Pivotal

Spring Boot Feature Introduction

Introduction to Spring Boot Features

Objectives

After completing this lesson, you should be able to

- Explain what Spring Boot is and how it simplifies application development
- Explain and use the Spring Boot features

Pivotal

Agenda

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

Pivotal

What is Spring Boot?

- An opinionated runtime for Spring Projects
- Supports different project types like Web and Batch
- Handles most low-level, predictable set-up 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)
http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle

Why Spring Boot?

- Provide a radically faster and widely accessible getting-started experience for all Spring development
- Be opinionated out of the box but get out of the way quickly as requirements start to diverge from the defaults
- Provide a range of non-functional features that are common to large classes of projects
  - Embedded servers, security, metrics, health checks, and externalized configuration, etc.

Agenda

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

Pivotal

How do you manage Dependencies?

- Modern Java application require a large number of dependencies - How do you make sure they are compatible?
  - Spring Boot JARs, Spring JARs, common 3rd party JARs, etc.

- Spring Boot’s parent or Starters to the rescue
  - Leverages existing dependency management schemes

- Use of a modern dependency management tool is recommended for fine-grained dependency management
  - … but one is not required
  - Maven, Gradle, Ivy supported

Spring Boot Parent POM

- Defines versions of key dependencies
  - Uses a **`dependencyManagement`** section internally
  - Through **`spring-boot-dependencies`** as a parent

```
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.2.2.RELEASE</version>
  </parent>
```

Defines properties for dependencies, for example:
`${spring.version} = 5.2.2.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*
…

