Introduction to Spring Framework

Trayan Iliev

tiliev@iproduct.org
http://iproduct.org

About me

Trayan Iliev

– CEO of IPT – Intellectual Products & Technologies

– Oracle® certified programmer 15+ Y

– end-to-end reactive fullstack apps with Java, ES6/7, TypeScript, Angular, React and Vue.js

– 12+ years IT trainer

– Voxxed Days, jPrime, jProfessionals, BGOUG, BGJUG, DEV.BG speaker

– Organizer RoboLearn hackathons and IoT enthusiast (http://robolearn.org)

What Will You Learn in the Course?

❖Component-based architecture, Inversion of Control (IoC) principle, and Dependency Injection (DI) pattern

❖Novelties in Spring Framework 5 and Java 8 & 9

❖Use Spring Boot 2 to rapidly develop Spring applications

❖Develop/secure Spring-based web applications and services using Spring MVC, WebSocket, and Spring Security

❖Manage data with SQL (PostgreSQL) and NoSQL (Mongo) databases with Spring Data: JDBC, Hibernate & JPA

❖Develop reactive REST services with Spring WebFlux

❖Test Spring applications using JUnit 5, Spring MVC Test, WebTestClient, TestContext, and Mock objects

Where to Find the Demo Code?

Introduction to Spring 5 demos and examples are available @ GitHub:

https://github.com/iproduct/course-spring5

Agenda for This Session

❖Introduction to Spring

❖Evolution of Spring framework

❖Main features

❖Spring main modules

❖Introduction to Maven and Gradle

❖Building a HelloSpring application using XML and annotation-based configurations

❖Introduction to Spring Boot 2.0 – building *HelloSpringMVC* & *HelloSpringWebFlux* simple web applications using *spring-boot-starter-web* & *spring-boot-starter-webflux*

❖Going reactive with *WebFlux* and *Spring Reactor* – intro

Why Spring?

❖ The origin of Spring Framework: Expert One-on-One: J2EE Design and Development by Rod Johnson (Wrox, 2002)

❖ For more than 15 years Spring has become a major Java enterprise development framework

❖ A lot of projects, features, and great community support

❖ Supports different application architectures, including messaging, transactional data, persistence, and web

❖ Out of the box support for Java 9 modules in Spring 5

❖ Integrates carefully selected Java EE specifications

❖ Spring 5 baseline: Java 8 & JavaEE 7 (Servlet 3.1, JPA 2.1)

Java EE Specs Supported by Spring

❖ Servlet API (JSR 340)

❖ WebSocket API (JSR 356)

❖ Concurrency Utilities (JSR 236)

❖ JSON Binding API (JSR 367)

❖ Bean Validation (JSR 303)

❖ JPA (JSR 338)

❖ JMS (JSR 914)

❖ JTA/JCA transaction coordination

❖ Dependency Injection (JSR 330)

❖ Common Annotations (JSR 250)

Which Problems Spring Addresses?

❖ Scalability and modularity

❖ Boiler plate code – using templates (JDBCTemplate, HibernateTemplate) and aspects (advises)

❖ Handling non-functional requirements – transactions, load scaling, security, logging, testability, maintainability, etc.

❖ Unit testing and integration testing

❖ Complex frameworks/application servers – POJO vs. EJB

❖ Code coupling – interfaces + Dependency Injection (DI)

❖ Separating What? From How? - declarative programmimg using XML config files, annotations & functional composition

Spring Framework Main Features

❖Core technologies – dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP.

❖Testing – mock objects, TestContext framework, Spring MVC Test, WebTestClient.

❖Data Access – transactions, DAO support, JDBC, ORM, Marshalling XML.

