helloPersonId" from-field="parameters.helloPersonId"/>
      <!-- find HelloPersonHobby for helloPersonId, equivalent to a delegator.findByAnd(...) -->
      <entity-and entity-name="HelloPersonHobby" list-name="allHobbies">
        <field-map env-name="helloPersonId"/>
        <order-by field-name="helloHobbyId"/>
      </entity-and>
      <!-- find HelloPerson from helloPersonId, equivalent to a delegator.findByPrimaryKey(...) -->
      <entity-one entity-name="HelloPerson" value-name="person">
        <field-map env-name="helloPersonId"/>
      </entity-one>
    </actions>
    <widgets>
      <decorator-screen name="CommonDecorator">
        <decorator-section name="body">
          <label>Hobbies for ${person.firstName} ${person.lastName}</label>
          <include-form name="HobbiesList" location="component://hello/webapp/hello/guestbook/GuestbookForms.ftl"/>
          <label>Add a hobby for ${person.firstName} ${person.lastName}</label>
          <include-form name="AddHobby" location="component://hello/webapp/hello/guestbook/GuestbookForms.ftl"/>
        </decorator-section>
      </decorator-screen>
    </widgets>
  </section>
</screen>
```

完整的应用分析

这部分建议结合例子代码来理顺一下各种情况下的代码结构。

主页



```
<%response.sendRedirect("control/main");%>
```

当以 <http://localhost:8080/hello> 访问的时间，会自动访问 index.jsp，这时自动跳转到 <http://localhost:8080/hello/control/main>，这个 main 的请求就在 controller.xml 中配置在这个文件中可以看到

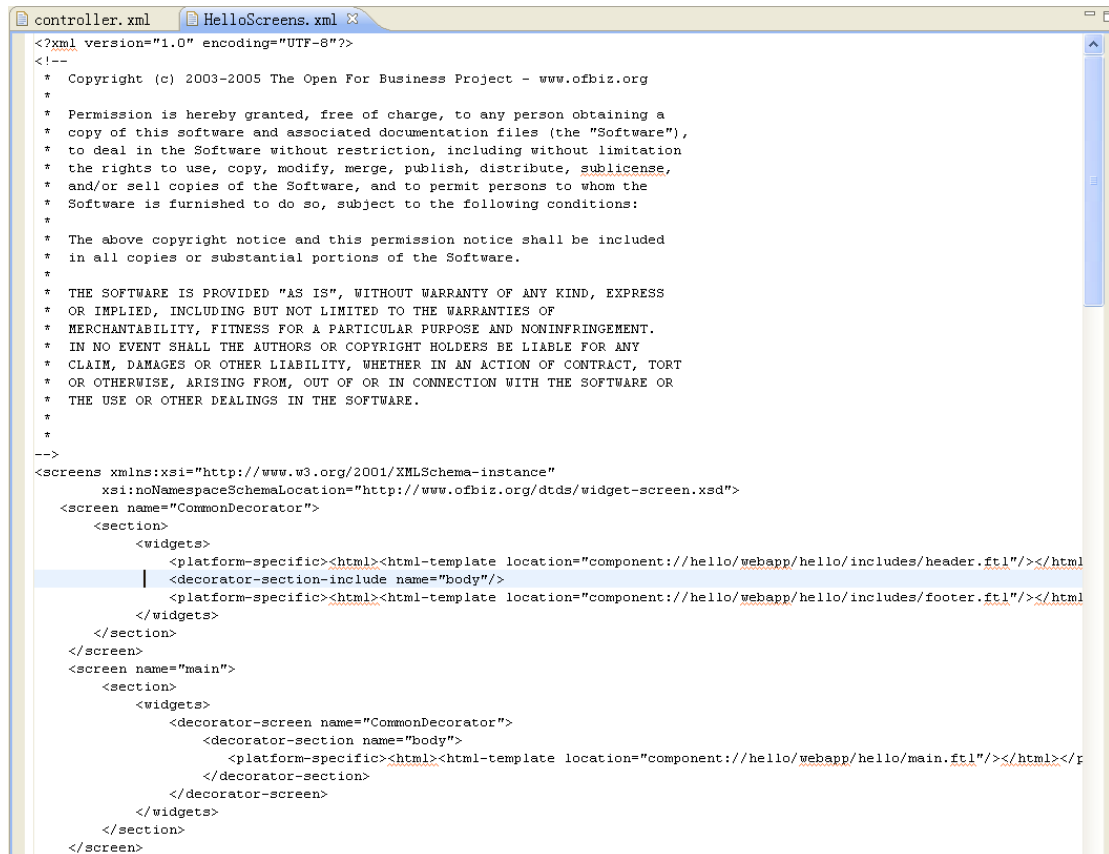
```
<request-map uri="main">
  <security https="false" auth="false"/>
  <response name="success" type="view" value="main"/>
</request-map>
```



