

arnaud.nauwynck@gmail.com

Introduction to SpringBoot (Spring Framework @AutoConfigure...)

This document:
<http://arnaud-nauwynck.github.io/docs/Intro-SpringBoot.pdf>

I will use Google before asking dumb questions. I will use Google before asking dumb questions. I will use Google before asking dumb questions.



springboot



All

Images

Videos

News

Books

More ▾

Search tools

About 1,500,000 results (0.39 seconds)

Did you mean: **spring boot**

Spring Boot - Projects

<https://projects.spring.io/spring-boot/> ▾

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring ...

Spring Boot Reference Guide

docs.spring.io/spring-boot/docs/current/reference/htmlsingle/ ▾

This section provides a brief overview of **Spring Boot** reference documentation. Think of it as map for the rest of the document. You can read this reference guide ...

Getting Started · Building an Application with Spring Boot

<https://spring.io/guides/gs/spring-boot/> ▾

This guide provides a sampling of how **Spring Boot** helps you accelerate and facilitate application development. As you read more Spring Getting Started guides ...

GitHub - spring-projects/spring-boot: Spring Boot

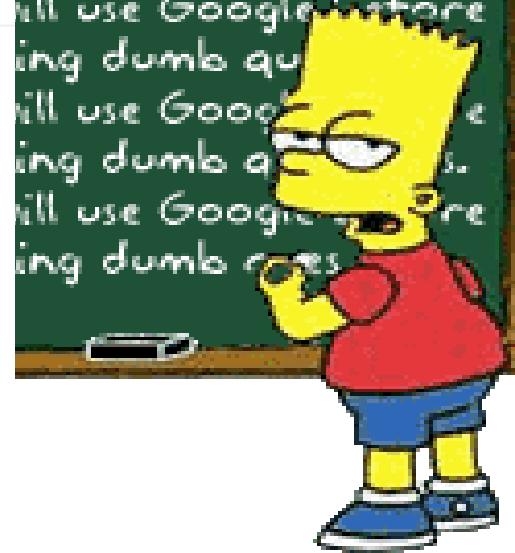
<https://github.com/spring-projects/spring-boot> ▾

Spring Boot makes it easy to create Spring-powered, production-grade applications and services with absolute minimum fuss. It takes an opinionated view of the ...

Spring Boot 1.4 Release Notes · spring-projects/spring-boot ...

<https://github.com/spring-projects/spring-boot/.../Spring-Boot-1.4-Release-Note...> ▾

Classes, methods and properties that were deprecated in **Spring Boot** 1.3 have been removed in this release. Please ensure that you aren't calling deprecated ...



Google Rank #1 “springboot”

<https://projects.spring.io/spring-boot/>

no Wikipedia?

The screenshot shows the Spring Projects website with the Spring logo at the top. The navigation bar includes links for DOCS, GUIDES, PROJECTS (which is highlighted in green), BLOG, QUESTIONS, and a search icon. Below the navigation, there's a section for 'PROJECTS' with a 'Spring Boot' card. The card contains a brief description: 'Takes an opinionated view of building production-ready Spring applications. Spring Boot favors convention over configuration and is designed to get you up and running as quickly as possible.' It features a 'QUICK START' button and a 'Spring with Convention Over Configuration' callout. A blue speech bubble on the right side of the card says 'A Sub-Project (among 100) of SpringFramework'. At the bottom of the page, there's a summary of Spring Boot's benefits: 'Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run". We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration.' A 'Features' section lists several benefits, and a 'RELEASE' table shows the current versions available.

A Sub-Project (among 100) of SpringFramework

Spring with Convention Over Configuration

Easier
Smaller
Just work

Spring Boot

RELEASE	DOCUMENTATION
2.0.0 SNAPSHOT	Reference API
1.5.0 SNAPSHOT	Reference API
1.4.3 SNAPSHOT	Reference API
1.4.2 CURRENT	Reference API
1.3.9 SNAPSHOT	Reference API

SpringBoot = a Sub-Project of SpringFramework

“What is SpringBoot ? “

=

“What is SpringFramework ?”

+

“What is in SpringBoot & not in SpringFramework ?”

<http://blog.mimacom.com/introduction-to-spring-boot/>



well the main concept of Spring Boot:



I think it is an excellent analogy about what Spring and Spring Boot are. With Spring framework you have a lot of great ingredients to make a yummy cake (or Spring application), and with Spring Boot you have a cook that will look the



springframework



All Images Videos News Shopping More ▾ Search tools

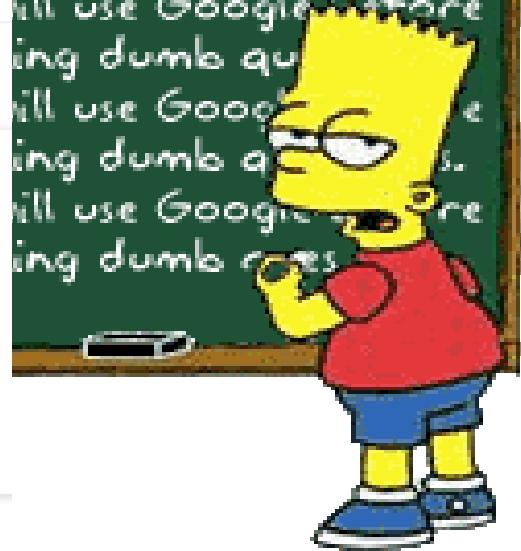
About 2,550,000 results (0.40 seconds)

The **Spring Framework** is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform.

Spring Framework - Wikipedia

https://en.wikipedia.org/wiki/Spring_Framework

[About this result](#) • [Feedback](#)



Spring Framework - Projects

<https://projects.spring.io/spring-framework/> ▾

A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Spring

<https://spring.io/> ▾

Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications.

Spring Framework Reference Documentation

docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/ ▾

I. Overview of **Spring Framework**. 1. Getting Started with Spring; 2. Introduction to the **Spring Framework**. 2.1. Dependency Injection and Inversion of Control; 2.2.

“SpringFramework” Wikipedia



WIKIPEDIA
The Free Encyclopedia

Article Talk

Read Edit View history

Not logged in Talk Contributions Create account Log in

Search Wikipedia



Spring Framework

From Wikipedia, the free encyclopedia

The **Spring Framework** is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions for building web applications on top of the Java EE platform. Although the framework does not impose any specific programming model, it has become popular in the Java community as an alternative to, replacement for, or even addition to the Enterprise JavaBeans (EJB) model. The Spring Framework is open source.

Contents [hide]

- 1 Version history
- 2 Modules
 - 2.1 Inversion of control container (dependency injection)
 - 2.2 Aspect-oriented programming framework
 - 2.3 Data access framework
 - 2.4 Transaction management framework
 - 2.5 Model-view-controller framework
 - 2.6 Remote access framework
 - 2.7 Convention-over-configuration rapid application development
 - 2.7.1 Spring Boot
 - 2.7.2 Spring Roo
 - 2.8 Batch framework
 - 2.9 Integration framework
- 3 Criticisms
- 4 See also

Spring Framework



Pivotal Software

Developer(s)	Pivotal Software
Initial release	1 October 2002; 14 years ago
Stable release	4.3.2 [1] / June 10, 2016
Preview release	5.0.0 M2 / September 21, 2016
Repository	github.com/spring-projects/spring-framework [2]
Development status	Active
Written in	Java
Operating system	Cross-platform
Platform	Java Virtual Machine
Type	Application framework
License	Apache License 2.0
Website	spring.io [3]



WIKIPEDIA
The Free Encyclopedia

Wikipedia Extract

The Spring Framework is an **application framework** and **inversion of control container** for the Java platform.

can be used by any Java application ..
.. extensions for web applications.

... popular in the Java community .. alternative EJB.

open source

History & Authors

- framework for J2EE-* libs in <xml>...
 - Developped by Rod Johnson & Juergen Hoeller
 - in ~ 2003

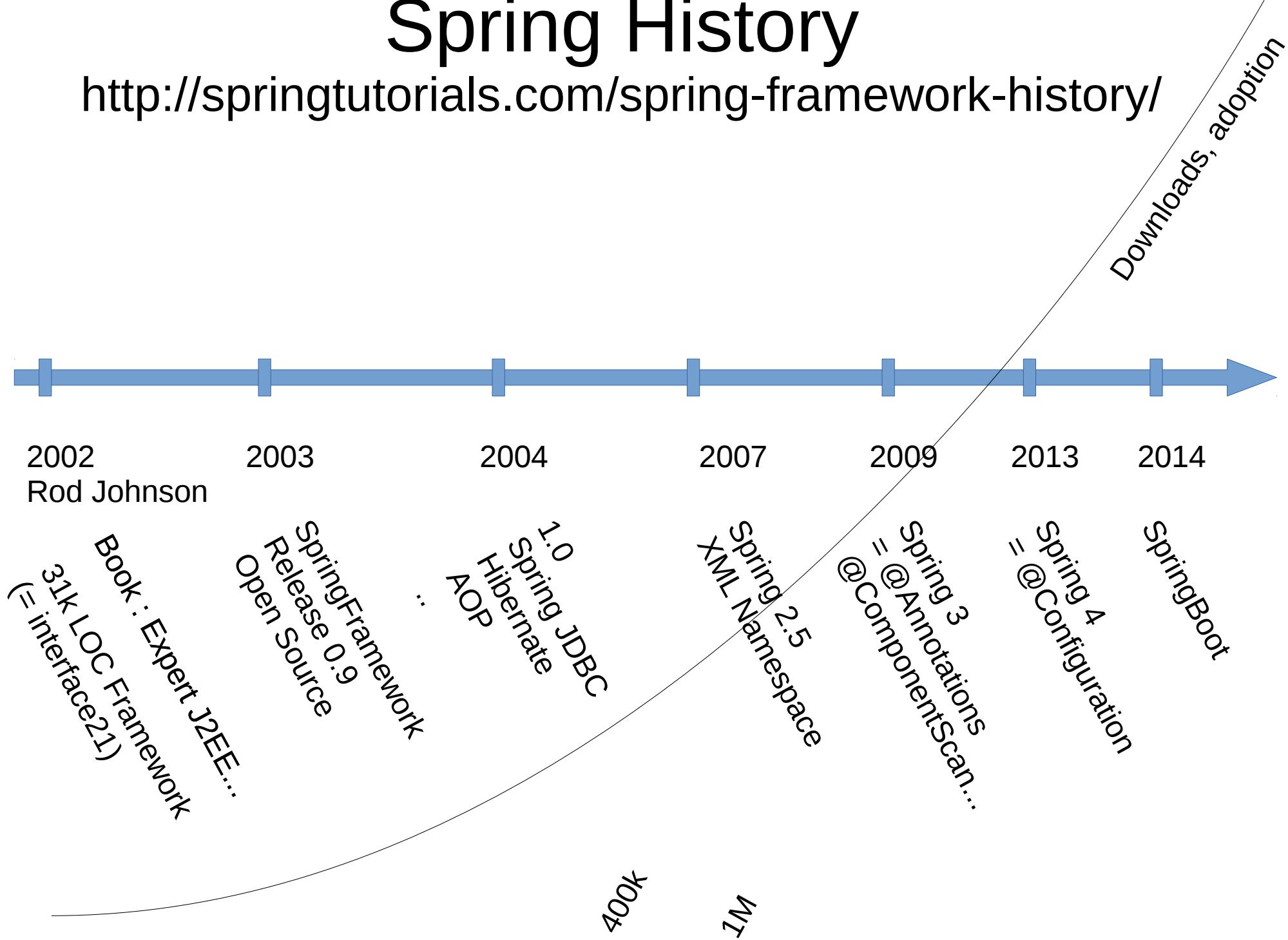


- SpringBoot is @Automagic without </xml>
 - From Dave Sayer & Phil Webb
 - In ~ 2014



Spring History

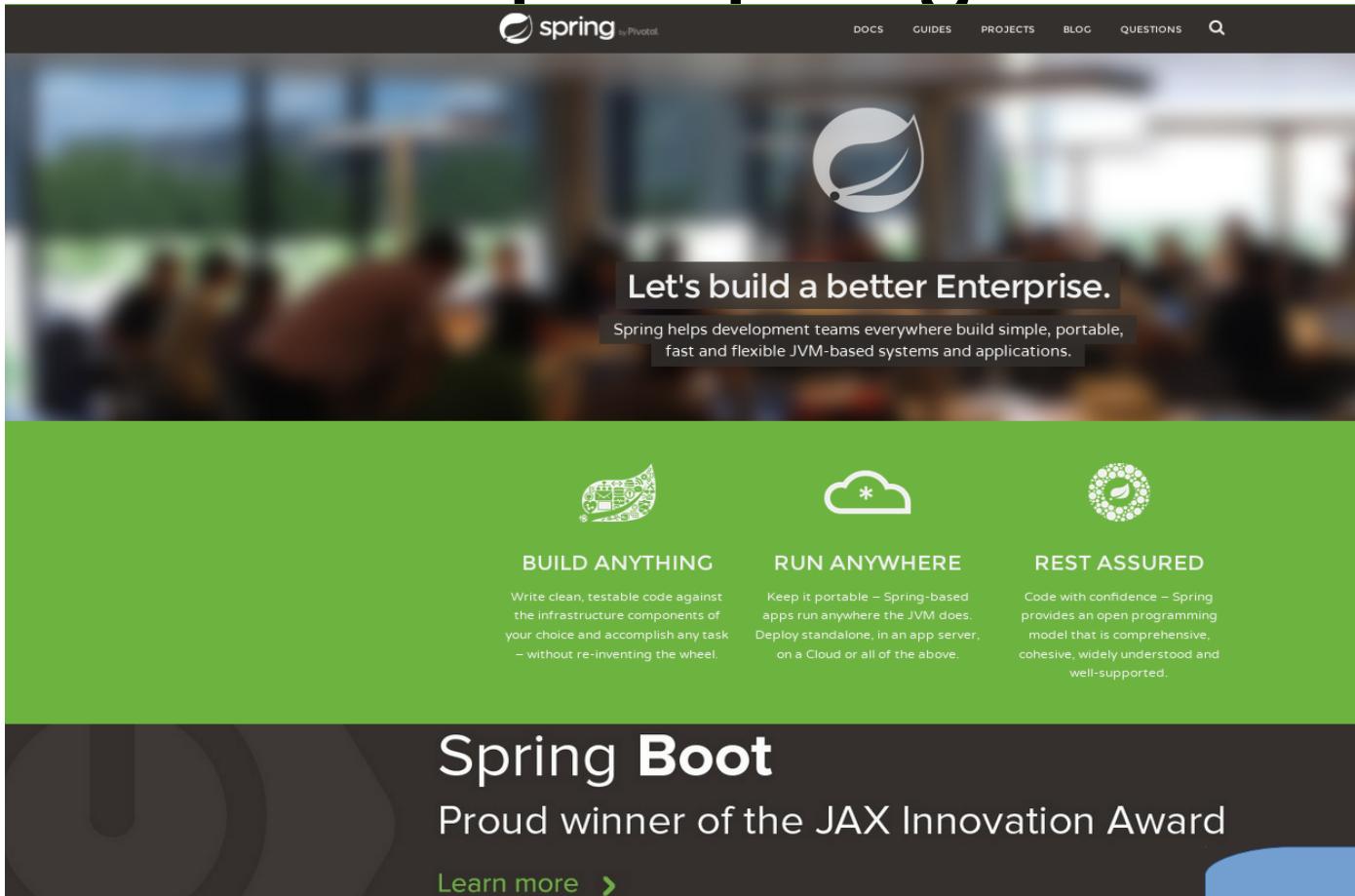
<http://springtutorials.com/spring-framework-history/>