❖Spring MVC and Spring WebFlux web frameworks

❖Integration – remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.
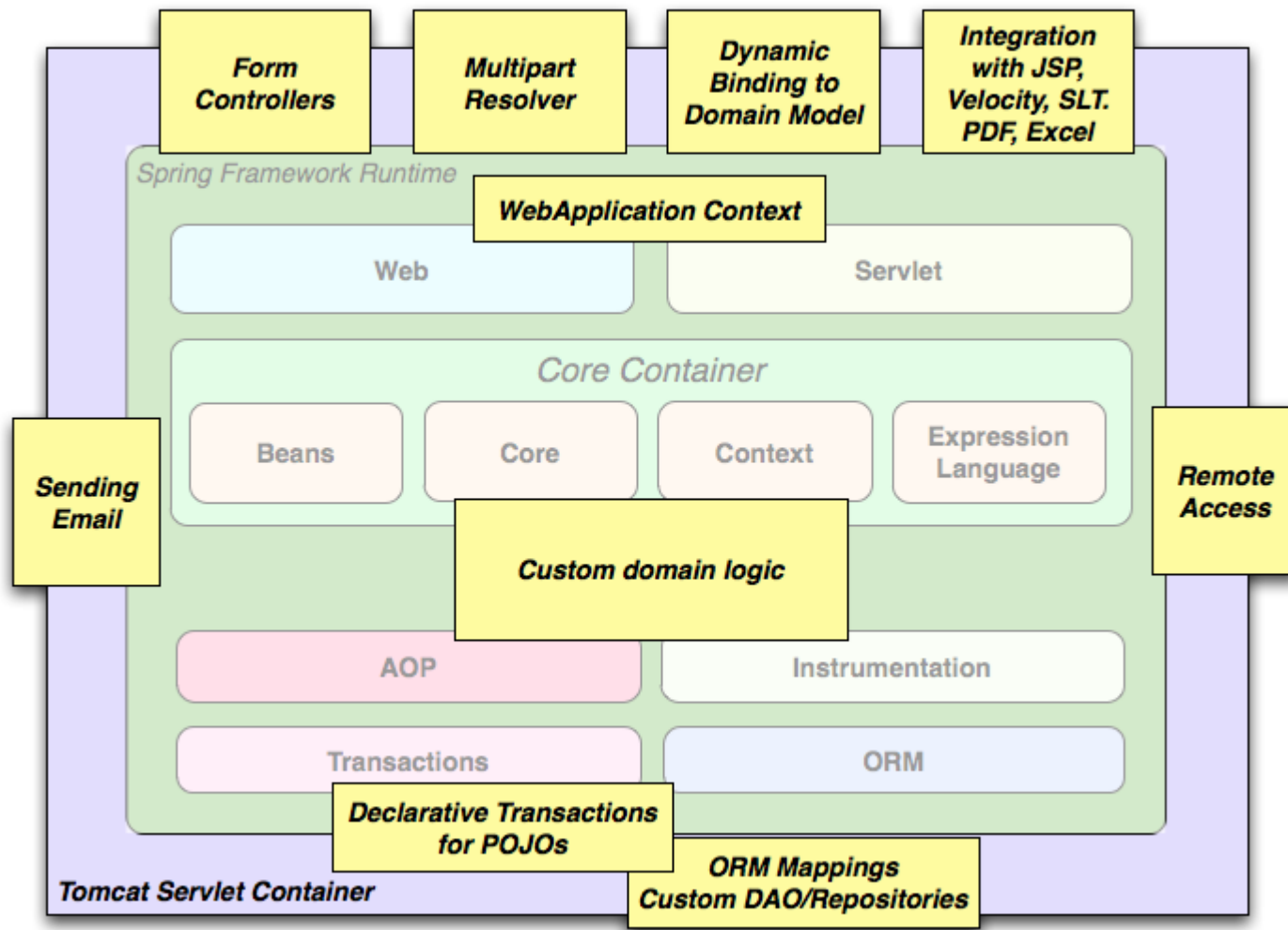
❖Languages – Kotlin, Groovy, dynamic languages.

Spring Framework 4.2 Main Modules
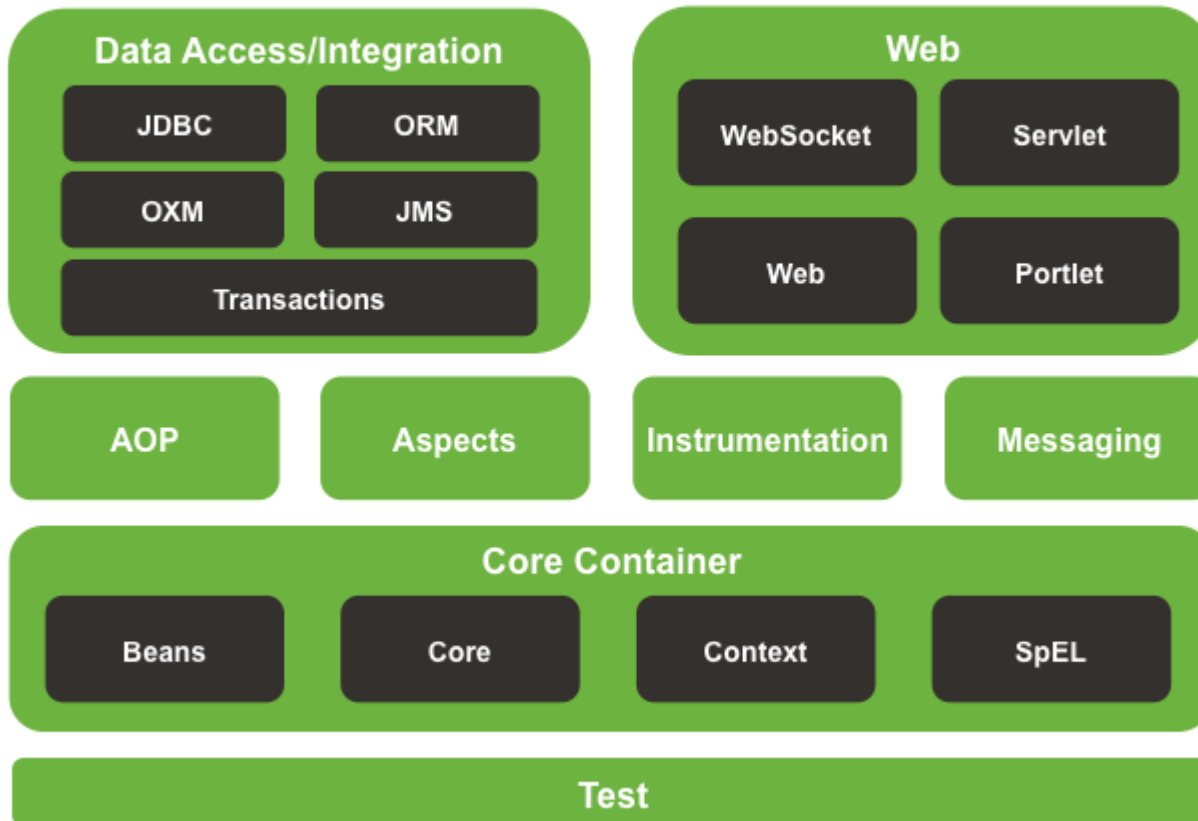
Fully Fledged Spring Web Application