```
</request-map>
```

```
<view-map name="main" type="screen"
page="component://hello/widget/HelloScreens.xml#main"/>
```

其中的 view-map 指向 HelloScreens.xml 中的 main screen 这部分。



根据上面的配置，增加几个文件，
includes/header.ftl ， includes/footer.ftl， main.ftl

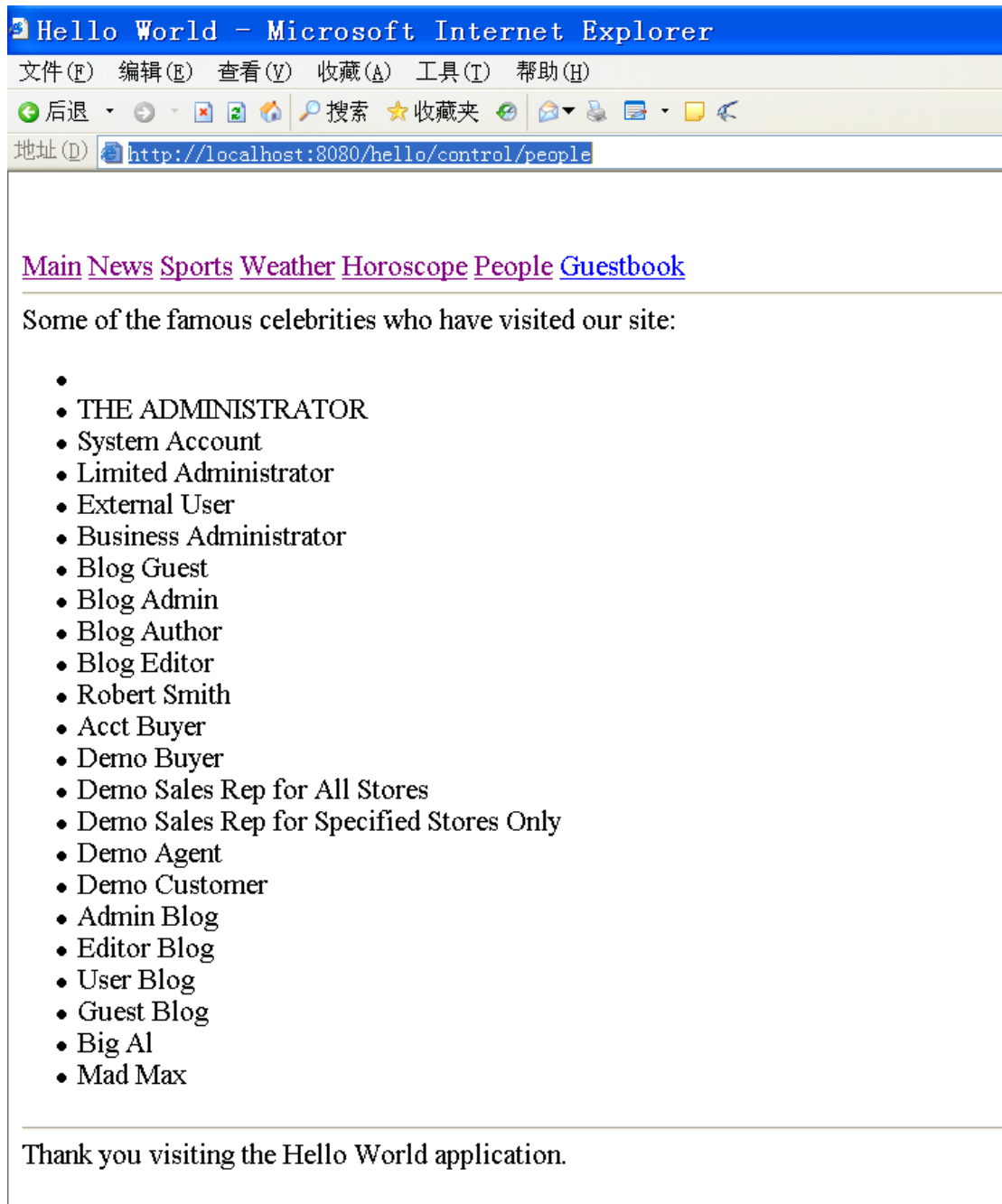
News 功能

这个功能超简单，只是链接一个静态页面，是在整体风格下

查看 people 功能

<http://localhost:8080/hello/control/people>

访问结果如下



在 controller.xml 中配置 request-map 名 people 和 view-map 名 people

下面是对应的 screen

