What is Microservice?

- Aime -

Today's Contents

- What is Microservice
- Benefits and Drawbacks
- Spring vs Microservice
- Getting Started with Spring framework and Microservice
- Example - Order Microservice System by Spring Framework

What is Microservice

> a particular way of designing software applications as suites of independently deployable services.

- http://martinfowler.com

> **microservices** is a software architecture style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs.

\- Wikipedia

“

Micro-services are not 1 million services that make up a single service.

Micro-services are also not many large services each covering a bunch of functionality and interacting via databases or an ESB in a large enterprise system

micro-services are not necessarily independently deployable

Instead of focusing on the services being independently deployable, it is sufficient to choose groups of services that can be deployed independently.

”

- David, http://davidmorgantini.blogspot.kr/

What is Microservice ?

Characteristic of "MicroService"

small scope

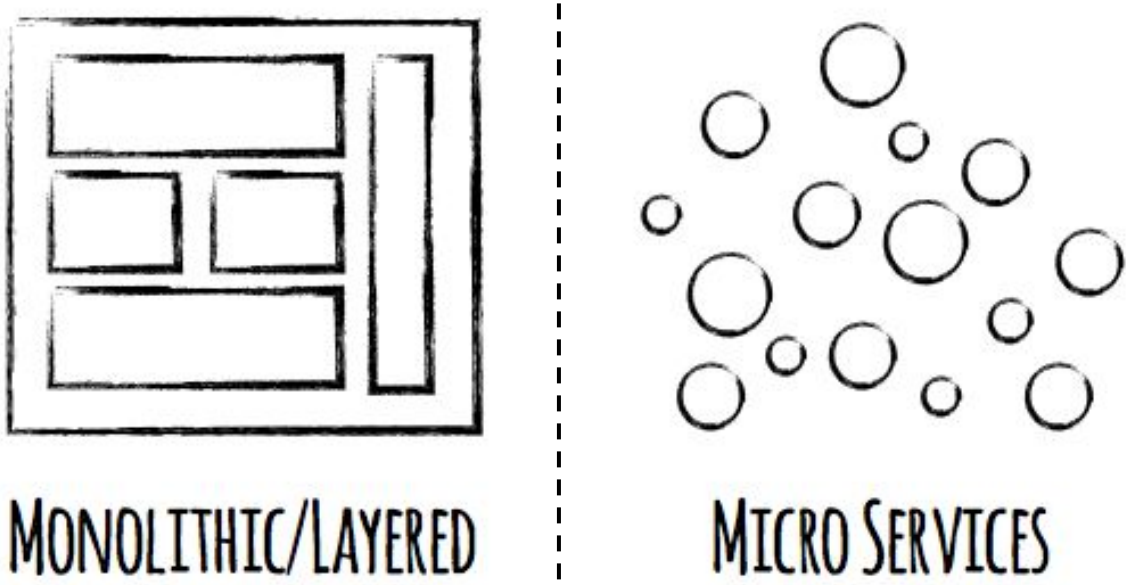
