HowToDoInJava

Bootstrapping a REST API with Spring Boot

Last Updated: January 28, 2022 By: Lokesh Gupta Spring Boot REST APIs

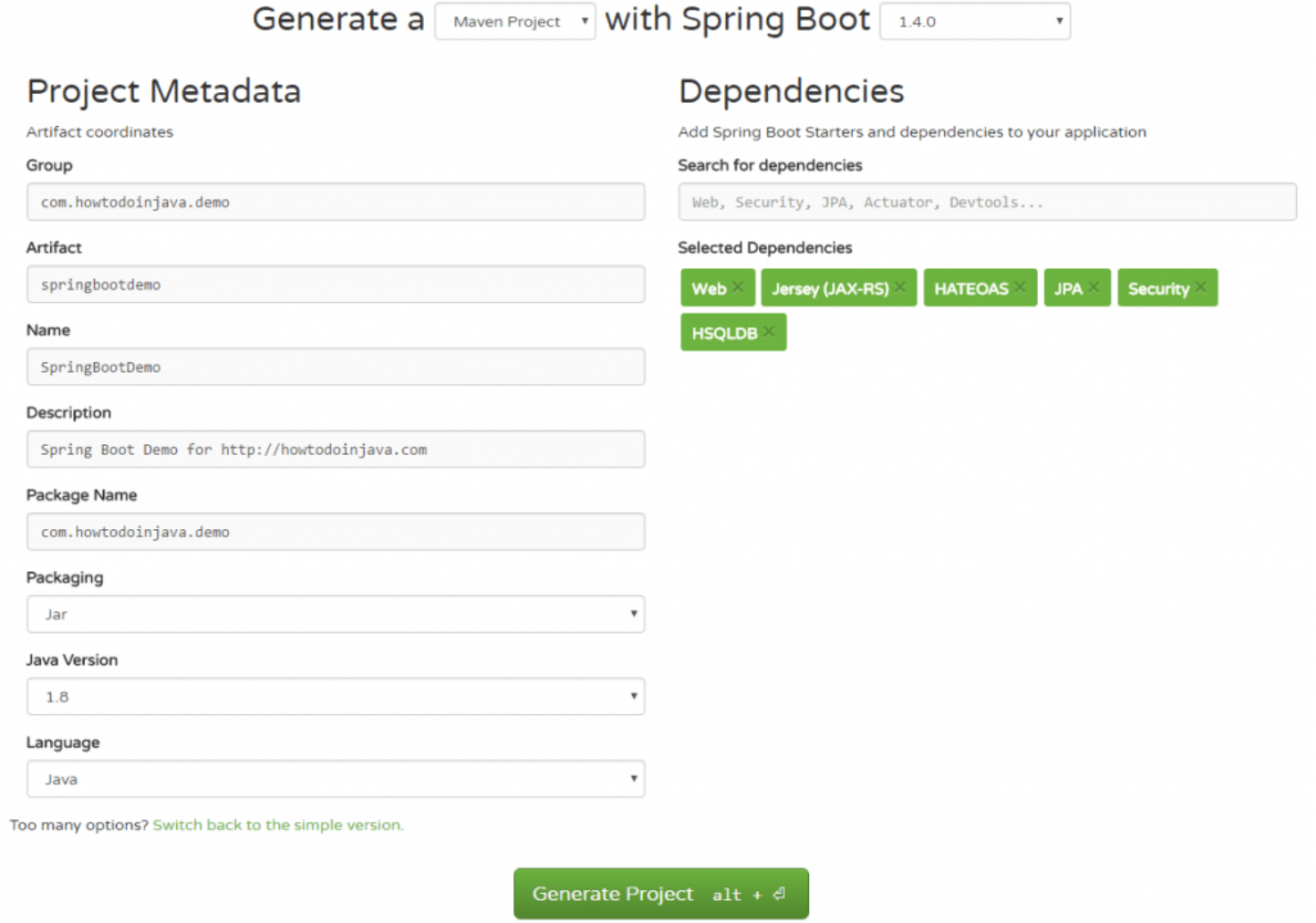
Spring boot is sub-project developed by developers of spring framework – to create stand-alone, production-grade application with minimum configuration possible. Spring boot applications are typically bundled as fat/uber jar files and can be deployed in any platform as a simple jar file. This is why spring boot applications are a good candidate for building microservices in java.