```
index.jsp controller.xml HelloScreens.xml
<html><html-template location="component://hello/webapp/hello/sports.ftl"/></html>
</platform-specific>
</decorator-section>
</decorator-screen>
</widgets>
</section>
</screen>
<screen name="horoscope">
<section>
<widgets>
<decorator-screen name="CommonDecorator">
<decorator-section name="body">
<platform-specific>
<html><html-template location="component://hello/webapp/hello/horoscope.ftl"/></html>
</platform-specific>
</decorator-section>
</decorator-screen>
</widgets>
</section>
</screen>
<screen name="people">
<section>
<actions>
<script location="component://hello/webapp/hello/WEB-INF/actions/people.bsh"/>
</actions>
<widgets>
<decorator-screen name="CommonDecorator">
<decorator-section name="body">
<platform-specific>
<html><html-template location="component://hello/webapp/hello/people.ftl"/></html>
</platform-specific>
</decorator-section>
</decorator-screen>
</widgets>
</section>
</screen>
```

Screen 中

people.bsh 是取数据

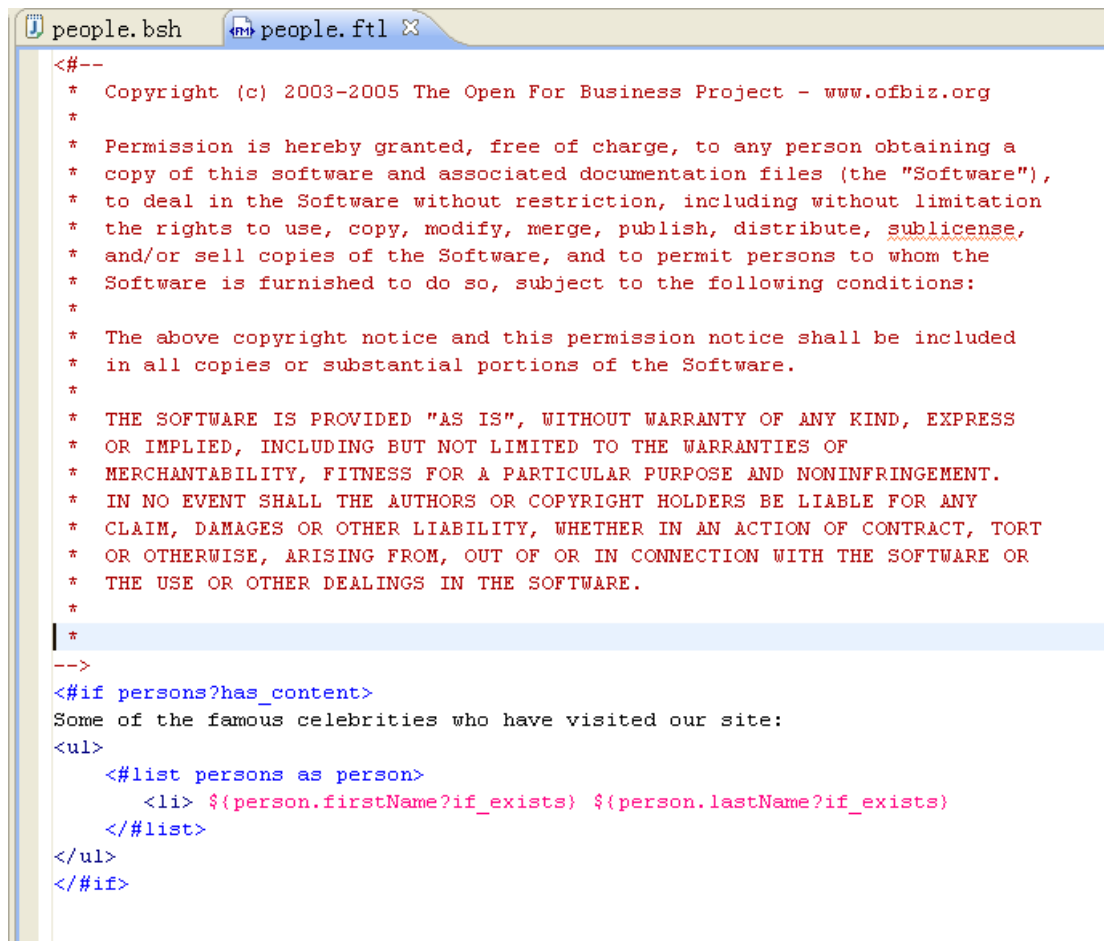
```
people.bsh
/*
 * Copyright (c) 2003-2005 The Open For Business Project - www.ofbiz.org
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
 * OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
 * THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
 */
import org.ofbiz.entity.*;

delegator = request.getAttribute("delegator");

persons = delegator.findAll("Person");

context.put("persons", persons);
```

people.ftl 是数据展示



```
<!--
 * Copyright (c) 2003-2005 The Open For Business Project - www.ofbiz.org
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
 * OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
 * THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 *
-->
<#if persons?has_content>
Some of the famous celebrities who have visited our site:
<ul>
  <#list persons as person>
    <li> ${person.firstName?if_exists} ${person.lastName?if_exists}
  </#list>
</ul>
</#if>
```

guestbook

这里面有个 form 对象: GuestbookList, AddGuest

```
controller.xml HelloScreens.xml GuestbookForms.xml
* the rights to use, copy, modify, merge, publish, distribute, sublicense,
* and/or sell copies of the Software, and to permit persons to whom the
* Software is furnished to do so, subject to the following conditions:
*
* The above copyright notice and this permission notice shall be included
* in all copies or substantial portions of the Software.
*
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
* OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
* MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
* IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
* CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
* OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
* THE USE OR OTHER DEALINGS IN THE SOFTWARE.
*
* @author Si Chen (sichen@opensourcestrategies.com)
-->

