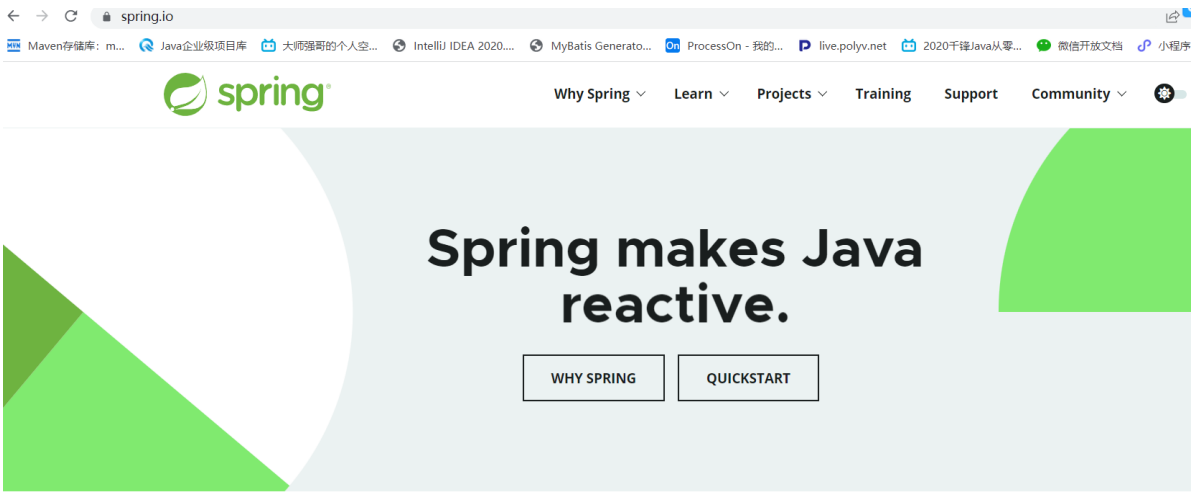
Spring框架

1. Spring框架

> Spring框架是一个 轻量级的控制反转和面向切面编程的容器 框架，用来解决企业项目开发的复杂度问题--解耦性

- 体积小，对代码没有入侵性
- 控制反转：（IOC) 把创建对象的工作交由Spring完成，Spring在创建对象的时候同时可以完成对象的赋值（DI）
- 面向切面：AOP 面向切面编程，可以在不改变原有的业务的逻辑的情况下实现对业务的增强。

2. Spring架构



- core 核心
- beans 实例管理
- context 容器上下文
- aop
- web springmvc
- test

3. Spring---基于注解版开发

3.1 spring框架部署

1. 添加spring依赖

```
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.16.RELEASE</version>
        </dependency>
    </dependencies>
```

2. 创建 spring的核心配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!--声明使用注解配置-->
    <context:annotation-config></context:annotation-config>

    <!--声明spring扫描的范围-->
    <context:component-scan base-package="org.qf.entity"></context:component-scan>


</beans>
```

3. spring常用的注解

- @Component

类注解，声明此类被spring容器进行管理，相当于bean标签的作用

@Component(value="stu") value属性用于指定当前bean的id，value属性也可以省略，如果省略当前类的id默认是类名首字母小写

- @Awtowired

属性注解、方法注解

- @Resource

属性注解

- @Service注解

业务层注解

- @Controller注解

控制器注解

4. 实体类

```
package org.qf.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.stereotype.Component;

@Data
@AllArgsConstructor
@NoArgsConstructor
@Component
public class Users {

    @Value("1")
    private int userId;

    @Value("aa")
    private String userName;

    @Value("123")
    private String userPwd;

    private String userRealname;

    private String userImg;

    @Autowired
    private Book book;  //将book对象作为属性注入到 users中
}
```

5. 测试

```
import org.qf.entity.Users;
import org.springframework.context.support.ClassPathXmlApplicationContext;


public class T {
    public static void main(String[] args) {

        ClassPathXmlApplicationContext context=new
ClassPathXmlApplicationContext("applicationContext.xml");

        Users users = (Users) context.getBean("users");  //创建对象


        System.out.println(users.getBook());

    }
}
```

SpringMVC框架

1. SpringMVC概述

> SpringMVC是由servlet封装 的用于实现MVC控制的框架，实现前端和服务器端的交互

1.1 springmvc优势

- 严格遵守MVC分层思想
- springmvc基于spring的扩展提供一套mvc的注解
- 对restful风格提供了良好的支持

1.2 springmvc本质工作

- 接收并解析请求
- 处理请求
- 数据渲染、响应请求

2. SpringMVC框架使用

2.1 创建 mavenweb工程

2.2 添加依赖

```
    <dependencies>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.2.16.RELEASE</version>
        </dependency>

    </dependencies>
```

2.3 创建springmvc配置文件

- 在resources目录下创建名为spring-servlet.xml的文件
- 添加mvc注解驱动

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/beans
```

```
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <context:annotation-config></context:annotation-config>

    <context:component-scan base-package="org.qf.controller"></context:component-scan>


    <mvc:annotation-driven></mvc:annotation-driven>

</beans>
```

