SSM 环境搭建与产品操作

1.环境准备

1.1 数据库与表结构

1.1.1 创建用户与授权

数据库我们使用Oracle

Oracle 为每个项目创建单独user，oracle数据表存放在表空间下，每个用户有独立表空间

创建用户及密码：

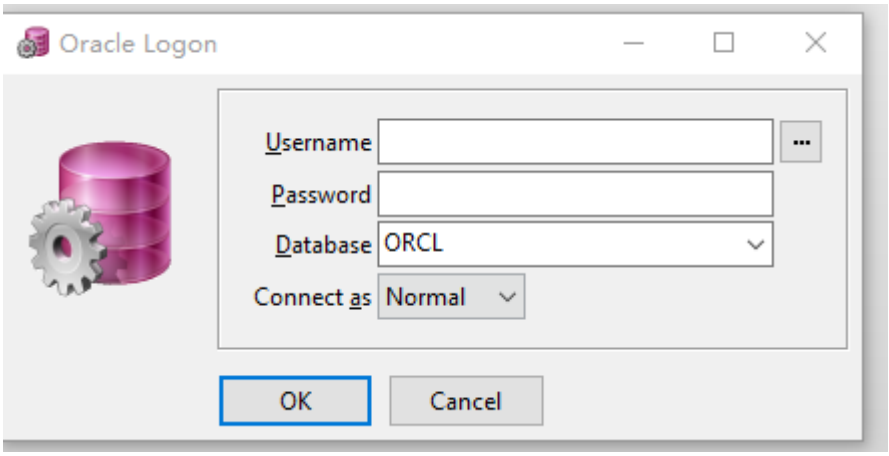
```
语法[创建用户]： create user 用户名 identified by 口令[即密码];
例子： create user test identified by test;
```

授权

```
语法： grant connect, resource to 用户名;
 例子： grant connect, resource to test;
```

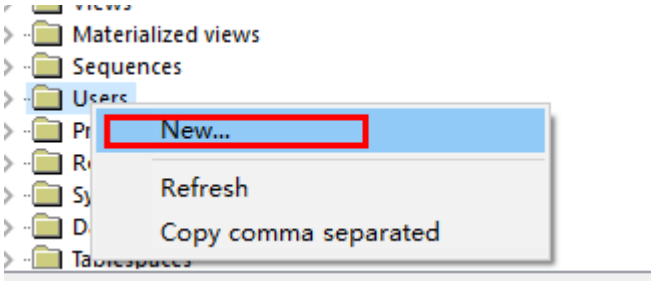
PL/SQL Developer是一个集成开发环境，专门面向Oracle数据库存储程序单元的开发PL/SQL Developer侧重于易用性、代码品质和生产力，充分发挥Oracle应用程序开发过程中的主要优势。