<forms xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/widget-form.xsd">
  <form name="GuestbookList" type="list" list-name="allGuests"> <!-- this form just lists all the values of allGuests -->
    <auto-fields-entity entity-name="HelloPerson" default-field-type="display"/> <!-- automatically display all fields from
    <field name="hobbies"> <!-- also create a link to see the hobbies -->
      <hyperlink target="hobbies?helloPersonId=${helloPersonId}" description="See hobbies"/>
    </field>
  </form>
  <form name="AddGuest" type="single" target="createPerson"> <!-- this form posts to "createPerson" (see controller.xml) -->
    <auto-fields-entity entity-name="HelloPerson"/> <!-- all fields of HelloPerson automatically become fields for form r
    <field name="helloPersonId"><hidden/></field> <!-- this becomes a hidden field, so we don't see it on form -->
    <field name="submitButton" title="Add a Guest" widget-style="standardSubmit"><submit button-type="button"/></field>
  </form>
  <form name="HobbiesList" type="list" list-name="allHobbies">
    <field name="helloHobbyId" title="Hobby"> <!-- the form will do a lookup on HelloHobby using helloHobbyId and display it
    <display-entity entity-name="HelloHobby" description="${description}"/>
    </field>
  </form>
</forms>
```

GuestbookList 这个 form 就是取数据列表显示下

AddGuest 这个 form 是提交数据到 **createPerson**

```
controller.xml HelloScreens.xml GuestbookForms.xml
</request-map>
<request-map uri="sports">
  <response name="success" type="view" value="sports"/>
</request-map>
<request-map uri="horoscope">
  <response name="success" type="view" value="horoscope"/>
</request-map>
<request-map uri="people">
  <response name="success" type="view" value="people"/>
</request-map>
<request-map uri="guestbook">
  <response name="success" type="view" value="guestbook"/>
</request-map>
<request-map uri="hobbies">
  <response name="success" type="view" value="hobbies"/>
</request-map>

<!-- these requests correspond to POSTs of forms and call services to do their work.
OFBiz automatically parses form fields to service inputs.
User can be re-directed to other requests or views after the service is called.
The re-direct can depend on if the service succeeded or failed. -->
<request-map uri="createPerson">
  <event type="service" invoke="createHelloPerson"/>
  <response name="success" type="view" value="guestbook"/>
  <response name="error" type="view" value="guestbook"/>
</request-map>
<request-map uri="createPersonHobby">
  <event type="service" invoke="createHelloPersonHobby"/>
  <response name="success" type="view" value="hobbies"/>
  <response name="error" type="view" value="hobbies"/>
</request-map>

<!-- end of request mappings -->

<!-- View Mappings -->
<view-map name="error" type="jsp" page="/error/error.jsp"/>
<view-map name="main" type="screen" page="component://hello/widget/HelloScreens.xml#main"/>
<view-map name="news" type="screen" page="component://hello/widget/HelloScreens.xml#news"/>
<view-map name="weather" type="screen" page="component://hello/widget/HelloScreens.xml#weather"/>
<view-map name="sports" type="screen" page="component://hello/widget/HelloScreens.xml#sports"/>
<view-map name="horoscope" type="screen" page="component://hello/widget/HelloScreens.xml#horoscope"/>
<view-map name="people" type="screen" page="component://hello/widget/HelloScreens.xml#people"/>
<view-map name="guestbook" type="screen" page="component://hello/widget/HelloScreens.xml#guestbook"/>
<view-map name="hobbies" type="screen" page="component://hello/widget/HelloScreens.xml#hobbies"/>
<!-- end of view mappings -->
</site-conf>
```

<request-map uri="createPerson">

```

        <event type="service" invoke="createHelloPerson"/>
        <response name="success" type="view" value="guestbook"/>
        <response name="error" type="view" value="guestbook"/>
    </request-map>

```

这个提交到请求 createPerson 时，先调用服务 createHelloPerson 服务的定义

```

<?xml version="1.0" encoding="UTF-8"?>
<!--
/**
 * Copyright (c) 2001-2005 The Open For Business Project - www.ofbiz.org
 *
 * Permission is hereby granted, free of charge, to any person obtaining a
 * copy of this software and associated documentation files (the "Software"),
 * to deal in the Software without restriction, including without limitation
 * the rights to use, copy, modify, merge, publish, distribute, sublicense,
 * and/or sell copies of the Software, and to permit persons to whom the
 * Software is furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included
 * in all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
 * OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
 * MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
 * IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
 * CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT
 * OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR
 * THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 */
-->

<services xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="http://www.ofbiz.org/dtds/services.xsd">
    <description>Hello Services</description>

    <!-- this will be implemented in Java -->
    <service name="createHelloPerson" engine="java">
        <location>org.ofbiz.hello.HelloServices</location>
        <description>Create a HelloPerson</description>
        <auto-attributes mode="IN" entity-name="HelloPerson" include="nonpk" optional="true"/>
        <attribute name="helloPersonId" mode="OUT" type="String" optional="false"/>
    </service>

    <!-- this will be implemented in OFBiz minilang -->
    <service name="createHelloPersonHobby" engine="simple">
        <location>org.ofbiz.hello.HelloServices.xml</location>
        <description>Create a HelloPersonHobby which links a person and a hobby</description>
        <auto-attributes mode="IN" entity-name="HelloPersonHobby" include="pk" optional="false"/>
    </service>
</services>

```

实行服务的类

