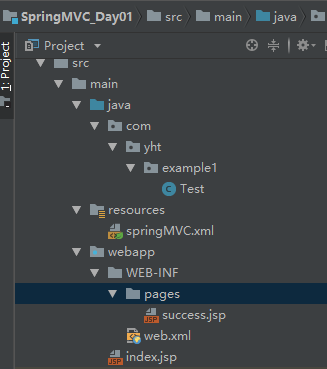
SpringMVC的简单示例程序：

首先，这个程序的主要功能是实现了通过点击JSP页面的超链接，然后跳转到新的的页面。

具体先来看一下这个工程的目录结构：



本工程是一个Maven工程，在pom文件中的配置信息主要如下：

|  |
| --- |
| <dependencies>  <dependency>  <groupId>junit</groupId>  <artifactId>junit</artifactId>  <version>4.11</version>  <scope>test</scope>  </dependency>  <!--日志-->  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-log4j12</artifactId>  <version>1.7.21</version>  </dependency>  <!--J2EE-->  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>javax.servlet-api</artifactId>  <version>3.1.0</version>  </dependency>  <dependency>  <groupId>javax.servlet.jsp</groupId>  <artifactId>jsp-api</artifactId>  <version>2.2</version>  </dependency>  <dependency>  <groupId>javax.servlet</groupId>  <artifactId>jstl</artifactId>  <version>1.2</version>  </dependency>  <!--mysql驱动包-->  <dependency>  <groupId>mysql</groupId>  <artifactId>mysql-connector-java</artifactId>  <version>5.1.35</version>  </dependency>  <!--springframework-->  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-web</artifactId>  <version>4.2.6.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-webmvc</artifactId>  <version>4.2.6.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>4.2.6.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-test</artifactId>  <version>4.2.6.RELEASE</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-jdbc</artifactId>  <version>4.2.6.RELEASE</version>  </dependency>  <dependency>  <groupId>com.github.stefanbirkner</groupId>  <artifactId>system-rules</artifactId>  <version>1.16.1</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.aspectj</groupId>  <artifactId>aspectjweaver</artifactId>  <version>1.8.9</version>  </dependency>  </dependencies> |

编写一个SpringMVC的程序，主要分为两步：

1. 启动服务器，加载一些配置文件。

这里的配置文件的信息主要包括：

在web.xml文件中指明拦截的路径以及拦截的类等信息。

|  |
| --- |
| <servlet>  <!--Servlet的别名，一般建议写为类名，首字母小写-->  <servlet-name>dispatcherServlet</servlet-name>  <!--类的路径-->  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!--加载的配置文件-->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springMVC.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <!--与上面的servlet-name保持一致-->  <servlet-mapping>  <servlet-name>dispatcherServlet</servlet-name>  <!--拦截的请求路径：/代表所有请求都拦截-->  <url-pattern>/</url-pattern>  </servlet-mapping> |

其他相关配置在第二步中作以阐述。

1. 发送请求，后台处理请求，并返回给客户端进行处理。

先来看一下代码：

首先是Test.java

|  |
| --- |
| package com.yht.example1;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.RequestMapping;  @Controller  public class Test {  //请求的url  @RequestMapping(path = "/hello")  public String sayHello(){  System.out.println("hello I am success!!!");  return "success";//这里返回的字符串，其实就是WEB-INF下pages文件下的success.jsp文件，因为已经在springMVC.xml文件中指明视图解析器的路径和文件后缀名，所以可以直接返回文件名  }  } |

springMVC.xml的配置：

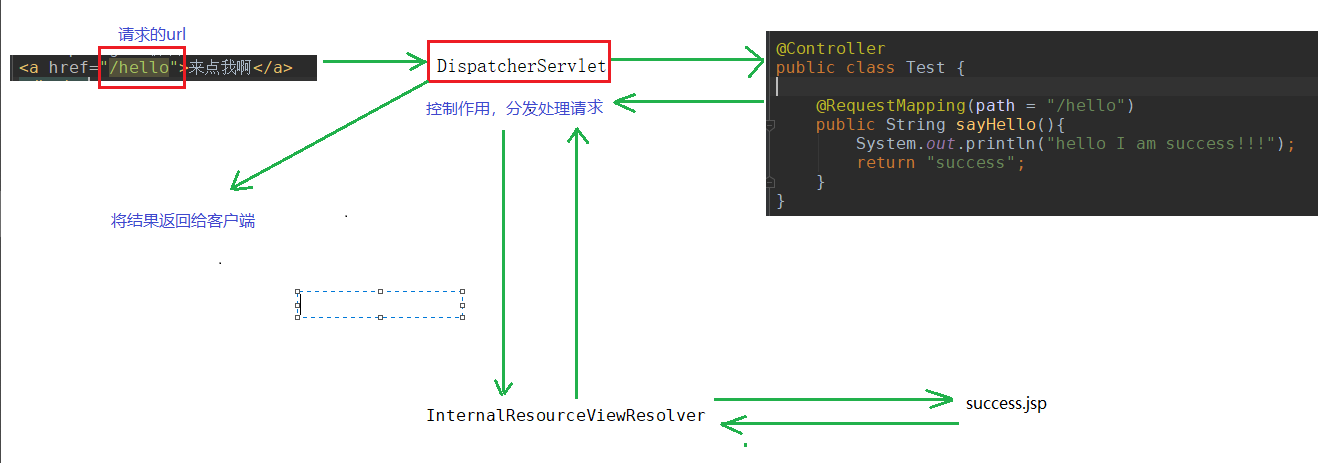
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xmlns:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd">  <!--配置注解要扫描得包-->  <context:component-scan base-package="com.yht.example1"></context:component-scan>  <!--配置视图解析器-->  <bean id="internalResourceViewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <!--文件所在的目录-->  <property name="prefix" value="/WEB-INF/pages/"></property>  <!--文件的后缀名-->  <property name="suffix" value=".jsp"></property>  </bean>  </beans> |

