1. Spring集成web环境
2. ApplicationContext应用上下文获取方式

应用上下文对象是通过new ClassPathXmlApplicationContext(Spring配置文件)方式获取的，但是这样每次从容器中获取Bean对象时都要创建一次应用上下文对象，很麻烦，且十分低效。

因此，我们想到将应用上下文对象放到最大的域中共享。在Web项目中，可以使用ServletContextListener监听Web应用的启动，我们可以在Web项目启动时，就加在Spring的配置文件，创建应用上下文对象ApplicationContext，再将该对象存储到最大的域servletContext域中，这样既可以在任意位置从域中获取应用上下文对象ApplicationContext了。

1. 自定义监听器

自定义一个监听器监听服务器的启动，加载配置文件，存储app应用上下文对象。

public class ContextLoaderListener implements ServletContextListener{

public void contextInitialized(ServletContextEvent servletContextEvent){

ApplicationContext app = new ClassPathXmlApplicationContext(“applicationContext.xml”);

ServletContext servletContext = servletContextEvent.getServletContext();

servletContext.setAttribute(“app”, app);

}

}

1. Spring封装监听器

Spring提供了一个监听器ContextLoaderListener，该监听器加载了Spring配置文件，创建应用上下文对象，并存储到ServletContext域中，并且提供了一个工具WebApplicationContextUtils供使用者获取应用上下文对象。

这样一来，我们要做的事情就少了很多，只需要：

1. 在web.xml中配置ContextLoaderListener监听器，提供配置文件的名称给监听器，同时还要导入spring-web的坐标。

配置监听器的class：

<listener>

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

</listener>

还要在web.xml中提供配置文件的名称给监听器，用全局初始化参数来配置：

<!—全局初始化参数-->

<context-parm>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</context-parm>

注意，路径名前面要加上classpath:

1. 使用WebApplicationContextUtils获取应用上下文对象ApplicationContext。

ServletContext servletContext =this.getServletContext();

ApplicationContext app = WebApplicationContextUtils.getWebApplicationContext(servletContext);

1. SpringMVC
2. 概述

SpringMVC是一种基于Java的实现MVC设计模型的请求驱动类型的轻量级Web框架。

1. SpringMVC原理

客户端向服务器发送请求 ，服务器接收到了请求后，会封装代表请求的request对象，和代表响应的response对象，并调用服务器中相应的Servlet资源。

web层中Servlet的行为分为公有行为和特有行为，公有行为如接受数据，指定视图等等，特有行为指的是不同的特有功能，如表单数据的校验，Cookie的编写。因此，我们可以把公有行为抽取为一个前端控制器，SpringMVC框架中的前端控制器是一个Servlet。特有行为抽取为POJO（简单JavaBean对象），用于不同的功能。因此，服务器在请求资源时，先经过前端控制器使用一些共有的功能，再通过POJO来调用不同的功能。

1. SpringMVC的开发步骤
2. 导入SpringMVC的包

注意，要导入spring-webmvc和spring-web的坐标。

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-web</artifactId>  
 <version>5.0.5.RELEASE</version>  
</dependency>  
<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.0.5.RELEASE</version>  
</dependency>

1. SpringMVC使用Servlet作为前端控制器，因此要配置Servlet，即SpringMVC的核心DispatcherServlet。在web.xml中配置Servlet，像一般的Servlet一样进行配置，配置就是创建一个DispatcherServlet。

<servlet>

<servlet-name>DispatcherServlet</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>  
 <servlet-name>DispatcherServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
</servlet-mapping>

注意： url-pattern 取为“/”意思是所有的请求都要经过DiapatcherServlet

1. 编写POJO特有功能，在SpringMVC中，把POJO称为控制器Controller，控制每种功能特有的工作，如调用业务层，指派视图。

创建controller包，在包中创建各个功能类，如创建UserController类，返回success,jsp视图：

package com.springmvc.controller;  
  
  
public class UserController {  
  
 public String save(){  
 System.*out*.println("Controller save running");  
 return "success.jsp";  
 }  
}

1. 将Controller控制器，使用注解配置到Spring容器中，由于这是web层，因此可以使用@Controller。同时还要配置Controller类中业务方法的映射地址，因为请求需要知道要调用哪个Controller类中的哪个方法才能实现功能。在save方法上用注解@RequestMapping(“/quick”)，表示服务器在访问quick资源时就会映射到UserController类中的save方法了。

package com.springmvc.controller;  
  
import org.springframework.stereotype.Controller;  
import org.springframework.web.bind.annotation.RequestMapping;  
  
@Controller  
public class UserController {  
  
 @RequestMapping("/quick")  
 public String save(){  
 System.*out*.println("Controller save running");  
 return "success.jsp";  
 }  
}

