

new 4、SpringMVC 实战






CRM 系统的客户 CRUD

使用 SpringMVC4 + Spring4 + Hibernate5 + restful 实现
















导入 Hibernate 和 spring 相关 jar 包:

JavaWeb 相关包:










PS: 本框架为手工从 JavaSE 项目开始搭建, 所以需要 javaWeb 相关包

 el-api.jar	2018/10/24 19:56	Executable Jar File	79 KB
 jsp-api.jar	2018/10/24 19:56	Executable Jar File	60 KB
 jstl.jar	2018/10/23 19:20	Executable Jar File	21 KB
 servlet-api.jar	2018/10/12 10:22	Executable Jar File	239 KB
 standard.jar	2018/10/23 19:20	Executable Jar File	385 KB




Spring 完整包:

 aopalliance.jar	2017/12/11 16:02	Executable Jar File	5 KB
 aspectjrt.jar	2017/12/11 16:02	Executable Jar File	115 KB
 aspectjweaver.jar	2017/12/11 16:02	Executable Jar File	1,816 KB
 commons-logging-1.2.jar	2017/12/11 16:02	Executable Jar File	61 KB
 spring-aop-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	372 KB
 spring-aspects-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	58 KB
 spring-beans-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	743 KB
 spring-context-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	1,109 KB
 spring-context-support-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	183 KB
 spring-core-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	1,085 KB
 spring-expression-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	258 KB
 spring-jdbc-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	417 KB
 spring-orm-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	466 KB
 spring-test-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	580 KB
 spring-tx-4.3.3.RELEASE.jar	2017/12/11 16:02	Executable Jar File	261 KB
 spring-web-4.3.3.RELEASE.jar	2017/12/11 21:48	Executable Jar File	795 KB
 spring-webmvc-4.3.3.RELEASE.jar	2017/12/11 21:45	Executable Jar File	893 KB

Hibernate 基本包:

 antlr-2.7.7.jar	2017/12/11 16:02	Executable Jar File	435 KB
 dom4j-1.6.1.jar	2017/12/11 16:02	Executable Jar File	307 KB
 geronimo-jta_1.1_spec-1.1.1.jar	2017/12/11 16:02	Executable Jar File	16 KB
 hibernate-commons-annotations-5.0.0.Final.jar	2017/12/11 16:02	Executable Jar File	74 KB
 hibernate-core-5.0.7.Final.jar	2017/12/11 16:02	Executable Jar File	5,453 KB
 hibernate-jpa-2.1-api-1.0.0.Final.jar	2017/12/11 16:02	Executable Jar File	111 KB
 jandex-2.0.0.Final.jar	2017/12/11 16:02	Executable Jar File	184 KB
 javassist-3.18.1-GA.jar	2017/12/11 16:02	Executable Jar File	698 KB
 jboss-logging-3.3.0.Final.jar	2017/12/11 16:02	Executable Jar File	66 KB

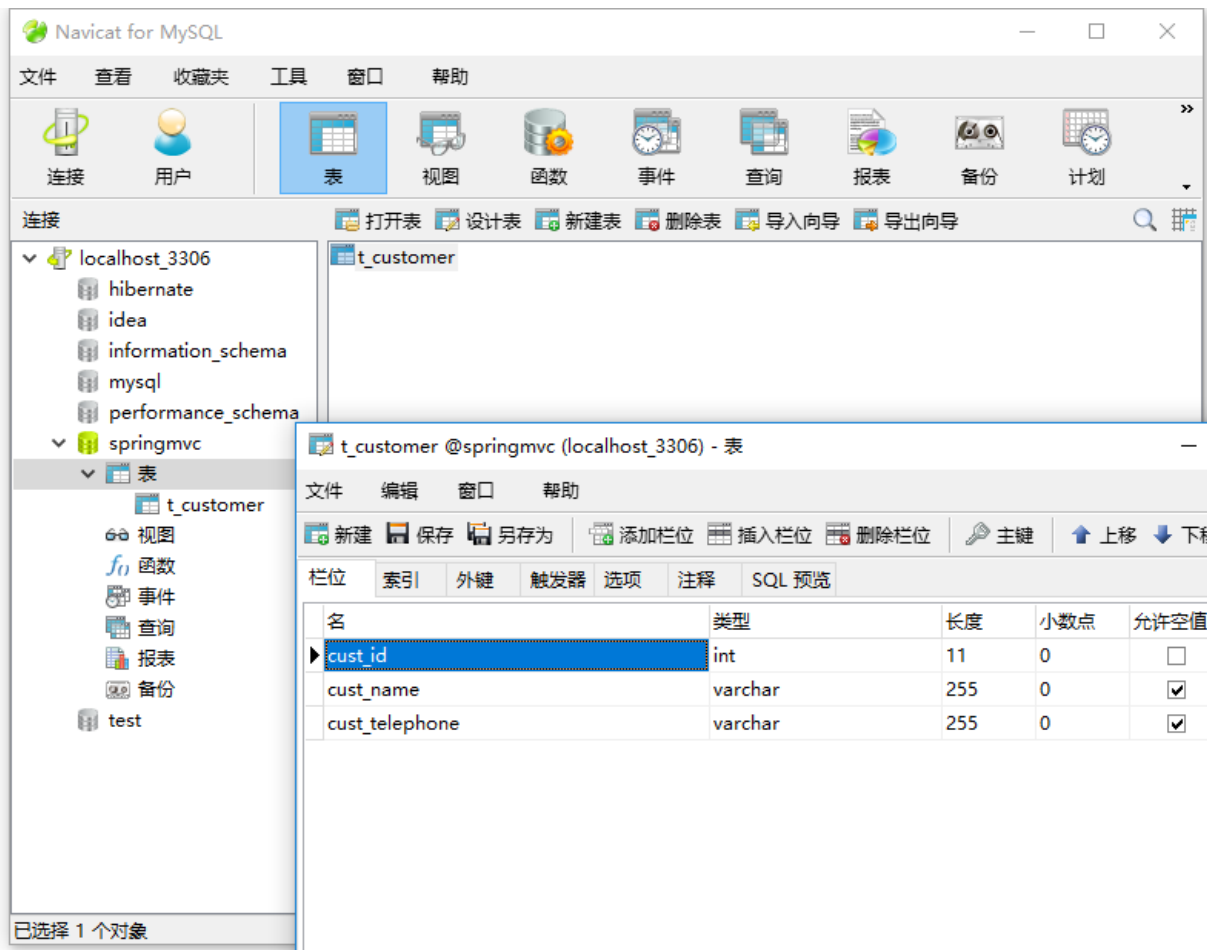
c3p0 连接池包:

 c3p0-0.9.2.1.jar	2017/12/11 16:03	Executable Jar File	414 KB
 hibernate-c3p0-5.0.7.Final.jar	2017/12/11 16:03	Executable Jar File	12 KB
 mchange-commons-java-0.2.3.4.jar	2017/12/11 16:03	Executable Jar File	568 KB

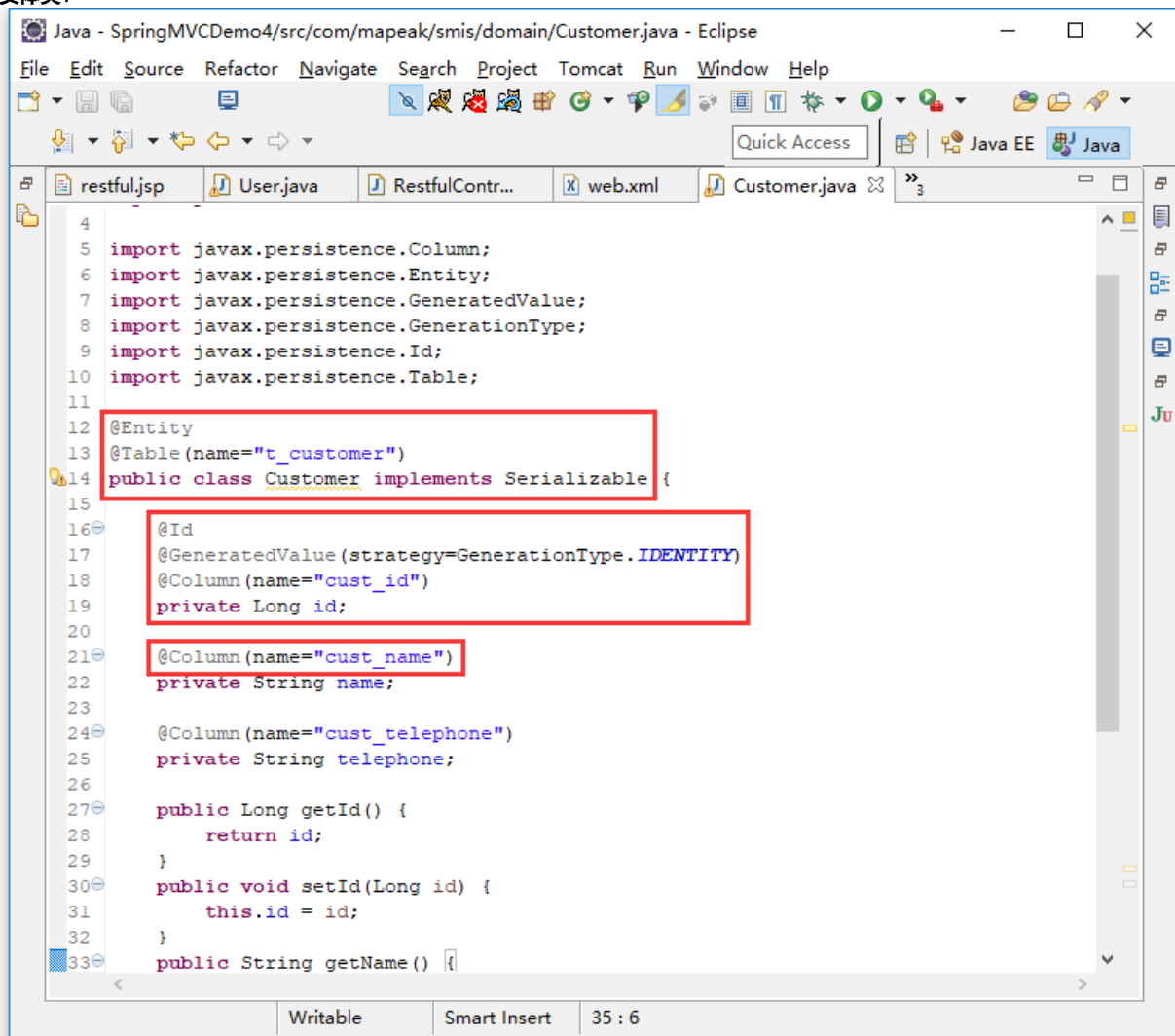
MySQL 驱动包:

 mysql-connector-java-5.1.7-bin.jar	2017/12/11 16:02	Executable Jar File	694 KB
--	------------------	---------------------	--------

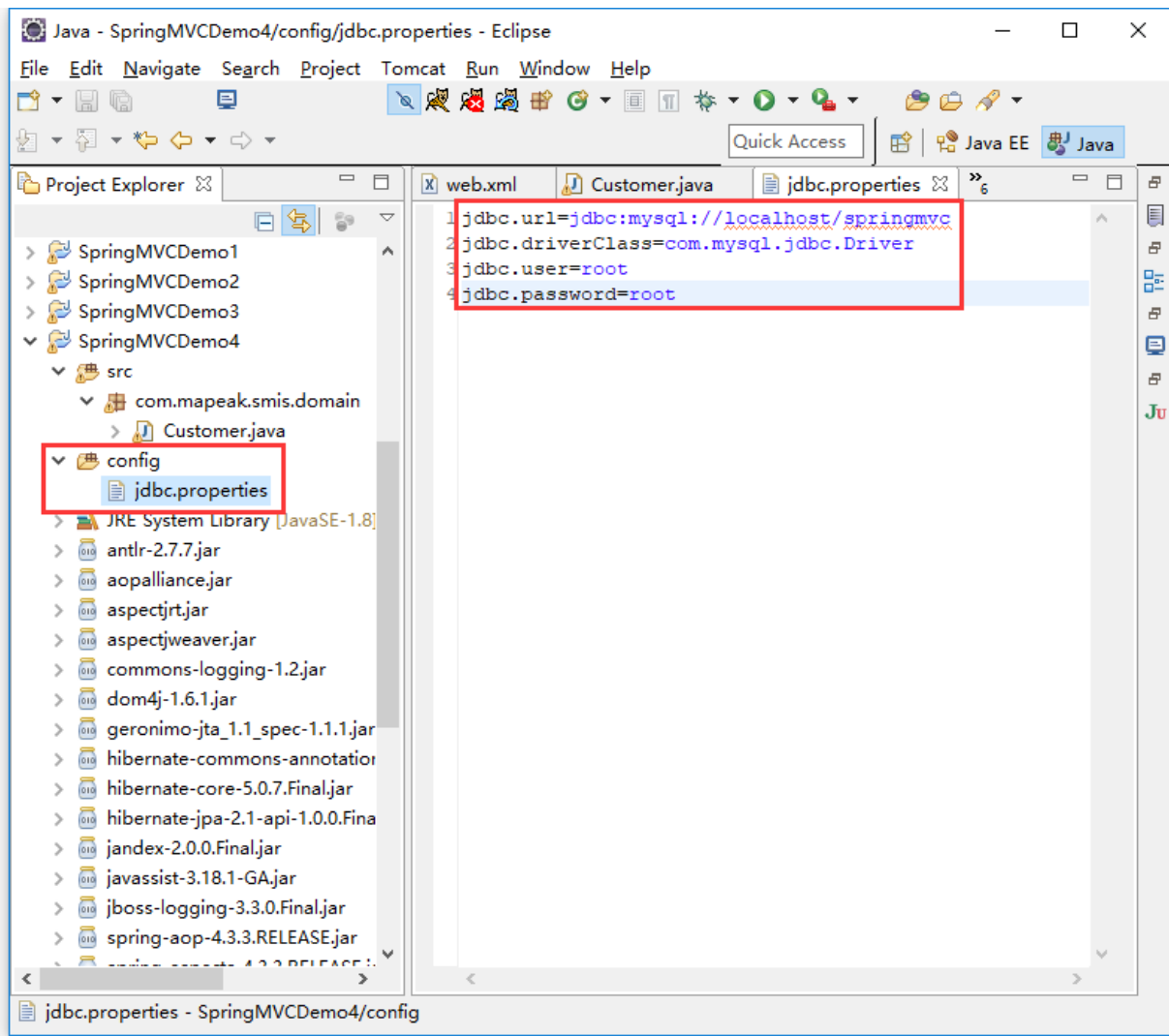
数据库表:

