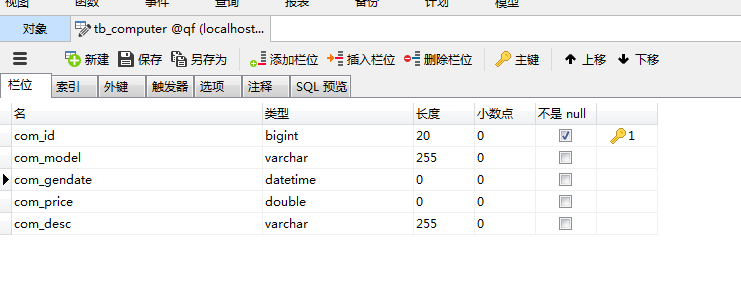
# Spring+mybatis+springMVC

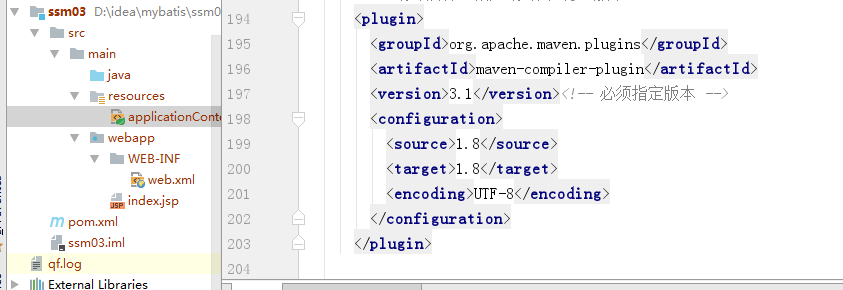
## 第一步：根据需求做设计

|  |  |  |
| --- | --- | --- |
| **字段名称** | **类型** | **说明** |
| com\_id | bigint | 电脑主键ID自增 |
| com\_model | varchar | 电脑型号 |
| com\_gendate | datetime | 生产日期 |
| com\_price | float | 价格 |
| com\_desc | varchar | 电脑简介 |

