Spring

Enterprise Framework

What’s Spring?

- *“Spring Framework is a Java platform that provides comprehensive infrastructure support for developing Java applications”*
- *“Spring handles the infrastructure so you can focus on your application”*
- *“Spring enables you to build applications from ‘plain old Java objects’ (POJOs) and to apply enterprise services non-invasively to POJOs”*

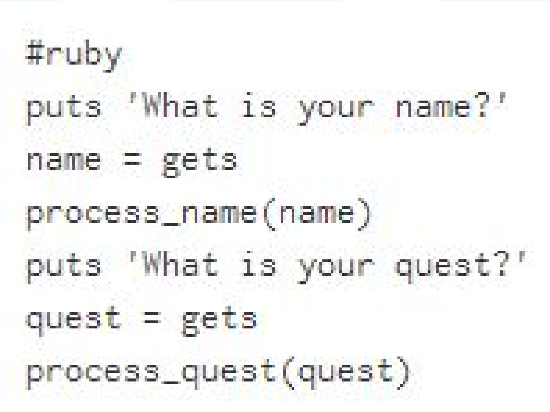
Spring main reference: http://docs.spring.io/spring/docs/3.0.x/reference/overview.html

Inversion of Control (IoC)

Design technique that delegates invoking a behavior to an assembler at runtime

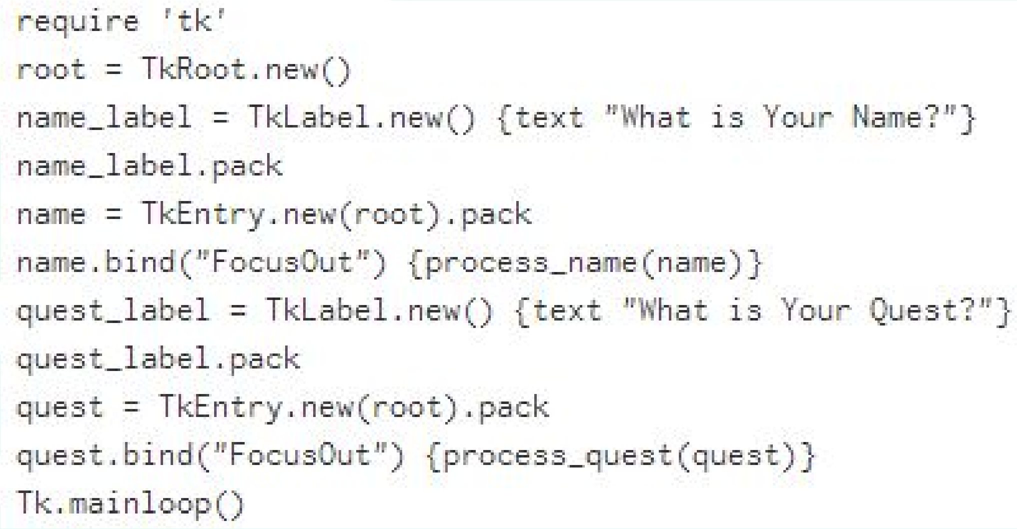
Example: program to get and process information from a user

Command line version



```
#ruby
puts 'What is your name?'
name = gets
process_name(name)
puts 'What is your quest?'
quest = gets
process_quest(quest)
```

Graphical version

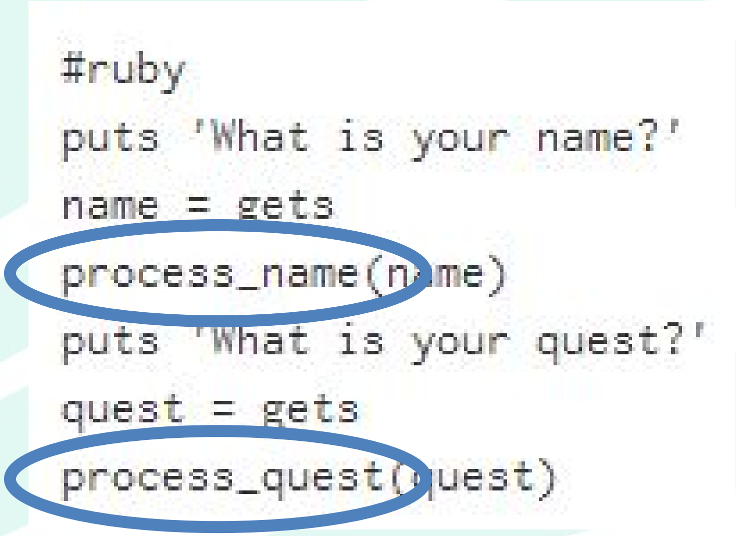


```
require 'tk'
root = TkRoot.new()
name_label = TkLabel.new() {text "What is Your Name?"}
name_label.pack
name = TkEntry.new(root).pack
name.bind("FocusOut") {process_name(name)}
quest_label = TkLabel.new() {text "What is Your Quest?"}
quest_label.pack
quest = TkEntry.new(root).pack
quest.bind("FocusOut") {process_quest(quest)}
Tk.mainloop()
```

Inversion of Control (IoC)

