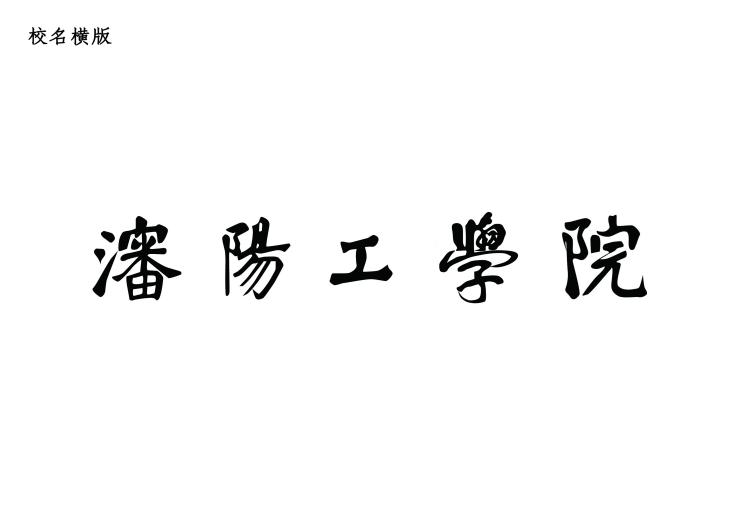
****

****

**企业实训报告**

**学 院： 信息与控制学院**

**专业班级: 计算机科学与技术**

**学 号： 1530110318**

**学生姓名：周天祥**

**成 绩：**

沈阳工学院

**企业实训日志**

**2017年12月18日 　　　星期一**

今天最开始，由大连华信的老师为我们介绍大连华信企业的一些基本情况和从今天开始为期一周的企业实训的基本内容，让我们对课程情况有大概的了解，学习软件开发所需的基本技术和工作的标准流程。使我跳出来学校所学的思维，将编程思想、规范和流程与社会需求接轨。

一开始，我们先讨论需求分析和项目可行性分析确定将本次实训项目的功能确定下来并且确认开发流程和开发时间规定并创建数据库

CREATE DATABASE WBS;

USE WBS;

CREATE TABLE BOOKINFO (

BOOKNAME varchar(50) NOT NULL,

BOOKTYPENAME varchar(50) NOT NULL,

ISBN varchar(50) DEFAULT NULL,

BOOKPRICE DECIMAL(10, 2) NOT NULL,

PUBNAME VARCHAR(50) NOT NULL,

PUBDATE VARCHAR(50) NOT NULL,

AUTHOR VARCHAR(50) NOT NULL,

INTRODUCE VARCHAR(100) NOT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

CREATE TABLE BOOKTYPE (

BOOKTYPEID INT PRIMARY KEY NOT NULL,

BOOKTYPENAME varchar(50) NOT NUL

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

CREATE TABLE BOOKINFO (

cus\_id varchar(255) PRIMARY KEY NOT NULL,

cus\_alias varchar(255) NOT NULL,

cus\_pwd varchar(255) DEFAULT NULL,

cus\_name varchar(255) NOT NULL,

cus\_tel VARCHAR(255) NOT NULL,

cus\_gender VARCHAR(255) NOT NULL,

cus\_birth VARCHAR(255) NOT NULL,

cus\_address VARCHAR(255) NOT NULL,

ulimit int NOT NULL

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

CREATE TABLE Shopping (

s\_bookname varchar(50) NOT NULL,

s\_bookprice decimal(10,2) NOT NULL,

total int NOT NUL

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8;

**2017年12月19日 　　　星期二**

开始搭建S2SH框架环境，导入所需JAR文件，创建网上图书商城项目并调整结构。由于我们是第一次使用Spring框架对相关的配置及操作还不熟悉，所以我在编写applicationContex.xml配置文件时出现了纰漏，导致程序无法连接到数据库，后经过调试成功连接到数据后，开始接下来的工作，创建VO类，DAO类，Service类和Action 类。开始开发实现查看全部图书信息功能，关键代码如下：

DAO类：

public List<BookInfo> findAllBookInfo(){  
 String hql = "from BookInfo";  
 List<BookInfo> list = (List<BookInfo>) this.getHibernateTemplate().find(hql);  
 if (list != null && list.size() > 0) {  
 return list;  
 }  
 return null;  
 }