Remoting Application
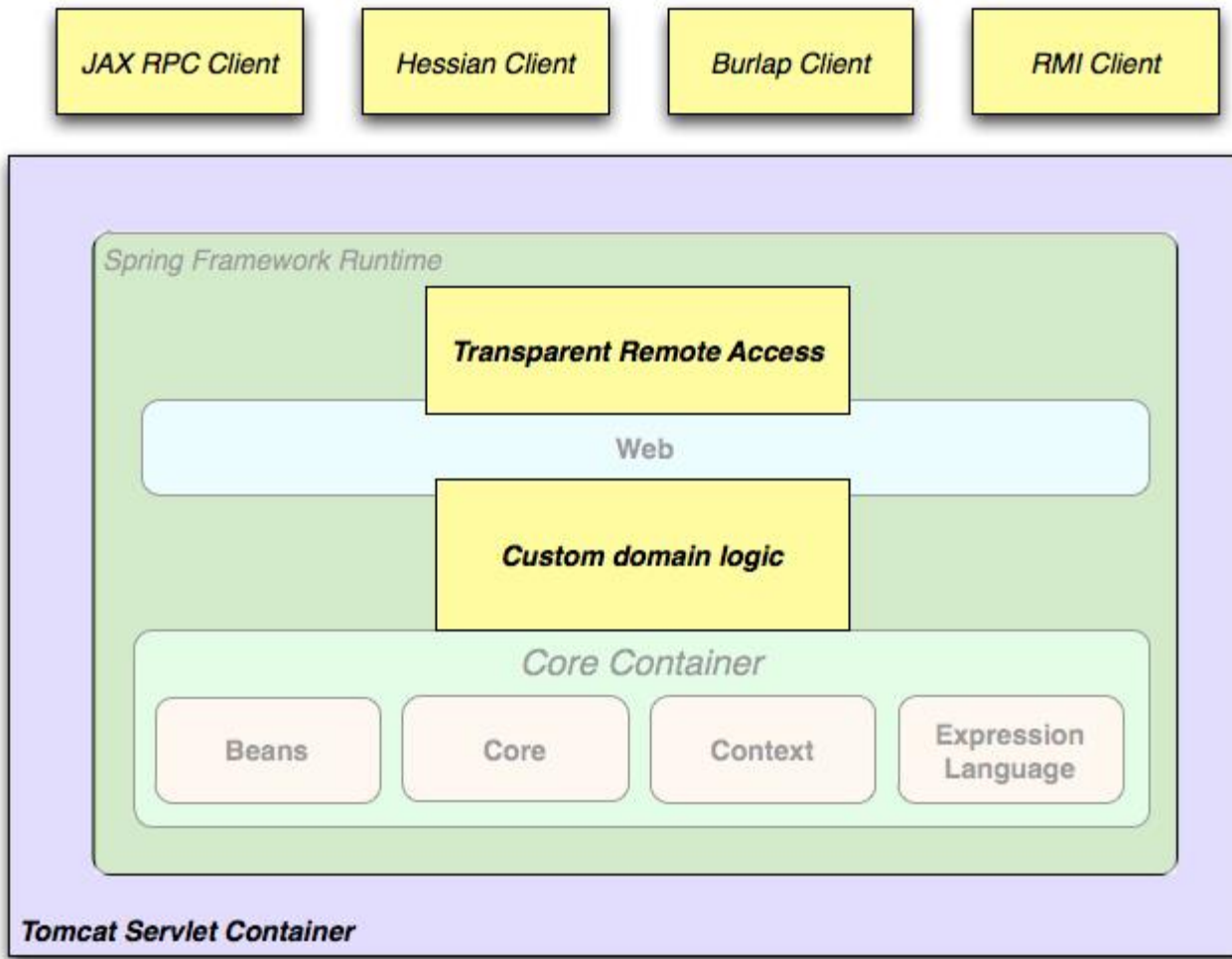
Spring 5 Framework Modules

Spring Boot 2.0

Web Servlet: Spring MVC, WebSocket, SockJS, STOMP

Web Reactive: Spring WebFlux, WebClient, WebSocket

Data Access: Transactions, DAO support, JDBC, ORM, OXM

Integration: Remoting, JMS, JCA, JMX, Email, Tasks, Scheduling, Cache

Spring Core: IoC container and beans, Events, Resources, i18n, Validation, Data Binding, Type Conversion, SpEL, AOP

Spring Testing: Mock objects, TestContext, MVC Test, WebTestClient

Evolution of Spring Framework - I

❖ Spring 1.x – Spring Core, Spring Context, Spring DAO, Spring ORM, Spring AOP, Spring Web, Spring WebMVC

❖ Spring 2.x (2006) – declarative transactions, @AspectJ, JPA, JMS, MVC form tags, Portlet MVC, Acegi Security

❖ Spring 2.5 (2007) – @Autowired, JSR-250(@Resource, @PostConstruct, @PreDestroy), stereotype annotations (@Component, @Repository, @Service, @Controller), automatic classpath scanning, AOP updates, TestContext

❖ Spring 3.x (2009) – Java-based @Configuration model, Spring Expression Language (SpEL), JSR-303:Bean Validation,REST

❖ Spring 3.1.x (2009) – WebApplicationInitializer, @Cacheable, @Profile, @EnableTransactionManagement..., c: namespace

Evolution of Spring Framework - II

❖ Spring 4.x (2013, Pivotal) – Java 8, Spring Boot, WebSocket, SockJS, and STOMP messaging, composed annotations, improvements in the core container, CORS, Hibernate 5.0, Spring IO, Spring XD

❖ Spring 5.x (2017) – JDK 9, Junit 5, XML configuration namespaces streamlined to unversioned schemas, Protobuf 3.0, Java EE7 API level required in Servlet 3.1, Bean Validation 1.1, JPA 2.1, JMS 2.0. Tomcat 8.5+, Jetty 9.4+, Wildfly 10+, Reactor, WebFlux, Spring Vault, Spring Cloud Stream, Micrometer

Spring Design Phylosophy

❖Provide choice at every level. Spring lets you defer design decisions as late as possible – e.g. persistence providers, infrastructure, third-party APIs

❖Accommodate diverse perspectives – not opinionated

❖Maintain strong backward compatibility

❖Care about API design – intuitive APIs

❖Code quality – high standards, meaningful, current, and accurate javadoc, clean code structure with no circular dependencies between packages.

