Feb 2024, IPT Course
Introduction to Spring 5

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 and Java
- Use Spring Boot 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 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 – building HelloSpringMVC & HelloSpringWebFlux simple web applications using *spring-boot-starter-web*

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
- Integrates carefully selected Jakarta EE specifications
- Spring 6 baseline: Java 17 & Jakarta EE 9

Spring Supported Jakarta/Java EE Specs

- Servlet API
- WebSocket API
- Concurrency Utilities
- JSON Binding API
- Bean Validation
- JPA
- JMS
- JTA/JCA transaction coordination
- Dependency Injection
- Common Annotations

Which Problems Spring Addresses?

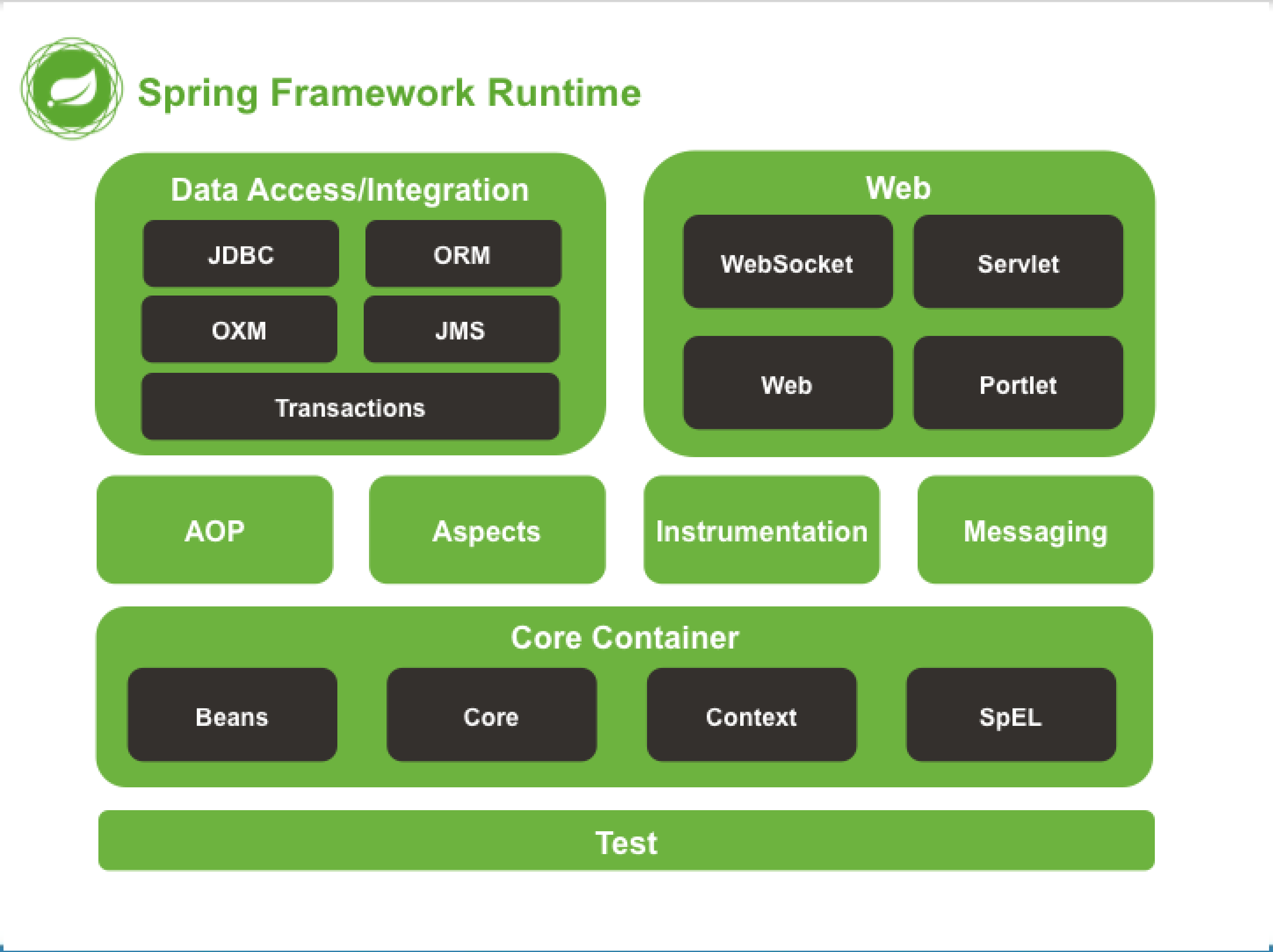
- Scalability and modularity
- Boiler plate code – using templates (JDBCTemplate, HibernateTemplate) and aspects (advises)
- Handling non-functional requirements – transactions, load scaling, security, logging, testability, maintainability, etc.
- Unit testing and integration testing
- Complex frameworks/application 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.