Service类：

private BookInfoDao bookInfoDao;  
 public BookInfoDao getBookInfoDao() {  
 return bookInfoDao;  
 }  
 public void setBookInfoDao(BookInfoDao bookInfoDao) {  
 this.bookInfoDao = bookInfoDao;  
 }

public List<BookInfo> findAllBookInfo(){  
 return bookInfoDao.findAllBookInfo();  
 }

Action类：

private BookInfo bookInfo = new BookInfo();  
 private BookInfoService bookInfoService;

public void setBookInfoService(BookInfoService bookInfoService) {  
 this.bookInfoService = bookInfoService;  
 }  
 @Override  
 public BookInfo getModel() {return bookInfo;}

public String findAllBook() throws Exception{  
 list = bookInfoService.findAllBookInfo();  
 session.put("bookinfo",list);  
 List<BookInfo> list = bookInfoService.findAllBookInfo();  
 session.put("bookinfo", list);  
 return SUCCESS;  
 }

**2017年12月20日 　　　星期三**

继续完善网上图书商店项目，同时开始编写用户登陆和注册功能，最初进行开发时发现页面上的值传不过来，后来经过查阅资料发现是我在编写Action类时使用了模型驱动进行开发，而这就需要前台页面的取值语句作出相应的改变，经过调试后创建相应的VO类，DAO类，Service类和Action类。同时开始听取老师的建议，采用版本控制技术来对项目进行开发合并。

Dao类：

public Customers login(Customers customers){  
 String sql = "from Customers where cusId = ? and cusPwd = ?";  
 List<Customers> list = (List<Customers>) this.getHibernateTemplate().find(sql, customers.getCusId(), customers.getCusPwd());  
 if (list != null && list.size() > 0) {  
 return list.get(0);  
 } else {  
 return null;  
 }  
 }

@Transactional(value = "transactionManager", rollbackFor = Exception.class)  
 public void register(Customers customers){  
 System.out.printf(customers.getCusId());  
 this.getHibernateTemplate().save(customers);  
 }

Service类

public Customers login(Customers customers){

return customerDao.login(customers);

}

public void register(Customers customers){

customerDao.register(customers);

}

Action类：

public String login() throws Exception {  
 Customers cus = customerService.login(customer);  
 if (cus == null){  
 return INPUT;  
 } else {  
 return SUCCESS;  
 }  
 }

public String register() throws Exception {  
 customerService.register(customer);  
 return SUCCESS;  
 }

**2017年12月21日 　　　星期四**

持续完善网上图书商城项目，由于昨天我们开始采用了Git+Github的版本控制，而我们每个人又不是很熟悉，所以在拉取分支，合并分支上出现了很多问题，项目进度一度被拖慢，后来每个人在开发的过程中逐渐熟悉了这种模式，效率也恢复了正常，同时开始制作向购物车添加图书信息功能，创建相应的VO类，DAO类，Service类和Action类。

DAO类：

public BookInfo findById(String isbn) {  
 List<BookInfo> list = (List<BookInfo>) this.getHibernateTemplate().find("from BookInfo where isbn = ?", isbn);  
 System.out.print(list.size());  
 return list.get(0);  
 }

Service类：

private ShoppingDao shoppingDao;  
 public ShoppingDao getShoppingDao() {  
 return shoppingDao;  
 }  
 public void setShoppingDao(ShoppingDao shoppingDao) {  
 this.shoppingDao = shoppingDao;  
 }

public BookInfo findByISBN (String isbn){  
 return shoppingDao.findById(isbn);  
 }

Action类：

Shopping shopping = new Shopping();  
 private ShoppingService shoppingService;  
 @Override  
 public Shopping getModel(){  
 return shopping;  
 }

public String addCar(){  
 HttpServletRequest request = ServletActionContext.getRequest();  
 HttpSession session = request.getSession();  
 String isbn = request.getParameter("isbn");  
 BookInfo bookInfo = shoppingService.findByISBN(isbn);  
 List<BookInfo> list = null;  
 if (session.getAttribute("shop") == null) {  
 list = new ArrayList<BookInfo>();  
 } else {  
 list = (List<BookInfo>)session.getAttribute("shop");  
 }  
 list.add(bookInfo);  
 session.setAttribute("shop", list);  
 return SUCCESS;  
 }

