链滴

HttpClient&RestTemplate

作者：Weixl

原文链接：https://ld246.com/article/1593566144583

来源网站：链滴

HttpClient

- HttpClient 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，
- HttpClient不是一个浏览器。它是一个客户端的HTTP通信实现库。HttpClient的目标是发送和接收TTP报文。HttpClient不会去缓存内容，执行嵌入在HTML页面中的javascript代码，猜测内容类型，新格式化请求/重定向URI，或者其它和HTTP运输无关的功能。
- 应用场景：
  - 在一个java服务器中，访问另一个java服务器，获取返回的资源

依赖坐标

```
<dependency>
    <groupId>org.apache.httpcomponents</groupId>
    <artifactId>httpclient</artifactId>
    <version>4.4</version>
</dependency>
```

springboot整合HttpClient使用

- 配置类：

```
#The config for HttpClient
http.maxTotal=300
http.defaultMaxPerRoute=50
http.connectTimeout=1000
http.connectionRequestTimeout=500
http.socketTimeout=5000
```

```
http.staleConnectionCheckEnabled=true
```

- 配置类：

```
list = template.getMessageConverters();
for (HttpMessageConverter? mc : list) {
if (mc instanceof StringHttpMessageConverter) {
((StringHttpMessageConverter) mc).setDefaultCharset(Charset.forName("UTF-8"));
}
}
return template;
}
```

```
public Integer getMaxTotal() {
    return maxTotal;
}

public void setMaxTotal(Integer maxTotal) {
    this.maxTotal = maxTotal;
}

public Integer getDefaultMaxPerRoute() {
    return defaultMaxPerRoute;
}

public void setDefaultMaxPerRoute(Integer defaultMaxPerRoute) {
    this.defaultMaxPerRoute = defaultMaxPerRoute;
}

public Integer getConnectTimeout() {
    return connectTimeout;
}

public void setConnectTimeout(Integer connectTimeout) {
    this.connectTimeout = connectTimeout;
}

public Integer getConnectionRequestTimeout() {
    return connectionRequestTimeout;
}

public void setConnectionRequestTimeout(Integer connectionRequestTimeout) {
    this.connectionRequestTimeout = connectionRequestTimeout;
}

public Integer getSocketTimeout() {
    return socketTimeout;
}

public void setSocketTimeout(Integer socketTimeout) {
    this.socketTimeout = socketTimeout;
```

```
  }
}
">

  /**
   * HttpClient的配置类
   *
   */
  @Configuration
  @ConfigurationProperties(prefix = "http", ignoreUnknownFields = true)
  public class HttpClientConfig {

      private Integer maxTotal;// 最大连接

      private Integer defaultMaxPerRoute;// 每个host的最大连接

      private Integer connectTimeout;// 连接超时时间

      private Integer connectionRequestTimeout;// 请求超时时间

      private Integer socketTimeout;// 响应超时时间

      /**
       * HttpClient连接池
       * @return
       */
      @Bean
      public HttpClientConnectionManager httpClientConnectionManager() {
          PoolingHttpClientConnectionManager connectionManager = new PoolingHttpClientCo
nectionManager();
          connectionManager.setMaxTotal(maxTotal);
          connectionManager.setDefaultMaxPerRoute(defaultMaxPerRoute);
          return connectionManager;
      }

      /**
       * 注册RequestConfig
       * @return
       */
      @Bean
      public RequestConfig requestConfig() {
          return RequestConfig.custom().setConnectTimeout(connectTimeout)
              .setConnectionRequestTimeout(connectionRequestTimeout).setSocketTimeout(socke
Timeout)
              .build();
      }

      /**
       * 注册HttpClient
       * @param manager
       * @param config
       * @return
       */
```

```
    @Bean
    public HttpClient httpClient(HttpClientConnectionManager manager, RequestConfig conf
g) {
        return HttpClientBuilder.create().setConnectionManager(manager).setDefaultRequestC
nfig(config)
                .build();
    }
    /**
     * 使用连接池管理连接
     * @param httpClient
     * @return
     */
    @Bean
    public ClientHttpRequestFactory requestFactory(HttpClient httpClient) {
        return new HttpComponentsClientHttpRequestFactory(httpClient);
    }
    /**
     * 使用HttpClient来初始化一个RestTemplate
     * @param requestFactory
     * @return
     */
    @Bean
    public RestTemplate restTemplate(ClientHttpRequestFactory requestFactory) {
        RestTemplate template = new RestTemplate(requestFactory);

        List<HttpMessageConverter<?>> list = template.getMessageConverters();
        for (HttpMessageConverter<?> mc : list) {
            if (mc instanceof StringHttpMessageConverter) {
                ((StringHttpMessageConverter) mc).setDefaultCharset(Charset.forName("UTF-8"));
            }
        }
        return template;
    }

    public Integer getMaxTotal() {
        return maxTotal;
    }

    public void setMaxTotal(Integer maxTotal) {
        this.maxTotal = maxTotal;
    }

    public Integer getDefaultMaxPerRoute() {
        return defaultMaxPerRoute;
    }

    public void setDefaultMaxPerRoute(Integer defaultMaxPerRoute) {
        this.defaultMaxPerRoute = defaultMaxPerRoute;
    }

    public Integer getConnectTimeout() {
        return connectTimeout;
    }
```

