SPRING 5 OVERVIEW

Nov 01, 2017

SPRING 5 OVERVIEW

NEW FEATURES EXPERIMENT

Gary Pan
gary_pan@epam.com
Nov 01, 2017

Agenda

- Quick overview of high level features
- Reactive programing
- Demo
- Q & A

New features in Spring Framework 5.0

- Baseline upgrades
- Usage of JDK 8 features in the Spring Framework code
- JDK 9 and Jigsaw modules
- HTTP/2 and Servlet 4.0
- Reactive programming support
- A functional web framework
- Kotlin support
- Spring Boot 2.0
- Dropped features

Baseline upgrades

- **JDK 8**
- **Java EE 7**
  - Servlet 3.1
  - JMS 2.0
  - JPA 2.1
  - JAX-RS 2.0
  - Bean Validation 1.1

- **minimum supported versions of prominent frameworks**
  - Hibernate 5
  - Jackson 2.6
  - EhCache 2.10
  - JUnit 5
  - Tiles 3

- **supported server versions**
  - Tomcat 8.5+
  - Jetty 9.4+
  - WildFly 10+
  - Netty 4.1+ (for web reactive programming with Spring Web Flux)
  - Undertow 1.4+ (for web reactive programming with Spring Web Flux)

Usage of JDK 8 features in Spring Framework code

- Java 8 default methods in core Spring interfaces
- Internal code improvements based on Java 8 reflection enhancements
- Use of functional programming in the framework code--lambdas and streams

- JDK 8+ for Spring Framework 5.x
- JDK 6+ for Spring Framework 4.x
- JDK 5+ for Spring Framework 3.x

• JDK 9 and Jigsaw modules

- JDK9 runtime compatibility
- support for Jigsaw modules
  - Java platform problems
    - Platform Bloat:
      - Internet of Things (IOT)
      - 10 MB -> 200+MB
    - JAR Hell
  - Open System Gateway initiative (OSGi)
    - imports: Other bundles that the module uses
    - exports: Packages that this bundle exports
  - Jigsaw bring modularity into Java
    - Defining and implementing a modular structure for JDK
    - Defining a module system for applications built on the Java platform

```
module my.app.db {
        requires java.sql;
        requires spring.jdbc;
}
```

HTTP/2 and Servlet 4.0

- HTTP/2 Enormous benefits over HTTP 1.1 (which dates back to 1996)
  - binary protocol
  - TLS (SSL) everywhere
  - connection multiplexing
  - headers compression
  - request prioritization
  - push of correlated resources
- Browsers already implement HTTP/2 over TLS

  major websites work with HTTP/2 already: Google, Twitter, etc
- Servlet 4.0
  - enforces support for HTTP/2 in Servlet containers
  - API features for stream prioritization and push resources

Reactive programming support

- one of the most important features of Spring Framework 5.0
- Microservices architectures event-based communication
- support for reactive web programming

Kotlin support

statically typed JVM language that enables code that is expressive, short, and readable.

- Short code

  setter()/getter()/equals()/hashCode()/toString()/copy()
- strongly typed

  ```
  val arrayList = arrayListOf("Item1", "Item2", "Item3")     // Type is ArrayList
  ```
- Named arguments

  ```
  var todo = Todo(description = "Learn Spring Boot", name = "Jack", targetDate = Date())
  ```
- default variables functional programming

  ```
  var first3TodosOfJack = students.filter { it.name == "Jack"}.take(3)
  ```
- default values for arguments

  ```
  import java.util.*
    data class Todo(var description: String, var name: String, var targetDate : Date = Date())
    fun main(args: Array<String>) {
      var todo = Todo(description = "Learn Spring Boot", name = "Jack")
    }
  ```

Kotlin support

Java

```java
public class User {
    private final String firstName;
    private final String lastName;
    private final int age;

    public User(String firstName, String lastName, int age) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.age = age;
    }

    public String getFirstName() {
        return firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public int getAge() {
        return age;
    }

    public String toString() {
        return firstName + " " + lastName + ", age " + age;
    }
}
```

```java
class Main {
    public static void main(String[] args) {
        System.out.println(new User("John", "Doe", 30));
    }
}
```

Kotlin

```kotlin
public class User(val firstName: String,
                  val lastName: String,
                  val age: Int) {

    fun toString() = "$firstName $lastName, age $age"

}
```

```kotlin
fun main(args : Array<String>) {
    println(User("John", "Doe", 30))
}
```