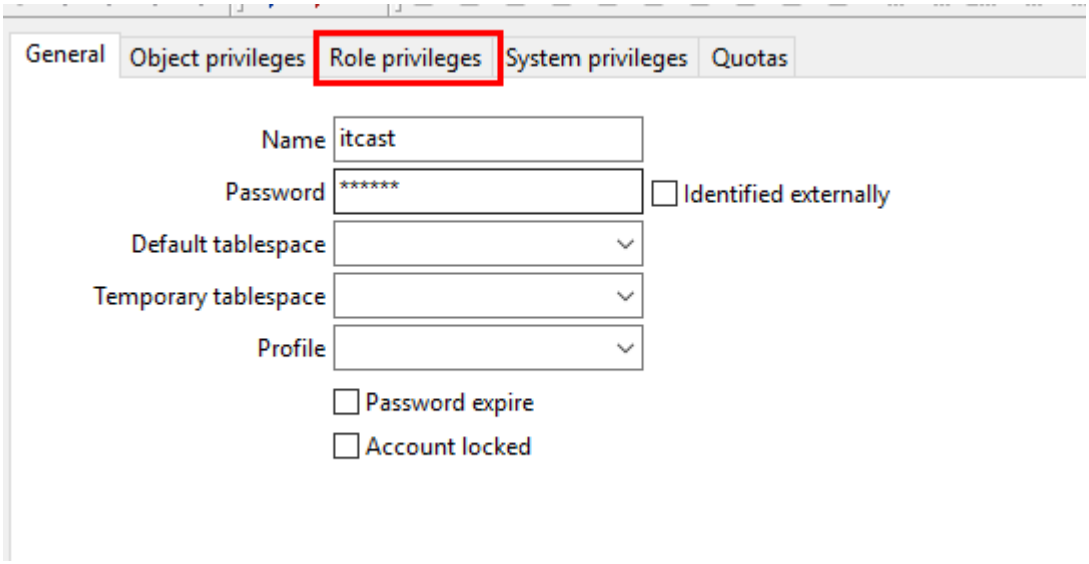
- 连接oracle数据库



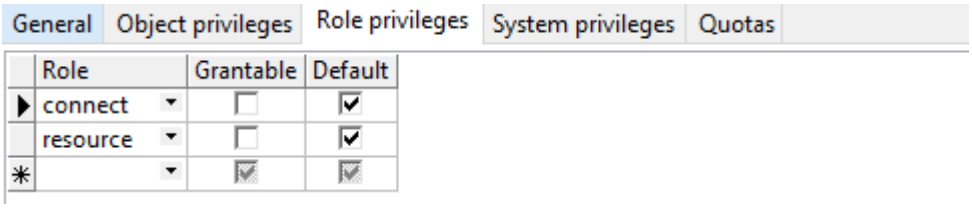
- 创建用户及授权

  a) 创建用户





b) 授权



对象权限是指针对于某一张表的操作权限,系统权限是指对表的CRUD操作权限, 角色权限是系统权限的集合,我们设置 时，一般是设置角色权限，设置resource与connect

