

SPRING

Ripasso

- Inversion of control
- Dependency injection
- MVC
- Data Binding

INTRODUZIONE A SPRING BOOT

Il problema

- Tonnellate di configurazione

Molto soggetto a errori

Facile fare un semplice errore

```
<!-- Step 1: Configure Spring MVC Dispatcher Servlet -->
<servlet>
    <servlet-name>dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</s
    <init-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>/WEB-INF/spring-mvc-demo-servlet.xml</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>

<!-- Step 2: Set up URL mapping for Spring MVC Dispatcher Servlet -->
<servlet-mapping>
    <servlet-name>dispatcher</servlet-name>
    <url-pattern>/</url-pattern>
</servlet-mapping>

<!-- Step 4: Add support for conversion, formatting and validation support -->
<mvc:annotation-driven/>

<!-- Step 5: Define Spring MVC view resolver -->
<bean
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>
```

Spring Boot: obiettivi

- Semplificare l'avvio di Spring
- Ridurre al minimo la quantità di configurazione manuale
- Eseguire la configurazione automatica in base agli oggetti definiti
- Aiuto per risolvere i conflitti di dipendenza (Maven or Gradle)
- Fornire un server HTTP incorporato in modo da poter avviare rapidamente
- Tomcat, Jetty, Undertow, ...

Spring Initializr

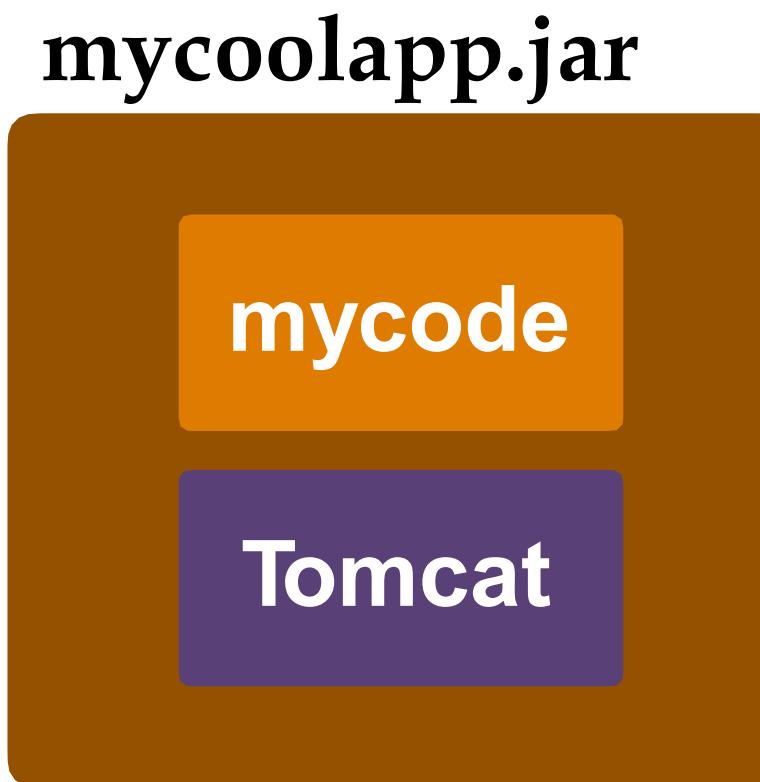
- Crea rapidamente un progetto Spring iniziale
- Selezionare le dipendenze
- Crea progetti Maven/Gradle
- Importare il progetto nell'IDE
 - Eclipse, IntelliJ, NetBeans etc ...

<http://start.spring.io>

The screenshot shows the Spring Initializr web application. At the top, it says "SPRING INITIALIZR bootstrap your application now". Below that, there's a form to "Generate a Maven Project with Java and Spring Boot 2.0.5". The "Project Metadata" section includes fields for "Group" (com.example) and "Artifact" (demo). The "Dependencies" section has a search bar for "Add Spring Boot Starters and dependencies to your application" with suggestions like "Web, Security, JPA, Actuator, Devtools...". A "Selected Dependencies" list is empty. At the bottom right is a green "Generate Project" button.

Spring Boot Embedded Server

- Fornire un server HTTP incorporato in modo da poter iniziare rapidamente
Tomcat, Jetty, Undertow, ...
- Non è necessario installare un server separatamente



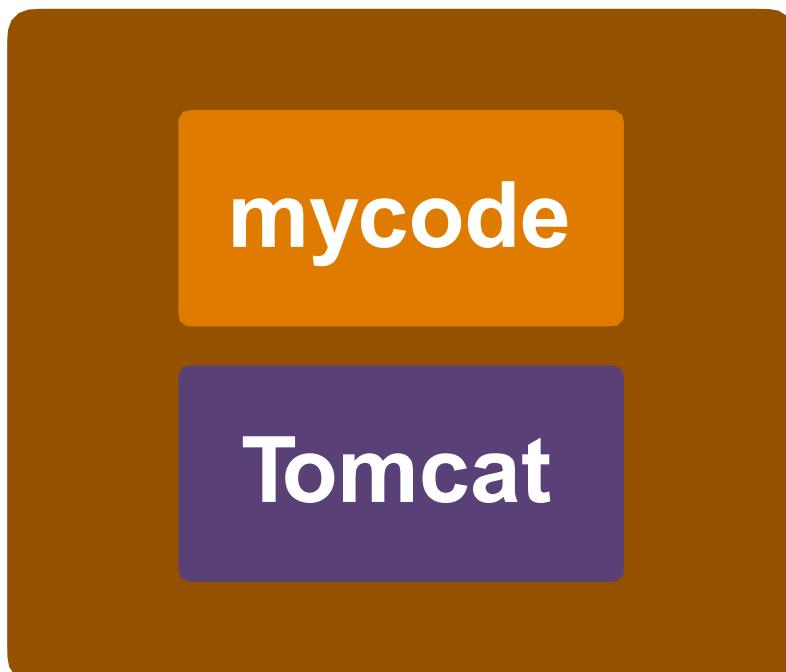
File JAR
include il codice
dell'applicazione e
include il server

Self-contained unit
Nient'altro da installare

Running Spring Boot Apps

- Le app costruite con Spring Boot possono essere eseguite autonomamente (server incorporato)
- L'app può essere eseguita da linea di comando (come semplice programma Java) o da un IDE

mycoolapp.jar

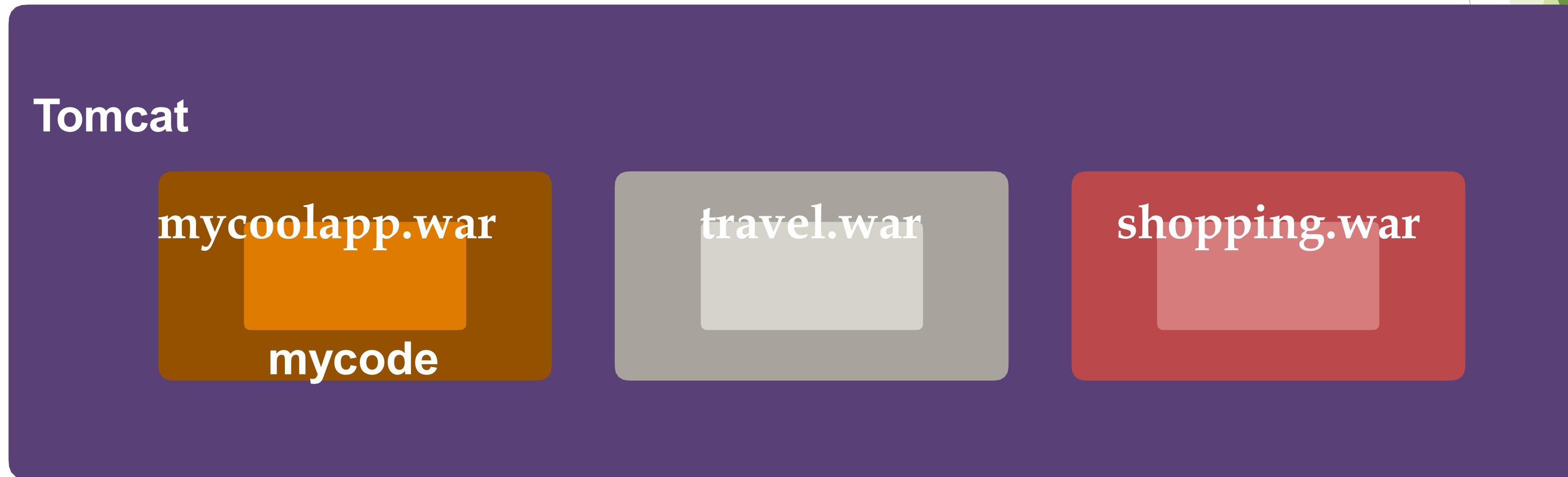


```
> java -jar mycoolapp.jar
```

Nome del nostro JARfile

Deploying Spring Boot

- Le app Spring Boot possono anche essere distribuite in modo tradizionale
- Distribuire il file WAR in un server esterno: Tomcat, JBoss, WebSphere etc ...



Spring Boot FAQ #1

Q: Spring Boot sostituisce Spring MVC, Spring REST ecc ...?

