API-GETEWAY

Netflix Zuul Microservice with Spring Boot

HARSHAL MORE

harshalmore.it@gmail.com

https://www.linkedin.com/in/harshal-more-patil/

API-GETEWAY

In a microservices application, there are many small services running separately each handling a specific task like user login, payments, or showing product details. Now, imagine a user wants to use your application. Instead of the user directly calling each microservice, we give them **one single entry point**. That entry point is called the **API Gateway**.

Netflix Zuul is a tool used to create this API Gateway in a Spring Boot project. It acts like a **smart gate** or **traffic controller** that receives all user requests and then sends them to the correct microservice behind the scenes. It can also add security, log requests, and handle errors.

Using **Netflix Zuul**, we can make our system simpler, more secure, and easier to manage.

Pointwise Explanation (Very Simple)

1. **Microservices** = Small parts of a big application, each doing one job (like login, payment, order, etc.).
2. **Problem** = User has to call each microservice directly — messy and hard to manage.
3. **Solution** = Use an **API Gateway** — one place where all user requests come in.
4. **Netflix Zuul** = A tool that helps create the API Gateway in a Spring Boot project.
5. **What Zuul Does**:
   o   Receives requests from users
   o   Sends them to the correct microservice
   o   Returns the response to the user
   o   Can also handle logging, authentication, and errors
6. **Why Use Zuul**:
   o   Easier to manage microservices
   o   More secure (you can add security rules in one place)
   o   Better control over traffic
7. **Spring Boot + Zuul** = Simple way to build and run an API Gateway quickly.
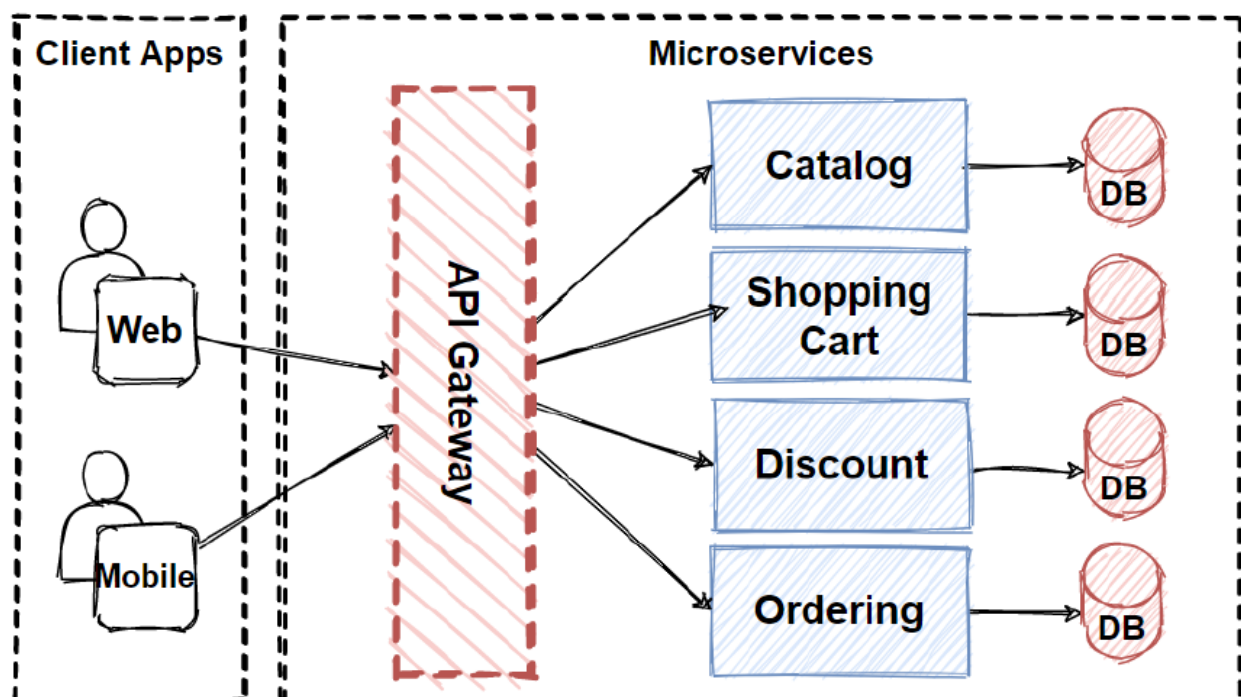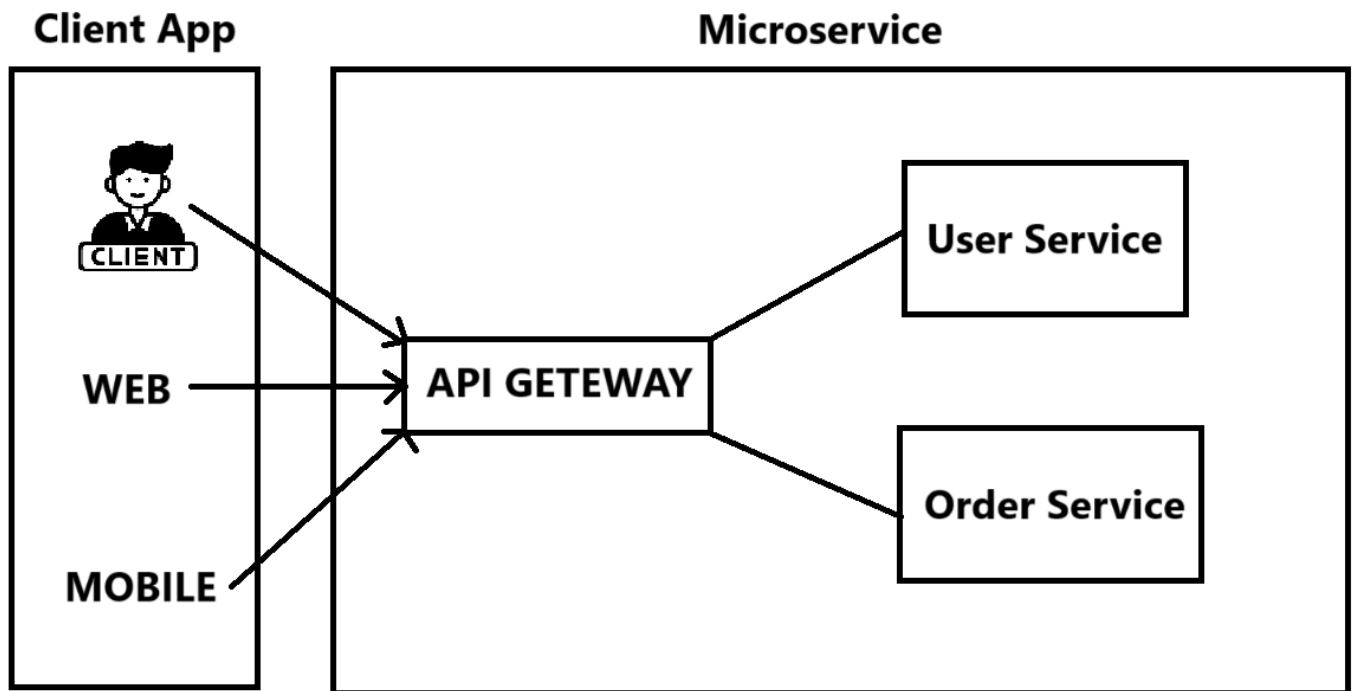
<u>**Netflix Zuul API Gateway Example with Spring Boot**</u>

