开发企业联系方式页面即CompanyInfo页面

需求：加载页面后，从sl\_company表中加载单条数据到autoform中。

sl\_company表脚本如下：

CREATE TABLE `sl\_company` (

`company\_id` int(11) NOT NULL AUTO\_INCREMENT,--企业Id

`company\_name` varchar(50) DEFAULT NULL,--企业简称

`full\_name` varchar(100) DEFAULT NULL,--企业全称

`web` varchar(45) DEFAULT NULL,--企业官网

`address` varchar(100) DEFAULT NULL,--企业地址

`postcode` varchar(45) DEFAULT NULL,--邮政编码

`phone` varchar(45) DEFAULT NULL,--热线电话

`email` varchar(45) DEFAULT NULL,--电子邮件

PRIMARY KEY (`company\_id`)

) ENGINE=InnoDB AUTO\_INCREMENT=16 DEFAULT CHARSET=utf8;

效果如下：



## 1、准备工作

### 1.1新建Package包：

com.batek.dorado.sample.standardlesson.entity用来放置实体映射类

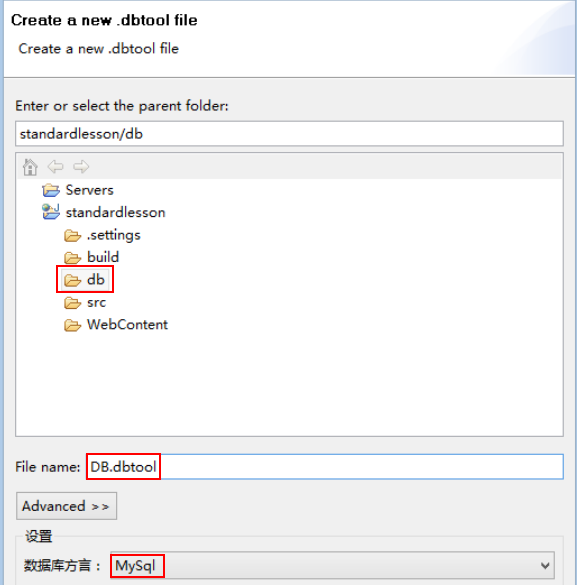
com.batek.dorado.sample.standardlesson.dao用来放置和数据库交互的DAO类

### 1.2由于要和数据库进行交互需要依赖hibernate等jar包。引入jar包

### 1.3生成实体映射类：

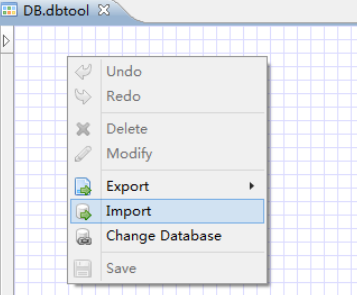
项目中采用hibernate3作为数据持久层，因此需要生成实体映射类。安装Hibernate Tools插件从数据库反向生成。Dorado7IDE中提供了dbtool数据库建模工具帮助我们快速从数据库中反向生成实体映射类。

