Intro to Spring Boot

Objectives

When we are done, you should be able to:

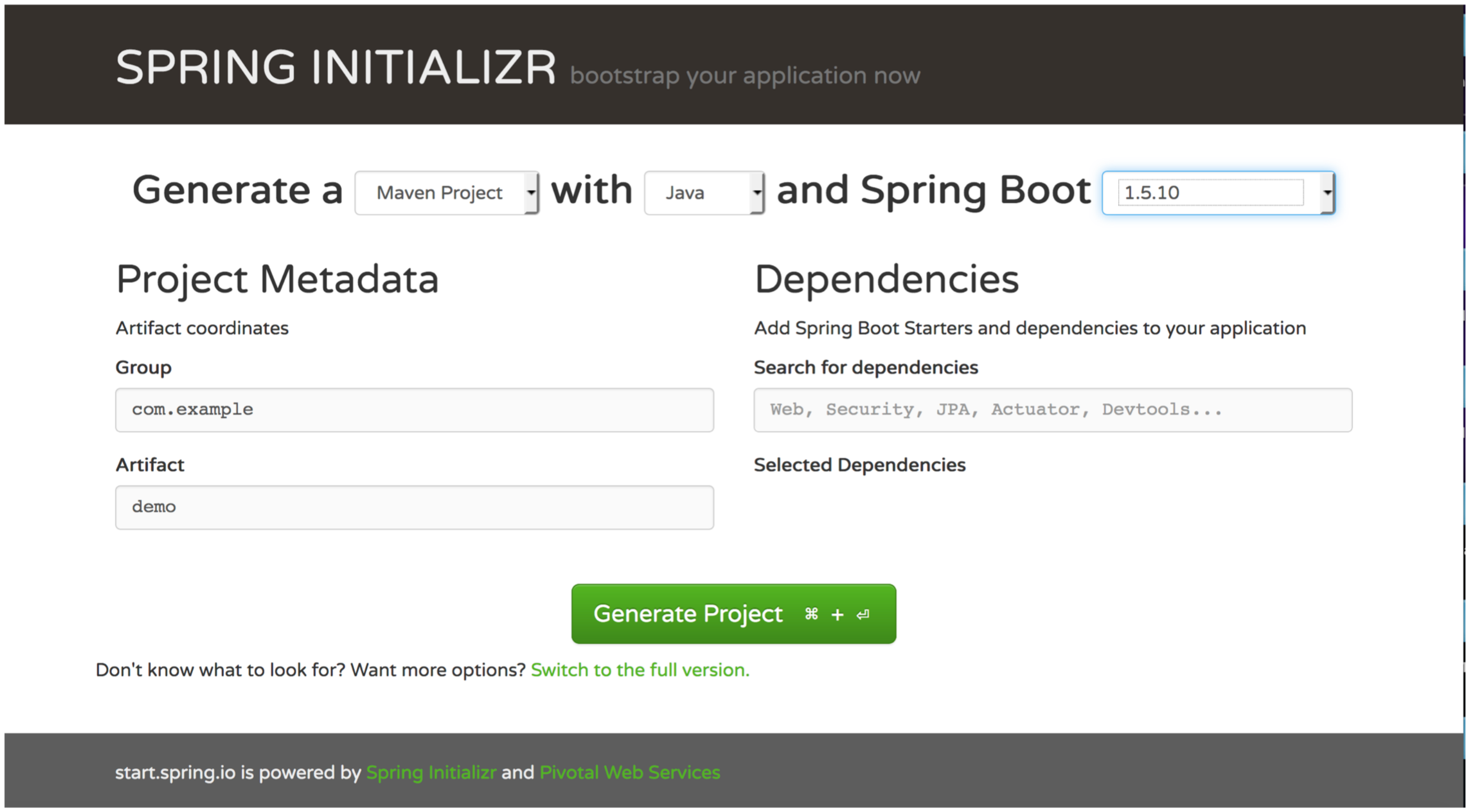
- Explain the purpose of Spring Boot
- Use Spring's Initializr to build a Spring Boot application
- Understand and utilize starter projects
- Turn on logging for auto-configuration

What is Spring Boot?