- No. Invece, Spring Boot utilizza effettivamente queste tecnologie



Una volta che si es utilizzano le normali configurazioni spring boot, si utilizza la normale codifica spring
@Component
@Controller
@Autowired
etc...

Spring Boot FAQ #2

Q: Spring Boot esegue il codice più velocemente rispetto al classico Spring?

No.

- Dietro le quinte, Spring Boot utilizza lo stesso codice del framework Spring
- Spring Boot vuole solo semplificare la fase iniziale
- Ridurre al minimo la configurazione ecc ...

Spring Boot FAQ #3

Q: Ho bisogno di un IDE speciale per Spring Boot?

- No.
- Si può utilizzare un qualsiasi IDE per le app Spring Boot ... Anche un semplice editor di testo.
- Il team spring fornisce gratuitamente Spring Tool Suite (STS) [plugin IDE]

Come usare Spring Initializr

Step-By-Step

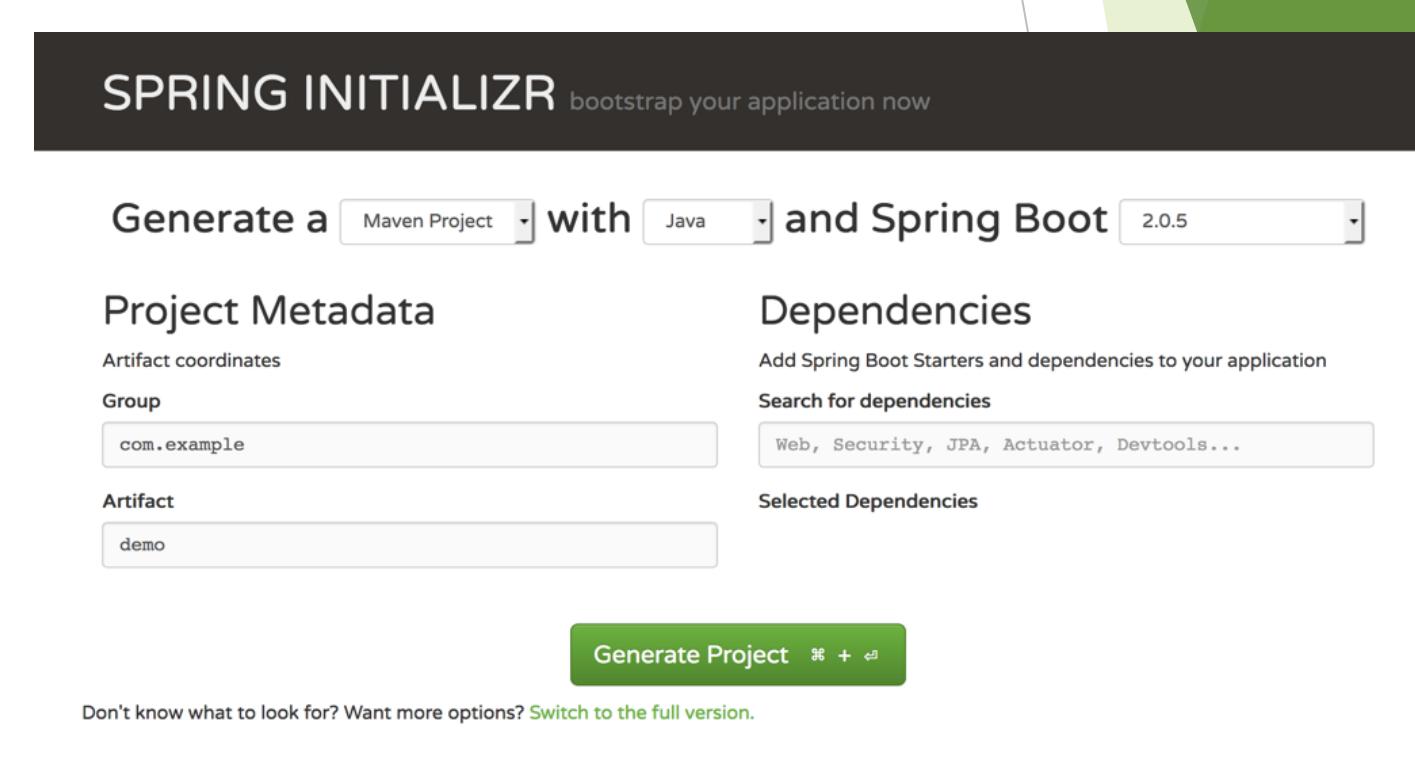
1. Configura il nostro progetto sul sito web di Spring Initializr

2. Scarica il file zip

3. Decomprimere il file

4. Importa il progetto Maven nel nostro IDE

<http://start.spring.io>



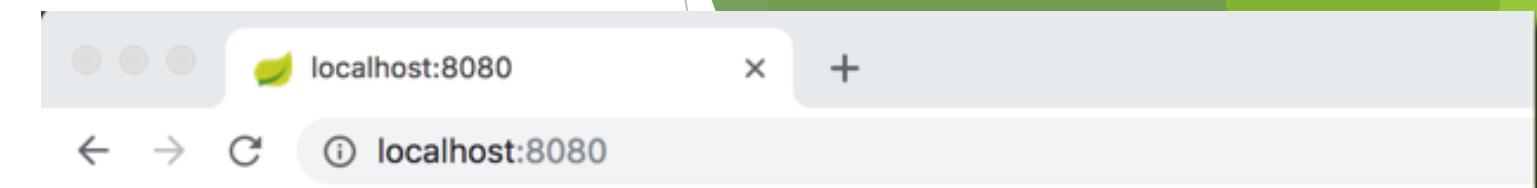
REST Controller

- Creiamo un controller REST molto semplice



Create REST Controller

```
@RestController  
public class FunRestController {  
  
    // expose "/" that return "Hello World"  
  
    @GetMapping("/")  
    public String sayHello() {  
        return "Hello World! Time on server is " + LocalDateTime.now();  
    }  
}
```

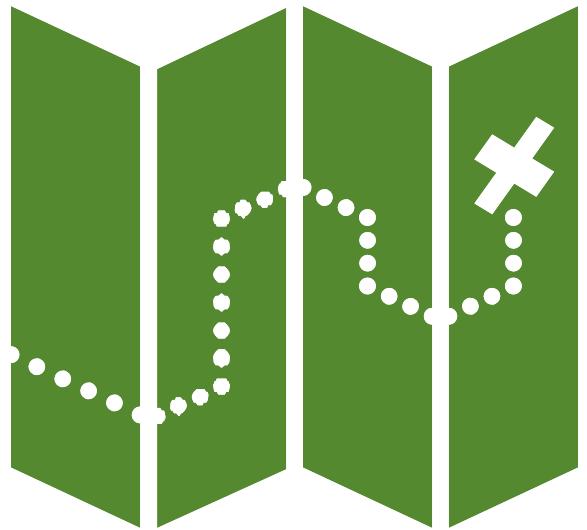


Hello World! Time on server is

Timestamp corrente

Spring Initializr

- Spring Initializr crea un progetto Maven per noi
- Esploriamo la struttura del progetto



<http://start.spring.io>

SPRING INITIALIZR bootstrap your application now

Generate a with and Spring Boot

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Dependencies

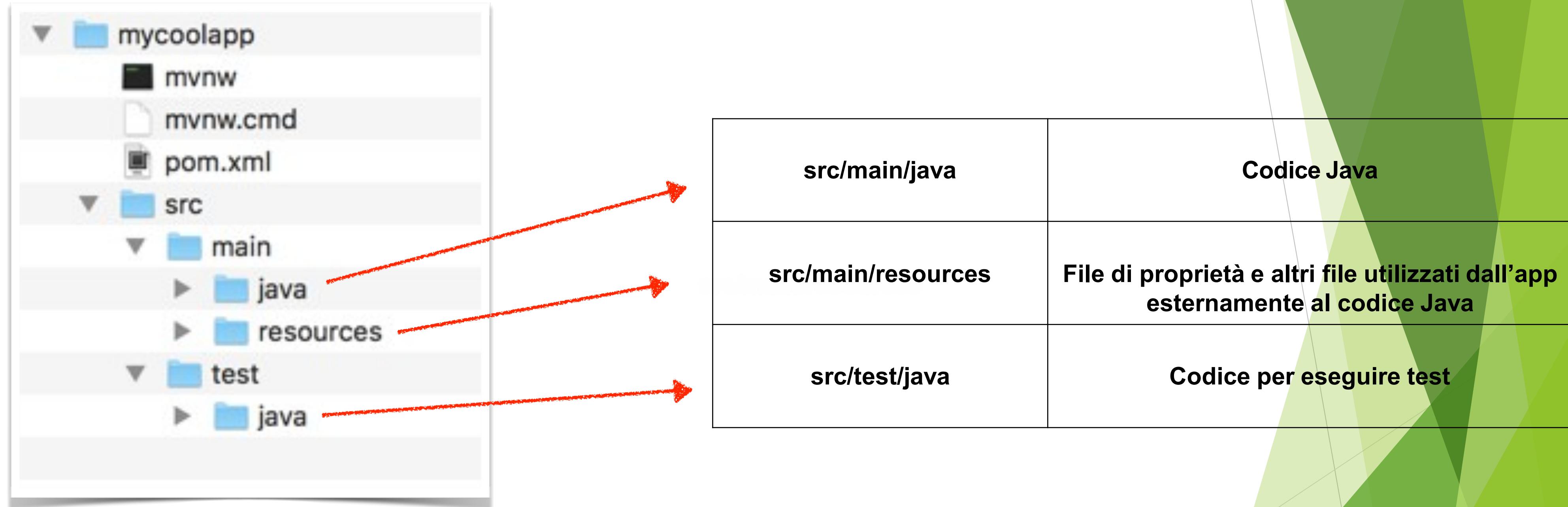
Add Spring Boot Starters and dependencies to your application

Search for dependencies: Web, Security, JPA, Actuator, Devtools...