在项目根目录下新建一个db文件夹，选中文件夹，ctrl+n使用新建文件向导，展开[BDF]文件夹，选中[创建数据库模型]，如下：

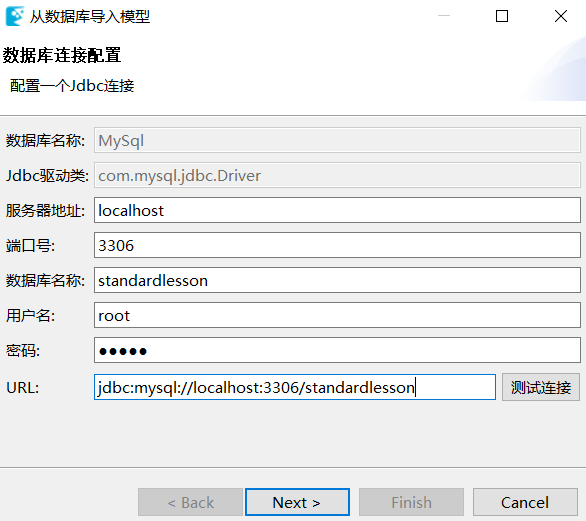


修改[File name]为DB.dbtool，数据库方言采用默认配置，点击[Finish]按钮，完成数据库模型文件创建。

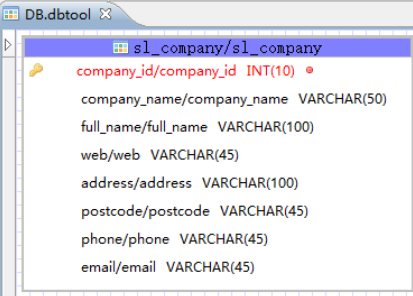
接下来需要导入数据库中表定义信息。在新建的DB.dbtool文件上右键->import：



提示需要引入mysql驱动，选择lib目录下的mysql驱动，然后从数据库导入驱动：

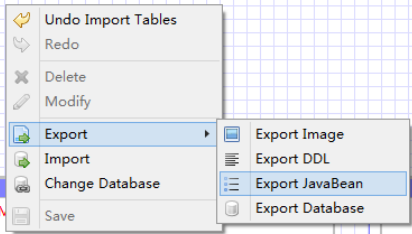


输入配置的数据库名称，用户名root，密码mysql，选中需要引入到模型文件中的表：

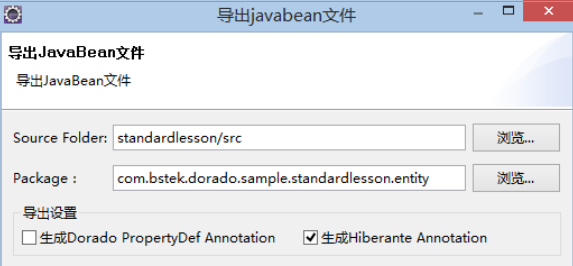


到目前为止，我们从数据库导入了表信息，接下来我们只需要将表信息导出为实体映射类即可。

在DB.dbtool文件右击，选择[Export]->[Export JavaBean]。



将[Source Folder]和[Package]配置选中[生成Hibernate Annotation]：



dbtool数据库建模工具不仅可以用作hibernate实体映射类生成工具，也是一个优秀的数据库建模工具。

### 1.4新建Dao文件

针对单表开发，那么就需要和数据交互的Dao，在com.bstek.dorado.sample.standardlesson.dao包下新建和数据库交互的Dao类，以SICompany实体为例，创建SICompanyDao并制定它的父类为com.bstek.darado.hibernate.HibernateDao。

@Repository

public class SICompany extends HibernateDao<SICompany, Long> {

}

注：HibernateDao是一个泛型父类，所以必须在子类指定具体的类型，修改类型参数T为实际的类，也就是当前子类需要维护的实体类对象，同时修改PK为实体对象中关联数据库表的主键的属性类型。

为了开发简便，工程中采用Dorado7提供的Hibernate Add On扩展，可以通过继承HibernateDao对象来实现自己的业务Dao对象，HibernateDao已经实现了增删改查的基本方法，所以Dao类暂时不需要编写具体内容。，依此创建所有实体类的Dao类。

### 1.5数据源连接配置

修改dorado-home中的configure.properties文件。在其中添加数据库的配置信息：

#数据库配置信息

jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql://localhost:3306/stdlesson?useUnicode=true&amp;characterEncoding=utf8

jdbc.userName=root

jdbc.password=mysql

jdbc.minldle=10

jdbc.maxActive=50

hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

hibernate.showSql=true

hibernate.formatSql=true

### 1.6修改app-context.xml

将数据库配置信息加入app-context.xml中：



* 配置dbcp数据源
* 整合hibernate中sessionFactory 交给Spring容器管理
* 配置事务管理器
* 配置注解方式管理驱动

### 1.7修改web.xml文件

<filter>

<filter-name>OpenSeesionInViewFilter</filter-name>

<filter-class>org.springframework.orm.hibernate3.support.OpenSessionInViewFilte</filter-class>