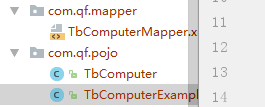


## 第二步：搭建环境

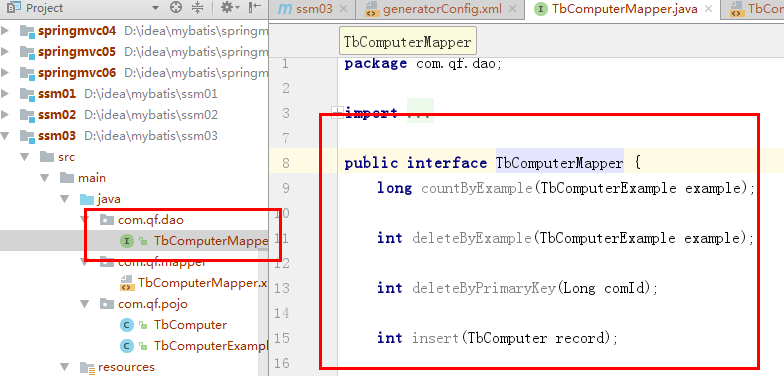
|  |
| --- |
| <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.qf.service</groupId>  <artifactId>ssm01</artifactId>  <packaging>war</packaging>  <version>1.0</version>  <name>ssm01 Maven Webapp</name>  <url>http://maven.apache.org</url>  <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>3.8.1</version>  <scope>test</scope>  </dependency>  <!-- 添加mybatis的核心包 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis</artifactId>  <version>3.4.5</version>  </dependency>  <!-- 添加mybatis与Spring整合的核心包 -->  <dependency>  <groupId>org.mybatis</groupId>  <artifactId>mybatis-spring</artifactId>  <version>1.3.1</version>  </dependency>  <dependency>  <groupId>org.mybatis.caches</groupId>  <artifactId>mybatis-ehcache</artifactId>  <version>1.1.0</version>  </dependency>  <!-- 自动生成插件包 -->  <dependency>  <groupId>org.mybatis.generator</groupId>  <artifactId>mybatis-generator-core</artifactId>  <version>1.3.5</version>  </dependency>  <!-- mysql驱动:根据数据库的版本选择驱动版本 -->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.34</version>  </dependency>  <!-- 日志包 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.7</version>  </dependency>  <!-- slf4j日志框架和log4j 转换包 -->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.7.7</version>  </dependency>  <!-- 添加Spring包 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <!-- 为了方便进行单元测试，添加spring-test包 -->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-aspects</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-expression</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-orm</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context-support</artifactId>  <version>4.3.11.RELEASE</version>  </dependency>  <!-- dbcp的连接池 -->  <dependency>  <groupId>commons-dbcp</groupId>  <artifactId>commons-dbcp</artifactId>  <version>1.4</version>  </dependency>  <dependency>  <groupId>commons-lang</groupId>  <artifactId>commons-lang</artifactId>  <version>2.6</version>  </dependency>  <!-- 添加servlet核心包 web容器提供了对应的包 关联tomcat8.5 -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>javax.servlet.jsp-api</artifactId>  <version>2.3.2-b01</version>  <scope>provided</scope>  </dependency>  <!-- jstl -->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <!-- 文件上传版本 -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  <version>1.3.1</version>  </dependency>  <dependency>  <groupId>commons-io</groupId>  <artifactId>commons-io</artifactId>  <version>2.4</version>  </dependency>  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-core</artifactId>  <version>2.7.4</version>  </dependency>  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.7.4</version>  </dependency>  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-annotations</artifactId>  <version>2.7.4</version>  </dependency>  </dependencies>  <build>  <finalName>ssm01</finalName>  <!-- mybatis插件 -->  <plugins>  <plugin>  <groupId>org.mybatis.generator</groupId>  <artifactId>mybatis-generator-maven-plugin</artifactId>  <version>1.3.5</version>  <configuration>  <verbose>true</verbose>  <overwrite>true</overwrite>  </configuration>  </plugin>  <!-- 编译插件，指定编译用的jdk版本 -->  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-compiler-plugin</artifactId>  <version>3.1</version><!-- 必须指定版本 -->  <configuration>  <source>1.8</source>  <target>1.8</target>  <encoding>UTF-8</encoding>  </configuration>  </plugin>  </plugins>  <!-- 把java目录中的配置文件，也能够打包到jar包中。 -->  <resources>  <resource>  <directory>src/main/resources</directory>  </resource>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <!-- 是否替换资源中的属性-->  <filtering>false</filtering>  </resource>  </resources>  </build>  </project> |



## 第三步：生成Mybatis映射文件和实体类



## 第四步：开发持久层(生成Mapper接口的方式)



步需要自己写实现类

需要配置然后动态生成代理实现类

## 第五步：开发服务层

|  |
| --- |
| **package** com.qf.service;  **import** com.qf.dto.ComputerDTO;  **import** java.util.List;  */\*\*  \* Thanks for Everything.  \*/* **public interface** ComputerService {   **public void** addComputer(ComputerDTO dto);   **public** List<ComputerDTO> findAll();   */\*\*  \* 分页插件  \** ***@param page*** *第几页  \** ***@param rows*** *每页显示的记录数  \** ***@return*** *\*/* **public** List<ComputerDTO> findByPage(**int** page,**int** rows);  } |
| **package** com.qf.service;  **import** com.qf.dao.TbComputerMapper; **import** com.qf.dto.ComputerDTO; **import** com.qf.pojo.TbComputer; **import** com.qf.pojo.TbComputerExample; **import** com.qf.utils.ComDate; **import** org.springframework.beans.BeanUtils; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.stereotype.Service;  **import** java.text.SimpleDateFormat; **import** java.util.ArrayList; **import** java.util.Date; **import** java.util.List;  */\*\*  \* Thanks for Everything.  \*/* @Service(**"computerService"**) **public class** ComputerServiceImpl **implements** ComputerService {  *//依赖持久层,注入,找DAO的接口自动注入* @Autowired  **private** TbComputerMapper **tbComputerMapper**;   @Override  **public void** addComputer(ComputerDTO dto) {  TbComputer record = **new** TbComputer();  BeanUtils.*copyProperties*(dto,record);*//时间会赋值不成功* String comGendate = dto.getComGendate();  *//时间赋值* record.setComGendate(ComDate.*strToDate*(comGendate));  **tbComputerMapper**.insertSelective(record);  }   @Override  **public** List<ComputerDTO> findAll() {  TbComputerExample example = **new** TbComputerExample();  List<TbComputer> tbComputers = **tbComputerMapper**.selectByExample(example);  List<ComputerDTO> list = **new** ArrayList<ComputerDTO>();  **for** (TbComputer tbComputer : tbComputers) {  ComputerDTO dto = **new** ComputerDTO();  BeanUtils.*copyProperties*(tbComputer,dto);  *//时间单独处理* dto.setComGendate(ComDate.*dateToStr*(tbComputer.getComGendate()));  }  **return** list;  }   @Override  **public** List<ComputerDTO> findByPage(**int** page, **int** rows) {  **return null**;  } } |

