Spring Boot

Features

Opinionated

It by default include what we will need without giving us option not to

Conventions over Configuration

Most of the required configuration is already done internally. Spring application conventions

Standalone

Provides us builtin web container and applications can be started from the command line

Production Ready

Applications are production ready, not just for dev testing and trying different features.

Bill-of-materials

Spring boot groups/configures all required and compatible libraries together by using gradle, or maven's <parent> or <dependencyManagement>

Embedded Tomcat Server

Spring boot web application comes with embedded tomcat server. The benefits it provides:

- Convenience
- Tomcat servlet container configurations are now application configs
- Standalone application
- Useful for microservices

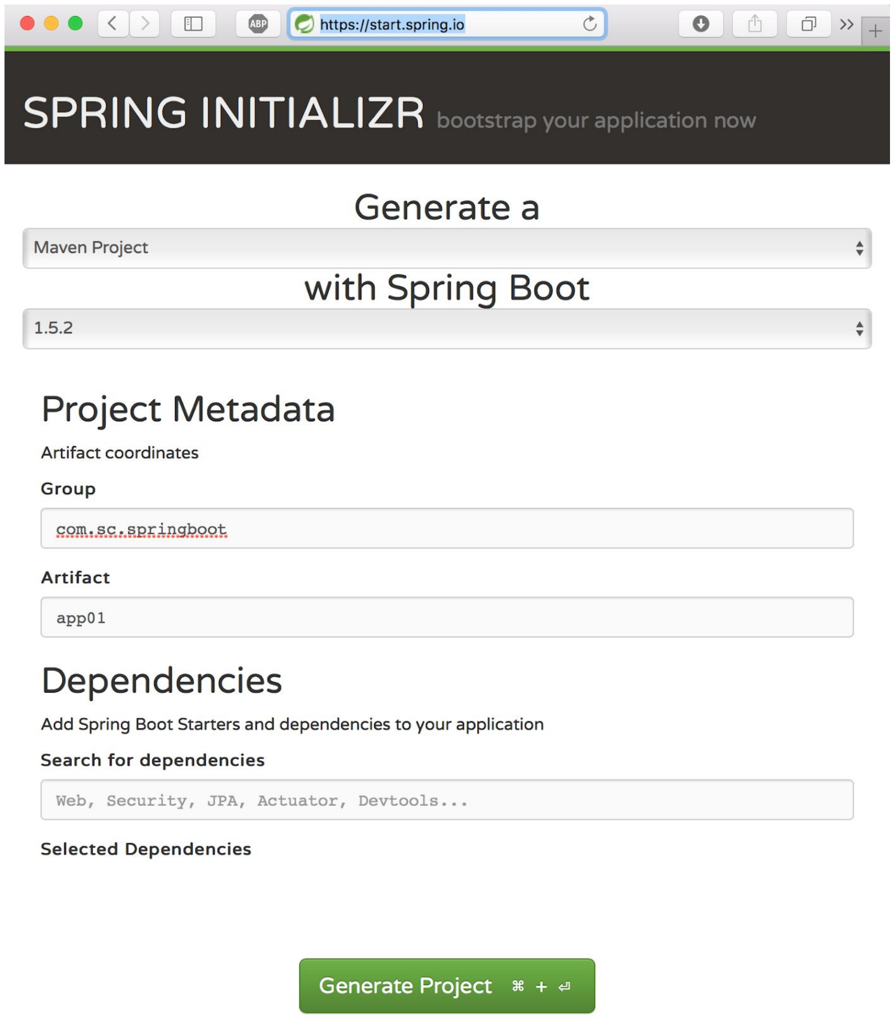
Create Spring Boot Project

We could manually create spring boot project by creating simple maven project and adding spring boot parent pom, starter dependency, and calling SpringApplication.*run*();.

There are some convenience methods to create Spring boot application:

- Spring CLI
  https://docs.spring.io/spring-boot/docs/current/reference/html/getting-started-installing-spring-boot.html
  https://docs.spring.io/spring-boot/docs/current/reference/html/cli-using-the-cli.html
- STS
  https://spring.io/tools/sts
  https://spring.io/blog/2015/03/18/spring-boot-support-in-spring-tool-suite-3-6-4
- Spring Initializr
  https://start.spring.io

Go to the website below make your selections and click generate button
https://start.spring.io



Generate a
Maven Project
with Spring Boot
1.5.2

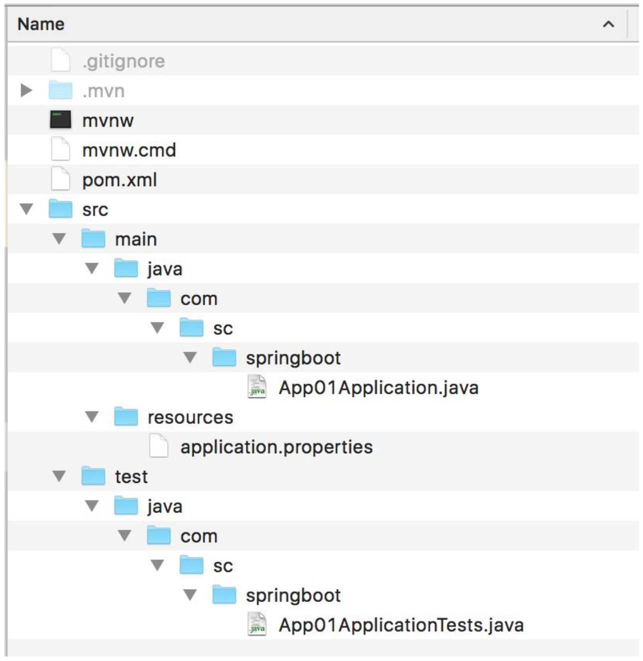
Project Metadata
Artifact coordinates
Group
com.sc.springboot
Artifact
app01

Dependencies
Add Spring Boot Starters and dependencies to your application
Search for dependencies
Web, Security, JPA, Actuator, Devtools...
Selected Dependencies