```
controller.xml HelloScreens.xml GuestbookForms.xml services.xml HelloServices.java
* Copyright (c) 2001-2005 The Open For Business Project - www.ofbiz.org

package org.ofbiz.hello;

import java.util.HashMap;

public class HelloServices {

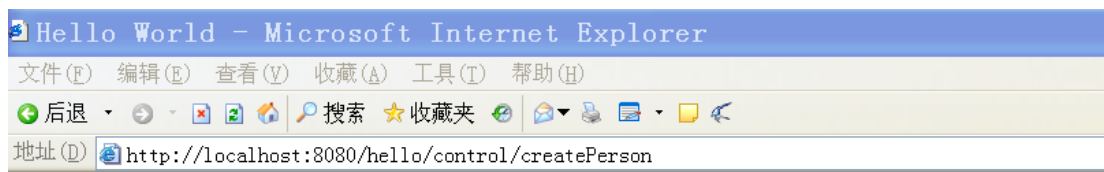
    public static final String module = HelloServices.class.getName(); // used for debugging

    public static Map createHelloPerson(DispatchContext dctx, Map context) {
        GenericDelegator delegator = dctx.getDelegator(); // always passed in with DispatchContext

        try {
            String helloPersonId = delegator.getNextSeqId("HelloPerson"); // gets next available key for HelloPerson
            Debug.logInfo("helloPersonId = " + helloPersonId, module); // prints to the console or console.log
            GenericValue helloPerson = delegator.makeValue("HelloPerson",
                UtilMisc.toMap("helloPersonId", helloPersonId)); // create a GenericValue from ID we just got
            helloPerson.setNonPKFields(context); // move non-primary key fields from input parameters to GenericValue
            delegator.create(helloPerson); // store the generic value, ie persists it