1.1.2 创建表

产品表信息描述

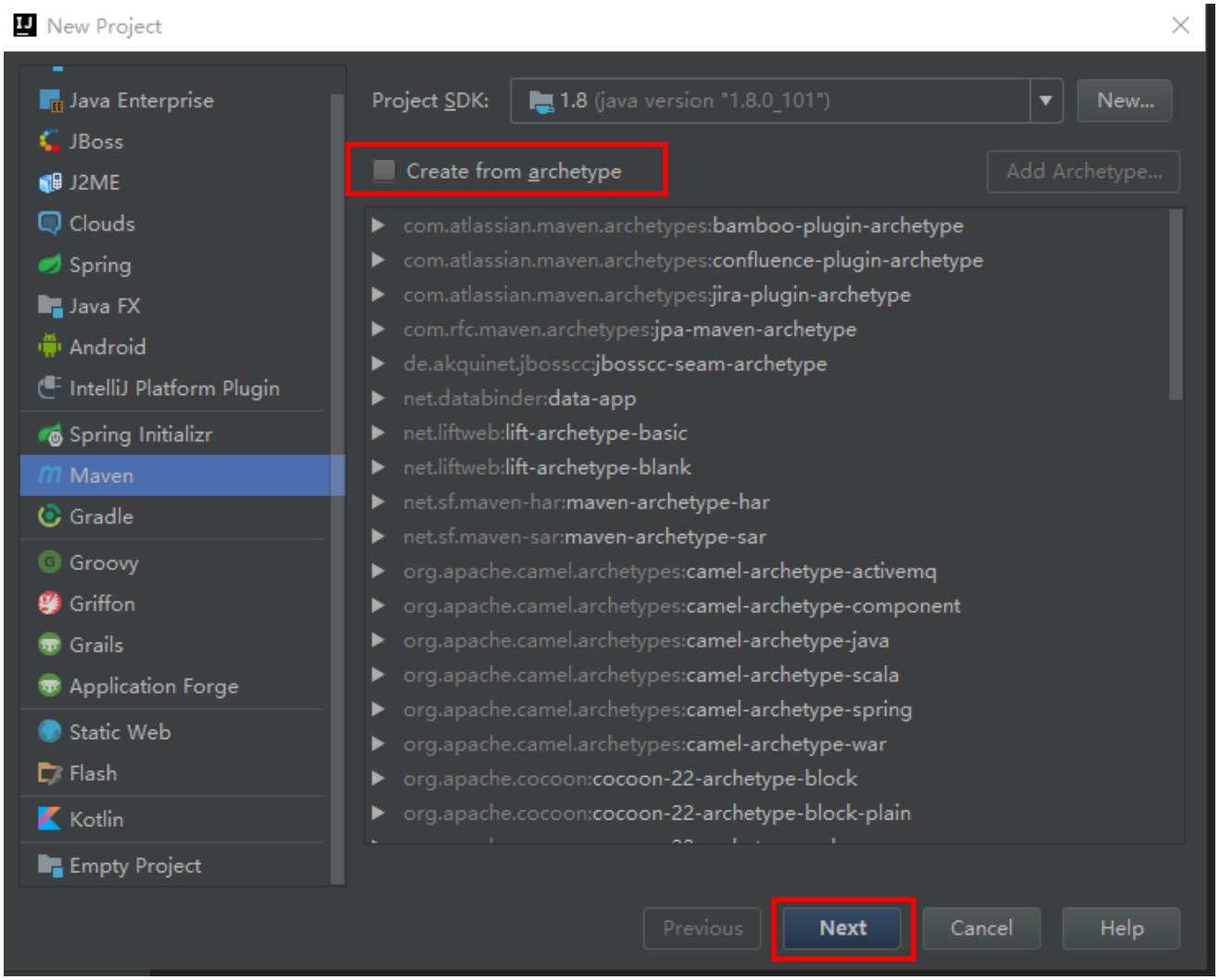
序号	字段名称	字段类型	字段描述
1	id	varchar2(32)	无意义，主键uuid
2	productNum	varchar2(50)	产品编号，唯一，不为空
3	productName	varchar2(50)	产品名称（路线名称）
4	cityName	varchar2(50)	出发城市
5	DepartureTime	timestamp	出发时间
6	productPrice	number	产品价格
7	productDesc	varchar2(500)	产品描述
8	productStatus	int	状态(0 关闭 1 开启)

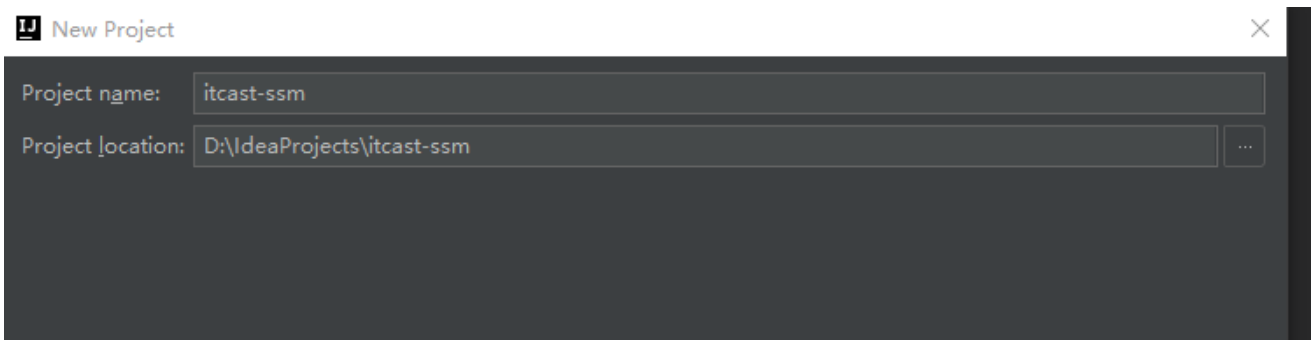
创建表sql

```
CREATE TABLE product(
  id varchar2(32) default SYS_GUID() PRIMARY KEY,
  productNum VARCHAR2(50) NOT NULL,
  productName VARCHAR2(50),
  cityName VARCHAR2(50),
  DepartureTime timestamp,
  productPrice Number,
  productDesc VARCHAR2(500),
  productStatus INT,
  CONSTRAINT product UNIQUE (id, productNum)
)

insert into PRODUCT (id, productnum, productname, cityname, departuretime, productprice,
productdesc, productstatus)
values ('676C5BD1D35E429A8C2E114939C5685A', 'itcast-002', '北京三日游', '北京', to_timestamp('10-
10-2018 10:10:00.000000', 'dd-mm-yyyy hh24:mi:ss.ff'), 1200, '不错的旅行', 1);
insert into PRODUCT (id, productnum, productname, cityname, departuretime, productprice,
productdesc, productstatus)
values ('12B7ABF2A4C544568B0A7C69F36BF8B7', 'itcast-003', '上海五日游', '上海', to_timestamp('25-
04-2018 14:30:00.000000', 'dd-mm-yyyy hh24:mi:ss.ff'), 1800, '魔都我来了', 0);
insert into PRODUCT (id, productnum, productname, cityname, departuretime, productprice,
productdesc, productstatus)
values ('9F71F01CB448476DAFB309AA6DF9497F', 'itcast-001', '北京三日游', '北京', to_timestamp('10-
10-2018 10:10:00.000000', 'dd-mm-yyyy hh24:mi:ss.ff'), 1200, '不错的旅行', 1);
```