The above will download a maven project with the following files. It includes

- Maven wrapper
- pom.xml

- application.properties
- Application.java with the main method
- ApplicationTests.java with default unittest for Application.java



```
Name
.gitignore
.mvn
mvnw
mvnw.cmd
pom.xml
src
  main
    java
      com
        sc
          springboot
            App01Application.java
    resources
      application.properties
  test
    java
      com
        sc
          springboot
            App01ApplicationTests.java
```

pom.xml

Parent

We need to add spring-boot-starter-parent as parent of out maven project. This do common Spring configurations

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>1.5.2.RELEASE</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
```

If you don't want to create spring-boot-starter-parent as parent of your project then use this: http://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-build-systems.html#using-boot-maven-without-a-parent

Java Version

Java version can be specified in maven properties

```
<properties>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <java.version>1.8</java.version>
</properties>
```

Starter Dependency

We need to add spring boot starter dependency. This will include all the jars for the type of project we want to create.

List of all starter dependencies:
https://mvnrepository.com/artifact/org.springframework.boot

Ideally a spring boot project will have one starter and one starter test dependency.

E.g. If we want to create web project and add spring test support we will add:

```
<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>
```

Spring boot plugin

Optionally we can add spring boot plugin to run application from maven instead of running main method class

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

Running Application

Running from Maven

If spring-boot-maven-plugin is added then we run application from maven

```
$ mvn spring-boot:run
```

To pass command line arguments to maven command

```
$ mvn spring-boot:run -Drun.arguments="arg1,arg2"
```

Running from command line

We can run built jar from command line

```
$ mvn clean install
$ cd target
$ java -jar app01-0.0.1-SNAPSHOT.jar
```

Access running application http://localhost:8080

Bootstrapping Spring Boot Application

To bootstrap spring boot application:

- Add @SpringBootApplication annotation on any class.
- Run static method SpringApplication.*run*(App01Application.**class**, args);. Give the class

```
package com.sc.springboot;

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

@SpringBootApplication
public class App01Application {

  public static void main(String[] args) {
    SpringApplication.run(App01Application.class, args);
  }
}
```

The above 2 steps will do the following:

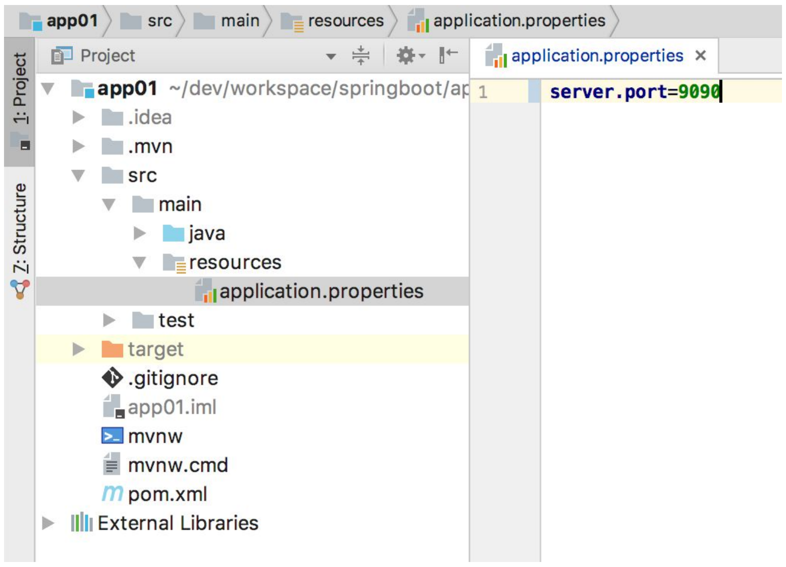
- Setup default configuration
- Start Spring application context

- Perform classpath scan
- Start Tomcat server

application.properties

Spring boot application configurations can be configured or overridden by adding properties in application.properties.
Here are all the common configuration of spring boot application:
https://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html

In the example below we are changing default tomcat's port from 8080 to 9090



```
server.port=9090
```

Dev Tools

https://docs.spring.io/spring-boot/docs/current/reference/html/using-boot-devtools.html
Dev tools are used to reload/restart application while development. Application will reload/restart automatically when anything in classpath is modified.

Dev tools dependency

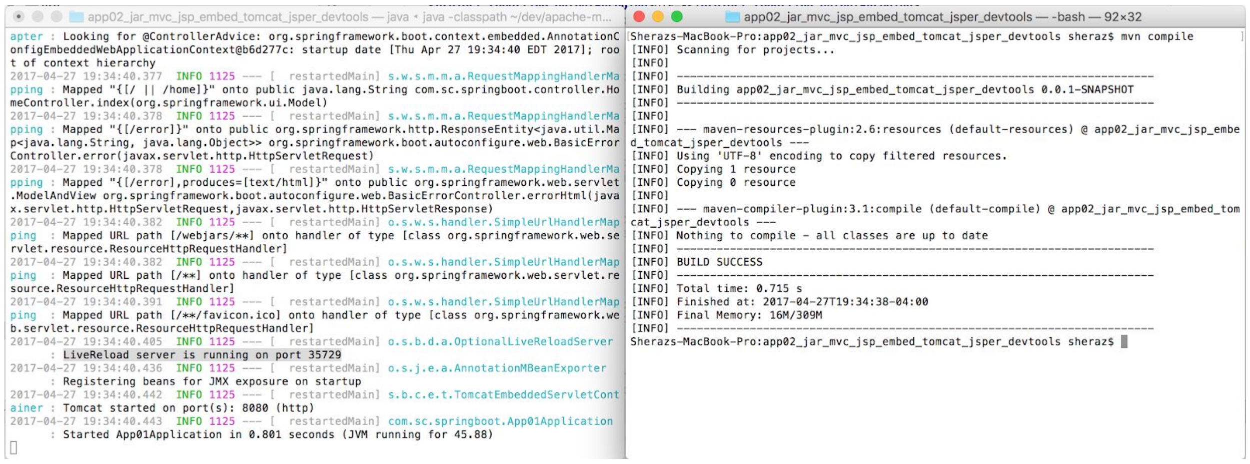
It is recommended, on spring guide, to add dev tools dependency as runtime scope and optional

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

