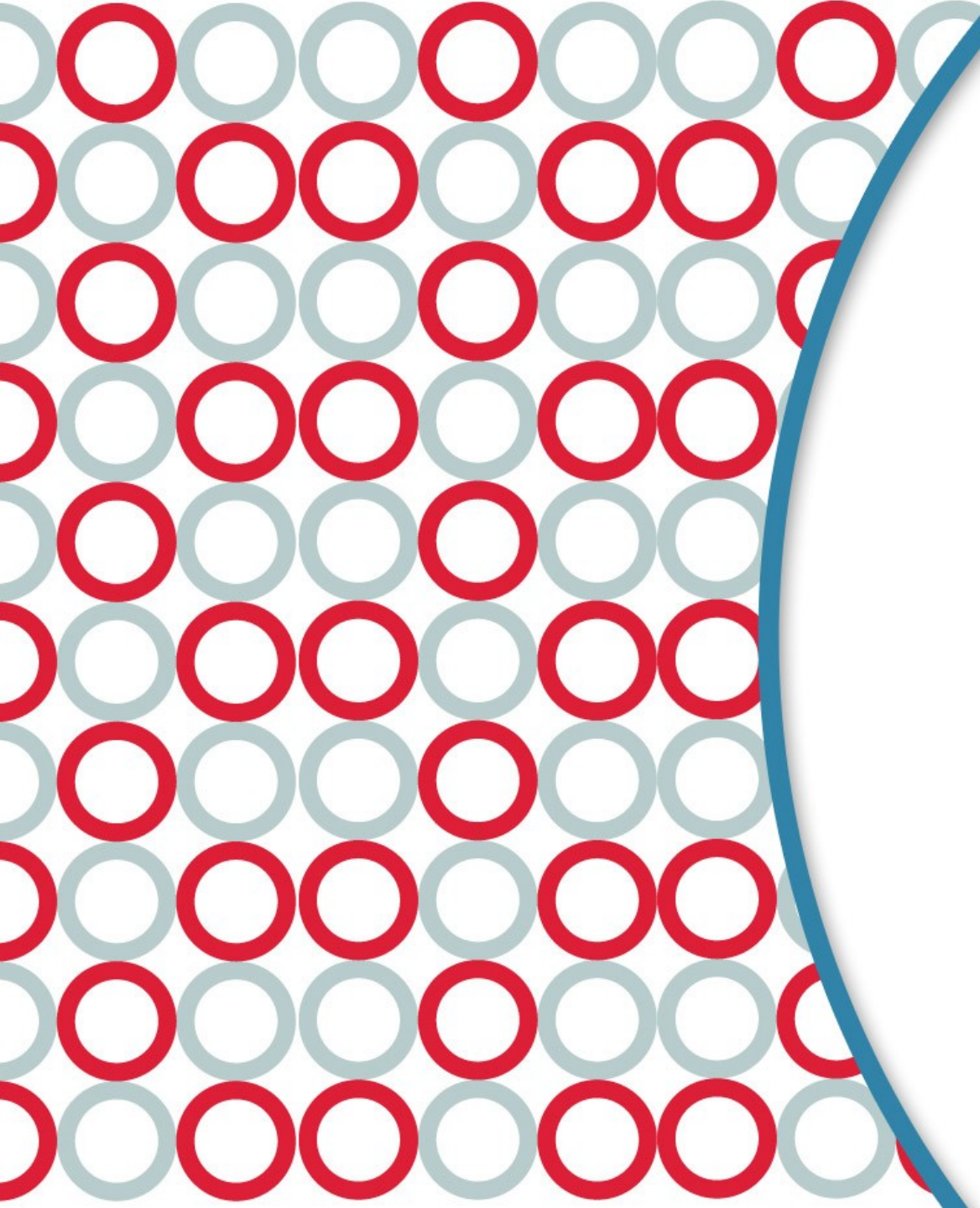


May 2019, IPT Course
Java Web Debelopment

Build Tools Basics

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)

Where to Find the Code?

Java Web Development projects and examples are available @ GitHub:

https://github.com/iproduct/course-java-web-development

Agenda for This Session

- Ant
- Maven
- Gradle
- Practical examples

Apache Ant

- Apache Ant is a software tool for automating software build processes, which originated from the Apache Tomcat project in early 2000.
- Replacement for the Make build tool of Unix, created due to a number of problems with Unix's make.
- Similar to Make but is implemented using the Java language, requires the Java platform, and is best suited to building Java projects.
- With Make, actions required to create a target are specified as shell commands which are specific to the platform on which runs.
- Ant solves this problem by providing a large amount of built-in functionality designed to behave similarly on all platforms.

