1. 版本控制工具

Hub

1. 开发工具

Ide

1. 数据库

Mysql

1. 框架

Spring+springmvc+mybatis

1. 工具

Maven等… jdk1.8

表分析

商品表

用户表

…..

在需要用的文件夹下右键 git bash here

GitHub使用命令记录

Git init 初始化git

Git status 查看状态（可以查看出当前文件夹下的文件）

解决乱码名称：git config --global core.quotepath false

上传文件 git add “文件名称”或者git add –A上传当前目录下所有文件

可以 git commit -m "更新的信息"  来提交

Git log查看日志

Git reset –hard 版本号前7位 版本回退

git clean -xf  删除没有上传的文件

本地关联GitHub 运行 git remote add origin （github中的地址）

git push -u origin master 查看是否关联成功

以后如果想更新文件到github 提交（ git commit -m "更新的信息"  来提交）完后输入 git push就可以同步了

Git

1. 初始化
2. 关联
3. 添加到暂存区
4. 提交到仓库
5. 同步

Idea

关闭一个项目 file-----colse project

快捷键：ctrl+d 复制当前行

Ctrl+x 剪贴选中文本

Ctrl+y删除选中的行

Ctrl+Alt+L，格式化代码

Maven

创建一个maven项目 导入需要的jar包 在pom.xml中

1. spring jar包
2. springMVC jar包
3. mybatis jar包

……..

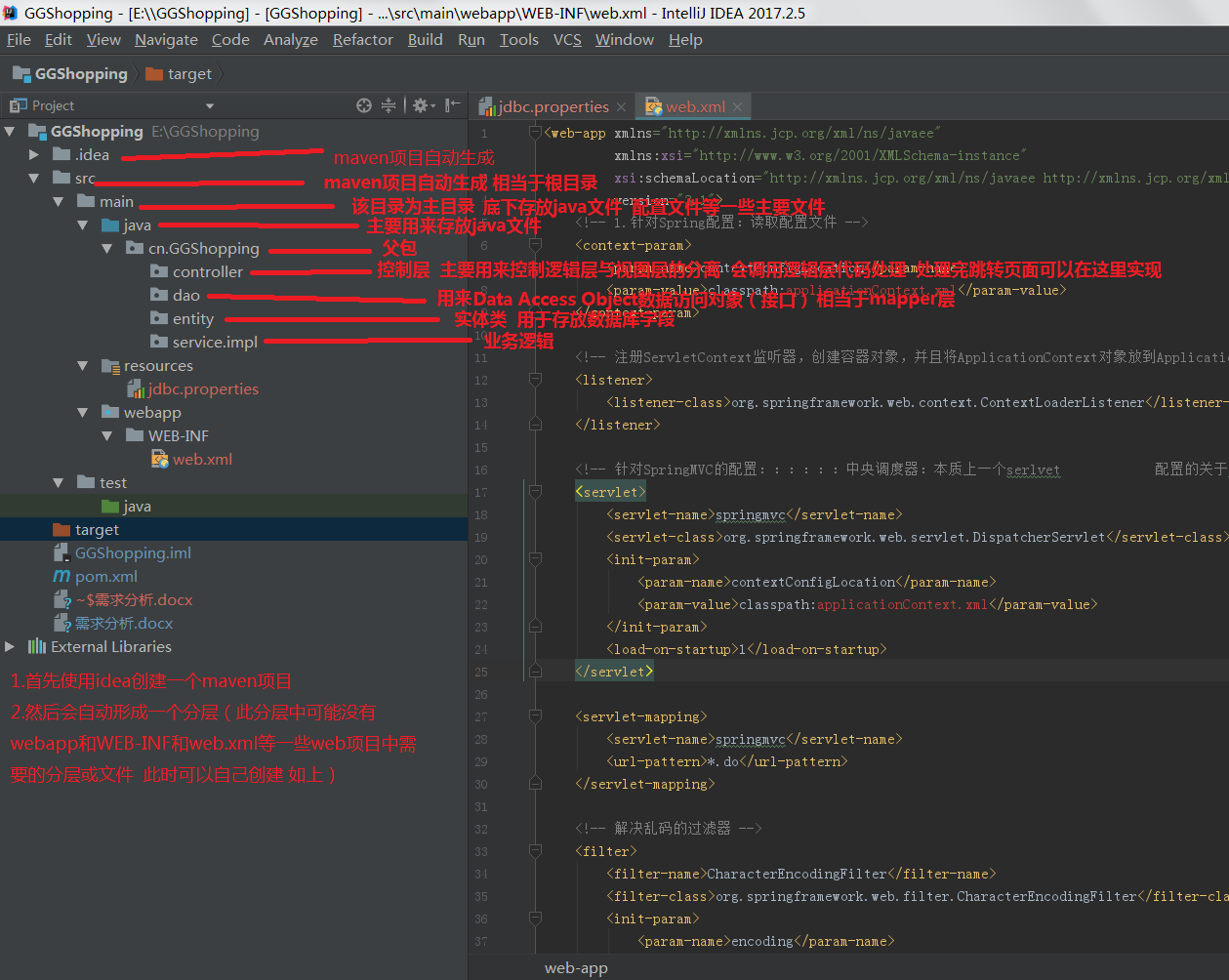
Spring4.2.5

Junit4.1.2

Ssm增删改查学习方式：

1. 首先列出123等几条 写出增删改查应该写的xml文件，java文件等，分出先后
2. 在写每个xml时弄清楚xml中每个标签 每个属性的用处并留笔记（可暂时借助网上参考）
3. 之后尽量不借助网上资料自己写出一个完整的增删改查
4. 每写一遍增删改查就要在这个增删改产中添加新的功能并记录如何实现的 慢慢形成自己的GGShopping

