Java嵌入式tomcat整合SpringMVC

原创 陈虎_63 于 2021-12-02 12:12:48 发布 阅读量1.1k 收藏 3 点赞数2 版权

分类专栏： spring Java 文章标签： tomcat java spring boot

前言：

本文将介绍如何使用Java的方式启动tomcat，并整合Spring MVC，做到就像Springboot使用main方法启动，就可以访问controller资源的效果；

首先导入依赖：

```
<properties>
        <embed.tomcat.version>9.0.21</embed.tomcat.version>
    </properties>

    <dependencies>

        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-core</artifactId>
            <version>${embed.tomcat.version}</version>
        </dependency>

        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
            <version>${embed.tomcat.version}</version>
            <!--<scope>provided</scope>-->
        </dependency>


        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.0.8.RELEASE</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.0.8.RELEASE</version>
        </dependency>
    </dependencies>
```

创建AppConfig配置类：

```
package com.hu.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

/**
 * @program: zdy-spring-boot
 * @description:
 * @author: hu.chen
 * @createDate: 2021年12月01日 22:02
 **/
@Configuration
//添加包扫描路径
@ComponentScan({"com.hu"})
public class AppConfig {


```

```
}
```

创建 MyWebApplicationInitializer 类实现 WebApplicationInitializer接口，重写onstartup方法

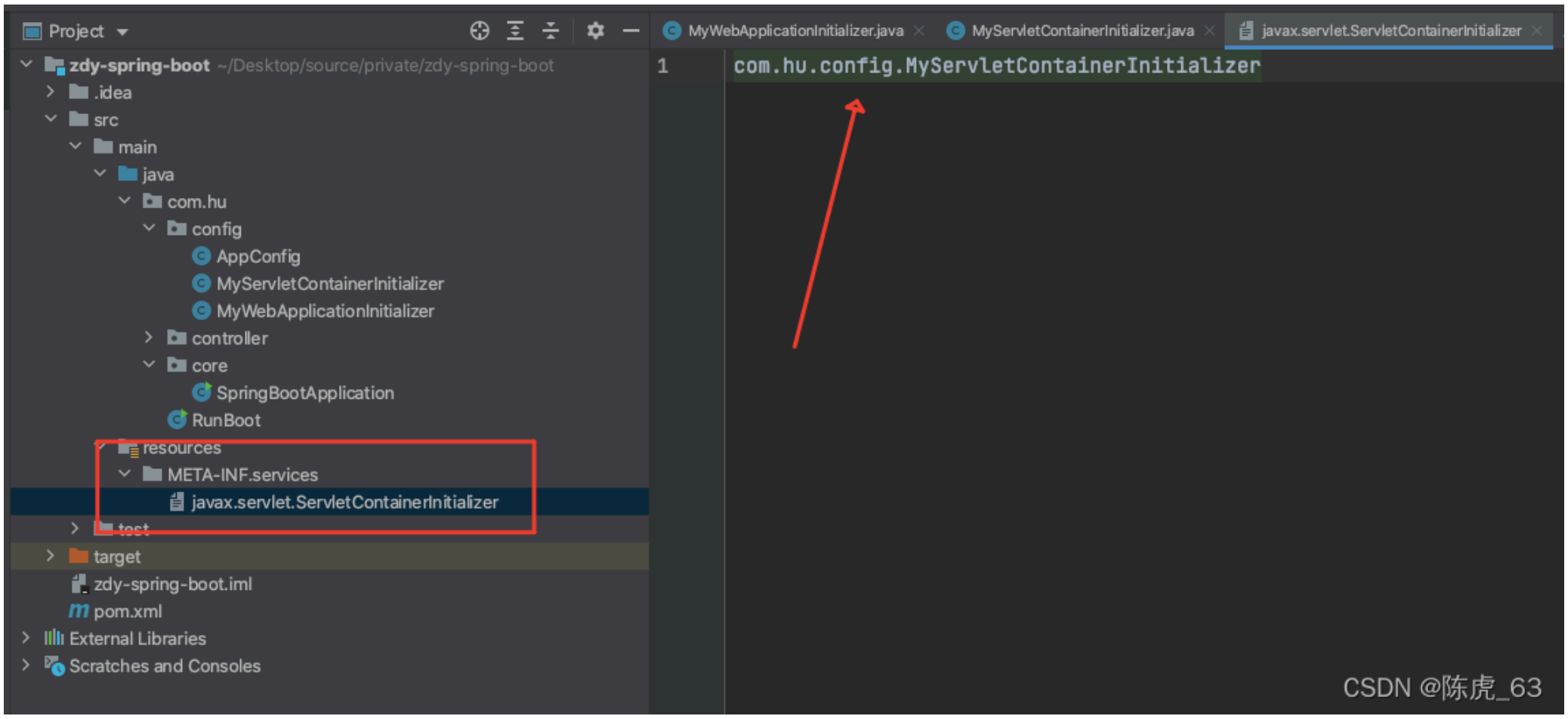
```
package com.hu.config;

import org.springframework.web.WebApplicationInitializer;
import org.springframework.web.context.support.AnnotationConfigWebApplicationContext;
import org.springframework.web.servlet.DispatcherServlet;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRegistration;

/**
 * @program: zdy-spring-boot
 * @description:
 * @author: hu.chen
 * @createDate: 2021年12月01日 21:59
 **/
public class MyWebApplicationInitializer implements WebApplicationInitializer {



    public void onStartup(ServletContext servletContext) throws ServletException {
        //通过注解的方式初始化Spring的上下文
        AnnotationConfigWebApplicationContext ac = new AnnotationConfigWebApplicationContext();

        //注册spring的配置类（替代传统项目中xml的configuration）
        ac.register(AppConfig.class);
        ac.refresh();


        //基于java代码的方式初始化DispatcherServlet
        DispatcherServlet servlet = new DispatcherServlet(ac);
        //绑定servlet
        ServletRegistration.Dynamic registration = servletContext.addServlet("dispatcherServlet", servlet);

        //设置tomcat启动立即加载 servlet
        registration.setLoadOnStartup(1);
        //浏览器访问uri
        registration.addMapping("/app/*");
    }
}