standalone

can integrate with other service via interface

(should) have it own database

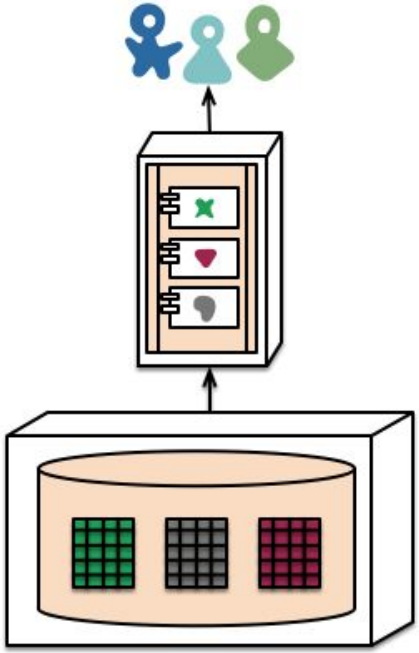
Difference between

Monolithic & Microservice

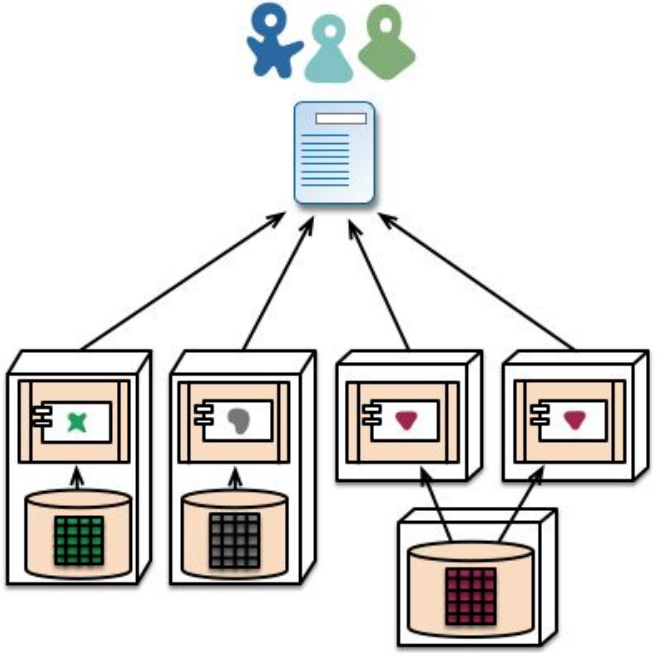


Difference between “Monolithic” and “microservice” - Concept

picture from eugenedvorkin.com



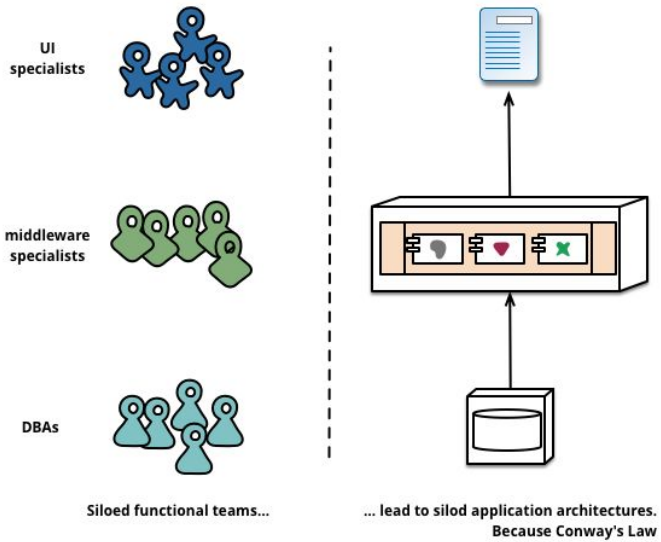
monolith - single database

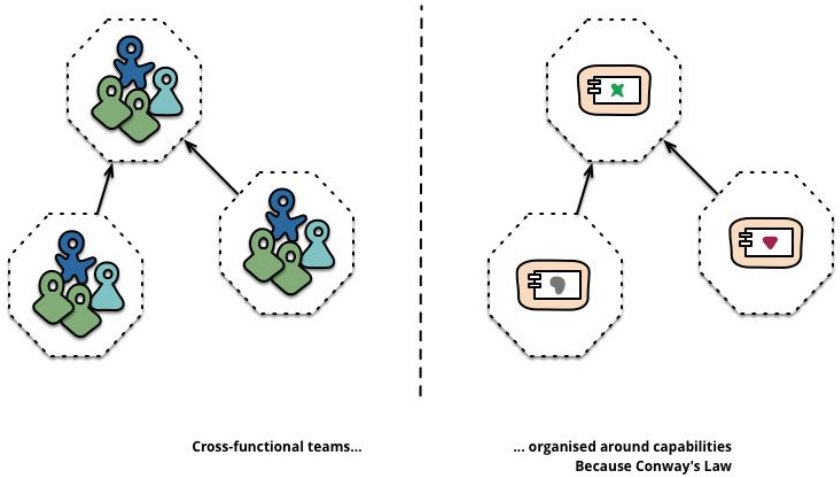


microservices - application databases

Difference between "Layered Architecture" and "microservice" - architecture

picture from martinfowler.com





Difference between "Layered Architecture" and "microservice" - development

picture from martinfowler.com

Benefits of Microservice

Benefits of "MicroService"

- Independent
- Speed up deployment
- Easy to Understand
- Easier to scale
- More Productive
- Improve fault isolation

DRAWBACKS ...

Drawbacks ...

Developer has to deal with developing distributed system

Tool Support

More Difficult on Testing

Use Cases

Inter-Service Communication Mechanism

Memory Consumption

Communication between teams

Nowaday...

