访问 Web 资源

极客时间

扫码试看/订阅

《玩转 Spring 全家桶》

通过 RestTemplate 访问 Web 资源

Spring Boot 中的 RestTemplate

RestTemplate 是由 Spring 提供的一个 HTTP 请求工具。
开发者也可以不使用 RestTemplate ，使用 Java 自带的 HttpUrlConnection 或者经典的网络访问框架 HttpClient 也可以完成上文的案例，只是在 Spring 项目中，使用 RestTemplate 显然更方便一些

- Spring Boot 中没有自动配置 RestTemplate
- Spring Boot 提供了 RestTemplateBuilder
  - RestTemplateBuilder.build()

常用方法

GET 请求

- getForObject() / getForEntity()

POST 请求

- postForObject() / postForEntity()

PUT 请求

- put()

DELETE 请求

- delete()

```
String result = restTemplate.getForObject(
        "http://example.com/hotels/{hotel}/bookings/{booking}", String.class, "42", "21");
```

构造 URI

构造 URI

- `UriComponentsBuilder`

构造相对于当前请求的 URI

- `ServletUriComponentsBuilder`

构造指向 Controller 的 URI

- `MvcUriComponentsBuilder`

构造 URI

```
URI uri = UriComponentsBuilder
        .fromUriString("http://example.com/hotels/{hotel}")
        .queryParam("q", "{q}")
        .encode()
        .buildAndExpand("Westin", "123")
        .toUri();
```

```
URI uri = UriComponentsBuilder
        .fromUriString("http://example.com/hotels/{hotel}?q={q}")
        .build("Westin", "123");
```

简化一下

```
UriComponents uriComponents = MvcUriComponentsBuilder
    .fromMethodCall(on(BookingController.class).getBooking(21)).buildAndExpand(42);

URI uri = uriComponents.encode().toUri();
```

“Talk is cheap, show me the code.”

Chapter 7 / simple-resttemplate-demo

RestTemplate 的高阶用法

RestTemplate 的高阶用法

传递 HTTP Header

- `RestTemplate.exchange()`
- `RequestEntity<T> / ResponseEntity<T>`

  可以传入RequestEntity跟响应的ResponseEntity一样

HTTP Header

```
String uriTemplate = "http://example.com/hotels/{hotel}";
URI uri = UriComponentsBuilder.fromUriString(uriTemplate).build(42);

RequestEntity<Void> requestEntity = RequestEntity.get(uri)
        .header("MyRequestHeader", "MyValue")
        .build();

ResponseEntity<String> response = template.exchange(requestEntity, String.class);

String responseHeader = response.getHeaders().getFirst("MyResponseHeader");
String body = response.getBody();
```

get方法是void, post就不是void了

常见的头已经有设置

```
RequestEntity<Void> req = RequestEntity.get(uri)
        .accept(MediaType.APPLICATION_XML)
        .build();
```

RestTemplate 的高阶用法

类型转换

- JsonSerializer / JsonDeserializer
- @JsonComponent 支持特定的序列化和反序列化，自定义

解析泛型对象

- RestTemplate.exchange()
- ParameterizedTypeReference<T> 比如List

解析泛型对象

因为泛型已经擦除的实际类型

```
ParameterizedTypeReference<List<Coffee>> ptr =
                    new ParameterizedTypeReference<List<Coffee>>() {};
ResponseEntity<List<Coffee>> list = restTemplate
                    .exchange(coffeeUri, HttpMethod.GET, null, ptr);
list.getBody().forEach(c -> log.info("Coffee: {}", c));
```

exchange的时候传入对象

如果使用：

list<Coffee> list1 = new ArrayList<>();
List1 = restTemplate.getForObject(coffeeUri, list1.getClass());
List1.forEach(c->log.info(“c; {}”,c.getClass()));

会直接报错：Java.lang.ClassCastException: class java.util.LinkedHashMap cannot be cast to xxx.xxx.xxx.Coffee

因为getForObject返回的是通过JacksonJSON处理的，只认出十个list, 没有找到一个类型的时候，会把JSON把对象变成linkedHashMap的List

“Talk is cheap, show me the code.”