<init-param>

<param-name>excludeSuffixs</param-name>

<param-value>js,css,jpg,gif,png</param-value>

</init-param>

</filter>

<filter-mapping>

<filter-name>OpenSessionInViewFilter</filter-name>

<url-pattern>\*.d</url-pattern>

</filter-mapping>

<filter-mapping>

<filter-name>OpenSessionInViewFilter</filter-name>

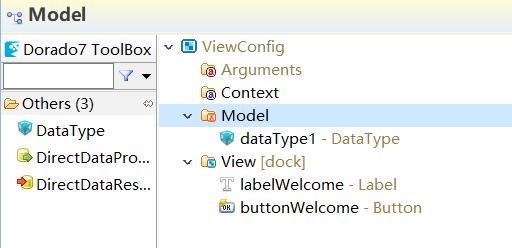
<url-pattern>/dorado/\*</url-pattern>

</filter-mapping>

### 1.8准备Model

Dorado7本身是采用基于数据模型驱动的开发模式。数据模型中DataType的基本概念和基本使用技巧（DataType就是对数据格式的定义，可类比数据库模式定义语言DDL）。

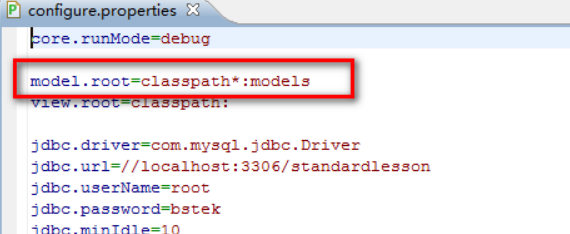
Model文件说明：在View节点中包含一个Model节点，可以在Model节点下添加DataType，鼠标选中Model节点，在左侧Dorado7ToolBox中双击[Others]文件夹下的DataType，即可新建DataType：



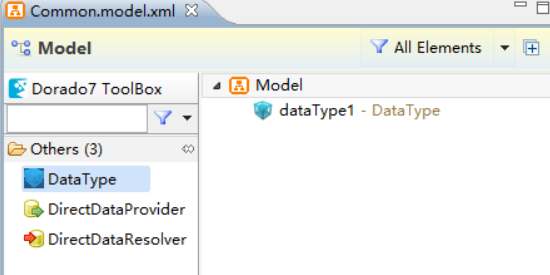
如果将一个DataType定义在View的Model节点下，通常说这个DataType是当前View私有的，如果另一个View也想使用相同的DataType就需要在哪个View中再次定义这个DataType。

为了能达到DataType重用的目的，或者说全局的DataType，Dorado7专门提供了Model配置文件，通过定义一个专门的Model文件，在其中添加若干个DataType给整个工程共享使用：

Dorado7工程通过系统配置，预先读取Model文件，并初始化DataType信息。相关的配置文件有：dorado-home文件夹下的configure.properties：



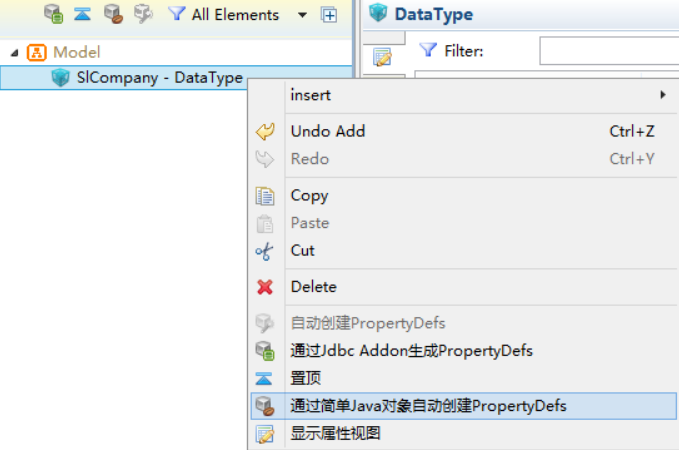
src下创建models目录（确保与配置文件中model.root的路径保持一致），新建Common.model.xml文件。在主工作区切换到Model视图。选中Model节点，新建DataType：