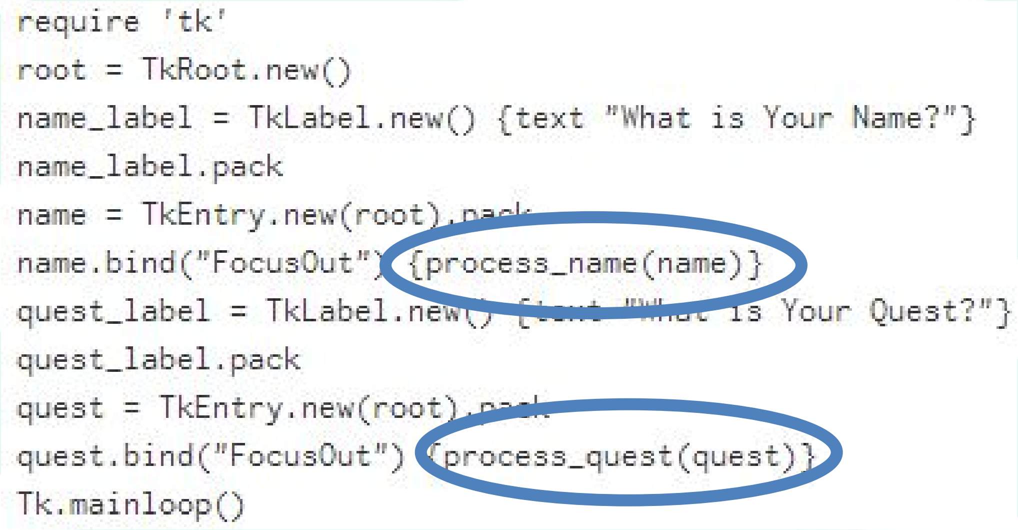
Example: program to get and process information from a user

Command line version



```
#ruby
puts 'What is your name?'
name = gets
process_name(name)
puts 'What is your quest?'
quest = gets
process_quest(quest)
```

Graphical version



```
require 'tk'
root = TkRoot.new()
name_label = TkLabel.new() {text "What is Your Name?"}
name_label.pack
name = TkEntry.new(root).pack
name.bind("FocusOut") {process_name(name)}
quest_label = TkLabel.new() {text "What is Your Quest?"}
quest_label.pack
quest = TkEntry.new(root).pack
quest.bind("FocusOut") {process_quest(quest)}
Tk.mainloop()
```

Control goes from my command line program module to the event manager module, which is instructed via "bind“

This is IoC, aka “Hollywood principle: don’t call us, we’ll call you”

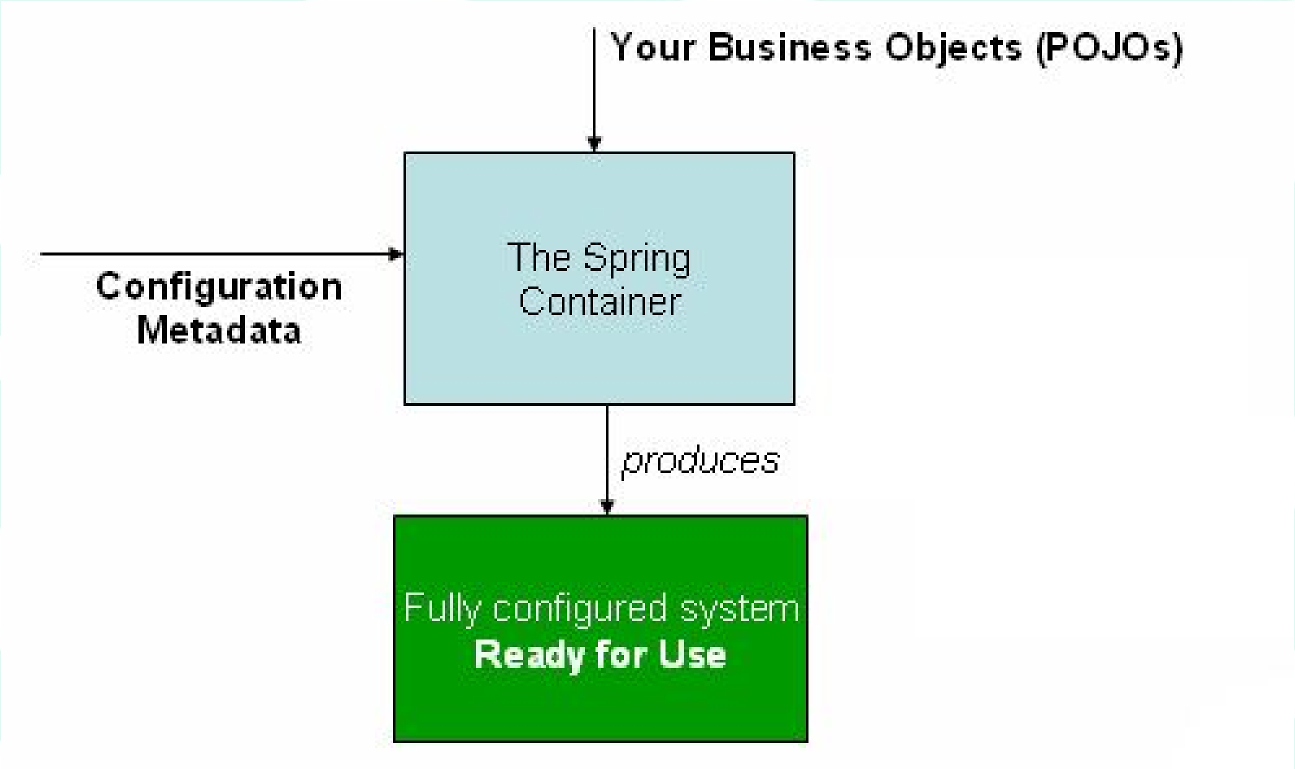
http://martinfowler.com/bliki/InversionOfControl.html

Dependency Injection (DI)