SPRING vs MICROSERVICE

spring

https://spring.io/tools

SPRING TOOL SUITE™

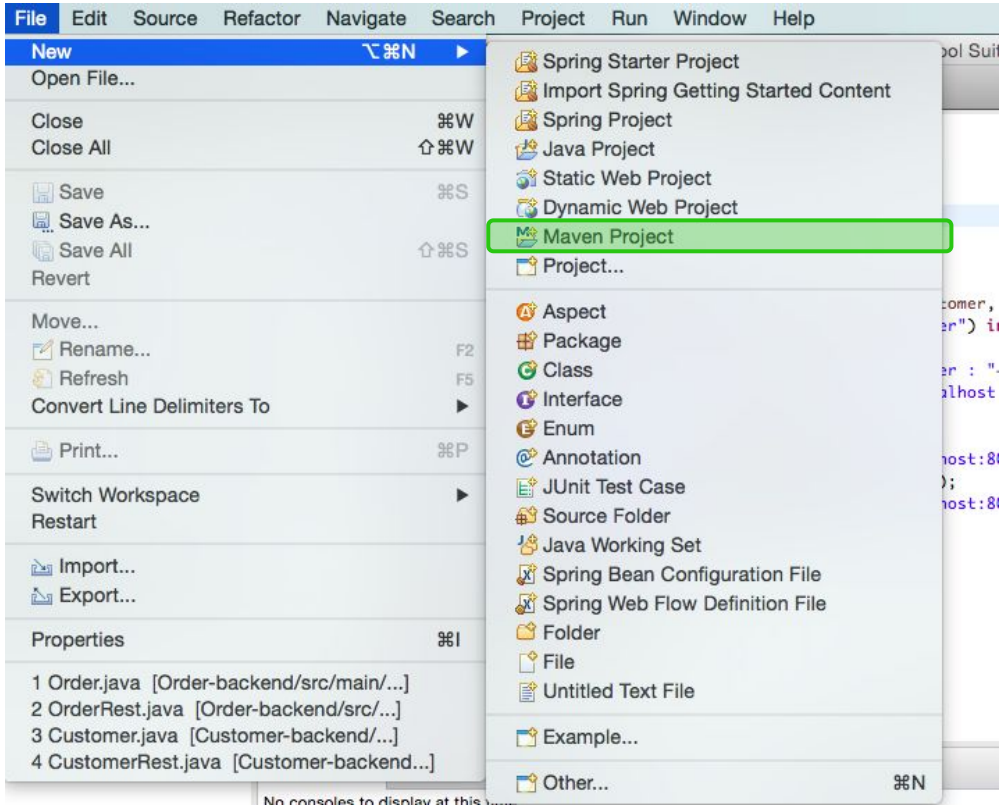
The Spring Tool Suite is an Eclipse-based development environment that is customized for developing Spring applications. It provides a ready-to-use environment to implement, debug, run, and deploy your Spring applications, including integrations for Pivotal tc Server, Pivotal Cloud Foundry, Git, Maven, AspectJ, and more.

LEARN MORE

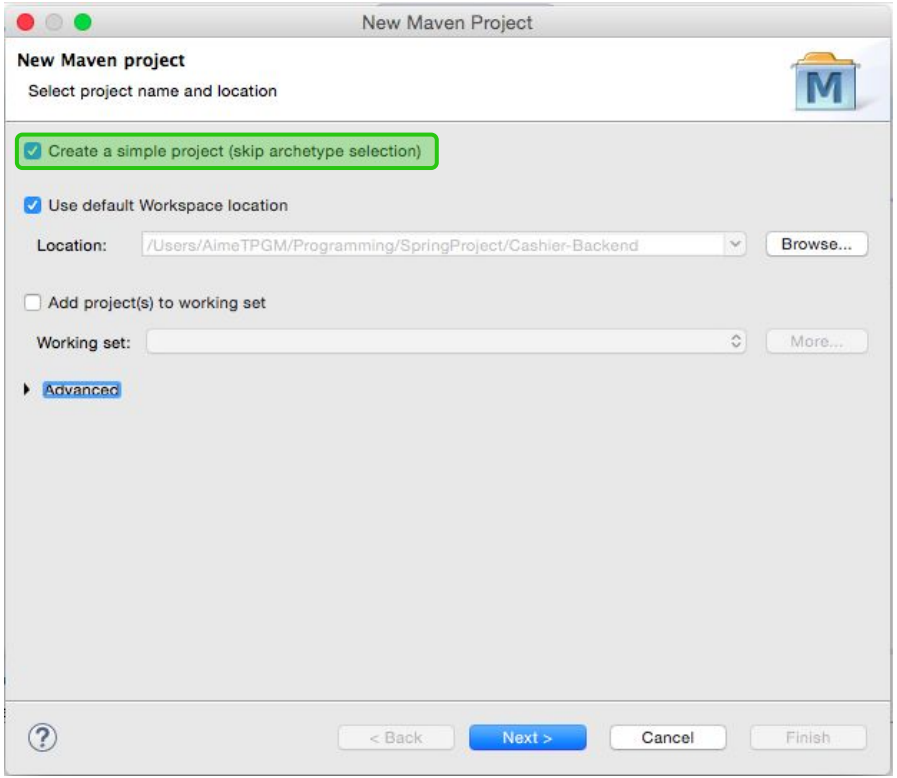
DOWNLOAD STS
(3.7.0.RELEASE for Mac)