- Faster way to set up Spring applications
- Makes assumptions based on machine's configurations
  - May mean that it brings in things you need but don't have
    - i.e. Tomcat if you are building a web application and don't have a server configured for it
- It does *not* generate code
- Based on 3 major pieces
  - Starter projects
  - Starter parent
  - Auto-configuration

How to Build Spring Boot Application

- Could build 'by hand'
- Use Spring's Initializr
  - Found at start.spring.io



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Java and Spring Boot 1.5.10

Project Metadata

Artifact coordinates

Group

com.example

Artifact

demo

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Web, Security, JPA, Actuator, Devtools...

Selected Dependencies

Generate Project ⌘ + ⏎

Don't know what to look for? Want more options? Switch to the full version.

start.spring.io is powered by Spring Initializr and Pivotal Web Services

Spring Boot Starter Projects

- Set of dependency descriptors that can be included in app
  - You don't need to know all of the jar files needed, just these 'starters'
    - `spring-boot-starter` would replace all dependencies currently in our `pom.xml` plus some
    - `spring-boot-starter-web` adds approximately 30 more JARs
  - Is "opinionated" import
    - There are presumptions made, but these presumptions can be overridden

Spring Boot Starter Parent

- The root of all boot starters
- Allows for management of multiple child projects
  - Allows for dependency versioning
  - Default plugin configuration
  - Configuration
- Inherits from `spring-boot-dependencies`
  - Gives us the default dependencies
  - Defaults can be changed in `src/main/resources/application.properties`
  - Full list of property options can be found: https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

Auto-Configuration

- There are things that will always be needed, so why should we always configure them?
  - Let Spring configure them for us
- Auto-configuration can be overridden
  - Frameworks available on CLASSPATH
    - i.e. Default is to use JavaSE 1.6. If it sees 1.8 on your CLASSPATH, then it will configure to use 1.8
  - Existing configurations
- Implemented in `spring-boot-autoconfigurer.jar`

Debugging Auto-Configuration

- Two ways
  - Turn on logging
  - Use Spring Boot Actuator
- To use logging:
  - Add property to `application.properties`
    - `logging.level.org.springframework:DEBUG`
    - Once application restarts, an auto-config report is printed to console
      - Positive matches – what was configured
      - Negative matches – what was not configured

Spring Boot Actuator

- Gives us runtime information about the application
  - Health, metrics, state, etc.
- Need to have `spring-boot-starter-actuator`
- Needs something to read it
  - Designed to use RESTful services that read or write to it
  - Can use HAL browser
    - groupId: org.springframework.data
    - artifactId: spring-data-rest-hal-browser
    - Results are found at: http://localhost:8080/actuator/#http://localhost:8080/autoconfig

`@SpringBootApplication`

- Annotation that defines this class as the 'starter' class for your application
- A convenience annotation that adds
  - `@Configuration`
  - `@ComponentScan`
  - `@EnableAutoConfiguration`
    - Tells Spring Boot to attempt to configure the application based on what is on your classpath
  - `@EnableWebMvc` if `spring-webmvc` is on classpath

Running Spring Boot Application

- Spring Initializr gives us class annotated with `@SpringBootApplication`
  - This class also has `main` method
  - `SpringApplication.run()`
    - This method call 'starts' Spring Boot
    - You would then call your own methods if you need to

```
@SpringBootApplication
public class LibraryApplication {

  public static void main(String[]args) {
    SpringApplication.run(LibraryApplication.class,args);
    MainApplication.oldMainMethod();
}
```

Note: Normally instead of calling the 'oldMainMethod', we would run unit tests. We'll discuss that in another module.