Pivotal

Available Starters

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

Auto-configuration enabled by @EnableAutoConfiguration

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

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

`@SpringBootConfiguration` simply extends `@Configuration`

`SpringApplication` is actually a Spring Boot class

Pivotal

Shortcut: @SpringBootApplication

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

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

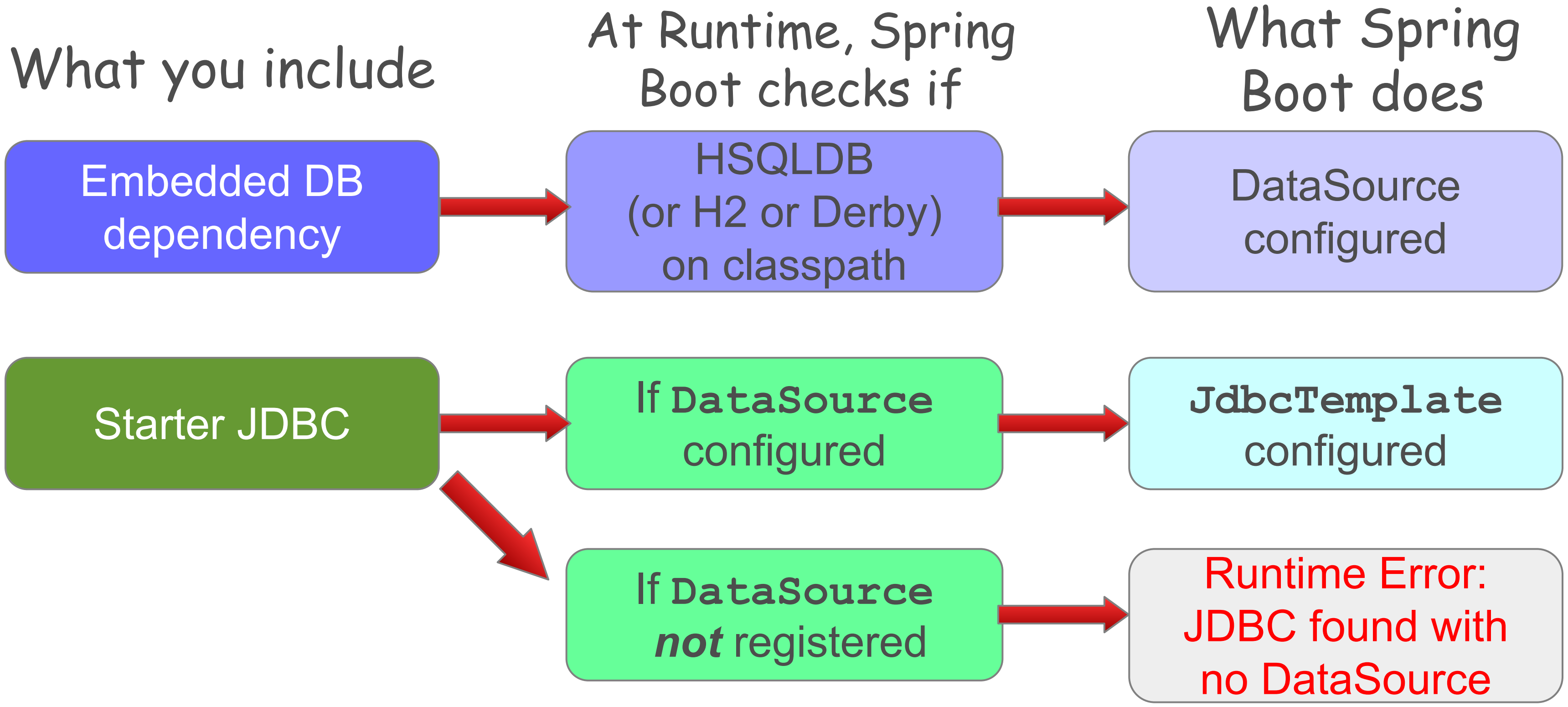
$\rightarrow$

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

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

Pivotal

Examples of Auto-configuration: DataSource, JdbcTemplate



What you include
At Runtime, Spring Boot checks if
What Spring Boot does
Embedded DB dependency
HSQLDB (or H2 or Derby) on classpath
DataSource configured
Starter JDBC
If `DataSource` configured
`JdbcTemplate` configured
If `DataSource` *not* registered
Runtime Error: JDBC found with no DataSource

Agenda

- What is and Why 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

Pivotal

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>
```

Pivotal

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*

Pivotal

Agenda

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

Pivotal

Test: *@SpringBootTest*

- Alternative to @SpringJUnitConfig

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

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

}
```

Loads the specified configuration applying same Spring Boot defaults

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

Pivotal

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

Pivotal

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?

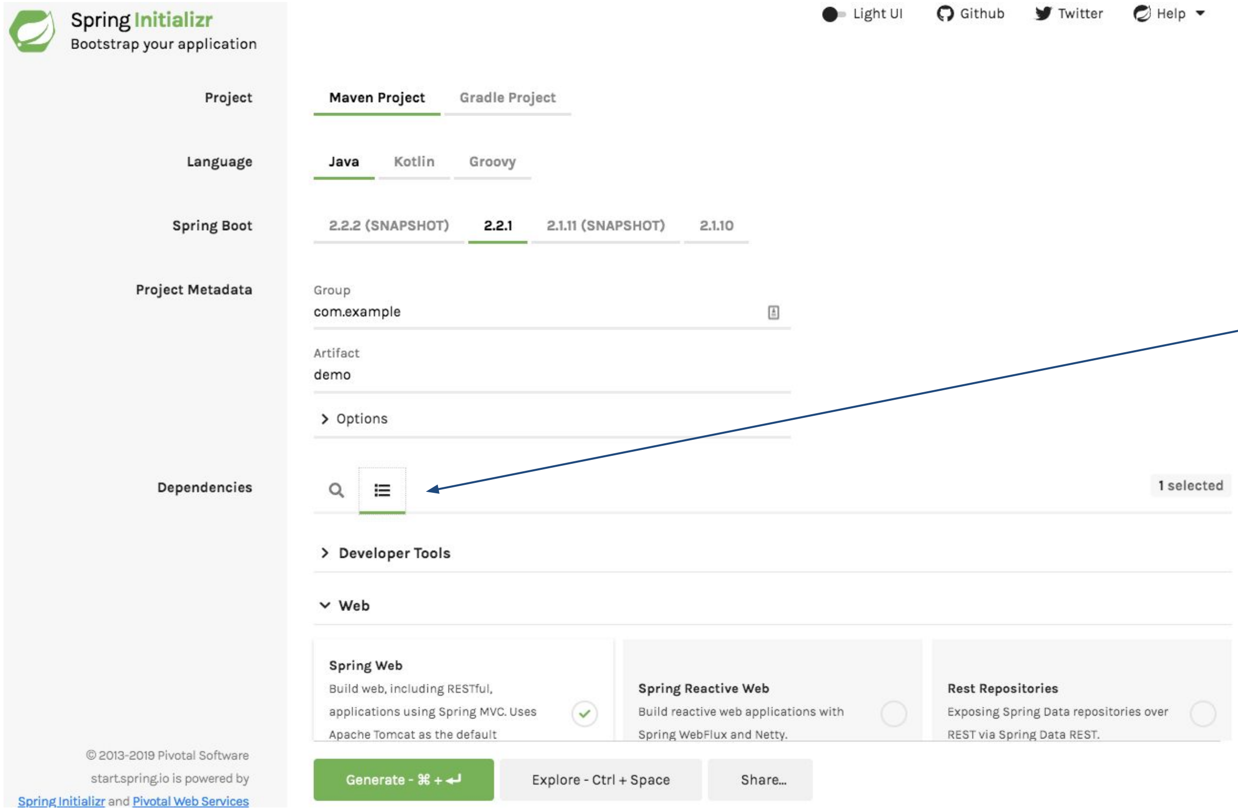
- Framework, API, and default implementation to generate initial Spring Boot application projects
- Spring's public web-site: http://start.spring.io
- Or build your own: https://github.com/spring-io/initializr

