

# Spring Boot - Basics

Getting started with Spring Boot

Starter POMs, Auto-Configuration

# What is Spring Boot?

- Spring applications typically require a lot of setup
  - Consider working with JPA. You need:
    - Datasource, TransactionManager, EntityManagerFactory ...
  - Consider a web MVC app. You need:
    - WebApplicationInitializer / web.xml, ContextLoaderListener, DispatcherServlet, ...
  - An MVC app using JPA would need all of this
- *BUT: Much of this is predictable*
  - Spring Boot can do most of this setup for you

# What is Spring Boot?

- An opinionated runtime for Spring Projects
- Supports different project types, like Web and Batch
- Handles most low-level, predictable setup for you
  
- It is not:
  - A code generator
  - An IDE plugin



See: [Spring Boot Reference](http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle)

<http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle>

# Topics in this Session

- **What is Spring Boot?**
  - **Definition and Hello World example**
- Dependency Management
- Ease of Use Features

# Opinionated Runtime?

- Spring Boot uses sensible defaults, “*opinions*”, mostly based on the classpath contents.
- For example
  - Sets up a JPA Entity Manager Factory if a JPA implementation is on the classpath.
  - Creates a default Spring MVC setup, if Spring MVC is on the classpath.
- Everything can be overridden easily
  - But most of the time not needed

# Hello World example

- Only 3 files to get a running Spring application

pom.xml

*Setup Spring Boot dependencies*

HelloController

*Basic Spring MVC controller*

Application class

*Application launcher*

# Hello World – Maven descriptor

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.3.0.RELEASE</version>
</parent>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

*pom.xml*

parent

Spring MVC  
Embedded Tomcat  
Jackson...



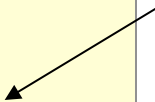
Maven is just one option. You can also use Gradle or Ant/Ivy

# Hello World – Spring MVC controller

- A RESTful controller to keep this example simple
  - Returns a String as the body of the HTTP Response
  - No view involved

```
@RestController
public class HelloController {
    @RequestMapping("/")
    public String hello() {
        return "Greetings from Spring Boot!";
    }
}
```

No separate View file to keep things simple



*Controller.java*



# Hello World – Application Class

- `@SpringBootApplication` annotation enables Spring Boot

```
@SpringBootApplication
public class Application {

    public static void main(String[] args) {
        SpringApplication.run(Application.class, args);
    }

}
```

*application.java*



Main method will be used to run the packaged application from the command line – *old style!*

# Deployment

- Our “Hello World” example bundles Tomcat inside the application
  - Runs as an executable *JAR*
- Spring Boot apps can also be deployed into an existing app server
  - As a familiar *WAR* file

# Putting it all together

```
mvn package
```

Maven command to generate an archive file

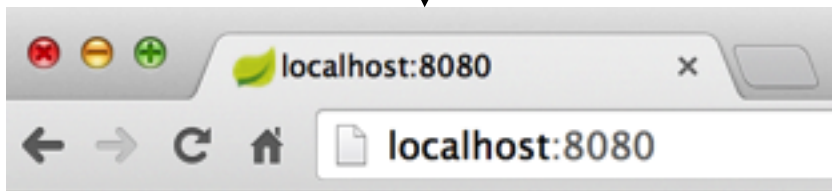
```
helloApp-0.0.1-SNAPSHOT.jar
```

*generated file*

Generated file

```
java -jar helloApp-0.0.1-SNAPSHOT.jar
```

App started in command line



App runs on port 8080

# Topics in this Session

- What is Spring Boot?
  - Definition and Hello World example
- **Dependency Management**
- Ease of Use Features

# How to use Spring Boot?

- Add the appropriate Spring Boot dependencies
- The easiest is to use a dependency management tool
- Spring Boot works with Maven, Gradle, Ant/Ivy
- Our content here will show Maven

# Spring Boot Parent POM

- Parent POM defines key versions of dependencies and Maven plugins

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.3.0.RELEASE</version>
</parent>
```

Defines properties for dependencies, for example: `${spring.version} = 4.2`

# Spring Web Dependencies

- Everything you need to develop a web application with Spring

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
</dependencies>
```

Resolves

- spring-web-\*.jar*
- spring-webmvc-\*.jar*
- tomcat-\*.jar*
- jackson-databind-\*.jar*
- ...

# Topics in this Session

- What is Spring Boot?
  - Definition and Hello World example
- Dependency Management
- **Ease of Use Features**



# Externalized Properties

## *application.properties*

- Developers commonly externalize properties to files
  - Easily consumable via Spring PropertySource
  - But developers name / locate their files different ways
- Spring Boot automatically looks for **application.properties** in the classpath root

```
database.host=localhost  
database.user=admin
```

*application.properties*

- Starter POMs declare the properties to use
  - Check the reference documentation to know which properties can be used

# Controlling Logging Level

- Boot can control the logging level
  - Just set it in `application.properties`
- Works with most logging frameworks
  - Java Util Logging, Logback, Log4J, Log4J2

```
logging.level.org.springframework=DEBUG  
logging.level.com.acme.your.code=INFO
```

*application.properties*



Try to stick to SLF4J in the application.

The *advanced* section covers how to change the logging framework

# DataSource Configuration

- Use either `spring-boot-starter-jdbc` or `spring-boot-starter-data-jpa` and include a JDBC driver on classpath
- Declare properties

*application.properties*

```
spring.datasource.url=jdbc:mysql://localhost/test
spring.datasource.username=dbuser
spring.datasource.password=dbpass
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

- That's It!
  - Spring Boot will create a DataSource with properties set
  - Will even use a connection pool if the library is found on the classpath!

# Web Application Convenience

- Boot automatically configures Spring MVC DispatcherServlet and `@EnableWebMvc` defaults
  - When *spring-webmvc\*.jar* on classpath
- Static resources served from classpath
  - `/static`, `/public`, `/resources` or `/META-INF/resources`
- Templates served from `/templates`
  - When Velocity, Freemarker, Thymeleaf, or Groovy on classpath
- Provides default `/error` mapping
  - Easily overridden

# Summary

- Spring Boot speeds up Spring application development
- You always have full control and insight
- Nothing is generated
- No special runtime requirements
- Stay tuned for even more features in future releases