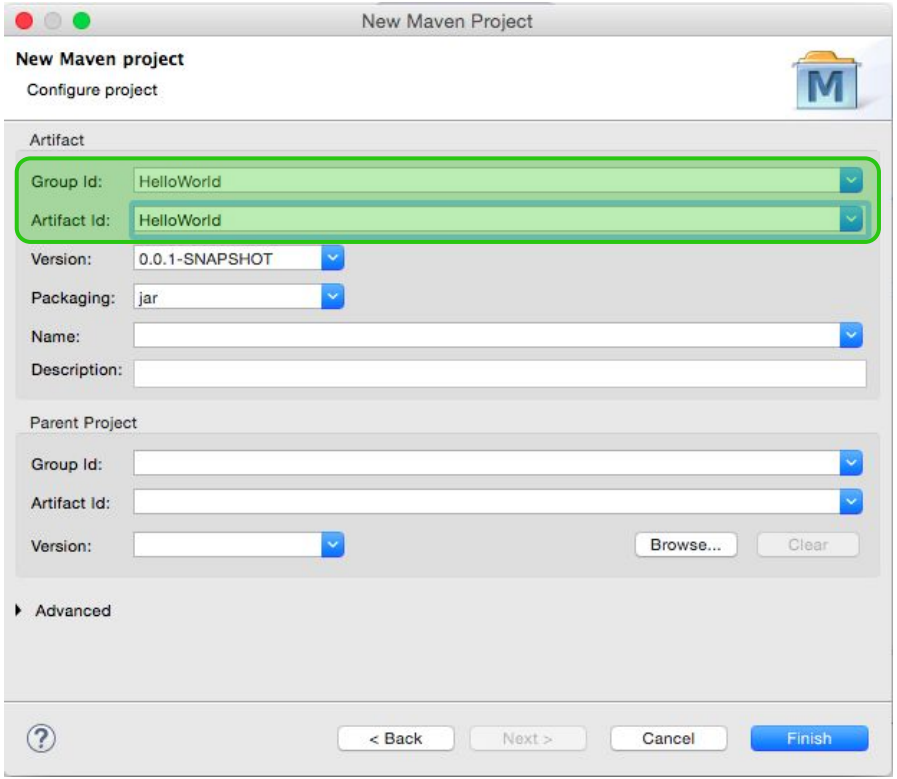
See all versions

Getting Started

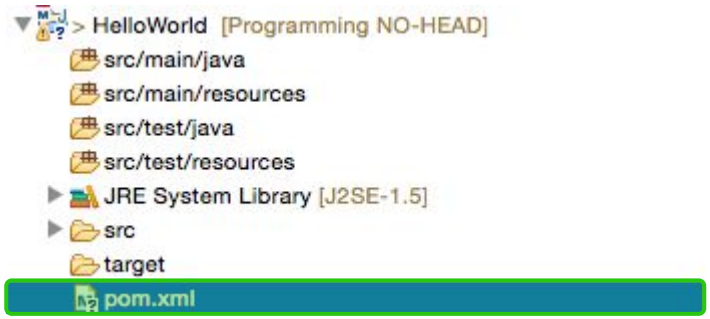


File > New > Maven Project





Check “Skip archetype selection” > Next > Enter Group id and Artifact id > Finish