Spring Boot 2.0 new features

- The baseline JDK version is Java 8
- The baseline Spring Version is Spring Framework 5.0
- Spring Boot 2.0 has support for Reactive Web programming with WebFlux

- Minimum supported versions of some important frameworks
  - Jetty 9.4
  - Tomcat 8.5
  - Hibernate 5.2
  - Gradle 3.4

Dropped features

- Portlet
- Velocity
- JasperReports
- XMLBeans
- JDO
- Guava

Spring Framework 4.3 has support until 2019

Reactive programing

Reactive approaches

Most applications from a few years back had the luxury of the following:

- Multi-second response times
- Multiple hours of offline maintenance
- Smaller volumes of data

New devices (mobiles, tablets, and so on) and newer approaches (cloud-based) have emerged

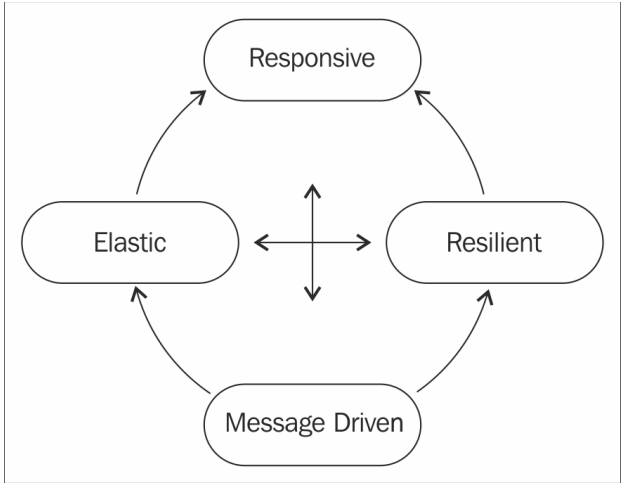
- Sub-second response times
- 100% availability
- An exponential increase in data volumes

Reactive Manifesto

Systems built as Reactive Systems are more flexible, loosely coupled, and scalable. This makes them easier to develop and amenable to change. They are significantly more tolerant of failure, and when failure does occur, they meet it with elegance rather than disaster. Reactive Systems are highly responsive, giving users effective interactive feedback.

Characteristics of Reactive

- **Responsive**
- **Resilient**
- **Elastic**
- **Message driven**



responsive to different kinds of stimulus

- React to events
- React to load
- React to failures
- React to users

Reactive Infrastructure All Around

- Reactive datastore drivers becoming available
  Postgres, Mongo, Couchbase, Redis

- Reactive HTTP clients
  Netty, Jetty, OkHttp

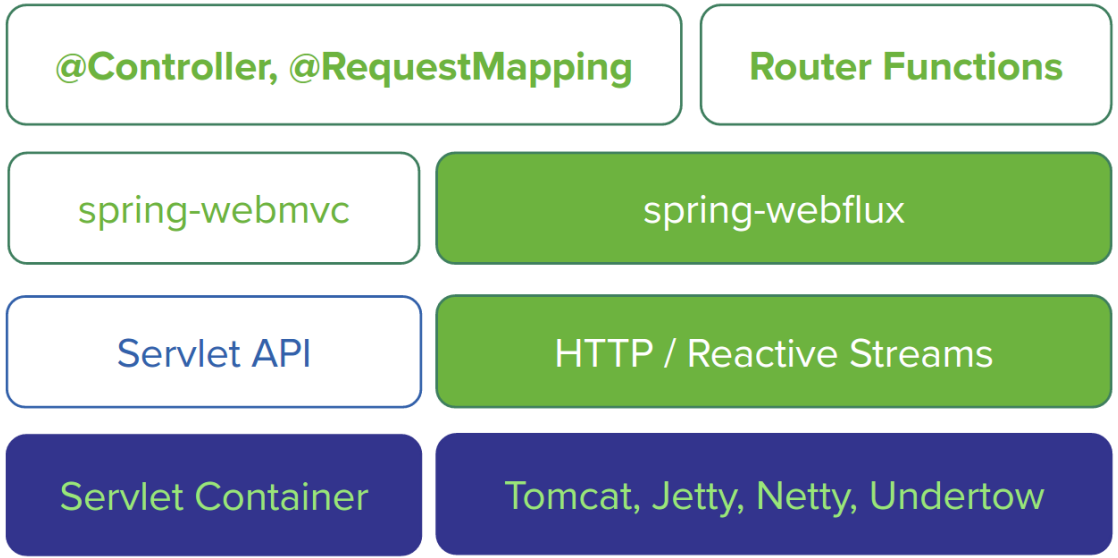
- Reactive Streams Commons project of Spring
  Servlet adapters: by default against Servlet 3.1 async I/O
  native container SPI for more efficiency at runtime
  a collaboration between Spring and Jetty / Tomcat

Reactive programing support

Reactive Streams: Language-neutral attempt to define reactive APIs.

