pojo的绑定

描述

通常ssm开发时候，我们为方便数据的扩容，同时多个表的操作，我们就需要在原有的pojo（或者叫domain）对象去进行封装一层。理论的话就是实现多组合少继承。

例如：

```
class A{
//该类由逆向工程生成，不要动所以在原有的类上我们去做扩展，再去操作
  属性....
  方法....
}
```

```
//该类由逆向工程生成，不要动所以在原有的类上我们去做扩展，再去操作
class B{
  属性....
  方法....
}
```

```
//A扩展类
class AExtender extends A{
  private A a;
生成相应的setter和getter...
}
```

```
//B扩展类
class BExtender extends B{
 private B b;
  生成相应的setter和getter...
}
```

```
//包装的pojo
class AandB{
 private A a;
 private B b;
 生成相应的setter和getter...
}
```

观察上面的代码得：当我在web层去操作封装好的pojo，数据能从应用层直接传到持久层，同时也降低了不同业务代码间的耦合性，不同代码内部高度内聚。同时数据库也方便扩容。

---

springMVC中包装类pojo的传递

传递的数据和接受的数据必须遵循ognl的的规范，才能传递成功。！

```
//页面定义
<input type="text" name="A.name" />
```

```
//controller定义
public String useraddsubmit(Model model,AandB ab)throws Exception{
 System.out.println(ab.getName());
```

这样在后台界面就能接受到收到的值

集合类的传递（数组，List，Map）

1. 字符串数组

```
//页面定义如下：
//页面选中多个checkbox向controller方法传递
<input type="checkbox" name="item_id" value="001"/>
<input type="checkbox" name="item_id" value="002"/>
<input type="checkbox" name="item_id" value="002"/>
```

传递到controller方法中的格式是：001,002,003

```
Controller方法中可以用String[]接收，定义如下：
public String deleteitem(String[] item_id)throws Exception{
  System.out.println(item_id);
}
```

2. List

List中存放对象，并将定义的List放在包装类中，action使用包装对象接收。

List中对象：

```
成绩对象
Public class QueryVo {
Private List<Items> itemList;//商品列表

  //get/set方法..
}
```

jsp代码如下：

```
<tr>
 <td><input type="text" name="itemsList[0].name" /></td>
 <td><input type="text" name="itemsList[0].price"/></td>
 .....
 .....
</tr>
```

Contrller方法定义如下：

```
public String useraddsubmit(Model model,QueryVo queryVo)throws Exception{
System.out.println(queryVo.getItemList());
}
```

3. Map

在包装类中定义Map对象，并添加get/set方法，action使用包装对象接收。

```
//包装类中定义Map对象如下：
Public class QueryVo {
private Map<String, Object> itemInfo = new HashMap<String, Object>();
  //get/set方法..
}
```

```
//页面定义如下：
<tr>
<td>学生信息：</td>
<td>
姓名：<inputtype="text"name="itemInfo['name']"/>
年龄：<inputtype="text"name="itemInfo['price']"/>
.. .. ..
</td>
</tr>
```

```
//Contrller方法定义如下：
public String useraddsubmit(Model model,QueryVo queryVo)throws Exception{
```

```
System.out.println(queryVo.getStudentinfo());
}
```

自定义参数绑定

描述

数据传递针对的是简单数据的传递，当再复杂的数据类型就需要我们手动去配置转换格式。才可以绑定成功。

例如：日期类型的传入

1. 自定义Converter实现Converter的接口，

```
public class CustomDateConverter implements Converter<String, Date> {

 @Override
 public Date convert(String source) {
  try {
   SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    return simpleDateFormat.parse(source);
  } catch (Exception e) {
   e.printStackTrace();
  }
  return null;
 }
}
```

2. 将自定义的转换器配置到适配器中（推荐）

```
<mvc:annotation-driven conversion-service="conversionService">
</mvc:annotation-driven>
<!-- conversionService-->
 <bean id="conversionService"
  class="org.springframework.format.support.FormattingConversionServiceFactoryBea">
  <!-- 自定义的转换器 -->
  <property name="converters">
   <list>
    <bean class="cn.yxd.controller.converter.CustomDateConverter"/>
   </list>
  </property>
 </bean>
```

2. 配置方式2(不使用mvc:annotation-driven的适配器配置)

```
<!--注解适配器 -->
 <bean
  class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter">
   <property name="webBindingInitializer" ref="customBinder"></property>
 </bean>

 <!-- 自定义webBinder -->
 <bean id="customBinder"
  class="org.springframework.web.bind.support.ConfigurableWebBindingInitializer">
  <property name="conversionService" ref="conversionService" />
 </bean>
 <!-- conversionService -->
 <bean id="conversionService"
  class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
  <!-- 转换器 -->
  <property name="converters">
   <list>
    <bean class="cn.itcast.ssm.controller.converter.CustomDateConverter"/>
   </list>
  </property>
 </bean>
```

完