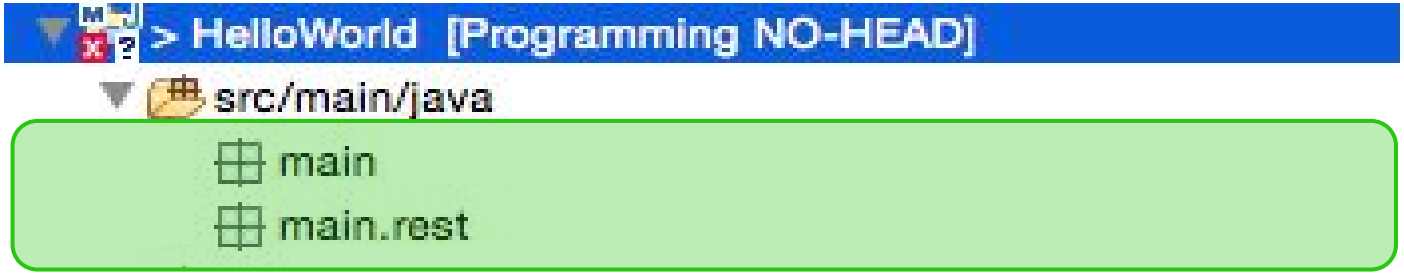
HelloWorld/pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi=
  <modelVersion>4.0.0</modelVersion>
  <groupId>HelloWorld</groupId>
  <artifactId>HelloWorld</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>1.2.5.RELEASE</version>
  </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-jersey</artifactId>
    </dependency>
  </dependencies>
</project>
```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml

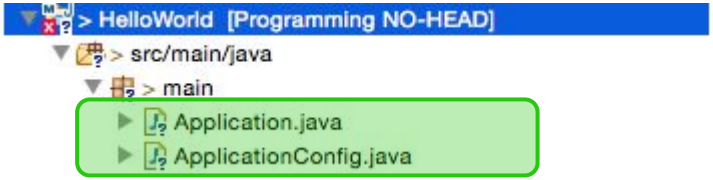
Open “pom.xml” > Add <parent> and each <dependency>

- to make a microservice, it needs to add spring-boot dependency



“main” is for storing main(), identifying Spring, Configuration
“main.rest” is for making RESTful WS, API, other components

Create “main” and “main.rest” packages under src/main/java




```java
package main;

import org.springframework.boot.SpringApplication;

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

}
```

```java
package main;

import javax.inject.Named;

@Configuration
public class ApplicationConfig {
    @Named
    static class JerseyConfig extends ResourceConfig{
        public JerseyConfig(){
            this.packages("main.rest");
        }
    }

}
```

"Application" contains main() and Defines Spring boot
"ApplicationConfig" contains Jersey configuration

Create "Application" and "ApplicationConfig" class under src/main/java/main

HelloWorld.java | HelloWorldRest.java

```
package main.rest;

public class HelloWorld {

    private String hello;