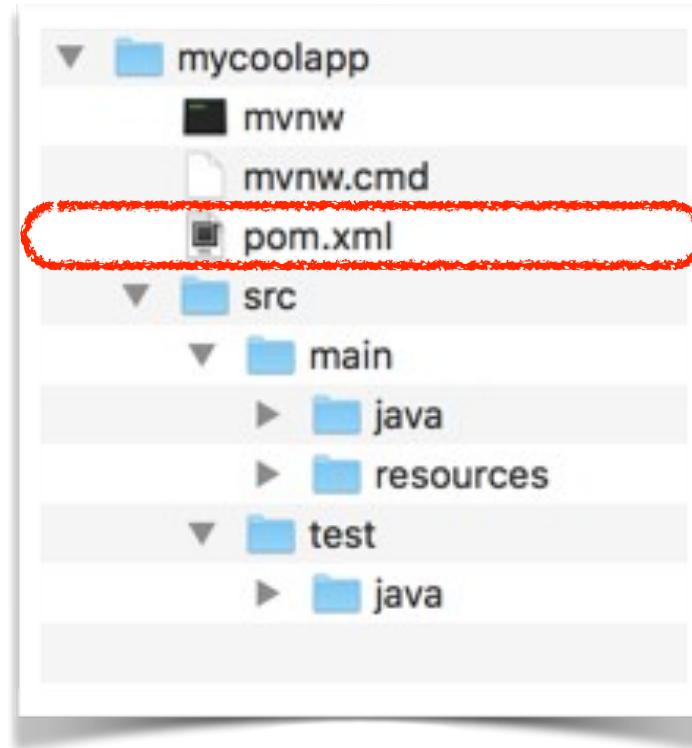
Selected Dependencies

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

Maven Standard Directory



Maven POM file

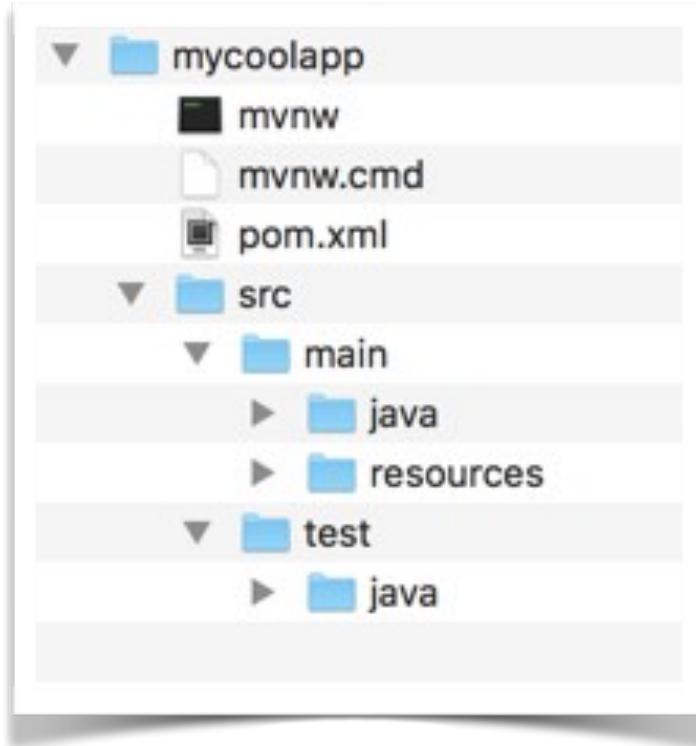


```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

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

Maven POM file

- Spring Boot Maven plugin

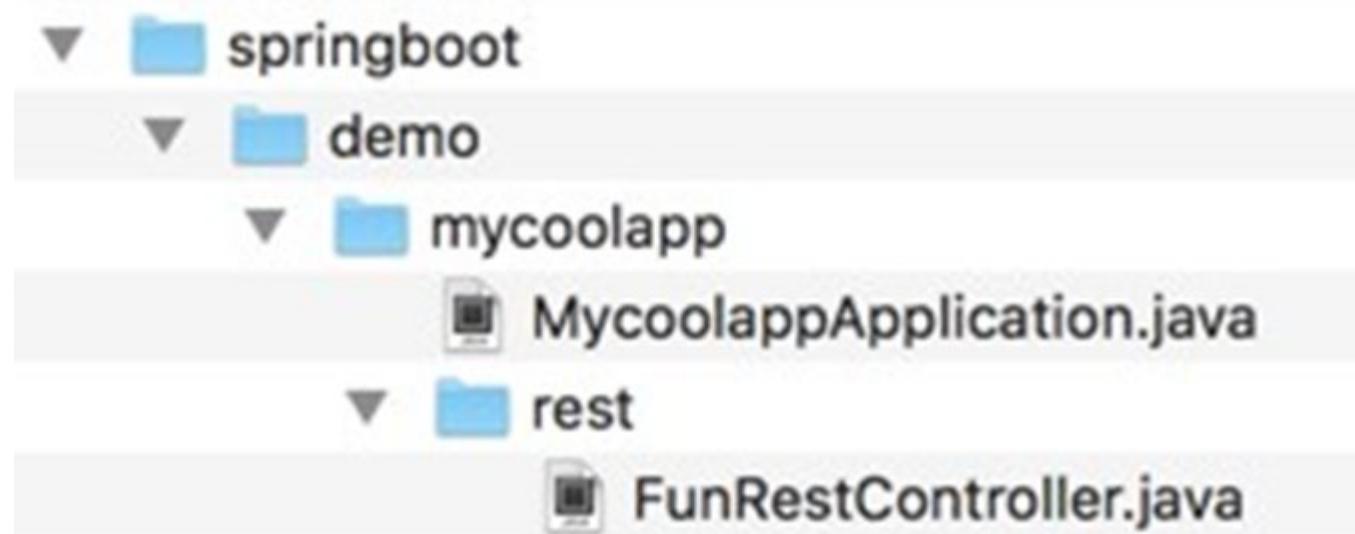


```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-
        plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

Java Source Code

Main Spring Boot application class

Creato da Spring Initializr



RestController che abbiamo
creato in precedenza

Java Source Code

File: MycoolappApplication.java

```
package it.examples;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class MycoolappApplication {

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

Abilità

Auto configuration
Component scanning
Additional configuration

Annotations

- **@SpringBootApplication** è composto dalle seguenti annotazioni:

@EnableAutoConfiguration

@ComponentScan

@Configuration

Java Source Code

File: MycoolappApplication.java

```
package it.examples;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.*;

@SpringBootApplication
public class MycoolappApplication {

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

Avvia l'applicazione Spring Boot

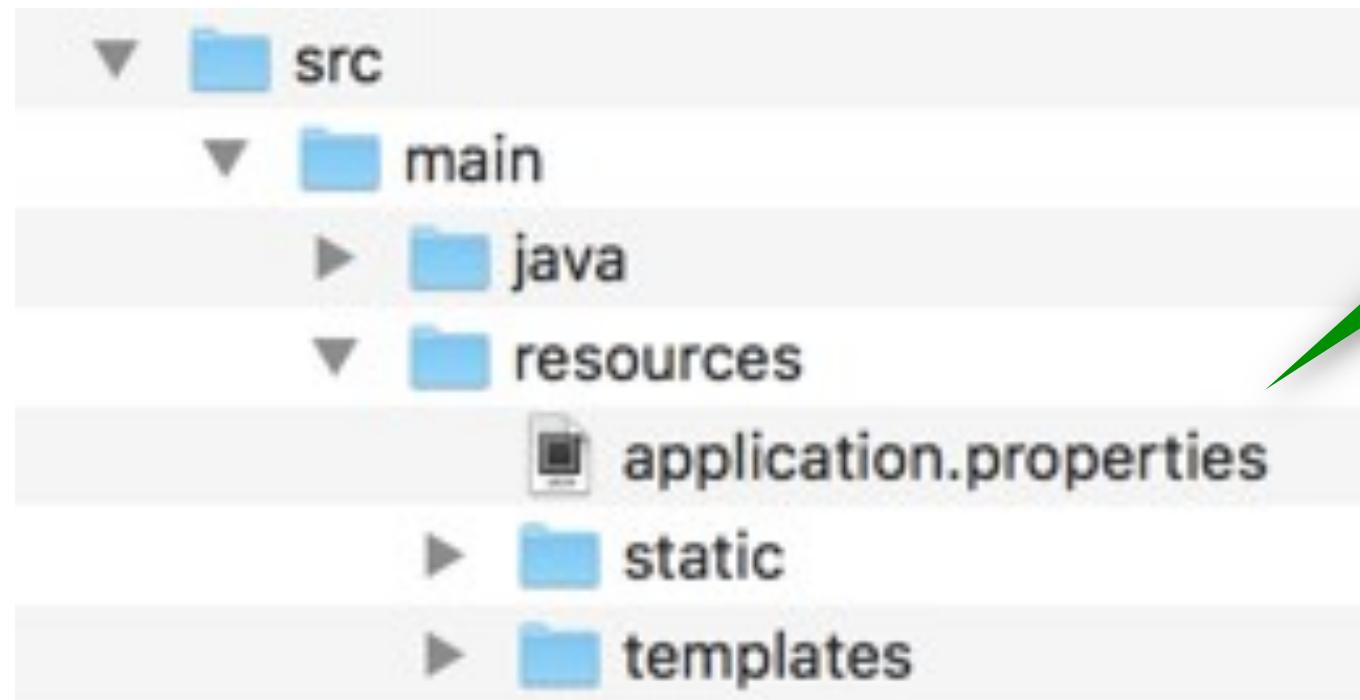
Dietro le quinte...

