Overview of Spring WebMVC

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 the structure & functionality of the Spring WebMVC framework supported by Spring Boot 2.0

See [docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html](docs.spring.io/spring-framework/docs/3.2.x/spring-framework-reference/html/mvc.html)
Learning Objectives in this Lesson

- Understand the structure & functionality of the Spring WebMVC framework supported by Spring Boot 2.0, e.g.
  - Its concurrency model
Learning Objectives in this Lesson

- Understand the structure & functionality of the Spring WebMVC framework supported by Spring Boot 2.0, e.g.
  - Its concurrency model
  - Its communication model
Overview of Spring WebMVC Concurrency
Overview of Spring WebMVC Concurrency

- Spring WebMVC concurrency
- Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model

Diagram:

- Servlet Containers
- Servlet API
- Spring Security
- Spring MVC
- Spring Data Repositories
  - JDBC, JPA, NoSQL
Overview of Spring WebMVC Concurrency

- Spring WebMVC concurrency
- Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
- Each request is handled by a thread that blocks until it is able to fully process the request

A request to a list of flights from a database over the network might take a few seconds, which blocks threads from servicing other requests & responses

See en.wikipedia.org/wiki/Blocking_(computing)
Overview of Spring WebMVC Concurrency

- Spring WebMVC concurrency
- Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
- Each request is handled by a thread that blocks until it is able to fully process the request
- Blocking calls are a natural form of back pressure

See medium.com/@jayphelps/backpressure-explained-the-flow-of-data-through-software-2350b3e77ce7
Overview of Spring WebMVC Concurrency

- Spring WebMVC concurrency
- Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
  - Each request is handled by a thread that blocks until it is able to fully process the request
  - Blocking calls are a natural form of back pressure
  - Forces the caller to wait

See [en.wikipedia.org/wiki/Rate_limiting](en.wikipedia.org/wiki/Rate_limiting)
Overview of Spring WebMVC Concurrency

- Spring WebMVC concurrency
- Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
  - Each request is handled by a thread that blocks until it is able to fully process the request
  - Blocking calls are a natural form of back pressure
    - Forces the caller to wait
    - Eliminates the need for end-to-end rate control

See en.wikipedia.org/wiki/Rate_limiting
Overview of Spring WebMVC Communication

• Spring WebMVC concurrency
• Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
  • Each request is handled by a thread that blocks until it is able to fully process the request
• However, a server may need many threads to handle bursty clients

See www.baeldung.com/java-web-thread-pool-config
Overview of Spring WebMVC Communication

- Spring WebMVC concurrency
  - Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
    - Each request is handled by a thread that blocks until it is able to fully process the request
  - However, a server may need many threads to handle bursty clients
    - Traditional Java Thread objects consume non-trivial system resources..
Overview of Spring WebMVC Communication

- Spring WebMVC concurrency
  - Built on the Servlet API & uses a synchronous I/O architecture w/one-thread-per-request model
    - Each request is handled by a thread that blocks until it is able to fully process the request
    - However, a server may need many threads to handle bursty clients
    - Java 19’s “virtual threads” provide much more scalability

See [www.happycoders.eu/java/virtual-threads](http://www.happycoders.eu/java/virtual-threads)
Overview of Spring WebMVC Communications
Overview of Spring WebMVC Communication

- Spring WebMVC communications
- Network communication uses common Java types

See docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html
Overview of Spring WebMVC Communication

- Spring WebMVC communications
- Network communication uses common Java types
  - e.g., Java String & Integer objects, as well as List & Map collections

```java
public class FlightController {
    ...
    @GetMapping(AIRPORTS)
    List<Airport> getAirports() {
        return flightService
            .getAirports();
    }
    ...
}
```

See [flights-microservices/-/blob/master/src/main/java/server/flight/FlightController.java](flights-microservices/-/blob/master/src/main/java/server/flight/FlightController.java)
Overview of Spring WebMVC Communication

- Spring WebMVC communications
  - Network communication uses common Java types
- WebMVC endpoints send & return Java collections in one fell swoop
Overview of Spring WebMVC Communication

• Spring WebMVC communications
  • Network communication uses common Java types
• WebMVC endpoints send & return Java collections in one fell swoop
  • Client latency may suffer & thus not be as responsive as possible

See [en.wikipedia.org/wiki/Spinning_pinwheel](en.wikipedia.org/wiki/Spinning_pinwheel)
Overview of Spring WebMVC Communication

- Spring WebMVC communications
  - Network communication uses common Java types
  - WebMVC endpoints send & return Java collections in one fell swoop
    - Client latency may suffer & thus not be as responsive as possible
    - Memory is needed to buffer this data at multiple points

See english.stackexchange.com/questions/337497/what-is-meant-by-memory-hog
Overview of Spring WebMVC Communication

- Spring WebMVC communications
  - Network communication uses common Java types
- WebMVC endpoints send & return Java collections in one fell swoop
  - Client latency may suffer & thus not be as responsive as possible
  - Memory is needed to buffer this data at multiple points
  - Addressed by Spring WebFlux & reactive programming

See [docs.spring.io/spring-framework/docs/current/reference/html/web-reactive.html#webflux]
End of Overview of Spring WebMVC