We have two microservices,

1. **User-Service**
2. **Order-Service,**

integrated with an **API-Gateway**

1. User Service:

Project Structure

```
∨ 🅼 user-service [boot]
   ∨ 🗁 src/main/java
      ∨ ⊞ com.app.userservice
         > 🗋 UserServiceApplication.java
      ∨ ⊞ com.app.userservice.controller
         > 🗋 UserController.java
   ∨ 🗁 src/main/resources
        🗋 application.yml
     🗁 src/test/java
   > 🗃 JRE System Library [JavaSE-11]
   > 🗃 Maven Dependencies
     🗁 target/generated-sources/annotations
     🗁 target/generated-test-sources/test-annotations
   > 🗁 src
   > 🗁 target
     🗎 HELP.md
     🗎 mvnw
     🗎 mvnw.cmd
     🗎 pom.xml
```

Class: UserServiceApplication.Java

```java
1  package com.app.userservice;
2
3  import org.springframework.boot.SpringApplication;
5
6  @SpringBootApplication
7  public class UserServiceApplication {
8
9      public static void main(String[] args) {
10         SpringApplication.run(UserServiceApplication.class, args);
11     }
12
13 }
```

Class : UserController

```java
1  package com.app.userservice.controller;
2
3  import org.springframework.web.bind.annotation.GetMapping;
10
11 @RestController
12 @RequestMapping("/users")
13 public class UserController {
14
15     @GetMapping("/{id}")
16     public Map<String, Object> getUser(@PathVariable String id) {
17         Map<String, Object> user = new HashMap<>();
18         user.put("id", id);
19         user.put("name", "User " + id);
20         user.put("email", "user" + id + "@example.com");
21         user.put("service", "user-service");
22         return user;
23     }
24
25     @GetMapping
26     public Map<String, Object> getAllUsers() {
27         Map<String, Object> response = new HashMap<>();
28         response.put("message", "All users from user-service");
29         response.put("service", "user-service");
30         response.put("port", "8081");
31         return response;
32     }
33 }
```

YAML: application.yml

```yaml
server:
  port: 8081

spring:
  application:
    name: user-service

logging:
  level:
    com.example.userservice: DEBUG
```

Pom.xml

```xml
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
<?xml version="1.0" encoding="UTF-8"?>
<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>com.example</groupId>
    <artifactId>user-service</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.12.RELEASE</version>
        <relativePath/>
    </parent>

    <properties>
        <java.version>11</java.version>
    </properties>

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

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

2. Order Service

Project Structure

```
order-service [boot]
  src/main/java
    com.app.orderservice
      OrderServiceApplication.java
    com.app.orderservice.controller
      OrderController.java
  src/main/resources
    application.yml
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  src
  target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

Class : OrderServiceApplication.java

```java
package com.app.orderservice;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class OrderServiceApplication {

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

}
```

Class: OrderController.java

OrderController.java ×

```java
 1 package com.app.orderservice.controller;
 2
 3 import org.springframework.web.bind.annotation.GetMapping;
10
11 @RestController
12 @RequestMapping("/orders")
13 public class OrderController {
14
15     @GetMapping("/{id}")
16     public Map<String, Object> getOrder(@PathVariable String id) {
17         Map<String, Object> order = new HashMap<>();
18         order.put("id", id);
19         order.put("product", "Product " + id);
20         order.put("amount", 100.00);
21         order.put("service", "order-service");
22         return order;
23     }
24
25     @GetMapping
26     public Map<String, Object> getAllOrders() {
27         Map<String, Object> response = new HashMap<>();
28         response.put("message", "All orders from order-service");
29         response.put("service", "order-service");
30         response.put("port", "8082");
31         return response;
32     }
33 }
```

YAML: application.yml

application.yml ×

```yaml
 1 server:
 2   port: 8082
 3
 4 spring:
 5   application:
 6     name: order-service
 7
 8 logging:
 9   level:
10     com.example.orderservice: DEBUG
```

Pom.xml

```xml
 order-service/pom.xml ×

     http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
 1  <?xml version="1.0" encoding="UTF-8"?>
 2  <project xmlns="http://maven.apache.org/POM/4.0.0"
 3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 4           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
 5           http://maven.apache.org/xsd/maven-4.0.0.xsd">
 6      <modelVersion>4.0.0</modelVersion>
 7
 8      <groupId>com.example</groupId>
 9      <artifactId>order-service</artifactId>
10      <version>1.0-SNAPSHOT</version>
11      <packaging>jar</packaging>
12
13      <parent>
14          <groupId>org.springframework.boot</groupId>
15          <artifactId>spring-boot-starter-parent</artifactId>
16          <version>2.3.12.RELEASE</version>
17          <relativePath/>
18      </parent>
19
20      <properties>
21          <java.version>11</java.version>
22      </properties>
23
24      <dependencies>
25          <dependency>
26              <groupId>org.springframework.boot</groupId>
27              <artifactId>spring-boot-starter-web</artifactId>
28          </dependency>
29      </dependencies>
30
31      <build>
32          <plugins>
33              <plugin>
34                  <groupId>org.springframework.boot</groupId>
35                  <artifactId>spring-boot-maven-plugin</artifactId>
36              </plugin>
37          </plugins>
38      </build>
39  </project>
```

After developing two services finally we have to develop API-GETEWAY, Lets do it…

API-GETEWAY

```
v api-gateway [boot]
  v src/main/java
    v com.app.geteway
      > ApiGatewayApplication.java
  v src/main/resources
      application.yml
  src/test/java
  > JRE System Library [JavaSE-11]
  > Maven Dependencies
  target/generated-sources/annotations
  target/generated-test-sources/test-annotations
  > src
  > target
  HELP.md
  mvnw
  mvnw.cmd
  pom.xml
```

