过滤器、拦截器、监听器、自定义监听器

拦截器

是springmvc提供了一个拦截器的机制，它专门用于拦截controller的路由请求。

它的本质是：AOP面向切面的编程，也就是说符合横切关注点的功能都可以考虑使用拦截器实现。比如一些应用场景：

- 权限检查

  例如：用户登录检查，访问项目的内部接口时，可以通过拦截器检测用户是否登录，如果登录，直接放回用户登录页面

  比如：接口的安全校验，使用JWT做权限拦截器校验。

- 日志记录

  更新推荐用原生的AOP机制会更好一点，粒度会更细，控制起来也更方便，

  如果你是针对某个接口或者某个请求，或者某个业务针对性的记录日志，其实也可以考虑用拦截器来完成。

- 性能监控

  记录接口访问过程中的开始时间和结束时间的处理机制。

- 通用行为

  比如获取cookie信息，获取用户信息，并将用户信息存放到请求头中，方便后续业务的使用。

02、自定义拦截器

在springboot中定义拦截器需要两个关键的步骤：

- 创建一个拦截器类实现一个接口HandlerInterceptor
- 创建一个springmvc的拦截器的配置类实现WebMvcConfigurer的接口，并且重写addInterceptors()的方法，注册拦截器类，并且为当前拦截器类定义规则

实现拦截器的步骤：

1：引入web依赖

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
</dependency>
```

2：开始定义拦截器类LoginInterceptor

```
package com.kuangstudy.config.handler;

import lombok.extern.slf4j.Slf4j;
import org.springframework.web.servlet.HandlerInterceptor;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦！
```

```
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/12/27 20:43
 */

@Slf4j
public class LoginInterceptor implements HandlerInterceptor {


    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        log.info("1-----preHandle----->");
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {
        log.info("3-----postHandle----->");
    }

    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {
        log.info("4-----afterCompletion----->");
    }
}

```

3：定义拦截器配置类

```
package com.kuangstudy.config.mvc;

import com.kuangstudy.config.handler.LoginInterceptor;
import org.springframework.context.annotation.Bean;
```

```java
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2021/12/27 20:45
 */
@Configuration
public class WebMvcConfiguration implements WebMvcConfigurer {


    @Bean
    // 初始化拦截器放入到ioc容器中
    public LoginInterceptor getLoginInterceptor(){
        return new LoginInterceptor();
    }

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry
                // 1:  拦截器注册
                .addInterceptor(getLoginInterceptor())
                // 2: 给拦截器配置并且定义规则
                .addPathPatterns("/api/**");
    }
}
```