1.2 maven工程搭建

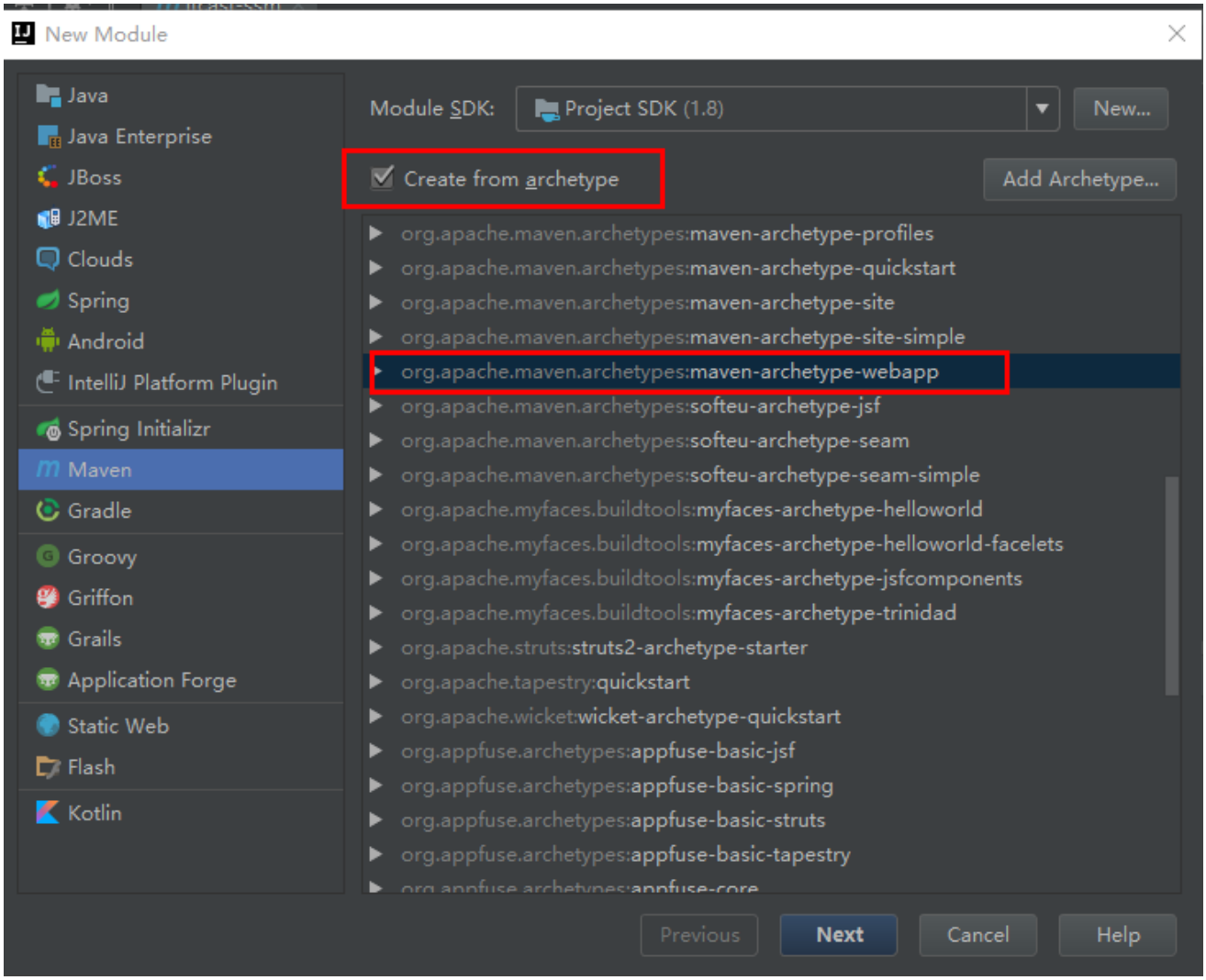
1.2.1 创建maven工程





创建子模块

itcast-ssm-web itcast-ssm-domain itcast-ssm-service itcast-ssm-dao itcast-ssm-utils 其中创建itcast-ssm-web时注意我们选择一个web工程



1.2.2 pom.xml

```
<properties>
     <spring.version>5.0.2.RELEASE</spring.version>
     <slf4j.version>1.6.6</slf4j.version>
     <log4j.version>1.2.12</log4j.version>
     <oracle.version>11.2.0.1.0</oracle.version>
     <mybatis.version>3.4.5</mybatis.version>
     <spring.security.version>5.0.1.RELEASE</spring.security.version>
 </properties>

 <dependencies>        <!-- spring -->
     <dependency>
         <groupId>org.aspectj</groupId>
         <artifactId>aspectjweaver</artifactId>

         <version>1.6.8</version>
```

```
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context-support</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-orm</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-test</artifactId>
            <version>${spring.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>

        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-tx</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <dependency>
```

```
        <groupId>junit</groupId>
        <artifactId>junit</artifactId>
        <version>4.12</version>
        <scope>test</scope>
    </dependency>

    <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>jsp-api</artifactId>
        <version>2.0</version>
        <scope>provided</scope>
    </dependency>
    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>        <!-- log start -->
    <dependency>
        <groupId>log4j</groupId>
        <artifactId>log4j</artifactId>
        <version>${log4j.version}</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
        <version>${slf4j.version}</version>
    </dependency>
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${slf4j.version}</version>
    </dependency>        <!-- log end -->

    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>${mybatis.version}</version>
    </dependency>
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>1.3.0</version>
    </dependency>
    <dependency>
        <groupId>c3p0</groupId>
        <artifactId>c3p0</artifactId>

        <version>0.9.1.2</version>
```

```
            <type>jar</type>
            <scope>compile</scope>
        </dependency>
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper</artifactId>
            <version>5.1.2</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-web</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-config</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-core</artifactId>
            <version>${spring.security.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.security</groupId>
            <artifactId>spring-security-taglibs</artifactId>
            <version>${spring.security.version}</version>
        </dependency>

        <dependency>
            <groupId>com.oracle</groupId>
            <artifactId>ojdbc14</artifactId>
            <version>${oracle.version}</version>
        </dependency>

        <dependency>
            <groupId>javax.annotation</groupId>
            <artifactId>jsr250-api</artifactId>
            <version>1.0</version>
        </dependency>

    </dependencies>
    <build>
         <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <source>1.8</source>
                        <target>1.8</target>

                        <encoding>UTF-8</encoding>
```

```
                    <showWarnings>true</showWarnings>
                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
```