Class: ApiGatewayApplication.java

```java
ApiGatewayApplication.java ×
1 package com.app.geteway;
2
3 import org.springframework.boot.SpringApplication;
6
7 @SpringBootApplication
8 @EnableZuulProxy
9 public class ApiGatewayApplication {
10
11     public static void main(String[] args) {
12         SpringApplication.run(ApiGatewayApplication.class, args);
13     }
14
15 }
```

YAML: application.yml

```yaml
application.yml ×
 1 server:
 2   port: 8080
 3
 4 spring:
 5   application:
 6     name: api-gateway
 7
 8 # Zuul Configuration
 9 zuul:
10   routes:
11     user-service:
12       path: /users/**
13       url: http://localhost:8081
14       strip-prefix: false
15
16     order-service:
17       path: /orders/**
18       url: http://localhost:8082
19       strip-prefix: false
20
21   # Global settings
22   host:
23     connect-timeout-millis: 20000
24     socket-timeout-millis: 20000
25
26   # Remove sensitive headers
27   sensitive-headers: Cookie,Set-Cookie,Authorization
28
29 logging:
30   level:
31     com.netflix.zuul: DEBUG
32     org.springframework.cloud.netflix.zuul: DEBUG
```

Pom.xml

```xml
    http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5           http://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7
8      <groupId>com.example</groupId>
9      <artifactId>api-gateway</artifactId>
10     <version>1.0-SNAPSHOT</version>
11     <packaging>jar</packaging>
12
13     <parent>
14         <groupId>org.springframework.boot</groupId>
15         <artifactId>spring-boot-starter-parent</artifactId>
16         <version>2.3.12.RELEASE</version>
17         <relativePath/>
18     </parent>
19
20     <properties>
21         <java.version>11</java.version>
22         <spring-cloud.version>Hoxton.SR12</spring-cloud.version>
23     </properties>

25     <dependencies>
26         <dependency>
27             <groupId>org.springframework.boot</groupId>
28             <artifactId>spring-boot-starter-web</artifactId>
29         </dependency>
30
31         <dependency>
32             <groupId>org.springframework.cloud</groupId>
33             <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
34         </dependency>
35
36         <dependency>
37             <groupId>org.springframework.cloud</groupId>
38             <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
39         </dependency>
40     </dependencies>
41
42     <dependencyManagement>
43         <dependencies>
44             <dependency>
45                 <groupId>org.springframework.cloud</groupId>
46                 <artifactId>spring-cloud-dependencies</artifactId>
47                 <version>${spring-cloud.version}</version>
48                 <type>pom</type>
49                 <scope>import</scope>
50             </dependency>
51         </dependencies>
52     </dependencyManagement>
```

```
54⊖    <build>
55⊖        <plugins>
56⊖            <plugin>
57                <groupId>org.springframework.boot</groupId>
58                <artifactId>spring-boot-maven-plugin</artifactId>
59            </plugin>
60        </plugins>
61    </build>
62 </project>
```

How to Run

Start the services in order

 # Terminal 1 - Start User Service
cd user-service
mvn spring-boot:run

 # Terminal 2 - Start Order Service
cd order-service
mvn spring-boot:run

 # Terminal 3 - Start API Gateway
cd api-gateway
mvn spring-boot:run

Test the API Gateway

 # Access User Service through Gateway
http://localhost:8080/users
http://localhost:8080/users/123

 # Access Order Service through Gateway
http://localhost:8080/orders
http://localhost:8080/orders/456

How API Gateway Works
1. **Single Entry Point**: All client requests go through the API Gateway (port 8080)
2. **Request Routing**: Zuul routes requests based on path patterns:
   - /users/** → User Service (port 8081)
   - /orders/** → Order Service (port 8082)

Key Zuul Concepts
- **@EnableZuulProxy**: Enables Zuul proxy functionality
- **Routes**: Define how requests are mapped to backend services
- **Filters**: Pre, Route, Post, and Error filters for request processing
- **Load Balancing**: Built-in load balancing with Ribbon
- **Circuit Breaker**: Integration with Hystrix for fault tolerance

GitHub Link: https://github.com/iam-harshal-more/Api-geteway

═══════════════Thank you══════════════════