Top New Features in Spring 5

❖Reactive Programming Model

❖Spring Web Flux – takes advantage of multi-core processors, handles massive number of connections

❖Reactive DB repositories & integrations + hot event streaming: MongoDB, CouchDB, Redis, Cassandra, Kafka

❖JDK 8+ and Java EE 7+ baseline

❖Testing improvements – WebTestClient (based on reactive WebFlux WebClient)

❖Kotlin functional DSL

Spring 5 Web Application Building Blocks

Spring Boot 2.0

Project Reactor

Servlet Stack (one request per thread)	Reactive Stack (async IO)
Every JEE Servlet Container (tomact, jetty, undertow, ...)	Nonblocking NIO Runtimes (Netty, Servlet 3.1 Containers)
Spring Security	Spring Security Reactive
Spring MVC	**Spring WebFlux**
Spring Data Repositories JDBC, JPA, NoSQL	Spring Data Reactive Repositories Mongo, Cassandra, Redis, Couchbase

Maven Dependency Management

❖Apache Maven – https://spring.io/guides/gs/maven/

❖Common arguments: `mvn compile, mvn package, mvn install, mvn clean deploy site-deploy`

❖Example configuration:

```xml
<?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>org.iproduct.spring</groupId>
    <artifactId>01-introduction-maven</artifactId>
    <version>1.0-SNAPSHOT</version>
```

Maven Configuration (continued)

```xml
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.0.5.RELEASE</version>
    </dependency>
</dependencies>

<repositories>
    <repository>
        <id>io.spring.repo.maven.release</id>
        <url>http://repo.spring.io/release/</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
</repositories>
```

Maven Configuration (continued)

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
                <source>9</source>
                <target>9</target>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>
```

Maven Configuration (enhanced)

```xml
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-framework-bom</artifactId>
            <version>5.0.5.RELEASE</version>
            <type>pom</type>
            <scope>import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>

<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
    </dependency>
</dependencies>
```

Gradle Dependency Management

❖ Gradle – https://spring.io/guides/gs/gradle/

❖ Init new project/ convert exisitng from Maven: **gradle init**

❖ Build project: **gradle build**

❖ Build project: **gradle run**

❖ Example configuration:

```
group 'org.iproduct.spring'
version '1.0-SNAPSHOT'
plugins {
    id 'java'
    id 'application'
}
mainClassName='course.spring.coredemo.SpringAnnotationConfigDI'
sourceCompatibility = 11
```

Gradle Configuration (continued)

```
task runApp(type : JavaExec ) {
    classpath = sourceSets.main.runtimeClasspath
    main = 'course.spring.coredemo.SpringAnnotationConfigDI'
}

repositories {
    mavenLocal()
    mavenCentral()
    maven { url "https://repo.spring.io/snapshot" }
    maven { url "https://repo.spring.io/milestone" }
}