```
36
```

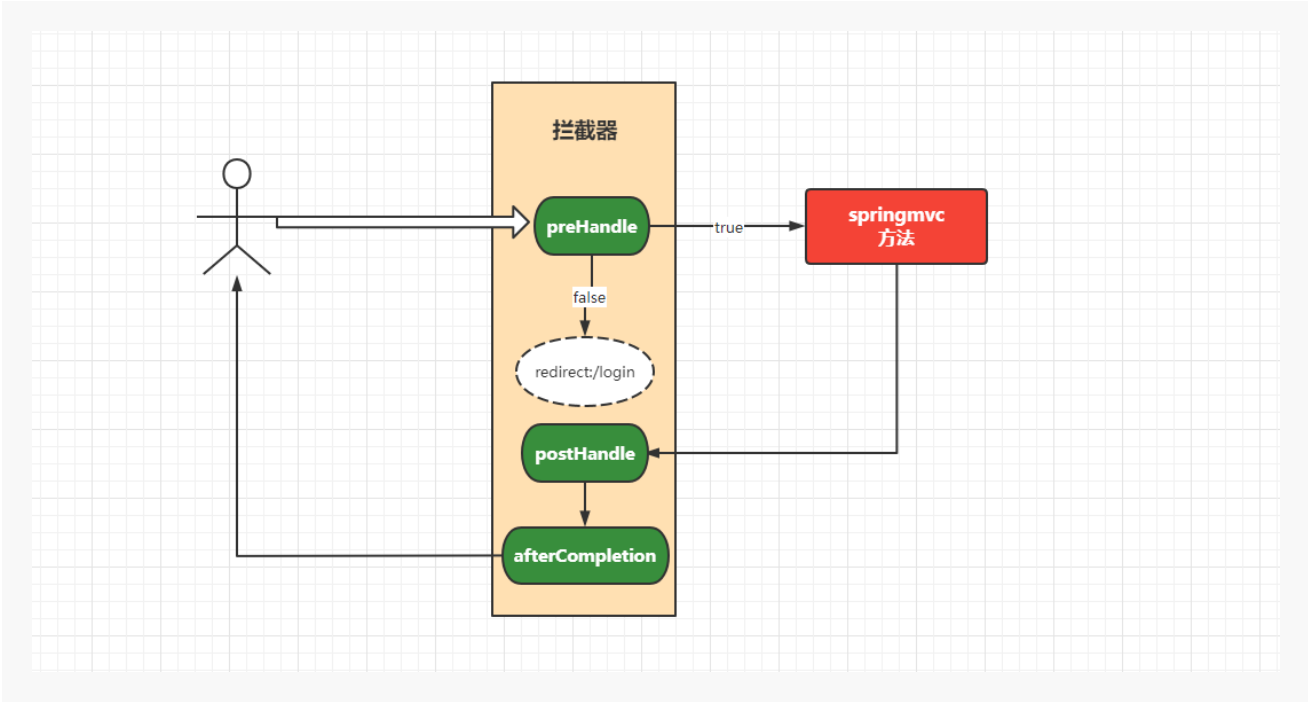
4：测试

```
1  package com.kuangstudy.web;
2
3  import org.springframework.stereotype.Controller;
4  import org.springframework.web.bind.annotation.GetMapping;
5
6  /**
7   * Description:
8   * Author: yykk Administrator
9   * Version: 1.0
10  * Create Date Time: 2021/12/11 21:25.
11  * Update Date Time:
12  *
13  * @see
14  */
15 @Controller
16 public class HelloWorldController {
17
18     @GetMapping("/api/index")
19     public String index() {
20         return "index";
21     }
22 }
23
```

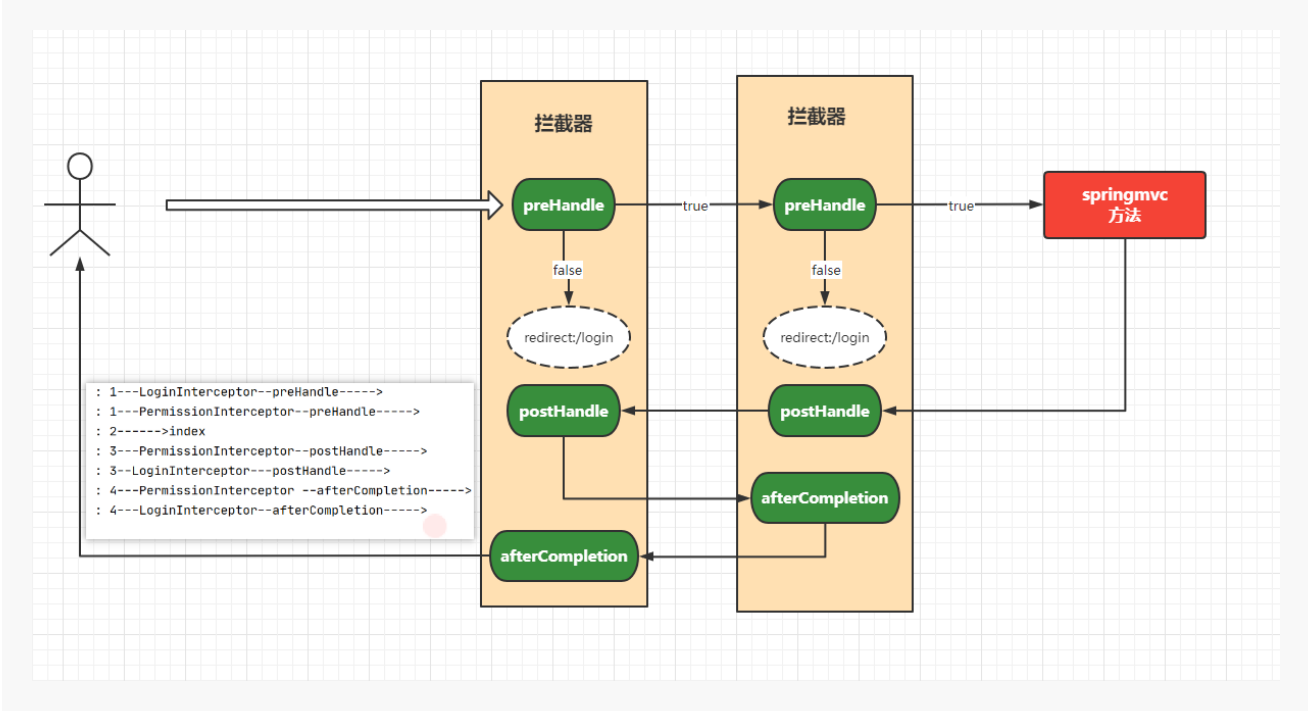
当用户在浏览器访问：

http://localhost:8987/api/index

执行流程如下：



多个拦截器的执行顺序是怎么样的呢？



规则就是：所以的prehandle执行完毕 --- postHandle ----afterCompletion----返回给用户