Reactor: Java implementation of Reactive Streams provided by the Spring Pivotal team.

Spring WebFlux: Enables the development of web applications based on reactive programming. Provides a programming model similar to Spring MVC.

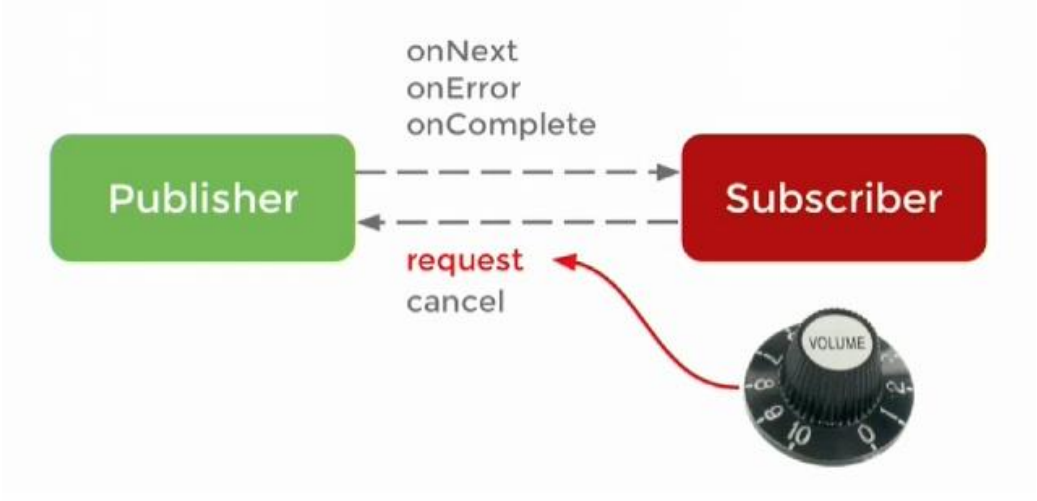


Reactive programing

- Reactive Streams
  - a minimal set of interfaces, methods, and protocols to enable reactive programming
  - a language-neutral approach with implementation in the Java (JVM-based) and JavaScript languages
  - Multiple transport streams (TCP, UDP, HTTP, and WebSockets) are supported
    - Interface Publisher
    - Interface Subscriber
    - Interface Subscription
  - repackaged into JDK 9 as java.util.concurrent.Flow