1.3编写实体类

```
public class Product {
    private String id; // 主键
    private String productNum; // 编号 唯一
    private String productName; // 名称
    private String cityName; // 出发城市
    private Date departureTime; // 出发时间
    private String departureTimeStr;
    private double productPrice; // 产品价格
    private String productDesc; // 产品描述
    private Integer productStatus; // 状态 0 关闭 1 开启
    private String productStatusStr;
 }
```

1.4 编写业务接口

```
public interface IProductService {

    List<Product> findAll() throws Exception;

}
```

1.5 编写持久层接口

```
public interface IProductDao {

    @Select("select * from product")
    List<Product> findAll() throws Exception;
}
```

2.SSM整合与产品查询

2.1 Spring环境搭建

2.1.1.编写Spring配置文件applicationContext.xml

```
<!-- 配置 spring 创建容器时要扫描的包 -->
<!-- 开启注解扫描，管理service和dao -->
<context:component-scan base-package="com.itheima.ssm.service">
</context:component-scan>
<context:component-scan base-package="com.itheima.ssm.dao">
</context:component-scan>
```

2.1.2.使用注解配置业务层

```
@Service
public class ProductServiceImpl implements IProductService{

    @Override
    public List<Product> findAll() throws Exception {
        return null;
    }

}
```

2.2 Spring MVC 环境搭建

2.2.1.web.xml配置Spring MVC核心控制器

```
<!-- 配置 spring mvc 的核心控制器 -->
<servlet>
    <servlet-name>springmvcDispatcherServlet</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
    <!-- 配置初始化参数，用于读取 springmvc 的配置文件 -->
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:springmvc.xml</param-value>
    </init-param>
    <!-- 配置 servlet 的对象的创建时间点：应用加载时创建。取值只能是非 0 正整数，表示启动顺 序 -->
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>springmvcDispatcherServlet</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>
<!-- 配置 springMVC 编码过滤器 -->
<filter>
    <filter-name>CharacterEncodingFilter</filter-name>
    <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
    <!-- 设置过滤器中的属性值 -->
```

```
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
    <!-- 启动过滤器 -->
    <init-param>
        <param-name>forceEncoding</param-name>
        <param-value>true</param-value>
    </init-param>
</filter>
<!-- 过滤所有请求 -->
<filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>*.do</url-pattern>
</filter-mapping>
```

2.2.2.Spring MVC配置文件springmvc.xml

```
<!-- 扫描controller的注解，别的不扫描 -->
<context:component-scan base-package="com.itheima.ssm.controller">
</context:component-scan>

<!-- 配置视图解析器 -->
<bean id="viewResolver"
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <!-- JSP文件所在的目录 -->
    <property name="prefix" value="/pages/" />
    <!-- 文件的后缀名 -->
    <property name="suffix" value=".jsp" />
</bean>

    <!-- 设置静态资源不过滤 -->
    <mvc:resources location="/css/" mapping="/css/**" />
    <mvc:resources location="/img/" mapping="/img/**" />
    <mvc:resources location="/js/" mapping="/js/**" />
    <mvc:resources location="/plugins/" mapping="/plugins/**" />

    <!-- 开启对SpringMVC注解的支持 -->
    <mvc:annotation-driven />
    <!--
        支持AOP的注解支持，AOP底层使用代理技术
        JDK动态代理，要求必须有接口
        cglib代理，生成子类对象，proxy-target-class="true" 默认使用cglib的方式
    -->
    <aop:aspectj-autoproxy proxy-target-class="true"/>
</beans>
```

2.2.3.编写Controller

- ProductController

```
@Controller
@RequestMapping("/product")
public class ProductController {

    @Autowired
    private IProductService productService;

    @RequestMapping("/findAll.do")
    public ModelAndView findAll() {
        return null;
    }
}
```