```
</dependency>
```

Dev tools from command line

Start project with the command
$ mvn clean install spring-boot:run
Make changes to java code or anything in resources. Then update classpath by giving compiling in another terminal command
$ mvn compile



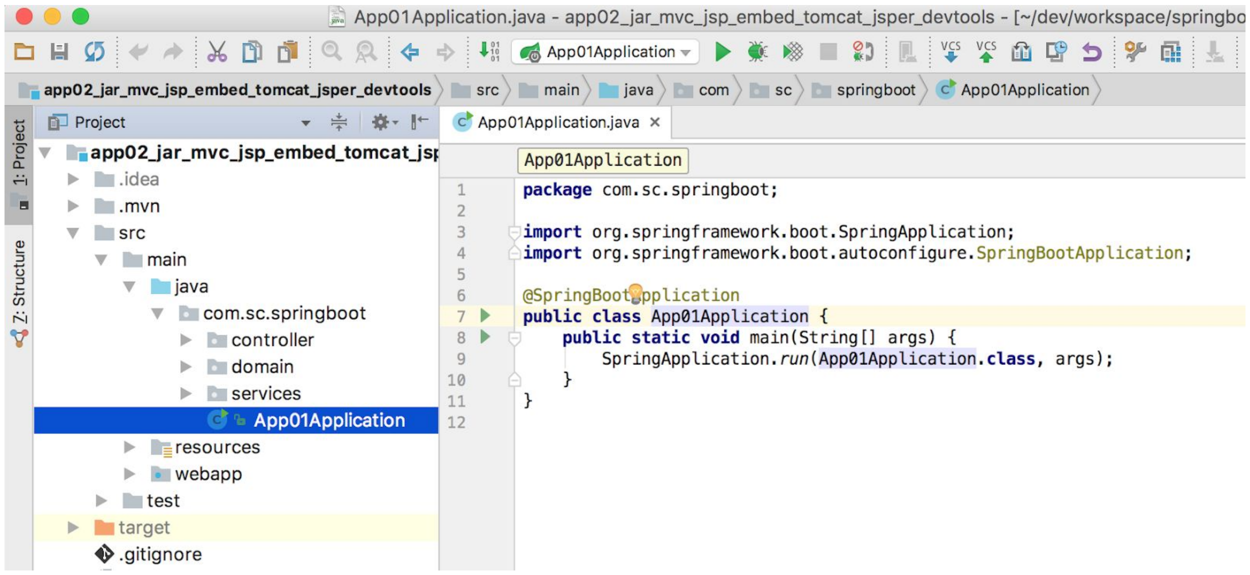
app02_jar_mvc_jsp_embed_tomcat_jsper_devtools — java ◂ java -classpath ~/dev/apache-m...
app02_jar_mvc_jsp_embed_tomcat_jsper_devtools — -bash — 92×32

```
apter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationC
onfigEmbeddedWebApplicationContext@b6d277c: startup date [Thu Apr 27 19:34:40 EDT 2017]; roo
t of context hierarchy
2017-04-27 19:34:40.377  INFO 1125 --- [  restartedMain] s.w.s.m.m.a.RequestMappingHandlerMa
pping : Mapped "{[/ || /home]}" onto public java.lang.String com.sc.springboot.controller.Ho
meController.index(org.springframework.ui.Model)
2017-04-27 19:34:40.378  INFO 1125 --- [  restartedMain] s.w.s.m.m.a.RequestMappingHandlerMa
pping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Ma
p<java.lang.String, java.lang.Object>> org.springframework.boot.autoconfigure.web.BasicError
Controller.error(javax.servlet.http.HttpServletRequest)
2017-04-27 19:34:40.378  INFO 1125 --- [  restartedMain] s.w.s.m.m.a.RequestMappingHandlerMa
pping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet
.ModelAndView org.springframework.boot.autoconfigure.web.BasicErrorController.errorHtml(java
x.servlet.http.HttpServletRequest,javax.servlet.http.HttpServletResponse)
2017-04-27 19:34:40.382  INFO 1125 --- [  restartedMain] o.s.w.s.handler.SimpleUrlHandlerMap
ping  : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.se
rvlet.resource.ResourceHttpRequestHandler]
2017-04-27 19:34:40.382  INFO 1125 --- [  restartedMain] o.s.w.s.handler.SimpleUrlHandlerMap
ping  : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.re
source.ResourceHttpRequestHandler]
2017-04-27 19:34:40.391  INFO 1125 --- [  restartedMain] o.s.w.s.handler.SimpleUrlHandlerMap
ping  : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.we
b.servlet.resource.ResourceHttpRequestHandler]
2017-04-27 19:34:40.405  INFO 1125 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer
      : LiveReload server is running on port 35729
2017-04-27 19:34:40.436  INFO 1125 --- [  restartedMain] o.s.j.e.a.AnnotationMBeanExporter
      : Registering beans for JMX exposure on startup
2017-04-27 19:34:40.442  INFO 1125 --- [  restartedMain] s.b.c.e.t.TomcatEmbeddedServletCont
ainer : Tomcat started on port(s): 8080 (http)
2017-04-27 19:34:40.443  INFO 1125 --- [  restartedMain] com.sc.springboot.App01Application
      : Started App01Application in 0.801 seconds (JVM running for 45.88)
```

```
Sherazs-MacBook-Pro:app02_jar_mvc_jsp_embed_tomcat_jsper_devtools sheraz$ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building app02_jar_mvc_jsp_embed_tomcat_jsper_devtools 0.0.1-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ app02_jar_mvc_jsp_embe
d_tomcat_jsper_devtools ---
[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) @ app02_jar_mvc_jsp_embed_tom
cat_jsper_devtools ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 0.715 s
[INFO] Finished at: 2017-04-27T19:34:38-04:00
[INFO] Final Memory: 16M/309M
[INFO] ------------------------------------------------------------------------
Sherazs-MacBook-Pro:app02_jar_mvc_jsp_embed_tomcat_jsper_devtools sheraz$
```

Dev tools from IntelliJ

Run the application's main(). After any Java or resources changes click:
Command + F9 or click "Build" -> "Build Project" or we can also configure IntelliJ to do auto build.