            Map result = ServiceUtil.returnSuccess(); // gets standard Map for successful service operations
            result.put("helloPersonId", helloPersonId); // puts output parameter into Map to return
            return result; // return Map
        } catch (GenericEntityException ex) { // required if you use delegator in Java
            return ServiceUtil.returnError(ex.getMessage());
        }
    }
}
```

按图所示添加一条记录



[Main](#) [News](#) [Sports](#) [Weather](#) [Horoscope](#) [People](#) [Guestbook](#)

Our Guests and what they've had to say:

Hello Person IdFirst NameMiddle NameLast NameCommentsHobbies

10000 first middle last test [See hobbies](#)

Add another guest

First Name	<input type="text" value="first"/>
Middle Name	<input type="text" value="middle"/>
Last Name	<input type="text" value="last"/>
Comments	<input type="text" value="test"/>
<input type="button" value="Add a Guest"/>	

Thank you visiting the Hello World application.

这里先得通过 webtools 把 data 倒入

OFBiz: Web Tools: XML Data Import - Microsoft Internet Explorer

文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)

后退 地址 搜索 收藏夹

地址 https://localhost:8443/webtools/control/EntityImport

Accounting Catalog Content Example Facility Manufacturing Marketing Order Party WebTools WorkEffort

Framework Web Tools

Main Stats Jobs Cache

XML Import to DataSource(s)

This page can be used to import exported Entity Engine XML documents. These documents all have a root tag of "<entity-engine-xml>".

Import:

Absolute Filename of FreeMarker template file to filter data by (optional):

Absolute Filename or URL:

☐ Is URL?

☐ Mostly Inserts?

☐ Maintain Timestamps?

☐ Create "Dummy" FKs

TX Timeout Seconds (for each entity or file):7200

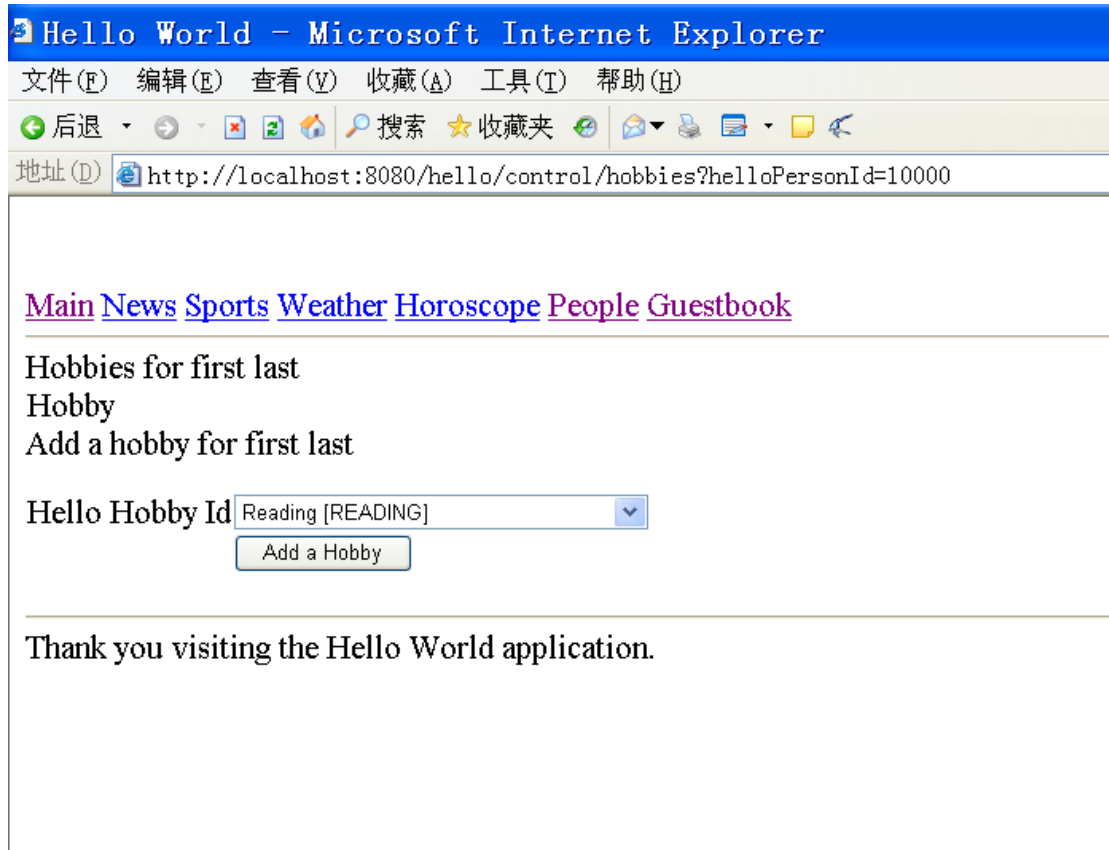
Import File

Complete XML document (root tag: entity-engine-xml):

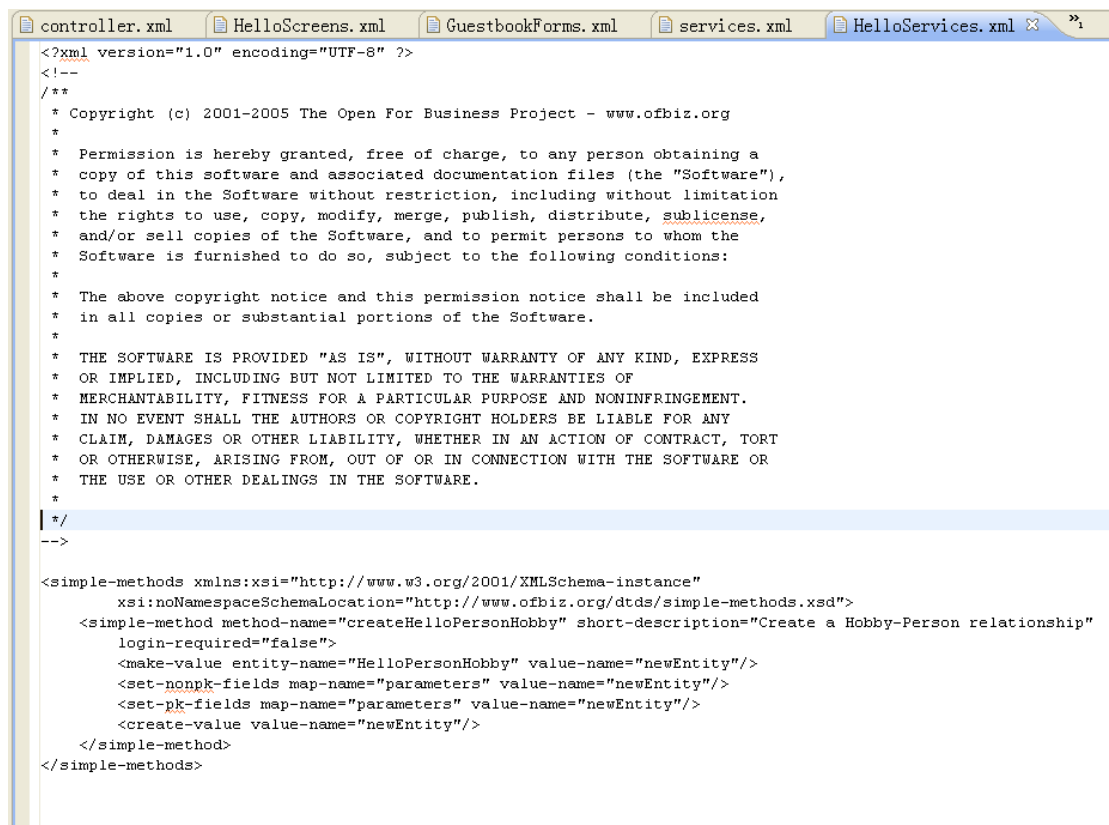
<entity-engine-xml>
<HelloHobby helloHobbyId="READING" description="Reading"/>
<HelloHobby helloHobbyId="MOVIES" description="Movies"/>
<HelloHobby helloHobbyId="THEATER" description="The theater"/>
<HelloHobby helloHobbyId="OPERA" description="The opera"/>
<HelloHobby helloHobbyId="SKIING" description="Skiing"/>
<HelloHobby helloHobbyId="SURFING" description="Surfing"/>
<HelloHobby helloHobbyId="WINDSURFING" description="Windsurfing"/>
<HelloHobby helloHobbyId="BASKETBALL" description="Basketball"/>
<HelloHobby helloHobbyId="FOOTBALL_US" description="American Football"/>
<HelloHobby helloHobbyId="FOOTBALL_OTHER" description="Football (Soccer)"/>
<HelloHobby helloHobbyId="COOKING" description="Cooking"/>
<!-- But data for fields can also be tags inside the entity name tags, so you can use
CDATA
for longer fields -->
<HelloHobby helloHobbyId="WINE">
<description>Wine</description>
</HelloHobby>
</entity-engine-xml>

Import Text

这时可以点击 See hobbies 链接



这里用到的是一个 minilang 写的服务



附录

在 event 中能访问的对象

As you will recall, the request is passed into the Java event as a parameter. The session can be obtained from the request in the standard way.

