الدرس العملي

سأشرح، في هذا الدرس العملي، كيفية استخدام [illegible] أو [illegible]، أو [illegible]، لتسهيل عملية التكامل مع واجهات برمجة التطبيقات الخارجية. سنبدأ بالأساسيات ثم ننتقل إلى أمثلة عملية خطوة بخطوة.

---

1. استخدام [illegible]

[illegible] هو العميل الذي يوفره [illegible]. يسمح لك بتحديد واجهات برمجة التطبيقات التي تريد التكامل معها بطريقة تصريحية. بدلاً من كتابة كود منخفض المستوى لإجراء طلبات HTTP، يمكنك تحديد واجهة بسيطة وسيقوم [illegible] بإنشاء التنفيذ تلقائيًا.

خطوات استخدام [illegible]

1. **إضافة التبعيات**: أضف التبعيات المطلوبة إلى مشروعك. إذا كنت تستخدم [illegible]، فابدأ بإضافة [illegible] إلى ملف `pom.xml` الخاص بك:

   ```
   <dependency>
       <groupId>org.springframework.cloud</groupId>
       <artifactId>spring-cloud-starter-openfeign</artifactId>
   </dependency>
   ```

   تأكد أيضًا من وجود تبعيات إدارة كتلة [illegible] المتوافق. الإصدار، مع [illegible]

2. **تفعيل [illegible]**: أضف التعليق `@EnableFeignClients` إلى فئة التطبيق الرئيسية أو إلى فئة الإعدادات لتفعيل [illegible]:

   ```
   import org.springframework.boot.SpringApplication;
   import org.springframework.boot.autoconfigure.SpringBootApplication;
   import org.springframework.cloud.openfeign.EnableFeignClients;

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

3. **تعريف واجهة [illegible]**: أنشئ واجهة مع التعليق `@FeignClient`، مع تحديد اسم الخدمة أو [illegible]، وحدد نقاط النهاية التي تتوافق مع [illegible]:

```java
import org.springframework.cloud.openfeign.FeignClient;
import org.springframework.web.bind.annotation.GetMapping;
import java.util.List;

@FeignClient(name = "user-service", url = "http://localhost:8080")
public interface UserClient {
    @GetMapping("/users")
    List<User> getUsers();
}
```

انه، name هو اسم منطقي للعميل، و url هو □□□ الأساس للخدمة المستهدفة. تعليق @GetMapping يتوافق مع نقطة النهاية /users.

4. **إدخال واستخدام العميل**: أدخل الواجهة في خدمة أو تحكم وتصل إلى أساليبها كما لو كانت محلية:

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class UserService {
    @Autowired
    private UserClient userClient;

    public List<User> fetchUsers() {
        return userClient.getUsers();
    }
}
```

النقاط الرئيسية

- □ □□□□□□□□□□□ متزامن بشكل افتراضي.
- □ هو مثالي للميكروسيرفيس مع اكتشاف الخدمة (مثل □□□□□□□) عندما تتركب url وتستحم لـ □□□□□□ □□□□□ بإيجاده.
- □ يمكن إضافة معالجة الأخطاء مع استرجاعات أو محولات الدائرة (مثل □□□□□□□ أو □□□□□□□□□□4□□).

---

2. استخدام □□□□□□ □□□□□□□□□□

RestTemplate لـ حديث كبديل 6.1 □□□□□□ □□□□□□□□□ في تقديمه تم متزامن □□□□ عميل هو □□□□□□ □□□□□□□□□□ المدمد. بالإنقاض. ويوفر واجهة متداخلة لبناء وإجراء الطلبات.

اس□□□□□□□□□□ خطوات

1. ات□□□□□□□□□□ الإعتبا: وهو جزء من spring-web، وهو جزء من spring-boot-starter-web لـ □□□□□□ □□□□. الحاجة إلى إعادة تعليقات إضافية:

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

2. □□□□□□□□□□ يمكن إنشاء: يمكن أن ينشئ RestClient باستخدام طريقة create() الثابتة أو قم بتخصيصه باستخدام الباني:

```
import org.springframework.web.client.RestClient;

RestClient restClient = RestClient.create();
```

RestClient.builder() استخدم الزمنية، الأوقات□ مثل الخصائص الإعدادات لاستخدام□.

3. **بناء وإجراء طلب**: استخدم واجهة الخادم لتحديد طريقة □□□□، □□□، الرؤوس، والجسم ثم استرجاع الرد:

```
import org.springframework.http.MediaType;
import org.springframework.web.client.RestClient;
import java.util.List;

public class UserService {
    private final RestClient restClient;

    public UserService() {
        this.restClient = RestClient.create();
    }

    public List<User> fetchUsers() {
        return restClient.get()
            .uri("http://localhost:8080/users")
            .accept(MediaType.APPLICATION_JSON)
            .retrieve()
            .body(new ParameterizedTypeReference<List<User>>() {});
    }
}
```

- □ get() لتحديد طريقة □□□□.
- □ uri() لتحديد نقطة النهاية.
- □ accept() لتحديد نوع المحتوى المتوقع.

- retrieve() يرجى، الطلب و body() جرخت يس، ردلا مادختساب ParameterizedTypeReference لثم عاونأ لأغراض ال قوائم.