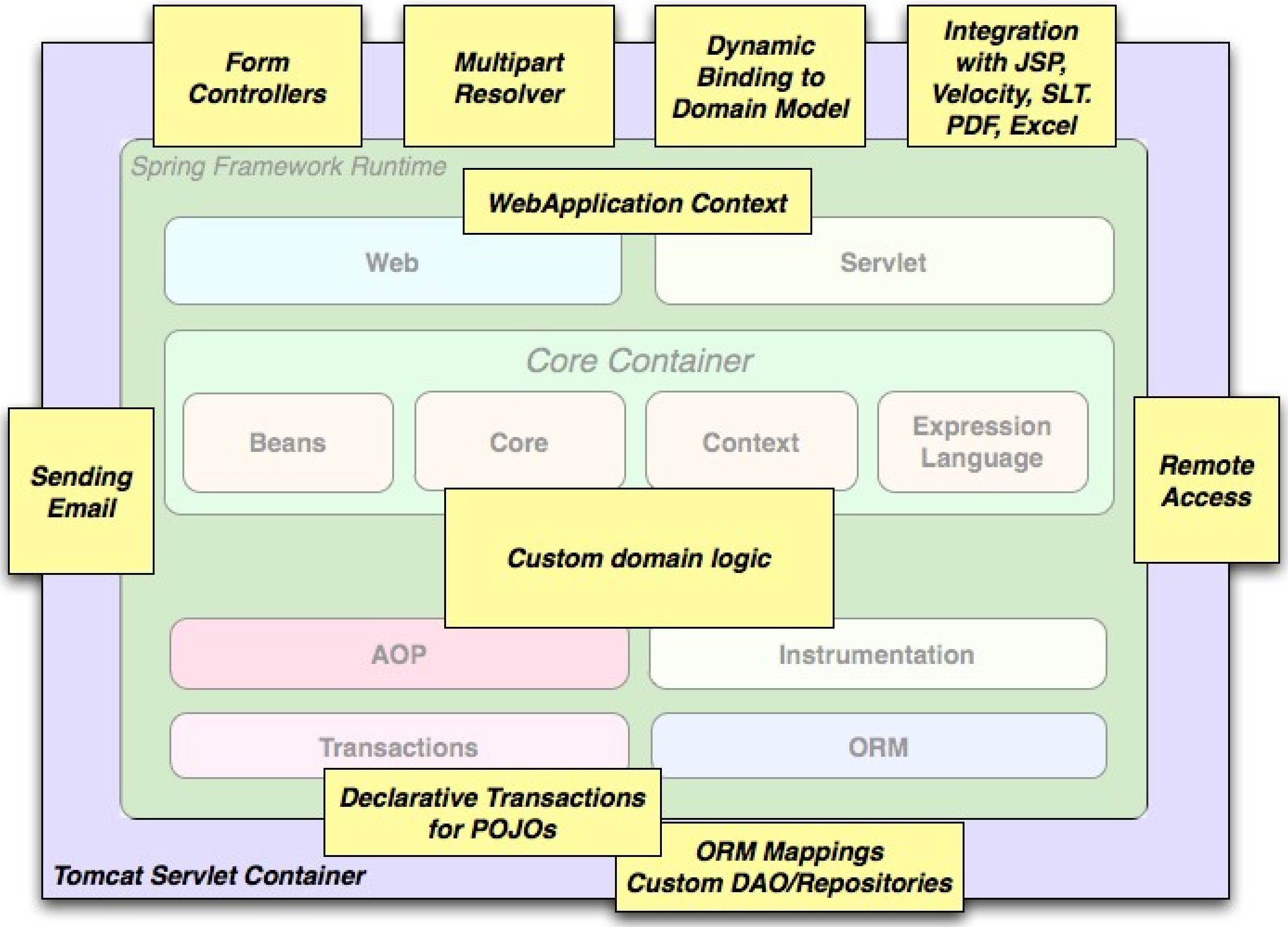
Source: https://projects.spring.io/spring-framework/ 9