Game of the Name (French Translation)

- Spring = “ressort”, “printemps”, “renouveau”
 - After the cold winter of ugly EJB specs 1.0, 2.0, ...
- Framework = “Cadre de Travail”
 - = Way of working, proposed / imposed by library
- Boot = “démarrage”
 - = your app is a 1 line main() with @magic

http://spring.io



The screenshot shows the official website for Spring Framework. At the top, there's a navigation bar with links for DOCS, GUIDES, PROJECTS, BLOG, and QUESTIONS, along with a search icon. Below the navigation is a large banner featuring a blurred background image of people at an event. In the center of the banner is a stylized leaf logo. Below the logo, the text "Let's build a better Enterprise." is displayed, followed by a subtitle: "Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications." The main content area has a green header with three sections: "BUILD ANYTHING" (with a gear icon), "RUN ANYWHERE" (with a cloud icon), and "REST ASSURED" (with a circular icon). Each section contains a brief description of the framework's capabilities. Below this is a dark banner for "Spring Boot", which is described as a "Proud winner of the JAX Innovation Award". A "Learn more" button is located below the banner. At the bottom, there's a "Spring Initializr" section with a "Generate now!" button, and a "Guides" section with a "Browse the Guides" button. A blue speech bubble on the left points to the "Guides" section, and another blue speech bubble on the right points to the "Initializr" section.

spring by Pivotal

DOCS GUIDES PROJECTS BLOG QUESTIONS

Let's build a better Enterprise.

Spring helps development teams everywhere build simple, portable, fast and flexible JVM-based systems and applications.

BUILD ANYTHING

Write clean, testable code against the infrastructure components of your choice and accomplish any task – without re-inventing the wheel.

RUN ANYWHERE

Keep it portable – Spring-based apps run anywhere the JVM does. Deploy standalone, in an app server, on a Cloud or all of the above.

REST ASSURED

Code with confidence – Spring provides an open programming model that is comprehensive, cohesive, widely understood and well-supported.

Spring Boot

Proud winner of the JAX Innovation Award

Learn more >

Spring Initializr

Bootstrap your Spring Boot application with [start.spring.io](#).

Generate now! >

Guides

Whether you're an expert or a newcomer our task-focused Getting Started Guides and Tutorials are designed to get you productive with Spring as quickly as possible.

Browse the Guides >

Amazing
Tutorials & Doc

Amazing Starter

<https://spring.io/guides>

[DOCS](#)[GUIDES](#)[PROJECTS](#)[BLOG](#)[QUESTIONS](#)

Guides

Whatever you're building, these guides are designed to get you productive as quickly as possible – using the latest Spring project releases and techniques as recommended by the [Spring team](#).

Have a suggestion for a new guide? Let us know at [@springcentral](#).



Getting Started Guides

Designed to be completed in 15-30 minutes, these guides provide quick, hands-on instructions for building the "Hello World" of any development task with Spring. In most cases, the only prerequisites are a JDK and a text editor.

[Building a RESTful Web Service](#)

Learn how to create a RESTful web service Learn how to schedule tasks with Spring.

[Building Java Projects with Gradle](#)

Learn how to build a Java project with Gradle.

[Uploading Files](#)

Learn how to build a Spring application that accepts multi-part file uploads.

[Scheduling Tasks](#)

Learn how to build a Java project with Maven.

[Authenticating a User with LDAP](#)

Learn how to secure an application with LDAP.

[Messaging with Redis](#)

[Registering an Application with](#)

[Consuming a RESTful Web Service](#)

Learn how to retrieve web page data with Spring's RestTemplate.

[Accessing Relational Data using JDBC with Spring](#)

Learn how to access relational data with Spring.

[Registering an Application with Facebook](#)

Learn how to register an application to integrate with Facebook.

[Messaging with RabbitMQ](#)

1/ Download + 2/ Mvn + 3/ Eclipse

<http://start.spring.io/>

SPRING INITIALizr bootstrap your application now

Generate a with Spring Boot

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 alt + ↵

Don't know what to look for? Want more options? [Switch to the full version.](#)

1/ Download + 2/ Mvn + 3/ Eclipse

```
$ mvn clean compile
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-clean-plugin:2.6.1:clean (default-clean) @ demo ---
[INFO] Deleting /mnt/a_1tera2/homeData/arnaud/perso/devPerso/my-github/test-snippet/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ demo ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ demo ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /mnt/a_1tera2/homeData/arnaud/perso/devPerso/my-github/test-springboot/target/classes
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 1.380 s
[INFO] Finished at: 2016-11-15T07:57:46+01:00
[INFO] Final Memory: 18M/82M
[INFO] -----
```

\$

1/ Download + 2/ Mvn + 3/ Eclipse

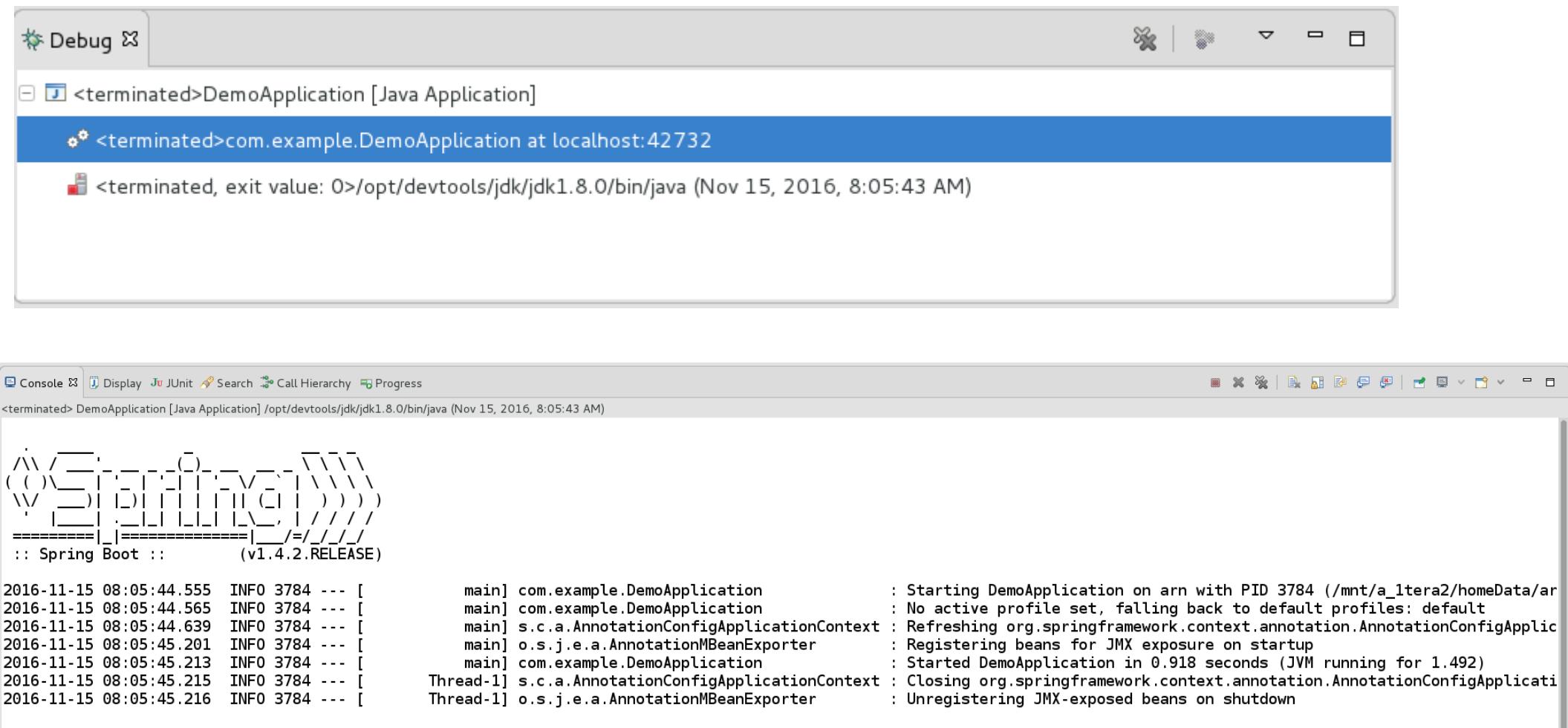
The screenshot shows the Eclipse IDE interface with the following details:

- Left Sidebar (Package Explorer):** Shows the project structure under "demo".
 - src/main/java:
 - com.example:
 - DemoApplication.java
 - src/main/resources
 - src/test/java
 - com.example
 - DemoApplicationTests.java
 - JRE System Library [JavaSE-1.8]
 - Maven Dependencies
 - src
 - target
 - mvnw
 - mvnw.cmd
 - pom.xml

Right Side (Editor): Displays the content of `DemoApplication.java`.

```
package com.example;  
  
import org.springframework.boot.SpringApplication;  
  
@SpringBootApplication  
public class DemoApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(DemoApplication.class, args);  
    }  
}
```

4/ Run IT ...



5/ Junit Test It !!

The screenshot shows a Java development environment with two main panes. On the left is the 'Package Explorer' pane, which displays the project structure of a 'demo' project. The structure includes 'src/main/java' containing 'com.example' with 'DemoApplication.java' and 'src/test/java' containing 'com.example' with 'DemoApplicationTests.java'. This file is currently selected and highlighted in blue. Other visible files in the package explorer include 'JRE System Library [JavaSE-1.8]', 'Maven Dependencies', 'src', 'target', 'mvnw', 'mvnw.cmd', and 'pom.xml'. On the right is the 'Code Editor' pane, which shows the content of 'DemoApplicationTests.java'. The code is annotated with Spring Boot and JUnit annotations:

```
package com.example;

import org.junit.Test;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.junit4.SpringRunner;

@SpringBootTest
public class DemoApplicationTests {

    @Test
    public void contextLoads() {
    }
}
```

5/ Junit OK

The screenshot shows the Eclipse IDE interface during a JUnit test execution. The top bar displays tabs for 'Debug' (selected), 'Breakpoints', 'Expressions', and 'Registers'. The left side features a code editor with 'DemoApplication.java' and 'DemoApplicationTests.java' tabs. The code editor displays Java test code using Spring Boot and JUnit annotations. The right side shows the 'Run' view with a summary of the test run results.

Debug

JUnit <terminated>DemoApplicationTests [JUnit]

<terminated>org.eclipse.jdt.internal.junit.runner.RemoteTestRunner at localhost:57359

<terminated, exit value: 0>/opt/devtools/jdk/jdk1.8.0/bin/java (Nov 15, 2016, 8:28:28 AM)

Hit count: Suspend thread

DemoApplication.java DemoApplicationTests.java

```
package com.example;

import org.junit.Test;

@RunWith(SpringRunner.class)
@SpringBootTest
public class DemoApplicationTests {

    @Test
    public void contextLoads() {
    }
}
```

Console Display JUnit Search Call Hierarchy Progress

Finished after 0.823 seconds

Runs: 1/1 Errors: 0 Failures: 0

com.example.DemoApplicationTests [Runner: JUnit 4] (0.008 s)

Failure Trace

contextLoads (0.008 s)

6/ Easy packaging to launch main ... java -jar

Small all-in-one target/jar

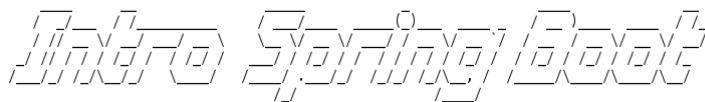
6.3 M “only”

```
$ ls -lh target
total 6.3M
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:29 classes
-rw-r--r-- 1 arnaud arnaud 6.3M Nov 16 21:29 demo-0.0.1-SNAPSHOT.jar
-rw-r--r-- 1 arnaud arnaud 2.7K Nov 16 21:29 demo-0.0.1-SNAPSHOT.jar.original
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:29 generated-sources
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:29 generated-test-sources
drwxr-xr-x 2 arnaud arnaud 4.0K Nov 16 21:29 maven-archiver
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:29 maven-status
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:29 test-classes
$ 
$ jar tf target/demo-0.0.1-SNAPSHOT.jar
META-INF/
META-INF/MANIFEST.MF
BOOT-INF/
BOOT-INF/classes/
BOOT-INF/classes/com/
BOOT-INF/classes/com/example/
BOOT-INF/classes/application.properties
BOOT-INF/classes/com/example/DemoApplication.class
META-INF/maven/
META-INF/maven/com.example/
META-INF/maven/com.example/demo/
META-INF/maven/com.example/demo/pom.xml
META-INF/maven/com.example/demo/pom.properties
BOOT-INF/lib/
BOOT-INF/lib/spring-beans-4.3.4.RELEASE.jar
BOOT-INF/lib/spring-boot-starter-logging-1.4.2.RELEASE.jar
BOOT-INF/lib/slf4j-api-1.7.21.jar
BOOT-INF/lib/jul-to-slf4j-1.7.21.jar
BOOT-INF/lib/spring-context-4.3.4.RELEASE.jar
BOOT-INF/lib/spring-core-4.3.4.RELEASE.jar
BOOT-INF/lib/spring-boot-starter-1.4.2.RELEASE.jar
BOOT-INF/lib/spring-aop-4.3.4.RELEASE.jar
BOOT-INF/lib/spring-boot-autoconfigure-1.4.2.RELEASE.jar
BOOT-INF/lib/logback-classic-1.1.7.jar
BOOT-INF/lib/spring-boot-1.4.2.RELEASE.jar
BOOT-INF/lib/spring-expression-4.3.4.RELEASE.jar
```

“Hello World” ok
But what is Really SpringBoot?