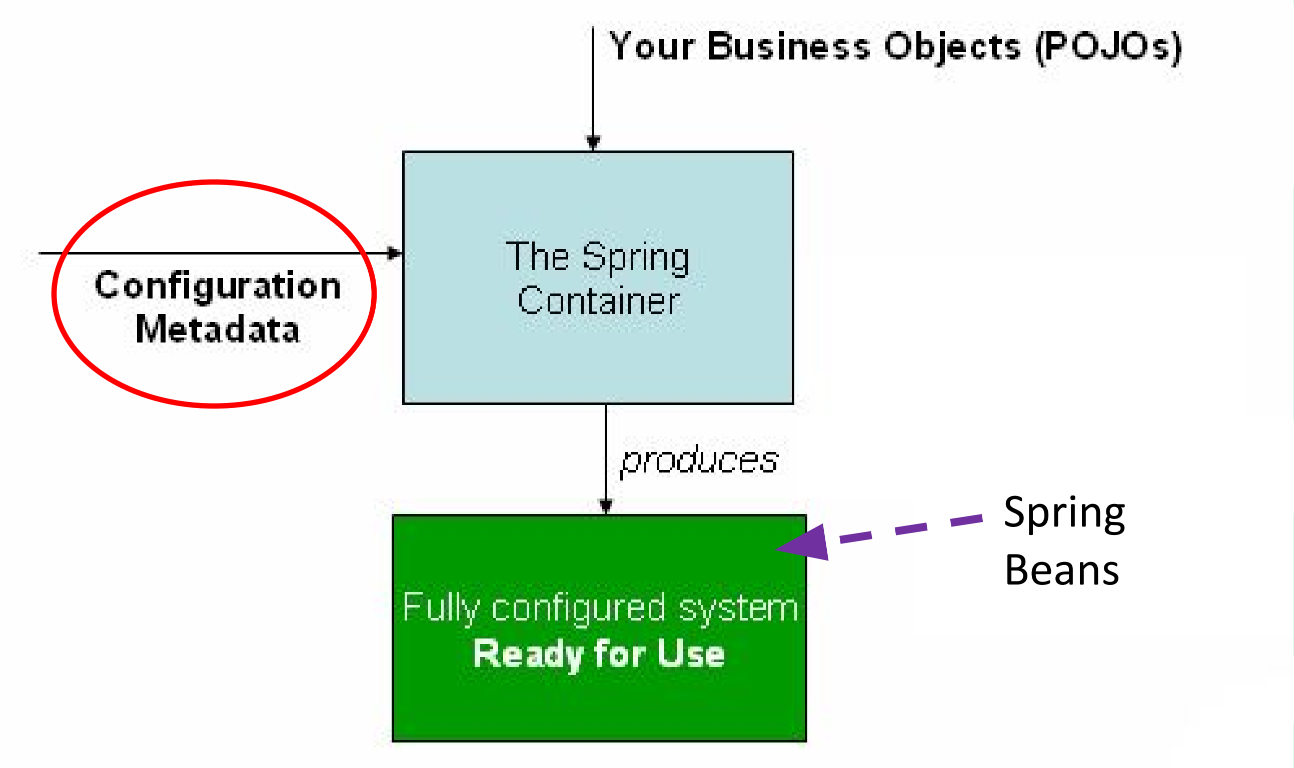
- Design pattern to create an object O1 another object O2 relies on, without knowing, at compile time, which class O1 is instance of
- 3 roles
  - Dependent consumer
  - Interface contract
  - Injector: create instances of classes implementing the interface contract and **inject** the dependency on the dependent consumer
    - The injector selects the class to instantiate

Spring heavily leverages IoC and DI

Spring IoC Container (IoCC)



Spring IoC Container (IoCC)



Configuration Metadata for IoCC

- 3 techniques
  - XML-Based configuration
  - Annotation-based configuration
    - Annotating classes, attributes, methods
  - Java-based configuration
    - Meta-data hard-coded in a Java Class

