Part 1: Foundational Spring

# 1. Getting started with Spring

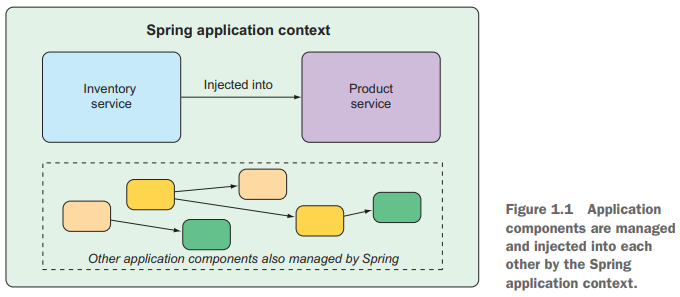
## 1.1 What is Spring

-Any application comprises many **components**, each responsible for its **functionality**. Spring offers a **container (Spring application context)** that creates and manages application components. These components (**beans**) are wired together inside **Spring application context (container)** to make a complete app.

-The act of wiring beans together based on **dependency injection (DI)**. A dependency-injected app relies on a separate entity (the container) to create and maintain all components and inject those into the beans that need them. This is done through **constructor arguments** or **property accessor methods**.

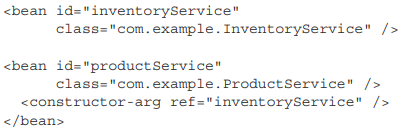
- Besides container, Spring offers a web framework, data persistence options, security framework, integration with other system, runtime monitoring, microservice support, reactive programming model…

-You would guild Spring application context to wire beans together by XML files that described the components and their relationship.

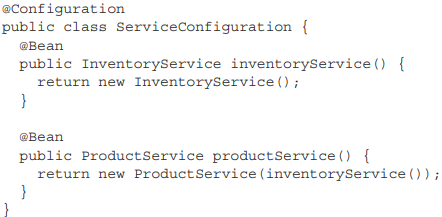


-Example:

+**XML configuration**



+**Java-based configuration**



The @Configuration annotation indicates to **Spring configuration class**. The configuration’s method are annotated with @Bean

-**Explicit configuration** is only necessary if Spring is unable to automatically configure components. **Automatic configuration** has its root in **component scanning**(Spring can automatically discover component from app’s classpath and create them as beans) and **autowiring** (Spring automatically injects the components with other beans they depend on)

-**Spring Boot** is an extension of Spring Framework, offer **autoconfiguratio***n.* It reduced the amount of explicit configuration required to build an app.

-In this book: we use **Spring Boot** as much as possible and explicit configuration only when necessary, focus on Spring’s **Java-based** configuration.

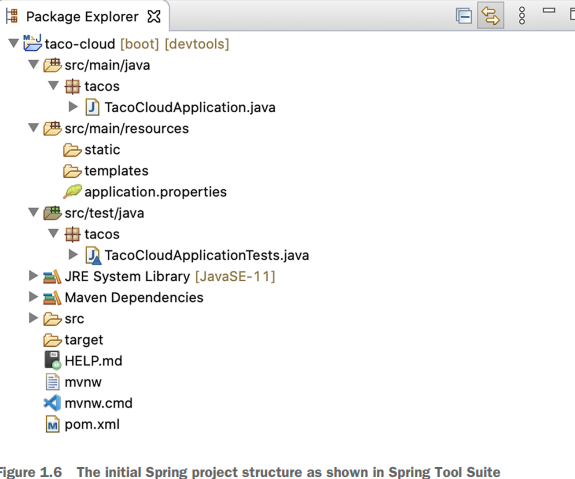
## 1.2 Initializing a Spring application

You lean on **Spring Initializr** to bootstrap application. Spring Initializr is both a browser-based web app and a REST API which can product a skeleton Spring project structure. In this book: we create new project with Spring Initializr support in **Spring Tool Suite**. It offers a Spring Boot Dashboard feature that make it easy to start, restart and stop Spring Boot app from IDE.