Crea l'application context e
registra tutti i bean

Avvia il server integrato
Tomcat etc...

Application Properties

- Per impostazione predefinita, Spring Boot caricherà le proprietà da: **application.properties**



Creata da Spring Initializr

Vuoto all'inizio

Può aggiungere proprietà di Spring Boot
server.port=8585

Aggiungere anche proprietà personalizzate
coach.name=Mickey Mouse

Application Properties

- Leggere i dati da:
application.properties

```
# configure server port
server.port=8484

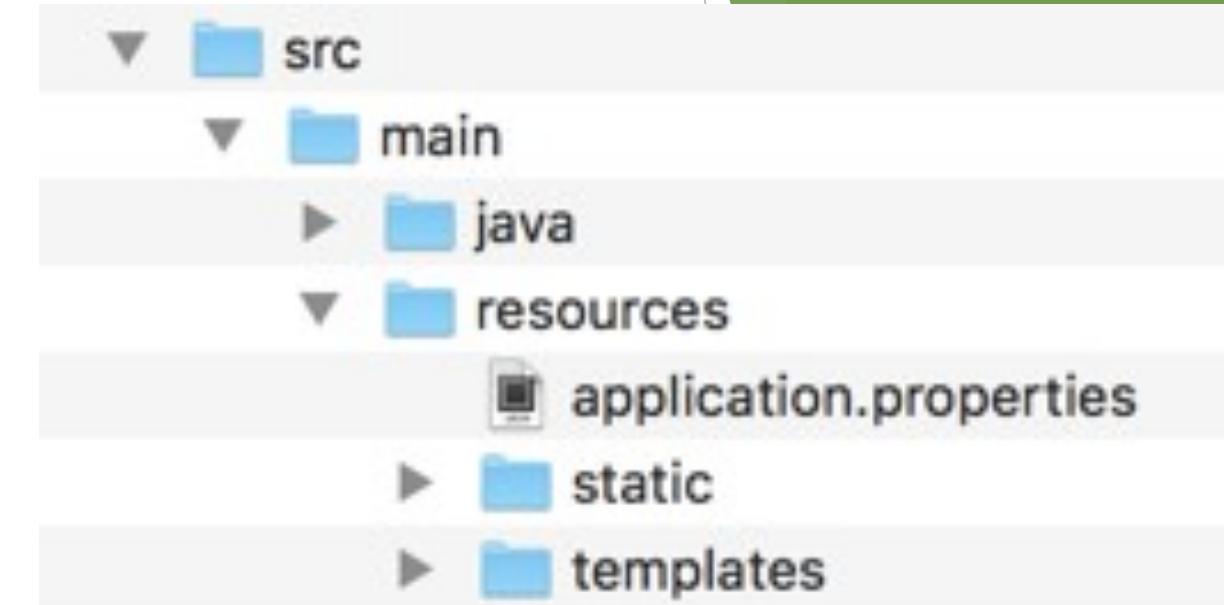
# configure my props
coach.name=Mickey Mouse
team.name=The Mouse Crew
```

```
@RestController
public class FunRestController {

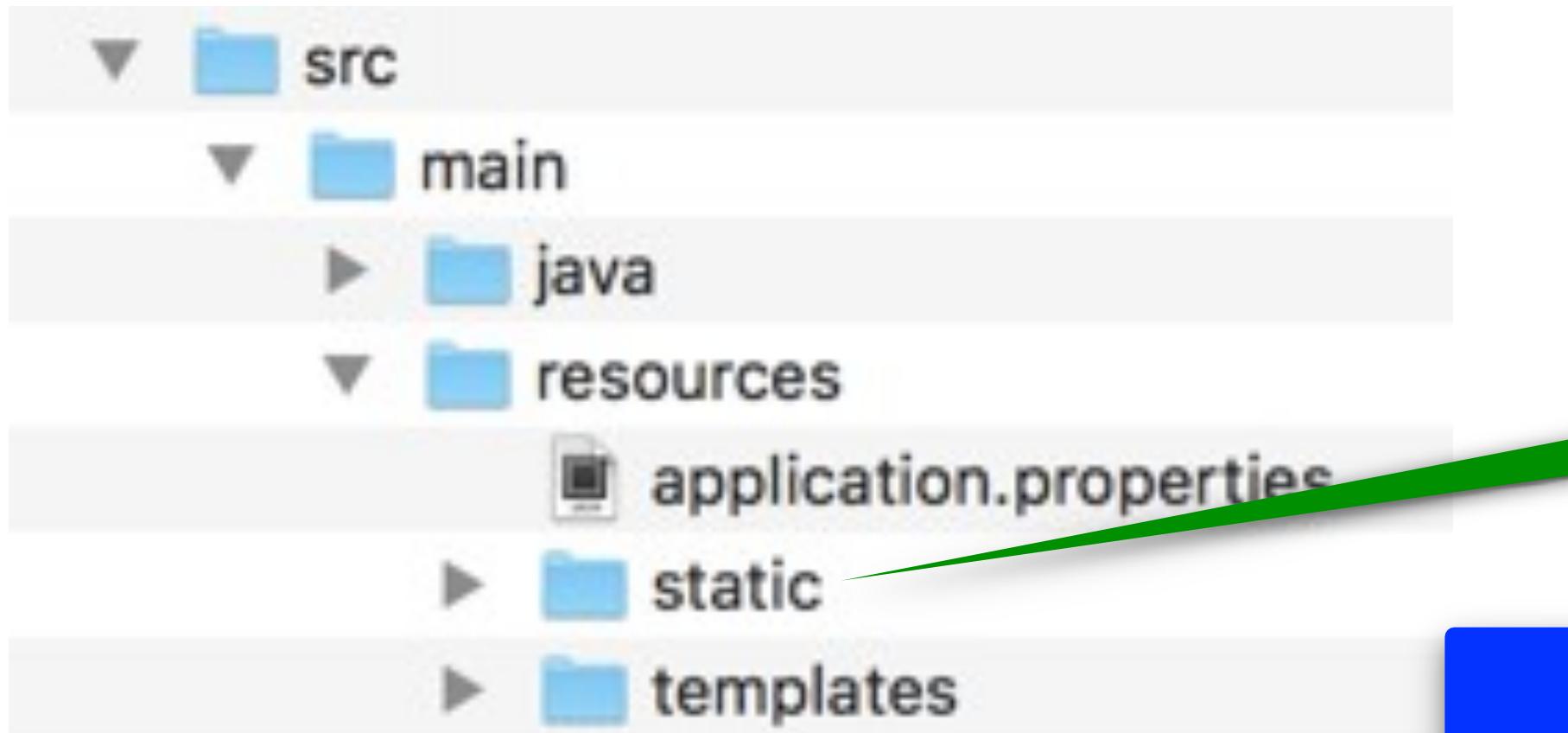
    @Value("${coach.name}")
    private String coachName;

    @Value("${team.name}")
    private String teamName;

}
```



