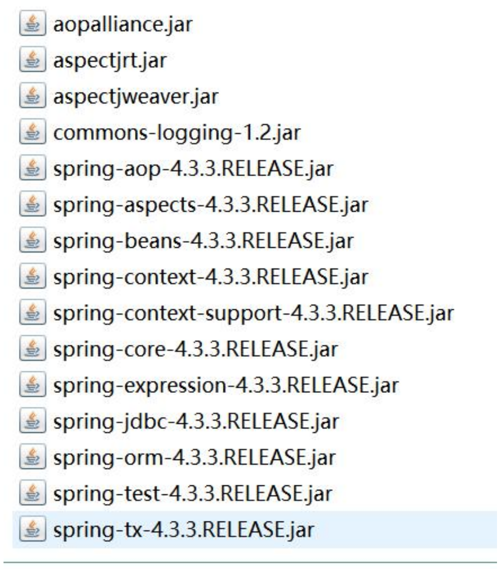
本章学习目标

- Spring 整合 Hibernate
- 整合 SpringMVC 完成客户添加功能
- 客户查询功能
- 客户修改功能
- 客户删除功能

1. Spring 整合 Hibernate

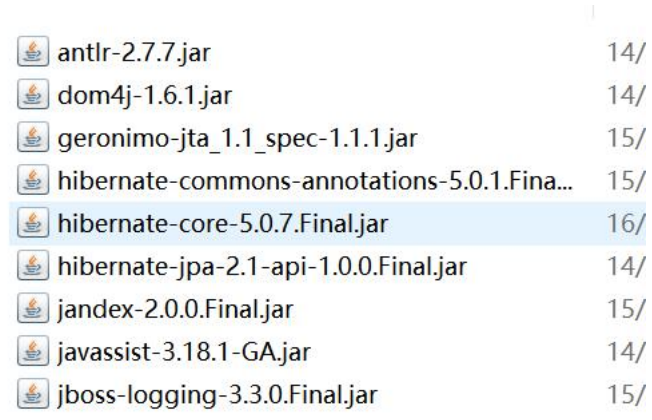
1.1. 导入 hibernate 和 Spring 相关 jar 包

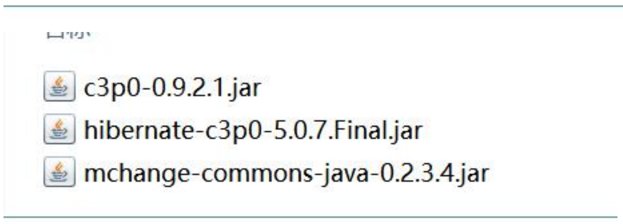
spring 包

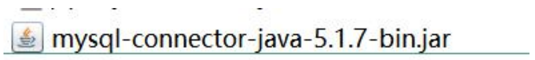


```
aopalliance.jar
aspectjrt.jar
aspectjweaver.jar
commons-logging-1.2.jar
spring-aop-4.3.3.RELEASE.jar
spring-aspects-4.3.3.RELEASE.jar
spring-beans-4.3.3.RELEASE.jar
spring-context-4.3.3.RELEASE.jar
spring-context-support-4.3.3.RELEASE.jar
spring-core-4.3.3.RELEASE.jar
spring-expression-4.3.3.RELEASE.jar
spring-jdbc-4.3.3.RELEASE.jar
spring-orm-4.3.3.RELEASE.jar
spring-test-4.3.3.RELEASE.jar
spring-tx-4.3.3.RELEASE.jar
```

hibernate 包


antlr-2.7.7.jar 14/
dom4j-1.6.1.jar 14/
geronimo-jta_1.1_spec-1.1.1.jar 15/
hibernate-commons-annotations-5.0.1.Fina... 15/
hibernate-core-5.0.7.Final.jar 16/
hibernate-jpa-2.1-api-1.0.0.Final.jar 14/
jandex-2.0.0.Final.jar 15/
javassist-3.18.1-GA.jar 14/
jboss-logging-3.3.0.Final.jar 15/