原文链接：HttpClient&RestTemplate

```
    public void setConnectTimeout(Integer connectTimeout) {
        this.connectTimeout = connectTimeout;
    }

    public Integer getConnectionRequestTimeout() {
        return connectionRequestTimeout;
    }

    public void setConnectionRequestTimeout(Integer connectionRequestTimeout) {
        this.connectionRequestTimeout = connectionRequestTimeout;
    }

    public Integer getSocketTimeout() {
        return socketTimeout;
    }

    public void setSocketTimeout(Integer socketTimeout) {
        this.socketTimeout = socketTimeout;
    }
}
```

总结

- 总结：HttpClient使用起来 太过繁琐。 一般使用RestTemplate，更多简洁方便调用。
- HttpComponents的作用是模拟浏览器获取网页内容。 就是HttpClient

RestTemplate

- 传统情况下在java代码里访问restful服务，一般使用Apache的HttpClient。不过此种方法使用起来过繁琐。spring提供了一种简单便捷的模板类来进行操作，这就是RestTemplate。

依赖

- RestTemplate就是spring其下的产品，就在封装在spring的依赖中

配置类

```
@Configuration
public class RestTempConfig {
    /**
     * @return 返回一个RestTemplate对象，注册到spring容器中。
     */
    @Bean
    public RestTemplate getRestTemplate(){
        // 使用HttpClient，支持GZIP
        RestTemplate restTemplate = new RestTemplate(new HttpComponentsClientHttpReques
Factory());
        //解决RestTemplate的中文乱码问题
```

```
        // 设置中文乱码问题方式一
        restTemplate.getMessageConverters().add(1,new StringHttpMessageConverter(Charset.f
rName("UTF-8")));
        // 设置中文乱码问题方式二
        //restTemplate.getMessageConverters().set(1,
        //new StringHttpMessageConverter(StandardCharsets.UTF_8)); // 支持中文编码
        return restTemplate;
    }
}
```

调用方法：

- 可以调用 RestTemplate的 get ， post 等..各种请求
- 代码：

```
post(@RequestBody Goods goods){
ResponseEntitystring response = restTemplate.postForEntity("http://localhost:8090/goods", g
ods , String.class);
System.out.println("状态码："+response.getStatusCode());
System.out.println("返回信息："+response.getBody());
System.out.println("Headers："+response.getHeaders());
return response;
}
@PutMapping
public ResponseEntitystring put(@RequestBody Goods goods){
restTemplate.put("http://localhost:8090/goods", goods , String.class);
return null;
}
@DeleteMapping("/{id}")
public ResponseEntitystring delete(@PathVariable Integer id){
restTemplate.delete("http://localhost:8090/goods/"+id);
return ResponseEntity.ok("删除成功");
}
">

  @Autowired
  private RestTemplate restTemplate;

  @GetMapping
  public ResponseEntity<String> get(){
     ResponseEntity<String> response = restTemplate.getForEntity("http://localhost:8090/go
ds?pageNum=1&pageSize=2&name=&key=", String.class);
     System.out.println("状态码："+response.getStatusCode());
     System.out.println("返回信息："+response.getBody());
     System.out.println("Headers："+response.getHeaders());
     return response;
```

```
    }

    @PostMapping
    public ResponseEntity<String> post(@RequestBody Goods goods){
        ResponseEntity<String> response = restTemplate.postForEntity("http://localhost:8090/goods", goods , String.class);
        System.out.println("状态码："+response.getStatusCode());
        System.out.println("返回信息："+response.getBody());
        System.out.println("Headers："+response.getHeaders());
        return response;
    }

    @PutMapping
    public ResponseEntity<String> put(@RequestBody Goods goods){
        restTemplate.put("http://localhost:8090/goods", goods , String.class);
        return null;
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String> delete(@PathVariable Integer id){
        restTemplate.delete("http://localhost:8090/goods/"+id);
        return ResponseEntity.ok("删除成功");
    }
```

两者总结

- 相同点：
  - 跨操作系统和编程语言的的访问技术
  - 访问另一个服务器请求的技术
- 不同的：
  - httpClient是Apache的，RestTemplate是spring的
  - RestTemplate更加简单

RestTemplate获取天气预报

```
ResponseEntity<String> response = restTemplate.getForEntity("http://wthrcdn.etouch.cn/weaher_mini?city=沭阳", String.class);
System.out.println("状态码："+response.getStatusCode());
System.out.println("返回信息："+response.getBody());
System.out.println("Headers："+response.getHeaders());
```

原文链接：HttpClient&RestTemplate