Reactive programing

REACTOR 3: REACTIVE STREAMS WITH BACKPRESSURE

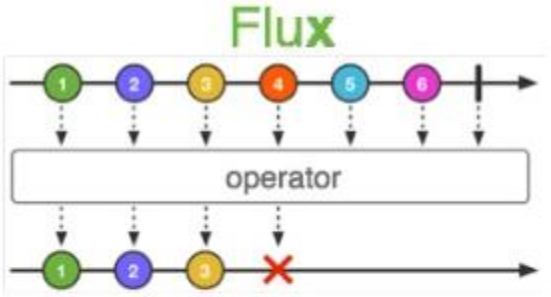


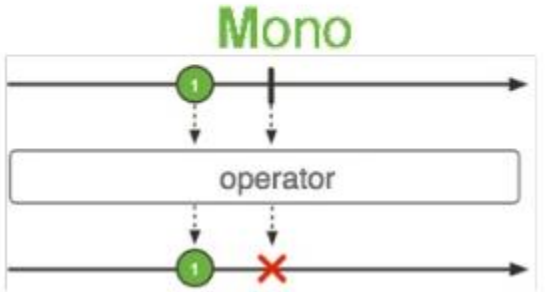
Reactive programing

- Reactor

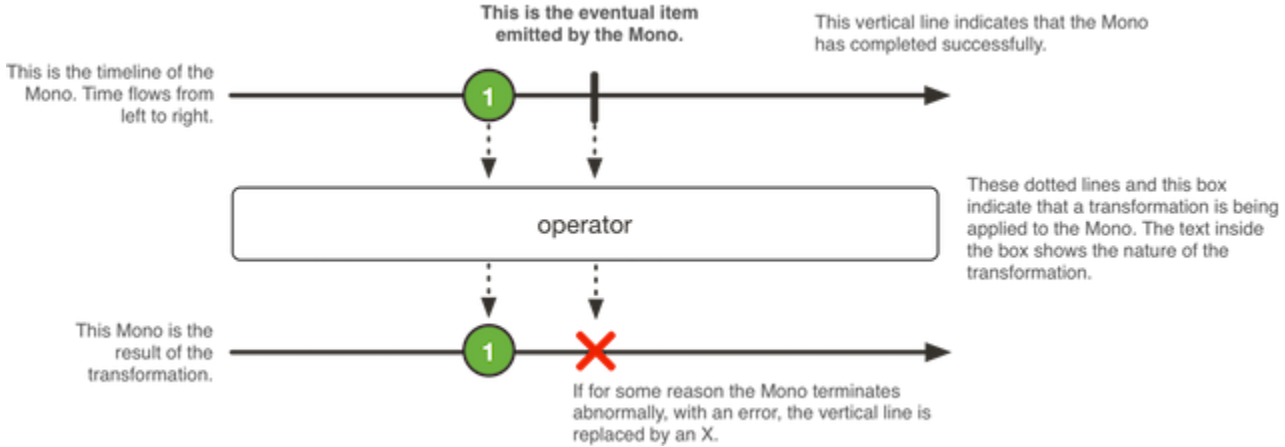
  a reactive framework from the Spring Pivotal team. It builds on top of Reactive Streams.

  - Flux: Flux represents a Reactive Stream that emits 0 to n element
  - Mono: Mono represents a Reactive Stream that emits either no elements or one element





Reactive programing

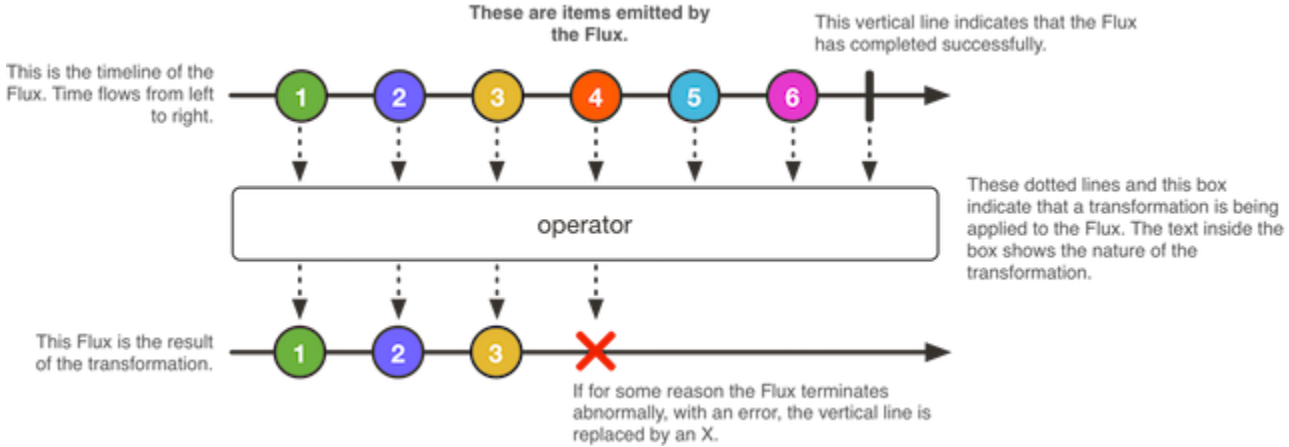


- MONO

```
@Test
public void monoExample() throws InterruptedException {
  Mono<String> stubMonoWithADelay =   Mono.just("Ranga").delayElement(Duration.ofSeconds(5));

   stubMonoWithADelay.subscribe(System.out::println);
 }
```

Reactive programing



- FLUX

```
@Test
public void simpleFluxStream() {
   Flux<String> stubFluxStream = Flux.just("Jane", "Joe");
   stubFluxStream.subscribe(new SystemOutConsumer());
 }
```

```
class SystemOutConsumer implements Consumer<String> {
    @Override
    public void accept(String t) {
      System.out.println("Received " + t + " at " + new Date());
    }
  }
```