```
package com.sc.springboot;

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

@SpringBootApplication
public class App01Application {
    public static void main(String[] args) {
        SpringApplication.run(App01Application.class, args);
    }
}
```

Chrome Livereload

http://livereload.com/
https://chrome.google.com/webstore/detail/livereload/jnihajbhpnppcggbcgedagnkighmdlei

RESTful Application

Spring RESTful web service

By default spring boot web application comes configured to create RESTful web services application. All we have to do is add Spring Rest

```
package com.sc.springboot.controller;

import com.sc.springboot.domain.Person;
import com.sc.springboot.services.PersonService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
public class PersonRestController {

  @Autowired
  private PersonService personService;
  @RequestMapping(
          value = "/",
          method = RequestMethod.GET,
          consumes = {"text/plain", "application/*"},
          produces = "application/json; charset=UTF-8"
  )
  public List<Person> index() {
     return this.personService.getAll();
  }
}
```

Spring MVC JSP JAR Embedded Tomcat Server

NOTE: It is not recommended to run JSP in an embedded tomcat server because of this limitation:
https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-developing-web-applications.html#boot-features-jsp-limitations

By just adding spring-boot-starter-web dependency we can run spring boot from command line, that would start embedded tomcat. But it will not be capable for transpiling JSP to Servlet. To enable JSP transpilation we need to add tomcat-embed-jasper dependency.

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

View prefix and suffix

We can keep JSP anywhere in the application. And set its prefix and suffix in application.properties

```
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

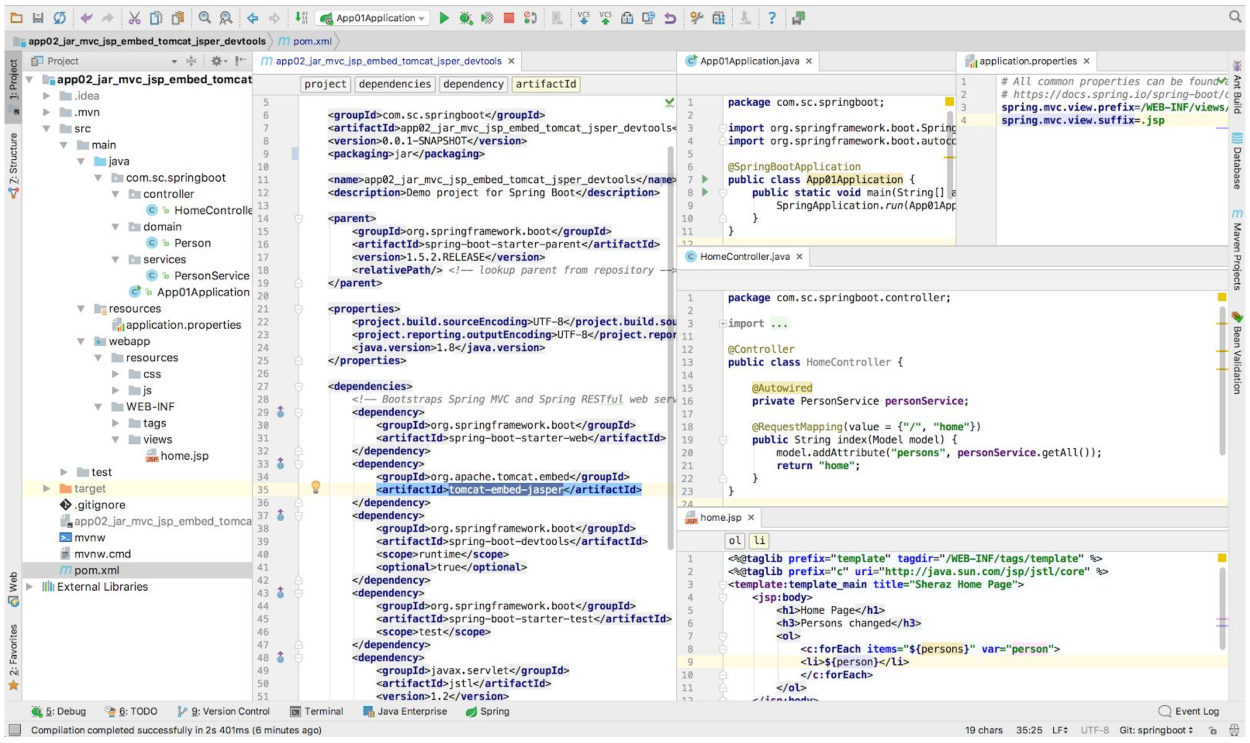
Controller

Add a controller. Controller below returns **"home"** so it be resolved to **/WEB-INF/views/home.jsp**.

```
package com.sc.springboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PersonMvcController {
  @RequestMapping(value = {"/", "home"})
  public String index() {
        return "home";
    }
}
```



```
    <groupId>com.sc.springboot</groupId>
    <artifactId>app02_jar_mvc_jsp_embed_tomcat_jsper_devtools<
    <version>0.0.1-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>app02_jar_mvc_jsp_embed_tomcat_jsper_devtools</name>
    <description>Demo project for Spring Boot</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.2.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sou
        <project.reporting.outputEncoding>UTF-8</project.repor
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <!-- Bootstraps Spring MVC and Spring RESTful web serv
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.tomcat.embed</groupId>
            <artifactId>tomcat-embed-jasper</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jstl</artifactId>
            <version>1.2</version>
```

```
package com.sc.springboot;

import org.springframework.boot.Spring
import org.springframework.boot.autoco

@SpringBootApplication
public class App01Application {
    public static void main(String[] a
        SpringApplication.run(App01App
    }
}
```

```
# All common properties can be found a
# https://docs.spring.io/spring-boot/c
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

```
package com.sc.springboot.controller;

import ...

