Netflix Feign Client in Spring Boot

Introduction

Feign is a declarative web service client. It makes writing web service clients easier. To use Feign create an interface and annotate it. It has pluggable annotation support including Feign annotations and JAX-RS annotations. Feign also supports pluggable encoders and decoders. Feign is a Java to HTTP client binder inspired by Retrofit, JAXRS-2.0, and WebSocket. Feign's first goal was reducing the complexity of binding Denominator uniformly to HTTP APIs regardless of ReSTfulness.

How does Feign work?

Feign works by processing annotations into a templatized request. Arguments are applied to these templates in a straightforward fashion before output. Although Feign is limited to supporting text-based APIs, it dramatically simplifies system aspects such as replaying requests. Furthermore, Feign makes it easy to unit test your conversions knowing this.

A complete example is given below.

Maven (pom.xml)

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>organisation</groupId>
 <artifactId>organisation</artifactId>
 <version>0.0.1-SNAPSHOT</version>
 <packaging>jar</packaging>
 <name>organisation</name>
 <url>http://maven.apache.org</url>
 <parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.0.0.RELEASE</version>
 </parent>
 <properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
 </properties>
 <dependencies>
  <dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
   <groupId>org.springframework.cloud</groupId>
   <artifactId>spring-cloud-starter-openfeign</artifactId>
  </dependency>
```

```
  <dependency>
   <groupId>org.projectlombok</groupId>
   <artifactId>lombok</artifactId>
   <scope>provided</scope>
  </dependency>
  <dependency>
   <groupId>io.github.openfeign</groupId>
   <artifactId>feign-core</artifactId>
  </dependency>
  <dependency>
   <groupId>io.github.openfeign</groupId>
   <artifactId>feign-okhttp</artifactId>
  </dependency>
  <dependency>
   <groupId>com.netflix.feign</groupId>
   <artifactId>feign-jackson</artifactId>
   <version>8.18.0</version>
  </dependency>
  <dependency>
   <groupId>io.github.openfeign</groupId>
   <artifactId>feign-slf4j</artifactId>
  </dependency>
 </dependencies>
 <dependencyManagement>
  <dependencies>
   <dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-dependencies</artifactId>
    <version>Finchley.RELEASE</version>
    <type>pom</type>
    <scope>import</scope>
   </dependency>
  </dependencies>
 </dependencyManagement>
 <build>
  <plugins>
   <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
     <source>1.8</source>
     <target>1.8</target>
    </configuration>
   </plugin>
   <plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
   </plugin>
  </plugins>
 </build>

</project>
```

SpringBoot Main Application – Web Layer

OrganisationBootApplication.java

```
package com.ddlab.boot.web;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

BootConfig.java

```
package com.ddlab.boot.web;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@ComponentScan(basePackages= {"com.ddlab.boot.*"})
@Configuration
public class BootConfig {}
```

Entity Layer

Location.java

```
package com.ddlab.boot.entity;
import java.io.IOException;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonPropertyOrder;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@JsonPropertyOrder({"officeName", "streetName", "city", "pincode"})
public class Location {
  @JsonProperty private String officeName;
  @JsonProperty private String streetName;
  @JsonProperty private String city;
  @JsonProperty private String pincode;

  public String toJSON() {
    ObjectMapper mapper = new ObjectMapper();
    String toJson = null;
    try {
      toJson = mapper.writerWithDefaultPrettyPrinter().writeValueAsString(this);
    } catch (IOException e) {
      e.printStackTrace();
    }
    return toJson;
  }
}
```

Service Layer

OrgService.java

```
package com.ddlab.boot.service;
import com.ddlab.boot.entity.Location;

public interface OrgService {
      Location getLocationByPincode(String pincode);
}
```

OrgServiceImpl.java

```
package com.ddlab.boot.service;
import org.springframework.stereotype.Service;
import com.ddlab.boot.entity.Location;

@Service
public class OrgServiceImpl implements OrgService {

  @Override
  public Location getLocationByPincode(String pincode) {
    Location location = new Location();
    location.setOfficeName("DDLAB INC");
    location.setCity("Bangalore, Karnataka");
    location.setPincode(pincode);
    location.setStreetName("13th main road, 5th block");
    return location;
  }
}
```

Resource or Controller Layer

OrgController.java

```
package com.ddlab.boot.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.ddlab.boot.entity.Location;
import com.ddlab.boot.service.OrgService;

@RestController
@RequestMapping("/home")
@CrossOrigin
public class OrgController {

  @Autowired private OrgService orgService;

  @GetMapping(path = "/location/{pincode}", produces =
MediaType.APPLICATION_JSON_VALUE)
  public ResponseEntity<Location> getLocationByPincode(@PathVariable("pincode")
String pincode) {
    System.out.println("Org Service :::" + orgService);
    return new ResponseEntity<Location>(orgService.getLocationByPincode(pincode),
HttpStatus.OK);
  }

  @PostMapping(
      path = "/create/location",
      consumes = MediaType.APPLICATION_JSON_VALUE,
      produces = MediaType.TEXT_PLAIN_VALUE)
  public ResponseEntity<String> createOrgSpace(@RequestBody Location location) {
    System.out.println("Created location : " + location);
    return new ResponseEntity<String>("Resource created successfully.",
HttpStatus.CREATED);
  }

  @PutMapping(
      path = "/update/location",
      consumes = MediaType.APPLICATION_JSON_VALUE,
      produces = MediaType.APPLICATION_JSON_VALUE)
  public ResponseEntity<Location> updateOrgSpace(@RequestBody Location location) {
    System.out.println("Updated location : " + location);
    return new ResponseEntity<Location>(location, HttpStatus.OK);
  }
```

```
  @DeleteMapping(
      path = "/delete/location",
      consumes = MediaType.APPLICATION_JSON_VALUE,
      produces = MediaType.TEXT_PLAIN_VALUE)
  public ResponseEntity<String> deleteOrgSpace(@RequestBody Location location) {
    System.out.println("Deleted location : " + location);
    return new ResponseEntity<String>("Resource deleted successfully.",
HttpStatus.OK);
  }
}
```