What's Next ?

An “Ascii Art Banner” Printer ?



Font Name: Slant

[Use Font](#) | [Select & Copy](#)

<http://patorjk.com/software/taag/>

src/main/resources/
banner.txt

The screenshot shows an IDE interface. On the left is a "Package Explorer" view showing a project structure with "Other Projects" containing "demo". Inside "demo" are "src/main/java" (containing "DemoApplication.java"), "src/main/resources" (containing "application.properties"), and "src/test/java". A blue callout bubble points to "src/main/resources/banner.txt". On the right is a "banner.txt" file viewer showing the ASCII art banner. Below it is a terminal window titled "terminated> DemoApplication [Java Application] /opt/devtools/jdk/jdk1.8.0/bin/java (Nov 16, 2016, 8:46:49 PM)" displaying the Java application's logs. The logs show the application starting up, registering beans, and shutting down.

```
2016-11-16 20:46:50.362 INFO 3401 ... [main] com.example.DemoApplication : Starting DemoApplication on arn with PID 3401
2016-11-16 20:46:50.370 INFO 3401 ... [main] com.example.DemoApplication : No active profile set, falling back to default
2016-11-16 20:46:50.414 INFO 3401 ... [main] s.c.a.AnnotationConfigApplicationContext : Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext
2016-11-16 20:46:50.971 INFO 3401 ... [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2016-11-16 20:46:50.981 INFO 3401 ... [main] com.example.DemoApplication : Started DemoApplication in 0.834 seconds (JVM startup time)
2016-11-16 20:46:50.985 INFO 3401 ... [Thread-1] s.c.a.AnnotationConfigApplicationContext : Closing org.springframework.context.annotation.AnnotationConfigApplicationContext
2016-11-16 20:46:50.987 INFO 3401 ... [Thread-1] o.s.j.e.a.AnnotationMBeanExporter : Unregistering JMX-exposed beans on shutdown
```

“Hello” that contains the “World” jars

The screenshot shows an IDE interface with several windows open:

- Package Explorer:** Shows the project structure under "demo".
- DemoApplication.java:** The main Java file.
- pom.xml:** The Maven configuration file.
- Dependency Hierarchy [test]:** A central window displaying the dependency tree. It shows the following structure:
 - spring-boot-starter: 1.4.2.RELEASE [compile]
 - spring-boot: 1.4.2.RELEASE [compile]
 - spring-core: 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-context: 4.3.4.RELEASE [compile]
 - spring-aop: 4.3.4.RELEASE [compile]
 - spring-beans: 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-core: 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-beans: 4.3.4.RELEASE [compile]
 - spring-core: 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-core: 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-expression: 4.3.4.RELEASE [compile]
 - spring-core: 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-boot-autoconfigure: 1.4.2.RELEASE [compile]
 - spring-boot: 1.4.2.RELEASE (omitted for conflict with 1.4.2.RELEASE) [compile]
 - spring-boot-starter-logging: 1.4.2.RELEASE [compile]
 - logback-classic: 1.1.7 [compile]
 - logback-core: 1.1.7 [compile]
 - slf4j-api: 1.7.21 (managed from 1.7.20) (omitted for conflict with 1.7.21) [compile]
 - jcl-over-slf4j: 1.7.21 [compile]
 - slf4j-api: 1.7.21 (omitted for conflict with 1.7.21) [compile]
 - jul-to-slf4j: 1.7.21 [compile]
 - slf4j-api: 1.7.21 (omitted for conflict with 1.7.21) [compile]
 - log4j-over-slf4j: 1.7.21 [compile]
- Resolved Dependencies:** A list of resolved dependencies with their versions and types.

At the bottom, tabs indicate the current view: Overview, Dependencies, Dependency Hierarchy, Effective POM, and pom.xml. A status bar at the bottom left says "Infinitest is waiting for changes" and "32 items selected".

mvn dependency:tree

Same as in eclipse, using command line

```
$ mvn dependency:tree
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO] -----
[INFO] --- maven-dependency-plugin:2.10:tree (default-cli) @ demo ---
[INFO] com.example:demo:jar:0.0.1-SNAPSHOT
[INFO] +- org.springframework.boot:spring-boot-starter:jar:1.4.2.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot:jar:1.4.2.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-context:jar:4.3.4.RELEASE:compile
[INFO] |  |  +- org.springframework:spring-aop:jar:4.3.4.RELEASE:compile
[INFO] |  |  +- org.springframework:spring-beans:jar:4.3.4.RELEASE:compile
[INFO] |  |  \- org.springframework:spring-expression:jar:4.3.4.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot-autoconfigure:jar:1.4.2.RELEASE:compile
[INFO] |  +- org.springframework.boot:spring-boot-starter-logging:jar:1.4.2.RELEASE:compile
[INFO] |  |  +- ch.qos.logback:logback-classic:jar:1.1.7:compile
[INFO] |  |  |  \- ch.qos.logback:logback-core:jar:1.1.7:compile
[INFO] |  |  +- org.slf4j:jcl-over-slf4j:jar:1.7.21:compile
[INFO] |  |  +- org.slf4j:jul-to-slf4j:jar:1.7.21:compile
[INFO] |  |  \- org.slf4j:log4j-over-slf4j:jar:1.7.21:compile
[INFO] |  +- org.springframework:spring-core:jar:4.3.4.RELEASE:compile
[INFO] |  \- org.yaml:snakeyaml:jar:1.17:runtime
[INFO] \- org.springframework.boot:spring-boot-starter-test:jar:1.4.2.RELEASE:test
[INFO]   +- org.springframework.boot:spring-boot-test:jar:1.4.2.RELEASE:test
[INFO]   +- org.springframework.boot:spring-boot-test-autoconfigure:jar:1.4.2.RELEASE:test
[INFO]   +- com.jayway.jsonpath:json-path:jar:2.2.0:test
[INFO]   |  +- net.minidev:json-smart:jar:2.2.1:test
[INFO]   |  |  \- net.minidev:accessors-smart:jar:1.1:test
[INFO]   |  |  \- org.ow2.asm:asm:jar:5.0.3:test
[INFO]   |  \- org.slf4j:slf4j-api:jar:1.7.21:compile
[INFO]   +- junit:junit:jar:4.12:test
[INFO]   +- org.assertj:assertj-core:jar:2.5.0:test
[INFO]   +- org.mockito:mockito-core:jar:1.10.19:test
[INFO]   |  \- org.objenesis:objenesis:jar:2.1:test
[INFO]   +- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO]   +- org.hamcrest:hamcrest-library:jar:1.3:test
[INFO]   +- org.skyscreamer:jsonassert:jar:1.3.0:test
[INFO]   |  \- org.json:json:jar:20140107:test
[INFO]   \- org.springframework:spring-test:jar:4.3.4.RELEASE:test
[INFO] -----
[INFO] BUILD SUCCESS
```

Dependencies ...

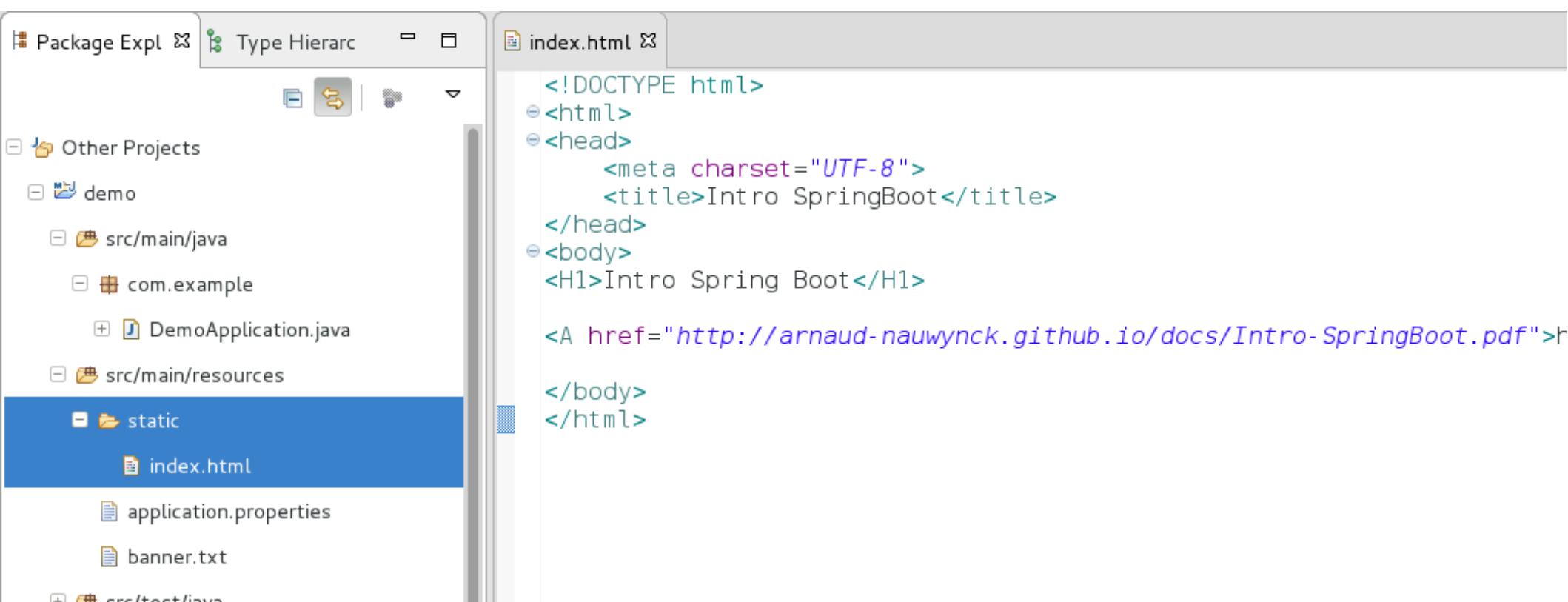
(excluding <scope>test</scope>)

```
$ mvn dependency:list | grep -v :test
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-dependency-plugin:2.10:list (default-cli) @ demo ---
[INFO]
[INFO] The following files have been resolved:
[INFO]   org.slf4j:jul-to-slf4j:jar:1.7.21:compile
[INFO]   org.springframework:spring-aop:jar:4.3.4.RELEASE:compile
[INFO]   org.yaml:snakeyaml:jar:1.17:runtime
[INFO]   org.springframework:spring-beans:jar:4.3.4.RELEASE:compile
[INFO]   org.springframework.boot:spring-boot-starter-logging:jar:1.4.2.RELEASE:compile
[INFO]   org.slf4j:slf4j-api:jar:1.7.21:compile
[INFO]   org.springframework:spring-context:jar:4.3.4.RELEASE:compile
[INFO]   org.springframework:spring-core:jar:4.3.4.RELEASE:compile
[INFO]   org.springframework.boot:spring-boot-starter:jar:1.4.2.RELEASE:compile
[INFO]   org.springframework.boot:spring-boot-autoconfigure:jar:1.4.2.RELEASE:compile
[INFO]   ch.qos.logback:logback-classic:jar:1.1.7:compile
[INFO]   org.springframework.boot:spring-boot:jar:1.4.2.RELEASE:compile
[INFO]   org.springframework:spring-expression:jar:4.3.4.RELEASE:compile
[INFO]   ch.qos.logback:logback-core:jar:1.1.7:compile
[INFO]   org.slf4j:jcl-over-slf4j:jar:1.7.21:compile
[INFO]   org.slf4j:log4j-over-slf4j:jar:1.7.21:compile
[INFO]
[INFO] -----
[INFO] BUILD SUCCESS
```

Almost a WebApp (add 4 lines)...

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

Add Web page resources/static/index.html



Restart... your Web Application running in 2 seconds

The screenshot shows a Java development environment with the following interface elements:

- Debug View:** Shows the application is running under the name "DemoApplication" at port 51469. It lists several threads: "Daemon Thread [ContainerBackgroundProcessor[StandardEngine[Tomcat]]] (Running)", "Thread [container-0] (Running)", and "Daemon Thread [NioBlockingSelector.BlockPoller-1] (Running)".
- Breakpoints View:** Displays the current state of breakpoints, with options to filter by hit count and control thread/suspend status.
- Code Editor:** The file "DemoApplication.java" is open, showing the main entry point of the application.