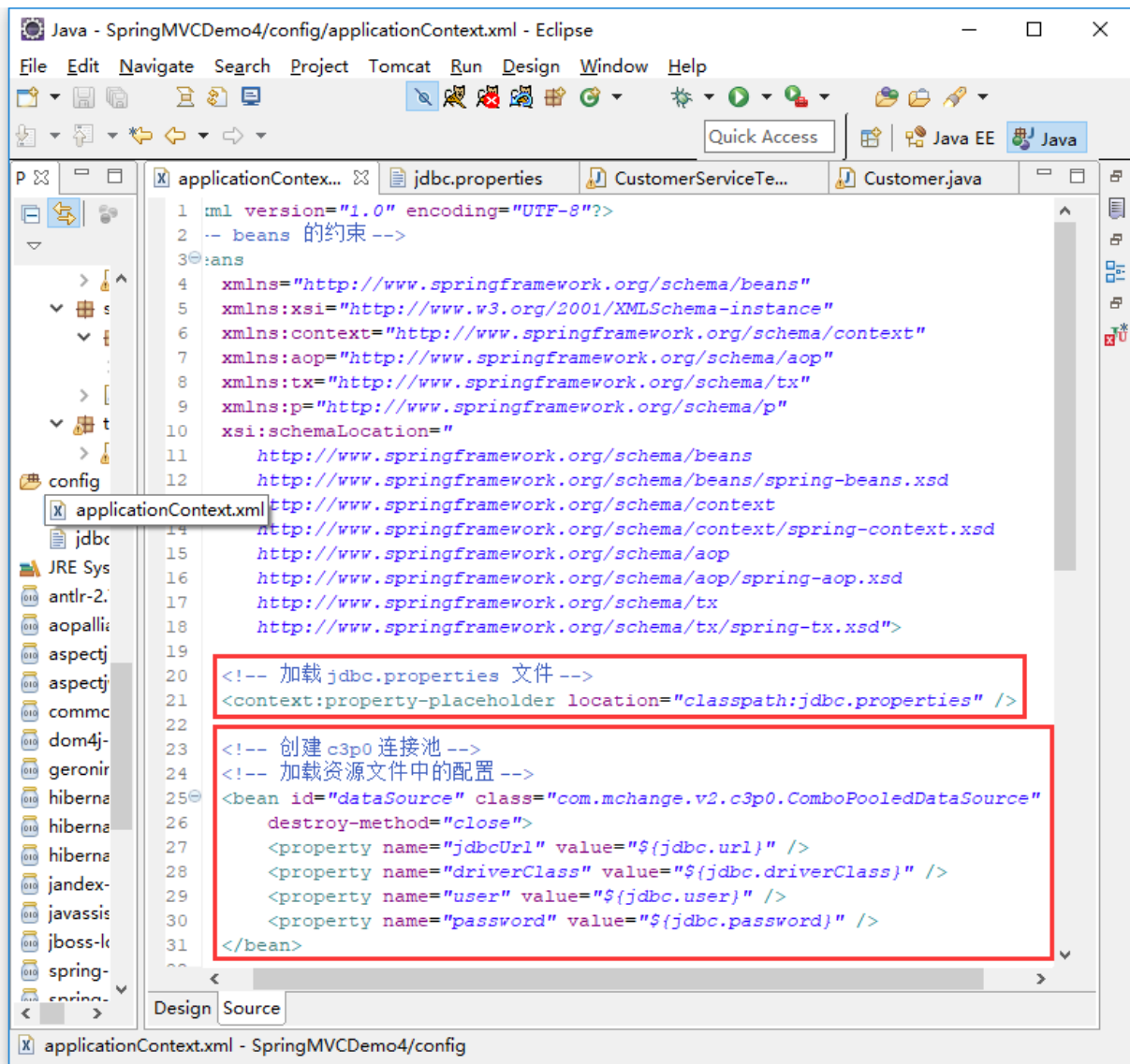


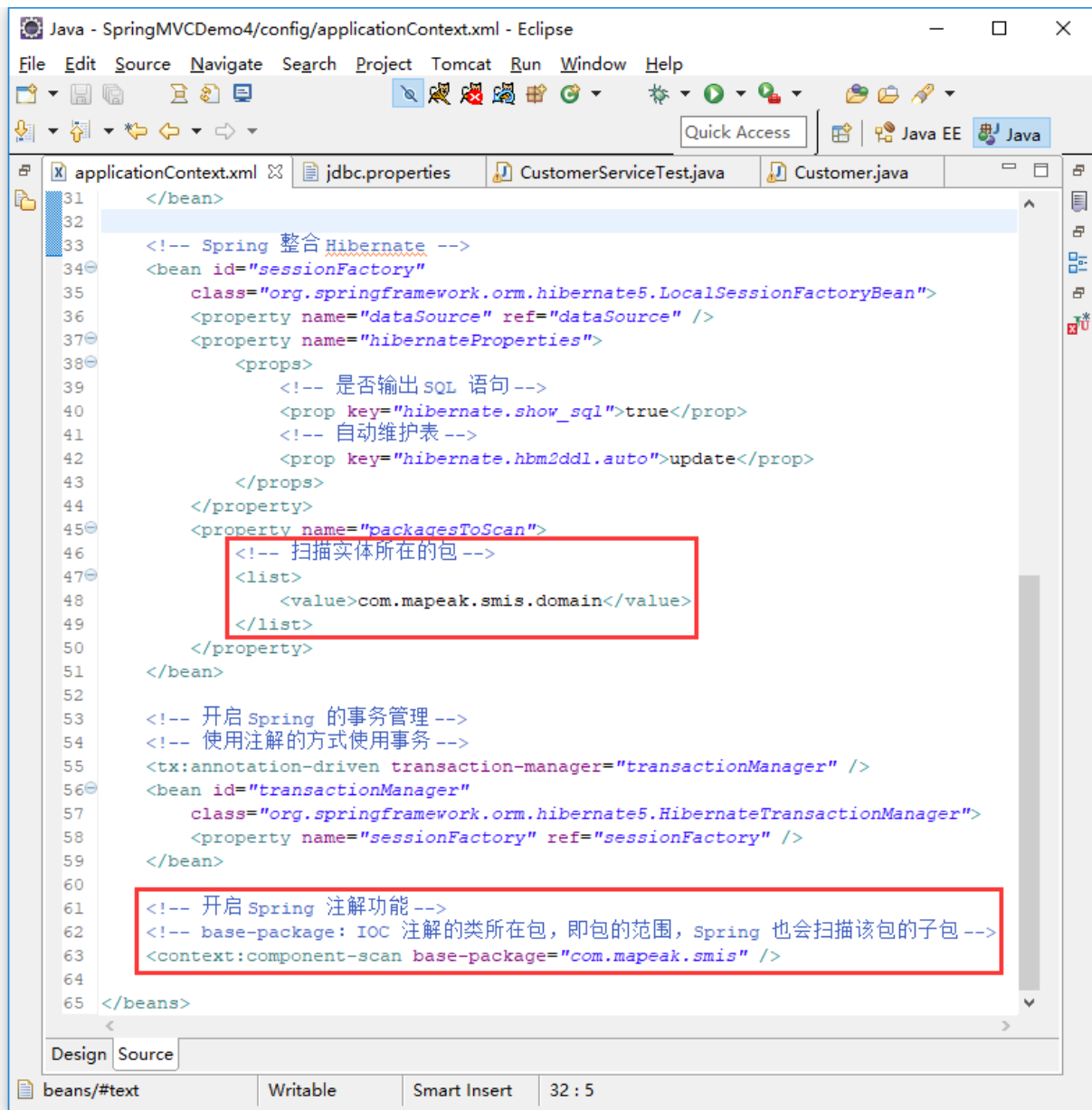
实体类:



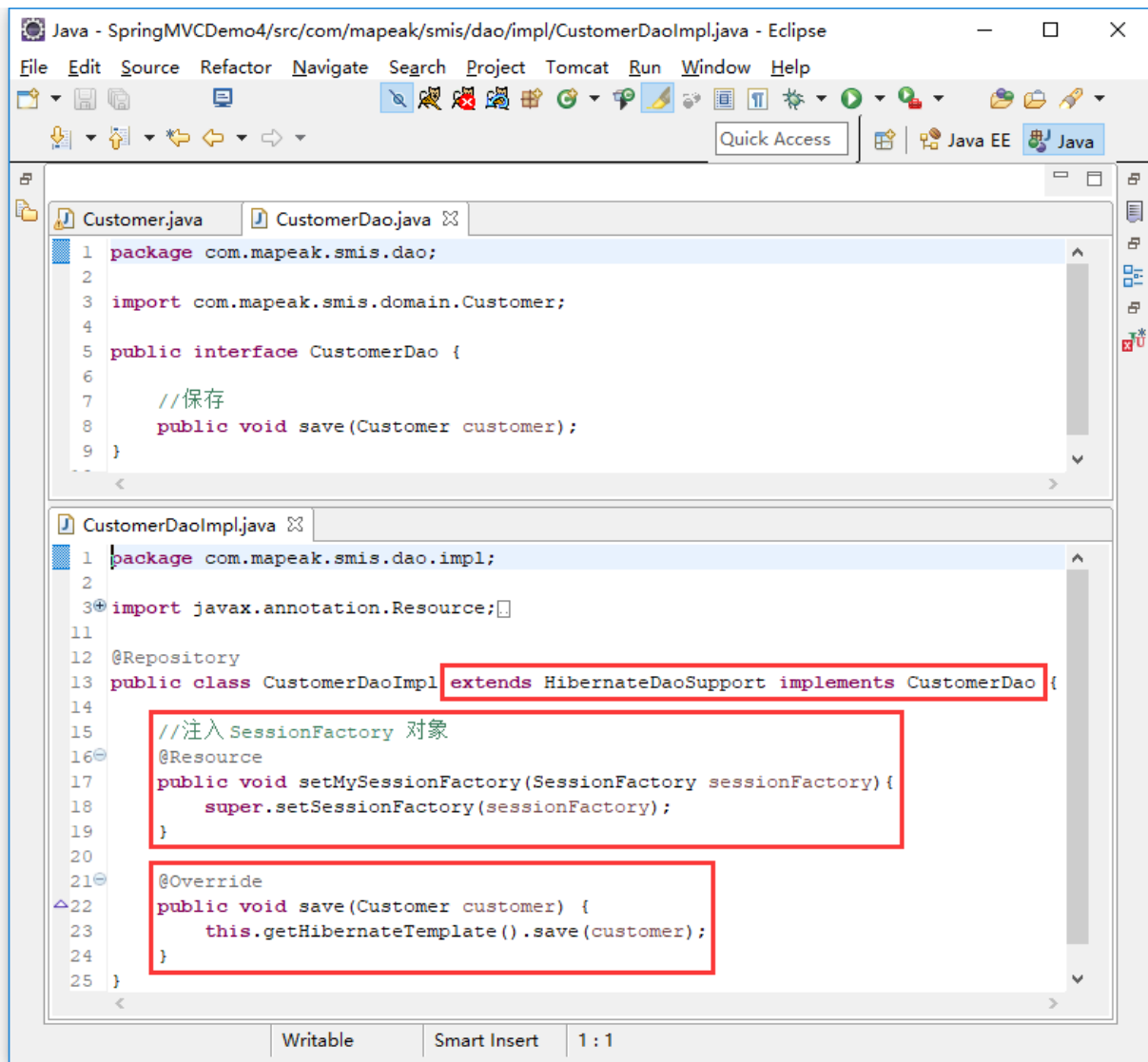
配置 applicationContext.xml 文件:



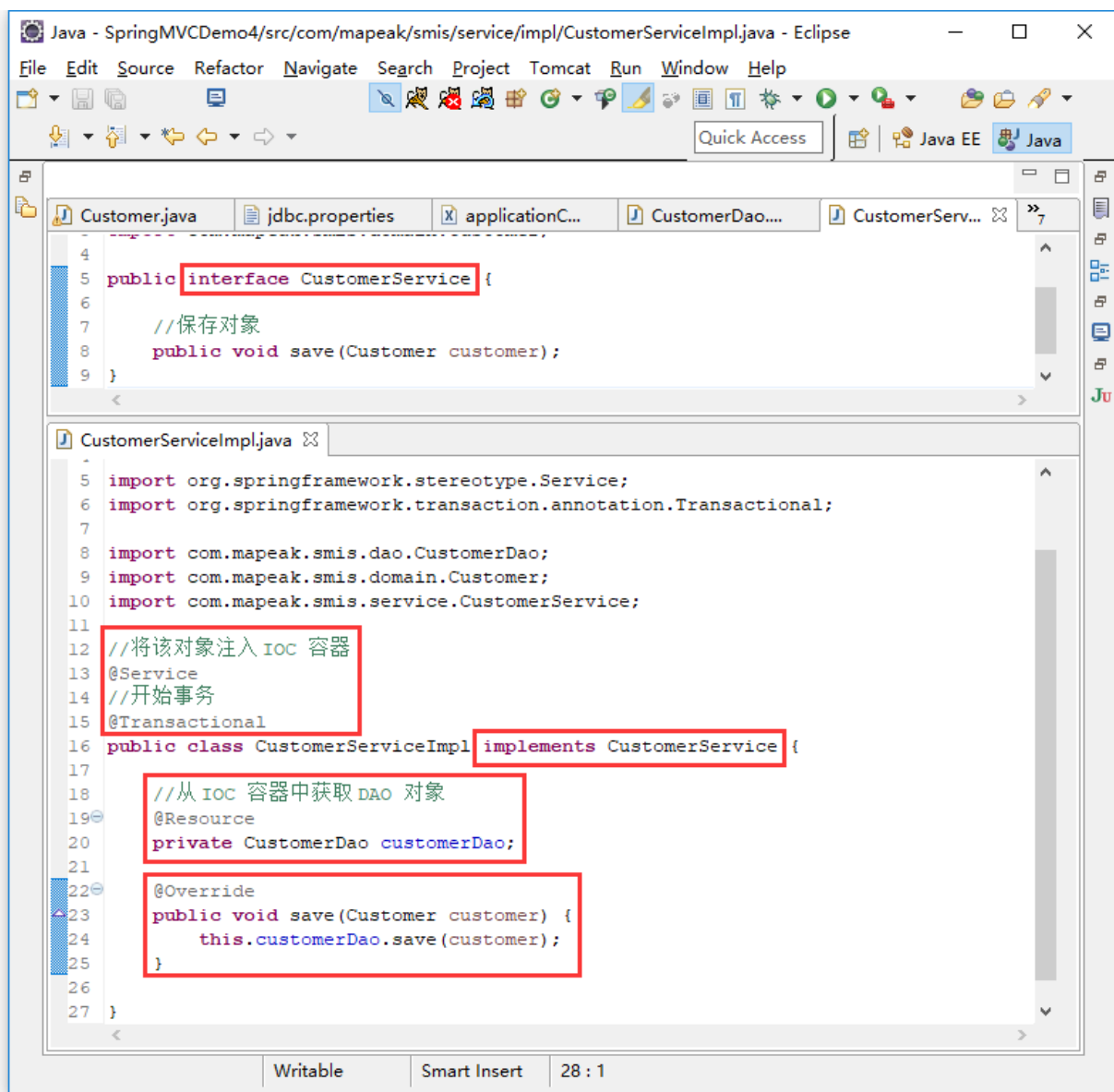




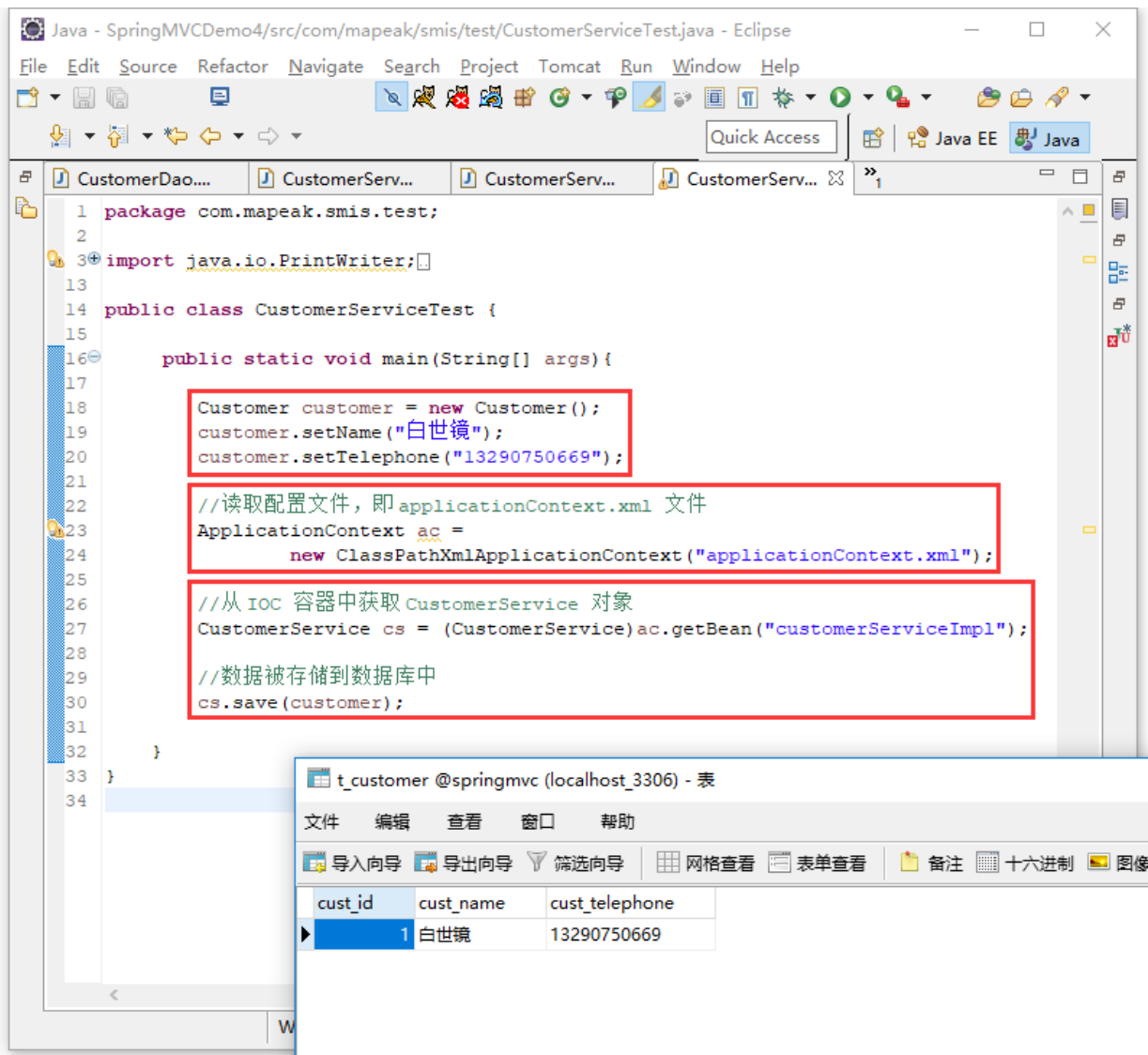
编写 DAO 接口和实现:



编写 Service 接口和实现：



测试 Service 方法:



SpringMVC 完成客户添加

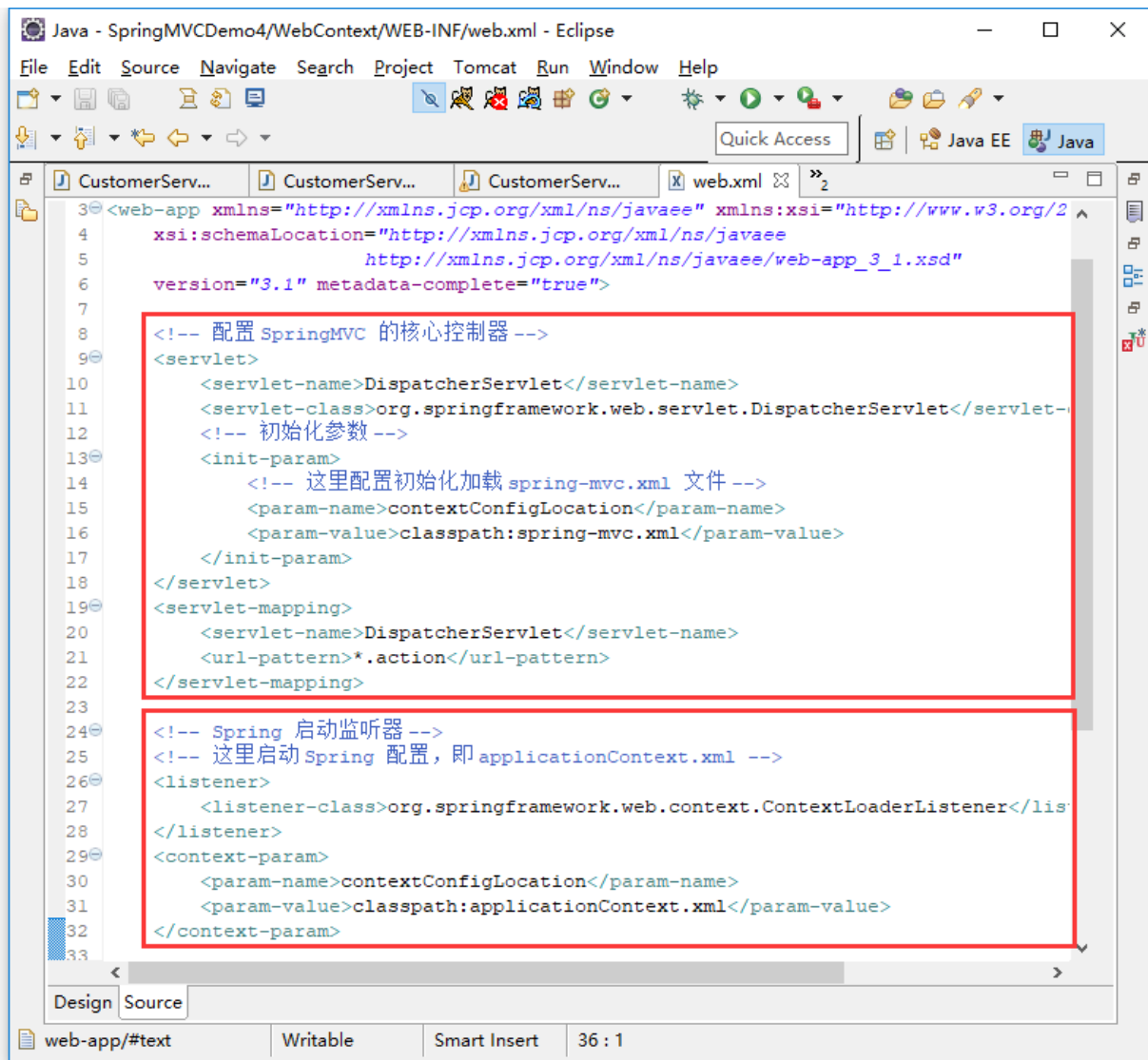
导入 SpringMVC 支持:

 spring-web-4.3.3.RELEASE.jar	2017/12/11 21:48	Executable Jar File	795 KB
 spring-webmvc-4.3.3.RELEASE.jar	2017/12/11 21:45	Executable Jar File	893 KB

PS:

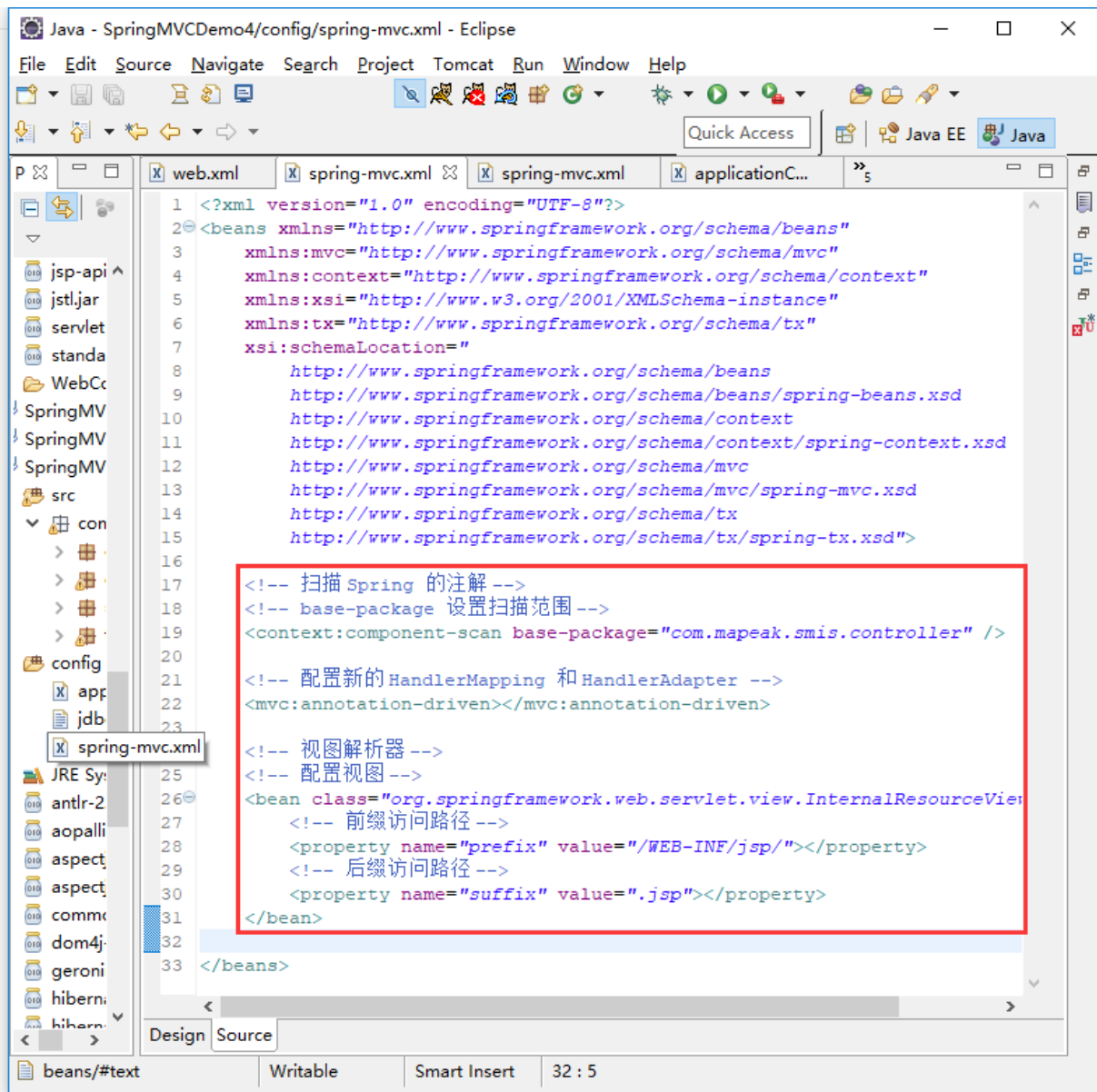
spring-web-4.3.3.RELEASE.jar (这个包启动 spring 的 web 环境)
这两个包之前已导入

编写 web.xml 启动 SpringMVC、Spring:



编写 SpringMVC 配置:

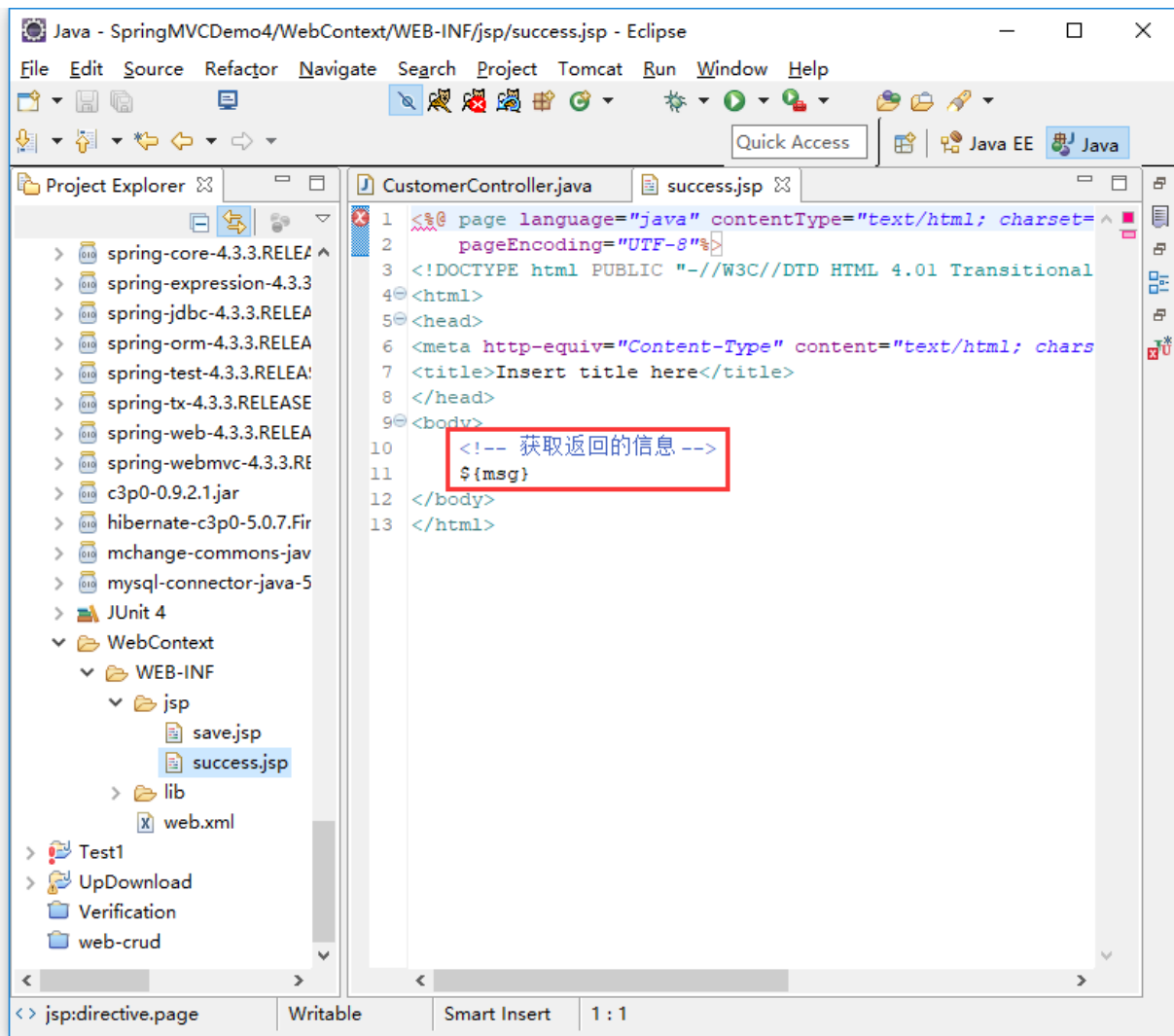
即配置 **spring-mvc.xml** 文件



页面:

以 restful 方式发送请求

PS: 这里表单名称必须与封装的实体对象 User 属性名称一致



Controller:

以 restful 方式执行 CREATE 操作, 即: method=RequestMethod.POST

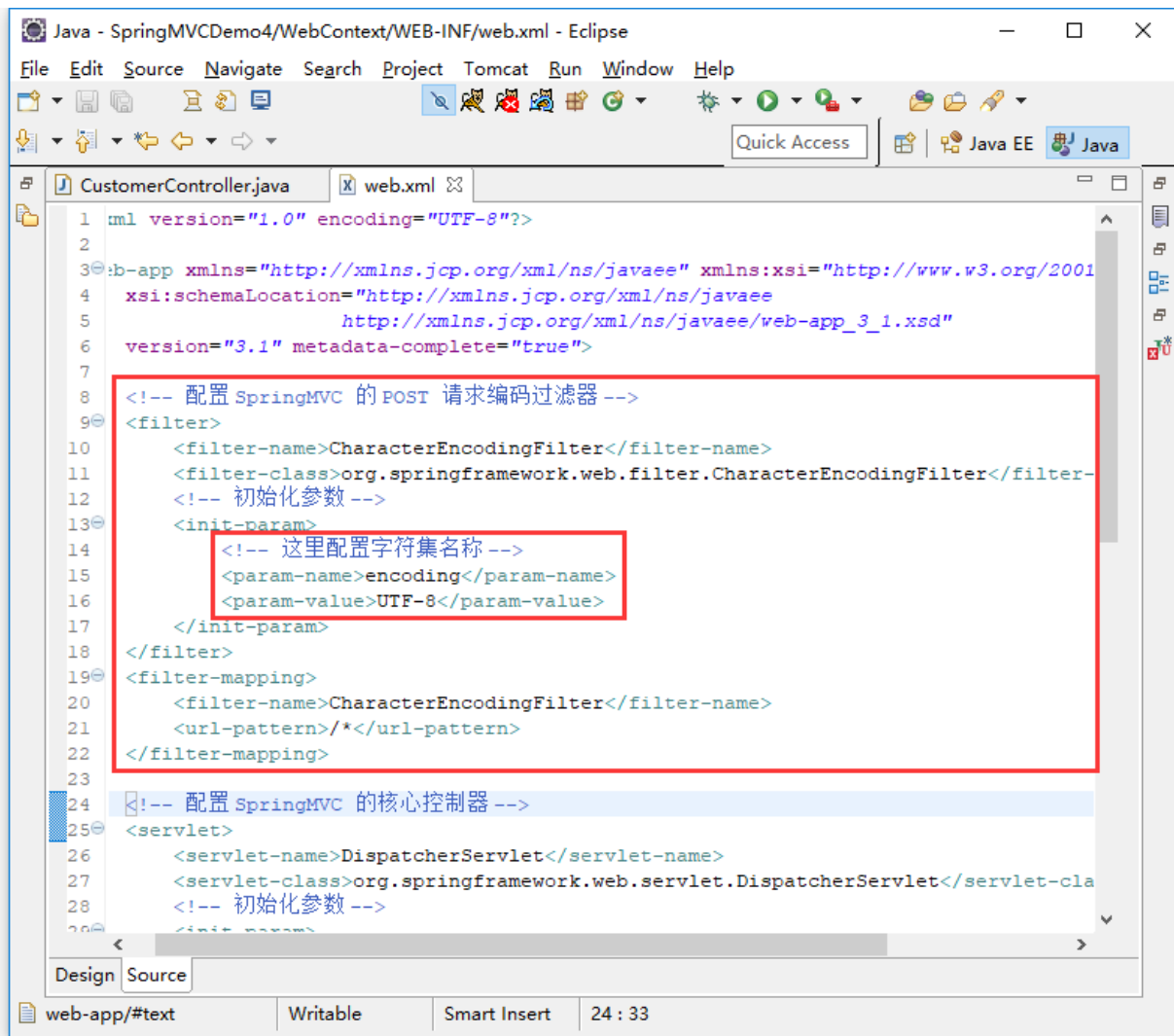
PS:

由于 save.jsp 在 WEB-INF 文件夹中, 无法直接访问, 所以需要跳转控制器
这里使用 restful 风格, 并将表单的数据封装到 Customer 实体类中
这里数据返回页面, 可以用 Model、ModelMap、Map, 将数据返回到页面

```
16
17 @Controller
18 @RequestMapping("/customer")
19 public class CustomerController {
20
21     //因为 save.jsp 页面在 WEB-INF 文件夹下，外部（浏览器）无法直接访问
22     //用于跳转到 WEB-INF/jsp/save.jsp 页面
23     @RequestMapping("/saveUI")
24     public String saveUI(){
25         return "save";
26     }
27
28     //注入 CustomerService 对象
29     @Resource
30     private CustomerService customerService;
31
32     //添加用户
33     //这里使用 restful 风格，并将表单的数据封装到 Customer 实体类中
34     //这里数据返回页面，可以用 Model、ModelMap、Map，将数据返回到页面
35     @RequestMapping(method=RequestMethod.POST)
36     public String save(Customer customer, Map<String, Object> model){
37         customerService.save(customer);
38         //保存提示信息
39         model.put("msg", "添加成功");
40         //请求转发跳转到 success.jsp 页面
41         return "success";
42     }
43
44 }
45
```

添加 SpringMVC 的编码过滤器:

web.xml 里配置，解决中文乱码问题。



测试:

Java - SpringMVCDemo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse

File Edit Source Refactor Navigate Search Project Tomcat Run Window Help

Quick Access Java EE Java

list.jsp *CustomerCon... save.jsp edit.jsp index.jsp restful.jsp »_1

```
22 //用于跳转到 WEB-INF/jsp/save.jsp 页面
23 @RequestMapping("/saveUI")
24 public String saveUI(){
25     return "save";
26 }
27
28 //注入 CustomerService 对象
29 @Resource
30 private CustomerService customerService;
31
32 //添加用户
33 //这里使用 restful 风格，并将表单的数据封装到 Customer 实体类中
34 //这里数据返回页面，可以用 Model、ModelMap、Map，将数据返回到页面
35 @RequestMapping(method=RequestMethod.POST)
36 public String save(Customer customer, Map<String, Object> model){
37     customerService.save(customer);
38     //保存提示信息
39     model.put("msg", "添加成功");
40     //请求转发跳转到 success.jsp 页面
41     return "success";
42 }
43
44 }
```