dependencies {
    implementation group: 'org.springframework',
            name: 'spring-context', version: '5.3.7'
    testImplementation group: 'junit',
            name: 'junit', version: '4.12'
}
```

Making Projects Easy: Spring Boot 2

Tips & Tricks

❖In order to automatically create **/src** folder when creating new Gradle project with IntelliJ IDEA, check that the following option is enabled: *Settings / Build, Execution, Depolyment / Gradle / Create directories for empty content roots automatically*

❖If you use project Lombok and Lombok plugin in IntelliJ – don't forget to enable Annotation processing from: *Settings / Build, Execution, Deployment / Compiler / Annotation Processors*.

❖Don't forget to add Lombok plugin from: *Settings/Plugins*
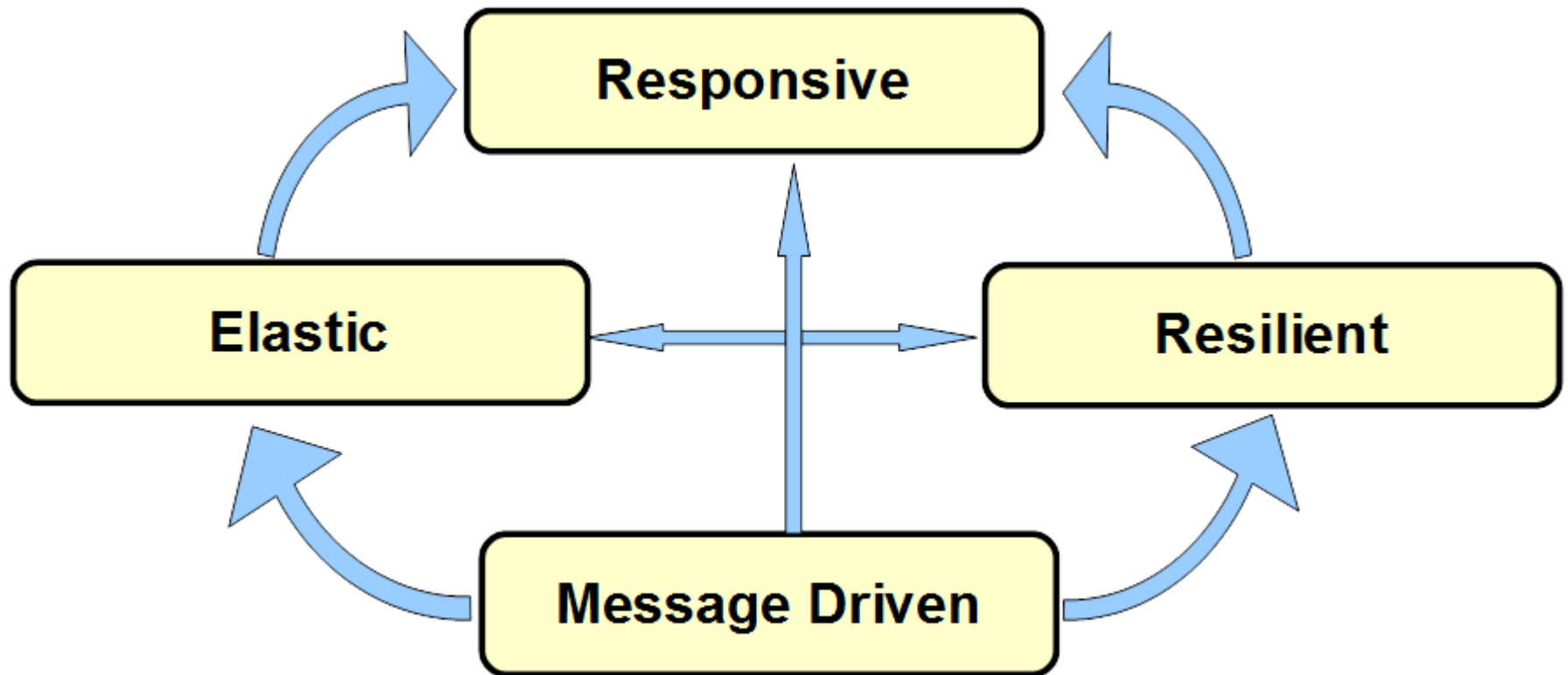
Synchronous vs. Asynchronous IO

Synchronous
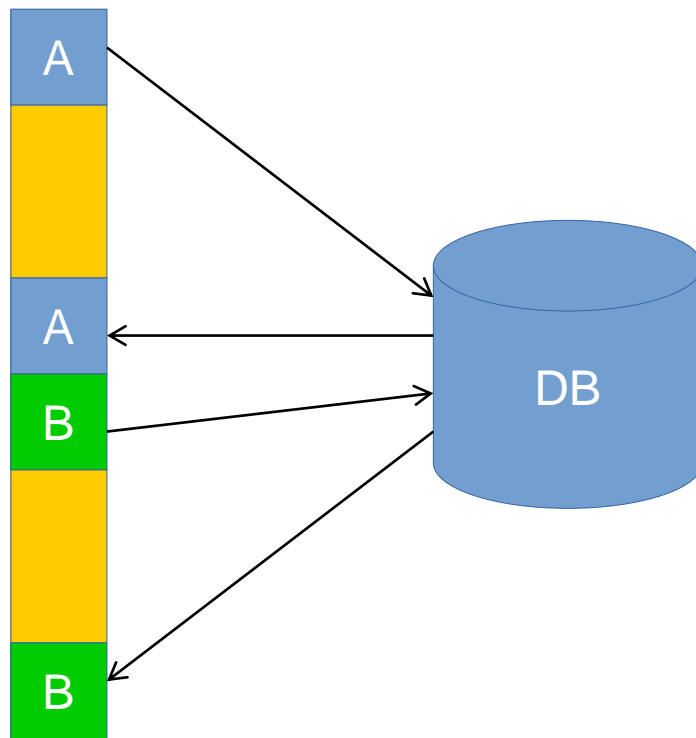
Asynchronous
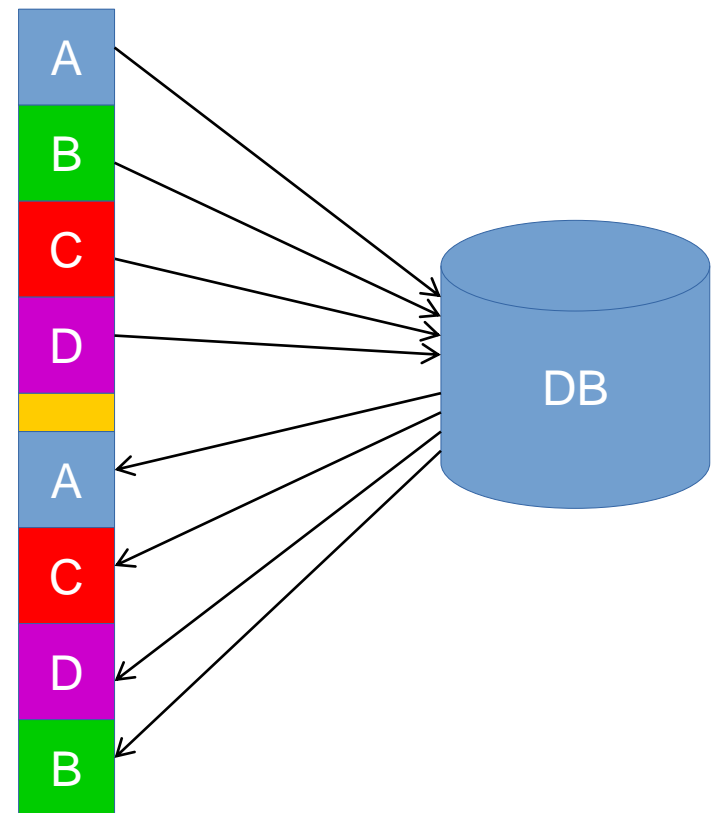
Reactive Manifesto

Spring 5 WebFlux

@Controller, @RequestMapping	Router Functions

spring-webmvc	spring-webflux
Servlet API	HTTP / Reactive Streams
Servlet Container	Tomcat, Jetty, Netty, Undertow

Imperative and Reactive

❖**Reactive Programming:** using static or dynamic data flows and propagation of change

Example: a := b + c

❖**Functional Programming:** evaluation of mathematical functions,

➢Avoids changing-state and mutable data, declarative programming

➢Side effects free => much easier to understand and predict the program behavior.

Example: books.stream().filter(book -> book.getYear() > 2010) .forEach(System.out::println)

Functional Reactive (FRP)

According to **Connal Elliot's** (ground-breaking paper @ Conference on Functional Programming, 1997), **FRP** is:

(a) Denotative

(b) Temporally continuous

FRP = Async Data Streams

❖ **FRP** is asynchronous data-flow programming using the building blocks of functional programming (e.g. map, reduce, filter) and explicitly modeling time

❖ Used for GUIs, robotics, and music. Example (RxJava):