index.jsp

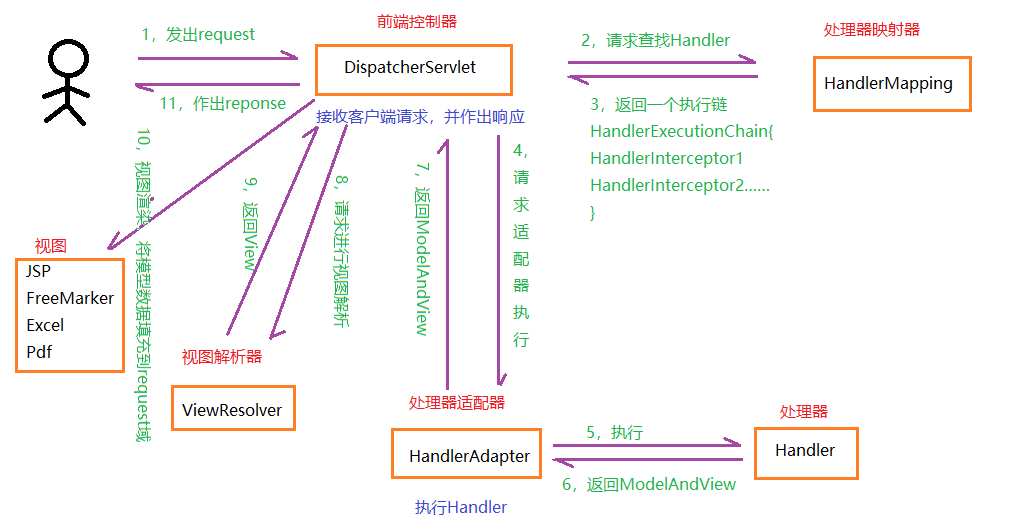
|  |
| --- |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>SpringMVC入门</title>  </head>  <body>  <h2>SpringMVC初识</h2>  <a href="/hello">来点我啊</a>  </body>  </html> |

success.jsp不在描述，只是一个跳转成功的页面。

根据以上代码，来简单看一下它的流程，具体如下图：



具体的整个SpringMVC的工作流程如下：



从上图不难发现，在SpringMVC的整个处理流程中，包含有五大组件，接下来对其加以说明：

1，DispatcherServlet：前端控制器

当用户的请求到达前端控制器，它就相当于mvc模式中的controller，它是整个流程控制的中心，由DispatcherServlet调用其他组件处理用户请求，它的存在降低了组件之间的耦合性。

2，HandlerMapping：处理器映射器

HandlerMapping负责根据用户请求找到Handler，即处理器。在SpringMVC中提供了不同的映射的实现方式，如：配置文件方式、实现接口方式、注解方式等。

1. Handler：处理器

Handler作为具体业务控制器，由DispatcherServlet把用户请求转发到Handler，由Handler对具体的用户请求进行处理。

1. HandlerADapter：处理器适配器

通过HandlerAdapter对处理器进行执行，这是适配器模式的应用，通过扩展适配器可以对更多类型的处理器进行执行。

1. View Resolver：视图解析器

View Resolver负责将处理结果生成View视图，View Resolver首先根据逻辑视图名解析成物理视图名，也就是具体的页面地址，再生成View视图对象，最后对View进行渲染，将处理结果展示给客户。

SpringMVC框架提供了很多的View视图类型的支持，包括：jstlView、freemarker、pdfView等，最常用的是jsp。

这里在提一个配置，如果在springMVC.xml中加入了<mvc:annotation-driven>，那么它可以自动加载RequestMappingHandlerMapping(处理器映射器)和RequestMappingHandlerAdapter(处理器适配器)，可用在配置文件中使用，替代注解处理器和适配器的配置。

RequestMapping的作用：用于建立请求URL和处理请求方法之间的对应关系，可以出现在类或者方法的前面。

上面的示例程序RequestMapping就是在方法的前面使用的。用于类上更有利于模块化开发，举个例子：前端页面有对用户注册和登录两个操作，那么对于用户模块，可以使用/user/login或者/user/register的请求URL的方式去对应user下的两个方法

|  |
| --- |
| @RequestMapping(path="/user")  public class User{  @RequestMapping(path="/register")  public String register(){  ……  return "register";  }  @RequestMapping(path="/login")  public String login(){  ……  return "login";  }  } |

常用属性：

value：用于指定请求的URL，其作用与path一样

path：用于指定请求的方式

method：用于指定限制请求参数的条件，支持简单的表达式，要求请求参数的key和vaLue必须和配置一模一样。超链接的请求方式是get，如果指定为post，那么无法跳转。

例如：

params = {“name”}，那么请求参数中必须带有name

params = {“name=tom”}，那么请求参数中必须带有name，且rname的值必须为tom

headers：用于指定限制请求消息头的条件---不常用，了解即可。