2.4 在web.xml中配置springmvc的前端控制器

> springmvc提供了一个名为dispathcerSerlvet类（前端控制器），用于拦截用户请求交由springmvc处理。

```
<?xml version="1.0" encoding="UTF-8"?>
<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_4_0.xsd"
         version="4.0">

    <!--前端控制器-->
    <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:spring-servlet.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>


</web-app>
```

2.5 控制器

```
package org.qf.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class UserController {


    @RequestMapping("/login")
    public String login(){
        System.out.println("-------------------login--------------------------");
```

```
        return "success.jsp";
    }

}
```

3. SpringMVC框架使用

> 在springMVC中，我们把接收用户请求、处理用户请求的类称为Controller(控制器)

3.1 创建控制器

3.1.1 创建控制器类

- 创建一个名为org.qf.web的包（包需要在spring注解扫描的范围内）
- 创建类（无需做任何的继承和实现）
- 在类上添加@Controller注解声明此类为SpringMVC的控制器
- 在类上添加@RequestMapping("url")声明此控制器类的请求url（可以省略）

```
package org.qf.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/user")
public class UserController {

}
```

3.1.2 在控制器类中定义处理请求的方法

- 在一个控制器类中可以定义多个方法处理不同的请求
- 在每个方法上添加@RequestMapping('url')用于声明当前方法请求的url地址

```
package org.qf.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/user")
public class UserController {


    @RequestMapping("/login")
    public String login(){
        System.out.println("-------------------login--------------------------");
        return "success.jsp";
    }

    @RequestMapping("/list")
    public String showList(){
        System.out.println("-------------------");
        return "";
```

```
    }
}
```

3.1.2 访问

http://localhost:8080/user/list

http:///localhsot:8080/user/login

3.2 静态资源配置

> 静态资源：项目中的HTML CSS JS Jquery 图片 文字等

3.2.1 /* 和/区别

- /*拦截所有的HTTP请求，包括.jsp的请求，都作为控制器类的请求路径来处理。
- /拦截所有的HTTP请求，但是不包括.jsp的请求，不会放行静态资源的请求(图片 js)

3.2.2 静态资源放行配置

- 在springmvc的配置文件，添加如下静态资源放行的配置

```
    <mvc:resources mapping="/images/**" location="/images/"></mvc:resources>
    <mvc:resources mapping="/css/**" location="/css/"></mvc:resources>
    <mvc:resources mapping="/js/**" location="/js/"></mvc:resources>
```

3.3 前端提交数据到控制器

3.3.1 表单提交

- 表单提交：输入框需要提供name属性，Springmvc控制器是通过name属性获取值

```
<form action="/login">
        <p>
            用户名：<input type="text" name="username">
        </p>
        <p>
            密码： <input type="text" name="password">
        </p>
        <p>
            <input type="submit" value="登录">
        </p>
    </form>
```

3.3.2 URL提交

```
    <a href="/login?username=root&password=123">点我跳转</a>
```

3.3.3 ajax提交

异步请求

3.4 控制器接收前端提交的数据

3.4.1 @RequestParam

- 表单提交
- URL提交
- ajax提交

`@RequestParam` 注解用于接收请求传递的数据,也可以省略

```
@RequestMapping("/login")
    public String login(@RequestParam("username") String username,
@RequestParam("password") String password){
        System.out.println(username+"======================");

        System.out.println(password+"====================");

        //获取前端页面传递过来的数据
        //获取数据跟数据库中的数据进行对比，如果一致，说明登录成功，否则失败
        return "success.jsp";
    }
```

如果控制器方法中接收数据的参数名与请求传值的key一致，则@RequestParam()可以省略

3.5控制器响应前端请求

3.5.1 控制器响应同步请求

> 同步请求：表单 超链接
>
> - 处理同步请求的方法的返回值类型定义为String或者ModelAndView，以实现页面跳转

- 返回String类型

  转发

  ```
  @RequestMapping("/login")
      public String login(){
          return "/success.jsp";
      }
  ```

  重定向

```
@RequestMapping("/login")
 public String login(){

     return "redirect:/success.jsp";
 }
```

- 返回 ModelAndView

```
@RequestMapping("/login")
 public ModelAndView login(){
     ModelAndView modelAndView=new ModelAndView("/success.jsp");

     return modelAndView;
 }
```

3.5.2 控制器响应异步请求

> 异步请求：ajax axios

- 控制器方法的返回值类型为void/封装一个类，返回类/返回集合
- 在方法中添加@ResponseBody

```
    @RequestMapping("/list")
    @ResponseBody
    public List<String> showList(){
        System.out.println("--------------------");
        return new ArrayList<String>();
    }
```