```java
Observable.from(
    new String[]{"Reactive", "Extensions", "Java"})
    .take(2).map(s -> s + " : on " + new Date())
    .subscribe(s -> System.out.println(s));
```

Result:
```
Reactive : on Wed Jun 17 21:54:02 GMT+02:00 2015
Extensions : on Wed Jun 17 21:54:02 GMT+02:00 2015
```

Reactive Streams Spec.

❖Reactive Streams – provides standard for asynchronous stream processing with non-blocking back pressure.

❖Minimal set of interfaces, methods and protocols for asynchronous data streams

❖April 30, 2015: has been released version 1.0.0 of Reactive Streams for the JVM (Java API, Specification, TCK and implementation examples)

❖Java 9: java.util.concurrent.Flow

Reactive Streams Spec.

❖**Publisher** – provider of potentially unbounded number of sequenced elements, according to Subscriber(s) demand.

Publisher.subscribe(Subscriber) => **onSubscribe onNext* (onError | onComplete)?**

❖**Subscribe**r – calls **Subscription.request(long)** to receive notifications

❖**Subscription** – one-to-one **Subscriber ↔ Publisher**, request data and cancel demand (allow cleanup).

❖**Processor** = **Subscriber** + **Publisher**

Project Reactor

❖Reactor project allows building high-performance (low latency high throughput) non-blocking asynchronous applications on JVM.

❖Reactor is designed to be extraordinarily fast and can sustain throughput rates on order of 10's of millions of operations per second.

❖Reactor has powerful API for declaring data transformations and functional composition.