```
HttpSession session = request.getSession();
```

The availableLocales object can be constructed like this:

```
List UtlavailableLocales = UtilMisc.availableLocales();
```

The locale object can be retrieved like this:

```
Locale locale = UtilHttp.getLocale(request);
```

The delegator object can be retrieved like this:

```
GenericDelegator delegator =  
    (GenericDelegator)request.getAttribute("delegator");
```

The dispatcher object can be retrieved like this:

```
LocalDispatcher dispatcher =  
    (LocalDispatcher)request.getAttribute("dispatcher");
```

The security object can be retrieved like this:

```
Security security = (Security)request.getAttribute("security");
```

The userLogin object can be retrieved from the session:

```
GenericValue userLogin =  
    (GenericValue)session.getAttribute("userLogin");
```

FTL 中常用说明

ftl 几个常用对象

```
sessionAttributes、requestAttributes
```

ftl 中类的调用方法

```
<#assign nowTimestamp=Static["org.ofbiz.base.util.UtilDateTime"].nowTimestamp()>
```

遍历 list 时，得到某个对象在 list 中的序号，是从 0 开始计数

```
<#list productList as product>  
    ${product_index}  
</#list>
```

格式化日期

```
${orderHeader.orderDate?string("yyyy-MM-dd")}
```

FreeMarker 对 substring 默认支持

方法很简单, `${string[0..4]}`
就是截取 0-4 个字符。

OFBiz form

ofbiz form 中下拉列表的代码

```
<field name="agreementTypeId"
  title="${uiLabelMap.AccountingAgreementTypeId}">
  <drop-down allow-empty="true">
    <entity-options
```

```
      description="${description}"

      entity-name="AgreementType"

      key-field-name="agreementTypeId"/>
    </drop-down>
  </field>
```

以上是在 form 中显示下拉列表的代码示例, title 是下拉列表前的说明文字, entity-name 是下拉列表表项的取值实体, description 是下拉列表显示的表项, 此处, 下拉列表的表项从实体 AgreementType 中的 description 域取值. 另外, 标签中的 allow-empty 如果为 true 则允许该下拉菜单为空, 如果为 false 则必须在下拉列表中选择其一.

OFBIZ FORM 表头汉化示例

```
<form name="ContactList" type="list" list-name="allContacts">
```

```
  <auto-fields-entity entity-name="Contact"
  default-field-type="display"/>

  <field name="contactId" title="联系人 ID"></field>
  <field name="name" title="姓名"></field>
  <field name="duty" title="职位"></field>
  <field name="responsibility" title="职责"></field>
  <field name="corporation" title="单位"></field>
  <field name="email" title="E-mail"></field>
  <field name="tel" title="电话"></field>
```

```
<field name="msn" title="MSN"></field>
<field name="qq" title="QQ"></field>
</form>
```

首先, `<auto-fields-entity entity-name="Contact" default-field-type="display"/>` 先将实体 Contact 的所有域取出来, 如果下面不对各域作具体指定则直接根据 display 的格式显示各域. 其次, 下面的每一个条 `<field name="contactId" title="联系人 ID"></field>` 语句都将对应域的表头进行汉化.

ofbiz form 中不显示实体某域的代码

在用 `<auto-fields-entity entity-name="Contact" default-field-type="display"/>` 读出实体的所有域后用 `<field name="...." title="...."><hidden/></field>` 指定具体的隐藏域即可.

ofbiz 查找功能关键代码

其中 FindTest 表单是用于输入查询条件的表单, ResultTest 表单是用于显示查询结果的表单. 两张表单在同一页面上显示. 其中, Test 是实体名.

```
<form name="FindTest" target="main" type="single">
  <auto-fields-entity entity-name="Test"
default-field-type="find"/>
  <field name="submitButton" title="查找"
widget-style="smallSubmit">
    <submit button-type="button"/>
  </field>
</form>

<form name="ResultTest" list-iterator-name="listIt" target=""
paginate-target="main" title="" type="list">
  <actions>
    <set field="entityName" value="Test"/>
    <service service-name="performFind" result-map-name="result"
result-map-list-iterator-name="listIt">
      <field-map field-name="inputFields"
env-name="requestParameters"/>
      <field-map field-name="entityName" env-name="entityName"/>
    </service>
  </actions>
```

```
<auto-fields-entity entity-name="Test"
default-field-type="display"/>
</form>
```

小结:

查询功能不需要 minilanguage 或 java 来实现. 输入查询条件的表单 type 为 single, target 指向的是当前页面, auto-fields-entity 元素的 type 为 find.

显示查询结果的表单比较特别, 该表单中有<action>部分, 其中的代码就是实现查询功能的代码, 具体使用时修改实体名即可. 和其它表单一样, 可以指定具体域有特殊的显示效果或隐藏.

如何基于 ofbiz 在页面中显示一张数据库表（利昂原创）

（备注：要看懂该文章必须具备 OFBIZ 的基础知识。）

关键代码 1: 在 widget 的 screen 中, action 部分用<entity-condition 标签指定实体名和实体列表名;

关键代码 2: 在 widget 的 screen 中, widgets 部分用<include-form 标签指定被引用表单的位置和表单名;