2.3 Spring与Spring MVC整合

```
<!-- 配置加载类路径的配置文件 -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath*:applicationContext.xml,classpath*:spring-security.xml</param-value>
</context-param>
<!-- 配置监听器 -->
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

2.4 Spring与MyBatis整合

2.4.1.整合思路

把 mybatis 配置文件（mybatis.xml）中内容配置到 spring 配置文件中 同时，把 mybatis 配置文件的内容清掉。

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

</configuration>
```

注意： 理 由于我们使用的是代理 Dao ， 的模式，Dao 具体实现类由 MyBatis 使用代理方式创建，所以此时 mybatis 配置文件不能删。 当我们整合 spring 和 mybatis 时，mybatis 创建的 Mapper.xml 文件名必须和 Dao 接口 文件 名一致

2.4.2.Spring接管mybatis的Session工厂

db.properties

```
jdbc.driver=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://localhost:3306/ssm?useUnicode=true&characterEncoding=utf8
jdbc.username=root
jdbc.password=root
```

```
<context:property-placeholder location="classpath:db.properties"/>
<!-- 配置连接池 -->
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
    <property name="driverClass" value="${jdbc.driver}" />
    <property name="jdbcUrl" value="${jdbc.url}" />
    <property name="user" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
</bean>
<!-- 把交给IOC管理 SqlSessionFactory -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
</bean>
```

2.4.3.自动扫描所有Mapper接口和文件

```
<!-- 扫描dao接口 -->
<bean id="mapperScanner" class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <property name="basePackage" value="com.itheima.ssm.dao"/>
</bean>
```

2.4.4.配置Spring事务

```
<!-- 配置Spring的声明式事务管理 -->
<!-- 配置事务管理器 -->
<bean id="transactionManager"
class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <property name="dataSource" ref="dataSource"/>
</bean>
<tx:annotation-driven transaction-manager="transactionManager"/>
```