**2017年12月22日 　　　星期五**

完成网上图书商城基本功能，完善购物车功能，支持从购物车中删除图书商品信息，创建Action类中的删除图书信息方法，同时在jsp页面加入java代码方便循环取值，同时加入计算总价功能。同时今天也是项目审查的时间，在上午老师带领我们将最后一个功能做完之后，我们每个小组都进行了项目的演示和源码的审计工作，结果基本达到我们的预期要求。

trolle.jsp

<%  
 double total = 0;  
 if(session.getAttribute("shop")!=null){  
 List<BookInfo> list=(List<BookInfo>)session.getAttribute("shop");  
 Iterator<BookInfo> it=list.iterator();  
 int i =1;  
 while(it.hasNext()){  
 BookInfo bookInfo= it.next();  
 total += bookInfo.getBookprice();  
 %>  
 <%=bookInfo.getBookname() %>

<%=bookInfo.getIsbn()%>

<%=bookInfo.getAuthor()%>

<%=bookInfo.getBookprice()%>

<%=total%>  
 <%  
 i++;

}  
 }  
 %>

Action类：

private Map<String,Object> session;

@Override  
 public void setSession(Map<String, Object> session) {  
 this.session = session;  
 }  
 public ShoppingService getShoppingService() {  
 return shoppingService;  
 }  
 public void setShoppingService(ShoppingService shoppingService) {  
 this.shoppingService = shoppingService;  
 }

public String delCar(){  
 HttpServletRequest request = ServletActionContext.getRequest();  
 HttpSession session = request.getSession();  
 List<BookInfo> list = (List<BookInfo>)session.getAttribute("shop");  
 String isbn = request.getParameter("isbn");  
 Iterator<BookInfo> it = list.iterator();  
 while (it.hasNext()){  
 BookInfo bookInfo1 = it.next();  
 if (bookInfo1.getIsbn() == isbn){  
 it.remove();  
 }  
 }  
 session.setAttribute("shop", list);  
 return SUCCESS;  
 }

**实训心得**

在为期一周的实训当中，我们接触到很多企业开发的相关规范，以往在学校或者自己做项目的时候，开发流程不规范，代码书写格式不标准等小毛病都在这次的实训中得到了一定的改善，为我们以后做软件项目开发积累经验。

这一周的实训课题是基于S2SH框架开发的一个网站，具备增删改查的功能。这是我们首次使用Spring FrameWork。Spring Framework 是一个开源的[Java](https://zh.wikipedia.org/wiki/Java)／[Java EE](https://zh.wikipedia.org/wiki/Java_EE)全功能栈（full-stack）的[应用程序框架](https://zh.wikipedia.org/wiki/%E8%BD%AF%E4%BB%B6%E6%A1%86%E6%9E%B6)，以[Apache许可证](https://zh.wikipedia.org/wiki/Apache%E8%AE%B8%E5%8F%AF%E8%AF%81)形式发布，也有[.NET](https://zh.wikipedia.org/wiki/.NET)平台上的移植版本。该框架基于 [Expert One-on-One Java EE Design and Development](http://www.wrox.com/WileyCDA/WroxTitle/productCd-0764543857.html)（[ISBN 0-7645-4385-7](https://zh.wikipedia.org/wiki/Special:%E7%BD%91%E7%BB%9C%E4%B9%A6%E6%BA%90/0764543857)）一书中的代码，最初由Rod Johnson和Juergen Hoeller等开发。Spring Framework提供了一个简易的开发方式，这种开发方式，将避免那些可能致使底层代码变得繁杂混乱的大量的属性文件和帮助类。

Spring中包含的关键特性

强大的基于[JavaBeans](https://zh.wikipedia.org/wiki/JavaBeans)的采用[控制反转](https://zh.wikipedia.org/wiki/%E6%8E%A7%E5%88%B6%E5%8F%8D%E8%BD%AC)（Inversion of Control，IoC）原则的配置管理，使得应用程序的组建更加快捷简易。

一个可用于从[applet](https://zh.wikipedia.org/wiki/Applet)到[Java EE](https://zh.wikipedia.org/wiki/Java_EE)等不同运行环境的核心[Bean](https://zh.wikipedia.org/w/index.php?title=Enterprise_Java_Beans&action=edit&redlink=1)[工厂](https://zh.wikipedia.org/w/index.php?title=%E8%99%9A%E6%8B%9F%E5%B7%A5%E5%8E%82%E6%A8%A1%E5%BC%8F&action=edit&redlink=1)。