Chapter 7 / complex-resttemplate-demo

简单定制 RestTemplate

RestTemplate 支持的 HTTP 库

通用接口

- `ClientHttpRequestFactory`

默认实现

- `SimpleClientHttpRequestFactory`

RestTemplate 支持的 HTTP 库

Apache HttpComponents

- `HttpComponentsClientHttpRequestFactory`

Netty

如果用Reactor相关的话建议使用WebClient

- `Netty4ClientHttpRequestFactory` 已经Deprecated了

OkHttp

- `OkHttp3ClientHttpRequestFactory` 安卓的

优化底层请求策略

连接管理

- PoolingHttpClientConnectionManager
- KeepAlive 策略

默认自动重试，建议关掉：
disableAutomaticRetries()

超时设置　保证系统不被下游拖死

- connectTimeout / readTimeout

SSL校验

- 证书检查策略

连接复用

```
public class CustomConnectionKeepAliveStrategy implements ConnectionKeepAliveStrategy {
    private final long DEFAULT_SECONDS = 30;

    @Override
    public long getKeepAliveDuration(HttpResponse response, HttpContext context) {
        return Arrays.asList(response.getHeaders(HTTP.CONN_KEEP_ALIVE))
                .stream()
                .filter(h -> StringUtils.equalsIgnoreCase(h.getName(), "timeout")
                        && StringUtils.isNumeric(h.getValue()))
                .findFirst()
                .map(h -> NumberUtils.toLong(h.getValue(), DEFAULT_SECONDS))
                .orElse(DEFAULT_SECONDS) * 1000;
    }
}
```

转换失败用默认值，30秒

默认实现

- org.apache.http.impl.client.DefaultConnectionKeepAliveStrategy

“Talk is cheap, show me the code.”

Chapter 7 / advanced-resttemplate-demo

通过 WebClient 访问 Web 资源

了解 WebClient

WebClient

- 一个以 Reactive 方式处理 HTTP 请求的非阻塞式的客户端

支持的底层 HTTP 库

- Reactor Netty - `ReactorClientHttpConnector` 用的更多
- Jetty ReactiveStream HttpClient - `JettyClientHttpConnector`