c3p0-0.9.2.1.jar
hibernate-c3p0-5.0.7.Final.jar
mchange-commons-java-0.2.3.4.jar


mysql-connector-java-5.1.7-bin.jar

1.2. 编写实体类

```
@Entity
@Table(name="t_customer")
public class Customer implements Serializable{

    @Id
    @GeneratedValue(strategy=GenerationType.IDENTITY)
    @Column(name="cust_id")
    private Long custId;
    @Column(name="cust_name")
    private String custName;
    @Column(name="cust_telephone")
    private String custTelephone;
    public Long getCustId() {
        return custId;
    }
    public void setCustId(Long custId) {
        this.custId = custId;
    }
    public String getCustName() {
        return custName;
    }
    public void setCustName(String custName) {
        this.custName = custName;
    }
    public String getCustTelephone() {
        return custTelephone;
    }
    public void setCustTelephone(String custTelephone) {
        this.custTelephone = custTelephone;
    }



}
```

1.3. 配置 applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:contenxt="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
         http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd">

    <!-- 加载 jdbc.properties -->
    <contenxt:property-placeholder
location="classpath:jdbc.properties"/>

    <!-- 创建 c3p0 连接池 -->
    <bean id="dataSource"
class="com.mchange.v2.c3p0.ComboPooledDataSource"
destroy-method="close">
        <property name="jdbcUrl" value="${jdbc.url}"/>
        <property name="driverClass" value="${jdbc.driver_class}"/>
        <property name="user" value="${jdbc.user}"/>
        <property name="password" value="${jdbc.password}"/>
    </bean>

    <!-- Spring 整合 Hibernate -->
    <bean id="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <property name="dataSource" ref="dataSource"/>
        <property name="hibernateProperties">
            <props>
                <!-- 是否输出 sql 语句 -->
                <prop key="hibernate.show_sql">true</prop>
                <!-- 自动维护表 -->
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
        <property name="packagesToScan">
            <!-- 扫描实体所在的包 -->
            <list>
                <value>cn.sm1234.domain</value>
            </list>
        </property>
    </bean>

    <!-- 开启 Spring 的事务管理 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>
    <bean id="transactionManager"
class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactory" ref="sessionFactory"/>
    </bean>

    <!-- 开启 Spring 注解扫描 -->
    <contenxt:component-scan
base-package="cn.sm1234.service,cn.s1234.dao"/>

</beans>
```

1.4. 编写 Dao 接口和实现

Dao 接口：

```
public interface CustomerDao {
    public void save(Customer customer);
}
```

Dao 实现类：

```
@Repository
public class CustomerDaoImpl extends HibernateDaoSupport implements
CustomerDao {

    @Resource
    public void setMySessionFactory(SessionFactory sessionFactory){
        super.setSessionFactory(sessionFactory);
    }


    @Override
    public void save(Customer customer) {
        this.getHibernateTemplate().save(customer);
    }

}
```

1.5. 编写 Service 接口和实现

Service 接口：

```
public interface CustomerService {

    public void save(Customer customer);
}
```

Service 实现：

```
@Service
@Transactional
public class CustomerServiceImpl implements CustomerService {

    @Resource
    private CustomerDao customerDao;

    @Override
    public void save(Customer customer) {
        customerDao.save(customer);
    }

}
```

1.6. 测试 Service 方法

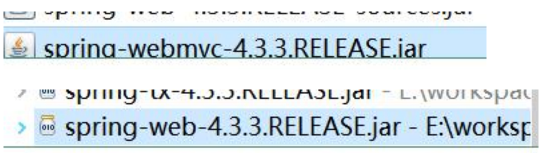
```
public class CustomerServiceTest {

    public static void main(String[] args) {
        Customer c = new Customer();
        c.setCustName("张三");
        c.setCustTelephone("13233334444");

        ApplicationContext ac = new
ClassPathXmlApplicationContext("spring.xml");
        CustomerService customerService =
(CustomerService)ac.getBean("customerServiceImpl");
        customerService.save(c);
    }
}
```

2. 整合 SpringMVC 完成客户添加功能

2.1. 导入 SpringMVC 支持



spring-webmvc-4.3.3.RELEASE.jar

spring-tx-4.3.3.RELEASE.jar - E:\workspac

spring-web-4.3.3.RELEASE.jar - E:\worksp

2.2. 编写 web.xml 启动 SpringMVC 和 Spring