4. **ةجالمعة ردلا**: متي اعرجتسا ردلا ةرشابم نأل [illegible] نمازتم. لوصحلل ىلع ديزم نم مكحتلا لثم [illegible] زومر، ةالحلا [illegible] مدختسا toEntity():

```
import org.springframework.http.ResponseEntity;

ResponseEntity<List<User>> response = restClient.get()
    .uri("http://localhost:8080/users")
    .accept(MediaType.APPLICATION_JSON)
    .retrieve()
    .toEntity(new ParameterizedTypeReference<List<User>>() {});
List<User> users = response.getBody();
```

ةيسيئرلا ةقاطبلا

- [illegible] نمازتم امم هلعجي اًسبًا نم ةقيبطتلل، ةيديلقتلا ةفوقوملا.
- [illegible] أطخاء عند [illegible] RestClientException لثم ةئاثتسالا ةلاحل يرمي. هيلع ضبقلا كنكمي، [illegible] وهعاجتها.
- [illegible] وه ليدبلا لـ RestTemplate عم ةهجاو رثكأ. أوضحًا.

---

3. مادختسا [illegible]

[illegible] وه ليمع [illegible]، لعافتم ريغ نمازتم تم هميدقت في [illegible]. تم هميمصت اياملع ريغ ةنمازتم جمدي مع ةقفدت لعافتم [illegible] و [illegible].

مادختسا [illegible] خطوات

1. **ةفاضإ ةيعبتلا**: فضأ ةيعبتلا [illegible] إلى مشروعك:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-webflux</artifactId>
</dependency>
```

2. **ءاشنإ ليثم [illegible]**: ئشنأ ليثم WebClient مع [illegible] الإعدادات أو الأساس الافتراضية:

```
import org.springframework.web.reactive.function.client.WebClient;

WebClient webClient = WebClient.create("http://localhost:8080");
```

واالفيلترات. الكودات، مثل مخصصة لإعدادات WebClient.builder() استخدم

3. **طلب وإجراء بناء**: استخدم واجهة الداخلة لبناء الطلب واسترجاع الرد: المتفاعل

```
import org.springframework.http.MediaType;
import org.springframework.web.reactive.function.client.WebClient;
import reactor.core.publisher.Mono;
import java.util.List;

public class UserService {
    private final WebClient webClient;

    public UserService(WebClient webClient) {
        this.webClient = webClient;
    }

    public Mono<List<User>> fetchUsers() {
        return webClient.get()
            .uri("/users")
            .accept(MediaType.APPLICATION_JSON)
            .retrieve()
            .bodyToFlux(User.class)
            .collectList();
    }
}
```

- bodyToFlux(User.class) لكائنات User من تدفق عالجي.
- collectList() لتحويل Flux<User> إلى Mono<List<User>>.

4. **الرد في الاشتراك**: لأن □□□□□□□□□□، يجب متفاعل عليك الاشتراك في Mono أو Flux لتفعيل الطلب:

```
Mono<List<User>> usersMono = fetchUsers();
usersMono.subscribe(users -> System.out.println(users));
```

أو سلسلة في خط بيانات أو متفاعل إيقاف غير موصى به في سياقات: المتفاعلة□

```
List<User> users = fetchUsers().block();
```

النقاط الرئيسية

- □□□□□□□□□□ غير متزامن ومتفاعل تم الوصول إليها بنماذج □□□□□□□ □□□□□□□□.
- استخدم Mono لردود ذات قيمة واحدة و Flux لردود ذات قيم متعددة.
- كن على دراية بمعالجة الأخطاء باستخدام عوامل مثل retry() أو onErrorResume().

مقارنة ووقت استخدام كل منها

- **[illegible]**: يستخدم للعملية المتزامنة وهو بسيط. إعدادات في الواجهة على القائم التوضيحي، يتم إرسال الطلبات بين الخدمات.
- **[illegible]**: يستخدم للطلبات [illegible] المتزامنة في التطبيقات التي تحتاج إلى توافق مع [illegible]. وهو حديث ويدعم RestTemplate.
- **[illegible]**: يستخدم للطلبات [illegible] غير المتزامنة في التطبيقات [illegible] أو عند التعامل مع كثافة عالية.

مثال استدعاء نقطة النهاية [illegible]:

افترض أنك تريد إنشاء مستخدم من خلال استدعاء POST /users مع كائن User.

[illegible]

```
@FeignClient(name = "user-service", url = "http://localhost:8080")
public interface UserClient {
    @PostMapping("/users")
    User createUser(@RequestBody User user);
}

@Service
public class UserService {
    @Autowired
    private UserClient userClient;

    public User createUser(User user) {
        return userClient.createUser(user);
    }
}
```

[illegible]

```
@Service
public class UserService {
    private final RestClient restClient = RestClient.create();

    public User createUser(User user) {
```

```
        return restClient.post()
            .uri("http://localhost:8080/users")
            .contentType(MediaType.APPLICATION_JSON)
            .body(user)
            .retrieve()
            .body(User.class);
    }
}
```

[illegible]

```
@Service
public class UserService {
    private final WebClient webClient;

    public UserService(WebClient webClient) {
        this.webClient = WebClient.create("http://localhost:8080");
    }

    public Mono<User> createUser(User user) {
        return webClient.post()
            .uri("/users")
            .contentType(MediaType.APPLICATION_JSON)
            .bodyValue(user)
            .retrieve()
            .bodyToMono(User.class);
    }
}
```

ةمتاخلا

- [illegible]: ددح ،ةهجاو ،اهليعفت اهمدختساو لثم نئاك .يلحم
- [illegible]: ءانب تابلطلا لكشب لخادتم عاجرتساو دودرلا لكشب .نمازتم
- [illegible]: ءانب تابلطلا لكشب ،لخادتم عاجرتسا تاقفدت ،ةلعافتم اهتجلاعمو لكشب ريغ .نمازتم

[illegible] و [illegible]ةقوقوملا ،[illegible] تاقيبطتلل [illegible] ،سيفريسوكيمللا [illegible] :كقيبطت ةينب ىلع ءانب رتخا
أعلاه. حوضوم وه امك طيسب ططم نم عبتي اهنم لكل ةلعافتملا تاقيبطتلل. مادختساو مادعإ