Spring Bean Autowiring

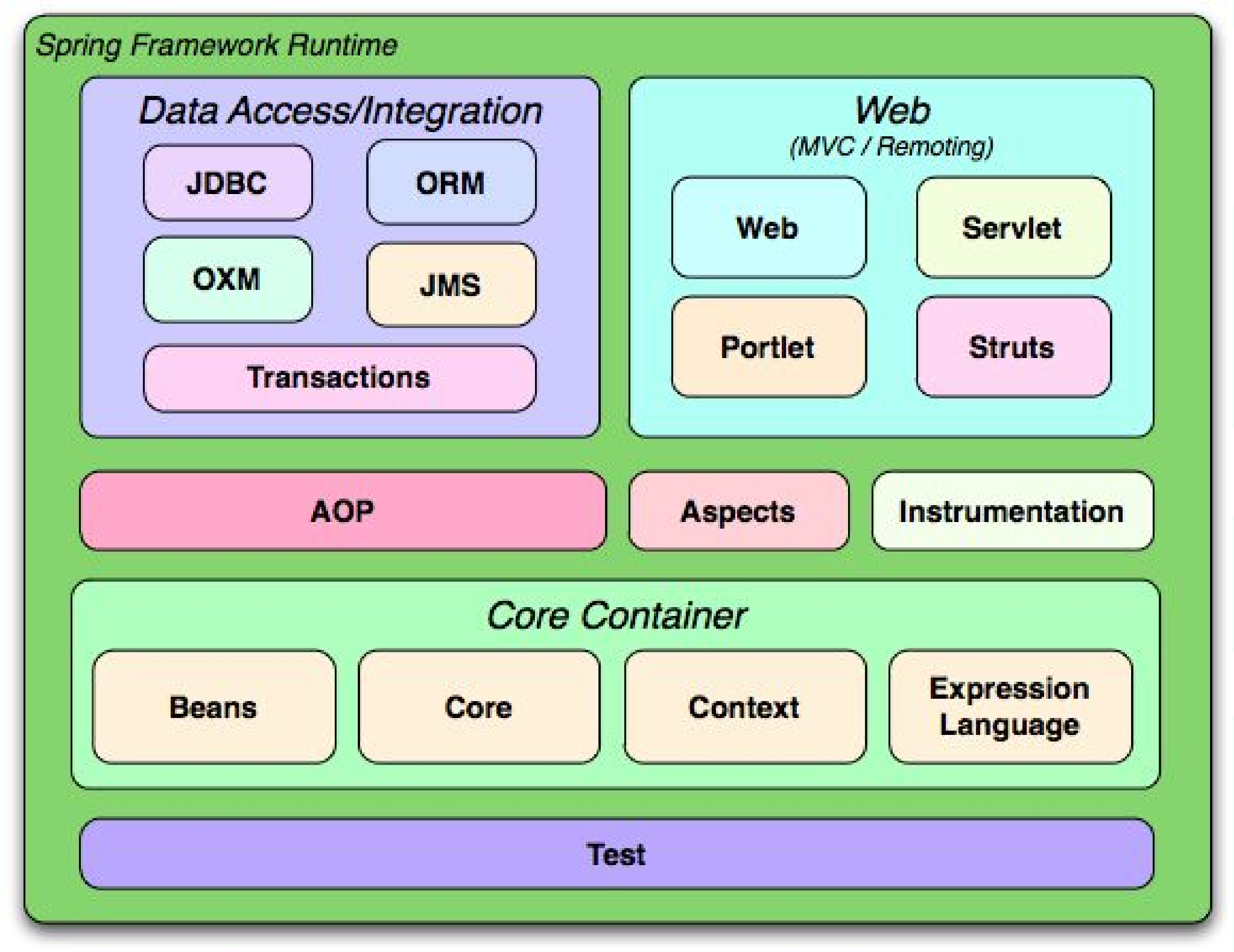
- Automatic inspection of Spring-managed beans
  - When a dependency of a bean on another bean is detected, it is resolved by the IoCC

- Mark a field as @Autowired (Spring-specific) or @Inject (Java standard)

Annotation-Based Configuration

- @Component
  - Identifies a generic Spring-managed bean
- @Service, @Controller and @Repository are specialization of @Component for future use
  - @Repository identifies a DAO
  - @Service annotates beans of the service layer (i.e. controllers in MVC)
  - @Controller annotates beans of the presentation layer (i.e. the layer between web view and service layer, e.g. the one managing navigation among pages)

Spring Framework Overview



Spring Framework Runtime

Data Access/Integration

JDBC

ORM

OXM

JMS

Transactions

Web

(MVC / Remoting)