@Controller
public class HomeController {

    @Autowired
    private PersonService personService;

    @RequestMapping(value = {"/", "home"})
    public String index(Model model) {
        model.addAttribute("persons", personService.getAll());
        return "home";
    }
}
```

```
<%@taglib prefix="template" tagdir="/WEB-INF/tags/template" %>
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<template:template_main title="Sheraz Home Page">
    <jsp:body>
        <h1>Home Page</h1>
        <h3>Persons changed</h3>
        <ol>
            <c:forEach items="${persons}" var="person">
                <li>${person}</li>
            </c:forEach>
        </ol>
```

Compilation completed successfully in 2s 401ms (6 minutes ago)

Spring MVC JSP WAR External Tomcat Server

To run spring boot application in external Tomcat we need to create a executable WAR.

pom.xml

`<packaging>war</packaging>` need to be of war type and we don't need to add tomcat-embed-jasper dependency:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.sc.springboot</groupId>
    <artifactId>app02_war_mvc_jsp_external_tomcat</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <name>app01</name>
    <description>Demo project for Spring Boot</description>
    <parent>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.2.RELEASE</version>
        <relativePath/>
    </parent>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>
    <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>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Initializing Servlets in Spring MVC

Since external Tomcat server will not run application's main(), so startup class annotated @SpringBootApplication could be extended by SpringBootServletInitializer class and implement configure() method to start.

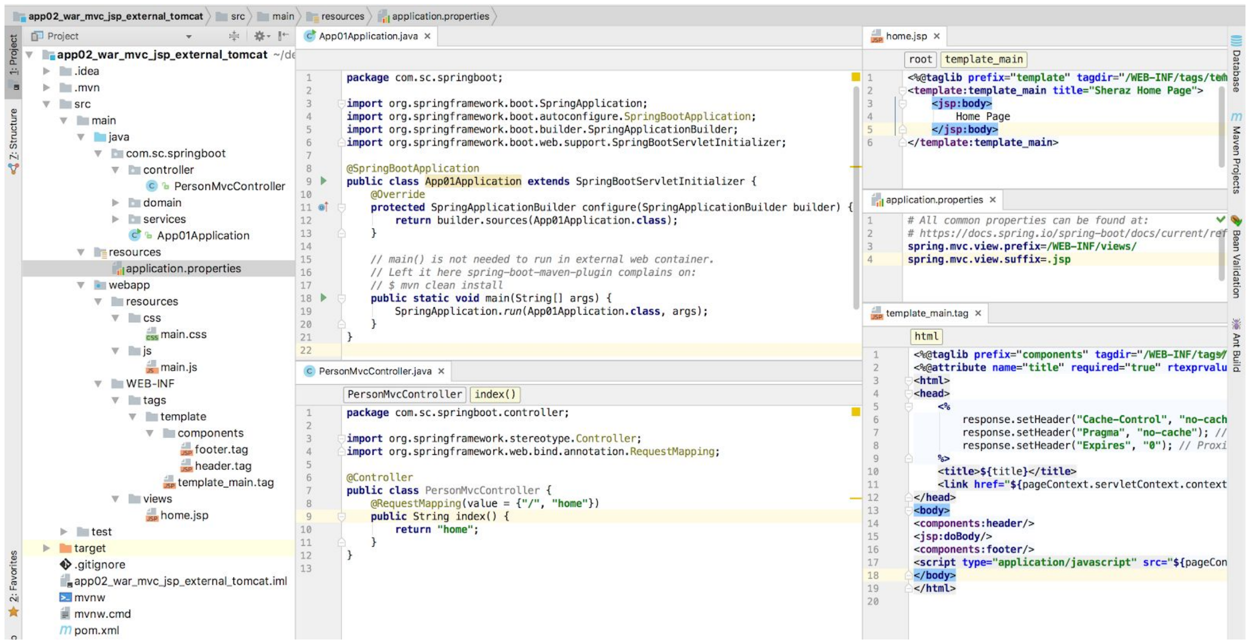
NOTE: main() is not needed to run in external web container. Left it here because spring-boot-maven-plugin complains on:
$ mvn clean install

```java
package com.sc.springboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.support.SpringBootServletInitializer;
```

```
@SpringBootApplication
public class App01Application extends SpringBootServletInitializer {
 @Override
 protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
   return builder.sources(App01Application.class);
 }

 // main() is not needed to run in external web container.
 // Left it here spring-boot-maven-plugin complains on:
 // $ mvn clean install
 public static void main(String[] args) {
   SpringApplication.run(App01Application.class, args);
 }
}
```



```
app02_war_mvc_jsp_external_tomcat ~/de
  .idea
  .mvn
  src
    main
      java
        com.sc.springboot
          controller
            PersonMvcController
          domain
          services
          App01Application
      resources
        application.properties
      webapp
        resources
          css
            main.css
          js
            main.js
        WEB-INF
          tags
            template
              components
                footer.tag
                header.tag
              template_main.tag
          views
            home.jsp
    test
  target
  .gitignore
  app02_war_mvc_jsp_external_tomcat.iml
  mvnw
  mvnw.cmd
  pom.xml
```

App01Application.java

```
package com.sc.springboot;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.builder.SpringApplicationBuilder;
import org.springframework.boot.web.support.SpringBootServletInitializer;

@SpringBootApplication
public class App01Application extends SpringBootServletInitializer {
    @Override
    protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
        return builder.sources(App01Application.class);
    }

    // main() is not needed to run in external web container.
    // Left it here spring-boot-maven-plugin complains on:
    // $ mvn clean install
    public static void main(String[] args) {
        SpringApplication.run(App01Application.class, args);
    }
}
```

PersonMvcController.java

PersonMvcController | index()

```
package com.sc.springboot.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
public class PersonMvcController {
    @RequestMapping(value = {"/", "home"})
    public String index() {
        return "home";
    }
}
```

home.jsp

root | template_main