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, IntelliJ IDE (Ultimate version only)

Spring Initializr Web Page

http://start.spring.io



Project: [✓ Maven Project] Gradle Project

Language: [✓ Java] Kotlin Groovy

Spring Boot: 2.2.2 (SNAPSHOT) [✓ 2.2.1] 2.1.11 (SNAPSHOT) 2.1.10

Project Metadata

Group: com.example

Artifact: demo

> Options

Dependencies — 1 selected

> Developer Tools

Web

- Spring Web — Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default ✓
- Spring Reactive Web — Build reactive web applications with Spring WebFlux and Netty.
- Rest Repositories — Exposing Spring Data repositories over REST via Spring Data REST.

© 2013-2019 Pivotal Software
start.spring.io is powered by Spring Initializr and Pivotal Web Services

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

Specify dependencies

Pivotal

Hello World (1a) - Maven descriptor

pom.xml

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

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

Parent POM

Defines dependencies for:
Spring JDBC, JDBC Connection Pool,
Spring Boot itself

Embedded
SQL Database

No versions –
defined by parent
POM

Pivotal

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

Pivotal

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) {
            SpringApplication.run(JdbcBootApplication.class, args);
    }

    @Bean
    CommandLineRunner commandLineRunner(JdbcTemplate jdbcTemplate){

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

Application.java

This annotation *turns on* Spring Boot

Automatically created by Boot

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

Pivotal

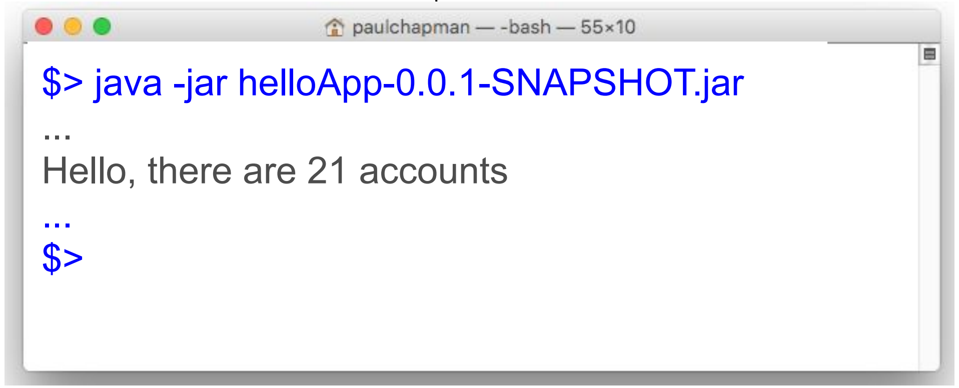
Hello World (4) - Putting it all together

```
mvn package
```