    public String getHello(){
        return hello;
    }

    public void setHello(String name){
        hello = "Hello"+name;
    }

}
```

HelloWorld.java | *HelloWorldRest.java

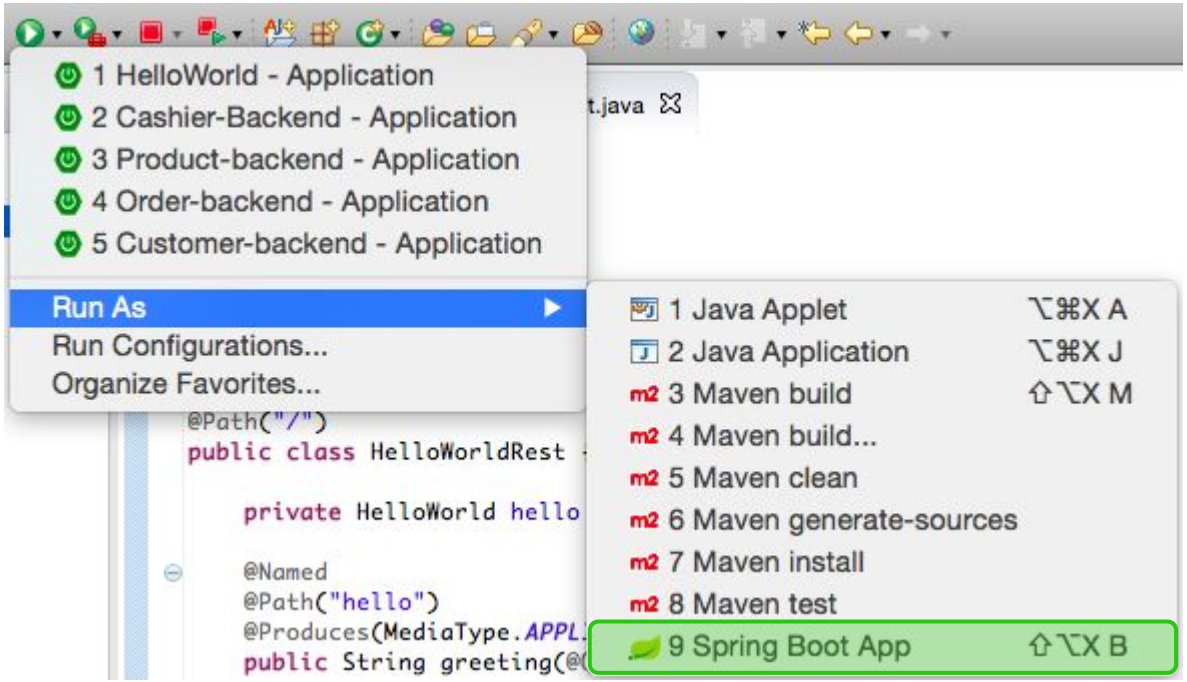
```
package main.rest;

import javax.inject.Named;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Named
@Path("/")
public class HelloWorldRest {

    private HelloWorld hello = new HelloWorld();

    @GET
    @Path("hello")
    @Produces(MediaType.APPLICATION_JSON)
    public String greeting(@QueryParam("name") String name ){
        hello.setHello(name);
        return hello.getHello();
    }