客户添加页面

客户名称: 天竺僧人

电话: 15823034527

添加

添加成功

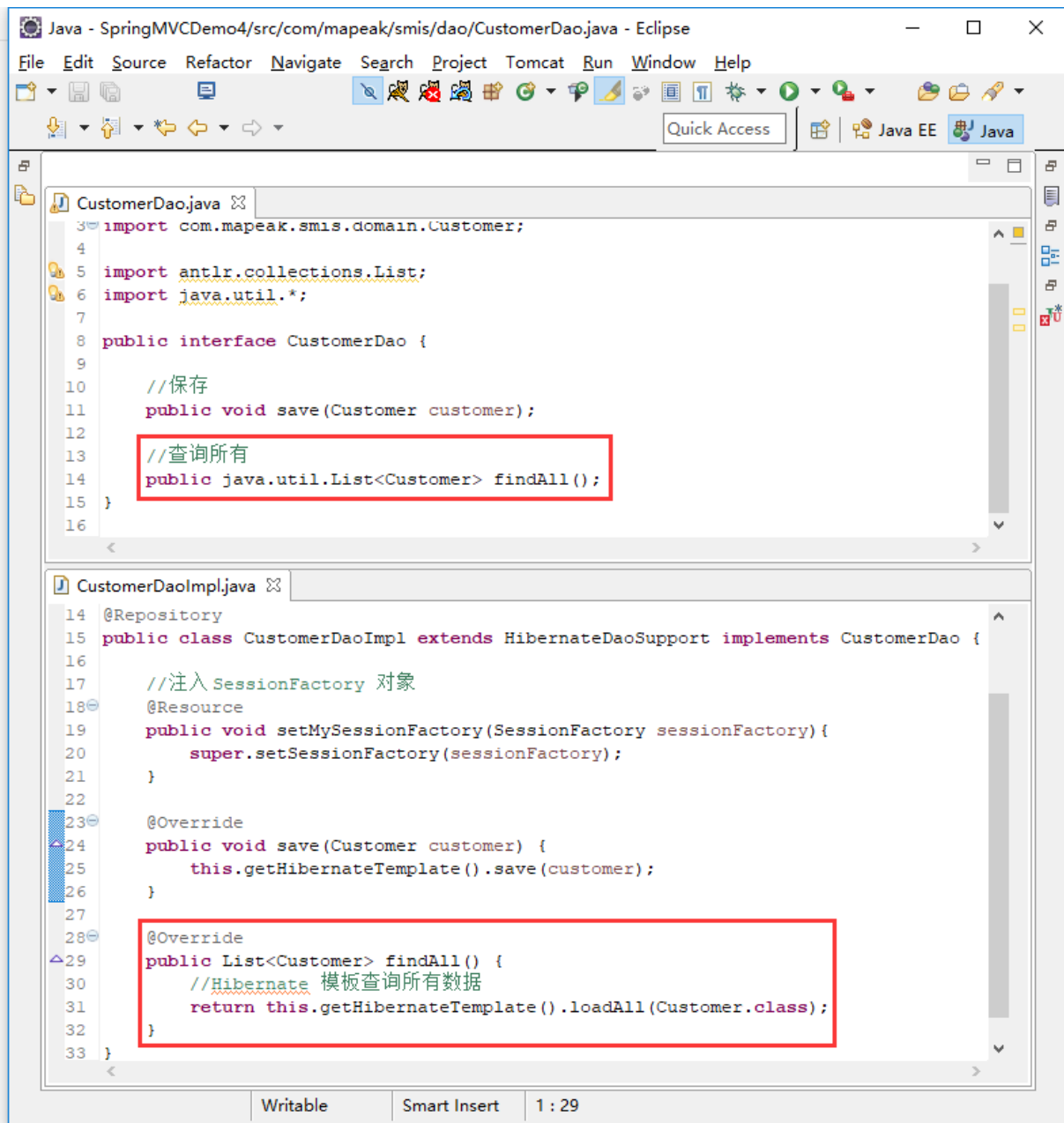
t_customer @springmvc (localhost_3306) - 表

cust_id	cust_name	cust_telephone
1	白世镜	13290750669
3	天竺僧人	15823034527

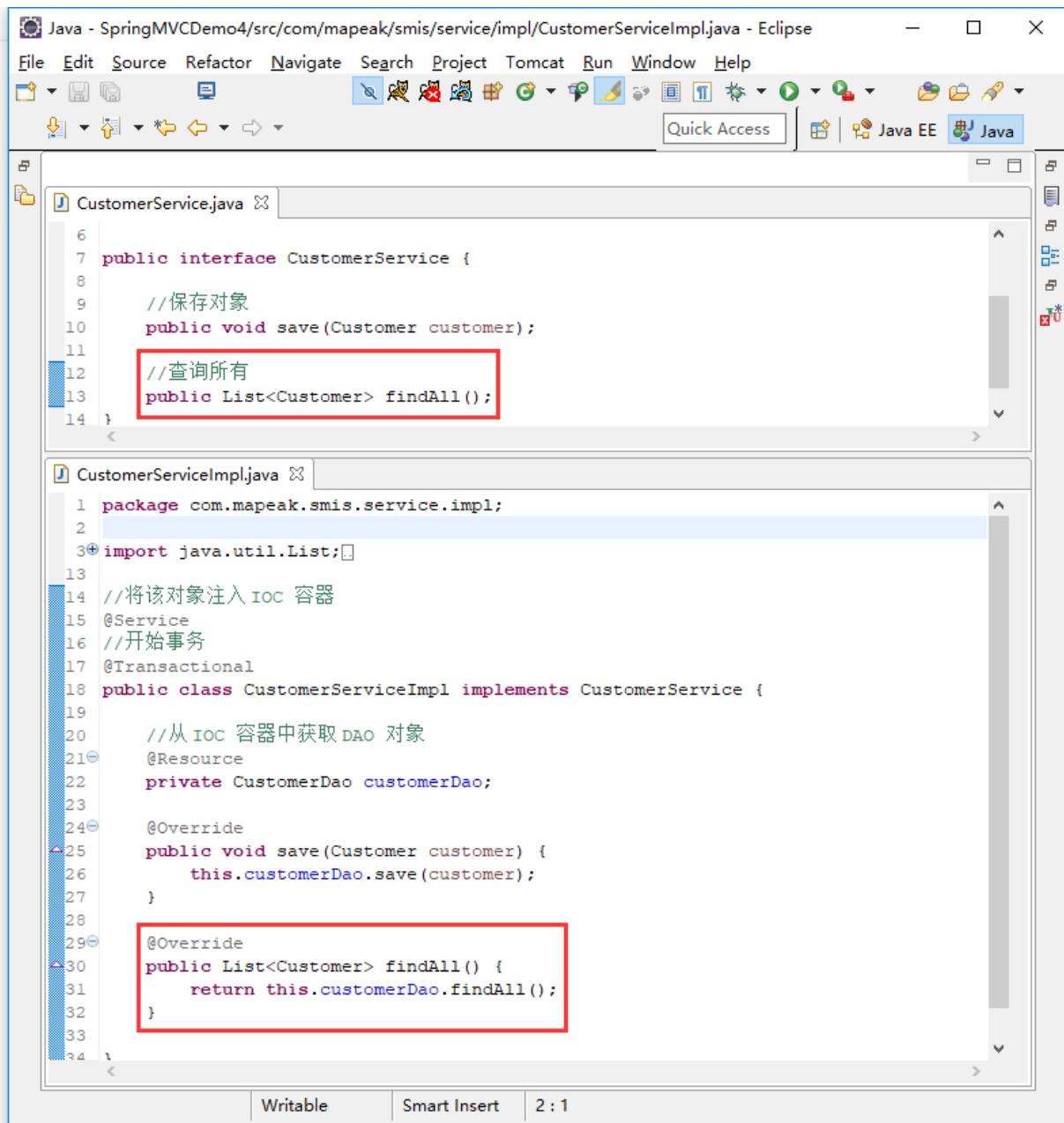
客户查询功能

DAO:

PS: HibernateDaoSupport 所在包: org.springframework.orm.hibernate5.support.HibernateDaoSupport, 不是 hibernate4



Service:



Controller:

以 restful 方式执行 SELECT 操作, 即: `method=RequestMethod.GET`

```
Java - SpringMVC Demo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access Java EE Java

CustomerService.java CustomerController.java
16
17 @Controller
18 @RequestMapping("/customer")
19 public class CustomerController {
20
21     //因为 save.jsp 页面在 WEB-INF 文件夹下，外部（浏览器）无法直接访问
22     //用于跳转到 WEB-INF/jsp/save.jsp 页面
23     @RequestMapping("/saveUI")
24     public String saveUI() {
25         return "save";
26     }
27
28     //注入 CustomerService 对象
29     @Resource
30     private CustomerService customerService;
31
32     //查询所有客户
33     @RequestMapping(method=RequestMethod.GET)
34     public String List(Model model) {
35         //获取数据
36         java.util.List<Customer> stuList = customerService.findAll();
37         //将数据封装到 Model 中
38         model.addAttribute("custList", stuList);
39         //获取数据后页面跳转
40         return "list";
41     }
42
43     //添加用户
44     //这里使用 restful 风格，并将表单的数据封装到 Customer 实体类中
45     //这里数据返回页面，可以用 Model、ModelMap、Map，将数据返回到页面

```

页面:

导入 JSTL 的标签库: <%@ taglib uri="<http://java.sun.com/jsp/jstl/core>" prefix="c" %>

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
5 <html>
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8 <title>查询所有客户</title>
9 </head>
10 <body>
11 <h3>客户所有数据</h3>
12 <table border="1" width="400px;">
13 <tr>
14 <th>客户名称</th>
15 <th>客户电话</th>
16 <th>操作</th>
17 </tr>
18 <c:forEach items="${custList}" var="cust">
19 <tr>
20 <td>${cust.name}</td>
21 <td>${cust.telephone}</td>
22 <td><a href="#">修改</a></td>
23 </tr>
24 </c:forEach>
25 </table>
26
27 </body>
28 </html>
```

测试:

PS: 浏览器的默认请求方式为 get, 所以能执行 GET 请求的 action 方法, 此时获取数据成功

Java - SpringMVC Demo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse

File Edit Source Refactor Navigate Search Project Tomcat Run Window Help

Quick Access Java EE Java

list.jsp CustomerController.java save.jsp Console

```

23 @RequestMapping("/saveUI")
24 public String saveUI(){
25     return "save";
26 }
27
28 //注入 CustomerService 对象
29 @Resource
30 private CustomerService customerService;
31
32 //查询所有客户
33 @RequestMapping(method=RequestMethod.GET)
34 public String List(Model model){
35     //获取数据
36     java.util.List<Customer> stuList = customerService.findAll();
37     //将数据封装到 Model 中
38     model.addAttribute("custList", stuList);
39     //获取数据后页面跳转
40     return "list";
41 }
42
43 //添加
44 //这里
45 //这里
46 @RequestMapping
47 public
48 cu
49 //
50 mo
51 //
52 re

```

D:\JDK\jdk1.8.0_101\bin\javaw.exe (2018年12月27日 下午6:25:15)

Hibernate: select this_.cust_id as cust_id1_0

查询所有客户

localhost:8080/mvc/customer.action

客户所有数据

客户名称	客户电话	操作
白世镜	13290750669	修改
天竺僧人	15823034527	修改

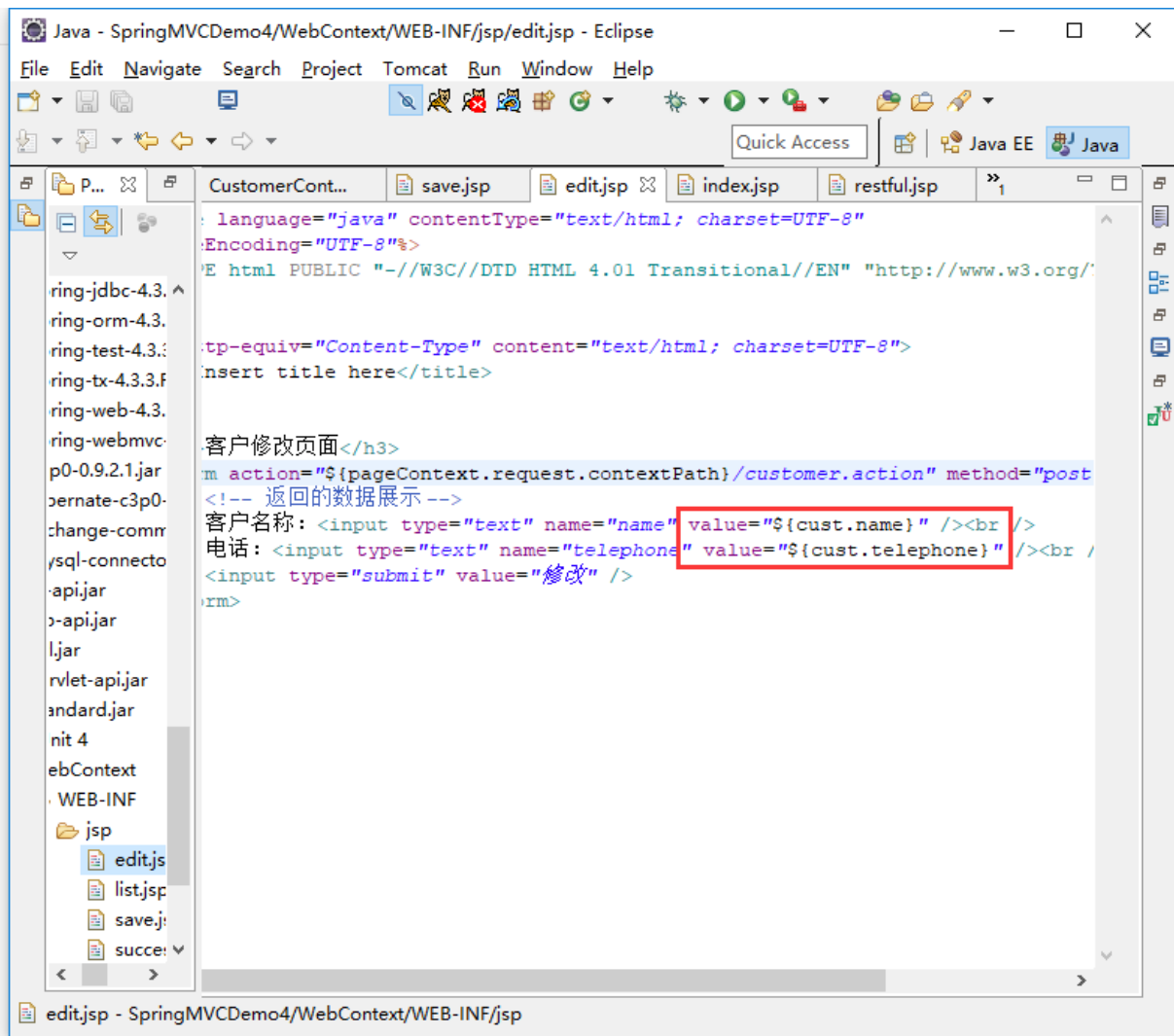
客户数据的回显

页面:

PS: restful 风格也可以在路径上加上参数

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
4 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
5 <html>
6 <head>
7 <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8 <title>查询所有客户</title>
9 </head>
10 <body>
11 <h3>客户所有数据</h3>
12 <table border="1" width="400px;">
13 <tr>
14 <th>客户名称</th>
15 <th>客户电话</th>
16 <th>操作</th>
17 </tr>
18 <c:forEach items="${custList}" var="cust">
19 <tr>
20 <td>${cust.name}</td>
21 <td>${cust.telephone}</td>
22 <!-- 带着 id 返回 -->
23 <!-- restful 风格可以在路径上加上参数 -->
24 <td><a href=
25 "${pageContext.request.contextPath}/customer/${cust.id}.action">修改</a>
26 </td>
27 </c:forEach>
28 </table>
29
30 </body>
```

