链滴

SpringCloud 系列 --3. 负载均衡 @LoadBalanced 注解原理

作者：289306290

原文链接：https://ld246.com/article/1584005098213

来源网站：链滴

restTemplate本身不具有负载均衡原理,但是加入了@LoadBalanced注解后具有负载功能，得益于RestTemplate的拦截器功能.

我们模拟实现一个自定义的@LoadBalanced注解 @MyLoadBalanced

```
package org.crazyit.cloud;

import java.lang.annotation.ElementType;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

import org.springframework.beans.factory.annotation.Qualifier;

@Target({ ElementType.FIELD, ElementType.PARAMETER, ElementType.METHOD })
@Retention(RetentionPolicy.RUNTIME)
@Qualifier
public @interface MyLoadBalanced {

}
```

然后实现org.springframework.http.HttpRequest 接口，将原先的URI进行重写，所有的请求转发到ttp://localhost:8080/hello这个地址. SpringCloud对RestTemplate进行拦截时候也做了同样的事情只不过更加灵活的处理,并不是像我们返回固定的URI。

```
package org.crazyit.cloud;

import java.net.URI;

import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.HttpRequest;

/**
 * 自定义的请求类，用于转换URI
 * @author 杨恩雄
 *
 */
public class MyHttpRequest implements HttpRequest {

    private HttpRequest sourceRequest;

    public MyHttpRequest(HttpRequest sourceRequest) {
        this.sourceRequest = sourceRequest;
    }

    public HttpHeaders getHeaders() {
        return sourceRequest.getHeaders();
    }

    public HttpMethod getMethod() {
```

```
        return sourceRequest.getMethod();
    }

    /**
     * 将URI转换
     */
    public URI getURI() {
        try {
            URI newUri = new URI("http://localhost:8080/hello");
            return newUri;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return sourceRequest.getURI();
    }
}
```

然后我们自定义拦截器, 将原先请求使用我们刚才定义的MyHttpRequest进行转换.

```
package org.crazyit.cloud;

import java.io.IOException;

import org.springframework.http.HttpRequest;
import org.springframework.http.client.ClientHttpRequestExecution;
import org.springframework.http.client.ClientHttpRequestInterceptor;
import org.springframework.http.client.ClientHttpResponse;

/**
 * 自定义拦截器
 *
 * @author 杨恩雄
 *
 */
public class MyInterceptor implements ClientHttpRequestInterceptor {

    public ClientHttpResponse intercept(HttpRequest request, byte[] body,
            ClientHttpRequestExecution execution) throws IOException {
        System.out.println("==============  这是自定义拦截器实现");
        System.out.println("              原来的URI：" + request.getURI());
        // 换成新的请求对象（更换URI）
        MyHttpRequest newRequest = new MyHttpRequest(request);
        System.out.println("              拦截后新的URI：" + newRequest.getURI());
        return execution.execute(newRequest, body);
    }
}
```

配置在容器启动的时候将我们的自定义拦截器MyInterceptor 加入到RestTemplate实例中

```
package org.crazyit.cloud;

import java.lang.annotation.ElementType;
```

```
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

import org.springframework.beans.factory.SmartInitializingSingleton;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.client.ClientHttpRequestInterceptor;
import org.springframework.web.client.RestTemplate;


@Configuration
public class MyAutoConfiguration {

    @Autowired(required=false)
    @MyLoadBalanced
    private List<RestTemplate> myTemplates = Collections.emptyList();

    @Bean
    public SmartInitializingSingleton myLoadBalancedRestTemplateInitializer() {
        System.out.println("====  这个Bean将在容器初始化时创建    =====");
        return new SmartInitializingSingleton() {

            public void afterSingletonsInstantiated() {
                for(RestTemplate tpl : myTemplates) {
                    // 创建一个自定义的拦截器实例
                    MyInterceptor mi = new MyInterceptor();
                    // 获取RestTemplate原来的拦截器
                    List list = new ArrayList(tpl.getInterceptors());
                    // 添加到拦截器集合
                    list.add(mi);
                    // 将新的拦截器集合设置到RestTemplate实例
                    tpl.setInterceptors(list);
                }
            }
        };
    }
}
```

最后使用我们的注解@MyLoadBalanced

```
package org.crazyit.cloud;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
```

```
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.client.RestTemplate;

@RestController
@Configuration
public class InvokerController {

    @Bean
    @MyLoadBalanced
    public RestTemplate getMyRestTemplate() {
        return new RestTemplate();
    }

    /**
     * 浏览器访问的请求
     */
    @RequestMapping(value = "/router", method = RequestMethod.GET,
            produces = MediaType.APPLICATION_JSON_VALUE)
    public String router() {
        RestTemplate restTpl = getMyRestTemplate();
        // 根据名称来调用服务，这个URI会被拦截器所置换
        String json = restTpl.getForObject("http://my-server/hello", String.class);
        return json;
    }

    /**
     * 最终的请求都会转到这个服务
     */
    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    @ResponseBody
    public String hello() {
        return "Hello World";
    }
}
```

启动类

```
package org.crazyit.cloud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TestInterceptorMain {

    public static void main(String[] args) {
        SpringApplication.run(TestInterceptorMain.class, args);
    }

}
```

最后访问http://localhost:8080/router 本来访问 http://my-server/hello
但是现在不管my-server原先是哪个地址都会转发到localhost:8080/hello这里。