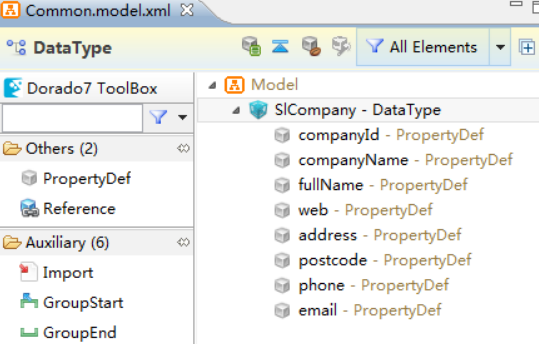
修改DataType属性：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 描述 |
| name | SlCompany | DataTypede的名称 |
| matchType | com.bstek.dorado.sample.standardlesson.entity.SlCompany | DataType可匹配的Java Bean的类型 |

右击DataType选择[通过简单Java对象自动创建PropertyDefs]：



IDE会自动根据这个DataType所设定的match Type定义的POJO对象映射成DataType的PropertyDef信息：



设置每个字段的中文说明，分别设置PropertyDef的label属性如下：



至此，名称为SlCompany的DataType已经创建完成，可以在项目中的任何视图使用。在后面的View开发中我们将会用到这个名称为SlCompany的DataType。

### 1.9启动测试

需要验证项目是否配置成功，重新启动服务器，没有错误输出则配置成功。

## 2、开发步骤

### 2.1创建视图并添加控件

1）创建contacts包

2）在contacts包下新建视图文件ConpanyInfo.view.xml

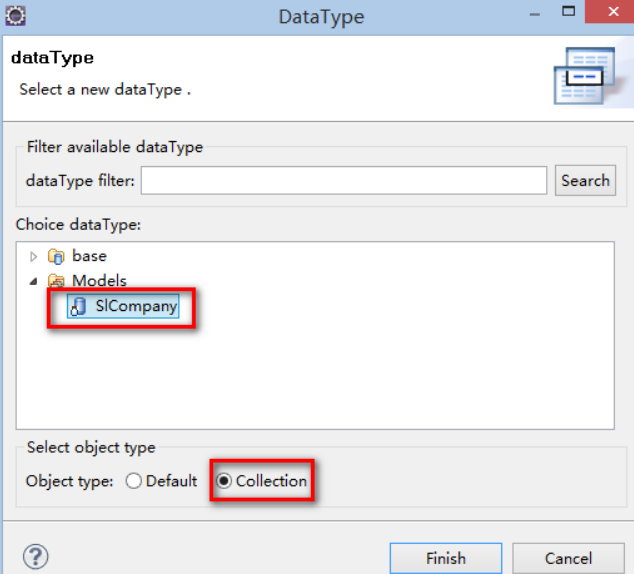
3）新建并配置DataSet

在[View]节点下添加一个DataSet控件并设置属性：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 说明 |
| id | datasetCompany | 控件的id |

DataSet是一个数据容器控件。例如：datasetCompany中可以包含1个到多个Company对象，Company对象的结构是由DataSet的DataType属性定义的。现在定义datasetCompany的dataType属性，我们选择dataType属性：

点击DataType选择向导，在向导中找到之前在Model中定义好的名称为SlCompany的DataType对象。



DataSet设定DataType属性时，在这个向导窗口需要特别注意Select object type

的选择，如果采用Default就表示当前数据容器DataSet只能包含一个DataType的数据实体，而如果选择Collection就表示它可包含这个DataType结构的一个Collection集合。对于DataSet而言这是具有明确区别的，在后期访问DataSet和存取DataSet中数据的相关代码会有所差别，因次在开发时开发人员要明确Select object type属性应该取哪个值。（多数情况下选择Collection）

接下来设定datasetCompany的dataProvider属性为：

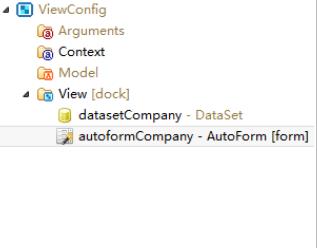
“companyService#getCompany”

定义的dataProvider跟AjaxAction的service同属于服务定位表达式，dataProvider是Dorado7中常见的数据加载技术，主要用于为DataSet加载数据，本身也是一种Ajax数据加载形式。

companyService#getCompany这个表达式的意思为在Spring的beanFactory中寻找id为companyService的bean，执行其中的getCompany方法为dataSetCompany提供数据。

