Fast Track to Spring Boot

Intermediate Spring Boot REST Services - 4 Day

Overview

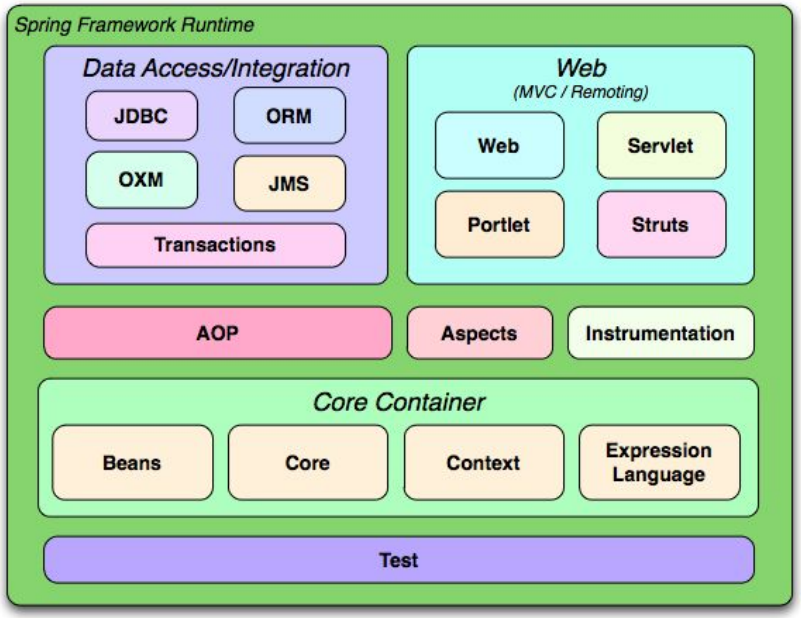
- Introductions
  - Instructor - Geoff Matrangola - geoff@matrangola.com @triglm
  - Company - DevelopIntelligence http://www.developintelligence.com/ (show 2 slides)
  - Students - Names, Current projects, Class Expectations
  - Course - How to develop a Rest API Using Spring
- Logistics
  - Start, end, break times
  - Facilities
- Class Agenda
  - See Class outline
- Class Flow
  - Slides
  - Demo
  - Lab

What is Spring Boot?

- Java based framework for stand-alone applications
- Rich set of libraries that can be integrated into your application
- Opinionated starter libraries (Maven Repos)
- Configuration by convention and automation
- Java Annotations
- Embedded Tomcat
- Easy Database configuration
- Spring - Dependency Injection and Inversion of Control

Key Elements of the Spring Framework

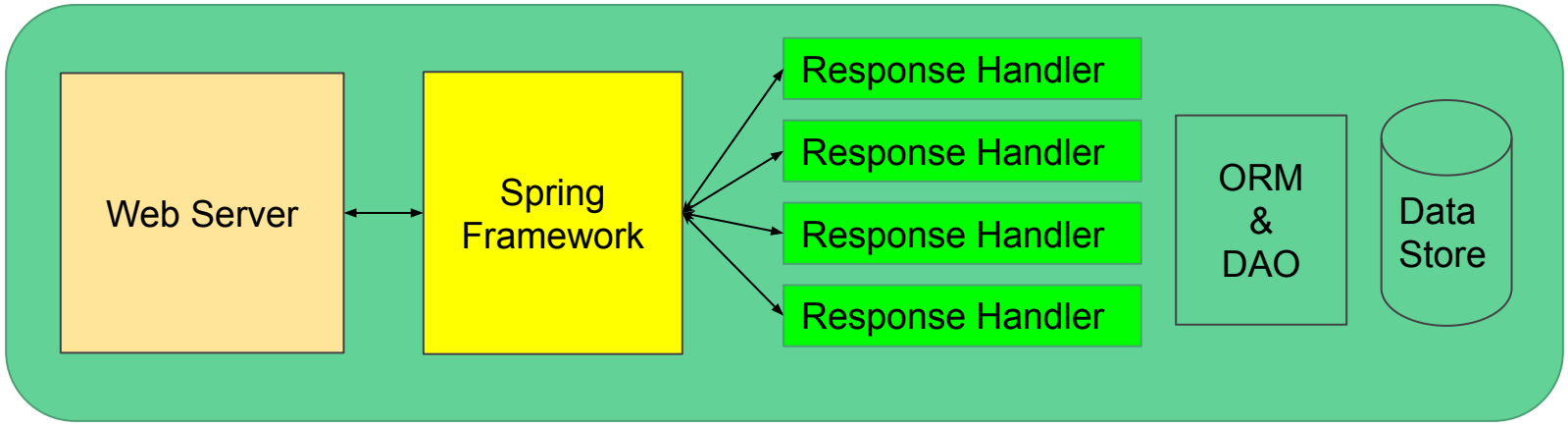
- Modules
- Core Container
- Beans
- Context
- AOP
- Other - Data, Web, Test, Instrumentation



https://docs.spring.io/spring/docs/3.0.x/spring-framework-reference

Inversion of Control (IoC)