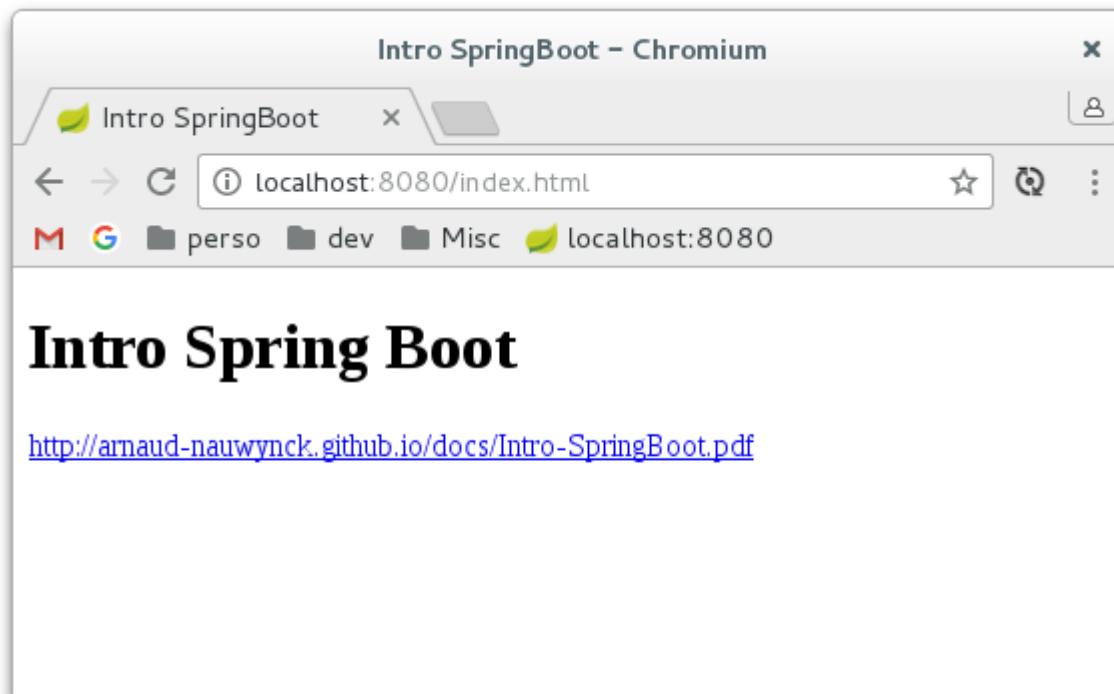
```
package com.example;
import org.springframework.boot.SpringApplication;
@SpringBootApplication
public class DemoApplication {
    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }
}
```

- Variables View:** Shows the current values of variables in the current scope.
- Outline View:** Shows the structure of the code, including classes, methods, and fields.
- Console View:** Displays the application logs and standard output.

```
DemoApplication [Java Application] /opt/devtools/jdk/jdk1.8.0/bin/java (Nov 16, 2016, 9:02:49 PM)
2016-11-16 21:02:51.091 INFO 4053 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8080 (http)
2016-11-16 21:02:51.101 INFO 4053 --- [           main] o.apache.catalina.core.StandardService : Starting service Tomcat
2016-11-16 21:02:51.102 INFO 4053 --- [           main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.6
2016-11-16 21:02:51.267 INFO 4053 --- [lost-startStop-1] o.a.c.c.C[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2016-11-16 21:02:51.267 INFO 4053 --- [lost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 1201 ms
2016-11-16 21:02:51.384 INFO 4053 --- [lost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2016-11-16 21:02:51.387 INFO 4053 --- [lost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [//*]
2016-11-16 21:02:51.388 INFO 4053 --- [lost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [//*]
2016-11-16 21:02:51.388 INFO 4053 --- [lost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [//*]
2016-11-16 21:02:51.388 INFO 4053 --- [lost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [//*]
2016-11-16 21:02:51.691 INFO 4053 --- [           main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationContext@51e1f3d2: startup sequence 1
2016-11-16 21:02:51.747 INFO 4053 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2016-11-16 21:02:51.748 INFO 4053 --- [           main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[[/error]]}" onto public org.springframework.http.ResponseEntity<java.util.Map> org.springframework.boot.autoconfigure.web.BasicErrorController.error(javax.servlet.http.HttpServletRequest)
2016-11-16 21:02:51.779 INFO 4053 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-11-16 21:02:51.779 INFO 4053 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-11-16 21:02:51.815 INFO 4053 --- [           main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.resource.ResourceHttpRequestHandler]
2016-11-16 21:02:51.843 INFO 4053 --- [           main] oConfiguration$WelcomePageHandlerMapping : Adding welcome page: class path resource [static/index.html]
2016-11-16 21:02:52.028 INFO 4053 --- [           main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2016-11-16 21:02:52.087 INFO 4053 --- [           main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2016-11-16 21:02:52.090 INFO 4053 --- [           main] com.example.DemoApplication : Started DemoApplication in 2.454 seconds (JVM running for 2.793)
```

<http://localhost:8080/>

... “Just work”



Details HTTP Protocol,Header



Intro Spring Boot

<http://arnaud-nauwynck.github.io/docs/Intro-SpringBoot.pdf>

A screenshot of the Chrome Developer Tools Network tab. The timeline shows a single request to "localhost" at port 8080. The response details show the following information:

- Name: localhost
- General
 - Request URL: http://localhost:8080/
 - Request Method: GET
 - Status Code: 304
 - Remote Address: [::1]:8080
- Response Headers
 - Date: Wed, 16 Nov 2016 20:39:38 GMT
 - Last-Modified: Wed, 16 Nov 2016 20:34:49 GMT
- Request Headers
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 - Accept-Encoding: gzip, deflate, sdch
 - Accept-Language: en-US,en;q=0.8,fr;q=0.6
 - Cache-Control: max-age=0
 - Connection: keep-alive
 - Cookie: remember-me=K1FMSTBXeVpEcUtPTXRDaDZMN0RBZz090mQyMlhINlhXavI
 - Host: localhost:8080
 - If-Modified-Since: Wed, 16 Nov 2016 20:34:49 GMT
 - Upgrade-Insecure-Requests: 1
 - User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)

```
$ curl -v http://localhost:8080/index.html
* Hostname was NOT found in DNS cache
*   Trying ::1...
* Connected to localhost (::1) port 8080 (#0)
> GET /index.html HTTP/1.1
> User-Agent: curl/7.38.0
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200
< Last-Modified: Wed, 16 Nov 2016 20:34:49 GMT
< Accept-Ranges: bytes
< Content-Type: text/html
< Content-Length: 279
< Date: Wed, 16 Nov 2016 20:37:25 GMT
<
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Intro SpringBoot</title>
</head>
<body>
<H1>Intro Spring Boot</H1>

<A href="http://arnaud-nauwynck.github.io/docs/Intro-SpringBoot.pdf">
</body>
* Connection #0 to host localhost left intact
</html>$
```

Add Dynamic Page (jsp, velocity, thymeleaf, ...)

For old-school html-1.0

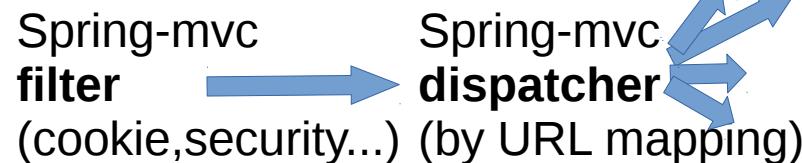
... only for poor PHP lost guies

Now in 2016 ... Using html-5

... You need/should/must NOT do like that

Dynamic Java Page = MVC

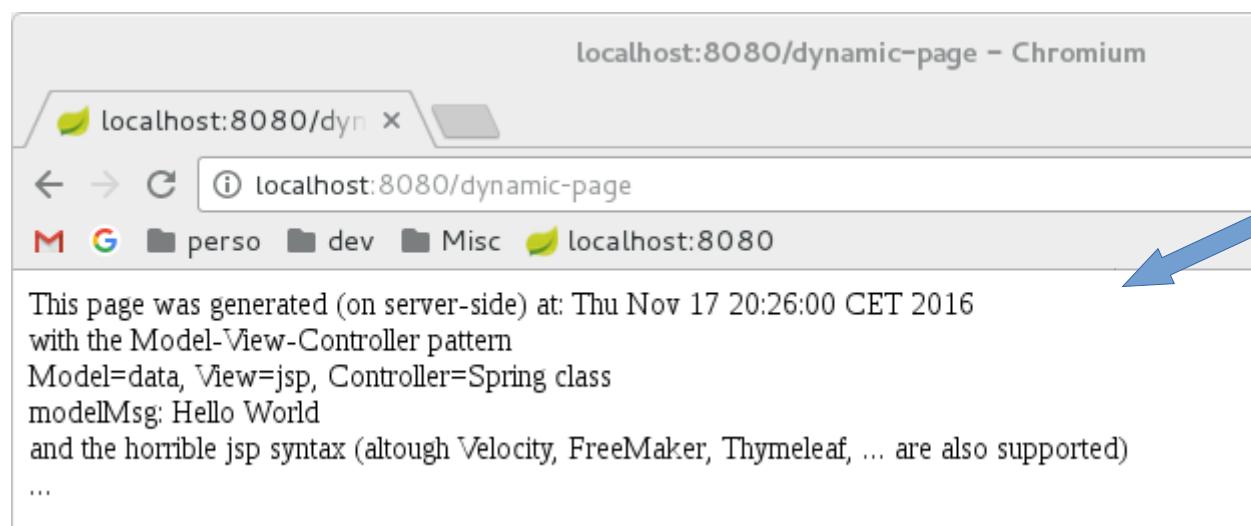
http://localhost:8080/dynamic-page



```
@Controller  
public class DynamicPageController {  
  
    @RequestMapping("/dynamic-page")  
    C = Controller  
  
    M = Model  
  
    return "page-view";  
}
```

Spring-mvc “viewResolver”
WEB-INF/jsp/page-view.jsp

V = View



WEB-INF = internal
..not public from outside
(no /page-view.jsp URL)

Springboot for Dynamic Pages

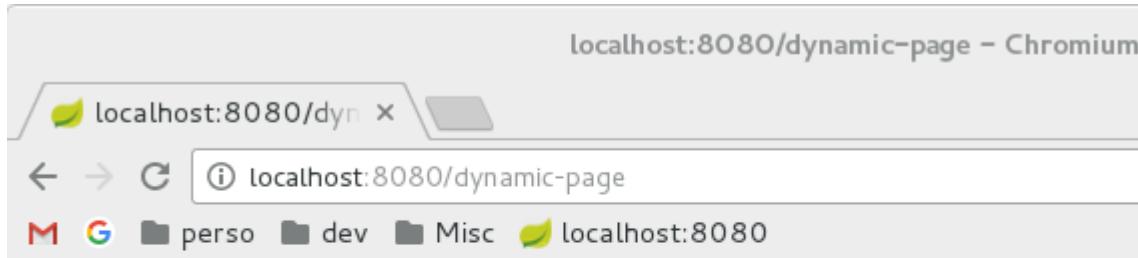
The screenshot shows a Java-based Springboot application structure in an IDE. The left sidebar lists project files:

- other Projects
- demo
- src/main/java
 - com.example
 - DemoApplication.java
 - DynamicPageController.java
 - HelloRestController.java
 - JdbcBatchCommand.java
 - src/main/resources
 - META-INF
 - resources
 - WEB-INF
 - jsp
 - page-view.jsp

The main area displays four code editor panes:

 - page-view.jsp**: Contains JSP code generating a timestamp and a loop that prints two dots.
 - DynamicPageController.java**: Contains a controller class with a single endpoint that returns "page-view".
 - application.properties**: Configuration setting the view prefix to "/WEB-INF/jsp" and suffix to ".jsp".
 - index.html**: A simple static file.

Dynamic Page JUST Work



Type-Safe, Multi-Thread-Safe, Compiled, Performant,
Garbage-Collected,
JRE Portable, Debuggable, Monitorable, Bytecode Agent
Object-Oriented, Aspect-Oriented, Annotations Processed,
Functional Lambda,
Multi-JDK-Langages (Groovy, Scala, Kotlin,...)
Rich Libs, Rich Ecosystem, Huge Community...

Poor PHP
Jalous Developpers

I am looking at you

Don't say "PHP" Again

Once Upon a Time ... Html5 JavaScript, Json, DOM, Ajax, WebSocket, WebComponent, ...

Your web-site = 1 web-page = 1 APPLICATION

running on the browser

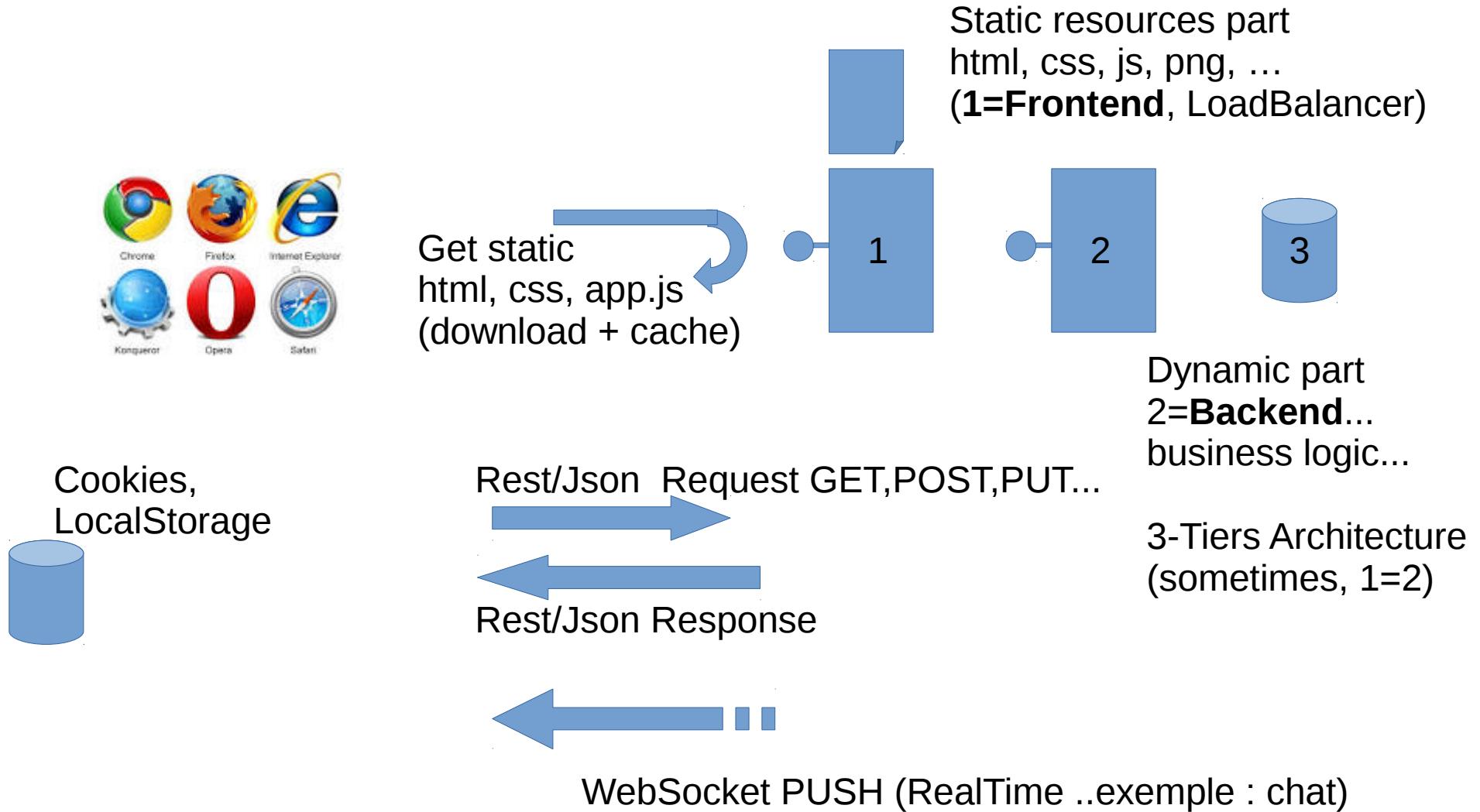
Using client-side CPU, GraphicCard,Cookies,WebStorage,..