```
<%@taglib prefix="template" tagdir="/WEB-INF/tags/tem
<template:template_main title="Sheraz Home Page">
    <jsp:body>
        Home Page
    </jsp:body>
</template:template_main>
```

application.properties

```
# All common properties can be found at:
# https://docs.spring.io/spring-boot/docs/current/ref
spring.mvc.view.prefix=/WEB-INF/views/
spring.mvc.view.suffix=.jsp
```

template_main.tag

html

```
<%@taglib prefix="components" tagdir="/WEB-INF/tags/
<%@attribute name="title" required="true" rtexprvalu
<html>
<head>
    <%
        response.setHeader("Cache-Control", "no-cach
        response.setHeader("Pragma", "no-cache"); //
        response.setHeader("Expires", "0"); // Proxi
    %>
    <title>${title}</title>
    <link href="${pageContext.servletContext.context
</head>
<body>
<components:header/>
<jsp:doBody/>
<components:footer/>
<script type="application/javascript" src="${pageCon
</body>
</html>
```

Listing all Beans for debugging

Spring boot configures and creates a lot of beans. To debug/see what bean are created we can investigate application context that gets returned by run() method. We could do that in standalone application in main() or by overriding run() of SpringBootServletInitializer application.

```
@SpringBootApplication
public class App extends SpringBootServletInitializer {
  @Override
  protected SpringApplicationBuilder configure(SpringApplicationBuilder builder) {
     return builder.sources(App.class);
  }
  public static void main(String[] args) {
      ConfigurableApplicationContext context = SpringApplication.run(
              App.class, args);
      for (String beanName : context.getBeanDefinitionNames()) {
        System.out.println(beanName + " = " + context.getBean(beanName).getClass().getName());
      }
  }

  @Override
  protected WebApplicationContext run(SpringApplication application) {
      WebApplicationContext context = super.run(application);
      for (String beanName : context.getBeanDefinitionNames()) {
        System.out.println(beanName + " = " + context.getBean(beanName).getClass().getName());
      }
      return applicationContext;
  }
}
```

In-memory DB support

By just adding drivers of in memory databases like H2, HSQL and Derby databases, spring boot will setup datasource.
"runtime" scope will create runtime/test application datasource and "test" will create unit test datasource

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <scope>runtime</scope>
</dependency>
```

NOTE: In my experience H2 is most full featured database. Like it supports "drop table if exists"

Initializing Database

https://docs.spring.io/spring-boot/docs/current/reference/html/howto-database-initialization.html#howto-initialize-a-database-using-spring-jdbc

Spring boot reads **schema.sql**, and **data.sql** in classpath(resources) to initialize database on startup.
schema.sql: is executed before loading entities
data.sql: is executed after loading entities

We can also add platform name by adding property **spring.datasource.platform={platform name}** and then add **schema-{platform name}.sql** and **data-{platform name}.sql**.

e.g.
spring.datasource.platform=h2
main/resources/schema-h2.sql
main/resources/data-h2.sql

MySQL support

To configure MySQL datasource we can set these properties in application.properties

```
# I have to "serverTimezone=UTC" because my system is in EDT and MySQL server is in UTC.
# MySQL drivers were giving error because of that.
spring.datasource.url=jdbc:mysql://localhost:8889/testdb?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root
# Setting driver class name because I get this warning below:
# Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'. The driver is automatically registered via the SPI and manual loading of the driver class is generally unnecessary.
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# Spring boot automatically detects DB platform but still
# optionally we can define DB platform.
spring.datasource.platform=mysql
```

JPA & Hibernate

To add support for JPA and Hibernate we add "spring-boot-starter-data-jpa" dependency. Spring boot will

- find any database driver in application's dependency list
- Initialize datasource
- Setup spring transaction
- Setup JPA
- Setup Hibernate as JPA vendor

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

Configure JPA & Hibernate

All default JPA and Hibernate configurations can be overwritten in application.properties. E.g.

```
#############
# By doing
# spring.jpa.generate-ddl=true
# spring.jpa.hibernate.ddl-auto=create
#
```

```
# Hibernate will generate/run DDL and and print it in logs.
# We can use it these DDL, DML in flyaway migrate script
# initialize DB on startup.
# After that change ddl-auto=create to ddl-auto=update
#
##############
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=none

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

Running DB Scripts using Flyway on initialization

https://docs.spring.io/spring-boot/docs/current/reference/html/howto-database-initialization.html
We can utilities like Flyway and Liquibase to run DB initializing scripts.

Flyway Dependency

```
<dependency>
  <groupId>org.flywaydb</groupId>
  <artifactId>flyway-core</artifactId>
  <version>4.1.2</version>
</dependency>
```

Flyway Scripts location

By default Flyway looks for DB scripts in:

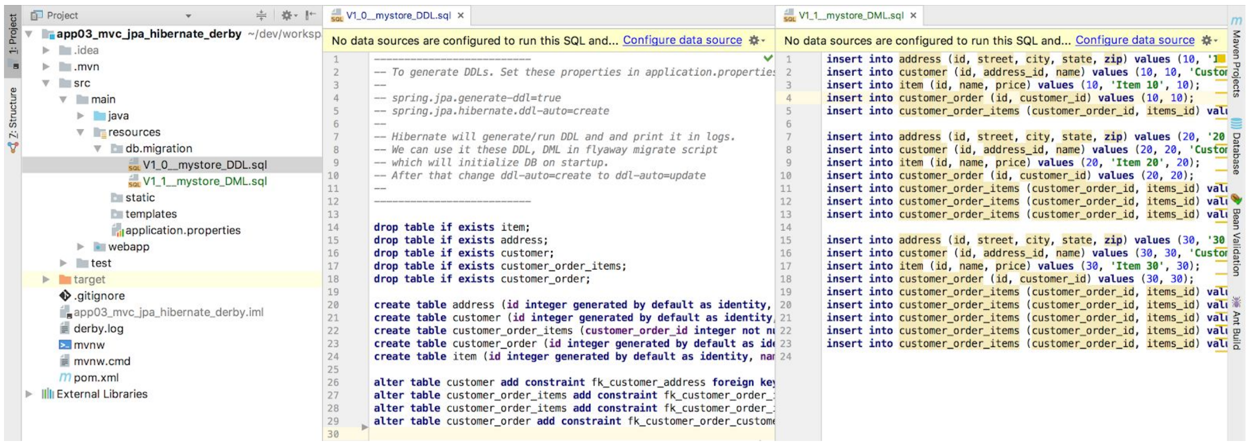
src/main/resources/db/migration/
Or
src/main/resources/db/migration/{vendor name}

To override default Flyway scripts location we can use this property in application.properties