注意：如果整个控制器类的方法都是异步请求，那么直接在控制器类上加`@RestController`

3.6 解决中文乱码的问题

3.6.1 前端编码

- jsp页面

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
```

- html页面

```
 <meta charset="UTF-8">
```

3.6.2 服务器编码

- tomcat/conf/server.xml

```
    <Connector port="8080" protocol="HTTP/1.1"
               connectionTimeout="20000" URIEncoding="UTF-8"
               redirectPort="8443" />
```

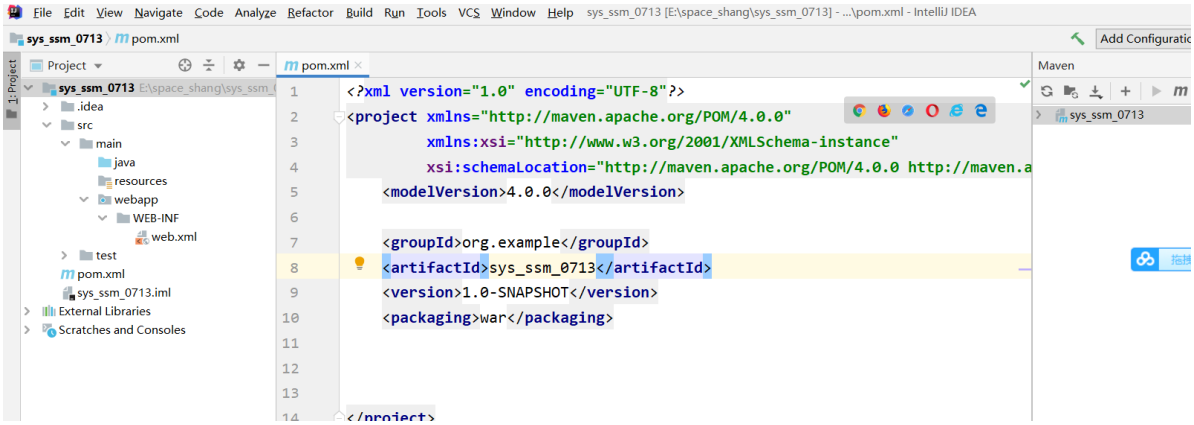
3.6.3 设置springmvc的编码方式

- 在web.xml中配置 springmvc编码过滤器的编码方式

```
 <filter>
        <filter-name>EncodingFilter</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>
    </filter>
    <filter-mapping>
        <filter-name>EncodingFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

4. SSM框架整合

4.1 创建 mavenweb项目



4.2 导入依赖

```
<dependencies>
        <!--mybatis-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis</artifactId>
            <version>3.5.10</version>
        </dependency>

        <!---lombok-->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
            <version>1.18.24</version>
        </dependency>

        <!---mysql-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.29</version>
        </dependency>

        <!--spring-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.2.16.RELEASE</version>
        </dependency>

        <!--spring-->
        <dependency>
            <groupId>org.mybatis</groupId>
            <artifactId>mybatis-spring</artifactId>
            <version>2.0.7</version>
        </dependency>

        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.2.16.RELEASE</version>
        </dependency>
```

```
        <!--日志包-->
        <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.17</version>
        </dependency>

        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.2</version>
        </dependency>

        <dependency>
            <groupId>org.slf4j</groupId>
            <artifactId>slf4j-log4j12</artifactId>
            <version>1.7.25</version>
        </dependency>

        <!--springmvc-->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.2.16.RELEASE</version>
        </dependency>

        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>druid</artifactId>
            <version>1.2.11</version>
        </dependency>

    </dependencies>
```

4.3 编写spring核心配置文件

- 在resources目录下 创建applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">


    <context:annotation-config></context:annotation-config>
    <context:component-scan base-package="org.qf"></context:component-scan>
    <!--1.加载数据库配置文件-->
    <context:property-placeholder location="classpath:db.properties" system-
properties-mode="NEVER"></context:property-placeholder>

    <!--2.数据源-->
    <bean id="dataSource" class="com.alibaba.druid.pool.DruidDataSource">
```

```
        <property name="driverClassName" value="${driver}"></property>
        <property name="url" value="${url}"></property>
        <property name="username" value="${username}"></property>
        <property name="password" value="${password}"></property>
    </bean>

    <!--3.配置sqlsessionFactory-->
    <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
        <property name="dataSource" ref="dataSource"></property>
        <property name="mapperLocations" value="classpath:mappers/*Mapper.xml">
</property>
        <property name="typeAliasesPackage" value="org.qf.entity"></property>
    </bean>

    <!--4.配置mapperscanner-->
    <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
        <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory">
</property>
        <property name="basePackage" value="org.qf.dao"></property>
    </bean>


</beans>
```

4.4 编写springmvc配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <context:annotation-config></context:annotation-config>
    <context:component-scan base-package="org.qf.web"></context:component-scan>

    <mvc:annotation-driven></mvc:annotation-driven>

</beans>
```

4.5 配置web.xml文件

```
<?xml version="1.0" encoding="UTF-8"?>
<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_4_0.xsd"
         version="4.0">

    <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:spring-servlet.xml</param-value>
        </init-param>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>classpath:applicationContext.xml</param-value>
    </context-param>

    <!--监听器-->
    <listener>
        <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
    </listener>

</web-app>
```