You don't explicitly "install" an app,

you just see **1 "html/css" page + also downloaded 1 "app.js"**

Then you GET+POST Rest/Json data ... and RealTime WS

Web App = (static) Html + Rest API

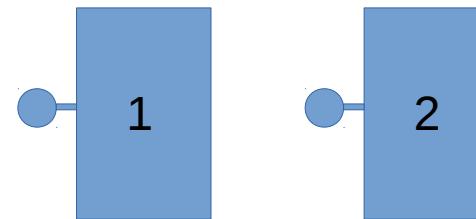


Springboot = for Backend Devs

De facto tools= nodejs gulp+npm+bower...
(in JS world ... also GWT, and others)



De facto tools (in JVM world)
springboot + maven + ...



Pure Web-Designer

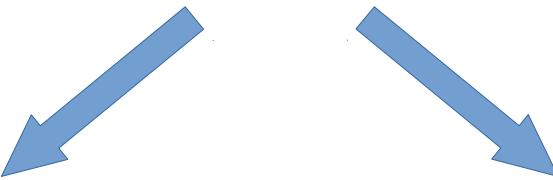
Front-end Developper

Back-end Developper

Full-Stack Developper

DBA, BigData-Analyst

Add REST/Json Web API



Using “Built-in”
spring-web-mvc

specific annotations
@RequestMapping
@RequestBody ...

Bind to JAX-RS
standard

standard annotations
@Path @GET
@ ...

Using Spring-web-mvc

@RestController

@RequestMapping

```
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

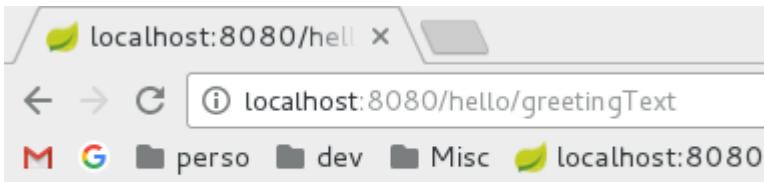
@RestController
@RequestMapping(path="/hello")
public class HelloRestController {

    public static class Greeting {
        public String msg;
        public Greeting(String msg) { this.msg = msg; }
    }

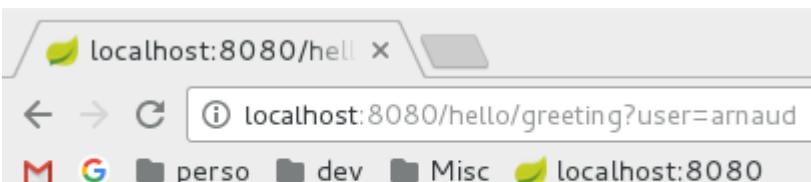
    @RequestMapping(path="/greeting")
    public Greeting greetingTo(
            @RequestParam(name="user", required=false) String user) {
        return new Greeting("Hello " + ((user != null)? user : "springboot user"));
    }

    @RequestMapping(path="/greetingText")
    public String greetingTextTo(
            @RequestParam(name="user", required=false) String user) {
        return "Hello " + ((user != null)? user : "springboot user");
    }
}
```

Run REST/Json Web API



Hello springboot user



Headers Preview Response Cookies Timing

General

Request URL: http://localhost:8080/hello/greeting?user=arnaud
Request Method: GET
Status Code: 200
Remote Address: [::1]:8080

Response Headers

view source
Content-Type: application/json; charset=UTF-8
Date: Thu, 17 Nov 2016 07:32:20 GMT
Transfer-Encoding: chunked

Request Headers

view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,fr;q=0.6
Connection: keep-alive
Cookie: remember-me=K1FMSTBXeVpEcUtPTXRDaDZMN0RBZz090mQyMlhINlhXaVNRFI
Host: localhost:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,

Query String Parameters

view source view URL encoded

user: arnaud

```
$ curl http://localhost:8080/hello/greeting?user=arnaud
{"msg": "Hello arnaud"}$
```

```
$ curl -v http://localhost:8080/hello/greeting?user=arnaud
* Hostname was NOT found in DNS cache
*   Trying ::1...
* Connected to localhost (::1) port 8080 (#0)
> GET /hello/greeting?user=arnaud HTTP/1.1
> User-Agent: curl/7.38.0
> Host: localhost:8080
> Accept: */*
>
< HTTP/1.1 200
< Content-Type: application/json; charset=UTF-8
< Transfer-Encoding: chunked
< Date: Thu, 17 Nov 2016 07:34:04 GMT
<
* Connection #0 to host localhost left intact
{"msg": "Hello arnaud"}$
```

AutoReload & LiveReload

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <optional>true</optional>
</dependency>
```

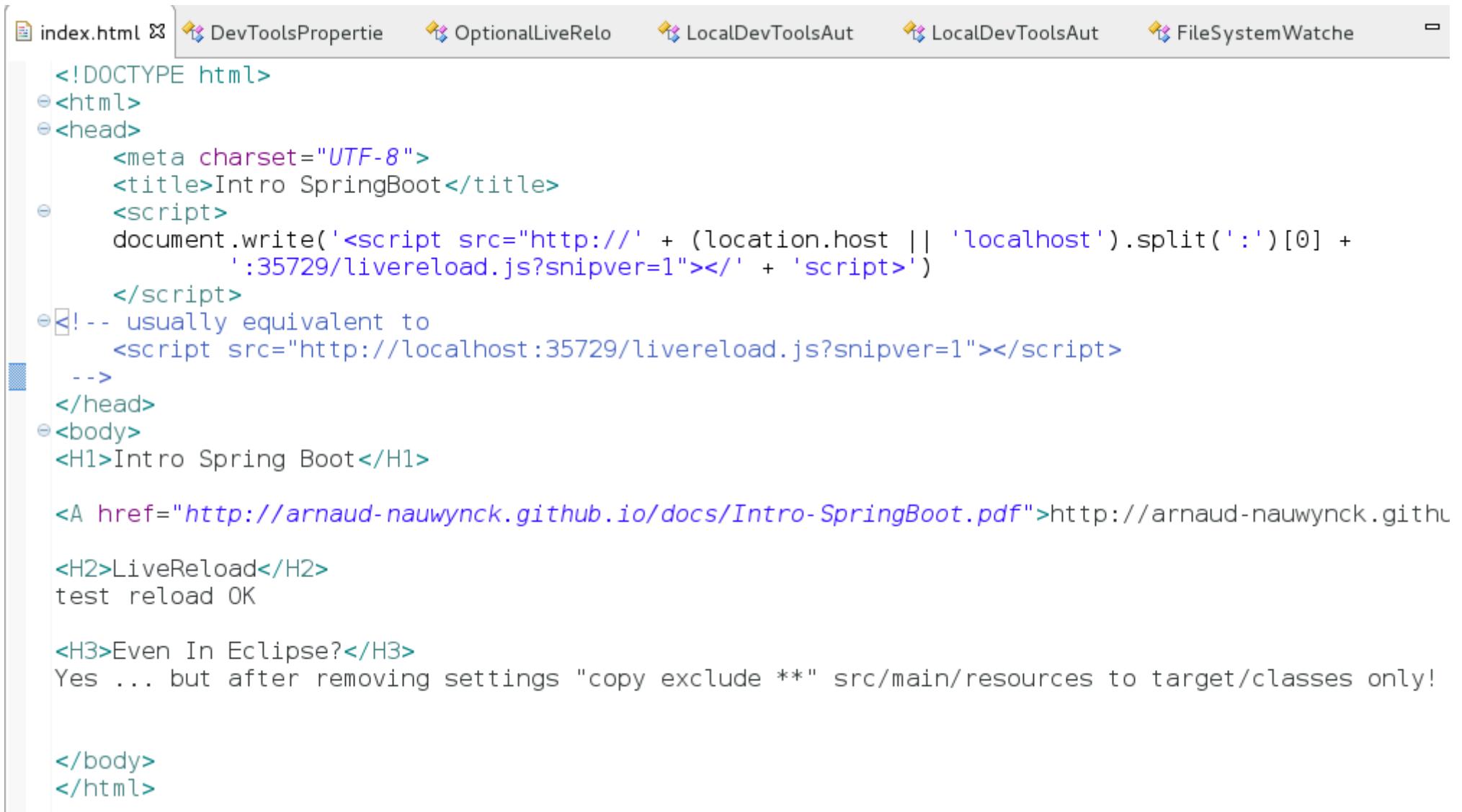
- AutoReload :
change your spring app
=> server auto restart (partial)

For fast developer experience
... prefer gulp ...
in particular if in Eclipse <=2016

- LiveReload:
change Html/Css/Typescript/...
=> client Web Browser auto refresh!!

```
2016-11-16 23:25:59.119 INFO 7916 --- [ restartedMain] oConfiguration$WelcomePageHandlerMapping : Adding welcome page: class path resource [static/]
2016-11-16 23:25:59.222 INFO 7916 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729
2016-11-16 23:25:59.324 INFO 7916 --- [ restartedMain] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2016-11-16 23:25:59.383 INFO 7916 --- [ restartedMain] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
2016-11-16 23:25:59.391 INFO 7916 --- [ restartedMain] com.example.DemoApplication : Started DemoApplication in 2.665 seconds (JVM runn
```

<script>... :35729/ livereload.js</script>



The screenshot shows a browser developer tools window with the Network tab selected. A single entry is listed: "livereload.js" from "http://localhost:35729". The file size is 0 bytes and it was received in 0ms. The status is "OK".

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Intro SpringBoot</title>
    <script>
        document.write('<script src="http://' + (location.host || 'localhost').split(':')[0] +
                      ':35729/livereload.js?snipver=1"></' + 'script>')
    </script>
    <!-- usually equivalent to
        <script src="http://localhost:35729/livereload.js?snipver=1"></script>
    -->
</head>
<body>
    <H1>Intro Spring Boot</H1>

    <A href="http://arnaud-nauwynck.github.io/docs/Intro-SpringBoot.pdf">http://arnaud-nauwynck.githu...
    <H2>LiveReload</H2>
    test reload OK

    <H3>Even In Eclipse?</H3>
    Yes ... but after removing settings "copy exclude **" src/main/resources to target/classes only!

</body>
</html>
```

Still Deploy with “java -jar”

```
$ ls -lh target/
total 14M
drwxr-xr-x 4 arnaud arnaud 4.0K Nov 16 21:45 classes
-rw-r--r-- 1 arnaud arnaud 14M Nov 16 21:45 demo-0.0.1-SNAPSHOT.jar
-rw-r--r-- 1 arnaud arnaud 3.3K Nov 16 21:45 demo-0.0.1-SNAPSHOT.jar.original
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:45 generated-sources
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:45 generated-test-sources
drwxr-xr-x 2 arnaud arnaud 4.0K Nov 16 21:45 maven-archiver
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:45 maven-status
drwxr-xr-x 3 arnaud arnaud 4.0K Nov 16 21:45 test-classes
$ 
$ java -jar target/demo-0.0.1-SNAPSHOT.jar
```

now 14M jar ...

java -jar ... = Web App !



```
2016-11-16 21:46:02.461  INFO 5415 --- [           main] com.example.DemoApplication          : Starting DemoApplication v0.0.1
n with PID 5415 (/mnt/a_1tera2/homeData/arnaud/perso/devPerso/my-github/test-snippets.github/test-springboot/target/demo-0.0.1-SNAP
d by arnaud in /mnt/a_1tera2/homeData/arnaud/perso/devPerso/my-github/test-snippets.github/test-springboot)
```

Dependencies Added for spring-boot-start-web

A screenshot of the IntelliJ IDEA interface showing the dependency tree for a project named "demo". A blue callout bubble on the left points to the "Tomcat embedded .jar" section of the tree.

The dependency tree is organized into several sections:

- Tomcat embedded .jar**: A blue-highlighted section containing:
 - spring-boot-starter-web : 1.4.2.RELEASE [compile]
 - spring-boot-starter : 1.4.2.RELEASE (omitted for conflict with 1.4.2.RELEASE) [compile]
 - spring-boot-starter-tomcat : 1.4.2.RELEASE [compile]
 - tomcat-embed-core : 8.5.6 [compile]
 - tomcat-embed-el : 8.5.6 [compile]
 - tomcat-embed-websocket : 8.5.6 [compile]
 - tomcat-embed-core : 8.5.6 (omitted for conflict with 8.5.6) [compile]
- hibernate-validator**: A section containing:
 - validation-api : 1.1.0.Final [compile]
 - jboss-logging : 3.3.0.Final (managed from 3.2.1.Final) [compile]
 - classmate : 1.3.3 (managed from 1.1.0) [compile]
- jackson-databind**: A section containing:
 - jackson-annotations : 2.8.4 (managed from 2.8.0) [compile]
 - jackson-core : 2.8.4 [compile]
- spring-web**: A section containing:
 - spring-aop : 4.3.4.RELEASE [compile]
 - spring-beans : 4.3.4.RELEASE [compile]
 - spring-context : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-core : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
- spring-webmvc**: A section containing:
 - spring-aop : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-beans : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-context : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
 - spring-core : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
- spring-expression**: A section containing:
 - spring-expression : 4.3.4.RELEASE [compile]
- spring-web**: A section containing:
 - spring-web : 4.3.4.RELEASE (omitted for conflict with 4.3.4.RELEASE) [compile]
- objenesis**: A section containing:
 - objenesis : 2.1 [test]
- slf4j-api**: A section containing:
 - slf4j-api : 1.7.21 [compile]
- snakeyaml**: A section containing:
 - snakeyaml : 1.17 [runtime]
- spring-aop**: A section containing:
 - spring-aop : 4.3.4.RELEASE [compile]
- spring-beans**: A section containing:
 - spring-beans : 4.3.4.RELEASE [compile]
- spring-boot**: A section containing:
 - spring-boot : 1.4.2.RELEASE [compile]
- spring-boot-autoconfigure**: A section containing:
 - spring-boot-autoconfigure : 1.4.2.RELEASE [compile]
- spring-boot-starter**: A section containing:
 - spring-boot-starter : 1.4.2.RELEASE [compile]
- spring-boot-starter-logging**: A section containing:
 - spring-boot-starter-logging : 1.4.2.RELEASE [compile]
- spring-boot-starter-test**: A section containing:
 - spring-boot-starter-test : 1.4.2.RELEASE [test]
- spring-boot-starter-tomcat**: A section containing:
 - spring-boot-starter-tomcat : 1.4.2.RELEASE [compile]
- spring-boot-starter-web**: A section containing:
 - spring-boot-starter-web : 1.4.2.RELEASE [compile]
- spring-boot-test**: A section containing:
 - spring-boot-test : 1.4.2.RELEASE [test]
- spring-boot-test-autoconfigure**: A section containing:
 - spring-boot-test-autoconfigure : 1.4.2.RELEASE [test]
- spring-context**: A section containing:
 - spring-context : 4.3.4.RELEASE [compile]
- spring-core**: A section containing:
 - spring-core : 4.3.4.RELEASE [compile]
- spring-expression**: A section containing:
 - spring-expression : 4.3.4.RELEASE [compile]
- spring-test**: A section containing:
 - spring-test : 4.3.4.RELEASE [test]
- spring-web**: A section containing:
 - spring-web : 4.3.4.RELEASE [compile]
- spring-webmvc**: A section containing:
 - spring-webmvc : 4.3.4.RELEASE [compile]
- tomcat-embed-core**: A section containing:
 - tomcat-embed-core : 8.5.6 [compile]
- tomcat-embed-el**: A section containing:
 - tomcat-embed-el : 8.5.6 [compile]
- tomcat-embed-websocket**: A section containing:
 - tomcat-embed-websocket : 8.5.6 [compile]
- validation-api**: A section containing:
 - validation-api : 1.1.0.Final [compile]

At the bottom of the window, there are tabs for Overview, Dependencies, Dependency Hierarchy, Effective POM, and pom.xml.