```
flyway.locations=db/migration/mysql
```

Flyway Scripts File name

By default Flyway runs files named like below
V{major version}_{minor version}_{patch version}___{description}.sql
Flyway will order files by major and minor and run them all before spring boot application starts up.



```
app03_mvc_jpa_hibernate_derby ~/dev/worksp
  .idea
  .mvn
  src
    main
      java
      resources
        db.migration
          V1_0__mystore_DDL.sql
          V1_1__mystore_DML.sql
        static
        templates
        application.properties
      webapp
    test
  target
  .gitignore
  app03_mvc_jpa_hibernate_derby.iml
  derby.log
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
```

V1_0__mystore_DDL.sql

No data sources are configured to run this SQL and... Configure data source

```sql
--------------------------
-- To generate DDLs. Set these properties in application.properties
--
-- spring.jpa.generate-ddl=true
-- spring.jpa.hibernate.ddl-auto=create
--
-- Hibernate will generate/run DDL and and print it in logs.
-- We can use it these DDL, DML in flyaway migrate script
-- which will initialize DB on startup.
-- After that change ddl-auto=create to ddl-auto=update
--
--------------------------


drop table if exists item;
drop table if exists address;
drop table if exists customer;
drop table if exists customer_order_items;
drop table if exists customer_order;

create table address (id integer generated by default as identity,
create table customer (id integer generated by default as identity,
create table customer_order_items (customer_order_id integer not nu
create table customer_order (id integer generated by default as ide
create table item (id integer generated by default as identity, nar

alter table customer add constraint fk_customer_address foreign key
alter table customer_order_items add constraint fk_customer_order_
alter table customer_order_items add constraint fk_customer_order_
alter table customer_order add constraint fk_customer_order_custome
```

V1_1__mystore_DML.sql

No data sources are configured to run this SQL and... Configure data source

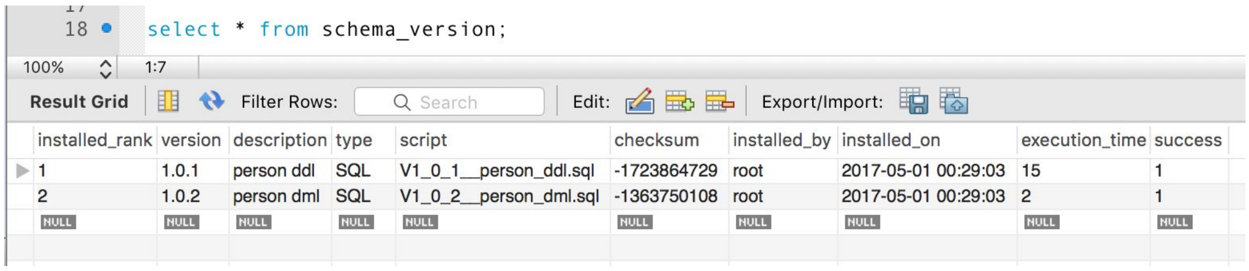
```sql
insert into address (id, street, city, state, zip) values (10, '1
insert into customer (id, address_id, name) values (10, 10, 'Custor
insert into item (id, name, price) values (10, 'Item 10', 10);
insert into customer_order (id, customer_id) values (10, 10);
insert into customer_order_items (customer_order_id, items_id) valu

insert into address (id, street, city, state, zip) values (20, '20
insert into customer (id, address_id, name) values (20, 20, 'Custor
insert into item (id, name, price) values (20, 'Item 20', 20);
insert into customer_order (id, customer_id) values (20, 20);
insert into customer_order_items (customer_order_id, items_id) valu
insert into customer_order_items (customer_order_id, items_id) valu
insert into customer_order_items (customer_order_id, items_id) valu

insert into address (id, street, city, state, zip) values (30, '30
insert into customer (id, address_id, name) values (30, 30, 'Custor
insert into item (id, name, price) values (30, 'Item 30', 30);
insert into customer_order (id, customer_id) values (30, 30);
insert into customer_order_items (customer_order_id, items_id) valu
insert into customer_order_items (customer_order_id, items_id) valu
insert into customer_order_items (customer_order_id, items_id) valu
insert into customer_order_items (customer_order_id, items_id) valu
insert into customer_order_items (customer_order_id, items_id) valu
```

Flyway migration tracking in SCHEMA_VERSION

For non in-memory databases we can get into migration scripts versioning issues. Flyway keeps track of all the scripts ran in table "schema_version"