Spring Framework 4 Main Modules



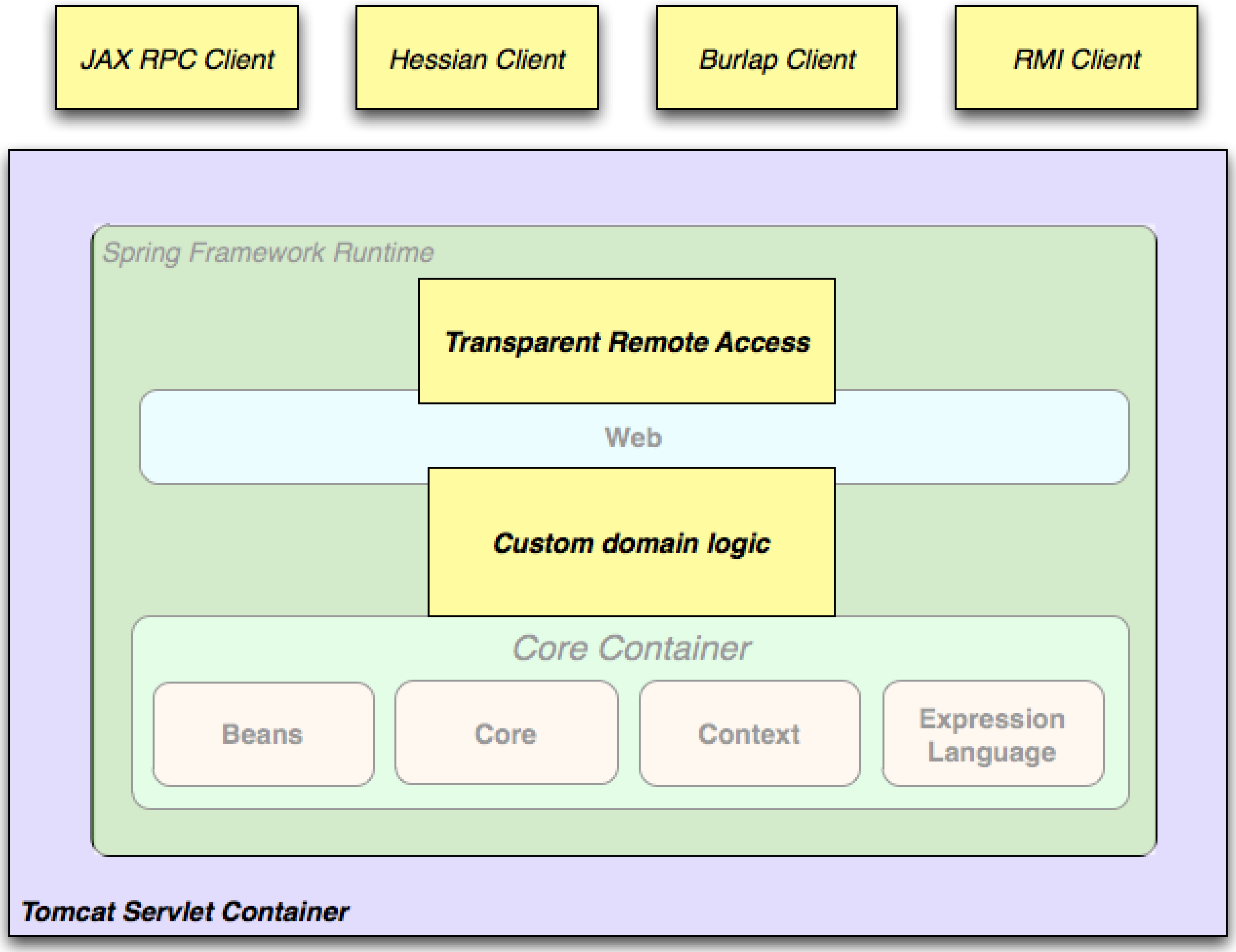
Source: https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/overview.html