JAR > WAR

ClassLoader in springboot can zip nested jar

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

```
$ jar tf target/demo-0.0.1-SNAPSHOT.jar | grep '\.jar'
BOOT-INF/lib/spring-web-4.3.4.RELEASE.jar
BOOT-INF/lib/jul-to-slf4j-1.7.21.jar
BOOT-INF/lib/spring-boot-starter-tomcat-1.4.2.RELEASE.jar
BOOT-INF/lib/hibernate-validator-5.2.4.Final.jar
BOOT-INF/lib/tomcat-embed-el-8.5.6.jar
BOOT-INF/lib/spring-aop-4.3.4.RELEASE.jar
BOOT-INF/lib/tomcat-embed-websocket-8.5.6.jar
  NF/lib/snakeyaml-1.17.jar
  NF/lib/tomcat-embed-core-8.5.6.jar
  NF/lib/spring-beans-4.3.4.RELEASE.jar
  .... _NF/lib/spring-boot-starter-logging-1.4.2.RELEASE.jar
  BOOT-INF/lib/slf4j-api-1.7.21.jar
  BOOT-INF/lib/spring-webmvc-4.3.4.RELEASE.jar
  BOOT-INF/lib/spring-boot-starter-web-1.4.2.RELEASE.jar
  BOOT-INF/lib/spring-context-4.3.4.RELEASE.jar
  BOOT-INF/lib/spring-core-4.3.4.RELEASE.jar
  BOOT-INF/lib/jackson-databind-2.8.4.jar
  BOOT-INF/lib/spring-boot-starter-1.4.2.RELEASE.jar
  BOOT-INF/lib/spring-boot-autoconfigure-1.4.2.RELEASE.jar
  BOOT-INF/lib/logback-classic-1.1.7.jar
  BOOT-INF/lib/jackson-core-2.8.4.jar
  BOOT-INF/lib/spring-boot-1.4.2.RELEASE.jar
  BOOT-INF/lib/spring-expression-4.3.4.RELEASE.jar
  BOOT-INF/lib/logback-core-1.1.7.jar
  BOOT-INF/lib/validation-api-1.1.0.Final.jar
  BOOT-INF/lib/classmate-1.3.3.jar
  BOOT-INF/lib/jcl-over-slf4j-1.7.21.jar
  BOOT-INF/lib/log4j-over-slf4j-1.7.21.jar
  BOOT-INF/lib/jboss-logging-3.3.0.Final.jar
  BOOT-INF/lib/jackson-annotations-2.8.4.jar
$ □
```

RIP WebLogic / WebSphere / GlassFish / WildFly / ...

Comparatively ...

1 day to install the server

1 week to read JEE doc

1 month to read specific doc

1 month to struggle in /console & wlst

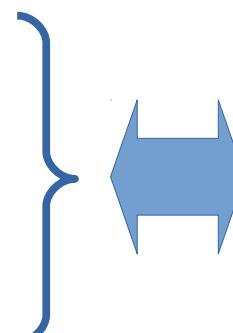
5 minutes to stop-restart a server

1 minute to rebuild

3 minutes to redeploy a war

(+ **5mn** : often restart anyway ..)

+ **1 minute** remote debugging



2 Seconds
to build / start / hot-restart

0 Value added

-10000 \$ on your bank account

either in Eclipse
Or java -jar

as Josh Long said in one of his talks
in the Spring IO
“it is better to make Jar, not War”.



Josh Long

the Spring Developer Advocate at Pivotal @starbuxman

Amazing Conferences
example at Devoxx



Home About Search Twitter Have an account? Log in

48.5K TWEETS **2,172** FOLLOWING **13.2K** FOLLOWERS **17.8K** LIKES **19** LISTS

[Follow](#)

Tweets **Tweets & replies** **Media**

Pinned Tweet

Josh Long (龙之春, जोश) @starbuxman · 27 Oct 2015

"Getting Started w/ **@SpringCloudOSS**"
"@SpringOne2GX 2015 w Dr. **@David_Syer**
and me (& appearance by Dr.
@SpringRod!)

Pivotal Getting Started with Spring Cloud
Recorded at SpringOne2GX 2015 Speakers: Dr. Dave
Syer, Phillip Webb

New to Twitter?
Sign up now to get your own
personalized timeline!

[Sign up](#)

You may also like · Refresh

Spring Boot @springboot

Phil Webb @philip_webb

Josh Long (龙之春, जोश) @starbuxman · 27 Oct 2015

Spring Advocate @Pivotal
spring.io/team/jlong; @david_syer's my
spirit animal; @pearson bit.ly/1HUCRWt;
@OReillyMedia oreil.ly/1CiSz85

Mission, San Francisco, CA

joshlong.com

Joined April 2007

#MakeJarNotWar @starbuxman

<https://twitter.com/hashtag/makejarnotwar>

← → C Twitter, Inc. [US] | <https://twitter.com/hashtag/makejarnotwar>

M G perso dev Misc

Home About #makejarnotwar

#makejarnotwar

Top | Latest | Accounts | Photos | Videos | More options ▾

New to Twitter?
Sign up now to get your own personalized timeline!

Sign up

Dimitri Hautot @D_H_ · 31 Dec 2015
Hey @starbuxman, in 2016, more than ever, #MakeJarNotWar !!! Happy New Year !!!

1 1 2 ...

Callum Watson @thecallumwatson · Mar 10
@starbuxman great presentation at @jpmorgan tech symposium today!#makejarnotwar

2 7 5 ...

Pulkit Kumar @pulkitkumar90 · Jul 20
Great talk by @starbuxman : bit.ly/2a8t3xd . Java is awesome.
#MakeJarNotWar

1 1 1 ...

vaadin @vaadin · May 19

About 2,040 results (0.35 seconds)

2000 "make jar not war"

Introduction to Spring Boot » blogblog.mimacom.com/introduction-to-spring-boot/ ▾

Sep 30, 2015 - ... WAR file and deploy it in any server of your choice, but as Josh Long says in the Spring IO "it is better to **make Jar, not War**".

Make Jar, Not War - GitHub<https://github.com/making/make-jar-not-war> ▾

Make Jar, Not War. Contribute to **make-jar-not-war** development by creating pull requests. Page 1 of 1

SSL secure Help?

Products 1 to 8 of 8

Images for "make jar not war"

(0 Articles : \$ 0.00)

Page 1 of 1

1

Make .jar, not .war geeky t-shirt

Men's T-Shirt
Size

\$17.99
\$13.99

Make .jar, not .war geeky t-shirt

Men's T-Shirt
Size

\$19.49
\$13.99

Make .jar, not .war geeky t-shirt

Men's T-Shirt by American Apparel
Size

\$26.99
\$21.99

More images for "make jar not war"**Make JAR not WAR ! - YouTube**<https://www.youtube.com/watch?v=O8JljPRD3OE>

May 22, 2016 - Uploaded by Pierre FEVRIER

Présentation en 30 min des principales fonctionnalités
... MVP Java 10,303 views ...

Make .jar, not .war geeky t-shirt

Men's Ringer T-Shirt
Size

Make .jar, not .war geeky t-shirt

Men's Hoodie
Size

Make .jar, not .war geeky t-shirt

Women's T-Shirt
Size

App Server Embedded Server > Deployed War



Your classes

Embedded tomcat

Your JAR packaging also contains embedded tomcat



Your paid \$\$\$ webserver

Your WAR + classes

Your WAR is deployed into a WebServer

Historical Reasons 1 Server = 1 JVM – Several Apps

1 Server = 1 JVM process
... serves several Apps

JSP Web Portal calling other JSP ?

Web \$\$\$ Expensive => use 1 server for many apps

Limited RAM (1-8 G) ...
jvm overhead = 100M
=> share RAM on workstations

Problems:
bad isolation ...
OutOfMemory Error App1 => also crash App2,App3...
Redeploy / Restart App1 => stop all App2,App3...



“Hello Web World” ok
Only yet another Web Framework ?

What's Next ?

Springboot Almost contains a Jdbc Database App (add 4 lines)...

```
<!-- choose your poison (Postgresql, MySql, Oracle, H2, ...) -->
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```

spring-jdbc

```
import java.sql.PreparedStatement;
import java.sql.ResultSet;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

@Component
public class JdbcBatchCommand implements CommandLineRunner {

    @Autowired
    protected JdbcTemplate jdbcTemplate;

    @Override
    public void run(String... args) throws Exception {
        jdbcTemplate.execute("select 1");

        String msg = jdbcTemplate.execute("select 'Hello' as msg", (PreparedStatement pstmt) -> {
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return rs.getString("msg");
            }
            return null;
        });
        assert "Hello".equals(msg);

    }
}
```

Run Jdbc App

The screenshot shows a Java application running in a debugger. The code is a `CommandLineRunner` implementation that performs a database query and asserts its result.

```
package com.example;

import java.sql.PreparedStatement;
import java.sql.ResultSet;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Component;

@Component
public class JdbcBatchCommand implements CommandLineRunner {

    @Autowired
    protected JdbcTemplate jdbcTemplate;

    @Override
    public void run(String... args) throws Exception {
        jdbcTemplate.execute("select 1");

        String msg = jdbcTemplate.execute("select 'Hello' as msg", (PreparedStatement pstmt) -> {
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return rs.getString("msg");
            }
            return null;
        });
        assertEquals("Hello", msg);
    }
}
```

The code uses `JdbcTemplate` to execute SQL statements. It first executes a simple query ("select 1"). Then it executes a query that returns a single string value ('Hello'). Finally, it asserts that the retrieved string equals "Hello".

The IDE interface shows the current thread is suspended at line 28 in `JdbcBatchCommand`. The `Breakpoints` tool window shows a breakpoint is set on line 28 of `JdbcBatchCommand.java`.

Springboot built-in supports JTA @Transactional

```
import org.springframework.transaction.annotation.Transactional;

@Component
@Transactional
public class JdbcBatchCommand implements CommandLineRunner {

    @Autowired
    protected JdbcTemplate jdbcTemplate;

    @Override
    // @Transactional // repeat on each method is useless, say once on class
    public void run(String... args) throws Exception {
        jdbcTemplate.execute("select 1");

        String msg = jdbcTemplate.execute("select 'Hello' as msg", (PreparedStatement pstmt) -> {
            ResultSet rs = pstmt.executeQuery();
            if (rs.next()) {
                return rs.getString("msg");
            }
            return null;
        });
        assert "Hello".equals(msg);
    } // after this "}" line, spring will commit (or rollback ACID Transaction)
}
```

@Transactional JUST Works

The screenshot shows a Java application running in a debugger. The stack trace indicates that the application has stopped at line 30 of the `JdbcBatchCommand` class. The source code for `JdbcBatchCommand.java` is visible below, showing a method that returns a string from a database query and includes an assertion.

Stack Trace:

- Thread [restartedMain] (Suspended (breakpoint at line 30 in JdbcBatchCommand))
- JdbcBatchCommand.run(String...) line: 30
- JdbcBatchCommand\$\$FastClassBySpringCGLIB\$\$749c813d.invoke(int, Object, Object[]) line: not available
- MethodProxy.invoke(Object, Object[]) line: 204
- CglibAopProxy\$CglibMethodInvocation.invokeJoinpoint() line: 720
- CglibAopProxy\$CglibMethodInvocation(ReflectiveMethodInvocation).proceed() line: 157
- TransactionInterceptor\$1.proceedWithInvocation() line: 99
- TransactionInterceptor(TransactionAspectSupport).invokeWithinTransaction(Method, Class<?>, InvocationCallback) line: 282
- TransactionInterceptor.invoke(MethodInvocation) line: 96
- CglibAopProxy\$CglibMethodInvocation(ReflectiveMethodInvocation).proceed() line: 179
- CglibAopProxy\$DynamicAdvisedInterceptor.intercept(Object, Method, Object[], MethodProxy) line: 655
- JdbcBatchCommand\$\$EnhancerBySpringCGLIB\$\$ce3bc846.run(String...) line: not available
- SpringApplication.callRunner(CommandLineRunner, ApplicationArguments) line: 800

JdbcBatchCommand.java:

```
        ResultSet rs = pstmt.executeQuery();
        if (rs.next()) {
            return rs.getString("msg");
        }
        return null;
    });
    assert "Hello".equals(msg);
} // after this "}" line, spring will commit (or rollback ACID Transaction)
```

Almost a Rich JPA-Data Db App (add 4 lines)...

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

<!-- choose your poison (Postgresql, MySql, Oracle, H2, ...) -->
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```

Code using JPA CRUD findOne,findBy,save()...