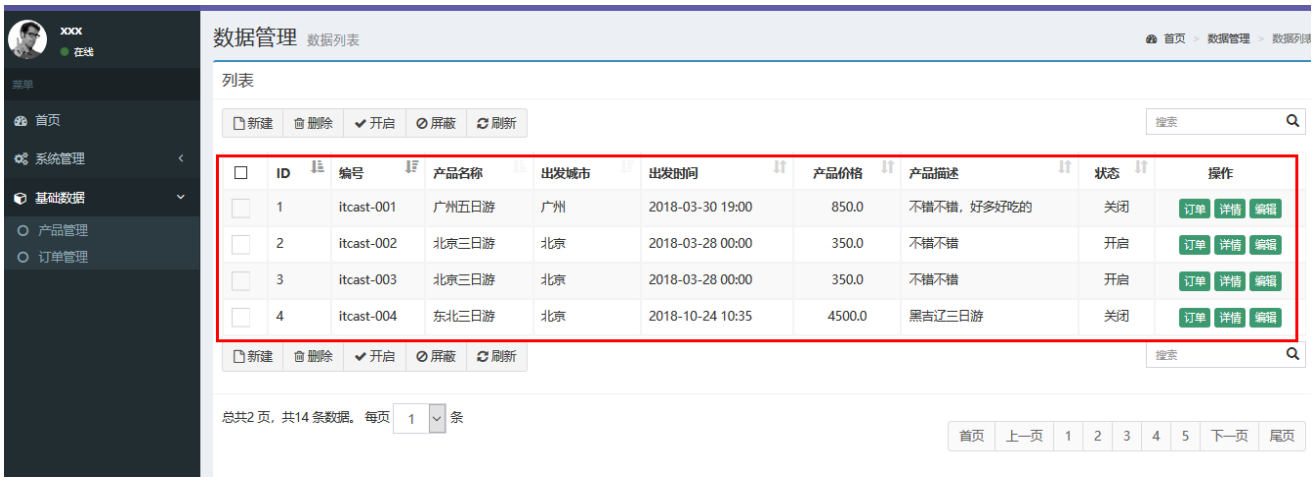
2.5 测试运行

2.5.1.编写jsp页面

2.5.1.1 请求发起页面 index.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
  pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>主页</title>
</head>
<body>
  <a href="${pageContext.request.contextPath}/product/findAll.do">查询产品信息</a>
</body>
</html>
```

2.5.1.2 显示产品页面 product-list.jsp



页面详细代码请查看今天课程资料

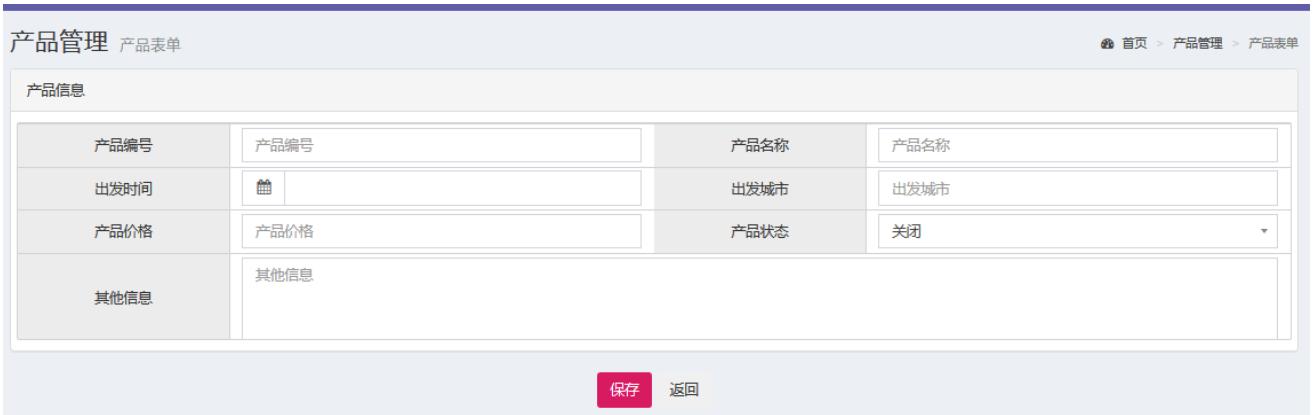
2.5.2.Controller

```
@Controller
@RequestMapping("/product")
public class ProductController {

    @Autowired
    private IProductService productService;

    @RequestMapping("/findAll.do")
    public ModelAndView findAll() throws Exception {
        ModelAndView mv = new ModelAndView();
        List<Product> products = productService.findAll();
        mv.addObject("productList", products);
        mv.setViewName("product-list");
        return mv;
    }
```

```
}
```

3.商品添加

3.1 商品添加页面 product-add.jsp



页面详细代码请查看今天课程资料

3.2 Controller

```
@Controller
@RequestMapping("/product")
public class ProductController {

    @Autowired
    private IProductService productService;

    @InitBinder
    public void initBinder(WebDataBinder binder) {
        binder.registerCustomEditor(Date.class, new MyDateEdit("yyyy-MM-dd HH:mm"));
    }


    @RequestMapping("/save.do")
    public String save(Product product) throws Exception {
        productService.save(product);
        return "redirect:findAll.do";
    }
    @RequestMapping("/findAll.do")
    public ModelAndView findAll() throws Exception {
        ModelAndView mv = new ModelAndView();
        List<Product> products = productService.findAll();
        mv.addObject("productList", products);
        mv.setViewName("product-list");

        return mv;
```

```
    }
}
```

3.3 Dao

```
public interface IProductDao {

    @Select("select * from product")
    List<Product> findAll() throws Exception;

    @Insert("insert into
product(productNum,productName,cityName,departureTime,productPrice,productDesc,productStatus)
values(#{productNum},#{productName},#{cityName},#{departureTime},#{productPrice},#
{productDesc},#{productStatus})")
    void save(Product product);
}
```