Static Content

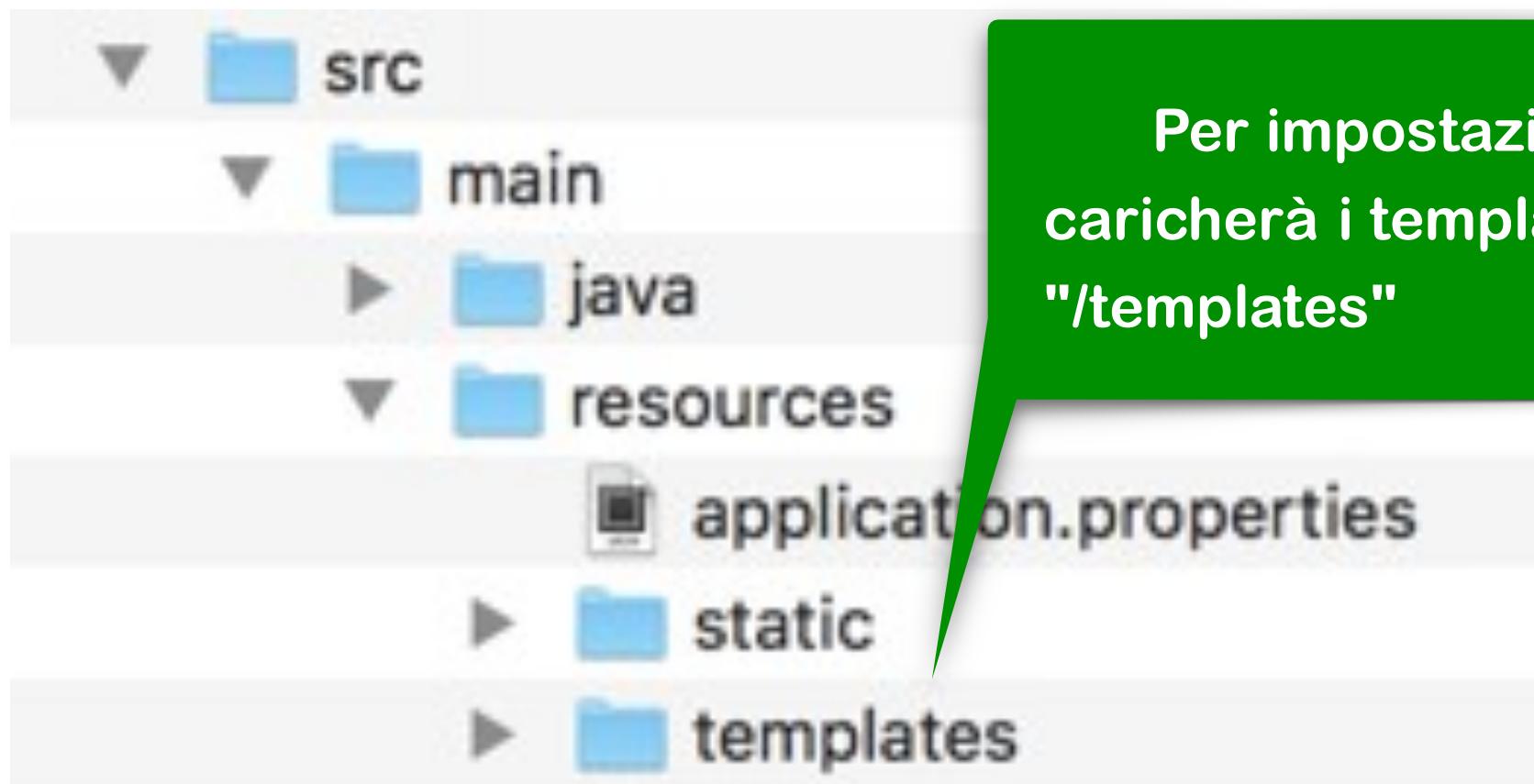


Per impostazione predefinita, Spring Boot caricherà risorse statiche dalla directory "/static"

Esempi di risorse statiche
HTML files, CSS, JavaScript, images, etc ...

Templates

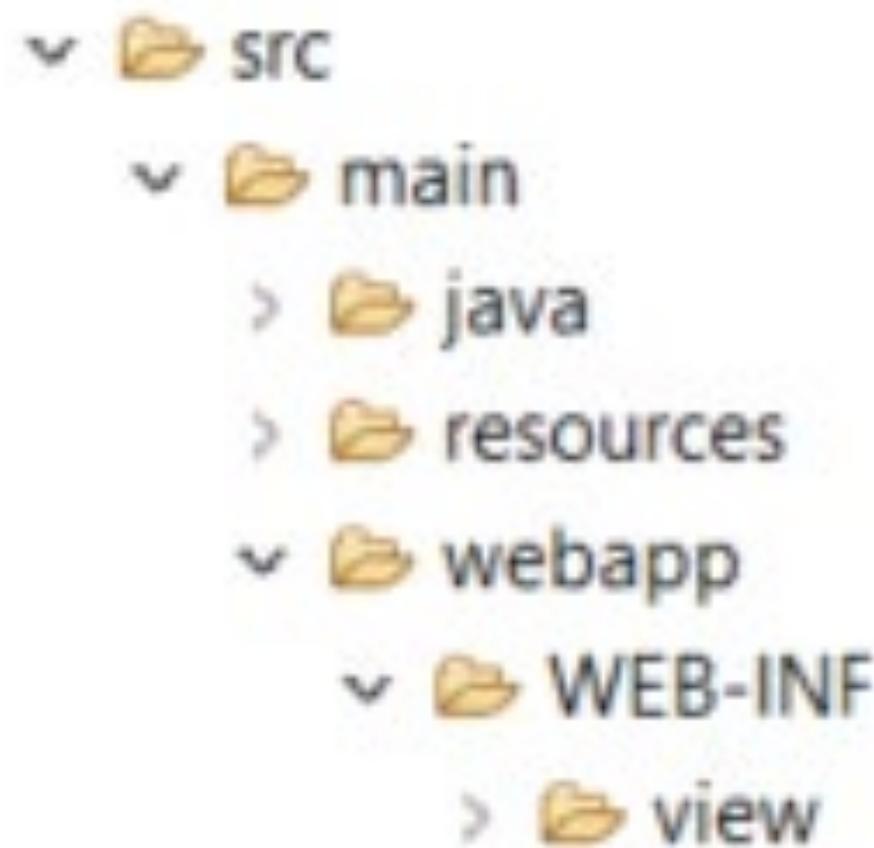
- Spring Boot include la configurazione automatica per i seguenti motori di templating
 - FreeMarker
 - Thymeleaf
 - Mustache



Per impostazione predefinita, Spring Boot caricherà i template (le view) dalla directory "/templates"

Thymeleaf è il più utilizzato

E JSP?



```
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>

<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
</dependencies>
```

Il problema delle dipendenze Maven

- Costruire un'applicazione Spring è davvero DIFFICILE!!!

FAQ: Di quali dipendenze Maven ho bisogno?

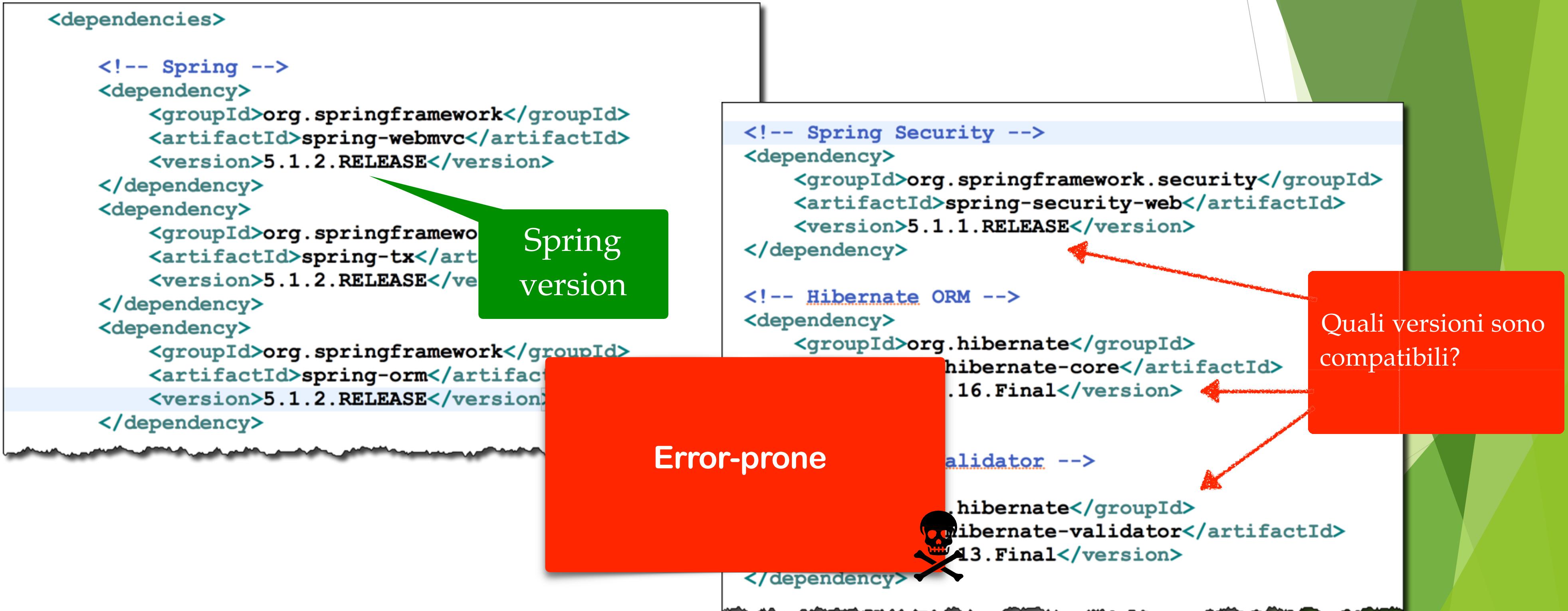
Hi, I want to know the minimum required Maven Dependencies for Spring MVC + Hibernate connection.