分层展示：



## 持久层：

### DAO层（mapper层）：

主要做数据持久的工作，与数据库进行联络 只是一个接口 可以在spring配置文件中定义此接口的实现类 可以在mybatis中使用此接口来进行增删改查的sql‘语句操作 而service层只需要调用dao层的接口就可以实现业务功能

## 业务层：

### Service层：

先写出接口然后写实现类impl 接着再在Spring的配置文件中配置其实现的关联。这样我们就可以在应用中调用Service接口来进行业务处理。 在service中需要调用dao层来实现具体的功能

## 表现层：

### Controller层（Handler层）

主要负责主要的业务模块流程的控制，需要调用service层 跳转view jsp页面

在resource包下新建Directory：“mapper”（用于存放xxxMapper.xml文件）和“spring”（用于存放spring-xxx.xml配置文件），新建文件：“jdbc.properties”（mysql数据库配置文件）,”log4j.properties”（日志输出配置文件）,”mybatis-config.xml”（mybatis框架配置文件）。

在web-inf目录下新建“jsp”包，存放xxx.jsp显示界面。

第一步：创建maven项目

第二步：分层搭骨架

第三步：pom.xml中导入jar包

Pom.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?> <project xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">   <modelVersion>4.0.0</modelVersion>  <groupId>cn.fyg.GGShopping</groupId>  <artifactId>GGShopping</artifactId>  <packaging>war</packaging>  <version>1.0-SNAPSHOT</version>  <name>GGShopping Maven Webapp</name>  <url>http://maven.apache.org</url>  <dependencies>  <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>4.2.5.RELEASE</version>  </dependency>  <!-- https://mvnrepository.com/artifact/junit/junit -->  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.12</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>javaee</groupId>  <artifactId>javaee-api</artifactId>  <version>5</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>4.2.2.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>4.2.2.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.2.2.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-tx</artifactId>  <version>4.1.9.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>4.2.1.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>4.1.9.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.2.3.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>4.1.9.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>4.2.3.RELEASE</version>  </dependency>   <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.1.9.RELEASE</version>  </dependency>   <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.2.2</version>  </dependency>   <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.3.0</version>  </dependency>   <dependency>  <groupId>org.javassist</groupId>  <artifactId>javassist</artifactId>  <version>3.17.1-GA</version>  </dependency>   <dependency>  <groupId>cglib</groupId>  <artifactId>cglib</artifactId>  <version>2.2.2</version>  </dependency>   <dependency>  <groupId>c3p0</groupId>  <artifactId>c3p0</artifactId>  <version>0.9.1.2</version>  </dependency>   <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.38</version>  </dependency>   </dependencies>  <build>  <finalName>GGShopping</finalName>  </build>   </project> |

第四步：建表

商品表：

|  |
| --- |
| CREATE TABLE `goods` (  `goods\_id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '商品id',  `good\_name` varchar(200) NOT NULL COMMENT '商品名称',  `goods\_detail` varchar(200) NOT NULL COMMENT '商品描述',  PRIMARY KEY (`goods\_id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 |

用户表：

|  |
| --- |
| CREATE TABLE `shop\_user` (  `user\_id` bigint(20) NOT NULL AUTO\_INCREMENT COMMENT '用户id',  `user\_name` varchar(100) NOT NULL COMMENT '用户名称',  `user\_sex` varchar(5) NOT NULL COMMENT '用户性别',  `user\_age` varchar(20) NOT NULL COMMENT '用户年龄',  `password` varchar(20) NOT NULL COMMENT '密码',  PRIMARY KEY (`user\_id`)  ) ENGINE=InnoDB DEFAULT CHARSET=utf8 |

第五步：创建get、set方法

User.java

|  |
| --- |
| package cn.GGShopping.entity;  public class User {  private long userId;  private String userName;  private String userSex;  private String userAge;  private String password;   public long getUserId() {  return userId;  }   public void setUserId(long userId) {  this.userId = userId;  }   public String getUserName() {  return userName;  }   public void setUserName(String userName) {  this.userName = userName;  }   public String getUserSex() {  return userSex;  }   public void setUserSex(String userSex) {  this.userSex = userSex;  }   public String getUserAge() {  return userAge;  }   public void setUserAge(String userAge) {  this.userAge = userAge;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  } } |

Goods.java

|  |
| --- |
| package cn.GGShopping.entity;  public class Goods {  private long goodsId;  private String goodsName;  private String goodsDetail;   public long getGoodsId() {  return goodsId;  }   public void setGoodsId(long goodsId) {  this.goodsId = goodsId;  }   public String getGoodsName() {  return goodsName;  }   public void setGoodsName(String goodsName) {  this.goodsName = goodsName;  }   public String getGoodsDetail() {  return goodsDetail;  }   public void setGoodsDetail(String goodsDetail) {  this.goodsDetail = goodsDetail;  } } |

Web.xml

|  |
| --- |
| <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  version="3.1">  <!-- 1.针对Spring配置：读取配置文件 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </context-param>   <!-- 注册ServletContext监听器，创建容器对象，并且将ApplicationContext对象放到Application域中 -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>   <!-- 针对SpringMVC的配置：：：：：：中央调度器：本质上一个serlvet 配置的关于SpringmVC组件 -->  <servlet>  <servlet-name>springmvc</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:applicationContext.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>   <servlet-mapping>  <servlet-name>springmvc</servlet-name>  <url-pattern>\*.do</url-pattern>  </servlet-mapping>   <!-- 解决乱码的过滤器 -->  <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>   <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>   <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> </web-app> |

ApplicationContext.xml

|  |
| --- |
|  |