1. 有了注解，就要配置组件扫描，使用context命名空间的component-scan，定义在Spring-mvc.xml文件进行配置，这是SpringMVC自己的配置文件。因此只需要设置base-package为controller包即可，service层，dao层的组件扫描可以在applicationContext.xml中配置。

<?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  
">  
  
<!-- Controller组件扫描-->  
 <context:component-scan base-package="com.springmvc.controller"/>  
  
</beans>

同时，还要让spring-mvc.xml这个配置文件加载到springMVC中，而SpringMVC的核心是DispatcherServlet，因此在配置Servlet时加载spring-mvc.xml，因此在web.xml中同样是使用全局初始化参数：

<servlet>  
 <servlet-name>DispatcherServlet</servlet-name>  
 <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring-mvc.xml</param-value>  
 </init-param>  
 <load-on-startup>1</load-on-startup>  
</servlet>  
<servlet-mapping>  
 <servlet-name>DispatcherServlet</servlet-name>  
 <url-pattern>/</url-pattern>  
</servlet-mapping>

1. SpringMVC的流程操作
2. 客户端发送请求到Tomcat引擎，该引擎接受请求，解析请求资源地址。
3. Tomcat引擎封装请求对象request，响应对象response，调用目标资源。
4. 通过Service(req, resp)方法获取请求资源的地址，映射访问真实资源，即Controller类中的方法。
5. 方法执行后，响应对象response封装成http响应返回给客户端。
6. SpringMVC的组件
7. SpringMVC的组件调用
8. 前端控制器DispatcherServlet只是一个调度器，并没有执行具体的功能，它只是调用SpringMVC中的不同组件。
9. 前端控制器通过请求查询Handler到HandlerMapping处理器映射器中查询要执行哪些资源，才能拿到请求的资源，处理器映射器返回的是处理器执行链HandlerExecutionChain，这是一连串的资源执行顺序。
10. 前端控制器拿到HandlerExecution处理器执行链后，再去通过请求执行Handler调用HandlerAdaptor处理器适配器，然后它会请求处理器Handler，即我们写的POJO（Controller）执行相应的操作。Handler返回响应给HandlerAdaptor，HandlerAdaptro返回模型和视图ModelAndView给前端控制器。
11. 前端控制器拿到ModelAndView之后，要展示视图，因此通过请求视图解析器ViewResolver，解析视图并返回View视图对象给前端控制器。
12. 前端控制器拿到View对象后，请求渲染视图页面，即将模型数据填充到视图中，并将渲染后的视图返回给客户端。
13. SpringMVC的注解解析
14. @RequestMapping

用于建立请求URL和Controller类中的处理请求方法之间的对应关系，一般放置在Controller类上或者方法上。

1. 放在Controller类上

当注解@RequestMapping()放在Controller类上时，代表这是请求URL的第一级访问目录，如果此处不写注解，那么就相当于应用的根目录。

如，@RequestMapping(“/first”)代表着请求URL的第一级访问目录为first，则在浏览器中输入请求URL时，就要先输入/first。

1. 放在方法上

@RequestMapping()注解放在方法上，代表着请求URL的第二级访问目录，与类上定义的一级访问目录一起组成了访问的虚拟路径。

如，@RequestMapping(“/second”)，如果没有第一级路径，则直接用localhost:8080/quick访问这个方法；如果类上的注解定义第一级路径为/first，则访问路径为 localhost:8080/first/second

1. 属性

value属性

用于指定请求的URL，就是上面所说的一级二级目录。

method属性

用于指定请求的方式，如GET或者POST，且只能够以这种方式请求访问，如果以其他的方式则会报错。

如，method=RequestMethod.GET

params属性

params用于指定限制请求参数的条件，可以限制必须有什么参数或者不能有什么参数。

如，params={“accountName”}表示请求参数中必须有accountName

params={“money!100”}表示请求参数中money不能是100

1. SpringMVC的XML配置解析
2. SpringMVC有默认的组件配置，默认组件都是在DispatcherServlet.properties配置文件中配置的，该配置文件的地址是 org.springframework.web.servlet.DispatcherServlet.properties，该文件中配置了默认的视图解析器， 如：

org.springframework.web.servlet.ViewResolver=org.springframework.web.servlet.view.InternalResourceViewResovler

要想配置内部资源解析器，通过在spring-mvc.xml文件中创建一个bean对象，设置资源的前缀prefix和后缀suffix，如设置解析前缀为/jsp/，后缀为.jsp的视图。

<bean id=”viewResolver” class=”org.springframework.web.servlet.view.InternalResourceViewResolver”>

<property name=”prefix” value=”/jsp/”></property>

<property name=”suffix” value=”.jsp”></property>

</bean>