```
 <!-- SpringMVC 的核心控制器 -->
  <servlet>
    <servlet-name>DispatcherServlet</servlet-name>
    
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</se
rvlet-class>
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:spring-mvc.xml</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>DispatcherServlet</servlet-name>
    <url-pattern>*.action</url-pattern>
  </servlet-mapping>
```

```
<!-- Spring 启动监听器 -->
  <listener>

    <listener-class>org.springframework.web.context.ContextLoaderListene
r</listener-class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:spring.xml</param-value>
  </context-param>
```

2.3. 编写 SpringMVC 配置

spring-mvc.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:contenxt="http://www.springframework.org/schema/context"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
         http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx.xsd">
    <!-- 扫描 Controller 包 -->
    <contenxt:component-scan base-package="cn.sm1234.controller"/>

    <!-- 开启注映射器和适配器 -->
    <mvc:annotation-driven></mvc:annotation-driven>

    <!-- 视图解析器 -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolve
r">
        <property name="prefix" value="/WEB-INF/jsp/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
</beans>
```

2.4. save.jsp 添加客户页面

```
 <form action="${pageContext.request.contextPath}/customer/save.action"
method="post">
    客户名称：<input type="text" name="custName"/><br/>
    客户电话：<input type="text" name="custTelephone"/><br/>
    <input type="submit" value="添加">
  </form>
```

2.5. CustomerController

```
@Controller
@RequestMapping("/customer")
public class CustomerController {

    /**
     * 用于跳转到 save.jsp 页面
     */
    @RequestMapping("/saveUI")
    public String saveUI(){
        return "save";
    }

    @Resource
    private CustomerService customerService;

    /**
     * 添加客户
     */
    @RequestMapping(method=RequestMethod.POST)
    public String save(Customer customer,Map<String,Object> model){
        customerService.save(customer);
        //保存提示信息
        model.put("msg", "添加成功");
        return "success";
    }
}
```

2.6. 添加 Spring 的编码过滤器

web.xml

```
 <filter>
    <filter-name>CharacterEncodingFilter</filter-name>

    <filter-class>org.springframework.web.filter.CharacterEncodingFilter
</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

3. 客户查询功能

3.1. Dao

接口：

```
public List<Customer> findAll();
```

实现：

```
@Override
    public List<Customer> findAll() {
        return this.getHibernateTemplate().loadAll(Customer.class);
    }
```

3.2. Service

接口：

```
public List<Customer> findAll();
```

实现：

```
@Override
    public List<Customer> findAll() {
        return customerDao.findAll();
    }
```

3.3. Controller

```
/**
     * 查询所有客户
     */
    @RequestMapping(method=RequestMethod.GET)
    public String list(Map<String,Object> model){
        List<Customer> custList = customerService.findAll();
        model.put("custList", custList);
        return "list";
    }
```

3.4. 页面

```
<h3>客户所有数据</h3>
```

```
<table border="1" width="400px;">
 <tr>
      <th>客户名称</th>
      <th>客户电话</th>
      <th>操作</th>
 </tr>
 <c:forEach items="${custList}" var="cust">
 <tr>
      <td>${cust.custName }</td>
      <td>${cust.custTelephone }</td>
      <td><a href="">修改</a></td>
 </tr>
 </c:forEach>
</table>
```

4. 客户修改功能

4.1. 客户数据的回显

4.1.1. 页面

```
  <tr>
        <td>${cust.custName }</td>
        <td>${cust.custTelephone }</td>
        <td><a
href="${pageContext.request.contextPath}/customer/${cust.custId}.action
">修改</a></td>
   </tr>
```

edit.jsp

```
 <form action="${pageContext.request.contextPath}/customer.action"
method="post">
    <input type="hidden" name="custId" value="${cust.custId}">
    客户名称：<input type="text" name="custName"
value="${cust.custName}"/><br/>
    客户电话：<input type="text" name="custTelephone"
value="${cust.custTelephone }"/><br/>
    <input type="submit" value="修改">
  </form>
```

4.1.2. Controller