4）新建并配置AutoForm

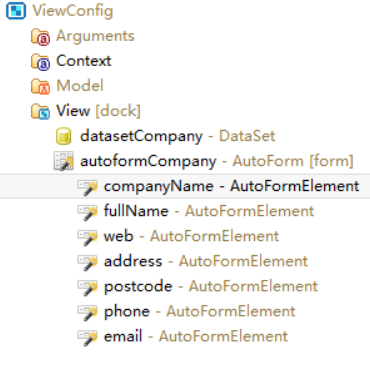
datasetCompany是一个数据容器控件，在页面上是不可见的。为了展现其中的数据我们需要添加一个可以展现数据的控件对象，如AutoForm。



设定AutoForm的属性：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 说明 |
| id | autoformCompany | 控件Id |
| dataSet | datasetCompany | 通过dataSet属性告知AutoForm绑定的哪些数据，以便在AutoForm中展示和处理。 |

接着右击autoformCompany选择[Generate AutoForm Elements]，生成相关字段：



（删除companyId这个AutoForm Elements）

假如不生成AutoForm Elements，那么AutoForm将显示绑定DataType的所有字段。

AutoForm默认是按2列显示，如果需要其单列显示，设置autoformCompany的属性如下：

|  |  |  |
| --- | --- | --- |
| 属性 | 值 | 说明 |
| cols | \* | AutoForm列显示布局表达式 |
| width | 450 | 控件的密度 |
| labelSeparator | : | AutoForm Element中标签和输入框间的分隔符 |

本页面仅作为企业信息的展示，因此需要设置autoformCompany为只读。

readOnly为true

如果需要设置autoformCompany只读还要另一种方法，直接设置datasetCompany的readOnly属性为true即可。AutoForm控件为数据敏感型控

件，当其数据来源的DataSet设置为只读时，那么AutoForm自然也变为只读，这样做的好处在于只要单独设置DataSet的readOnly属性就能控制所有与之绑定的数据敏感型控件的只读状态。

### 2.2准备后台服务

1）在service包中新建CompanyService.java：

@Component

public class CompanyService {

@Resource

private SlCompanyDao slCompanyDao;

@DataProvider

public Collection<SlCompany> getCompany() {

//执行SlCompanyDao的父类HibernateDao中的getAll方法

return slCompanyDao.getAll();

}

}

@Resource注解被用来激活一个命名资源（named resources）的依赖注入，在这里将SlCompanyDao注入。

@DataProvider是Dorado专门提供的标记，与@Expose相似，@DataProvider会自动将当前方法对应的服务，根据一系列的规则，在运行时注册成全局的DataProviderManager的对象中，用以对DataSet提供数据加载服务。

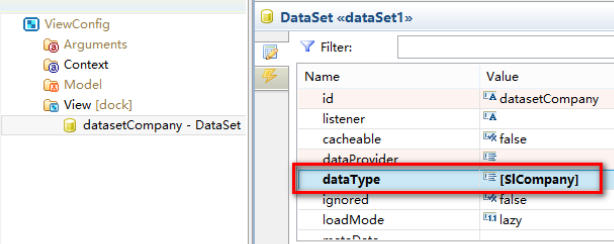
### 2.3查看运行效果

<http://localhost:8080/standard-lesson/com.bstek.dorado.sample.standardlesson.junior.contacts.CompanyInfo.d>



### 2.4另一种实现方式

之前设置datasetCompany的dataType属性为[SlCompnay]表示此DataType结构的一个Collection集合：



假如数据库中只有一条Company记录，我们仅需要加载一条记录该怎么办呢？

首先修改datasetCompany的dataType属性，选择Object Type时设置为Default，采用default就表示当前数据容器DataSet只能包含一个DataType的数据实体。

后台服务也需要更改：

@DataProvider

public SlCompany getCompany() {

List<SlCompany> lsit = slCompanyDao.getAll();

for (SlCompany slCompany : lsit) {

return slCompany;

}

return null;

}

方法返回单个对象，先从数据库中查询所有的记录，然后遍历出一个对象。

## 3、小结

* AutoForm控件的使用和布局
* 了解DataSet和DataType的概念