Pivotal

Spring Boot Introduction

Introduction to Spring Boot

Objectives

After completing this lesson, you should be able to

- Explain what Spring Boot is and why it is opinionated
- Explain the basic features in Spring Boot

Agenda

- **What is Spring Boot?**
- Spring Boot Features
  - Dependency management
  - Auto-Configuration
  - Packaging and Runtime
  - Integration Testing
- Getting Started with Spring Boot
- Summary

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 plug-in

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

Opinionated Runtime?

- Uses sensible defaults, *“opinions”*, mostly based on the classpath contents
  - Sets up a DataSource if an embedded DB implementation is on the classpath
  - Creates a **`JdbcTemplate`** if **`spring-jdbc.jar`** is on the classpath AND **`DataSource`** bean defined
- Everything can be overridden easily but often not necessary

Agenda

- What is Spring Boot?
- Spring Boot Features
  - **Dependency management**
  - Auto-Configuration
  - Packaging and Runtime
  - Integration Testing
- Getting Started
- Summary

Spring Applications Need Dependencies

- Spring Boot applications require dependencies:
  - Spring Boot JARs, Spring JARs, common 3rd party JARs, etc.
- Spring Boot recommends use of a modern dependency management tool
  - … but does not *require* one
  - Maven, Gradle, Ivy supported
  - Our content here will only show Maven

Spring Boot Parent POM

- Defines versions of key dependencies
  - Uses a **`dependencyManagement`** section internally

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.2.0.RELEASE</version>
</parent>
```

Defines properties for dependencies, for example:
`${spring.version} = 5.2.0.RELEASE`

- Defines Maven plugins
- Sets up Java version

Spring Boot *"Starter"* Dependencies

- Easy way to bring in multiple coordinated dependencies
  - Including *"Transitive"* Dependencies

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

Version not needed!
Defined by parent

Resolves ~ 16 JARs!

spring-boot-.jar*
spring-core-.jar*
spring-context-.jar*
spring-aop-.jar*
spring-beans-.jar*
aopalliance-.jar*
...

Test *"Starter"* Dependencies

- Common test libraries

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

Resolves

spring-test-.jar*
junit-.jar*
mockito-.jar*
…

Available Starter POMs

- Not essential but *strongly* recommended for getting started
- Coordinated dependencies for common Java enterprise frameworks
  - Pick the starters you need in your project
- To name a few:
  - **`spring-boot-starter-jdbc`**
  - **`spring-boot-starter-data-jpa`**
  - **`spring-boot-starter-web`**
  - **`spring-boot-starter-batch`**

See: Spring Boot Reference, Starter POMs
https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#using-boot-starter

Agenda

- What is Spring Boot?
- Spring Boot Features
  - Dependency management
  - **Auto-Configuration**
  - Packaging and Runtime
  - Integration Testing
- Getting Started with Spring Boot
- Summary

Spring Boot @EnableAutoConfiguration

- `@EnableAutoConfiguration` annotation on a Spring Java configuration class
  - Spring Boot automatically creates the beans it thinks you need