Web

Servlet

Portlet

Struts

AOP

Aspects

Instrumentation

Core Container

Beans

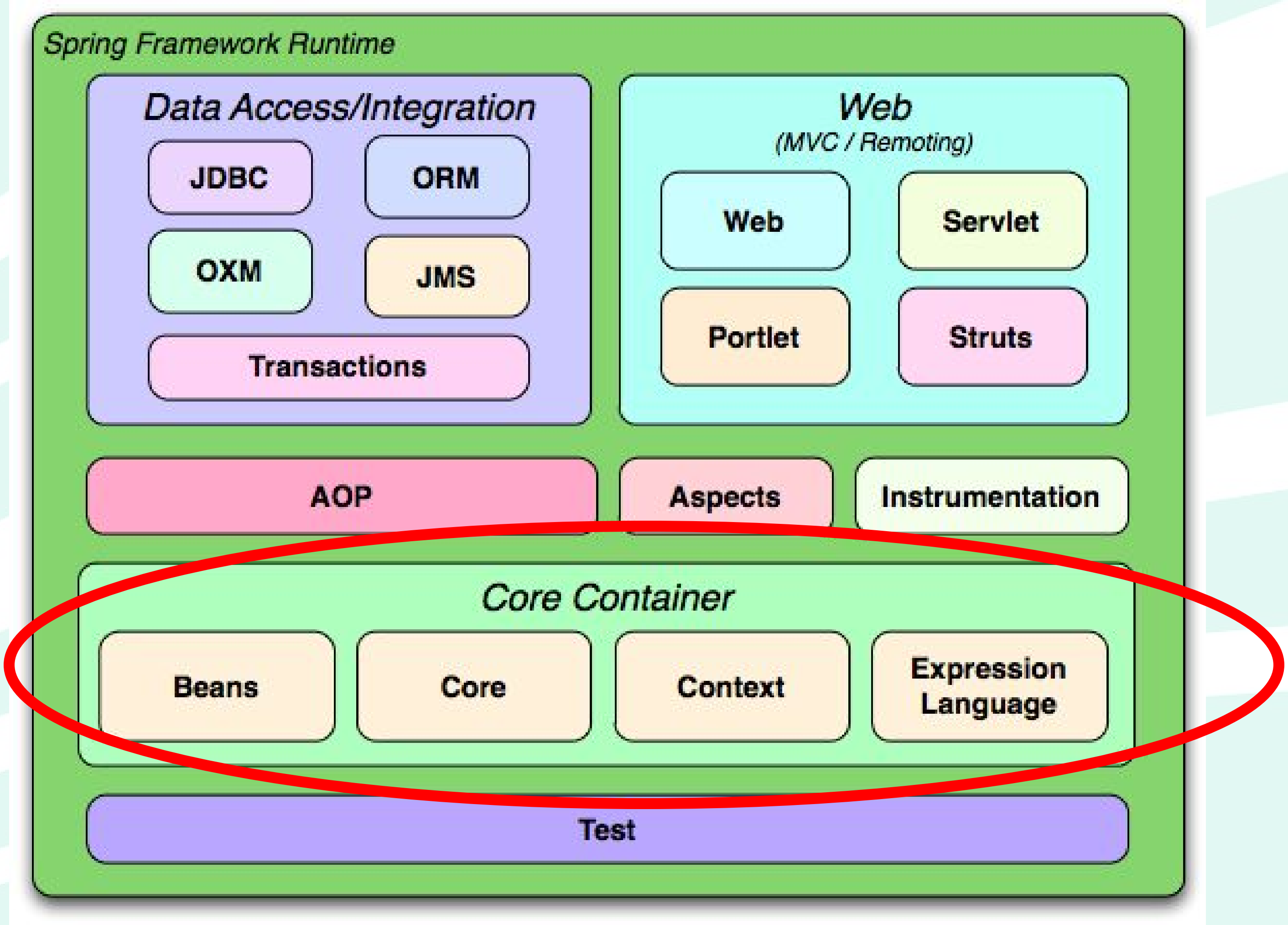
Core

Context

Expression Language

Test

Spring Framework Overview

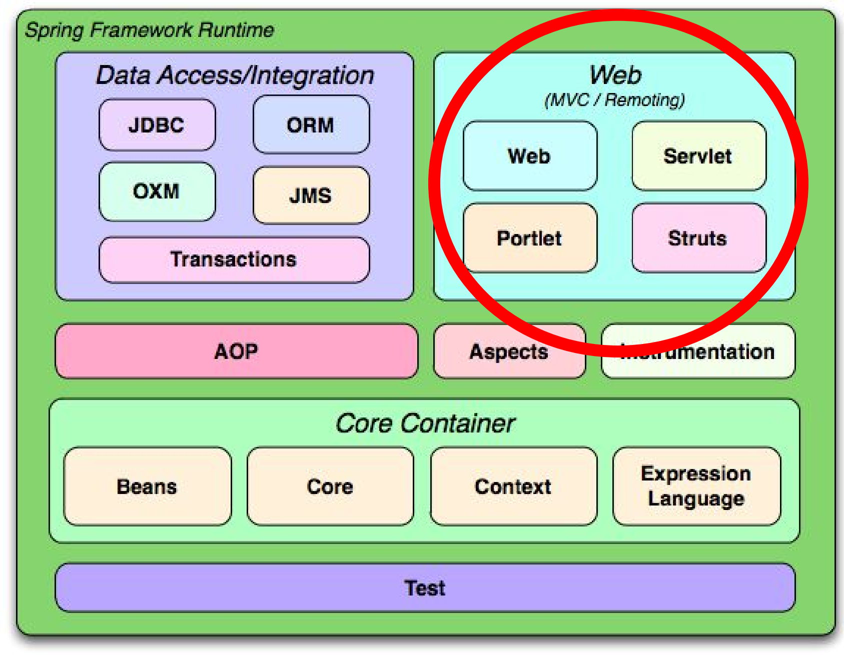


Spring Framework Overview

- Core Container
  - Beans
    - Bean definitions and management
  - Core
    - Inversion of Control Container and Dependency Injection features
      - BeanFactory is the main interface
  - Context
    - Java EE features for framework-managed objects
      - ApplicationContext is the main interface
  - Expression Language
    - Querying and manipulating framework-managed objects at runtime

Spring Web

- Web
  - Features for multipart file management, web services…
- Servlet
  - Spring's MVC implementation
- Portlet
- Struts



© Manuel Mastrofini

Spring MVC

- Spring component to support the development of web applications
- Web applications require
  - Dispatcher servlet
    - Server-side component that intercepts web requests and decides the Spring controller that will manage each request
  - Handler Mappings
    - Configuration to bridge the Dispatcher servlet and controllers
  - Controller
    - Java class and Spring bean that processes requests and produce valuable output
  - GUI resources (View)
    - E.g. HTML pages, CSS, Javascript
  - View resolver
    - Mediator between controllers and views to select which physical GUI resources are used to render certain outputs

Spring MVC Annotations

- @RequestMapping
  - Maps a URL to a method of a Controller class to execute when opening such URL
- @RequestParameter
  - POST parameter sent by the client and embodied in the HTTP request
- @PathVariable
  - GET parameter sent by the client
- @ResponseBody
  - Return parameter serialized by the server and embodied in the HTTP response

Spring REST

- REST
  - REpresentational
  - State
  - Transfer
- Main REST constraints
  - Client server (on the web)
  - Stateless (no state stored between requests)
  - Uniform interface for communication
- @RestController annotations is the same as @Controller + @ResponseBody for all methods