时间工具类:

|  |
| --- |
| **package** com.qf.utils;  **import** java.text.ParseException; **import** java.text.SimpleDateFormat; **import** java.util.Date;  */\*\*  \* Thanks for Everything.  \*/* **public class** ComDate {  **private static** SimpleDateFormat *simpleDateFormat* = **new** SimpleDateFormat(**"yyyy-MM-dd"**);   **public static** Date strToDate(String strDate) {  Date parse = **null**;  **try** {  parse = *simpleDateFormat*.parse(strDate);  } **catch** (ParseException e) {  e.printStackTrace();  }  **return** parse;  }   **public static** String dateToStr(Date date ){  String format = *simpleDateFormat*.format(date);  **return** format;  } } |

## 第六步：开发表现 层

|  |
| --- |
| **package** com.qf.action;  **import** com.qf.dto.ComputerDTO; **import** com.qf.service.ComputerService; **import** org.springframework.beans.factory.annotation.Autowired; **import** org.springframework.beans.factory.annotation.Qualifier; **import** org.springframework.stereotype.Controller; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.servlet.ModelAndView;  **import** javax.servlet.http.HttpServletRequest; **import** java.util.List;  */\*\*  \* Thanks for Everything.  \*/* @Controller **public class** ComputerAction {  *//注入服务层* @Autowired  @Qualifier(**"computerService"**)  **private** ComputerService **computerService**;   @RequestMapping(**"/addCom.do"**)  **public** String addCom(ComputerDTO dto){  **computerService**.addComputer(dto);  **return "forward:/listCom.do"**;  }   @RequestMapping(**"/listCom.do"**)  **public** ModelAndView listCom(HttpServletRequest request){  ModelAndView mv = **new** ModelAndView();  List<ComputerDTO> all = **computerService**.findAll();  *//设置值* mv.addObject(**"comList"**,all);  *//跳转页面* mv.setViewName(**"comlist"**);  **return** mv;  }  } |