```
@Configuration
@ComponentScan
@EnableAutoConfiguration
public class Application {
    public static void main(String[] args) {
            SpringApplication.run(Application.class, args);
    }
}
```

No packages specified = scan current package and all sub-packages

`SpringApplication` is actually a Spring Boot class

Shortcut: @SpringBootApplication

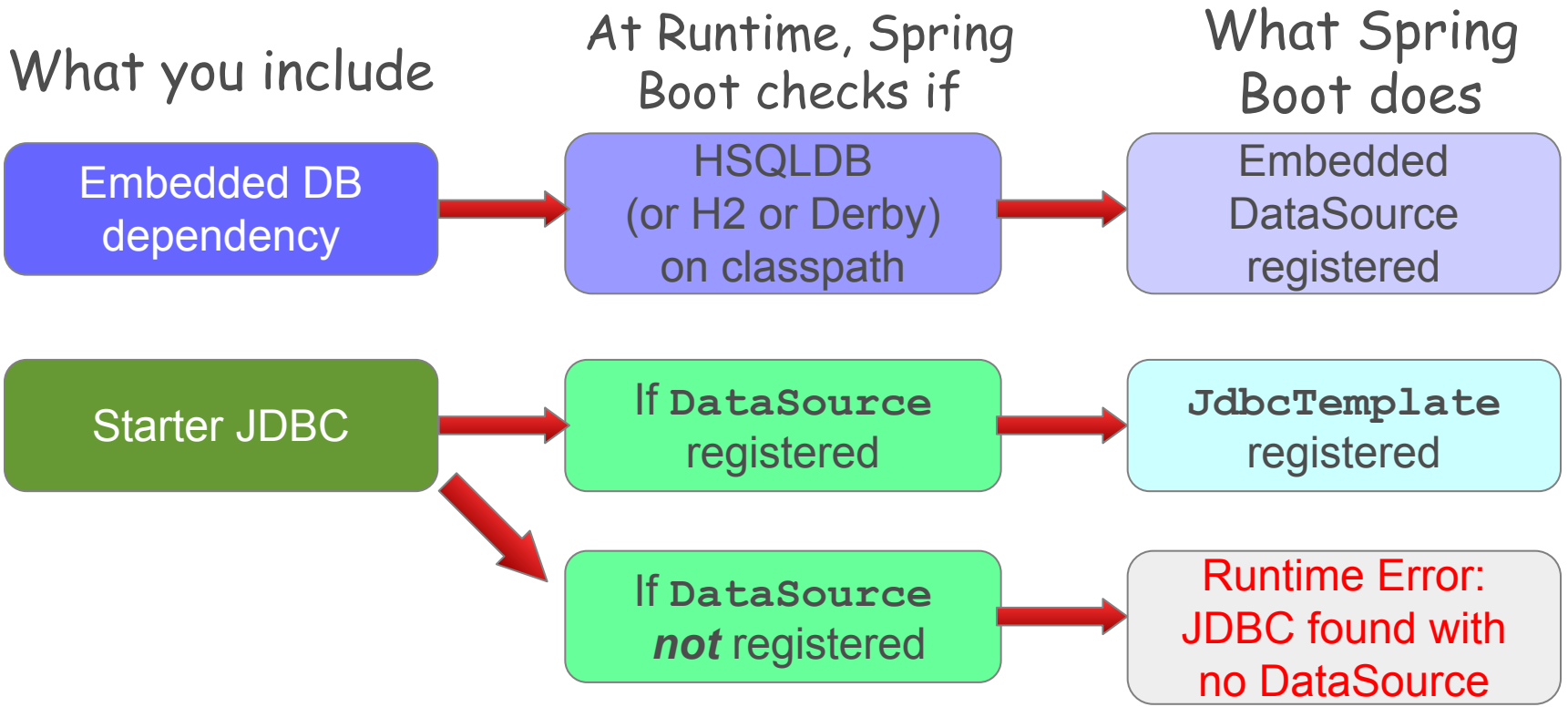
- Very common to use @EnableAutoConfiguration, @SpringBootConfiguration, and @ComponentScan together

```
@SpringBootConfiguration
@ComponentScan("example.config")
@EnableAutoConfiguration
 public class Application {
     ...
 }
```

```
@SpringBootApplication
(scanBasePackages="example.config")
public class Application {
   ...
}
```

@SpringBootConfiguration simply extends **@Configuration** – see **@SpringBootTest** for why.

Auto-Configuration: Simplified Examples



Agenda

- What is Spring Boot?
- Spring Boot Features
  - Dependency management
  - Auto-Configuration
  - **Packaging and Runtime**
  - Integration Testing
- Getting Started with Spring Boot
- Summary

Fat JARs and the Spring Boot Plugin

- A “fat” JAR contains all its dependencies
  - Can be run directly using **`java -jar`** command

- To create
  - Add plugin to your Maven POM or Gradle Build file
  - Build JAR in usual way
    - **`gradle assemble`** or **`mvn package`**
  - Creates two JARs
    - **`my-app.jar`** the executable “fat” JAR
    - **`my-app.jar.original`** the “usual” JAR

Spring Boot Plugin - Maven

- What it does
  - Extend **`package`** goal to create fat JAR
  - Add **`spring-boot:run`** goal to run your application