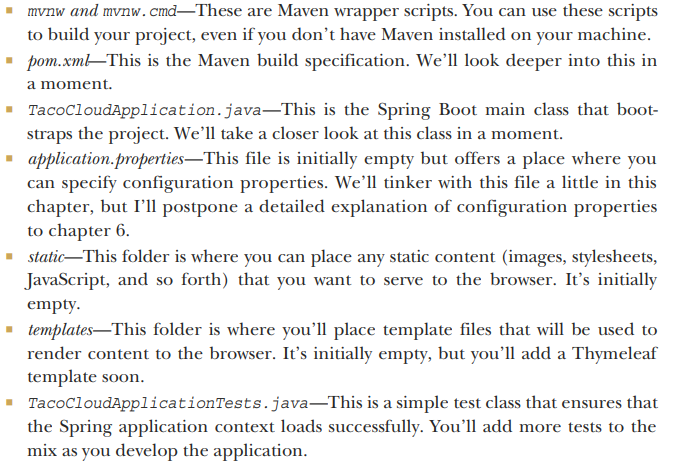
https://spring.io/tools

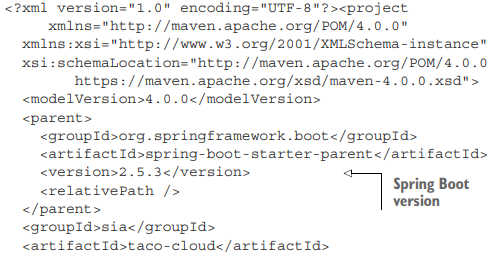
-Initializing a Spring project with Spring Tool Suite

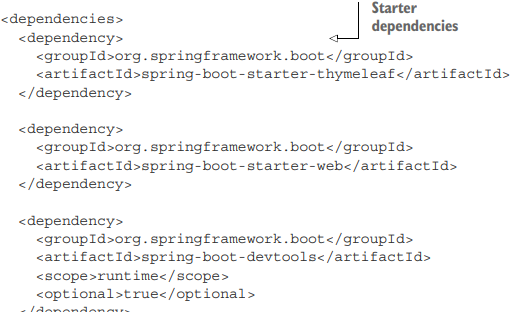
-Examining the Spring project structure

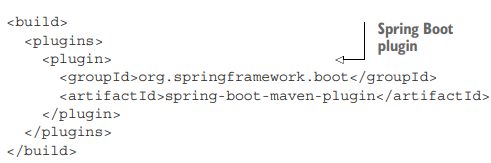


1.2.1 Exploring the build specification



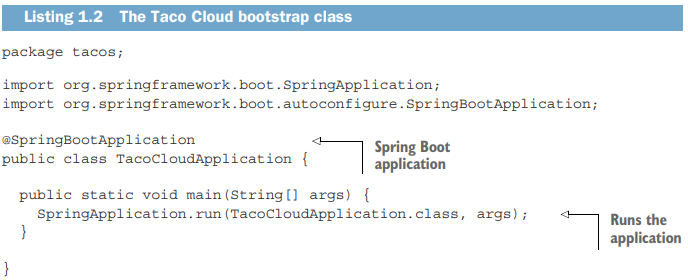






1.2.2 Bootstrap the application

-It’s important to have a main class that will be executed with amount of Spring configuration.



+@SpringBootApplication annotation signifies that this is a Spring boot application. It is composite annotation that combines 3 annotations: @SpringBootConfiguration, @EnableAutoConfiguration, @ComponentScan

+**main()**: run when JAR file is executed.

+**SpringApplication.run()**: perform the bootstrapping of the app, creating the Spring application context. 2 parameters are: configuration class and the command-line argument.