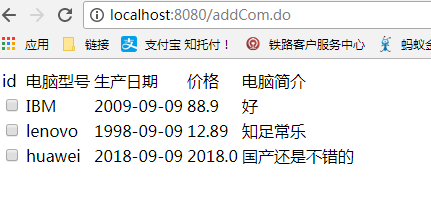
## 第七步：整合持久层配置

|  |
| --- |
| *<!-- 1，配置DBCP的数据源 --> <!-- 可以把数据库的四大属性配置在属性文件中,同时需要加载属性文件 -->* <**context:property-placeholder location="db.properties"**></**context:property-placeholder**> *<!-- 1,配置德鲁伊连接池 -->* <**bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource" destroy-method="close"**>  <**property name="url" value="${myurl}"**></**property**>  <**property name="driverClassName" value="${mydriver}"**></**property**>  <**property name="username" value="${myname}"**></**property**>  <**property name="password" value="${mypass}"**></**property**>  <**property name="maxActive" value="10"**></**property**>  <**property name="minIdle" value="5"**></**property**>  *<!-- 最大等待时间，超时抛链接超时 -->* <**property name="maxWait" value="300"**></**property**> </**bean**>  *<!-- 2,创建SqlSessionFactory ，由Spring提供 -->* <**bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean"**>  *<!-- 思考mybatis的核心配置部分:数据源，注册实体 -->* <**property name="dataSource" ref="dataSource"**></**property**>  *<!-- 注册实体，模糊匹配多个mapper映射文件-->* <**property name="mapperLocations" value="classpath:com/qf/mapper/\*Mapper.xml"**></**property**>  </**bean**>  *<!-- 3,没有DAO的具体实现类，怎么办？DAO接口扫描动态生成DAO接口的代理实现类 不需要加id属性，这里扫描的是所有的DAO接口。 如何把DAO注入给服务层呢？使用注解方式：注入Mapper代理接口。使用自动装配就可以了。 -->* <**bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>  *<!--必须定位到具体的dao接口包 -->* <**property name="basePackage" value="com.qf.dao"**></**property**> </**bean**>  *<!-- 4,创建服务层对象使用注解方式，注解注入DAO 目标类对象--> <!-- 开启注解扫描服务层的注解 -->* <**context:component-scan base-package="com.qf.action,com.qf.service"**></**context:component-scan**> *<!-- 5,创建事务管理通知类对象-->* <**bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager"**>  *<!-- 注入数据源-->* <**property name="dataSource" ref="dataSource"**></**property**> </**bean**> *<!-- 6,定义事务通知切面，让事务应用到具体的方法 -->* <**tx:advice id="txAdvice" transaction-manager="transactionManager"**>  <**tx:attributes**>  *<!--希望哪些方法需要应用到事务 -->* <**tx:method name="add\*" propagation="REQUIRED"**/>  <**tx:method name="insert\*" propagation="REQUIRED"**/>  <**tx:method name="del\*" propagation="REQUIRED"**/>  <**tx:method name="remove\*" propagation="REQUIRED"**/>  <**tx:method name="update\*" propagation="REQUIRED"**/>  <**tx:method name="find\*" read-only="true"**></**tx:method**>  <**tx:method name="get\*" read-only="true"**></**tx:method**>  </**tx:attributes**> </**tx:advice**> *<!-- 7,AOP编程：让切入点和通知关联形成切面 -->* <**aop:config**>  <**aop:pointcut id="myPoint" expression="execution(\* com.qf.service.\*.\*(..))"**></**aop:pointcut**>  <**aop:advisor advice-ref="txAdvice" pointcut-ref="myPoint"**></**aop:advisor**> </**aop:config**> |

## 第八步：整合表现层配置

|  |
| --- |
| *<!-- 整合表现层配置 --> <!-- 8,注解的处理器映射器和适配器 -->* <**mvc:annotation-driven**></**mvc:annotation-driven**>  *<!-- 9视图解析器:逻辑视图-->* <**bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  *<!-- 前缀 -->* <**property name="prefix" value="/"**></**property**>  *<!-- 后缀 -->* <**property name="suffix" value=".jsp"**></**property**> </**bean**> *<!-- 10扫描 Action-->* <**context:component-scan base-package="com.qf.action"**></**context:component-scan**> |
| Web.xml  添加初始化容器的配置，服务器启动的时候，加载所有的bean,如果有异常，启动就可以看见失败的部分。  *<!-- 全局参数 -->* <**context-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:applicationContext.xml</**param-value**> </**context-param**>  <**filter**>  <**filter-name**>myfilter</**filter-name**>  <**filter-class**>org.springframework.web.filter.CharacterEncodingFilter</**filter-class**>  *<!-- 指定编码 -->* <**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**>myfilter</**filter-name**>  <**url-pattern**>/\*</**url-pattern**> </**filter-mapping**>  *<!-- Spring容器的监听 -->* <**listener**>  <**listener-class**>org.springframework.web.context.ContextLoaderListener</**listener-class**>  *<!-- 加载配置文件 读取全局的属性contextConfigLocation-->* </**listener**>  <**servlet**>  <**servlet-name**>springmvc</**servlet-name**>  <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  *<!-- 读取springmvc的配置文件 -->* <**init-param**>  <**param-name**>contextConfigLocation</**param-name**>  <**param-value**>classpath:applicationContext.xml</**param-value**>  </**init-param**> </**servlet**> <**servlet-mapping**>  <**servlet-name**>springmvc</**servlet-name**>  <**url-pattern**>\*.do</**url-pattern**> </**servlet-mapping**> |