```
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Packaging Result

- "**`mvn package`**" execution produces (in **`target`**)

```
22M  yourapp-0.0.1-SNAPSHOT.jar
 5K  yourapp-0.0.1-SNAPSHOT.jar.original
```

  - **`.jar.original`** contains only your code (a traditional JAR file)
  - **`.jar`** contains your code *and* all libs – executable
    - *Notice that it is much bigger*

Agenda

- What is Spring Boot?
- Spring Boot Features
  - Dependency management
  - Auto-Configuration
  - Packaging and Runtime
  - **Integration Testing**
- Getting Started with Spring Boot
- Summary

Test: @*SpringBootTest*

- Replaces @SpringJUnitConfig used previously

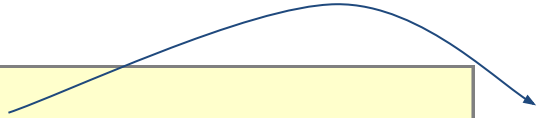
Loads the specified configuration applying same Spring Boot defaults

```
@SpringBootTest(classes=Application.class)
public class TransferServiceTests {
    @Autowired
    private TransferService transferService;

    @Test
    public void successfulTransfer() {
        TransferConfirmation conf = transferService. transfer(...);
        ...
    }

}
```

```
@SpringBootApplication(scanBasePackages="transfers")
public class Application {
        // Bean methods
}
```

Testing: *@SpringBootConfiguration*

- Spring Boot can find configuration class for itself
  - Provided it is in a package *above* the test
  - Only one `@SpringBootConfiguration` allowed in a hierarchy



```
@SpringBootTest  // classes not needed
public class TransferServiceTests {
    // Same tests as previous slide

}
```

```
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan("transfers")
public class Application {
    // Bean methods
}
```

Agenda

- What is Spring Boot?
- Spring Boot Features
  - Dependency management
  - Auto-Configuration
  - Packaging and Runtime
  - Integration Testing
- **Getting Started with Spring Boot**
- Summary

Hello World example

- Just three files to get a running Spring application

pom.xml

Setup Spring Boot (and any other) dependencies

application.properties

General configuration

Application class

Application launcher

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

Spring Initializr - What is it?

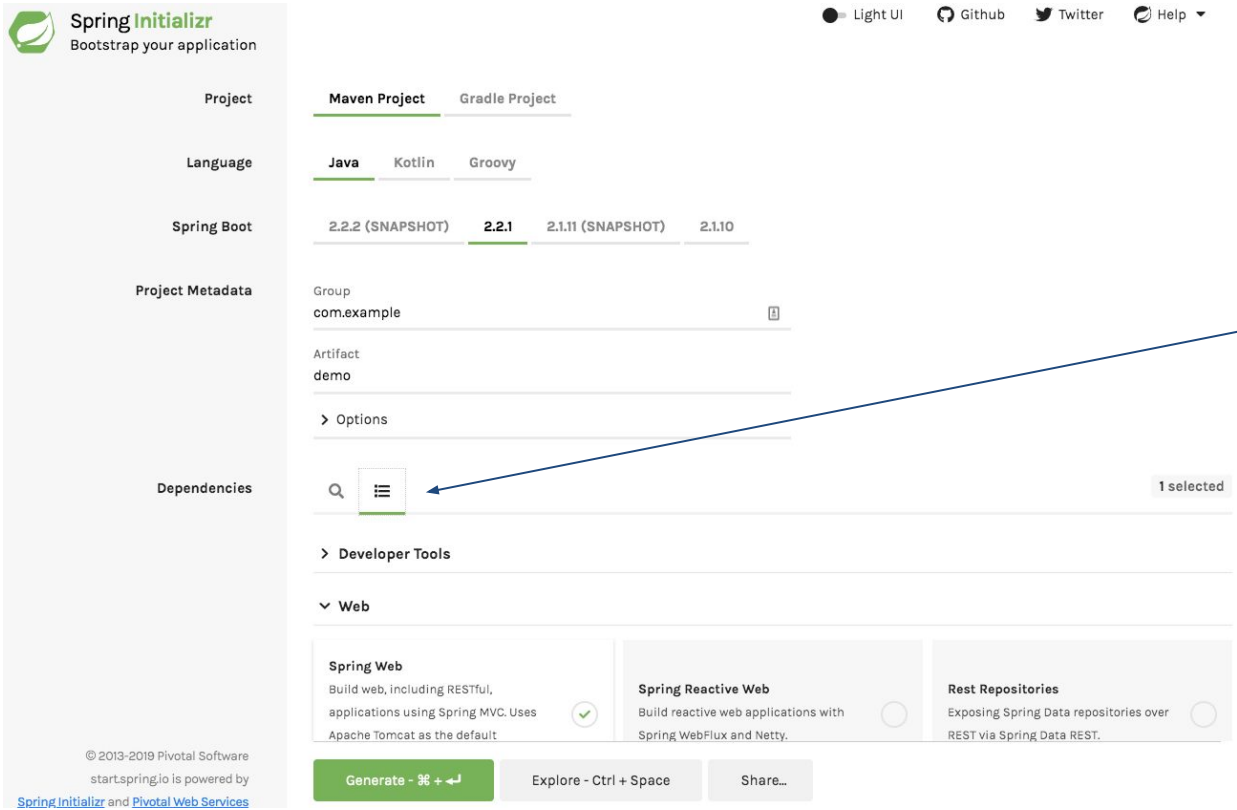
- Framework, API, and default implementation to generate initial Spring Boot application projects
- Spring's public web-site: http://start.spring.io
- Or build your own: https://github.com/spring-io/initializr