1.2.3 Testing the application

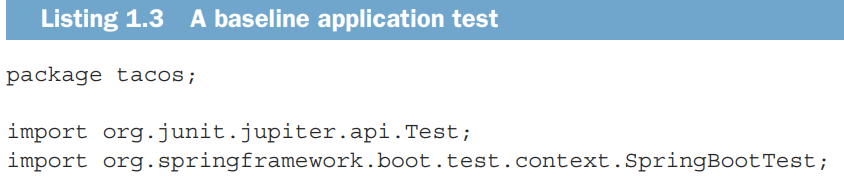
-Test the project manually by building and running it from cmd:

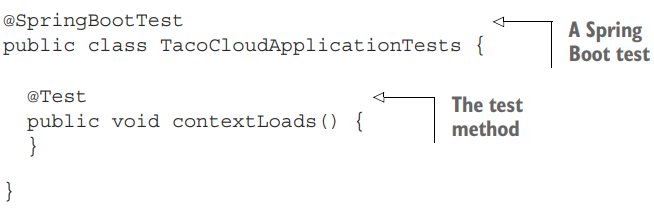




+Spring Boot: 

-Spring Initializr gives you an automated test





+Run any test classes:

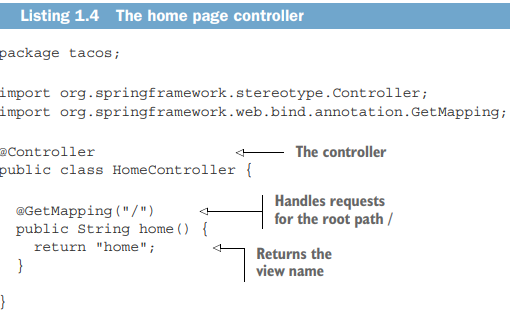


## 1.3 Writing a Spring application

-Add the home page, you’ll create 2 code artifacts: a controller class and a view table.

1.3.1 Handling web requests

-Spring comes with a web framework **Spring MVC**. **Controller**: class handles requests and responds.



+@Controller annotation identifies this class as a component for component scanning

+**home()** annotated with **@GetMapping**: this method handle the HTTP GET request with “/”. It returns logical name of view.

-You define the view template with **Thymeleaf**. The template name is derived from the logical view name by /templates/ and .html. So you need to place the template in /src/main/resources/templates/home.html

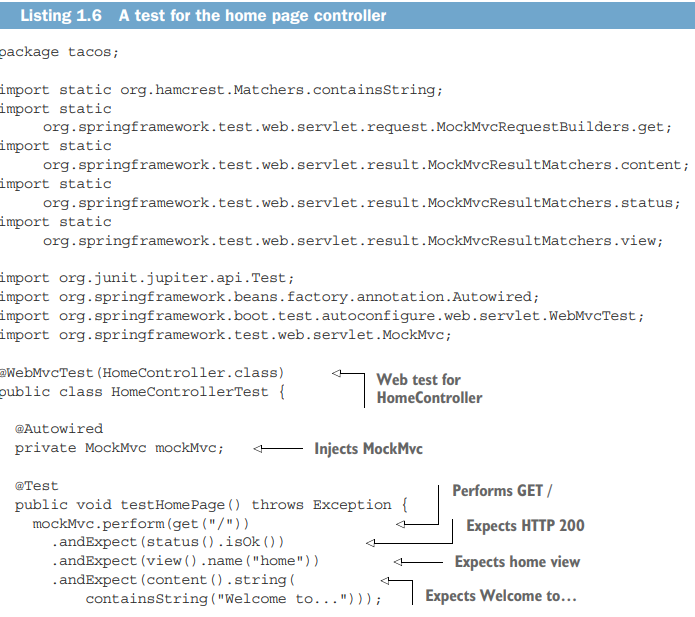
1.3.2 Defining the view

-The basic Thymeleaf template home page:



+Thymeleaf **th:src** attribute and an **@{}** expression to reference the image with a context-relative path.