开发页面整合测试：

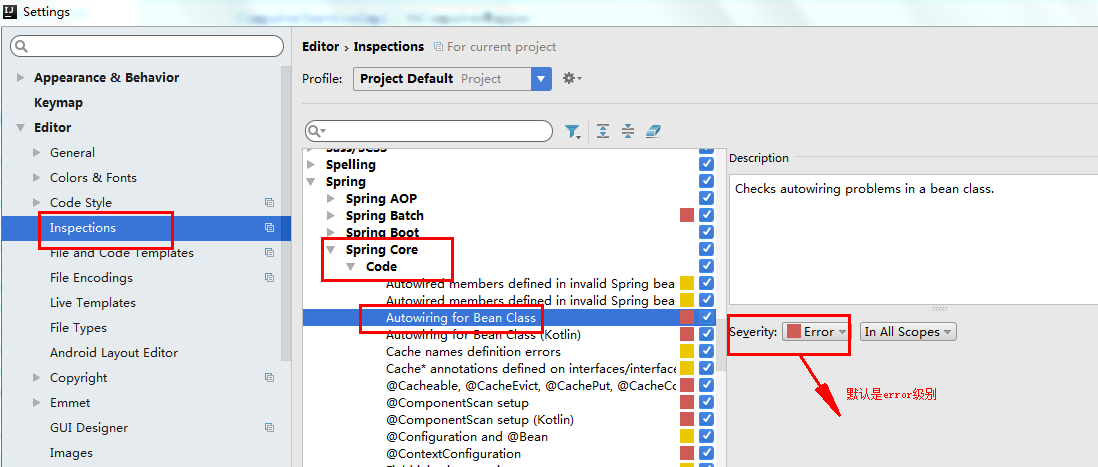


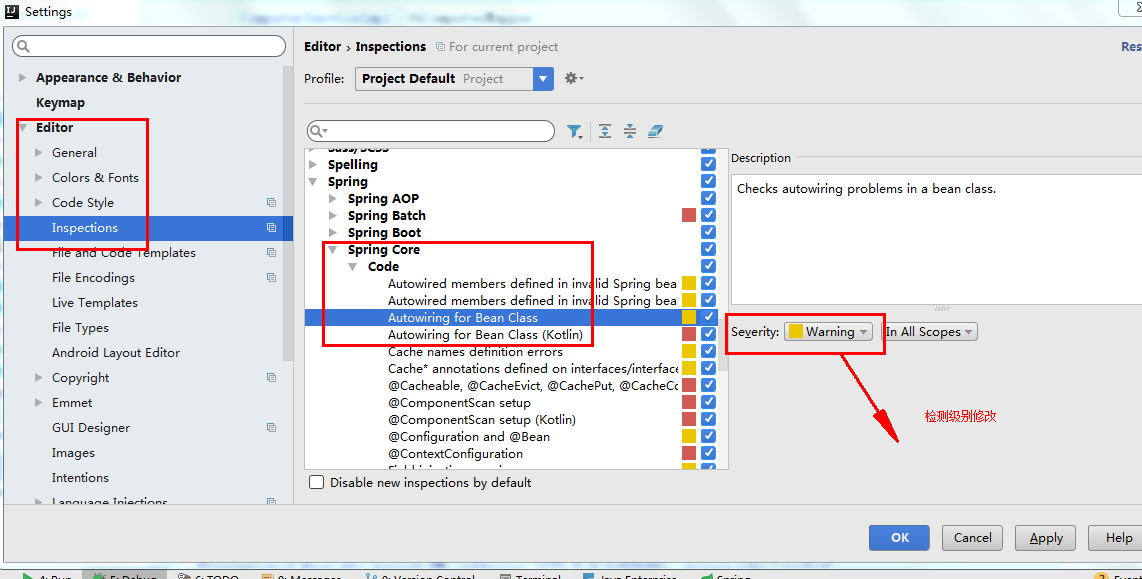
# 降低检测级别

默认情况：



降低：





再次查看：



没有了红色下划线。