PS: `value="${cust.name}"` 展示返回的数据



Controller:

PS: 获取 url 中的 id 值

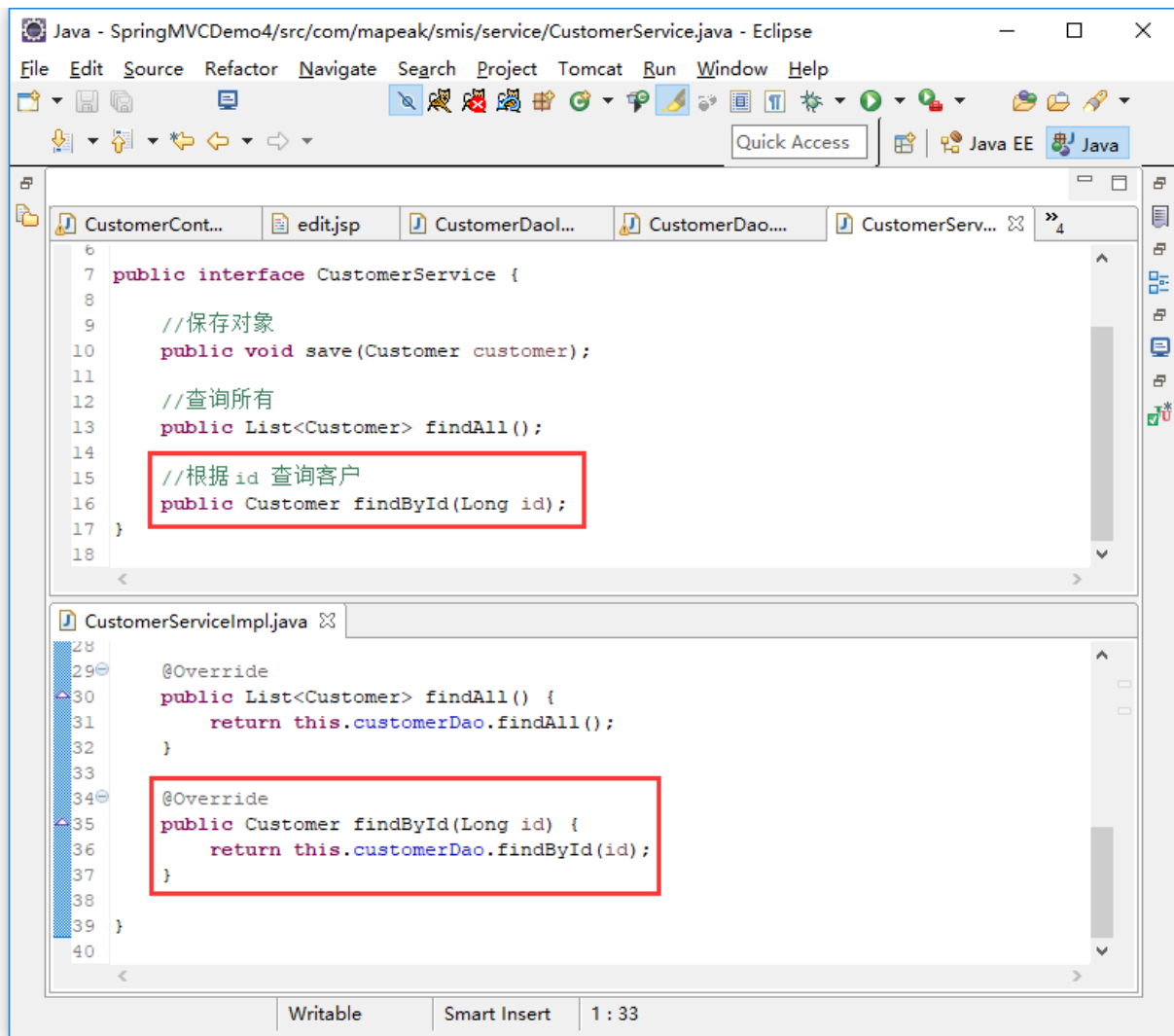
```
Java - SpringMVCDemo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access Java EE Java

CustomerCont... CustomerDaol... CustomerServ... CustomerServ... »6

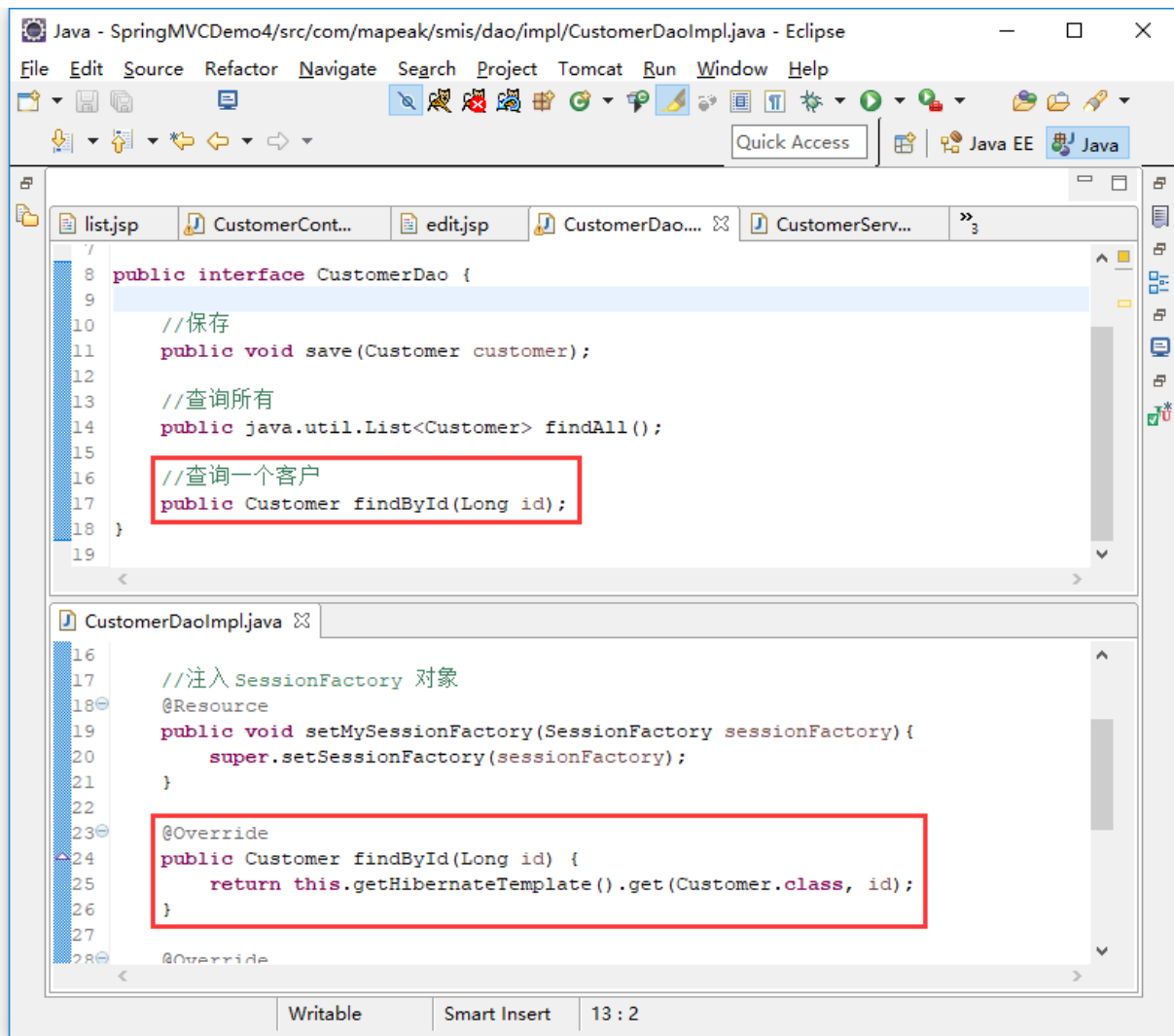
27     return "save";
28 }
29
30 //注入 CustomerService 对象
31 @Resource
32 private CustomerService customerService;
33
34 //查询一个客户
35 //使用 restful 风格访问
36 //返回的 id 值, 获取 url 中的 id 值
37 @RequestMapping(value="/{id}", method=RequestMethod.GET)
38 public String findById(@PathVariable("id")Long id, Map<String, Object> model)
39 {
40     //根据返回的 id 查询数据, 将查询的数据封装到 Map 集合
41     Customer customer = this.customerService.findById(id);
42     model.put("cust", customer);
43
44     //跳转页面
45     return "edit";
46 }
47
48 //查询所有客户
49 @RequestMapping(method=RequestMethod.GET)
50 public String list(Model model){
51     //获取数据
52     java.util.List<Customer> stuList = customerService.findAll();
53     //将数据封装到 Model 中
54     model.addAttribute("custList", stuList);
55     //获取数据后页面跳转
56     return "list";
57 }
```

Writable Smart Insert 28 : 7

Service:



Dao:



测试:

数据正确回显, 请求的路径中带 id 值

Java - SpringMVCDemo4/WebContext/WEB-INF/jsp/list.jsp - Eclipse

File Edit Source Refactor Navigate Search Project Tomcat Run Window Help

list.jsp CustomerCont... edit.jsp CustomerDao... CustomerServ...

```
9 </head>
10 <body>
11 <h3>客户所有数据</h3>
12 <table border="1" width="400px;">
13 <tr>
14 <th>客户名称</th>
15 <th>客户电话</th>
16 <th>操作</th>
17 </tr>
18 <c:forEach items="${custList}" var="cust">
19 <tr>
20 <td>${cust.name}</td>
21 <td>${cust.telephone}</td>
22 <!-- 带着 id 返回 -->
23 <!-- restful 风格可以在路径上加上参数 -->
24 <td><a href=
25 "${pageContext.request.contextPath}/customer/${cust.id}.action">修改</a>
26 </td>
</tr>
```