WebClient 的基本用法

创建 WebClient

- `WebClient.create()`
- `WebClient.builder()`

发起请求

- `get() / post() / put() / delete() / patch()`

WebClient 的基本用法

获得结果

跟restTemplate一样

- `retrieve()` / `exchange()`

处理 HTTP Status

- `onStatus()`

应答正文

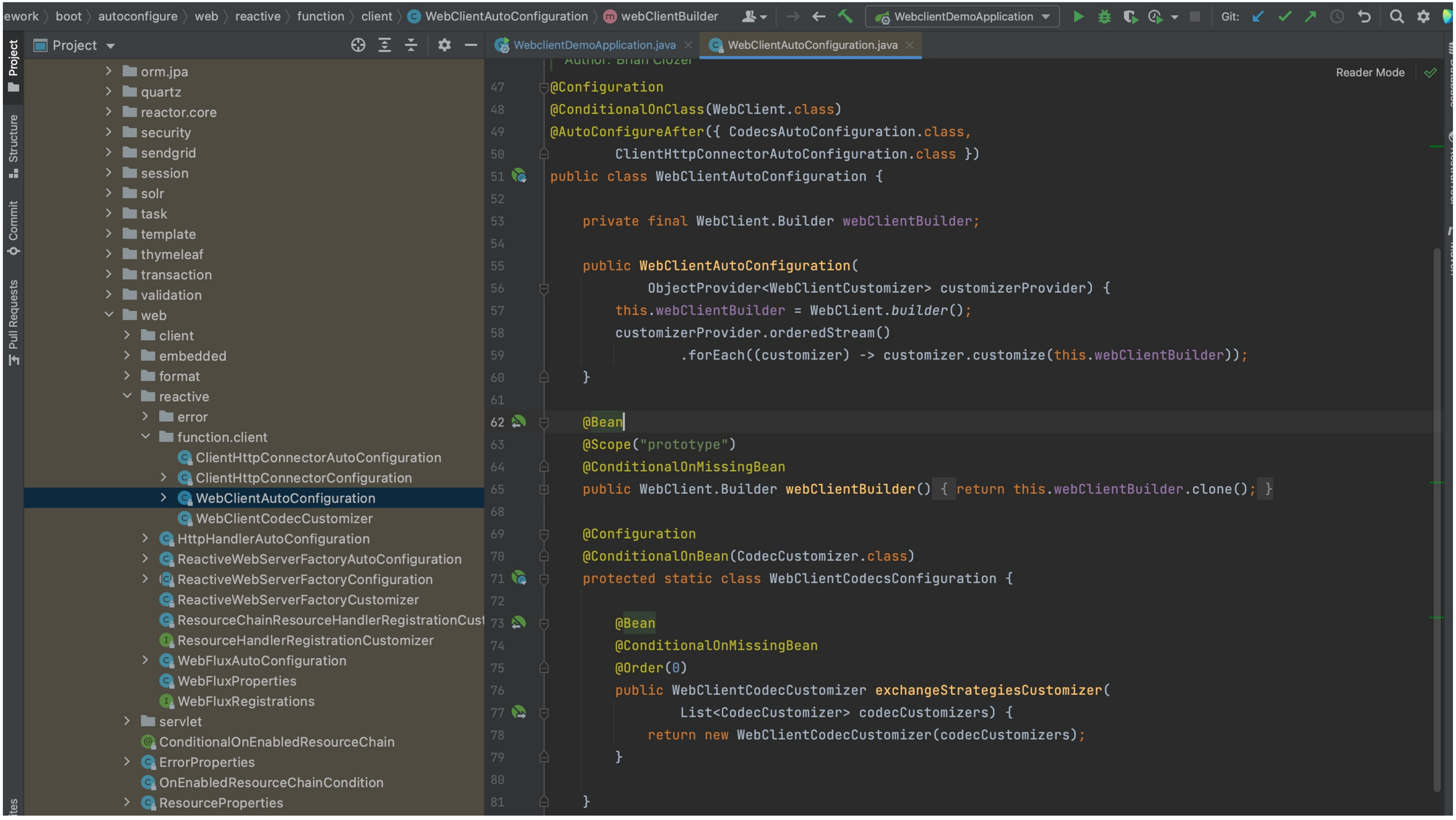
- `bodyToMono()` / `bodyToFlux()`

单个响应

多个响应

“Talk is cheap, show me the code.”

Chapter 7 / webclient-demo


ework boot autoconfigure web reactive function client WebClientAutoConfiguration webClientBuilder
WebclientDemoApplication
Git:
Project
WebclientDemoApplication.java
WebClientAutoConfiguration.java
Reader Mode
orm.jpa
quartz
reactor.core
security
sendgrid
session
solr
task
template
thymeleaf
transaction
validation
web
client
embedded
format
reactive
error
function.client
ClientHttpConnectorAutoConfiguration
ClientHttpConnectorConfiguration
WebClientAutoConfiguration
WebClientCodecCustomizer
HttpHandlerAutoConfiguration
ReactiveWebServerFactoryAutoConfiguration
ReactiveWebServerFactoryConfiguration
ReactiveWebServerFactoryCustomizer
ResourceChainResourceHandlerRegistrationCust
ResourceHandlerRegistrationCustomizer
WebFluxAutoConfiguration
WebFluxProperties
WebFluxRegistrations
servlet
ConditionalOnEnabledResourceChain
ErrorProperties
OnEnabledResourceChainCondition
ResourceProperties
@Configuration
@ConditionalOnClass(WebClient.class)
@AutoConfigureAfter({ CodecsAutoConfiguration.class,
ClientHttpConnectorAutoConfiguration.class })
public class WebClientAutoConfiguration {
private final WebClient.Builder webClientBuilder;
public WebClientAutoConfiguration(
ObjectProvider<WebClientCustomizer> customizerProvider) {
this.webClientBuilder = WebClient.builder();
customizerProvider.orderedStream()
.forEach((customizer) -> customizer.customize(this.webClientBuilder));
}
@Bean
@Scope("prototype")
@ConditionalOnMissingBean
public WebClient.Builder webClientBuilder() { return this.webClientBuilder.clone(); }
@Configuration
@ConditionalOnBean(CodecCustomizer.class)
protected static class WebClientCodecsConfiguration {
@Bean
@ConditionalOnMissingBean
@Order(0)
public WebClientCodecCustomizer exchangeStrategiesCustomizer(
List<CodecCustomizer> codecCustomizers) {
return new WebClientCodecCustomizer(codecCustomizers);
}
}

SpringBucks 进度小结

本章小结

- RestTemplate 的各种用法
- RestTemplate 的简单定制
- WebClient 的基本用法

SpringBucks 进度小结

增加了 customer-service

- 通过编码方式查询咖啡
- 通过编码方式创建订单

“Talk is cheap, show me the code.”

Chapter 7 / customer-service

极客时间

扫码试看/订阅

《玩转 Spring 全家桶》