The Flight Listing App (FLApp) Case Study

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Lesson

• Understand how object-oriented, functional, & reactive streams programming is applied in a case study that lists airline flights via various web apps
Overview of the Flight Listing App (FLApp)
Overview of the Flight Listing App (FLApp)

- The Flight Listing App (FLApp) case study showcases a wide range of Java concurrency & parallelism frameworks that synchronously & asynchronously communicate with various Spring-based platforms to list airline flights.
Overview of the Flight Listing App (FLApp)

- FLApp provides a monolithic client-server architecture implemented via Spring to sequentially list airline flights using objects within one process (which could be accessed via a load balancer).

See gitlab.com/Creasor/flights-monolithic
Overview of the Flight Listing App (FLApp)

- The monolithic implementation of FLApp uses sync two-way calls & Java object-oriented programming features & functional sequential streams

See docs.oracle.com/javase/tutorial/collections/streams
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.

See [gitlab.com/Creasor/flights-microservices](https://gitlab.com/Creasor/flights-microservices) & [gitlab.com/Creasor/flights-reactive-microservices](https://gitlab.com/Creasor/flights-reactive-microservices)
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.
- Flight is a “front-end” app gateway that uses Eureka service discovery.

See [www.baeldung.com/spring-cloud-netflix-eureka](http://www.baeldung.com/spring-cloud-netflix-eureka)
Overview of the Flight Listing App (FLApp)

• Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment

• Flight is a “front-end” app gateway that uses Eureka service discovery
  • Used to find & communicate with back-end microservices
  • Without hard-coding ports & hostnames

See microservices.io/patterns/server-side-discovery.html
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.
- Flight is a “front-end” app gateway that uses Eureka service discovery.
- Used to find & communicate with back-end microservices.
  - Without hard-coding ports & hostnames.
  - Each back-end microservice registers with a service registry.

See [microservices.io/patterns/service-registry.html](microservices.io/patterns/service-registry.html)
Overview of the Flight Listing App (FLApp)

• Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment

• Flight is a “front-end” app gateway that uses Eureka service discovery

• Used to find & communicate with back-end microservices

• Without hard-coding ports & hostnames

• Each back-end microservice registers with a service registry

• Performed declaratively via annotations & property files

```java
@EnableDiscoveryClient
class AirportApplication {
    public static void main(...) {
        SpringApplication.run(AirportApplication.class, args);
    }
    
    server.port=0
    eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
    spring.cloud.eureka.enabled=true
    eureka.client.enabled=true
    ...
```

See microservices.io/patterns/service-registry.html
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.
- Flight is a “front-end” app gateway that uses Eureka service discovery.
- Used to find & communicate with back-end microservices.
- The Flight app gateway can locate the microservices it uses by name.

```java
List<String> getAirlineServices() {
    return discoveryClient.getServices().stream()
        .filter(id -> id.toLowerCase().contains("airline"))
        .collect(toList());
}
```
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.
- Flight is a "front-end" app gateway that uses Eureka service discovery.
  - Used to find & communicate with back-end microservices.
  - The Flight app gateway can locate the microservices it uses by name.
- RestTemplate performs sync calls.

```java
Airport[] airports = restTemplate
    .getForEntity("http://" + AIRPORT + "/" + AIRPORTS, Airport[].class)
    .getBody();
```

See [springframework/web/client/RestTemplate.html](https://springframework/web/client/RestTemplate.html)
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.
- Flight is a “front-end” app gateway that uses Eureka service discovery.
  - Used to find & communicate with back-end microservices.
  - The Flight app gateway can locate the microservices it uses by name.
- RestTemplate performs sync calls.
  - It uses Eureka to redirect HTTP requests to the microservice.

```java
Airport[] airports = restTemplate
                    .getForEntity("http://" + AIRPORT + "/" + AIRPORTS,
                                 Airport[].class)
                    .getBody();
```

Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment
- Flight is a “front-end” app gateway that uses Eureka service discovery
  - Used to find & communicate with back-end microservices
  - The Flight app gateway can locate the microservices it uses by name
  - RestTemplate performs sync calls
    - It uses Eureka to redirects HTTP requests to the microservice
    - Load balancing can also be enabled!

```java
Airport[] airports = restTemplate
    .getForEntity("http://" + AIRPORT + "/" + AIRPORTS, Airport[].class)
    .getBody();
```

See piotrminkowski.com/2020/05/13/a-deep-dive-into-spring-cloud-load-balancer
Overview of the Flight Listing App (FLApp)

• Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment

• Flight is a “front-end” app gateway that uses Eureka service discovery
  • Used to find & communicate with back-end microservices
  • The Flight app gateway can locate the microservices it uses by name
  • RestTemplate performs sync calls
  • WebClient performs async calls

```java
webClient
  .get()
  .uri(baseUrl + AIRPORT + "/" + AIRPORTS)
  .retrieve()
  .bodyToFlux(Airport.class);```

See springframework/web/reactive/function/client/WebClient.html
Overview of the Flight Listing App (FLApp)

• Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment

• Flight is a “front-end” app gateway that uses Eureka service discovery
  • Used to find & communicate with back-end microservices
  • The Flight app gateway can locate the microservices it uses by name

• RestTemplate performs sync calls
• WebClient performs async calls
  • It also uses Eureka & load balancing

```java
webClient
  .get()
  .uri(baseUrl + AIRPORT + "/" + AIRPORTS)
  .retrieve()
  .bodyToFlux(Airport.class);
```

See [spring.io/blog/2020/03/25/spring-tips-spring-cloud-loadbalancer](spring.io/blog/2020/03/25/spring-tips-spring-cloud-loadbalancer)
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment
  - Flight is a “front-end” app gateway that uses Eureka service discovery
  - The “back-end” microservices perform various tasks
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment
  - Flight is a “front-end” app gateway that uses Eureka service discovery
- The “back-end” microservices perform various tasks, e.g.
  - Return a list of all known airports
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment.

- Flight is a “front-end” app gateway that uses Eureka service discovery.

- The “back-end” microservices perform various tasks, e.g.
  - Return a list of all known airports.
  - Return currency exchange rates.

---

Microservices App

- Flight
- Airport
- ExchangeRate
- AA-airline
- SWA-airline
- …
Overview of the Flight Listing App (FLApp)

- Later FLApp versions list airline flights & related information via Spring microservices that can run in separate processes in a cluster environment
  - Flight is a “front-end” app gateway that uses Eureka service discovery
  - The “back-end” microservices perform various tasks, e.g.
    - Return a list of all known airports
    - Return currency exchange rates
    - Return flight info for various airlines
Overview of the Flight Listing App (FLApp)

- The object-oriented implementation of FLApp uses sync two-way calls & various Java concurrent Executor frameworks
  - e.g., Java threads & the Java executor framework

See docs.oracle.com/javase/tutorial/essential/concurrency
Overview of the Flight Listing App (FLApp)

- The functional implementation of FLApp uses sync & async two-way calls & various Java functional parallel & async programming frameworks
  - e.g., Java parallel streams & completable futures frameworks

See www.manning.com/books/modern-java-in-action
Overview of the Flight Listing App (FLApp)

• The reactive implementation of FLApp uses async two-way calls & various Java reactive streams frameworks that support various concurrency models
  • e.g., Project Reactor & RxJava

See en.wikipedia.org/wiki/Reactive_Streams
Overview of the Flight Listing App (FLApp)

- The FLApp case study also showcases advanced GUI, persistence, & testing frameworks & tools
- e.g., JPA, R2DBC, Android, & mocking tools

See spring.io/projects/spring-data-jpa, r2dbc.io, developer.android.com & mockk.io
End of the Flight Listing App (FLApp) Case Study