}
```

“Hello” contains application model

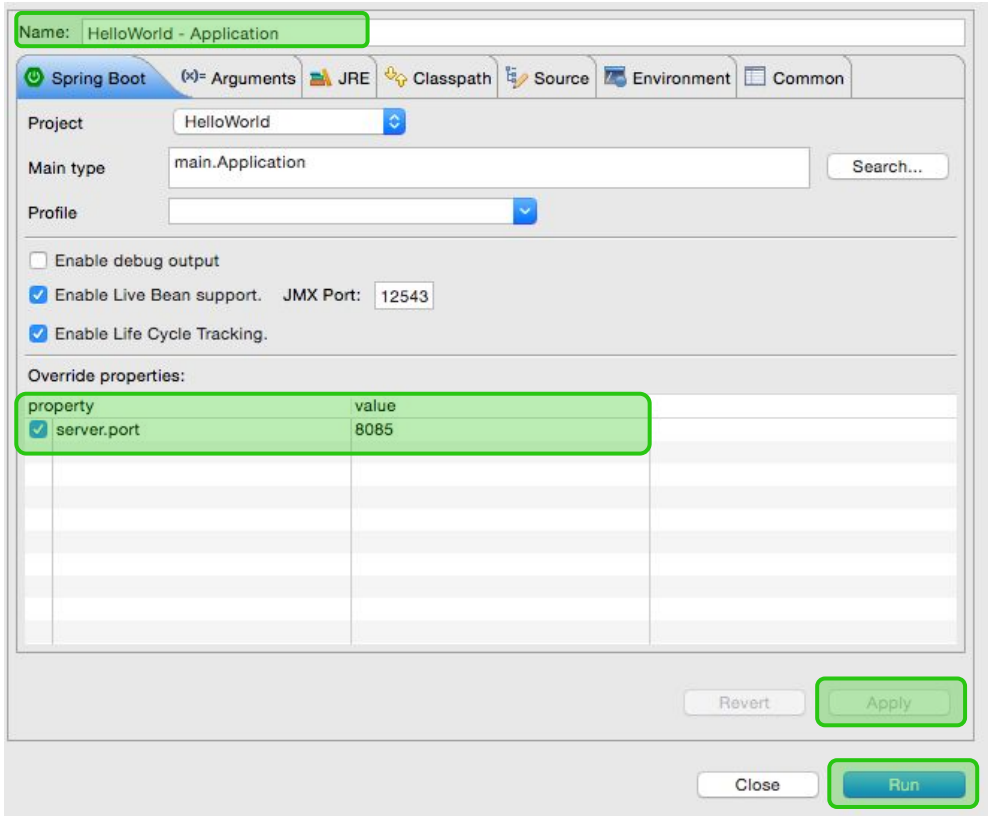
“HelloWorldRest” provides API

* you can move “Hello” to other package, i.e., main.model, as well if needed

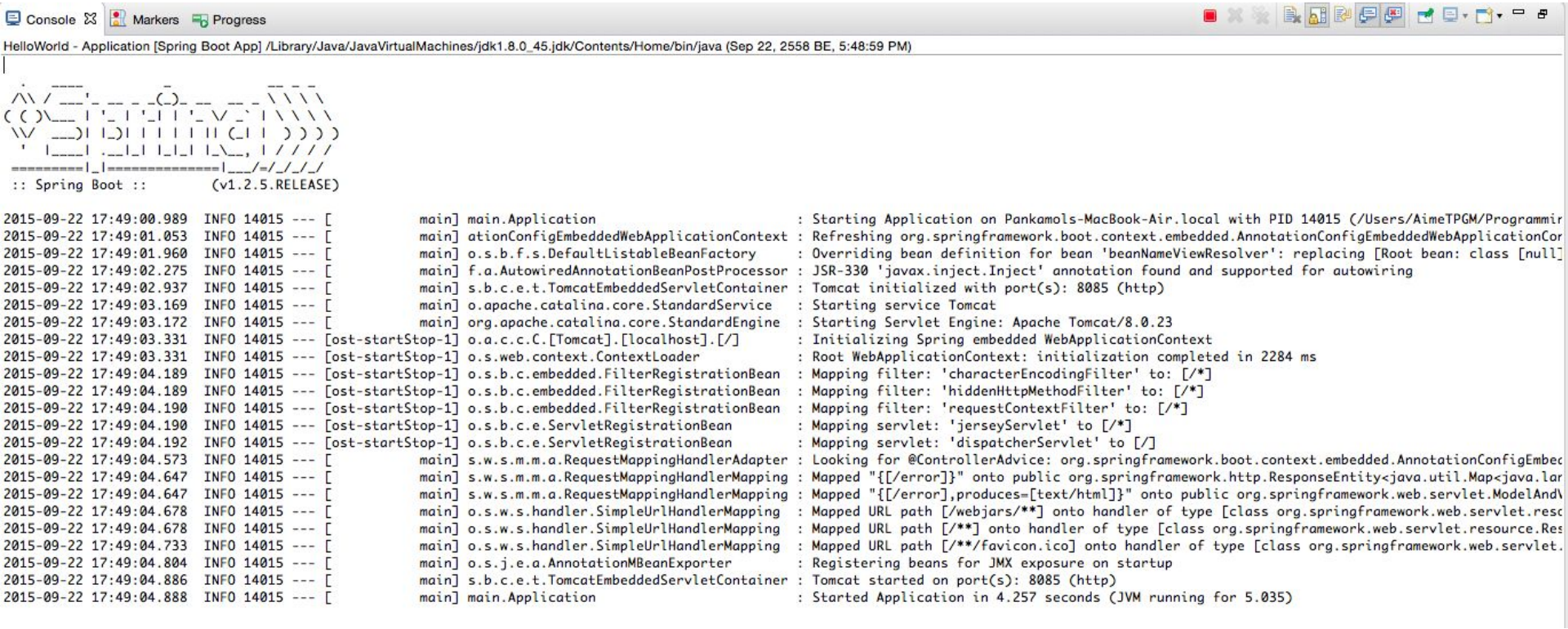
Create “HelloWorld” and “HelloWorldRest” class under src/main/java/main/rest



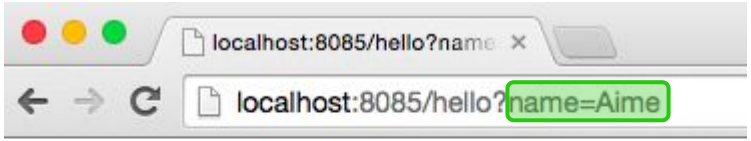
Run as Spring Boot App



Run Configuration ...


Console
Markers
Progress
HelloWorld - Application [Spring Boot App] /Library/Java/JavaVirtualMachines/jdk1.8.0_45.jdk/Contents/Home/bin/java (Sep 22, 2558 BE, 5:48:59 PM)
:: Spring Boot :: (v1.2.5.RELEASE)
2015-09-22 17:49:00.989 INFO 14015 --- [main] main.Application : Starting Application on Pankamols-MacBook-Air.local with PID 14015 (/Users/AimeTPGM/Programmir
2015-09-22 17:49:01.053 INFO 14015 --- [main] ationConfigEmbeddedWebApplicationContext : Refreshing org.springframework.boot.context.embedded.AnnotationConfigEmbeddedWebApplicationCor
2015-09-22 17:49:01.960 INFO 14015 --- [main] o.s.b.f.s.DefaultListableBeanFactory : Overriding bean definition for bean 'beanNameViewResolver': replacing [Root bean: class [null]
2015-09-22 17:49:02.275 INFO 14015 --- [main] f.a.AutowiredAnnotationBeanPostProcessor : JSR-330 'javax.inject.Inject' annotation found and supported for autowiring
2015-09-22 17:49:02.937 INFO 14015 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat initialized with port(s): 8085 (http)
2015-09-22 17:49:03.169 INFO 14015 --- [main] o.apache.catalina.core.StandardService : Starting service Tomcat
2015-09-22 17:49:03.172 INFO 14015 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.0.23
2015-09-22 17:49:03.331 INFO 14015 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2015-09-22 17:49:03.331 INFO 14015 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2284 ms
2015-09-22 17:49:04.189 INFO 14015 --- [ost-startStop-1] o.s.b.c.embedded.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/*]
2015-09-22 17:49:04.189 INFO 14015 --- [ost-startStop-1] o.s.b.c.embedded.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2015-09-22 17:49:04.190 INFO 14015 --- [ost-startStop-1] o.s.b.c.embedded.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [/*]
2015-09-22 17:49:04.190 INFO 14015 --- [ost-startStop-1] o.s.b.c.e.ServletRegistrationBean : Mapping servlet: 'jerseyServlet' to [/*]