Fully Fledged Spring Web Application



Source: https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/overview.html

Remoting Application



Spring Framework Modules

Spring Boot

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

Source: https://docs.spring.io

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

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
- Testing improvements – WebTestClient (based on reactive WebFlux WebClient)
- Kotlin functional DSL

Spring 6.x

- JDK 17+ & Jakarta EE 9+ Baseline,
- Entire framework codebase based on Java 17 source code level now.
- Migration from javax to jakarta namespace for Servlet, JPA, etc.
- Runtime compatibility with Jakarta EE 9 as well as Jakarta EE 10 APIs.
- Compatible with latest web servers: Tomcat 10.1, Jetty 11, Undertow 2.3.
- Early compatibility with virtual threads (in preview as of JDK 19).
- General Core Revision
- Upgrade to ASM 9.4 and Kotlin 1.7.
- Complete CGLIB fork with support for capturing CGLIB-generated classes.
- Comprehensive foundation for Ahead-Of-Time transformations.
- First-class support for GraalVM native images (Spring Boot 3).

Spring 6.x

Data Access and Transactions:

- Support for predetermining JPA managed types (for inclusion in AOT processing).
- JPA support for Hibernate ORM 6.1 (retaining compatibility with Hibernate ORM 5.6).
- Upgrade to R2DBC 1.0 (including R2DBC transaction definitions).
- Aligned data access exception translation between JDBC, R2DBC, JPA and Hibernate.
- Removal of JCA CCI support.

Spring 6.x

Spring Messaging

- RSocket interface client based on @RSocketExchange service interfaces.
- Early support for Reactor Netty 2 based on Netty 5 alpha.
- Support for Jakarta WebSocket 2.1 and its standard WebSocket protocol upgrade mechanism.

Spring 6.x

Web MVC

- HTTP interface client based on @HttpExchange service interfaces.
- Support for RFC 7807 problem details.
- Unified HTTP status code handling.
- Support for Jackson 2.14.
- Alignment with Servlet 6.0 (while retaining runtime compatibility with Servlet 5.0).
- Spring MVC
- PathPatternParser used by default (with the ability to opt into PathMatcher).
- Removal of outdated Tiles and FreeMarker JSP support.

Spring 6.x

Spring WebFlux:

- New PartEvent API to stream multipart form uploads (both on client and server).
- New ResponseEntityExceptionHandler to customize WebFlux exceptions and render RFC 7807 error responses.
- Flux return values for non-streaming media types (no longer collected to List before written).
- Early support for Reactor Netty 2 based on Netty 5 alpha.
- JDK HttpClient integrated with WebClient.
-

Spring 6.x

Observability:

- Direct Observability instrumentation with Micrometer Observation in several parts of the Spring Framework. The spring-web module now requires io.micrometer:micrometer-observation:1.10+ as a compile dependency.
- RestTemplate and WebClient are instrumented to produce HTTP client request observations.
- Spring MVC can be instrumented for HTTP server observations using the new org.springframework.web.filter.ServerHttpObservationFilter.
- Spring WebFlux can be instrumented for HTTP server observations using the new org.springframework.web.filter.reactive.ServerHttpObservationFilter.
- Integration with Micrometer Context Propagation for Flux and Mono return values from controller methods.