Apache Ant - I

```
<?xml version="1.0"?>
<project name="Invoicing with Views" default="compile">
<presetdef name="javac"><javac includeantruntime="false" /></presetdef>
<property name="build.dir" location="${basedir}/bin"/>
    <target name="clean" description="remove intermediate files">
        <delete dir="${build.dir}"/>
    </target>
    <target name="clobber" depends="clean" description="remove all artifact files">
        <delete file="invoicing.jar"/>
    </target>
    <target name="compile" description="compile the Java source code to class files">
        <mkdir dir="${build.dir}"/>
        <javac srcdir="." destdir="${build.dir}"/>
    </target>
```

Apache Ant - II

```
...
    <target name="jar" depends="compile" description="create a Jar file for the app">
        <jar destfile="invoicing.jar">
            <fileset dir="${build.dir}" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="invoicing.view.MainMenu"/>
            </manifest>
        </jar>
    </target>
    <target name="run" depends="compile" description="run the application">
        <java classname="invoicing.view.MainMenu"
                classpath="${java.class.path};${build.dir}" dir="." fork="true" />
    </target>
    <target name="runJar" depends="jar" description="run the app from the jar file">
        <java jar="invoicing.jar" dir="." failonerror="true" fork="true" />
    </target>
</project>
```

Apache Ant Sample Tasks

```
<java classname="test.Main">
  <arg value="-h"/>
  <classpath>
    <pathelement location="dist/test.jar"/>
    <pathelement path="${java.class.path}"/>
  </classpath>
</java>

<java dir="${exec.dir}"
  jar="${exec.dir}/dist/test.jar"
      fork="true"
      failonerror="true"
      maxmemory="128m">
  <arg value="-h"/>
  <classpath>
    <pathelement location="dist/test.jar"/>
    <pathelement path="${java.class.path}"/>
  </classpath>
</java>
```

```
<junit printsummary="yes" haltonfailure="yes">
    <classpath>
        <pathelement location="${build.tests}"/>
        <pathelement path="${java.class.path}"/>
    </classpath>
    <formatter type="plain"/>
    <test name="my.test.TestCase"
        haltonfailure="no" outfile="result">
        <formatter type="xml"/>
    </test>
    <batchtest fork="yes" todir="${reports.tests}">
        <fileset dir="${src.tests}">
            <include name="**/*Test*.java"/>
            <exclude name="**/AllTests.java"/>
        </fileset>
    </batchtest>
</junit>
```

What is Maven?

What is Maven Apache Maven is a software project management and comprehension tool. Based on the concept of a **Project Object Model (POM)**, Maven can manage a project's build, reporting and documentation from a central piece of information.

Advantages over Ant:

- Eliminate the hassle of maintaining complicated scripts
- All the functionality required to build your project , i.e. , clean, compile, copy resources, install, deploy etc is built right into the Maven
- Cross Project Reuse- Ant has no convenient way to reuse target across projects, But Maven does provide this functionality

Maven Main Phases and Commands

Although hardly a comprehensive list, these are the most common default lifecycle phases executed.

- **validate**: validate the project is correct and all necessary information is available
- **compile**: compile the source code of the project
- **test**: test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
- **package**: take the compiled code and package it in its distributable format, such as a JAR.
- **integration-test**: process and deploy the package if necessary into an environment where integration tests can be run
- **verify**: run any checks to verify the package is valid and meets quality criteria
- **install**: install the package into the local repository, for use as a dependency in other projects locally
- **deploy**: done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects.