```
@Component
@Transactional
public class JPRepositoryBatchCommand implements CommandLineRunner {

    private static final Logger LOG = LoggerFactory.getLogger(JPRepositoryBatchCommand.class);

    @Autowired
    protected EmployeeRepository employeeDAO;

    @Override
    public void run(String... args) throws Exception {
        Employee empById1 = employeeDAO.findOne(1);
        if (empById1 != null) LOG.info("Hello employee #1 : " + empById1.getFirstName() + " " +
Employee empJohn = employeeDAO.findOneByEmail("john.smith@gmail.com");
        if (empJohn == null) {
            empJohn = new Employee();
            empJohn.setEmail("john.smith@gmail.com");
            employeeDAO.save(empJohn);
        }
    }
}
```

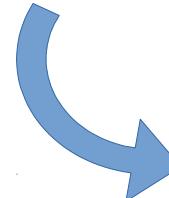
Repository ... autoconfigured at runtime

EmployeeRepository.java

```
package com.example.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
  
    Employee findOneByEmail(String email);  
}
```

By naming convention .. Equivalent to:
“Select * from EMPLOYEE where email=?”

Extends JpaRepository
built-in CRUD ...
findAll, by Page, save, delete...
(no update, use Setters)



```
com.example.repository ~ com.example.repository.EmployeeRepository.java  
EmployeeRepository ~ com.example.repository  
+ findOneByEmail(String) : Employee ~ com.example.repository.EmployeeRepository  
+ findAll() : List<T> ~ org.springframework.data.jpa.repository.JpaRepository  
+ findAll(Sort) : List<T> ~ org.springframework.data.jpa.repository.JpaRepository  
+ findAll(Iterable<ID>) : List<T> ~ org.springframework.data.jpa.repository.JpaRepository  
+ save(Iterable<S>) <S extends T> : List<S> ~ org.springframework.data.jpa.repository.JpaRepository  
+ flush() : void ~ org.springframework.data.jpa.repository.JpaRepository  
+ saveAndFlush(S) <S extends T> : S ~ org.springframework.data.jpa.repository.JpaRepository  
+ deleteInBatch(Iterable<T>) : void ~ org.springframework.data.jpa.repository.JpaRepository  
+ deleteAllInBatch() : void ~ org.springframework.data.jpa.repository.JpaRepository  
+ getOne(ID) : T ~ org.springframework.data.jpa.repository.JpaRepository  
+ findAll(Example<S>) <S extends T> : List<S> ~ org.springframework.data.jpa.repository.JpaRepository  
+ findAll(Example<S>, Sort) <S extends T> : List<S> ~ org.springframework.data.jpa.repository.JpaRepository  
+ findAll(Sort) : Iterable<T> ~ org.springframework.data.repository.PagingAndSortingRepository  
+ findAll(Pageable) : Page<T> ~ org.springframework.data.repository.PagingAndSortingRepository  
+ save(S) <S extends T> : S ~ org.springframework.data.repository.CrudRepository  
+ save(Iterable<S>) <S extends T> : Iterable<S> ~ org.springframework.data.repository.CrudRepository  
+ findOne(ID) : T ~ org.springframework.data.repository.CrudRepository  
+ exists(ID) : boolean ~ org.springframework.data.repository.CrudRepository  
+ findAll() : Iterable<T> ~ org.springframework.data.repository.CrudRepository  
+ findAll(Iterable<ID>) : Iterable<T> ~ org.springframework.data.repository.CrudRepository  
+ count() : long ~ org.springframework.data.repository.CrudRepository  
+ delete(ID) : void ~ org.springframework.data.repository.CrudRepository  
+ delete(T) : void ~ org.springframework.data.repository.CrudRepository  
+ delete(Iterable<T>) : void ~ org.springframework.data.repository.CrudRepository  
+ deleteAll() : void ~ org.springframework.data.repository.CrudRepository  
+ findOne(Example<S>) <S extends T> : S ~ org.springframework.data.repository.query.QueryByExampleExecutor  
+ findAll(Example<S>) <S extends T> : Iterable<S> ~ org.springframework.data.repository.query.QueryByExampleExecutor  
+ findAll(Example<S>, Sort) <S extends T> : Iterable<S> ~ org.springframework.data.repository.query.QueryByExampleExecutor
```

@Entity, @Id (optionnal @Version) @OneToMany, @ManyToOne

```
J Employee.java ✘

package com.example.domain;

+import java.util.ArrayList;

@Entity
public class Employee {

    @Id
    private int id;

    @Version
    private int version;

    private String firstName, lastName, email, address;

    private Date birthDate;

    @ManyToOne
    private Department department;

    @OneToMany
    private List<UserProject> projects = new ArrayList<>();
```

```
J Department.java ✘

package com.example.domain;

+import javax.persistence.Entity;

@Entity
public class Department {

    @Id
    private int id;

    private String name;
```



Database table “EMPLOYEE”
(id (PK), version, first_name, last_name,
department_id (FK department.id)



Database table “DEPARTMENT”
(id (PK), name)

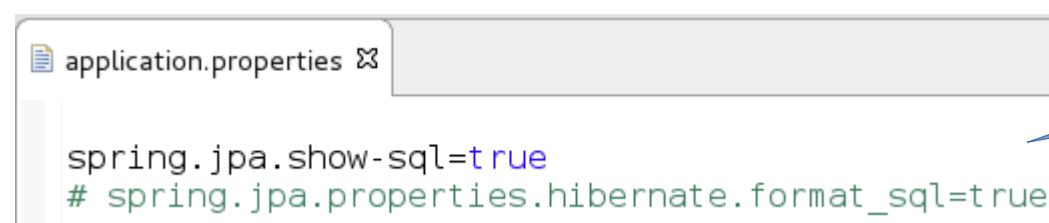
Springboot @JPA configuration

```
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
//{@EnableJpaRepositories(basePackages="com.example.repository")}  
//{@EntityScan(basePackages="com.example.domain")}  
public class DemoJPAConfig {  
}
```

springboot dark magic
not even 1 line .. all optional !!!

```
import org.springframework.boot.autoconfigure.domain.EntityScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
  
@Configuration  
@EnableJpaRepositories(basePackages="com.example.repository")  
@EntityScan(basePackages="com.example.domain")  
public class DemoJPAConfig {  
}
```

Useless equivalent
2 implicit lines



application.properties

```
spring.jpa.show-sql=true  
# spring.jpa.properties.hibernate.format_sql=true
```

Optional
for debug (see next)

Springboot hibernate... “JUST work”



```
2016-11-17 22:42:01.527 INFO 9504 --- [ restartedMain] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8080 (http)
Hibernate: select employee0_.id as id1_10_, employee0_.address as address2_10_, employee0_.birth_date as birth_da3_10_, employee0_.department_id as department10_
2016-11-17 22:48:26.530 INFO 9504 --- [ restartedMain] o.h.h.i.QueryTranslatorFactoryInitiator : HHH000397: Using ASTQueryTranslatorFactory
Hibernate: select employee0_.id as id1_1_, employee0_.address as address2_1_, employee0_.birth_date as birth_da3_1_, employee0_.department_id as department1_1_, employee0_
Hibernate: insert into employee (address, birth_date, department_id, email, first_name, last_name, version, id) values (?, ?, ?, ?, ?, ?, ?, ?)
2016-11-17 22:48:28.194 INFO 9504 --- [ restartedMain] com.example.DemoApplication           : Started DemoApplication in 391.821 seconds (JVM running for 392.191)
```

CRUD ...
select * from EMPLOYEE where ...
insert into EMPLOYEE (...) values (...)

When/How/Where are my “create Table ()” ???

Tables are created/updated at startup

```
2016-11-17 22:41:59.753 INFO 9504 --- [ restartedMain] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
2016-11-17 22:41:59.851 INFO 9504 --- [ restartedMain] org.hibernate.dialect.Dialect   : HHH000400: Using dialect: org.hibernate.dialect.H2Dialect
2016-11-17 22:42:00.269 INFO 9504 --- [ restartedMain] org.hibernate.tool.hbm2ddl.SchemaExport : HHH000227: Running hbm2ddl schema export
Hibernate: drop table department if exists
Hibernate: drop table employee if exists
Hibernate: drop table employee_projects if exists
Hibernate: drop table user_project if exists
Hibernate: create table department (id integer not null, name varchar(255), primary key (id))
Hibernate: create table employee (id integer not null, address varchar(255), birth_date timestamp, email varchar(255), first_name varchar(255), last_name varchar(255),
Hibernate: create table employee_projects (employee_id integer not null, projects_id integer not null)
Hibernate: create table user_project (id integer not null, employee_id integer, primary key (id))
Hibernate: alter table employee_projects add constraint UK_4jypfcavfedhsivky8co9wvqa unique (projects_id)
Hibernate: alter table employee add constraint FKbejtwvg9bxus2mffsm3swj3u9 foreign key (department_id) references department
Hibernate: alter table employee_projects add constraint FK25ggdalg559udqdvhwu3p4j3 foreign key (projects_id) references user_project
Hibernate: alter table employee_projects add constraint FK97jl81fsrbblkqfoqwg2o7yps foreign key (employee_id) references employee
Hibernate: alter table user_project add constraint FKmlfr43vjmqqlnaleuarwhhveq foreign key (employee_id) references employee
2016-11-17 22:42:00.286 INFO 9504 --- [ restartedMain] org.hibernate.tool.hbm2ddl.SchemaExport : HHH000230: Schema export complete
2016-11-17 22:42:00.318 INFO 9504 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
```

Detected H2 Database SQL language

Also create PK/FK indexes

How??

The screenshot shows a Java debugger interface with a stack trace. The stack trace starts with a call from `SessionImpl$IdentifierLoadAccessImpl<T>.load(Serializable)` at line 2687. It then traces back through several methods in `SessionImpl`, `EntityManagerImpl`, and `NativeMethodAccessorImpl`. The stack trace continues through `DelegatingMethodAccessorImpl`, `Method.invoke`, and `SharedEntityManagerCreator$SharedEntityManagerInvocationHandler.invoke` before reaching a call to `$Proxy81.find` at line 298. Finally, it traces back to `SimpleJpaRepository<T, ID>.findOne(ID)` at line 239, which is annotated with `@Override` and `@SuppressWarnings("unchecked")`. The code editor window below shows the implementation of the `load` method in `SessionImpl.class`.

Debug

- Thread [main] (Running)
- Thread [Thread-0] (Running)
- Thread [restartedMain] (Suspended)

```
SessionImpl$IdentifierLoadAccessImpl<T>.load(Serializable) line: 2687
SessionImpl.get(Class<T>, Serializable) line: 975
EntityManagerImpl(AbstractEntityManagerImpl).find(Class<A>, Object, LockModeType, Map<String, Object>) line: 1075
EntityManagerImpl(AbstractEntityManagerImpl).find(Class<T>, Object, Map<String, Object>) line: 1039
NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native method]
NativeMethodAccessorImpl.invoke(Object, Object[]) line: 62
DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: 43
Method.invoke(Object, Object...) line: 483
SharedEntityManagerCreator$SharedEntityManagerInvocationHandler.invoke(Object, Method, Object[]) line: 298
$Proxy81.find(Class, Object, Map) line: not available
SimpleJpaRepository<T, ID>.findOne(ID) line: 239
NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native method]
NativeMethodAccessorImpl.invoke(Object, Object[]) line: 62
```

JPARepositoryBatchCommand.java SessionImpl.class