查询所有客户

localhost:8080/mvc/customer.action

客户所有数据

客户名称	客户电话
白世镜	13290750669
天竺僧人	15823034527

Insert title here

localhost:8080/mvc/customer/1.action

客户修改页面

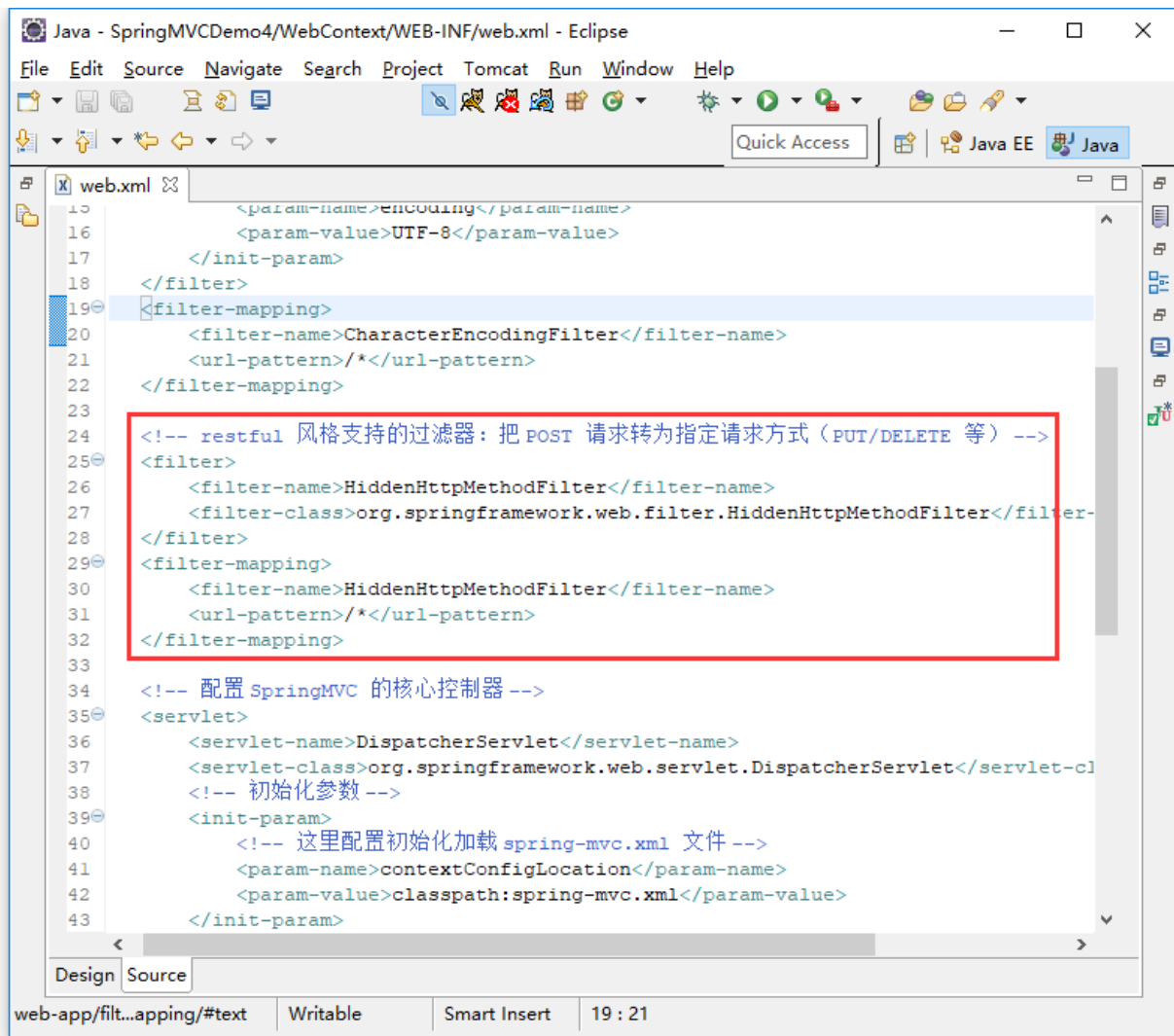
客户名称:

电话:

客户数据保存更新

配置请求方法过滤器

作用: 把表单的 post 请求转换为 put 请求



页面：

PS:

由于 Spring 不支持 put 和 delete 的传输方式，所以这里需要传输方式的转换，将 POST 方法转换成 PUT 方法

```
1 page language="java" contentType="text/html; charset=UTF-8"
2 pageEncoding="UTF-8"%>
3 OCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/1
4<html>
5<ad>
6 ta http-equiv="Content-Type" content="text/html; charset=UTF-8">
7 tle>Insert title here</title>
8 ead>
9<body>
10 <h3>客户修改页面</h3>
11<form action="${pageContext.request.contextPath}/customer.action" method="post">
12
13     <!-- 隐藏域 -->
14     <input type="hidden" name="id" value="${cust.id}" />
15
16     <!-- 由于 Spring 不支持 put 和 delete 的传输方式，所以这里需要传输方式的转换 -->
17     <!-- 将 POST 方法转换成 PUT 方法 -->
18     <input type="hidden" name="_method" value="put" />
19
20     <!-- 返回的数据展示 -->
21     客户名称: <input type="text" name="name" value="${cust.name}" /><br />
22     电话: <input type="text" name="telephone" value="${cust.telephone}" /><br />
23     <input type="submit" value="修改" />
24 </form>
25 </body>
26 </html>
27
```

Controller:

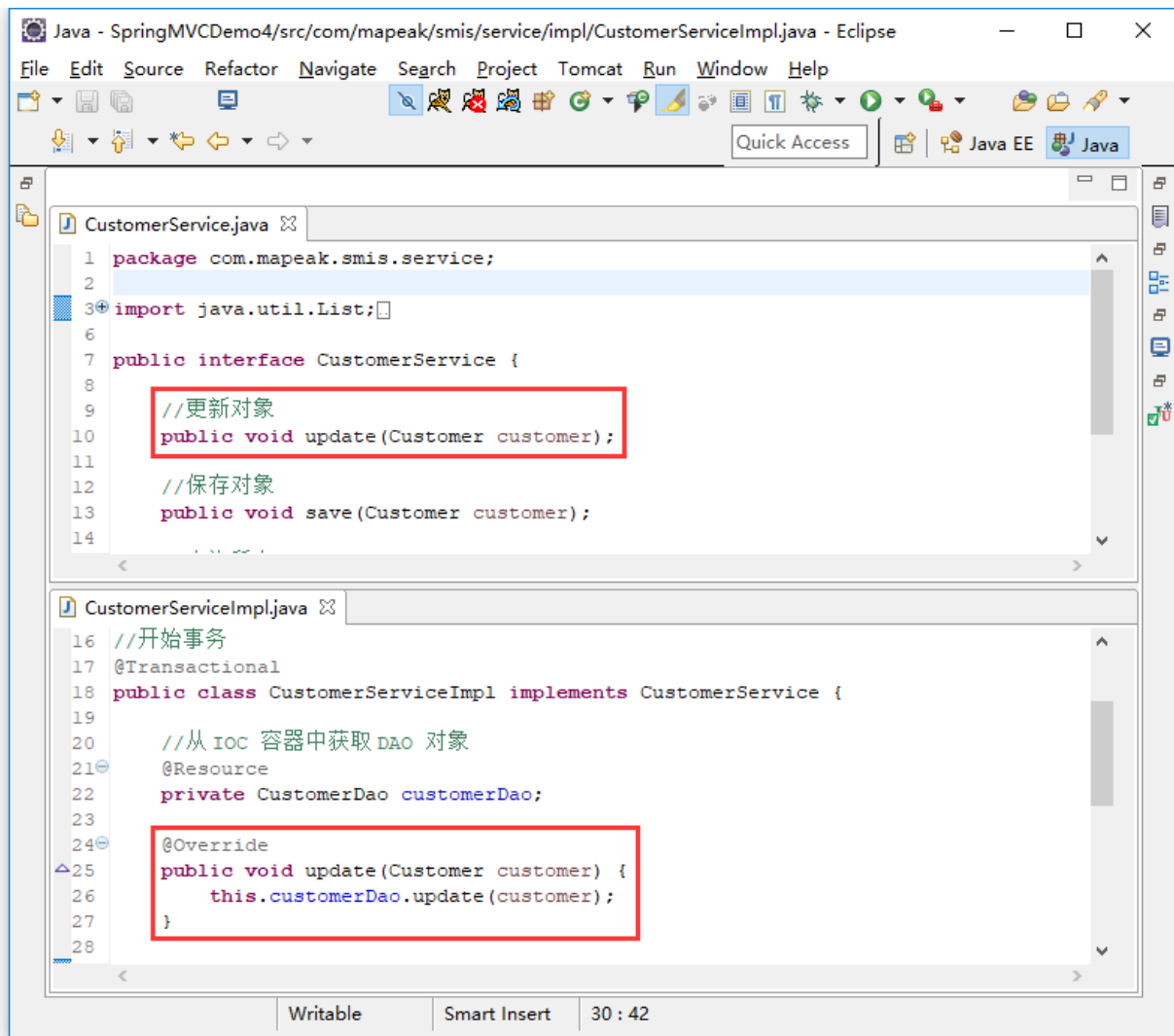
以 restful 方式执行 UPDATE 操作, 即: `method=RequestMethod.PUT`

PS: PUT、DELETE 方式必须加上 `@ResponseBody` 注解, 不然会报错

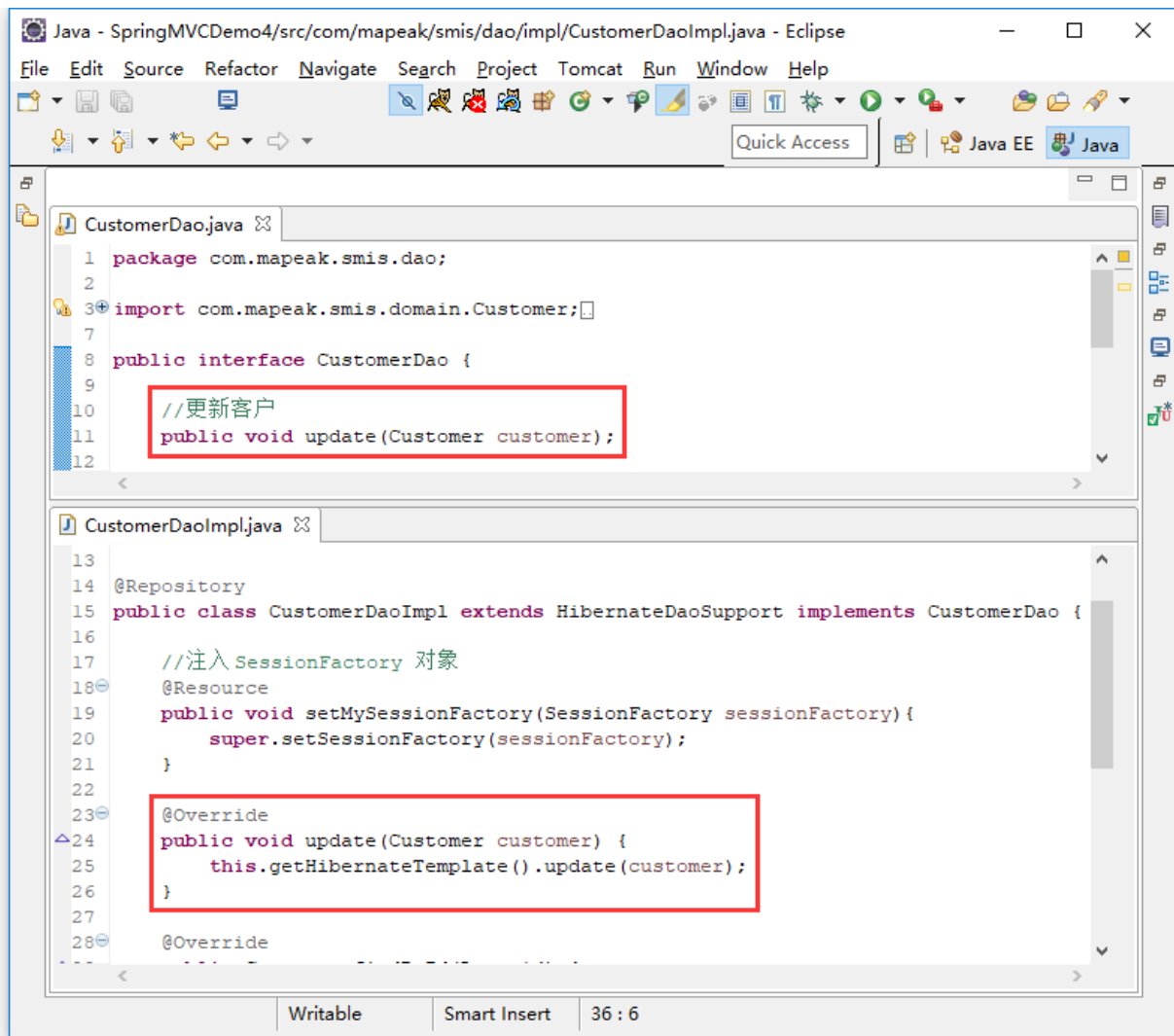
```
Java - SpringMVCDemo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse
File Edit Source Refactor Navigate Search Project Tomcat Run Window Help
Quick Access Java EE Java

CustomerController.java
27 @RequestMapping("/saveUI")
28 public String saveUI(){
29     return "save";
30 }
31
32 //注入 CustomerService 对象
33 @Resource
34 private CustomerService customerService;
35
36 //修改客户
37 //将前端返回的数据封装到 JavaBean 对象
38 //restful 风格, PUT 请求方式, 即为: UPDATE
39 //PUT、DELETE 方式必须加上 @ResponseBody 注解, 否则会报错
40 @ResponseBody
41 @RequestMapping(method=RequestMethod.PUT)
42 public String Update(Customer customer, Model model){
43
44     //更新数据
45     this.customerService.update(customer);
46     //返回处理结果
47     model.addAttribute("msg", "修改成功");
48
49     //处理后跳转页面
50     return "success";
51 }
52
53 //查询一个客户
54 //使用 restful 风格访问
55 //返回的 id 值, 获取 url 中的 id 值
56 @RequestMapping(value="/{id}", method=RequestMethod.GET)
```

Service:



Dao:



测试:

Java - SpringMVC Demo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse

File Edit Source Refactor Navigate Search Project Tomcat Run Window Help

Quick Access Java EE Java

CustomerController.java

```
30 }
31
32 //注入 CustomerService 对象
33 @Resource
34 private CustomerService customerService;
35
36 //修改客户
37 //将前端返回的数据封装到 JavaBean 对象
38 //restful 风格，PUT 请求方式，即为：UPDATE
39 //PUT、DELETE 方式必须加上 @ResponseBody 注解，否则会报错
40 @ResponseBody
41 @RequestMapping(method=RequestMethod.PUT)
42 public String Update(Customer customer, Model model){
43
44     //更新数据
45     this.customerService.update(customer);
46     //返回处理结果
47     model.addAttribute("msg", "修改成功");
48 }
```

Console

D:\JDK\jdk1.8.0_101\bin\javaw.exe (2018年12月27日 下午7:56:13)

Hibernate: select this_.cust_id as cust_id1_0

Hibernate: select this_.cust_id as cust_id1_0

查询所有客户

localhost:8080/mvc/customer.action

客户所有数据

客户名称	客户电话
白世镜	13290750669
天竺僧人	15823034527

查询所有客户

localhost:8080/mvc/customer.action

客户所有数据

客户名称	客户电话	操作
白世镜2	134444455555	修改
天竺僧人	15823034527	修改

客户批量删除功能

页面:

PS: checkbox 的 name 必须与 Controller 的 delete 方法参数名称一致


```
7 ta http-equiv="Content-Type" content="text/html; charset=UTF-8">
8 tle>查询所有客户</title>
9 ead>
10 dy>
11 <h3>客户所有数据</h3>
12 <a href="${pageContext.request.contextPath}/customer/saveUI.action">添加客户</a>
13 <!-- restful 风格 -->
14 <form action="${pageContext.request.contextPath}/customer.action"
15     method="post">
16     <!-- 提交方法交换，隐藏域 -->
17     <input type="hidden" name="_method" value="delete" />
18
19     <table border="1" width="400px;">
20         <tr>
21             <th>选择</th>
22             <th>客户名称</th>
23             <th>客户电话</th>
24             <th>操作</th>
25         </tr>
26         <c:forEach items="${custList}" var="cust">
27             <tr>
28                 <td><input type="checkbox" name="ids" value="${cust.id}" /></td>
29                 <td>${cust.name}</td>
30                 <td>${cust.telephone}</td>
31                 <td><!-- 带着 id 返回 -->
32                     <!-- restful 风格可以在路径上加上参数 -->
33                     <a href="${pageContext.request.contextPath}/customer/${cust.id}">
34                         </td>
35             </tr>
36         </c:forEach>
37     </table>
38     <input type="submit" value="删除" />
39 </form>
40
41 ody>
42 tml>
```

Controller:

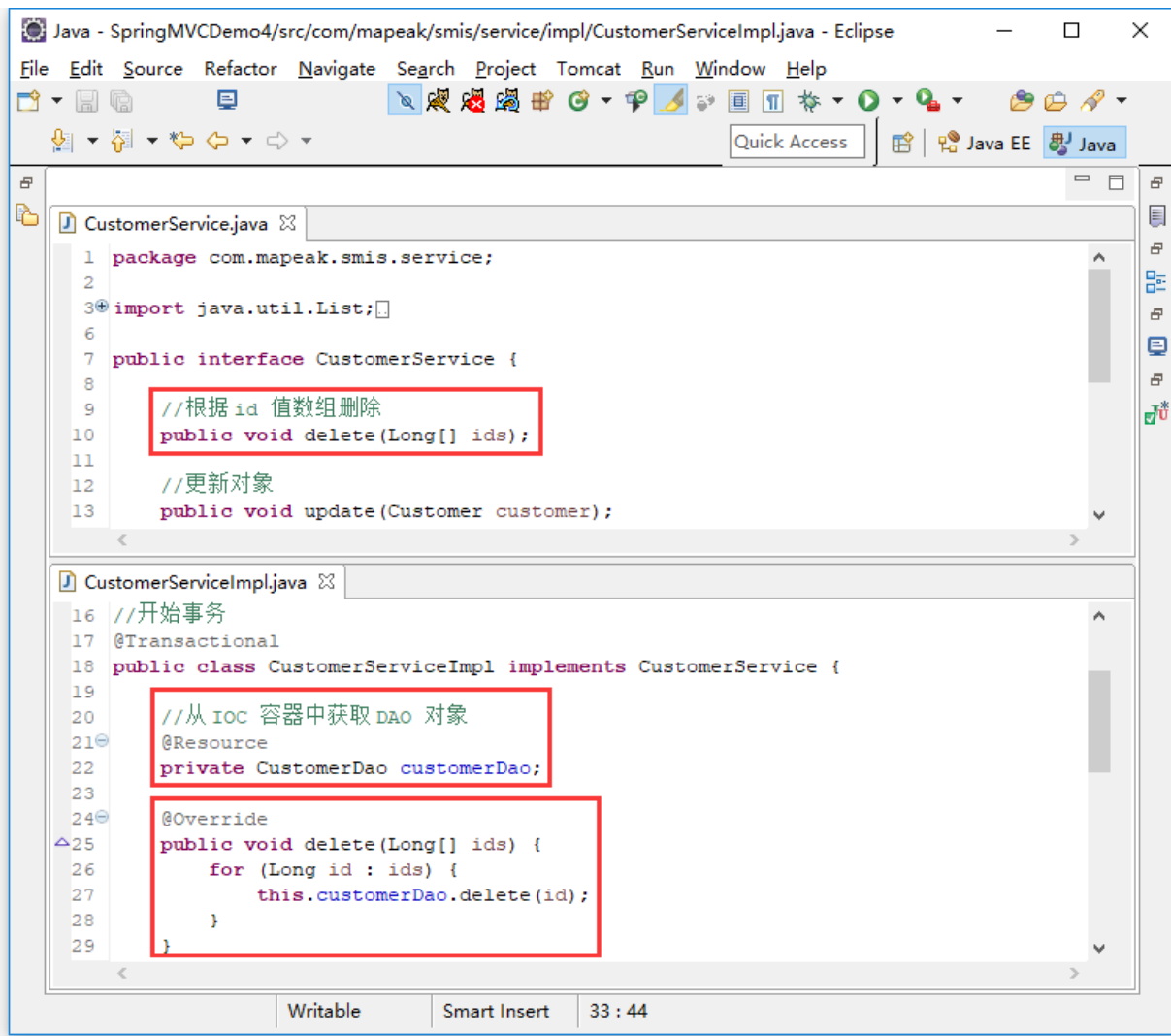
接收页面传过来的 id 集合，名称必须一致

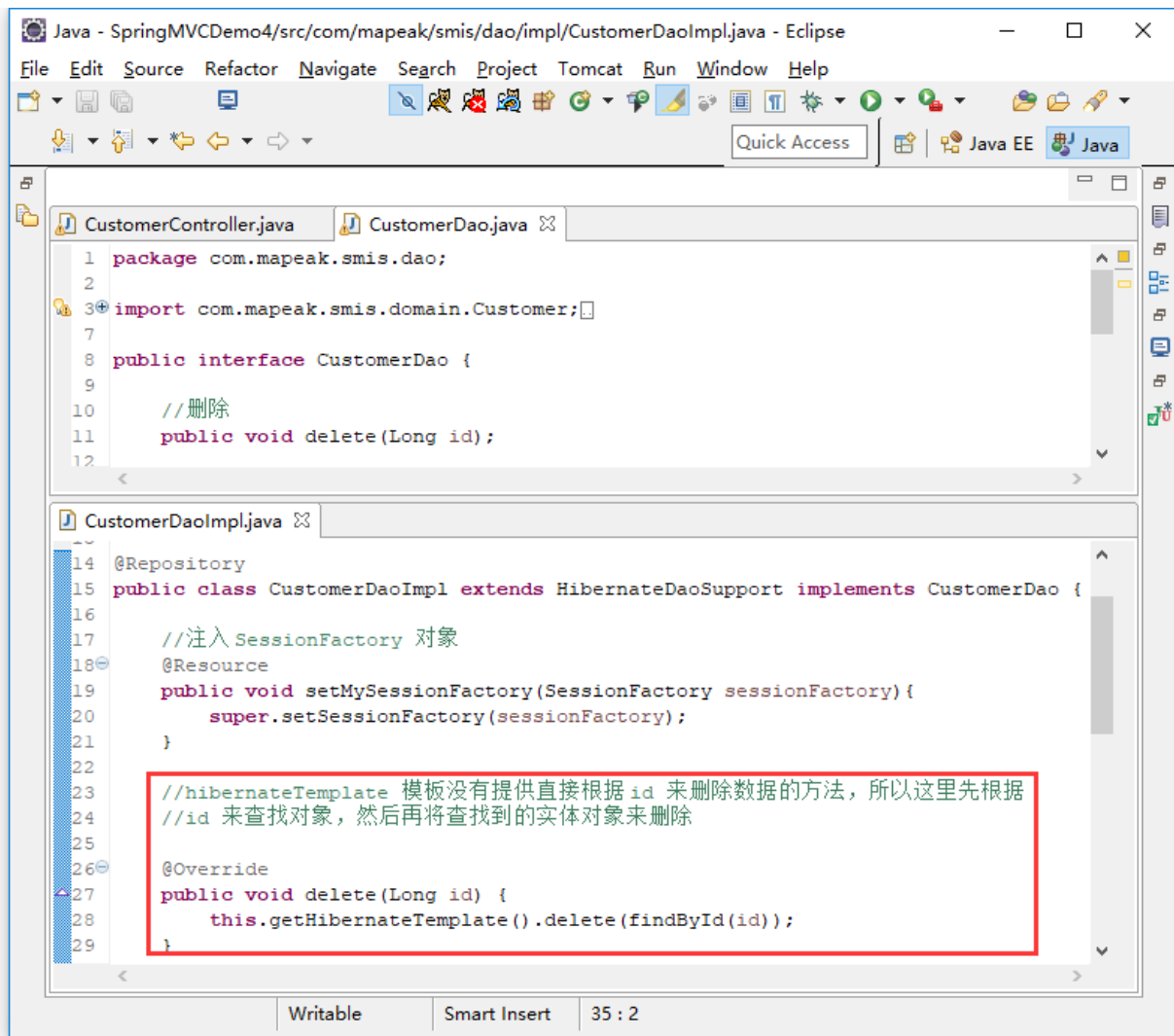
restful 风格，DELETE 请求方式，即为：DELETE

PUT、DELETE 方式必须加上 **@ResponseBody** 注解，不然会报错

```
21
22 @Controller
23 @RequestMapping("/customer")
24 public class CustomerController {
25
26     //因为 save.jsp 页面在 WEB-INF 文件夹下，外部（浏览器）无法直接访问
27     //用于跳转到 WEB-INF/jsp/save.jsp 页面
28     @RequestMapping("/saveUI")
29     public String saveUI() {
30         return "save";
31     }
32
33     //注入 CustomerService 对象
34     @Resource
35     private CustomerService customerService;
36
37     //删除客户
38     //接收页面传过来的 id 集合，名称必须一致
39     //restful 风格，DELETE 请求方式，即为：DELETE
40     //PUT、DELETE 方式必须加上 @ResponseBody 注解，否则会报错
41     @ResponseBody
42     @RequestMapping(method=RequestMethod.DELETE)
43     public String delete(Long[] ids, Map<String, Object> model) {
44         this.customerService.delete(ids);
45         model.put("msg", "删除成功");
46         return "success";
47     }
48
49
50     //修改客户
```

Service:





测试:

Java - SpringMVC Demo4/src/com/mapeak/smis/controller/CustomerController.java - Eclipse

File Edit Source Refactor Navigate Search Project Tomcat Run Window Help

Quick Access Java EE Java

CustomerController.java save.jsp edit.js Console

```
33 //注入 CustomerService 对象
34 @Resource
35 private CustomerService customerService;
36
37 //删除客户
38 //接收页面传过来的 id 集合，名称必须
39 //restful 风格，DELETE 请求方式，同时
40 //PUT、DELETE 方式必须加上 @ResponseBody
41 @ResponseBody
42 @RequestMapping(method=RequestMethod.DELETE)
43 public String delete(Long[] ids, Model model) {
44     this.customerService.delete(ids);
45     model.addAttribute("msg", "删除成功");
46     return "success";
47 }
48
```

D:\JDK\jdk1.8.0_101\bin\javaw.exe (2018年12月27日 下午8:44:15)

Hibernate: select customer0_.cust_id as cust_id from t_customer where cust_id in (?, ?, ?)

Hibernate: select customer0_.cust_id as cust_id from t_customer where cust_id in (?, ?, ?)

Hibernate: delete from t_customer where cust_id in (?, ?, ?)

Hibernate: delete from t_customer where cust_id in (?, ?, ?)

Hibernate: select this_.cust_id as cust_id1_0 from t_customer where cust_id in (?, ?, ?)

查询所有客户 x +

← → ↺ local:localhost:8080/mvc/customer.action

客户所有数据

[添加客户](#)

选择	客户名称	客户电话
<input type="checkbox"/>	白世镜2	13444445555
<input type="checkbox"/>	天竺僧人	15823034527
<input type="checkbox"/>	静智大师	15598563215

删除

查询所有客户 x +

← → ↺ local:localhost:8080/mvc/customer.action

客户所有数据

[添加客户](#)

选择	客户名称	客户电话	操作
<input type="checkbox"/>	白世镜2	13444445555	修改
<input type="checkbox"/>	天竺僧人	15823034527	修改