

//web容器启动的时候创建对象；调用方法来初始化容器以前前端控制器

Publicclass MyWebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

//获取根容器的配置类；（Spring的配置文件） 父容器；

//spring配置类

@Override

protected Class<?>[] getRootConfigClasses() {

// TODO Auto-generated method stub

return new Class<?>[]{RootConfig.class};

}

//获取web容器的配置类（SpringMVC配置文件） 子容器；

//springMVC配置类

@Override

protected Class<?>[] getServletConfigClasses() {

// TODO Auto-generated method stub

return new Class<?>[]{AppConfig.class};

}

//获取DispatcherServlet的映射信息

// /：拦截所有请求（包括静态资源（xx.js,xx.png）），但是不包括\*.jsp；

// /\*：拦截所有请求；连\*.jsp页面都拦截；jsp页面是tomcat的jsp引擎解析的；

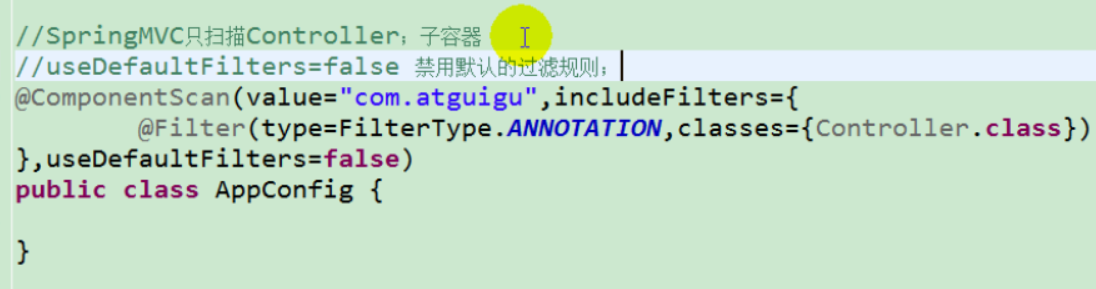
@Override

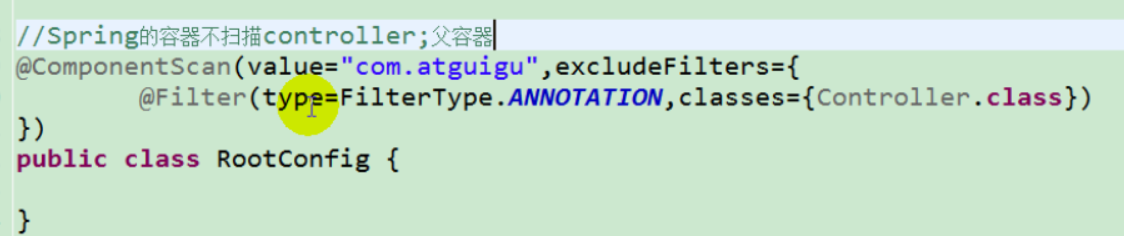
protected String[] getServletMappings() {

// TODO Auto-generated method stub

return new String[]{"/"};

}





定制springMVC：

查看官网文档：

<https://docs.spring.io/spring/docs/5.0.12.BUILD-SNAPSHOT/spring-framework-reference/web.html#mvc-config-enable>

异步请求：

可以参考上面的文档

@ResponseBody

@RequestMapping("/createOrder")

public DeferredResult<Object> createOrder(){

DeferredResult<Object> deferredResult = new DeferredResult<>((long)3000, "create fail...");

DeferredResultQueue.save(deferredResult);

return deferredResult;

}

@ResponseBody

@RequestMapping("/create")

public String create(){

//创建订单

String order = UUID.randomUUID().toString();

DeferredResult<Object> deferredResult = DeferredResultQueue.get();

deferredResult.setResult(order);

return "success===>"+order;

}

创建完订单之后，在上面返回。