JSON

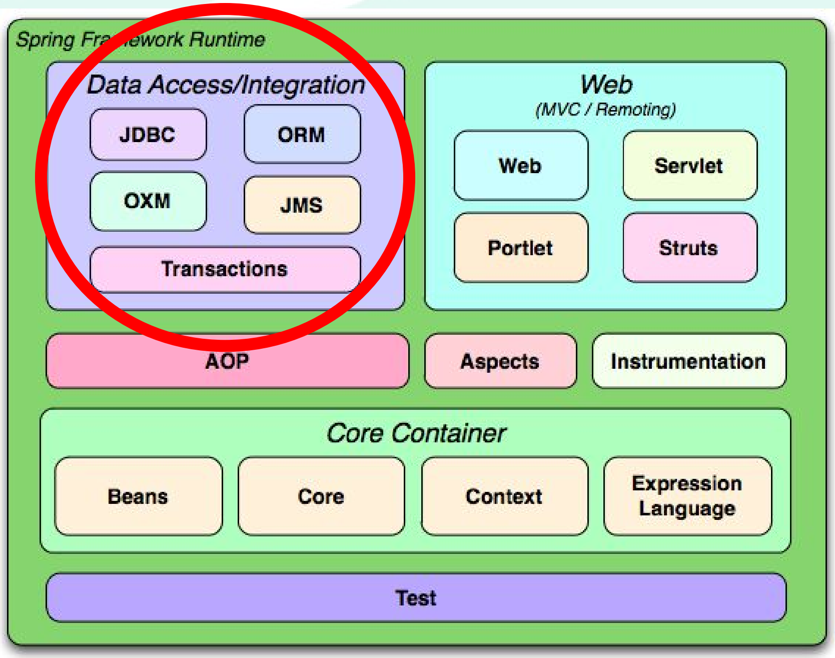
- Javascript Object Notation
- Open standard to exchange data between applications
- Used to exchange data between server and client of a web application
  - Alternative to XML
- Data types: number, string, boolean, array and complex object
  - null as special value