Spring 6.x

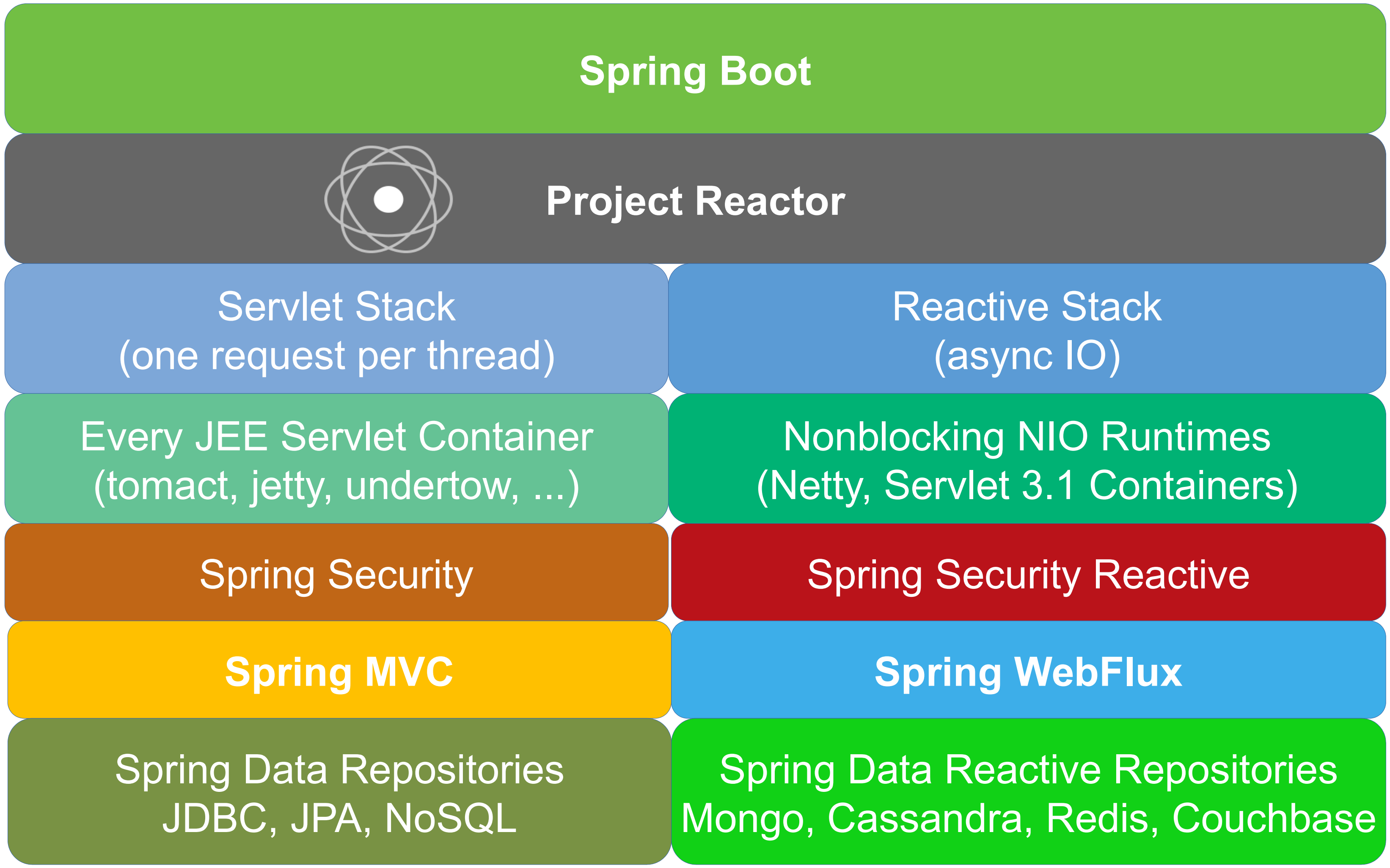
Testing:

- Support for testing AOT-processed application contexts on the JVM or within a GraalVM native image.
- Integration with HtmlUnit 2.64+ request parameter handling.
- Servlet mocks (MockHttpServletRequest, MockHttpSession) are based on Servlet API 6.0 now.
- New MockHttpServletRequestBuilder.setRemoteAddress() method.
- The four abstract base test classes for JUnit 4 and TestNG no longer declare listeners via @TestExecutionListeners and instead now rely on registration of default listeners.

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.