Reactive Web Framework

- Spring Web Reactive

.	Spring MVC	Spring Web Reactive
Use	Traditional web application	Reactive web applications
Programming Model	`@Controller` with `@RequestMapping`	The same as Spring MVC
Base API	The Servlet API	Reactive HTTP
Runs on	Servlet Containers	Servlet Containers(>3.1), Netty, and Undertow

Functional web Programing

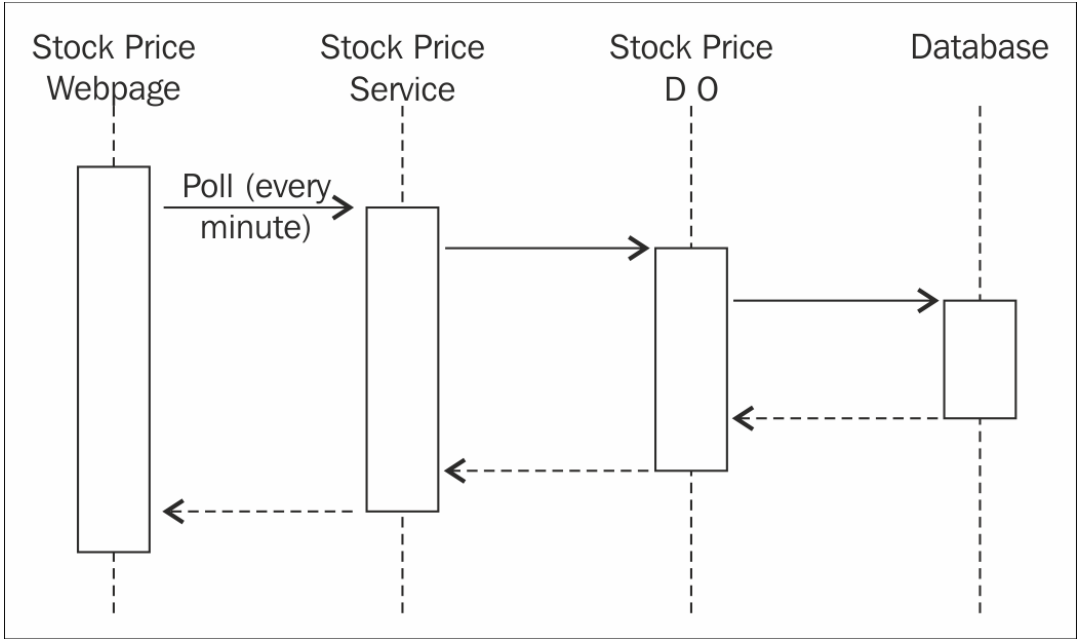
- Spring Web Reactive
  - RouterFunction

    evaluates the matching condition to route requests to the appropriate handler function

```
RouterFunction<String> route =
route(GET("/hello-world"), request -> Response.ok().body(fromObject("Hello World")))
.and(route(.....));
```

Reactive use case - a stock price page

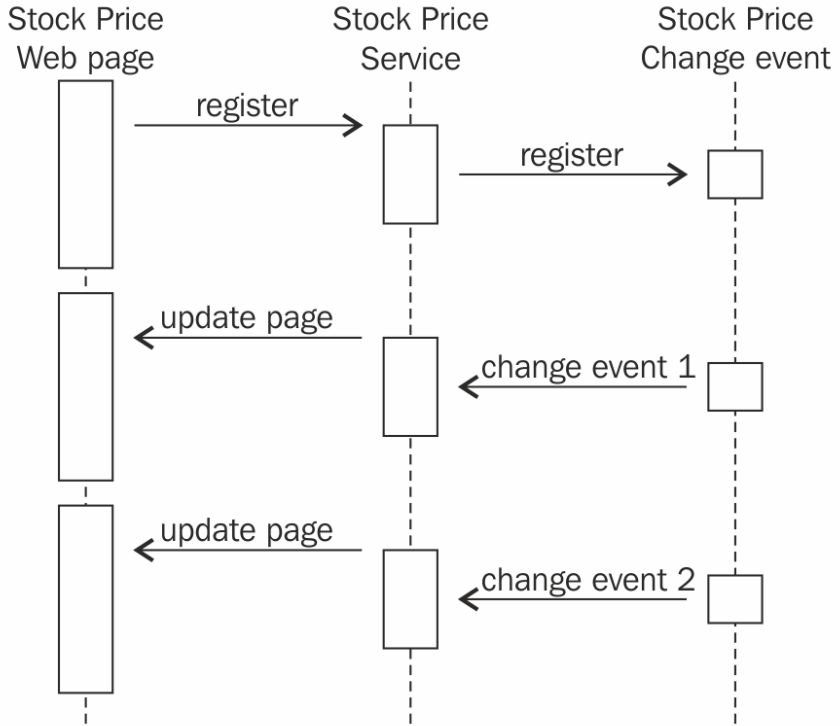
- The traditional approach
  - Polling
    AJAX requests
  - Long polling (Hanging GET / COMET)
    websocket



