Spring Boot

- Servlet / Filter / Listener / 拦截器

by万夜

Java新兵

扫一扫二维码，加入该群。

欢迎大家加入QQ群“Java新兵”，参与讨论。

群号:654287735

目录

- 注册servlet的两种方式
- 实现servlet/filter/listener/拦截器
- 注解含义
- 总结

注册Servlet方式

- **通过代码注册**

  - 代码注册通过ServletRegistrationBean、FilterRegistrationBean 和 ServletListenerRegistrationBean 获得控制。

- **注解自动注册**

  - 在 SpringBootApplication 上使用@ServletComponentScan 注解后，Servlet、Filter、Listener 可以直接通过 @WebServlet、@WebFilter、@WebListener 注解自动注册，无需其他代码。

通过代码注册Servlet

- 1. 创建servlet

```
public class HelloServlet extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
        System.out.println(">>doGet<<");
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        System.out.println(">>doPost<<");
        resp.setContentType("text/html;charset=utf-8");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<head><title>Hello 小红</title></head>");
        out.println("<body>");
        out.println("Hello 小红");
        out.println("</body>");
        out.println("</html>");
    }
}
```

- 2. 注册servlet到spring

```
@Bean
public ServletRegistrationBean servletRegistrationBean() {
    return new ServletRegistrationBean(new HelloServlet(),"/xiaohong");
}
```

通过注解注册Servlet

1. 增加注解，开启servlet扫描

```
@SpringBootApplication
@ServletComponentScan //
public class Start {
    public static void main(String[] args) {
        SpringApplication.run(Start.class, args);
    }
}
```

2. 增加注解，标识该类是servlet，并声明urlPath

```
@WebServlet("/xiaohong1") //
public class HelloServlet1 extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println(">>doGet<<");
        doPost(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        System.out.println(">>doPost<<");
        resp.setContentType("text/html;charset=utf-8");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<head><title>Hello 小红</title></head>");
        out.println("<body>");
        out.println("Hello 小红");
        out.println("</body>");
        out.println("</html>");
    }
}
```

通过注解创建Filter

实现功能：过滤所有请求，判断请求参数中是否包含”key”，同时”key”==“xiaohong”，如不包含，则认为是非法请求，返回“param error”，如合法则继续访问。

- 1. 增加注解@ServletComponentScan，开启servlet扫描
- 2. 增加注解@WebFilter，标识该类是Filter

```
@WebFilter
public class HelloFilter implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException {
        System.out.println(">>filter init<<");
    }

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain) throws IOException,
ServletException {
        System.out.println(">>filter done<<");
        String key = servletRequest.getParameter("key");
        if (null != key && "xiaohong".equals(key)) {
            System.out.println(">>filter match<<");
            filterChain.doFilter(servletRequest, servletResponse);
        } else {
            System.out.println("filter param error");
            PrintWriter out = servletResponse.getWriter();
            out.print("param error");
            out.close();
        }
    }

    @Override
    public void destroy() {
        System.out.println(">>filter destroy<<");
    }
}
```

通过注解创建Listener

实现功能：系统启动时，加载请求参数配置”key”==”xiaoming”，并在过滤器中进行合法参数验证。

1. 增加注解@ServletComponentScan，开启servlet扫描
2. 增加注解@WebListener，标识该类是Listener

```
@WebListener
public class HelloServletListener implements ServletContextListener {
    @Override
    public void contextInitialized(ServletContextEvent servletContextEvent) {
        // 配置key==xiaoming
        servletContextEvent.getServletContext().setAttribute(“key","xiaoming");
        System.out.println(">>context listener init<<");
    }

    @Override
    public void contextDestroyed(ServletContextEvent servletContextEvent) {
        System.out.println(">>context listener destroyed<<");
    }
}
```

```
修改HelloFilter，动态加载key
String _key = (String) servletRequest.getServletContext().getAttribute("key");
if (null != key && _key.equals(key)) {
```

创建http拦截器

- 1. 创建拦截器类并实现 HandlerInterceptor接口

```
public class HelloInterceptor implements HandlerInterceptor {
   @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        System.out.println(">>interceptor preHandle<<");
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView
modelAndView) throws Exception {
        System.out.println(">>interceptor postHandle<<");
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex)
throws Exception {
        System.out.println(">>interceptor afterCompletion<<");
    }
}
```

- 2. 创建一个Java类继承WebMvcConfigurerAdapter，并重写 addInterceptors 方法，@Configuration

- 3. 实例化我们自定义的拦截器，然后将对像手动添加到拦截器链中（在addInterceptors方法中添加）

```
@Configuration
public class HelloConfig extends WebMvcConfigurerAdapter {
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(new HelloInterceptor()).addPathPatterns("/**");
    }
}
```

注解含义

```
@Bean // @Bean明确地指示了产生一个bean的方法，并且交给Spring容器管理
```

```
@ServletComponentScan // 当使用@ServletComponentScan扫描Servlet组件时，Servlet、过滤器和监听器可以是通过@WebServlet、@WebFilter和@WebListener自动注册
```

```
@WebServlet(“/hello")
```

```
@WebFilter
```

```
@WebListener
```

总结

- 注册Servlet/Filter/Listener的两种方式
- 注解
    - @Bean
    - @ServletComponenScan
        - @WebServlet/@WebFilter/@WebListener
- 通过注解注册servlet/filter/listener
- 两个功能：
    - 过滤器验证所有请求参数
    - 监听器，初始化验证规则
- 创建拦截器