· Lecture 314 · 3 days ago

if we use pom . What are the exact dependencies

· Lecture 268 · 19 days ago

Il problema



Soluzione: Spring Boot starters

- **Spring Boot Starters**
- Un elenco curato delle dipendenze di Maven
- Raccolta di dipendenze raggruppate
- Rende molto più facile l'inizializzazione del progetto
- Riduce la quantità di configurazione Maven



Spring MVC

- Ad esempio, quando si costruisce un'app Spring MVC, di solito è necessario includere le seguenti librerie

```
<!-- Spring MVC -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.1.2.RELEASE</version>
</dependency>

<!-- Form validation: Hibernate Validator -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-validator</artifactId>
    <version>6.0.13.Final</version>
</dependency>

<!-- Web template: JSP or Thymeleaf etc -->
<dependency>
    <!-- ... -->
</dependency>
```

Solution: Spring Boot Starter - Web

- Spring Boot fornisce: `spring-boot-starter-web`

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

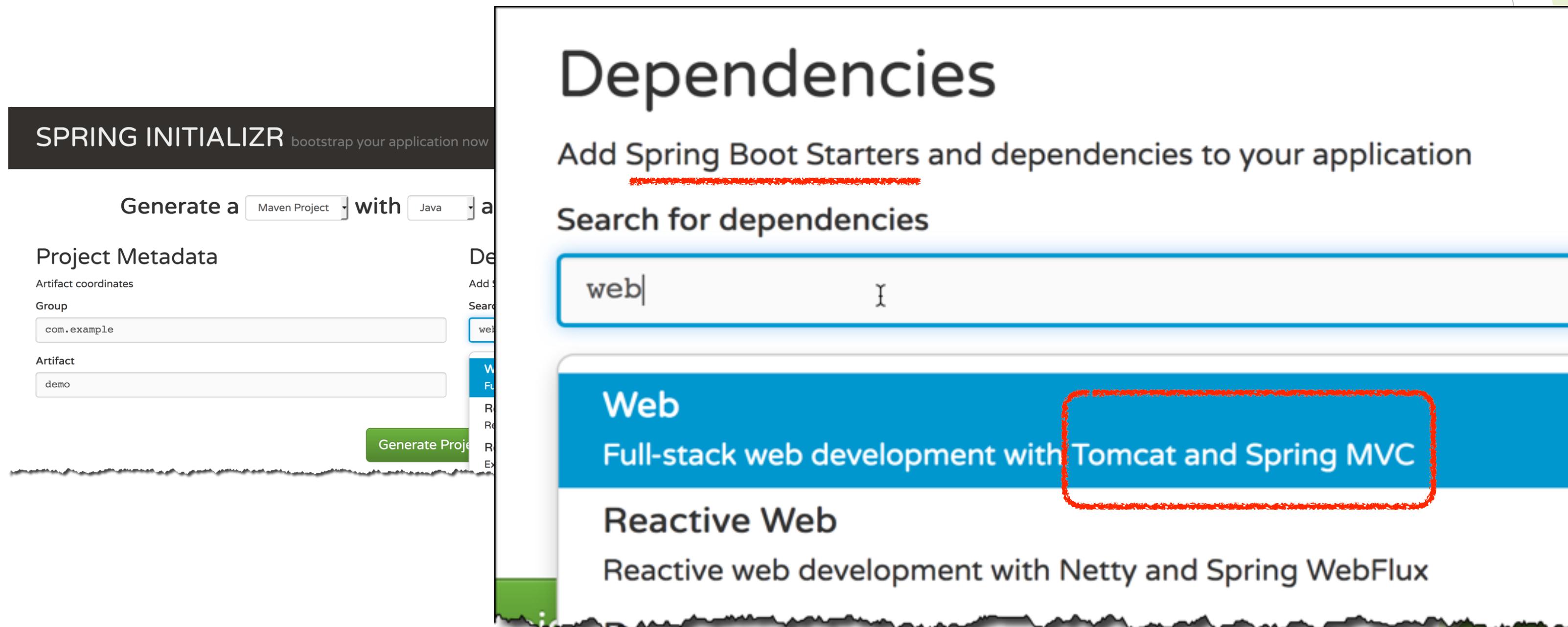
Salva lo sviluppatore dal dover elencare tutte le singole dipendenze

Inoltre, fa in modo che si abbiano versioni compatibili

CONTIENE
spring-web
spring-webmvc
json
tomcat
...

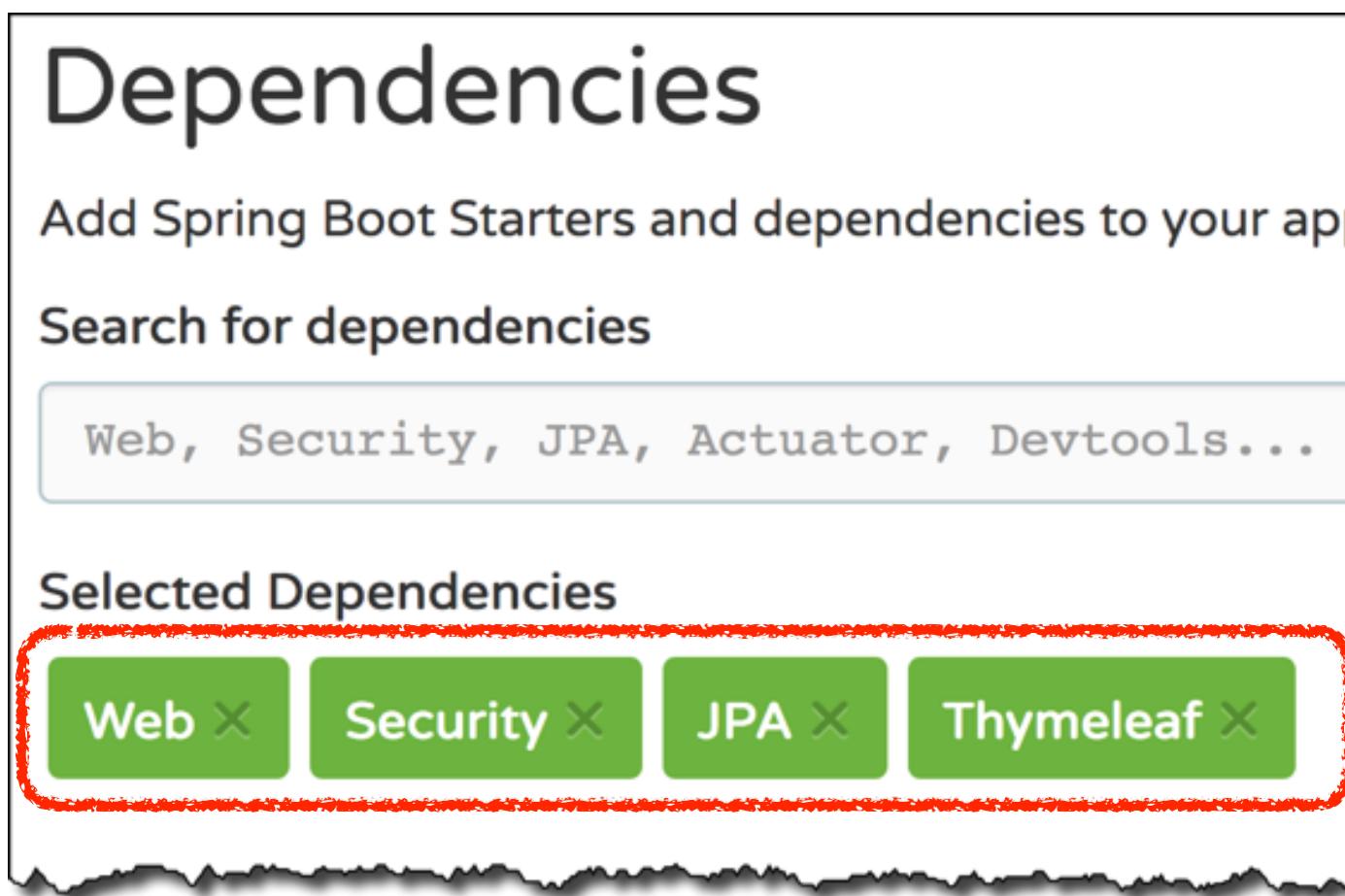
Spring Initializr

- In Spring Initializr, selezionare semplicemente dipendenza Web
- Si ottiene automaticamente **spring-boot-starter-web** nel **pom.xml**



Spring Initializr

- Se stiamo costruendo un'app Spring che ha bisogno di: Web, Sicurezza, ...
- Basta selezionare le dipendenze in Spring Initializr
- Spring Initializr aggiungerà i pacchetti starters necessari nel **pom.xml**



pom.xml

Spring Initializr

Dependencies

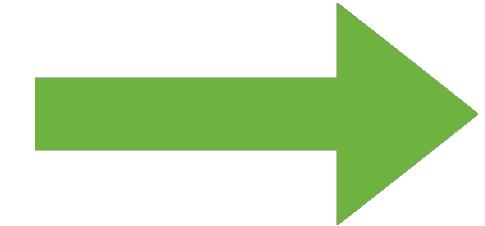
Add Spring Boot Starters and dependencies to your app