Spring Initializr - What is its value?

- Simplify and curate dependency management
  - Gradle or Maven supported
  - Java, Groovy or Kotlin
- Constructs starting template of Spring Boot projects
  - Mainly folder structure, Maven/Gradle files
- Accessible as a “New Project” wizard in STS/Eclipse, IntelliJ IDE (Ultimate version only)

Spring Initializr Web Page

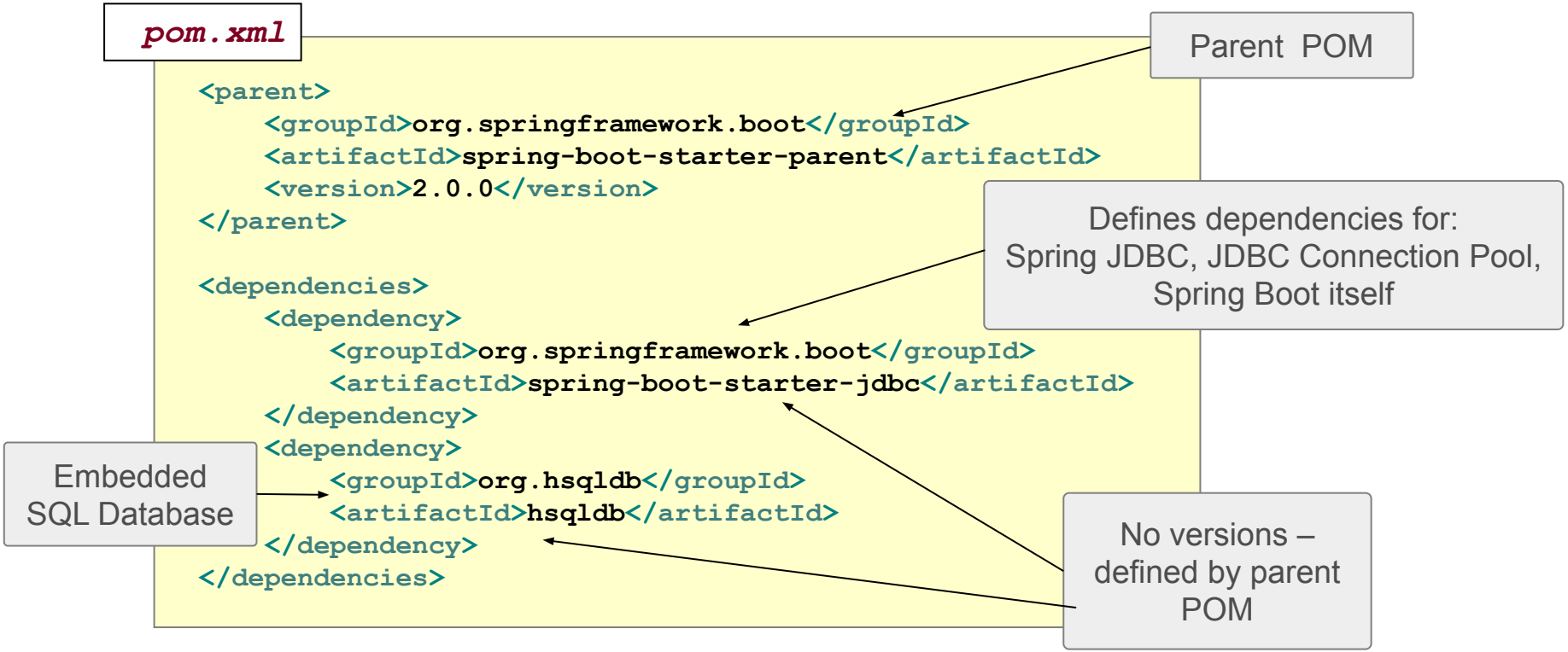
http://start.spring.io



Switch to full version: more options, explicit check-list of dependencies

Specify dependencies

Hello World (1a) - Maven descriptor



Hello World (1b) - Maven descriptor

- Will also use the Spring Boot plugin

pom.xml (continued)

```
<!-- Continued from previous slide -->
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
    </plugins>
</build>
```

Makes “fat” *executable* jars

Hello World (2) - application.properties

- Properties can be defined to supplement autoconfiguration or override autoconfiguration

application.properties

```
# Set the log level for all modules to 'ERROR'
logging.level.root=ERROR

# Tell Spring JDBC Embedded DB Factory where
# to obtain DDM and DML files
spring.datasource.schema=rewards/schema.sql
spring.datasource.data=rewards/data.sql
```

Hello World (3) - Application Class

```
@SpringBootApplication
public class Application {

    public static final String QUERY = "SELECT count(*) FROM T_ACCOUNT";

    public static void main(String[] args) {
        ApplicationContext ctx = SpringApplication.run(Application.class, args);
        JdbcTemplate jdbcTemplate = ctx.getBean(JdbcTemplate.class);

        System.out.println("Hello, there are "
            + jdbcTemplate.queryForObject(QUERY, Long.class)
            + " accounts");
    }
}
```