```
@Override
@SuppressWarnings("unchecked")
public final T load(Serializable id) {
    if ( this.lockOptions != null ) {
        LoadEvent event = new LoadEvent( id, entityPersister.getEntityName(), lock(
            fireLoad( event, LoadEventListener.GET ) );
        return (T) event.getResult();
    }
}
```

Which call JPA..
→ Hibernate...
→ JDBC

Call springboot-data
JpaRepository

You code
calls a generated
Proxy..

This was “only” 10% of springboot and spring-*

SPRING INITIALIZR bootstrap your application now

Generate a with Spring Boot

Project Metadata

Artifact coordinates

Group

Artifact

Dependencies

Add Spring Boot Starters and dependencies to your application

Search for dependencies

Selected Dependencies

5% Web 5% JPA

Generate Project alt + ↵

Don't know what to look for? Want more options? [Switch to the full version.](#)

90% Remains ... & More in progress

Springboot-* page1/5

Generate Project alt + w

Core

- **Security**
Secure your application via spring-security
- **AOP**
Aspect-oriented programming including spring-aop and AspectJ
- **Atomikos (JTA)**
JTA distributed transactions via Atomikos
- **Bitronix (JTA)**
JTA distributed transactions via Bitronix
- **Narayana (JTA)**
JTA distributed transactions via Narayana
- **Cache**
Spring's Cache abstraction
- **DevTools**
Spring Boot Development Tools
- **Configuration Processor**
Generate metadata for your custom configuration keys
- **Validation**
JSR-303 validation infrastructure (already included with web)
- **Session**
API and implementations for managing a user's session information
- **Retry**
Provide declarative retry support via spring-retry
- **Lombok**
Java annotation library which helps to reduce boilerplate code and code faster

Web

- **Web**
Full-stack web development with Tomcat and Spring MVC
- **Websocket**
Websocket development with SockJS and STOMP
- **Web Services**
Contract-first SOAP service development with Spring Web Services
- **Jersey (JAX-RS)**
RESTful Web Services framework
- **Ratpack**
Spring Boot integration for the Ratpack framework
- **Vaadin**
Vaadin java web application framework
- **Rest Repositories**
Exposing Spring Data repositories over REST via spring-data-rest-webmvc
- **HATEOAS**
HATEOAS-based RESTful services
- **Rest Repositories HAL Browser**
Browsing Spring Data REST repositories in your browser
- **Mobile**
Simplify the development of mobile web applications with spring-mobile
- **REST Docs**
Document RESTful services by combining hand-written and auto-generated documentation

Template Engines

- **Freemarker**
FreeMarker templating engine
- **Velocity**
Velocity templating engine
- **Groovy Templates**
Groovy templating engine
- **Thymeleaf**
Thymeleaf templating engine, including integration with Spring
- **Mustache**
Mustache templating engine

Springboot-* page 2/5

SQL

- **JPA**
Java Persistence API including spring-data-jpa, spring-orm and Hibernate
- **JOOQ**
Persistence support using Java Object Oriented Querying
- **MyBatis**
Persistence support using MyBatis
- **JDBC**
JDBC databases
- **H2**
H2 database (with embedded support)
- **HSQLDB**
HSQLDB database (with embedded support)
- **Apache Derby**
Apache Derby database (with embedded support)
- **MySQL**
MySQL JDBC driver
- **PostgreSQL**
PostgreSQL JDBC driver
- **Flyway**
Flyway Database Migrations library
- **Liquibase**
Liquibase Database Migrations library

NoSQL

- **MongoDB**
MongoDB NoSQL Database, including spring-data-mongodb
- **Cassandra**
Cassandra NoSQL Database, including spring-data-cassandra
- **Couchbase**
Couchbase NoSQL database, including spring-data-couchbase
- **Neo4j**
Neo4j NoSQL graph database, including spring-data-neo4j
- **Redis**
REDIS key-value data store, including spring-redis
- **Gemfire**
GemFire distributed data store including spring-data-gemfire
- **Solr**
Apache Solr search platform, including spring-data-solr
- **Elasticsearch**
Elasticsearch search and analytics engine including spring-data-elasticsearch

NoSQL Databases:

Key-Values,
Columns,
Graphs
FullText

Cloud Core

- **Cloud Connectors**
Simplifies connecting to services in cloud platforms, including spring-cloud-connector and spring-cloud-cloudfoundry-connector
- **Cloud Bootstrap**
spring-cloud-context (e.g. Bootstrap context and @RefreshScope)
- **Cloud Security**
Secure load balancing and routing with spring-cloud-security
- **Cloud OAuth2**
OAuth2 and distributed application patterns with spring-cloud-security
- **Cloud Task**
Task result tracking along with integration with batch and streams

Springboot-* page 3/5

PostgreSQL JDBC driver

- Flyway

Flyway Database Migrations library

- Liquibase

Liquibase Database Migrations library

Cloud Core

- Cloud Connectors

Simplifies connecting to services in cloud platforms, including spring-cloud-connector and spring-cloud-cloudfoundry-connector

- Cloud Bootstrap

spring-cloud-context (e.g. Bootstrap context and @RefreshScope)

- Cloud Security

Secure load balancing and routing with spring-cloud-security

- Cloud OAuth2

OAuth2 and distributed application patterns with spring-cloud-security

- Cloud Task

Task result tracking along with integration with batch and streams

Cloud Config

- Config Client

spring-cloud-config Client

- Config Server

Central management for configuration via a git or svn backend

- Zookeeper Configuration

Configuration management with Zookeeper and spring-cloud-zookeeper-config

- Consul Configuration

Configuration management with Hashicorp Consul

Cloud Discovery

- Eureka Discovery

Service discovery using spring-cloud-netflix and Eureka

- Eureka Server

spring-cloud-netflix Eureka Server

- Zookeeper Discovery

Service discovery with Zookeeper and spring-cloud-zookeeper-discovery

- Cloud Foundry Discovery

Service discovery with Cloud Foundry

- Consul Discovery

Service discovery with Hashicorp Consul

Cloud Routing

- Zuul

Intelligent and programmable routing with spring-cloud-netflix Zuul

- Ribbon

Client side load balancing with spring-cloud-netflix and Ribbon

- Feign

Declarative REST clients with spring-cloud-netflix Feign

Cloud Circuit Breaker

- Hystrix

Circuit breaker with spring-cloud-netflix Hystrix

- Hystrix Dashboard

Circuit breaker dashboard with spring-cloud-netflix Hystrix

- Turbine

Circuit breaker metric aggregation using spring-cloud-netflix with Turbine and server-sent events

- Turbine AMQP

Circuit breaker metric aggregation using spring-cloud-netflix with Turbine and AMQP

- Turbine Stream

Circuit breaker metric aggregation using spring-cloud-netflix with Turbine and Spring Cloud Stream (choose a specific Stream binder implementation to complement this)

Cloud
=

For DataCenters
(10 000 PCs..)

discovery+failover
+load-balancer+

...

Springboot-* page 4/5

Cloud Tracing

- **Sleuth**
Distributed tracing via logs with spring-cloud-sleuth
- **Zipkin Client**
Distributed tracing with an existing Zipkin installation and spring-cloud-sleuth-zipkin. Alternatively, consider Sleuth Stream.
- **Sleuth Stream**
Marshals Spring Cloud Sleuth Spans over a Spring Cloud Stream binder
- **Zipkin Stream**
Consumes span data in messages from Spring Cloud Sleuth Stream and writes them to a Zipkin store
- **Zipkin UI**
add the Zipkin UI module to the Zipkin server to get a Zipkin service that accepts Spans and provides visualization
- **Zipkin Server**
Consumes span data over HTTP and writes them to a span store

Cloud Data Flow

- **Local Data Flow Server**
Local Data Flow Server implementation
- **Data Flow Shell**
Data Flow Shell

Cloud Contract

- **Cloud Contract Verifier**
Test dependencies required for autogenerated tests
- **Cloud Contract Stub Runner**
Stub Runner for HTTP/Messaging based communication
- **Cloud Contract WireMock**
Test dependencies required for the WireMock HTTP server

Cloud Messaging

- **Cloud Bus AMQP**
A simple control bus with AMQP and spring-cloud-bus-amqp
- **Cloud Bus Kafka**
A simple control bus with Kafka and spring-cloud-bus
- **Stream Rabbit**
Messaging microservices with RabbitMQ
- **Stream Kafka**
Messaging microservices with Kafka

Asynchronous
Messages Bus

Cloud AWS

- **AWS Core**
AWS native services from spring-cloud-aws
- **AWS JDBC**
Relational databases on AWS with RDS and spring-cloud-aws-jdbc
- **AWS Messaging**
Messaging on AWS with SQS and spring-cloud-aws-messaging

Cloud Cluster

- **Cluster Redis**
Leadership election and global state with Redis and spring-cloud-cluster-redis
- **Cluster Zookeeper**
Leadership election and global state with Zookeeper and spring-cloud-cluster-zookeeper
- **Cluster Hazelcast**
Leadership election and global state with Hazelcast and spring-cloud-cluster-hazelcast
- **Cluster Etcd**
Leadership election and global state with Etcd and spring-cloud-cluster-etcd

Pivotal Cloud Foundry

- **Config Client (PCF)**
Config client on Pivotal Cloud Foundry
- **Service Registry (PCF)**
Eureka service discovery on Pivotal Cloud Foundry
- **Circuit Breaker (PCF)**
Hystrix circuit breaker on Pivotal Cloud Foundry

Springboot-* page 5/5

Social

- Facebook
spring-social-facebook
- LinkedIn
spring-social-linkedin
- Twitter
spring-social-twitter

Ops

- Actuator
Production ready features to help you monitor and manage your application
- Actuator Docs
API documentation for the Actuator endpoints
- Remote Shell
CRaSH shell integration

Monitoring
Statistics,
Management

I/O

- Batch
Spring Batch including HSQLDB database support
- Integration
Common spring-integration modules
- Activiti
Activiti BPMN workflow engine
- Apache Camel
Integration using Apache Camel
- JMS (ActiveMQ)
Java Message Service API via Apache ActiveMQ
- JMS (Artemis)
Java Message Service API via Apache Artemis
- JMS (HornetQ)
Java Message Service API via HornetQ
- AMQP
Advanced Message Queuing Protocol via spring-rabbit
- Mail
javax.mail

Input-Processing-Output
for files/messages/events

Mail

Experimental

- Reactive Web
Reactive web development with Apache Tomcat and Spring Reactive (experimental)

Springboot : Lot of Cloud ...

Because it is used by HUUUGE companies

to run their business ≈ 

devoxx springboot josh long



DataCenters

- Alibaba
- Amazon
- Netflix
- Linked-In
- Twitter
- ...



Play This Video !!!

Getting started with Spring Cloud by Josh Long



Devoxx

Subscribe

11,871

15,620 views

Add to

Share

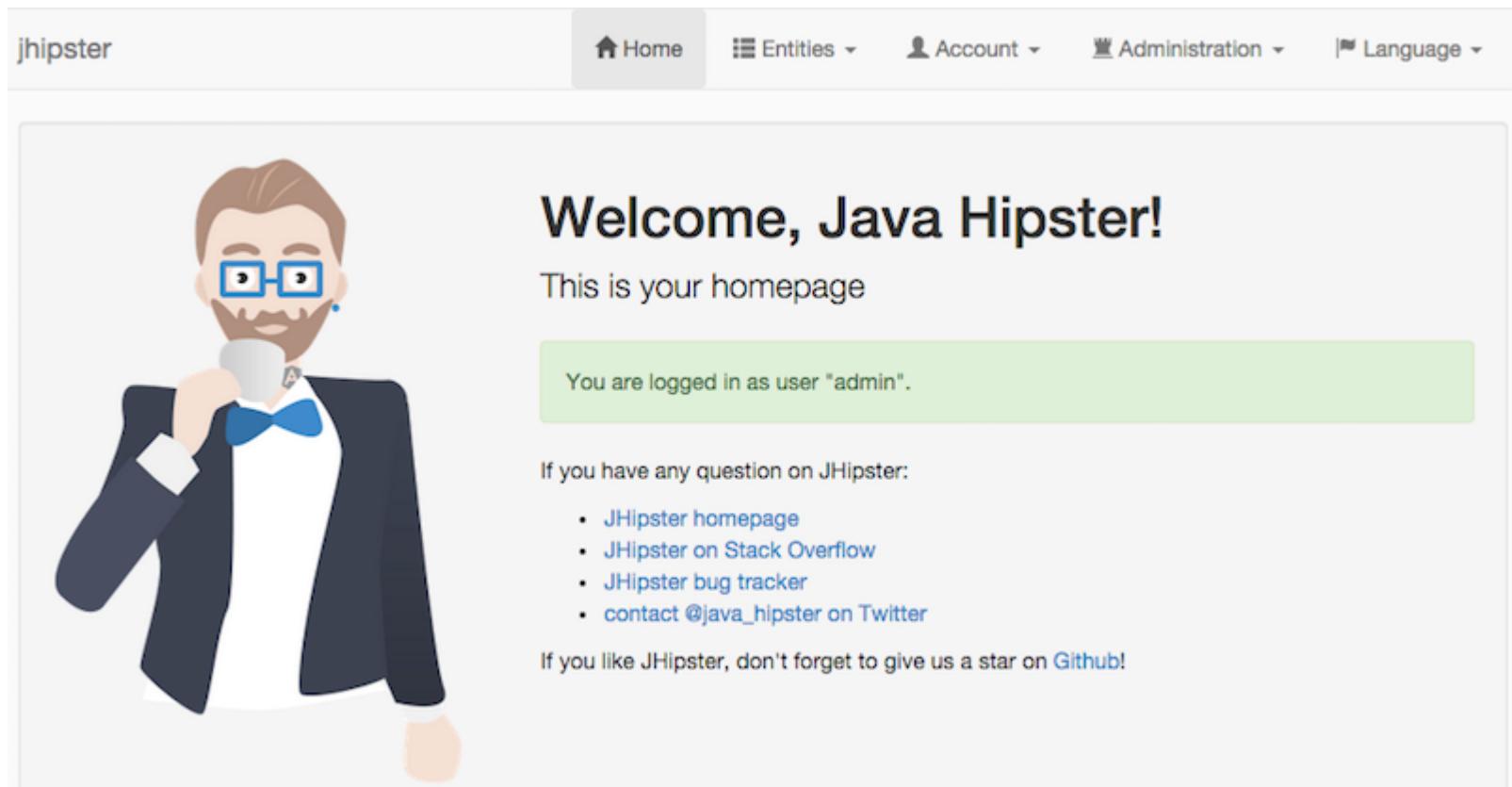
More

158

3

AngularJS + Springboot

In Jhipster ... you have 100% springboot on server-side
... with Code Generator
And also Hipe code on client-side : Html / Css + AngularJS + ..

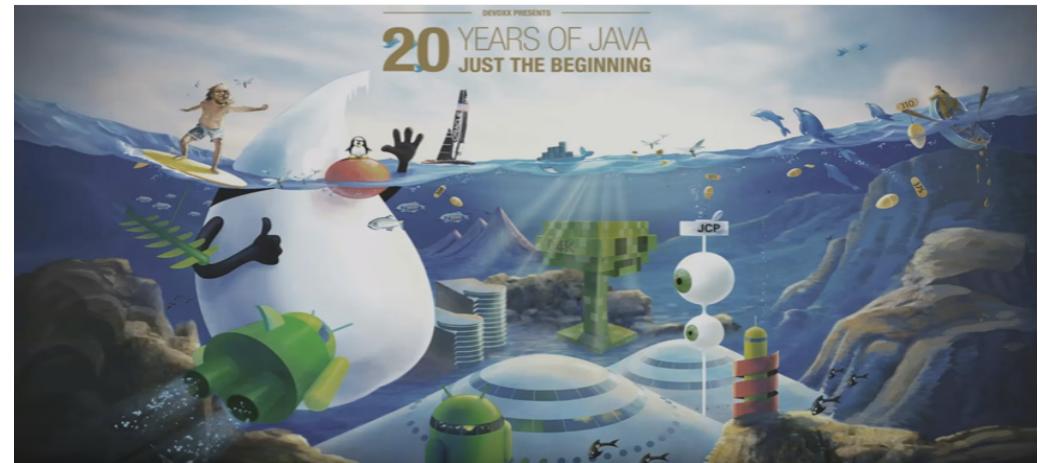


The screenshot shows the JHipster application's homepage. At the top, there is a navigation bar with the brand name "jhipster" on the left and links for "Home", "Entities", "Account", "Administration", and "Language" on the right. The main content area features a large, friendly cartoon character of a man with a beard and glasses, wearing a suit and bow tie, holding a microphone. To the right of the character, the text "Welcome, Java Hipster!" is displayed in a large, bold font. Below this, a message says "This is your homepage". A green callout box contains the text "You are logged in as user 'admin'.". Further down, there is a section titled "If you have any question on JHipster:" followed by a bulleted list of links: "JHipster homepage", "JHipster on Stack Overflow", "JHipster bug tracker", and "contact @java_hipster on Twitter". At the bottom, a final message encourages users to "If you like JHipster, don't forget to give us a star on [Github](#)!".

Java is Hipe

Make Jar nor WAR – NO PHP NO NodeJS on server

20 years of Java
Just The Beginning



Its HIPE ...because of springboot
& open-source & java community

Conclusion

Conclusion

Only a very short introduction to spring-boot

This document:

<http://arnaud-nauwynck.github.io/docs/Intro-SpringBoot.pdf>