- The framework maintains the flow of execution & setting object dependencies
- You wire in the custom business routines
- You define the objects
- You are provided objects with all their properties wired up.
- Request protocol handled by Spring and the Web Server- you write the response handler



Dependency Injection

- Objects define their dependencies ONLY
  - Constructor Arguments
  - Factory Method Arguments
  - Properties, set by Factory Method
- The container *injects* the *dependencies* when it creates the object instance
- Objects that are managed in this way are called **Spring Beans**
- **Spring Beans** are instantiated, and managed by the Spring IoC Container.

https://docs.spring.io/spring/docs/3.0.0.M3/reference/html/ch04s03.html

Spring Bean Scope

Scope	Description
singleton	Scopes a single bean definition to a single object instance per Spring IoC container.
prototype	Scopes a single bean definition to any number of object instances.
request	Scopes a single bean definition to the lifecycle of a single HTTP request; that is each and every HTTP request will have its own instance of a bean created off the back of a single bean definition. Only valid in the context of a web-aware SpringApplicationContext.
session	Scopes a single bean definition to the lifecycle of a HTTP Session. Only valid in the context of a web-aware SpringApplicationContext.
global session	Scopes a single bean definition to the lifecycle of a global HTTP Session. Typically only valid when used in a portlet context. Only valid in the context of a web-aware Spring ApplicationContext.

Demo/Lab 1
Setup & RestController

Demo/Lab 1: Hello World REST Web Service

- Simple lab to verify your configuration
- Using Spring Initializer to build base project
- Incremental development to bring explore concepts of the Spring Boot throughout the entire class.
- REST service responds with JSON
- Intellij, Gradle, Spring Boot, Tomcat, etc.

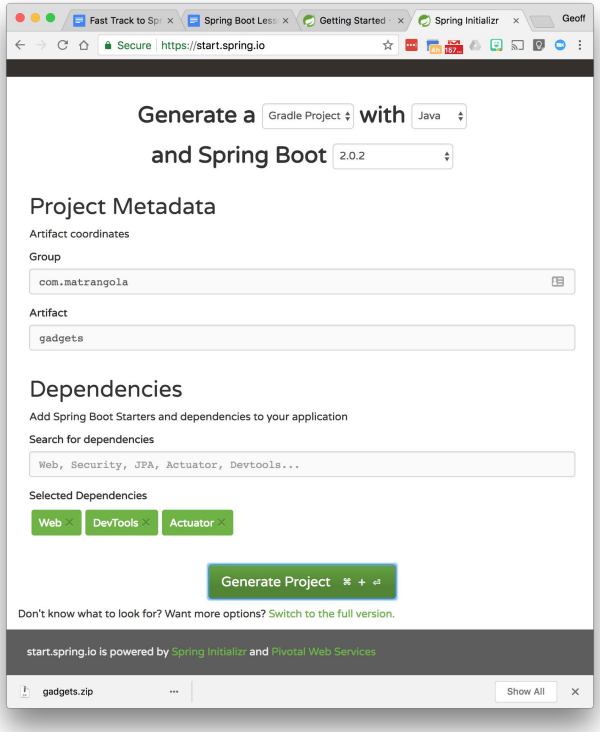
Setup

- Intellij Idea 2018.1.2
- Java JDK 8
- Chrome Web Browser
- MySQL
- Postman to verify REST
- Internet Access

Spring Initializr

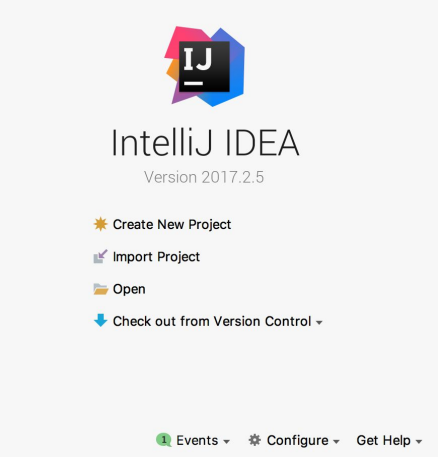
https://start.spring.io/

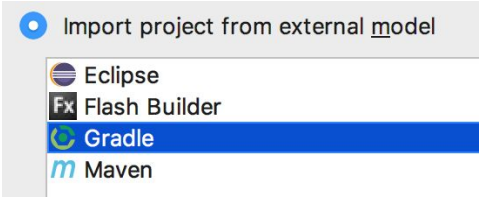
- Gradle Project
- Java
- 2.0.2
- Group: com.whatever
- Artifact: gadgets
- Dependencies: Web, Actuator, DevTools
- Download
- Unzip