Search for dependencies

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

Selected Dependencies

Web X Security X JPA X Thymeleaf X



File: pom.xml

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

<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency> _____

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

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

Spring Boot Starters

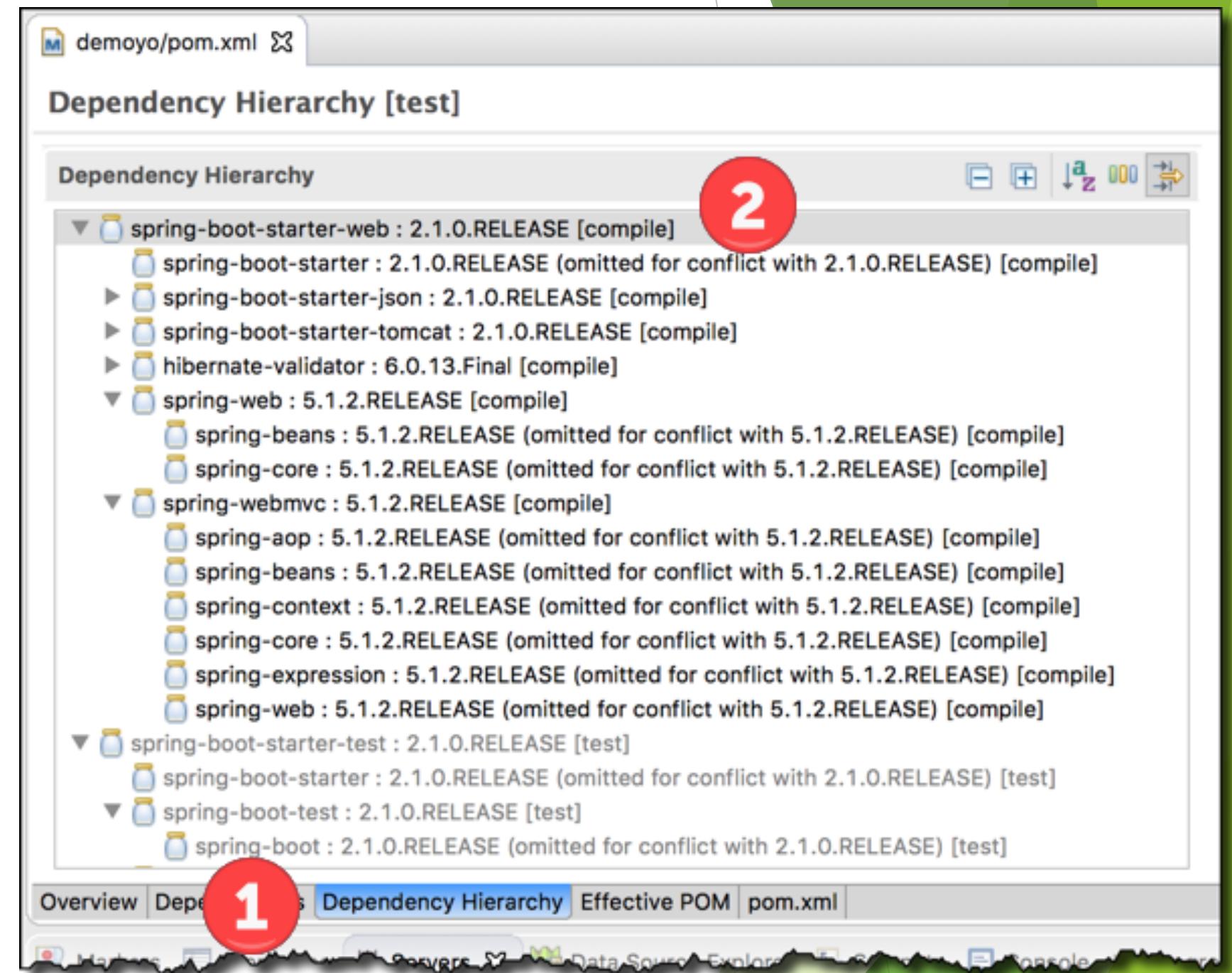
- Sono disponibili più di 30 pacchetti starter in Spring Boot...

spring-boot-starter-web	Creazione di app Web e REST. Utilizza Tomcat come server incorporato predefinito.
spring-boot-starter-security	Aggiunta del supporto per la sicurezza
spring-boot-starter-data-jpa	Supporto del database con JPA e Hibernate
...	

Cos'è uno Starter?



- In Eclipse...
- Apriamo il pom.xml
- Selezioniamo la tab **Dependency Hierarchy**
- Apriamo uno starter...



Spring Boot Starter Parent

- Spring Boot fornisce uno "Starter Parent", ovvero uno starter «speciale» fornito da Maven dopo l'inizializzazione del progetto in Spring Initializr

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.0.RELEASE</version>
  <relativePath/>
</parent>
```

Spring Boot Starter Parent

- Per i successivi pacchetti **spring-boot-starter-***, non è necessario elencare la versione

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>2.1.0.RELEASE</version>
  <relativePath/>
</parent>

<dependencies>

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

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

Versione di
Spring Boot

I pacchetti starter
ereditano la versione

Non c'è bisogno di elencare singole versioni
Ottimo in termini di manutenzione!

Il problema delle modifiche a run-time

- Quando si eseguono le applicazioni Spring Boot

Se si apportano modifiche al codice sorgente è
necessario riavviare manualmente l'applicazione :-(

Soluzione: Spring Boot DevTools

- **spring-boot-devtools** in soccorso!
- Riavvia automaticamente l'applicazione quando il codice viene aggiornato
- Basta aggiungere la dipendenza al file POM
- Non è necessario scrivere codice aggiuntivo :-)

Spring Boot Dev Tools

- Aggiunta della dipendenza al file POM

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

Riavvia automaticamente l'applicazione
quando il codice viene aggiornato

Soluzione: Application Properties file

- Per impostazione predefinita, Spring Boot legge le informazioni da un file properties standard presente in `src/main/resources/application.properties`
- È possibile definire **QUALSIASI** proprietà personalizzata in questo file
- L'app Spring Boot può accedere alle proprietà utilizzando `@Value`

Processo di sviluppo

Step-By-Step

1. Definire proprietà personalizzate in **application.properties**
2. Iniettare proprietà nell'applicazione Spring Boot utilizzando **@Value**

Step 1: Definire proprietà personalizzate dell'applicazione

File: src/main/resources/application.properties

```
#  
# Define custom properties  
#  
coach.name=Mickey Mouse  
team.name=The Mouse Club
```

È possibile assegnare NOMI DI
PROPRIETÀ PERSONALIZZATI

Step 2: Iniettare proprietà

```
@RestController  
public class FunRestController {
```

// inject properties for: coach.name and team.name

```
@Value("${coach.name}")  
private String coachName;
```

```
@Value("${team.name}")  
private String teamName;
```

Nessun codice aggiuntivo

File: src/main/resources/application.properties

```
#  
# Define custom properties  
#  
coach.name=Mickey Mouse  
team.name=The Mouse Club
```

Spring Boot Properties

- Spring Boot può essere configurato nel file **application.properties**
- Server port, data, security etc ...
- Spring Boot ha più di 1,000 proprietà...

Spring Boot Properties

Le proprietà sono raggruppate nelle seguenti categorie:

Core

Web

Security

Data

Actuator

Integration

DevTools

Testing

Core Properties

Core

File: src/main/resources/application.properties

```
# Log levels severity mapping
logging.level.org.springframework=DEBUG
logging.level.org.hibernate=WARN
logging.level.it.examples=ERROR

# Log file name
logging.file=my-crazy-stuff.log
```

Logging Levels

TRACE
DEBUG
INFO
WARN
ERROR
FATAL
OFF

Web Properties

Web

File: src/main/resources/application.properties

```
# HTTP server port  
server.port=7070  
  
# Context path of the application  
server.servlet.context-path=/my-silly-app
```

http://localhost:7070/my-silly-app/fortune

Security Properties

File: src/main/resources/application.properties

```
# Default user name  
spring.security.user.name=admin  
  
# Password for default user  
spring.security.user.password=topsecret
```

Data Properties

Data

File: src/main/resources/application.properties

```
# JDBC URL of the database
spring.datasource.url=jdbc:mysql://localhost:3306/ecommerce