```
<screen name="guestbook">
  <section>
    <actions>
<!-- find HelloPerson by condition.  since there is no conditions, all
values of HelloPerson are returned in allGuests -->
      <entity-condition entity-name="HelloPerson"
list-name="allGuests"><order-by
field-name="helloPersonId"/></entity-condition>
      <!--窗口中调用的表单用到的实体数据在该处指出!! 如果该句设置得不对表单中将
不显示数据!!! -->
    </actions>
    <widgets>
      <decorator-screen name="CommonDecorator">
        <decorator-section name="body">
          <label>我们的客人和他们说的话</label> <!-- a
```

```

label/header -->
        <!-- use this form directly, instead of going
through a Freemarker template again -->
        <include-form name="GuestbookList"
location="component://hello3/webapp/hello3/guestbook/GuestbookFor
ms.xml"/>

        </decorator-section>
    </decorator-screen>
</widgets>
</section>
</screen>

```

关键代码 3：在表单文件内的对应（上面的<include-form 标签指出的**表单名**）的表单代码处，<form 标签的 type 要设成 list，list-name 要和 widget 中窗口中的 action 部分的**列表名**一致。

```

<form name="GuestbookList" type="list" list-name="allGuests">

<!-- 该处的列表名和 widget 的窗口 action 中指定的列表名要对应! this form just
lists all the values of allGuests -->
    <auto-fields-entity entity-name="HelloPerson"
default-field-type="display"/>

<!-- 自动显示指定实体的所有域 automatically display all fields from
HelloPerson -->
</form>

```

总体思路就是：form 根据实体定义创建表头，action 里的 entity condition 根据条件从数据库中取出数据显示于表中，这就完成了在页面上显示数据库表的过程。

参考代码

这部分主要方便大家找到各类型的开发模式和功能的参考代码，可能通过模仿来快速提高开发效率。

分页查询

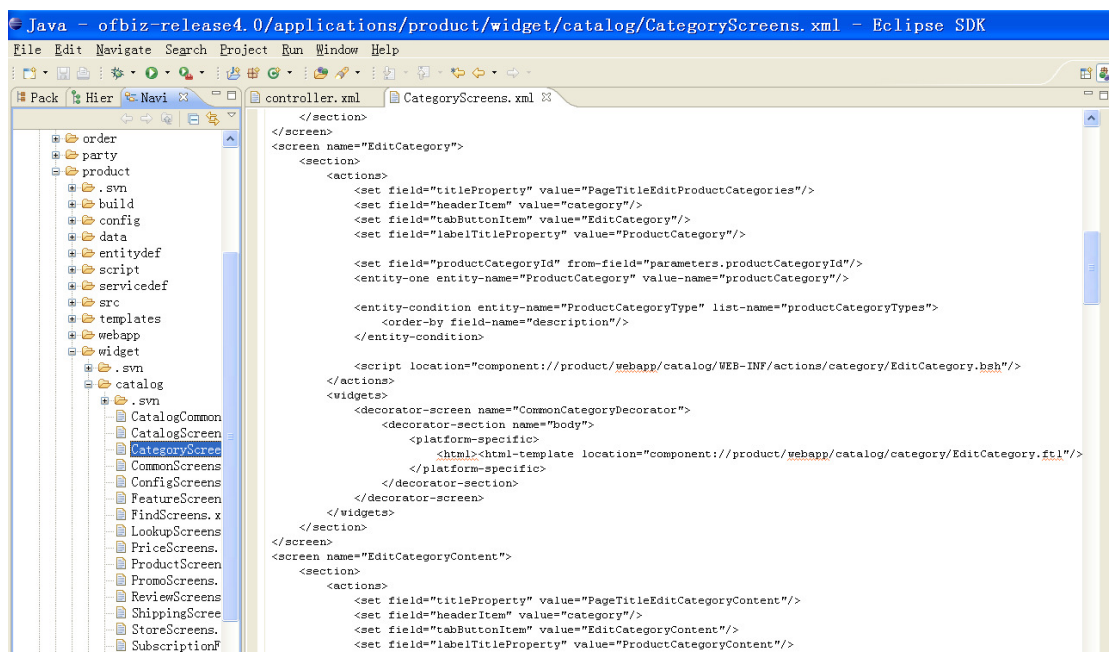
Party 模块中的 findparty

调用的是 findparty 服务，

主要代码在类 `org.ofbiz.party.party.PartyServices` 中

新增修改记录

Catalog 中的编辑产品分类



EditCategory.ftl 页面中根据传来的数据设置所提交到的请求 createProductCategory 或者 updateProductCategory

```
<request-map uri="createProductCategory">
    <security https="true" auth="true"/>
    <event type="service" path="" invoke="createProductCategory"/>
    <response name="success" type="view" value="EditCategory"/>
    <response name="error" type="view" value="EditCategory"/>
</request-map>

<request-map uri="updateProductCategory">
    <security https="true" auth="true"/>
    <event type="service" path="" invoke="updateProductCategory"/>
    <response name="success" type="view" value="EditCategory"/>
    <response name="error" type="view" value="EditCategory"/>
</request-map>
```

分别调用服务 createProductCategory 或者 updateProductCategory

可以在 product/sevicedef/services.xml 中找到相关的服务定义

form 用法参考