Spring Boot Configuration

Application.properties

```
spring.application.name=org
server.servlet.context-path=/org
server.port=8081
server.error.whitelabel.enabled=false
```

Usage of Feign Client

LocationFeign.java

```
package com.ddlab.rnd.feign.client;
import org.springframework.cloud.openfeign.FeignClient;
import com.ddlab.boot.entity.Location;
import feign.Headers;
import feign.Param;
import feign.RequestLine;

@FeignClient(name = "location") // It can be any name of your choice
public interface LocationFeign {
  @RequestLine("GET /location/{pincode}")
  String getLocation(@Param("pincode") String pincode);

  @RequestLine("POST /create/location")
  @Headers("Content-Type: application/json")
  String createLocation(Location location);

  @RequestLine("PUT /update/location")
  @Headers("Content-Type: application/json")
  Location updateLocation(Location location);

  @RequestLine("DELETE /delete/location")
  @Headers("Content-Type: application/json")
  String deleteLocation(Location location);
}
```

LocationFeignTest.java

```
package com.ddlab.rnd.feign.client;
import com.ddlab.boot.entity.Location;
import feign.Feign;
import feign.Target;
import feign.codec.Decoder;
import feign.codec.Encoder;
import feign.jackson.JacksonDecoder;
import feign.jackson.JacksonEncoder;
import feign.okhttp.OkHttpClient;
import feign.slf4j.Slf4jLogger;

public class LocationFeignClientTest {
  private static final Target<LocationFeign> target =
      new Target.HardCodedTarget<>(LocationFeign.class, "http://localhost:8081/org/home/");

  public static void showLocationDetails() {
    LocationFeign feignLocation =
        Feign.builder()
            .client(new OkHttpClient())
            .encoder(new Encoder.Default())
            .decoder(new Decoder.Default())
            .logger(new Slf4jLogger(LocationFeign.class))
            .logLevel(feign.Logger.Level.BASIC)
            .target(target);
    System.out.println("Location details :::" + feignLocation.getLocation("11"));
  }

  public static void showCreatedLocation() {
    LocationFeign feignLocation =
        Feign.builder()
            .client(new OkHttpClient())
            .encoder(new JacksonEncoder())
            .decoder(new Decoder.Default())
            .logger(new Slf4jLogger(LocationFeign.class))
            .logLevel(feign.Logger.Level.BASIC)
            .target(target);
    Location location = getLocation();
    System.out.println("~~~~~~~~~~~~~~" + feignLocation.createLocation(location));
  }

  public static void showUpdatedLocation() {
    LocationFeign feignLocation =
        Feign.builder()
            .client(new OkHttpClient())
            .encoder(new JacksonEncoder())
            .decoder(new JacksonDecoder())
            .logger(new Slf4jLogger(LocationFeign.class))
            .logLevel(feign.Logger.Level.BASIC)
            .target(target);
    Location location = getLocation();
    System.out.println("------->" + feignLocation.updateLocation(location));
  }
```

```
  public static void showDeletedLocation() {
    LocationFeign feignLocation =
        Feign.builder()
            .client(new OkHttpClient())
            .encoder(new JacksonEncoder())
            .decoder(new Decoder.Default())
            .logger(new Slf4jLogger(LocationFeign.class))
            .logLevel(feign.Logger.Level.BASIC)
            .target(target);
    Location location = getLocation();
    System.out.println("=====>" + feignLocation.deleteLocation(location));
  }

  private static Location getLocation() {
    Location location = new Location();
    location.setOfficeName("DDLab Inc");
    location.setStreetName("14th Cannin Stree");
    location.setCity("Bangalore");
    location.setPincode("123456");
    return location;
  }

  public static void main(String[] args) {
    showLocationDetails();
    showCreatedLocation();
    showUpdatedLocation();
    showDeletedLocation();
  }
}
```

How to Use

Start Spring Boot Application and then execute the LocationFeignTest.java class as a standalone.

The following output is expected.

```
~~~~~~~~~~~~~Resource created successfully.
00:50:15.492 [main] DEBUG com.ddlab.rnd.feign.client.LocationFeign -
[LocationFeign#updateLocation] ---> PUT
http://localhost:8081/org/home/update/location HTTP/1.1
00:50:15.509 [main] DEBUG com.ddlab.rnd.feign.client.LocationFeign -
[LocationFeign#updateLocation] <--- HTTP/1.1 200  (16ms)
------->Location(officeName=DDLab Inc, streetName=14th Cannin Stree,
city=Bangalore, pincode=123456)
00:50:15.553 [main] DEBUG com.ddlab.rnd.feign.client.LocationFeign -
[LocationFeign#deleteLocation] ---> DELETE
http://localhost:8081/org/home/delete/location HTTP/1.1
00:50:15.563 [main] DEBUG com.ddlab.rnd.feign.client.LocationFeign -
[LocationFeign#deleteLocation] <--- HTTP/1.1 200  (9ms)
=====>Resource deleted successfully.
```