Import Part 1

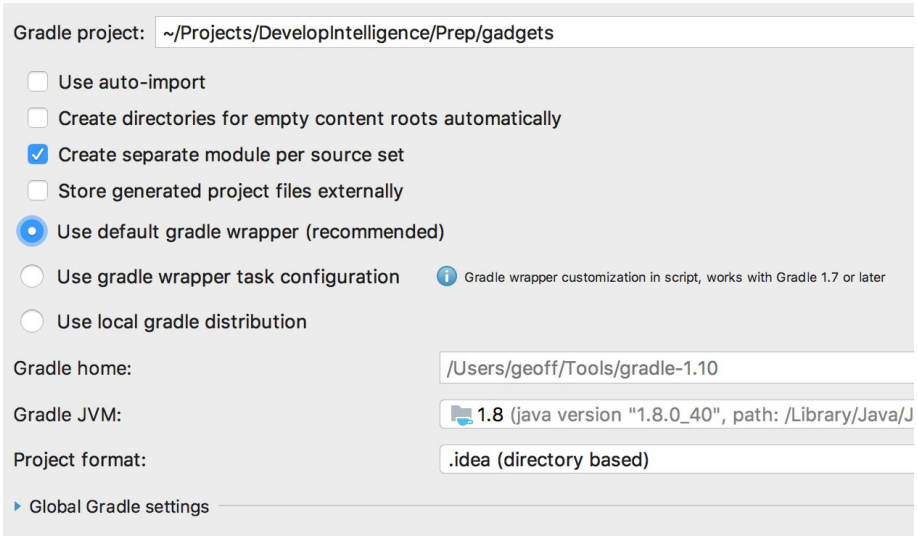
- Launch IntelliJ Idea
- Import Project
- Select Downloaded & Unzipped Directory
- Select Import project...
- Gradle





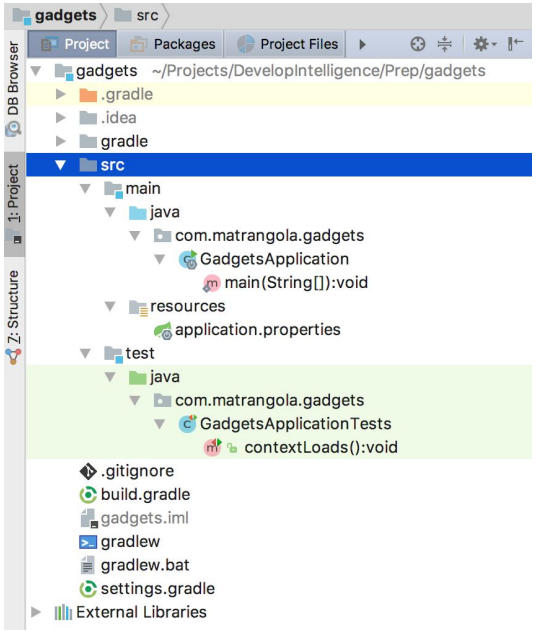
Import Part 2

- Gradle project: ~/your/project/dir
- Create separate module...
- Use default gradle wrapper
- Finish



Project Structure

- .idea - IDE stuff
- gradle - automated build stuff
- src - Java and Resources
- build.gradle - build configuration
- Other files



Annotations Used in Demo

@RestController - Identify the Rest Controller for the Framework

@RequestMapping - Path of the URL mapped from the web server to the code

@RequestParam - Request params in the URL mapped to method parameters

Live Demo

```
package com.matrangola.gadgets.data.model;

public class User {
  private String firstName;
  private String lastName;

  public String getFirstName() {
    return firstName;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public String getLastName() {
    return lastName;
  }

  public void setLastName(String lastName) {
    this.lastName = lastName;
  }
}
```

```
@RestController
public class UserController {
  @RequestMapping("/makeUser")
  public User greeting(@RequestParam(value="last") String lastName,
                       @RequestParam(value="first") String firstName) {
    User user = new User();
    user.setFirstName(firstName);
    user.setLastName(lastName);
    return user;
  }
}
```