There are two other Maven lifecycles of note beyond the default list above. They are

- **clean**: cleans up artifacts created by prior builds
- **site**: generates site documentation for this project

Maven Dependency Management

- Apache Maven – https://spring.io/guides/gs/maven/
- Common arguments: **`mvn compile`, `mvn package`, `mvn clean dependency:copy-dependencies 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)

```
<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'
apply plugin: 'java'
apply plugin: 'application'
mainClassName =
     'org.iproduct.spring.demo.xmlconfig.HelloWorldSpringDI'

sourceCompatibility = 1.8
```

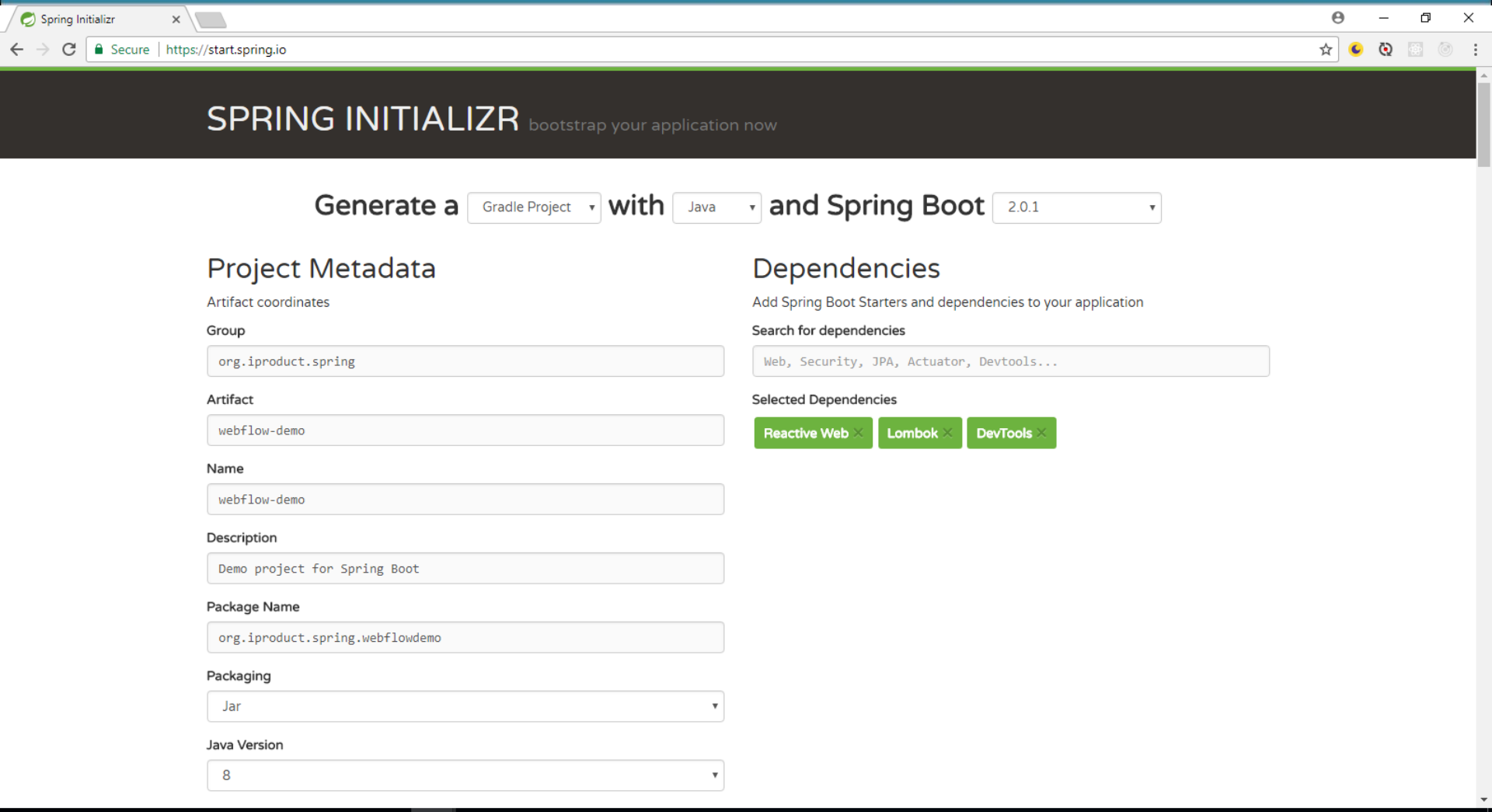
Gradle Configuration (continued)

```
task runApp(type: JavaExec) {
    classpath = sourceSets.main.runtimeClasspath
    main =
'org.iproduct.spring.demo.xmlconfig.HelloWorldSpringDI'
}

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

dependencies {
    compile group: 'org.springframework',
            name: 'spring-context', version: '5.0.5.RELEASE'
    testCompile group: 'junit', name: 'junit', version: '4.12'
}
```

Making Projects Easy: Spring Boot 2



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