```sql
select * from schema_version;
```

100%   1:7

Result Grid   Filter Rows: Search   Edit: Export/Import:

installed_rank	version	description	type	script	checksum	installed_by	installed_on	execution_time	success
1	1.0.1	person ddl	SQL	V1_0_1__person_ddl.sql	-1723864729	root	2017-05-01 00:29:03	15	1
2	1.0.2	person dml	SQL	V1_0_2__person_dml.sql	-1363750108	root	2017-05-01 00:29:03	2	1
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Handling SCHEMA_VERSION issues

To repair flyway schema_version issues we can use flyway command line utility or it maven plugin
https://flywaydb.org/documentation/maven/
$ mvn flyway:clean
$ mvn flyway:repair

Or in worst case start fresh by dropping and recreating database.

In-memory DB for test

https://www.leveluplunch.com/java/tutorials/022-preload-database-execute-sql-spring-testing/

In-memory DB for test & MySQL for application

Profile for application.properties

https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-profiles.html
https://docs.spring.io/spring-boot/docs/current/reference/html/howto-properties-and-configuration.html
We can create multiple application.properties files for different profiles.

To do this we will have to attach profile name to application.properties file like:
application-{profile name}.properties

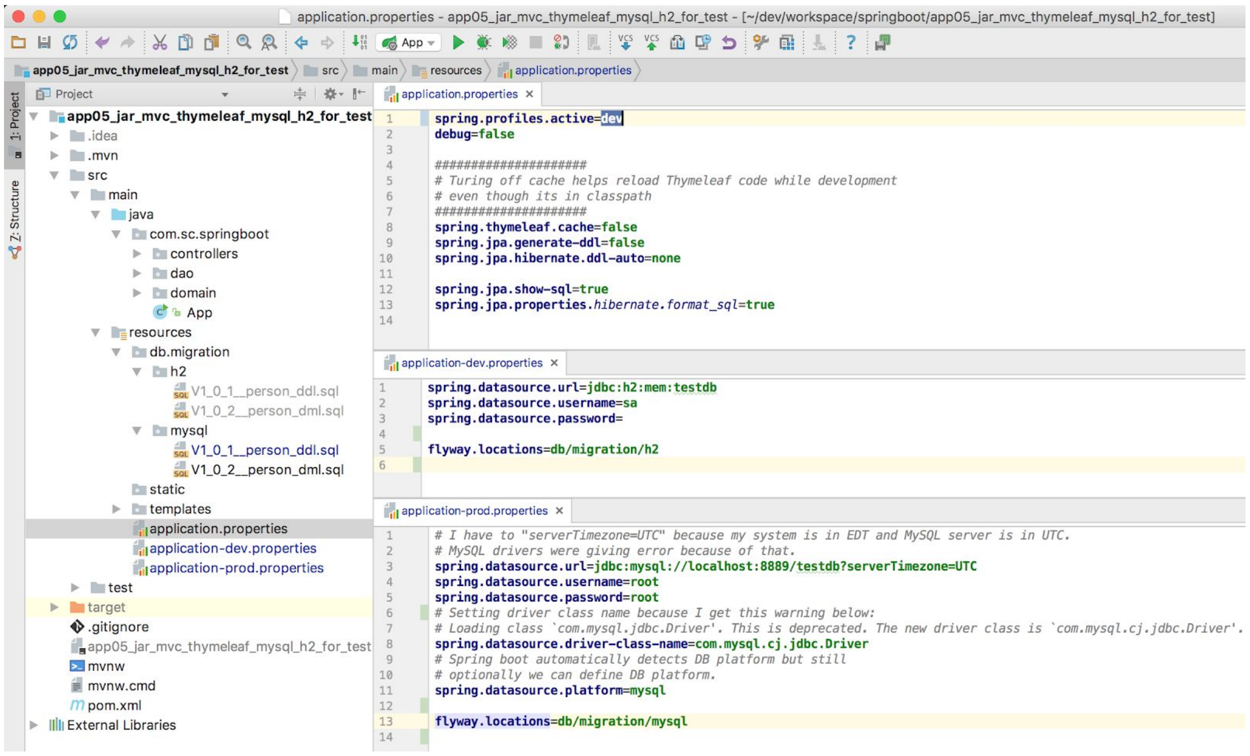
And then set the profile name in:
spring.profiles.active=**dev**

Spring boot will first read application.properties file then application-{profile name}.properties file

application.properties profile example

Let's say we need 2 data sources for our application but we are only going to use one of them at a time. An in-memory H2 datasource for development and MySQL datasource for production. Both have different set of Flyway DB initializing scripts.

In the example below application.properties contain all common configurations and **spring.profiles.active**=**dev**. application-dev.properties contains all configuration unique to "dev" profile and application-prod.properties contains all configuration unique to "prod" profile


application.properties - app05_jar_mvc_thymeleaf_mysql_h2_for_test - [~/dev/workspace/springboot/app05_jar_mvc_thymeleaf_mysql_h2_for_test]

```
app05_jar_mvc_thymeleaf_mysql_h2_for_test
  .idea
  .mvn
  src
    main
      java
        com.sc.springboot
          controllers
          dao
          domain
          App
      resources
        db.migration
          h2
            V1_0_1__person_ddl.sql
            V1_0_2__person_dml.sql
          mysql
            V1_0_1__person_ddl.sql
            V1_0_2__person_dml.sql
        static
        templates
        application.properties
        application-dev.properties
        application-prod.properties
    test
  target
  .gitignore
  app05_jar_mvc_thymeleaf_mysql_h2_for_test
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
```

application.properties

```
spring.profiles.active=dev
debug=false

####################
# Turing off cache helps reload Thymeleaf code while development
# even though its in classpath
####################
spring.thymeleaf.cache=false
spring.jpa.generate-ddl=false
spring.jpa.hibernate.ddl-auto=none

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

application-dev.properties

```
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.username=sa
spring.datasource.password=

flyway.locations=db/migration/h2
```

application-prod.properties

```
# I have to "serverTimezone=UTC" because my system is in EDT and MySQL server is in UTC.
# MySQL drivers were giving error because of that.
spring.datasource.url=jdbc:mysql://localhost:8889/testdb?serverTimezone=UTC
spring.datasource.username=root
spring.datasource.password=root
# Setting driver class name because I get this warning below:
# Loading class `com.mysql.jdbc.Driver'. This is deprecated. The new driver class is `com.mysql.cj.jdbc.Driver'.
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
# Spring boot automatically detects DB platform but still
# optionally we can define DB platform.
spring.datasource.platform=mysql

flyway.locations=db/migration/mysql
```

In the above example we hard coded active profile in application.properties but we can still run an alternative profile using the command below:

```
$ java -jar app05_jar_mvc_thymeleaf_mysql_h2_for_test-0.0.1-SNAPSHOT.jar
--spring.profiles.active=prod
```