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

generated file

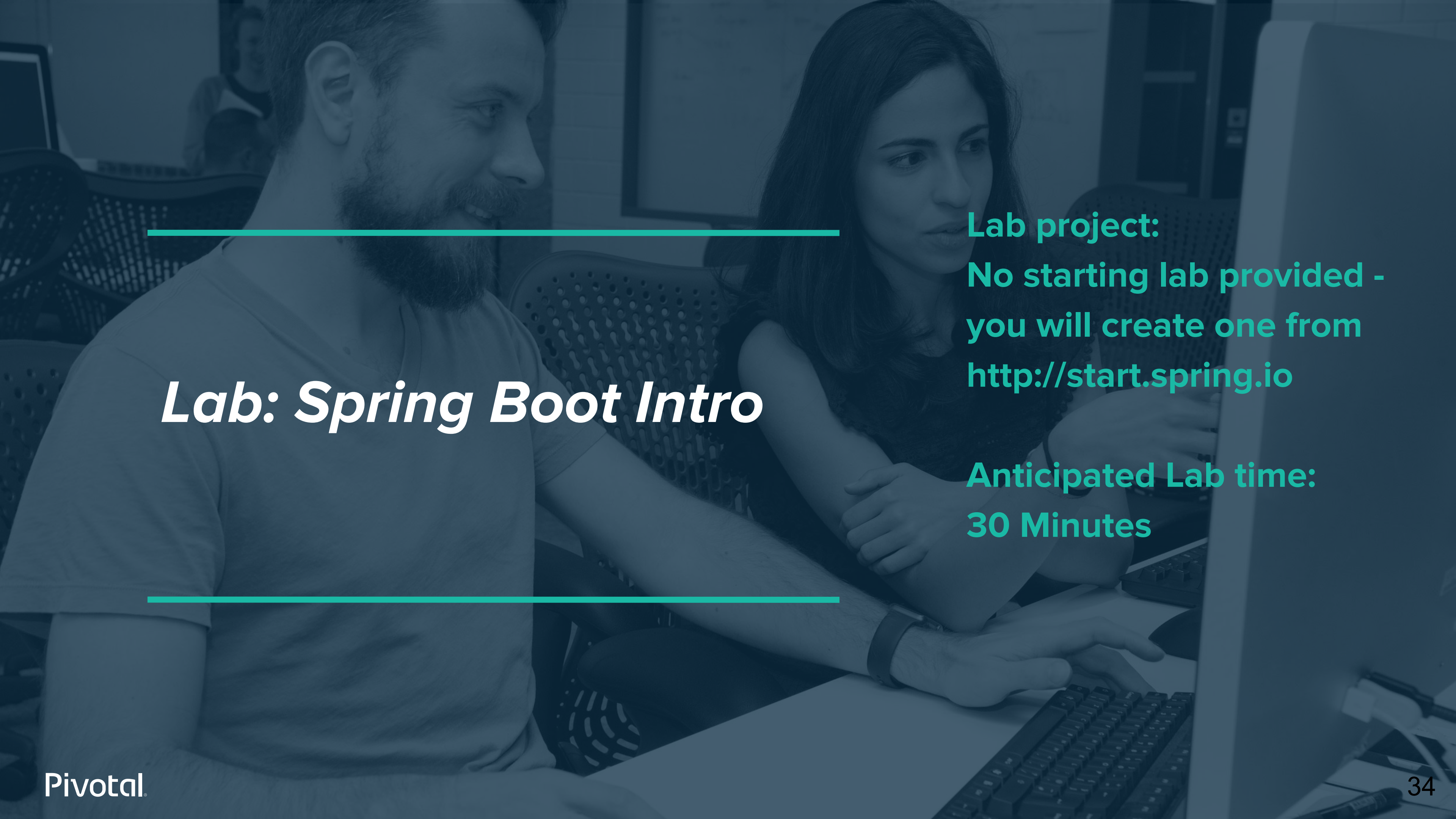
```
java -jar helloApp-0.0.1-SNAPSHOT.jar
```



```
$> java -jar helloApp-0.0.1-SNAPSHOT.jar
...
Hello, there are 21 accounts
...
$>
```

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
  - Provides `@SpringBootTest` for enhanced testing features



Lab: Spring Boot Intro

Lab project:
No starting lab provided - you will create one from http://start.spring.io

Anticipated Lab time:
30 Minutes

Pivotal