This annotation *turns on* Spring Boot

Automatically created by Boot

Application.java

Main method will be used to run the packaged application from the command line

Hello World (4) - Putting it all together

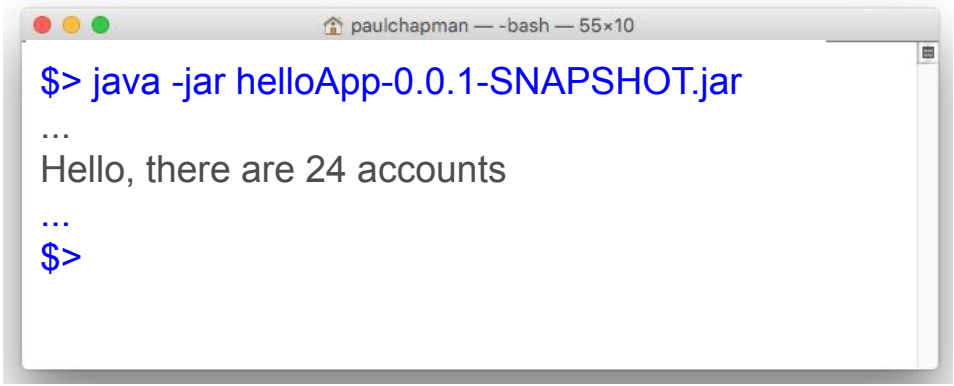
```
mvn package
```

↓

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

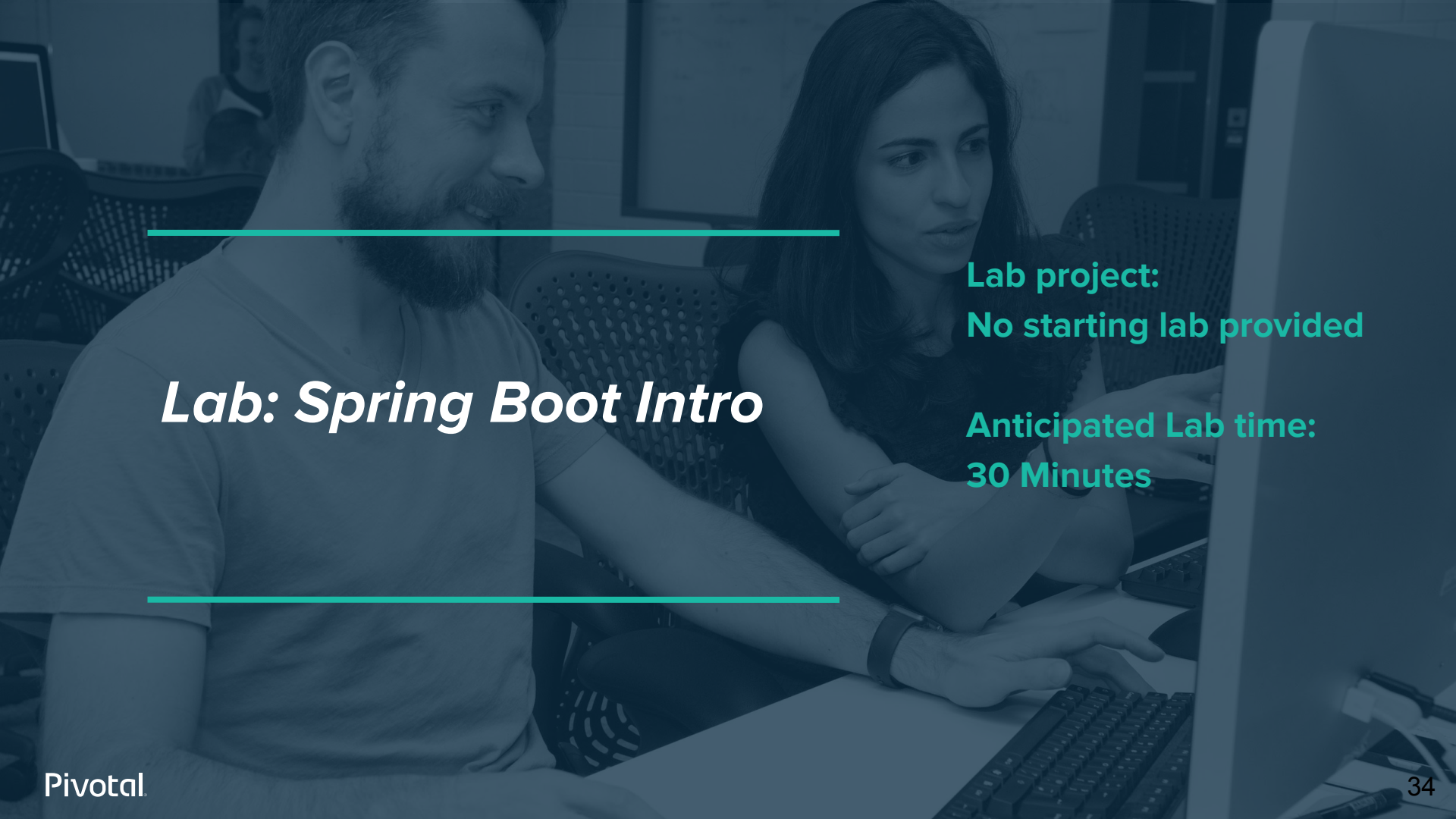
generated file

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



Summary

- Spring Boot significantly simplifies Spring setup
  - Will setup much of your application for you
  - Simplifies dependency management
  - Uses in-built defaults (opinions) to do the obvious setup
    - Automatically creates beans it thinks you need
  - Builds “fat” JARs
  - Use `@SpringBootTest` to enable Spring Boot in tests



Lab: Spring Boot Intro

Lab project:
No starting lab provided

Anticipated Lab time:
30 Minutes