```

创建类MySpringServletContainerInitializer 并实现 ServletContainerInitializer 重写onstartup方法

```
package com.hu.config;


import javax.servlet.ServletContainerInitializer;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import java.util.Set;

/**
 * @program: zdy-spring-boot
 * @description:
 * @author: hu.chen
 * @createDate: 2021年12月01日 21:46
 **/
public class MyServletContainerInitializer implements ServletContainerInitializer {



    /**

```

```
     * @param set Servlet 3.0+容器启动时将自动扫描类路径以查找实现Spring的Webapplicationinitializer接口的所有实现，
     *      将其放进一个Set集合中，提供给ServletContainerInitializer中onStartup方法的第一个参数。
     * @param servletContext
     * @throws ServletException
     */
    public void onStartup(Set<Class<?>> set, ServletContext servletContext) throws ServletException {
        System.out.println("加载.......");
    }
}
```

在resources目录下，创建META-INF/services/javax.servlet.ServletContainerInitializer
在文件中添加 MyServletContainerInitializer 类的全限定名，servlet3.0规范，规定了tomcat在启动时会去扫描项目包括项目的jar包下这个（META-INF/services）目录下的 javax.servlet.ServletContainerInitializer 这个文件，加载这个文件中配置的类的全限定名，实例化并调用这个类中的onstartup方法（所以这个类必须实现ServletContainerInitializer这个接口）



创建启动类在main方法中实例化tomcat并启动

```
package com.hu.core;

import org.apache.catalina.LifecycleException;
import org.apache.catalina.connector.Connector;
import org.apache.catalina.startup.Tomcat;

/**
 * @program: zdy-spring-boot
 * @description:
 * @author: hu.chen
 * @createDate: 2021年12月01日 22:26
 **/
public class SpringBootApplication {

    private static int port = 8080;
    private static String contextPath = "/";


    public static void run(){
        Tomcat tomcat = new Tomcat();
        String baseDir = Thread.currentThread().getContextClassLoader().getResource("").getPath();
        //设置tomcat启动后的工作目录
        tomcat.setBaseDir(baseDir);
        //设置端口
        tomcat.setPort(port);
        //获取执行器，并设置io协议
        Connector connector = new Connector("org.apache.coyote.http11.Http11NioProtocol");
        //设置端口
```

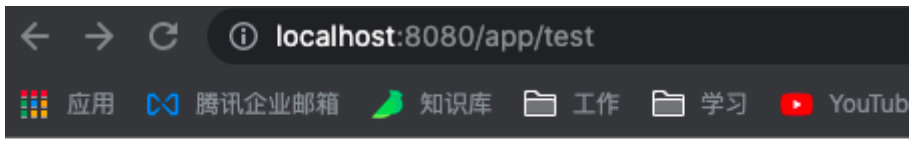
```
        connector.setPort(port);
        //设置执行器
        tomcat.setConnector(connector);

        tomcat.addWebapp(contextPath, baseDir);
        tomcat.enableNaming();
        try {
            tomcat.start();
        } catch (LifecycleException e) {
            System.err.println("tomcat 启动失败");
        }
        //tomcat启动后，让其阻塞，不让当前线程结束，等待处理请求，
        tomcat.getServer().await();
    }

    public static void main(String[] args) {
        run();
    }
}


```

创建controller：

```
package com.hu.controller;

import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

/**
 * @program: zdy-spring-boot
 * @description:
 * @author: hu.chen
 * @createDate: 2021年12月01日 22:23
 **/
@RestController
public class TestController {


    @GetMapping("/test")
    public String test(){

        return "hello";
    }
}

```

浏览器访问：

http://localhost:8080/app/test



hello