JSON Example

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 25,
  "height_cm": 167.6,
   "address": {
       "streetAddress": "21 2nd Street",
       "city": "New York",
       "state": "NY",
       "postalCode": "10021-3100"
     },
  "phoneNumbers": [
       {
         "type": "home",
         "number": "212 555-1234"
       },
       {
         "type": "office",
         "number": "646 555-4567"
       }
     ],
   "children": [],
   "spouse": null
}
```

Spring Data

- Data Access/Integration
  - JDBC
    - Abstraction layer from vendor-specific coding (e.g. exceptions)
  - ORM
    - Integration with popular Object-Relational mapping APIs, e.g. Hibernate
  - OXM
    - Integration with popular Object-XML mapping APIs, e.g. JAXB
  - JMS
    - Features for message exchange
  - Transactions
    - Feature for declarative and programmatic transactions management



© Manuel Mastrofini

Spring Data Annotations

- @Repository
  - Mark a class/interface as DAO
  - Can be a class
    - Implement JPARepository and define custom methods
      - Leverage the EntityManager
      - Leverage ORM specific features
  - Can be an interface
    - Define operations according to some “convention”
    - Obtain their implementations automatically
      - Generated and provided by Spring
      - E.g. findByUsernameAndPassword(String username, String password)
      - E.g. findByNameLike(String nameLike)

Spring Data

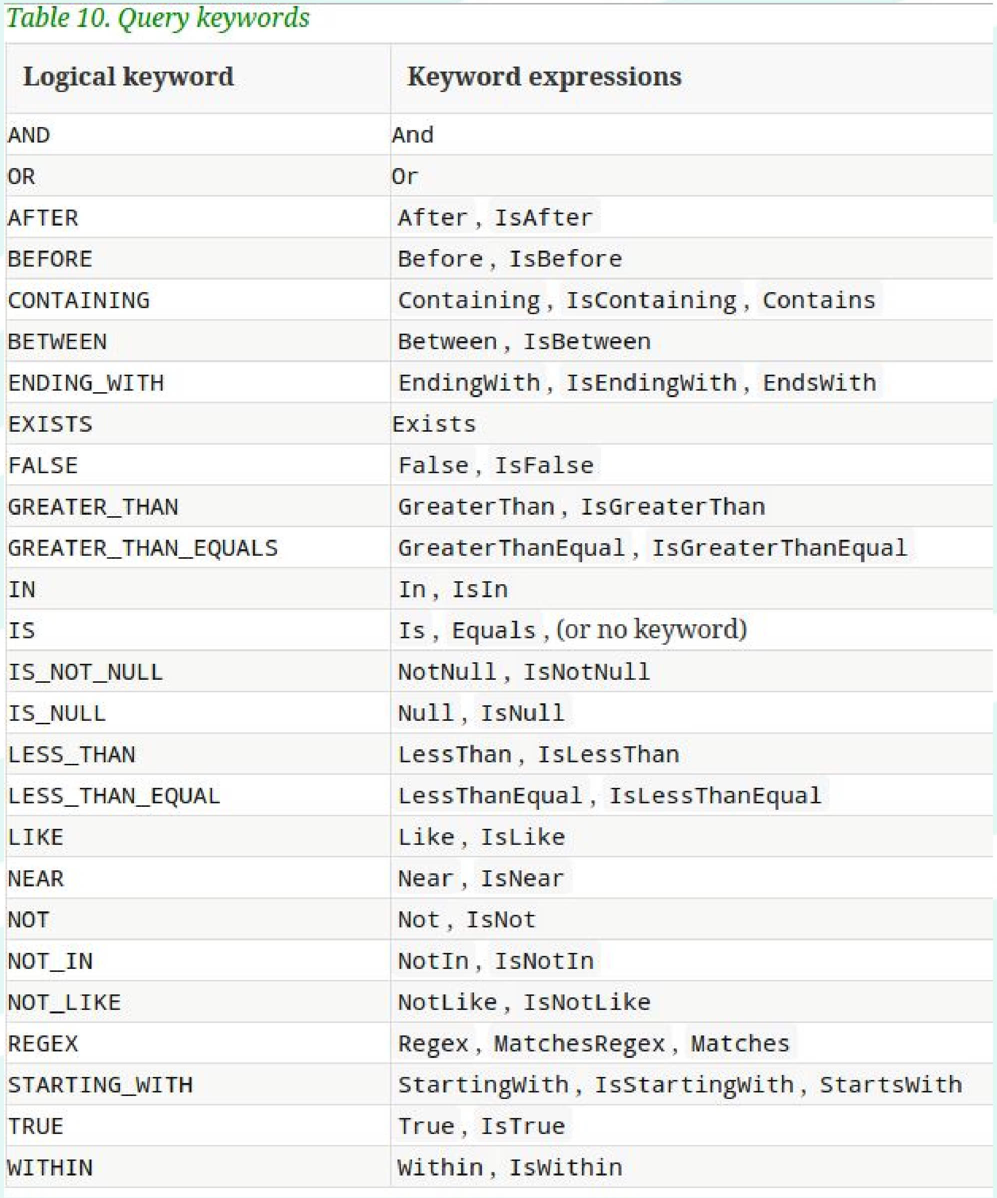
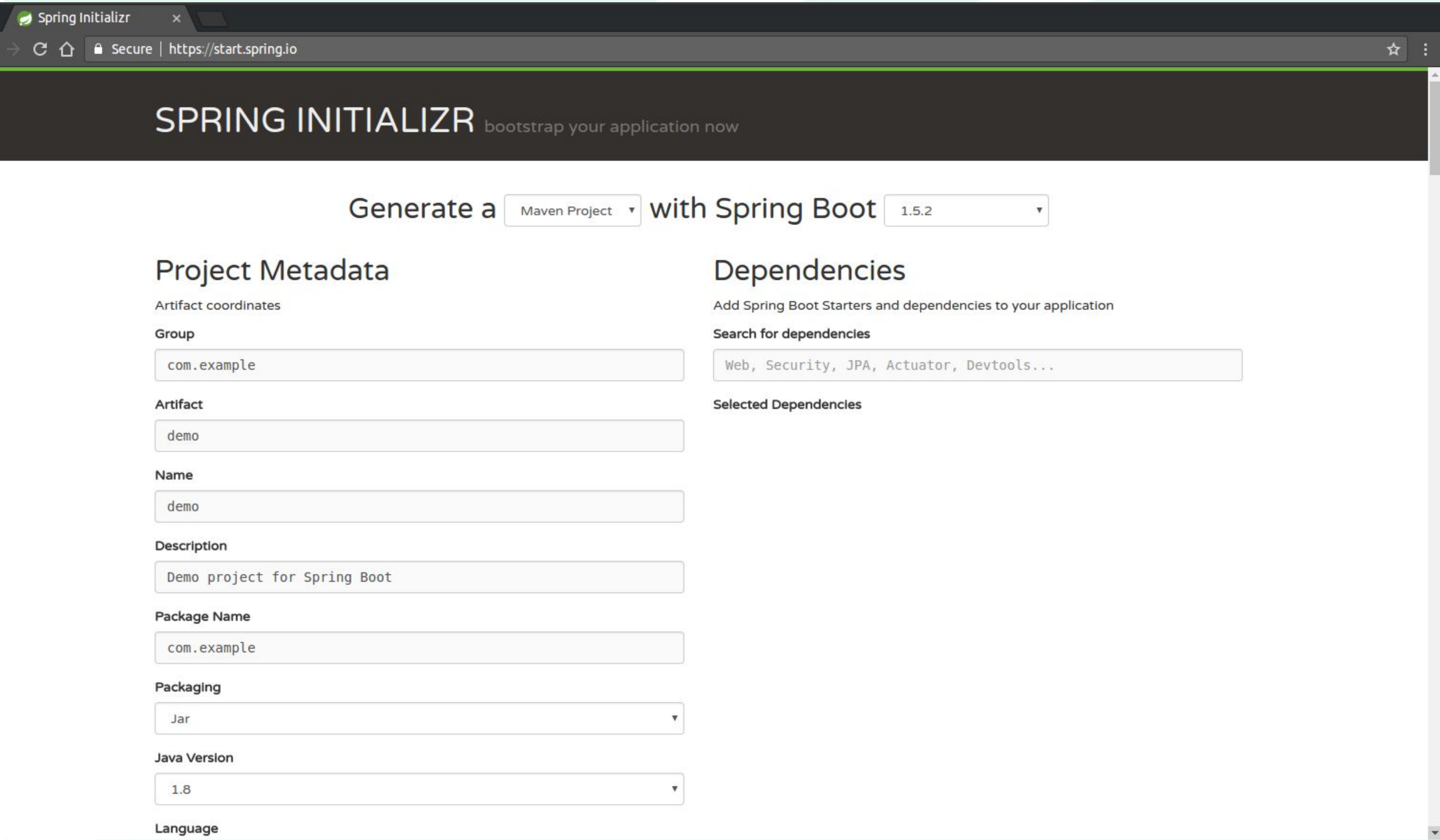


Table 10. Query keywords

Logical keyword	Keyword expressions
AND	And
OR	Or
AFTER	After , IsAfter
BEFORE	Before , IsBefore
CONTAINING	Containing , IsContaining , Contains
BETWEEN	Between , IsBetween