❖Makes use of the concept of Mechanical Sympathy built on top of Disruptor / RingBuffer.
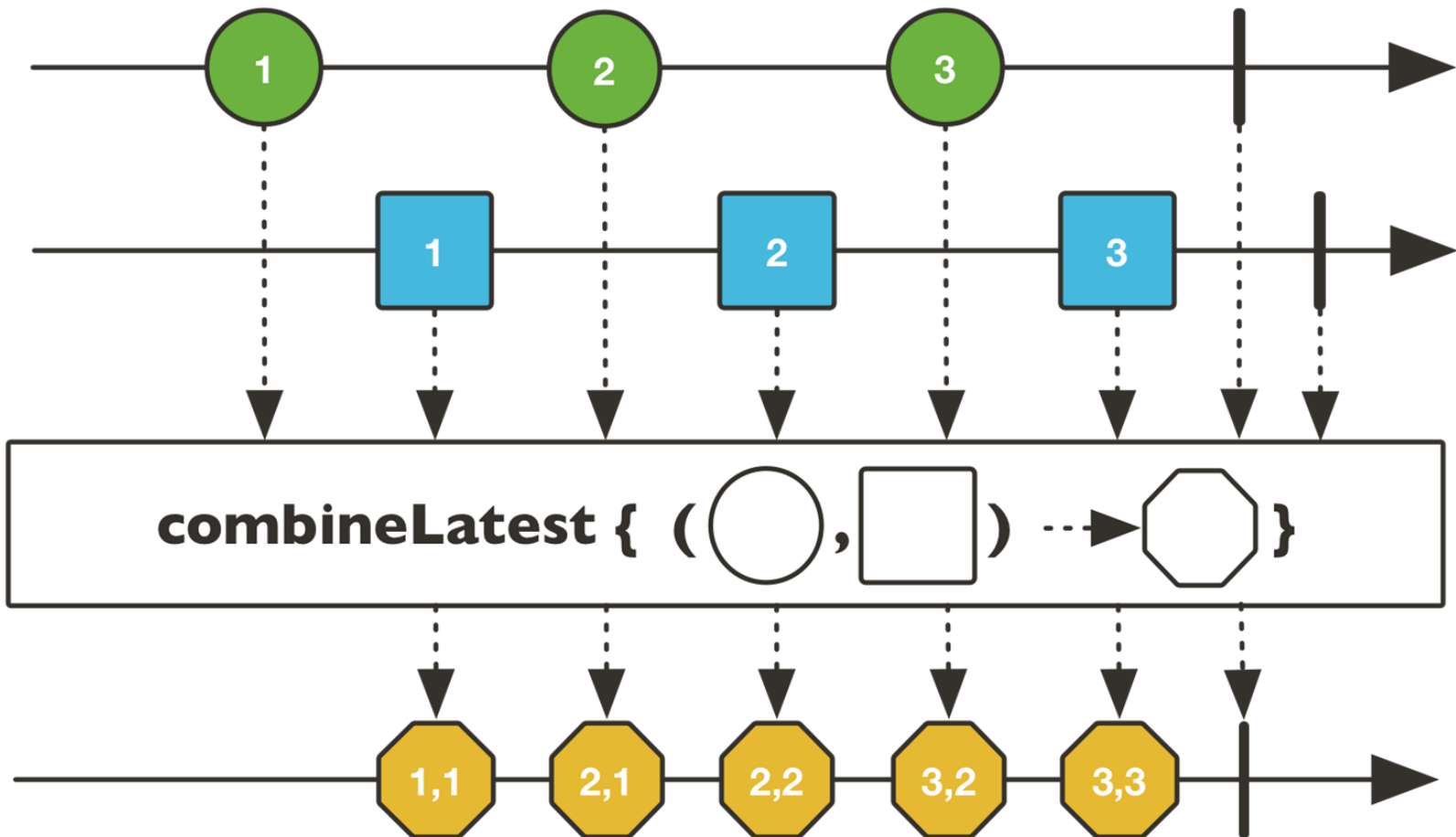
Reactor Projects

ADDONS

Spring

CORE

Reactive Streams
Commons

Rx

IPC – Netty, Kafka, Aeron
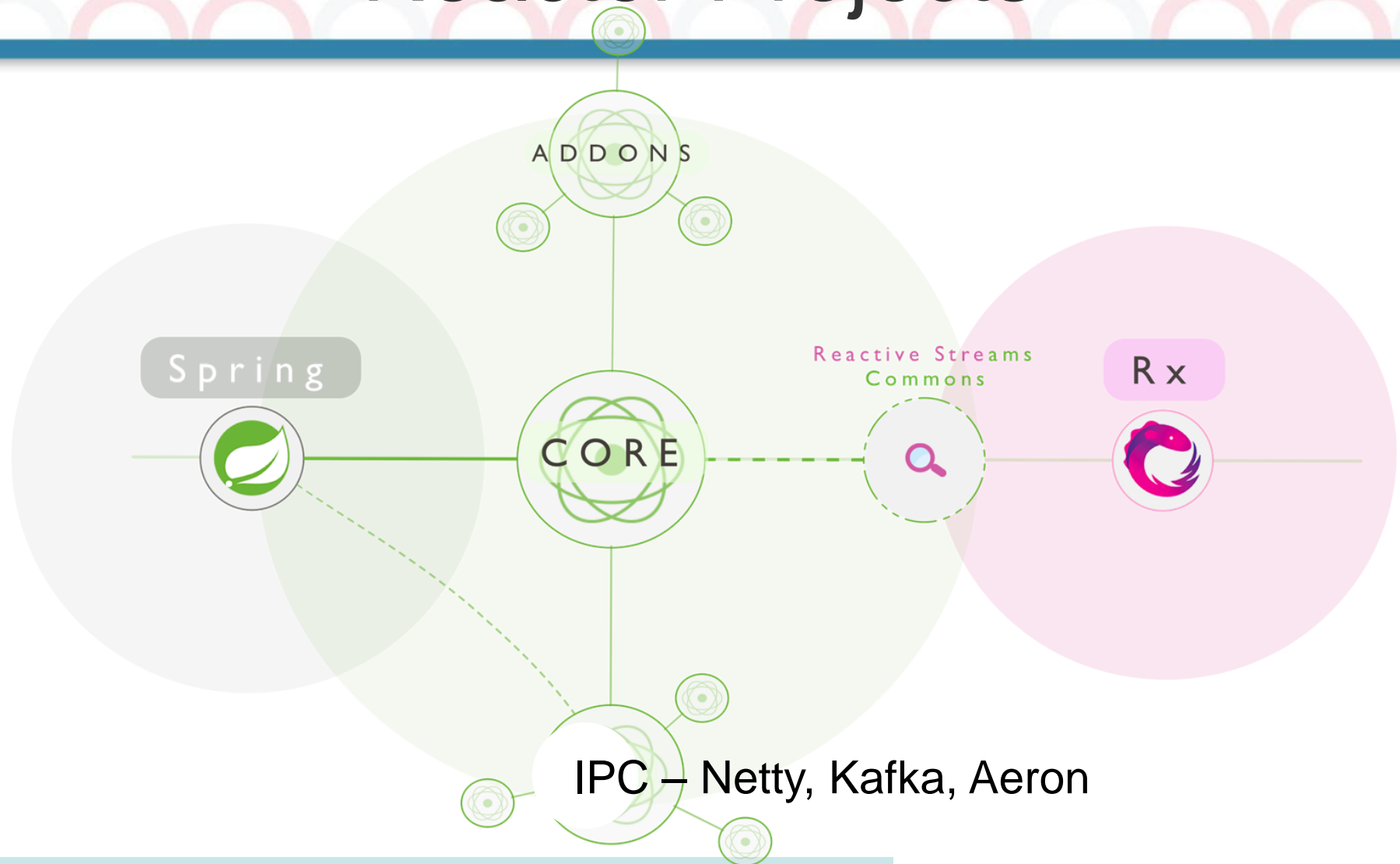
Reactor Flux
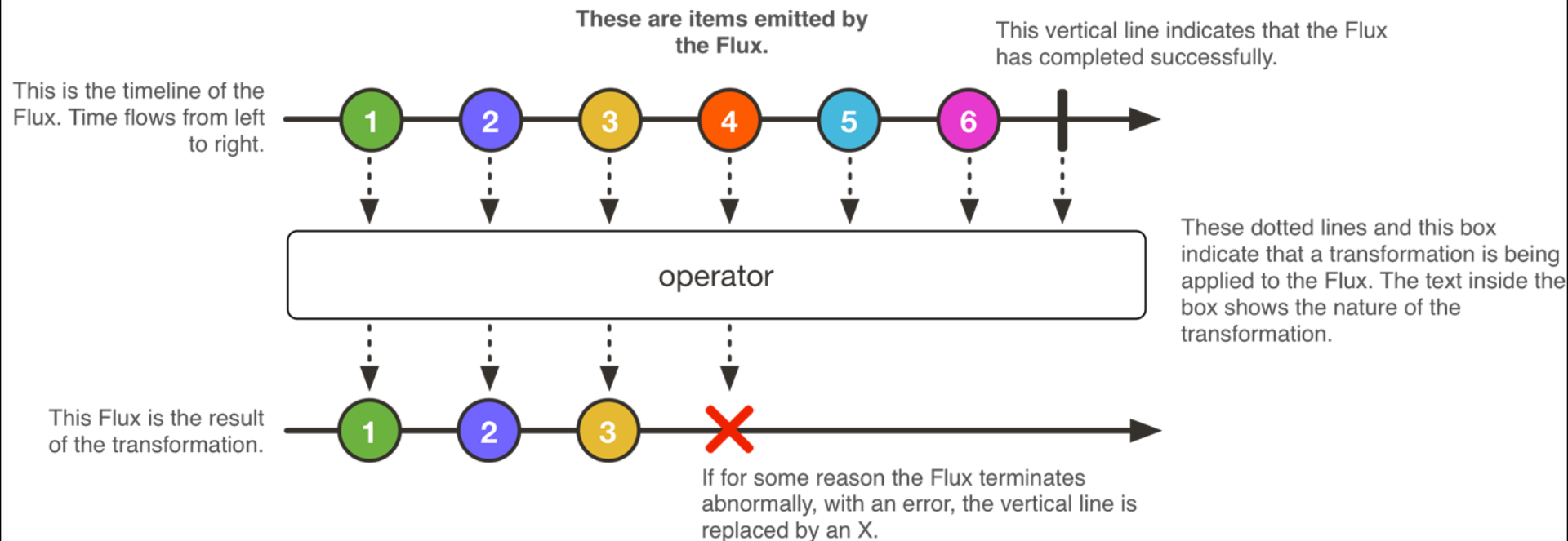
Example: Flux.combineLatest()

Redux == Rx Scan Opearator

scan { (◇ , □) -▷ ◇ }
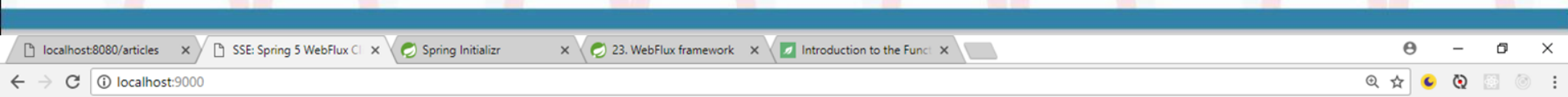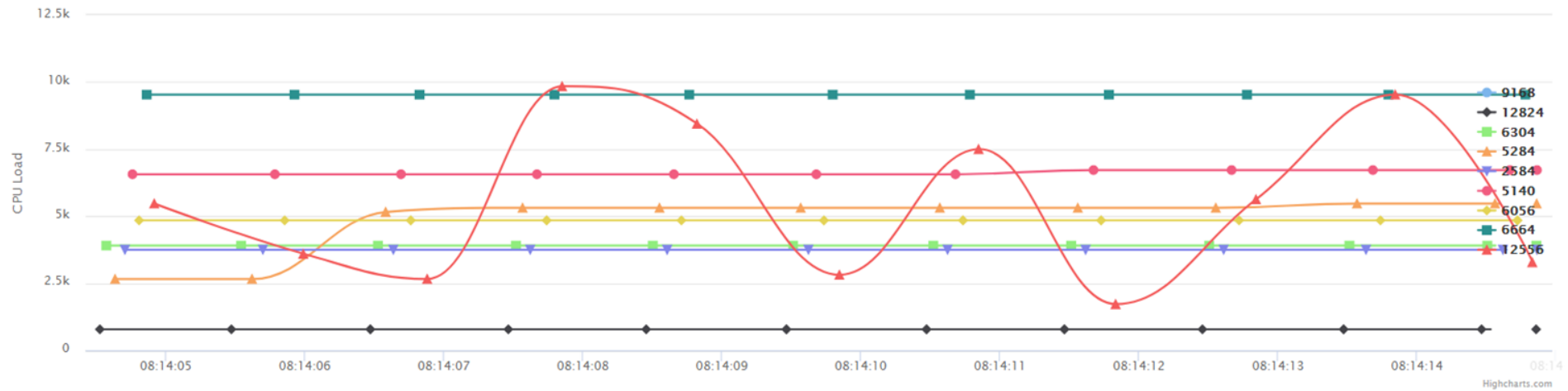
Demo Time :)

Thank's for Your Attention!

Trayan Iliev

CEO of IPT – Intellectual Products & Technologies

http://iproduct.org/

http://robolearn.org/

https://github.com/iproduct

https://twitter.com/trayaniliev

https://www.facebook.com/IPT.EACAD

https://plus.google.com/+IproductOrg