数据库事务的一般化抽象层，允许宣告式（Declarative）事务管理器，简化事务的划分使之与底层无关。

内建的针对[JTA](https://zh.wikipedia.org/wiki/Java%E4%BA%8B%E5%8A%A1API)和单个[JDBC](https://zh.wikipedia.org/wiki/Java_Database_Connectivity)数据源的一般化策略，使Spring的事务支持不要求[Java EE](https://zh.wikipedia.org/wiki/Java_EE)环境，这与一般的JTA或者[EJB](https://zh.wikipedia.org/wiki/EJB) CMT相反。

JDBC 抽象层提供了有针对性的异常等级（不再从SQL异常中提取原始代码），简化了错误处理，大大减少了程序员的编码量。再次利用JDBC时，你无需再写出另一个'终止'（finally）模块。并且面向JDBC的异常与Spring通用[数据访问对象](https://zh.wikipedia.org/wiki/%E6%95%B0%E6%8D%AE%E8%AE%BF%E9%97%AE%E5%AF%B9%E8%B1%A1)（Data Access Object）异常等级相一致。

以资源容器，[DAO](https://zh.wikipedia.org/wiki/DAO)实现和事务策略等形式与[Hibernate](https://zh.wikipedia.org/wiki/Hibernate)，[JDO](https://zh.wikipedia.org/w/index.php?title=Java_Data_Objects&action=edit&redlink=1)和[iBATIS](https://zh.wikipedia.org/wiki/IBATIS) [SQL Maps](https://zh.wikipedia.org/w/index.php?title=SQL_Maps&action=edit&redlink=1)集成。利用众多的翻转控制方便特性来全面支持，解决了许多典型的Hibernate集成问题。所有这些全部遵从Spring通用事务处理和通用数据访问对象异常等级规范。

灵活的基于核心Spring功能的[MVC](https://zh.wikipedia.org/wiki/%E6%A8%A1%E5%9E%8B-%E8%A7%86%E5%9B%BE-%E7%A8%8B%E5%BA%8F%E6%8E%A7%E5%88%B6)[网页应用程序](https://zh.wikipedia.org/wiki/%E7%BD%91%E9%A1%B5%E5%BA%94%E7%94%A8%E7%A8%8B%E5%BA%8F)框架。开发者通过策略接口将拥有对该框架的高度控制，因而该框架将适应于多种呈现（View）技术，例如[JSP](https://zh.wikipedia.org/wiki/JSP)、[FreeMarker](https://zh.wikipedia.org/wiki/FreeMarker)、[Velocity](https://zh.wikipedia.org/w/index.php?title=Velocity_(software)&action=edit&redlink=1)、[Tiles](https://zh.wikipedia.org/w/index.php?title=Tiles&action=edit&redlink=1)、[iText](https://zh.wikipedia.org/wiki/IText)以及[POI](https://zh.wikipedia.org/wiki/Apache_POI)。值得注意的是，Spring中间层可以轻易地结合于任何基于MVC框架的网页层，例如[Struts](https://zh.wikipedia.org/wiki/Struts)、[WebWork](https://zh.wikipedia.org/w/index.php?title=WebWork&action=edit&redlink=1)或[Tapestry](https://zh.wikipedia.org/wiki/Tapestry_(programming))。

提供诸如事务管理等服务的[面向方面编程](https://zh.wikipedia.org/wiki/%E9%9D%A2%E5%90%91%E6%96%B9%E9%9D%A2%E7%BC%96%E7%A8%8B)框架。

在设计应用程序Model时，MVC[模式](https://zh.wikipedia.org/wiki/%E8%AE%BE%E8%AE%A1%E6%A8%A1%E5%BC%8F_(%E8%AE%A1%E7%AE%97%E6%9C%BA))（例如[Struts](https://zh.wikipedia.org/wiki/Struts)）通常难于给出一个简洁明了的框架结构。Spring却具有能够让这部分工作变得简单的能力。程序开发员们可以使用Spring的JDBC抽象层重新设计那些复杂的框架结构。

通过本次企业实训我们掌握了自己以前不熟悉的技术和规范流程，对自己已经掌握的技术有了更深一层次的了解，为我们以后毕业进入企业工作打好了坚实的基础。