|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 课程名称 | 基于JAVAEE的高级程序设计 | | | | |
| 实验名称 | SSM框架整合 | | | | |
| 实验时间 |  | | 实验地点 |  | |
| 学 院 | 信息科学与工程学院 | | 专 业 | 计算机科学与技术 | |
| 姓 名 | 蔡涛 | 班 级 | 计科1702 | 学 号 | 20170609037 |
| 同组实验者  姓名 |  | | | | |
| 实验成绩 |  | | 指导教师  （签字） |  | |
| **实验报告内容基本要求参考格式**  一、实验目的  二、实验环境  三、实验步骤/过程  四、实验结果  五、实验分析及反馈 | | | | | |
| 一、实验目的   1. 框架的整合思路 2. 整合所需JAR包介绍 3. 编写配置文件 4. 整合应用测试   二、实验环境  Eclipse、MySQL  三、实验步骤/过程   1. 在Eclipse中使用Maven创建ch17的Web项目，在pom.xml中配置三个框架的依赖，拷贝三个框架的配置文件。 2. 在src目录下，创建一个cn.edu.ujn.ch17.po包，并在包中创建持久化类Customer。 3. 在src目录下，创建一个cn.edu.ujn.ch17.dao包，并在包中创建接口文件CustomerDao以及对应的映射文件CustomerDao.xml | | | | | |

|  |
| --- |
| 1. 在src目录下，创建一个cn.edu.ujn.ch17.service包，然后在包中创建接口文件CustomerService，并在ICustomerService中定义通过id查询客户的方法。   public Customer findCustomerById(Integer id);   1. 在src目录下，创建一个cn.edu.ujn.ch17.service.impl包，然后在包中创建CustomerService的接口实现类，CustomerServiceImpl。   关键代码：  @Override  public int addCustomer(Customer customer) {  int insert = this.customerDao.insertSelective(customer);  return insert;  }   1. 在src目录下，创建一个cn.edu.ujn.ch17.controller包，然后在包中创建用于创建用于处理页面请求的控制器类CustomerController。其中findCustomerById()方法根据id查询客户详情。   关键代码：  @RequestMapping("/findById")  public String findCustomerById(Integer id, Model model) {  Customer customer = this.customerService.findCustomerById(id);  model.addAttribute("customer", customer);  return "customer";  }   1. 在WEN-INF目录下，创建一个jsp文件夹，在其中创建一个用于显示客户详情的页面文件customer.jsp，页面表格通过EL表达式来获取后台控制层返回的客户信息。   关键代码：  <tr>  <td>${customer.id}</td>  <td>${customer.username}</td>  <td>${customer.jobs}</td>  <td>${customer.phone}</td>  </tr>   1. 将项目发布到Tomcat并启动，在浏览器中访问地址：   http://localhost:9090/ch17/findById?id=1   1. 在ICustomerService中定义通过列表查询客户的方法。   关键代码：  public List<Customer> findAllCustomer();   1. 在CustomerServiceImpl类中实现findAllCustomer()方法。   关键代码：  @Override  public List<Customer> findAllCustomer() {  return this.customerDao.findAllCustomer();  } |

|  |
| --- |
| 1. 在CustomerMapper接口中定义列表查询的方法。   关键代码  List<Customer> findAllCustomer();   1. 在CustomerMapper.xml文件中添加列表查询语句：   关键代码：  <select id="findAllCustomer" resultMap="BaseResultMap">  select \*  from customer  </select>   1. 控制器类CustomerController。其中findAllCustomer ()方法列表查询客户详情。   关键代码：  @RequestMapping("/findAll")  public String findAllCustomer(Customer customer, Model model) {  List<Customer> customer3 = this.customerService.findAllCustomer();  model.addAttribute("list", customer3);  return "listCustomer";  }   1. WEN-INF目录下jsp文件夹，在其中创建一个用于显示客户详情的页面文件listCustomer.jsp，页面表格通过EL表达式来获取后台控制层返回的客户信息。   关键代码：  <c:forEach items="${list}" var="row">  <tr>  <td>${row.id}</td>  <td>${row.username}</td>  <td>${row.jobs}</td>  <td>${row.phone}</td>  </tr>  </c:forEach>   1. 在浏览器中访问地址：   http://localhost:9090/ch17/findAll   1. 在ICustomerService中定义添加客户的方法。   关键代码：  public int addCustomer(Customer customer);   1. 在CustomerServiceImpl类中实现addCustomer ()方法。   关键代码：  @Override  public int addCustomer(Customer customer) {  int insert = this.customerDao.insertSelective(customer);  return insert;  } |

|  |
| --- |
| 1. 控制器类CustomerController。其中addCustomer ()方法添加客户详情。   关键代码：  @RequestMapping("/add")  public String add() {  return "addCustomer";  }  @RequestMapping("/addCustomer")  public String addCustomer(Customer customer, Model model) {  int customer2 = this.customerService.addCustomer(customer);  model.addAttribute("customer", customer);  return "customer";  }   1. WEN-INF目录下jsp文件夹，在其中创建一个用于显示客户详情的页面文件addCustomer.jsp。 2. 在浏览器中访问地址：   http://localhost:9090/ch17/add  四、实验结果  添加用户：    添加成功：    数据库显示：    查询ID为6的信息：    列表查询：    五、实验分析及反馈  本次实验将本学期所学的三个框架都整合到了一起，从一个比较全面的角度感受到了三个框架的不同分工和作用。  Spring是一个开源框架，Spring使用基本的JavaBean来完成以前只可能由EJB完成的事情。然而，Spring的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何Java应用都可以从Spring中受益。简单来说，Spring是一个轻量级的控制反转和面向切面的容器框架。  Spring MVC属于Spring。Spring MVC 分离了控制器、模型对象、分派器以及处理程序对象的角色，这种分离让它们更容易进行定制。  MyBatis是一个基于Java的持久层框架，MyBatis 使用简单的XML或注解用于配置和原始映射，将接口和Java的POJO类映射成数据库中的记录。可编写灵活和高性能的sql语句。sql语句与代码分离，存放于xml配置文件中，便于管理。  三个框架的整合，功能变得更加强大，更加具体，同时方便了我们来运用和设计。 |