ENDING_WITH	EndingWith , IsEndingWith , EndsWith
EXISTS	Exists
FALSE	False , IsFalse
GREATER_THAN	GreaterThan , IsGreaterThan
GREATER_THAN_EQUALS	GreaterThanEqual , IsGreaterThanEqual
IN	In , IsIn
IS	Is , Equals , (or no keyword)
IS_NOT_NULL	NotNull , IsNotNull
IS_NULL	Null , IsNull
LESS_THAN	LessThan , IsLessThan
LESS_THAN_EQUAL	LessThanEqual , IsLessThanEqual
LIKE	Like , IsLike
NEAR	Near , IsNear
NOT	Not , IsNot
NOT_IN	NotIn , IsNotIn
NOT_LIKE	NotLike , IsNotLike
REGEX	Regex , MatchesRegex , Matches
STARTING_WITH	StartingWith , IsStartingWith , StartsWith
TRUE	True , IsTrue
WITHIN	Within , IsWithin

Spring Boot



SPRING INITIALIZR bootstrap your application now

Generate a Maven Project with Spring Boot 1.5.2

Project Metadata

Artifact coordinates

Group: com.example

Artifact: demo

Name: demo

Description: Demo project for Spring Boot

Package Name: com.example

Packaging: Jar

Java Version: 1.8

Language

Dependencies

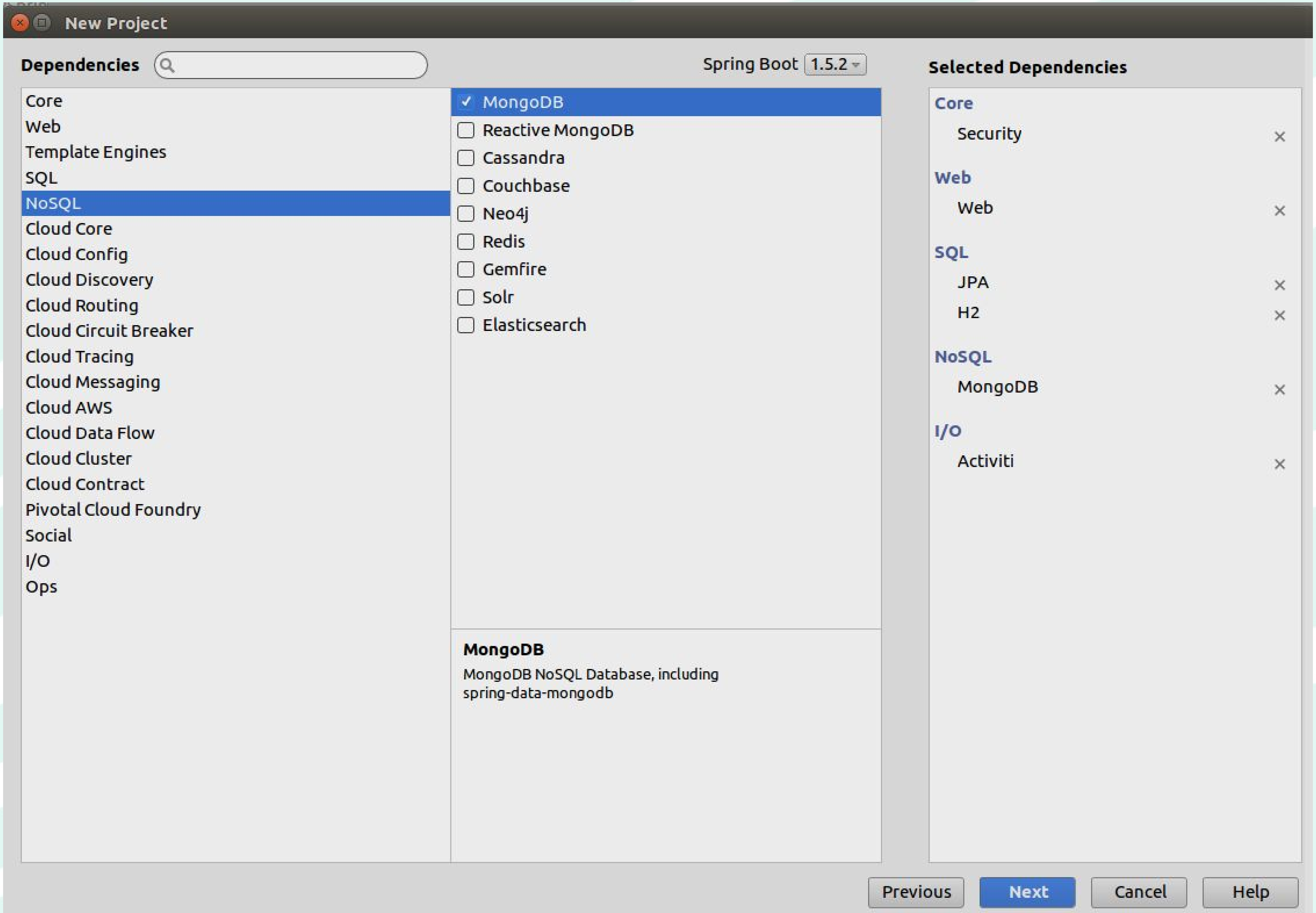
Add Spring Boot Starters and dependencies to your application

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

Selected Dependencies

© Manuel Mastrofini

Spring Boot



Dependencies

Spring Boot 1.5.2

- Core
- Web
- Template Engines
- SQL
- NoSQL
- Cloud Core
- Cloud Config
- Cloud Discovery
- Cloud Routing
- Cloud Circuit Breaker
- Cloud Tracing
- Cloud Messaging
- Cloud AWS
- Cloud Data Flow
- Cloud Cluster
- Cloud Contract
- Pivotal Cloud Foundry
- Social
- I/O
- Ops

☑ MongoDB
□ Reactive MongoDB
□ Cassandra
□ Couchbase
□ Neo4j
□ Redis
□ Gemfire
□ Solr
□ Elasticsearch

MongoDB

MongoDB NoSQL Database, including spring-data-mongodb

Selected Dependencies

Core
- Security ×

Web
- Web ×

SQL
- JPA ×
- H2 ×

NoSQL
- MongoDB ×

I/O
- Activiti ×