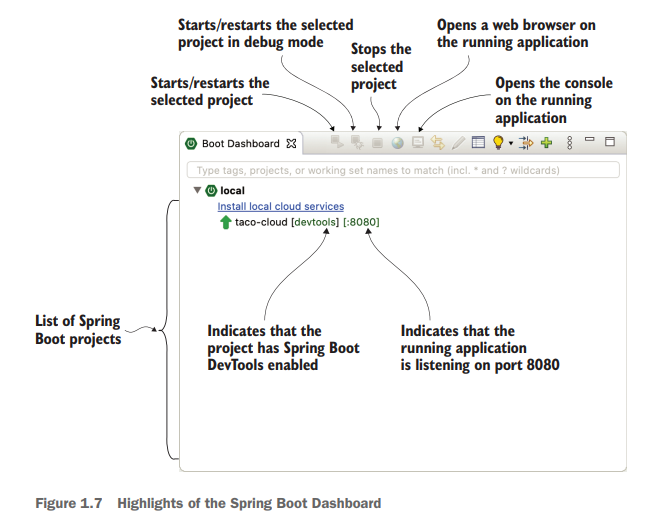
1.3.2 Testing the controller

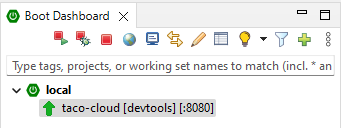


-Maven test



1.3.4 Building and running the application



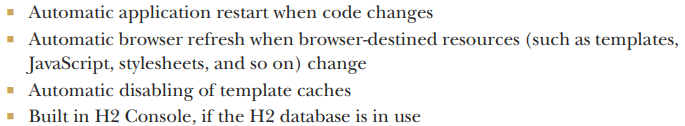




+Deploy the application to a Tomcat web server. Tomcat is a part of your application.

1.3.5 Getting to know Spring Boot DevTools

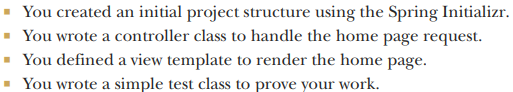
-**DevTools** provides development-time tool:



-DevTools isn’t an IDE plugin. It works well in STS4, Intelij, NetBeans. It disables itself when deploying.

1.3.6 Let’s review

-In short, you’ve do these steps:



## 1.4 Surveying the Spring landscape

-The core Spring Framework

+The foundation of everything else in Spring universe.

+Provide core container and DI framework.

+These are Spring MVC, Spring’s web framework. It also offers data persistence ( JDBC), reactive-style programming (WebFlux)

-Spring Boot

+starter dependencies and autoconfiguration.

+Runtime insight, specification of environment properties, testing support.

+An alternative programming model Spring Boot CLI.

-Spring Data

+Define your application’s data repositories as simple Java interfaces, using a name convention when defining methods to drive how data is stored and retrieved

+Work with different databases: relational (via JDBC or JPA), document (Mongo), graph (Neo4j)…

-Spring Security

+Authentication, authorization, API security.

-Spring Integration and Spring Batch

+Spring Integration addresses real-time integration. In contrast, Spring Batch addresses batched integration.

-Spring Cloud

+Microservices

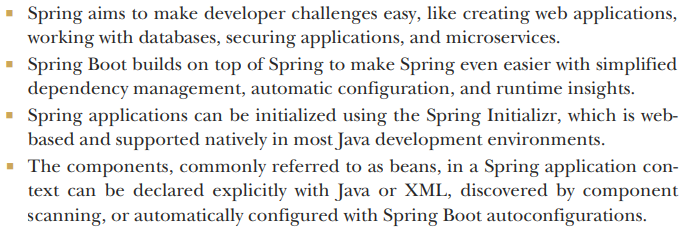
+http://www.manning.com/books/cloud-native-spring-in-action

-Spring Native

+Enable compilation of Spring Boot projects into native executable using GraalVM native-image compile.

+https://github.com/spring-projects-experimental/spring-native

## Summary

****