# Login username of the database
spring.datasource.username=scott

# Login password of the database
spring.datasource.password=tiger
```

APPROFONDIMENTO SULLA VALIDAZIONE

Java Standard Bean Validation API

Annotation	Description
@NotNull	Checks that the annotated value is not null
@Min	Must be a number \geq value
@Max	Must be a number \leq value
@Size	Size must match the given size
@Pattern	Must match a regular expression pattern
@Future / @Past	Date must be in future or past of given date
others ...	

Quale dipendenza Maven?

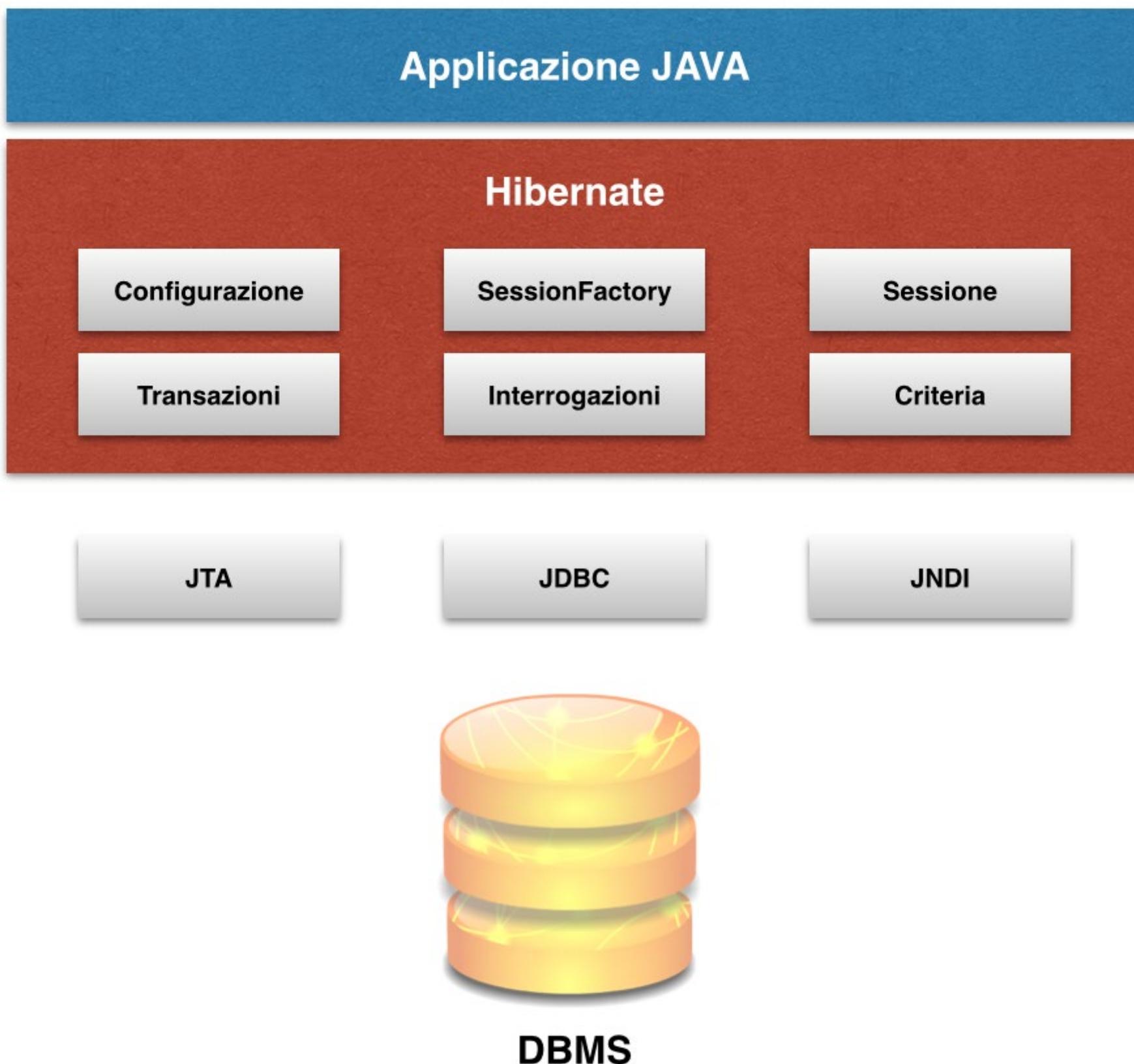
```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-validation</artifactId>
</dependency>
```



The background features a large, light green triangle on the right side, composed of several smaller, semi-transparent green triangles of varying shades. A thin white line starts from the bottom center and extends upwards towards the top right corner, intersecting the green triangles.

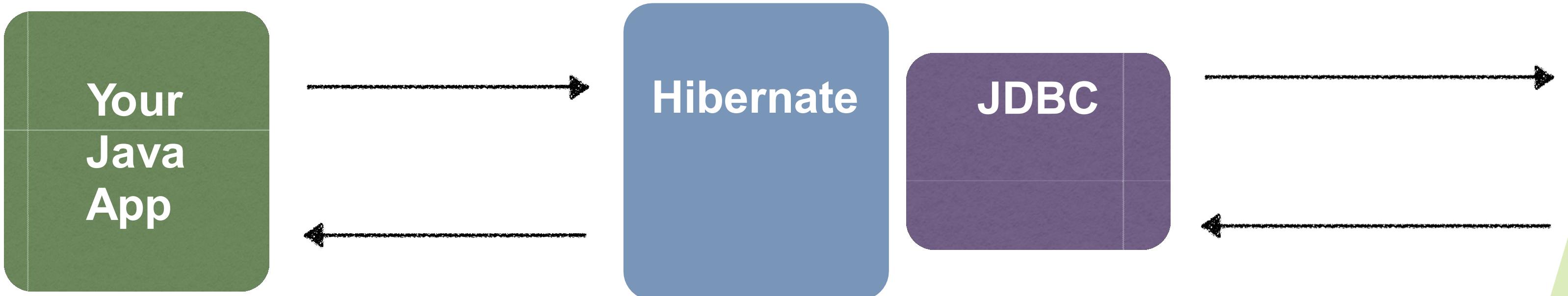
SPRING DATA

Cos'è Hibernate



Hibernate e JDBC

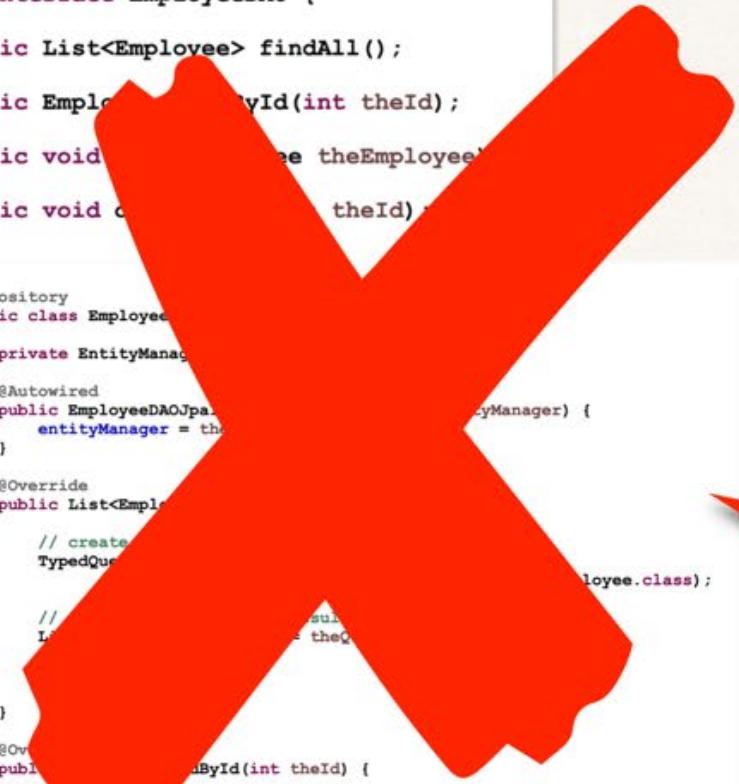
- Hibernate utilizza JDBC per tutte le comunicazioni con il database



Con Spring Data

Before Spring Data JPA

```
public interface EmployeeDAO {  
    public List<Employee> findAll();  
    public Employee findById(int theId);  
    public void create(Employee theEmployee);  
    public void delete(int theId);  
}  
  
@Repository  
public class EmployeeDAOJpa implements EmployeeDAO {  
    private EntityManager entityManager;  
    @Autowired  
    public EmployeeDAOJpa(EntityManager theEntityManager) {  
        entityManager = theEntityManager;  
    }  
    @Override  
    public List<Employee> findAll() {  
        // create query  
        TypedQuery<Employee> query = entityManager.createQuery("select e from Employee e");  
        return query.getResultList();  
    }  
    @Override  
    public Employee findById(int theId) {  
        // get employee  
        Employee theEmployee = entityManager.find(Employee.class, theId);  
        // return employee  
        return theEmployee;  
    }  
}
```



Tanti file ed eccessiva configurazione

After Spring Data JPA

```
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
    // that's it ... no need to write any code LOL!  
}
```

Un file e 3 linee di codice



Nessuna implementazione dell'interfaccia necessaria

Quali dipendenze Maven?

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

<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <scope>runtime</scope>
</dependency>
```

Terminologia

Primary Key

Identifica in modo univoco ogni riga di una tabella

Deve essere un valore univoco

Non può contenere valori NULL

MySQL - Auto Increment

```
CREATE TABLE student (
    id int(11) NOT NULL AUTO_INCREMENT,
    first_name varchar(45) DEFAULT NULL,
    last_name varchar(45) DEFAULT NULL,
    email varchar(45) DEFAULT NULL,
    PRIMARY KEY (id)
)
```

Hibernate Identity - Primary Key

```
@Entity  
@Table(name="student")  
public class Student {  
  
    @Id  
    @Column(name="id")  
    private int id;  
  
    ...  
}
```

Hibernate Identity - Primary Key

```
@Entity  
@Table(name="student")  
public class Student {  
  
    @Id  
    @GeneratedValue(strategy=GenerationType.IDENTITY)  
    @Column(name="id")  
    private int id;  
  
    ...  
}
```

Strategie di generazione degli ID

GenerationType.AUTO	Sceglie una strategia appropriata per il database specifico
GenerationType.IDENTITY	Assegna chiavi primarie utilizzando la colonna identità del database
GenerationType.SEQUENCE	Assegnare chiavi primarie utilizzando una sequenza interna al database
GenerationType.TABLE	Assegnare chiavi primarie utilizzando una tabella di database sottostante per garantire l'univocità

Mapping di base

Java Class

Student	
- id : int	
- firstName : String	
- lastName : String	
- email : String	
...	

Hibernate

Database Table

student	
!	id INT
◆	first_name VARCHAR(45)
◆	last_name VARCHAR(45)
◆	email VARCHAR(45)
Indexes	