2015-09-22 17:49:04.192 INFO 14015 --- [ost-startStop-1] o.s.b.c.e.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2015-09-22 17:49:04.573 INFO 14015 --- [main] s.w.s.m.m.a.RequestMappingHandlerAdapter : Looking for @ControllerAdvice: org.springframework.boot.context.embedded.AnnotationConfigEmbed
2015-09-22 17:49:04.647 INFO 14015 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error]}" onto public org.springframework.http.ResponseEntity<java.util.Map<java.lar
2015-09-22 17:49:04.647 INFO 14015 --- [main] s.w.s.m.m.a.RequestMappingHandlerMapping : Mapped "{[/error],produces=[text/html]}" onto public org.springframework.web.servlet.ModelAndV
2015-09-22 17:49:04.678 INFO 14015 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/webjars/**] onto handler of type [class org.springframework.web.servlet.resc
2015-09-22 17:49:04.678 INFO 14015 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**] onto handler of type [class org.springframework.web.servlet.resource.Res
2015-09-22 17:49:04.733 INFO 14015 --- [main] o.s.w.s.handler.SimpleUrlHandlerMapping : Mapped URL path [/**/favicon.ico] onto handler of type [class org.springframework.web.servlet.
2015-09-22 17:49:04.804 INFO 14015 --- [main] o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
2015-09-22 17:49:04.886 INFO 14015 --- [main] s.b.c.e.t.TomcatEmbeddedServletContainer : Tomcat started on port(s): 8085 (http)
2015-09-22 17:49:04.888 INFO 14015 --- [main] main.Application : Started Application in 4.257 seconds (JVM running for 5.035)

On Console



HelloWorld.java | *HelloWorldRest.java

```
package main.rest;

import javax.inject.Named;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;

@Named
@Path("/")
public class HelloWorldRest {

    private HelloWorld hello = new HelloWorld();

    @GET
    @Path("hello")
    @Produces(MediaType.APPLICATION_JSON)
    public String greeting(@QueryParam("name") String name ){
        hello.setHello(name);
        return hello.getHello();
    }

}
```

HelloWorld.java | HelloWorldRest.java

```
package main.rest;

public class HelloWorld {

    private String hello;

    public String getHello(){
        return hello;
    }

    public void setHello(String name){
        hello = "Hello"+name;
    }

}
```

GET Request

Microservice Example