Spring Web Application Building Blocks


Spring Boot
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

Source: https://spring.io

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 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)

```
<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)

```
    <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
```

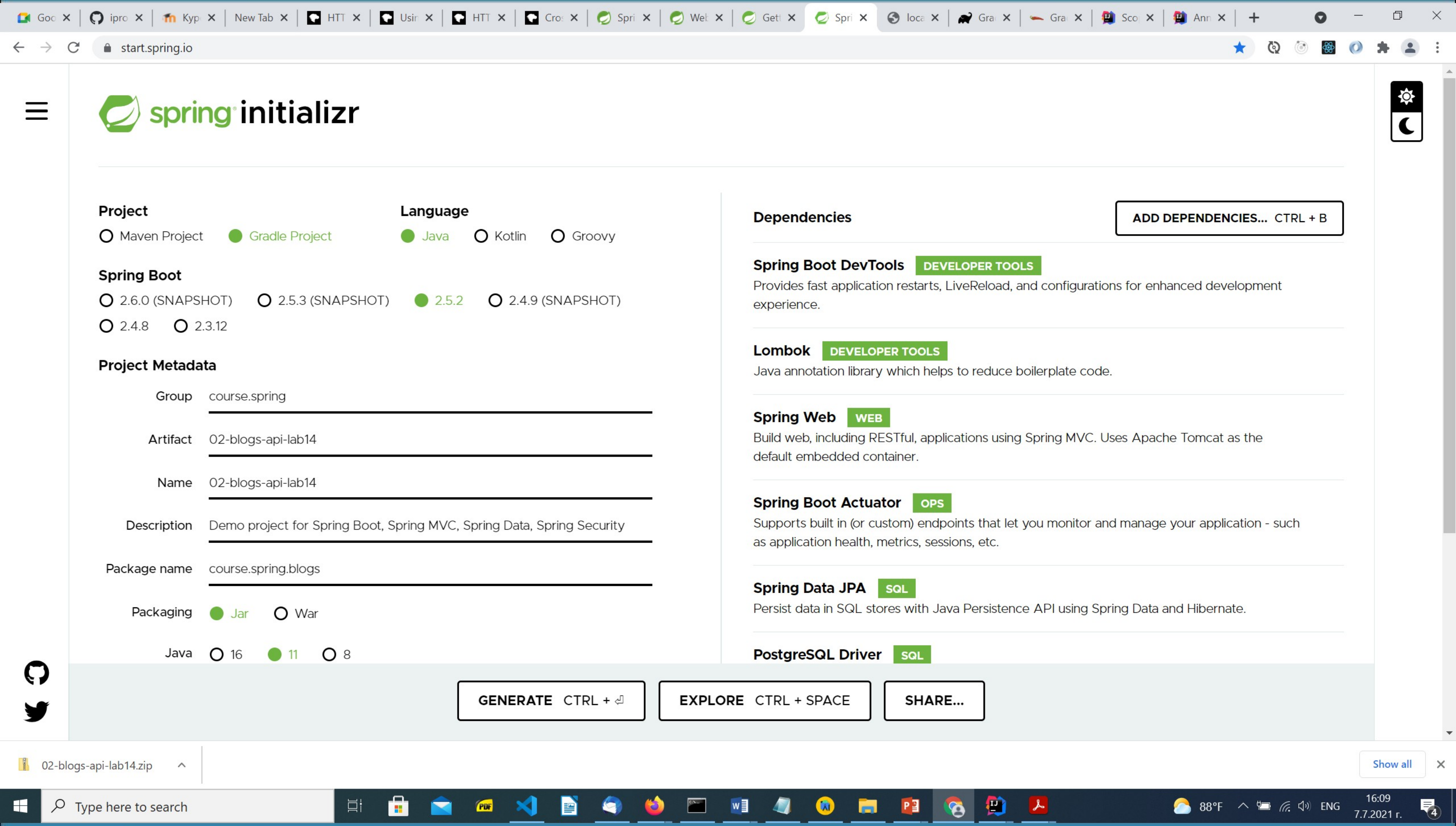
Gradle Configuration (continued)

```
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



start.spring.io

spring initializr

Project
Maven Project [✓ Gradle Project]

Language
[✓ Java] Kotlin Groovy

Spring Boot
2.6.0 (SNAPSHOT) 2.5.3 (SNAPSHOT) [✓ 2.5.2] 2.4.9 (SNAPSHOT) 2.4.8 2.3.12

Project Metadata

Group	course.spring
Artifact	02-blogs-api-lab14
Name	02-blogs-api-lab14
Description	Demo project for Spring Boot, Spring MVC, Spring Data, Spring Security
Package name	course.spring.blogs
Packaging	[✓ Jar] War
Java	16 [✓ 11] 8

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

PostgreSQL Driver SQL

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