Reactive use case - a stock price page

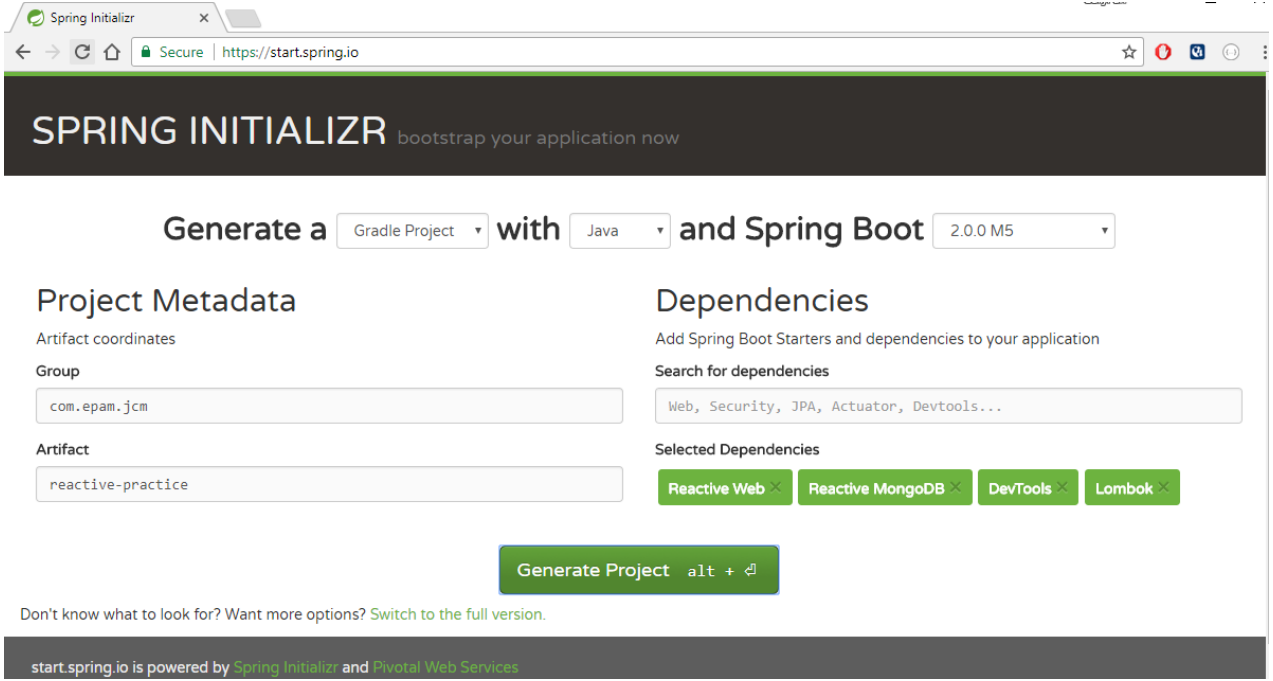
- The reactive approach
  1. Subscribing to events
  2. Occurrence of events
  3. Unregistering
- Server Side Events
  - MIME type: text/event-stream
  - HTML5: EventSource(no IE browser)

  browser automatically reconnect to the source after ~3 seconds



Demo

Demo: start with creating a project using Spring Initializr (http://start.spring.io/)



Spring 5 Overview

Q&A

Links

https://docs.spring.io/spring/docs/5.0.0.BUILD-SNAPSHOT/spring-framework-reference/htmlsingle/#web-reactive

https://stackify.com/reactive-spring-5/

http://start.spring.io/

https://spring.io/blog/2016/06/07/notes-on-reactive-programming-part-i-the-reactive-landscape

https://dzone.com/articles/spring-5-reactive-web-services
https://oschina.net/translate/spring-5-reactive-web-services (zh)

https://medium.com/@ggonchar/reactive-spring-5-and-application-design-impact-159f79678739