```
/**
     * 查询一个客户
     */
    @RequestMapping(value="/{id}",method=RequestMethod.GET)
    public String findById(@PathVariable("id")Long
custId,Map<String,Object> model){
        Customer cust = customerService.findById(custId);
        model.put("cust", cust);
        return "edit";
    }
```

4.1.3. Service

接口：

```
public Customer findById(Long custId);
```

实现：

```
@Override
    public Customer findById(Long custId) {
        return customerDao.findById(custId);
    }
```

4.1.4. Dao

接口：

```
public Customer findById(Long custId);
```

实现：

```
@Override
    public Customer findById(Long custId) {
        return this.getHibernateTemplate().get(Customer.class, custId);
    }
```

4.2. 保存修改的客户数据

4.2.1. 配置请求方法过滤器

作用：就是把表单的 post 请求转换为 put 请求

```
 <!-- 请求方法的过滤器 -->
  <filter>
    <filter-name>HiddenHttpMethodFilter</filter-name>


    <filter-class>org.springframework.web.filter.HiddenHttpMethodFilter<
```

```
/filter-class>
  </filter>
  <filter-mapping>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

4.2.2. 页面

```
<form action="${pageContext.request.contextPath}/customer.action"
method="post">
    <input type="hidden" name="_method" value="put"/>
    <input type="hidden" name="custId" value="${cust.custId}">
    客户名称：<input type="text" name="custName"
value="${cust.custName}"/><br/>
    客户电话：<input type="text" name="custTelephone"
value="${cust.custTelephone }"/><br/>
    <input type="submit" value="修改">
  </form>
```

4.2.3. Controller

```
    /**
     * 客户修改
     */
    @RequestMapping(method=RequestMethod.PUT)
    public String update(Customer customer,Map<String,Object> model){
        customerService.update(customer);
        model.put("msg", "修改成功");

        return "success";

    }
```

4.2.4. Service

接口：

```
public void update(Customer customer);
```

实现

```
@Override

    public void update(Customer customer) {

        customerDao.update(customer);

    }
```

4.2.5. Dao

接口：

```
public void update(Customer customer);
```

实现：

```
@Override

    public void update(Customer customer) {

        this.getHibernateTemplate().update(customer);

    }
```

5. 客户删除功能

5.1.1. 页面

```
 <h3>客户所有数据</h3>
  <a href="${pageContext.request.contextPath}/customer/saveUI.action">添
加客户</a>
  <form action="${pageContext.request.contextPath}/customer.action"
method="post">
    <input type="hidden" name="_method" value="delete"/>
  <table border="1" width="400px;">
   <tr>
        <th>选择</th>
        <th>客户名称</th>
        <th>客户电话</th>
        <th>操作</th>
   </tr>
   <c:forEach items="${custList}" var="cust">
   <tr>
        <td><input type="checkbox" name="custIds"
value="${cust.custId}"/></td>
        <td>${cust.custName }</td>
        <td>${cust.custTelephone }</td>
        <td><a
href="${pageContext.request.contextPath}/customer/${cust.custId}.action
">修改</a></td>
   </tr>
   </c:forEach>
  </table>
   <input type="submit" value="删除"/>
  </form>
```

5.1.2. Controller

```
/**
     * 删除客户
     */
    @RequestMapping(method=RequestMethod.DELETE)
    public String delete(Long[] custIds,Map<String,Object> model){
        customerService.delete(custIds);
        model.put("msg", "删除成功");
        return "success";
    }
```

5.1.3. Service

接口：

```
public void delete(Long[] custIds);
```

实现：

```
@Override
    public void delete(Long[] custIds) {
        if(custIds!=null){
            for (Long custId : custIds) {
                customerDao.delete(custId);
            }
        }
    }
```

5.1.4. Dao

接口：

```
public void delete(Long custId);
```

实现：

```
@Override
    public void delete(Long custId) {
        this.getHibernateTemplate().delete(findById(custId));
    }
```