Let's learn it by starting with a **spring boot hello world example in eclipse** step by step.

Table of Contents

1. Create spring boot hello world project template

To create a template for spring boot application, I will suggest to use http://start.spring.io/. Here, you can select all dependencies which you have currently in mind, and generate the project.



Spring Boot Options

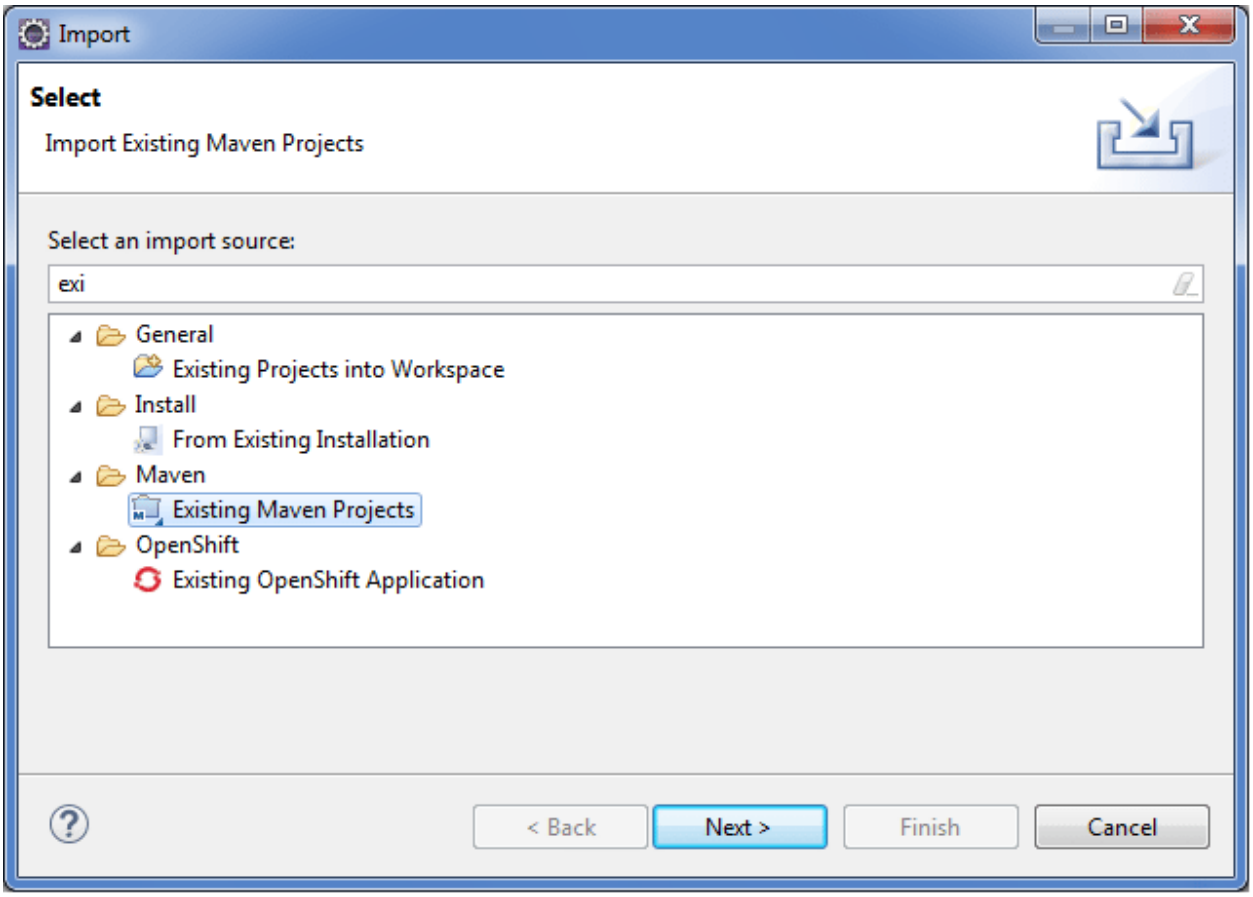
I have selected dependencies like Jersey, Spring Web, Spring HATEOAS, Spring JPA and Spring Security etc. You can add more dependencies after you have downloaded and imported the project or in future when requirements arise.

`Generate Project` button will generate a `.zip` file. Download and extract the file into your workspace.

2. Import spring boot project to eclipse

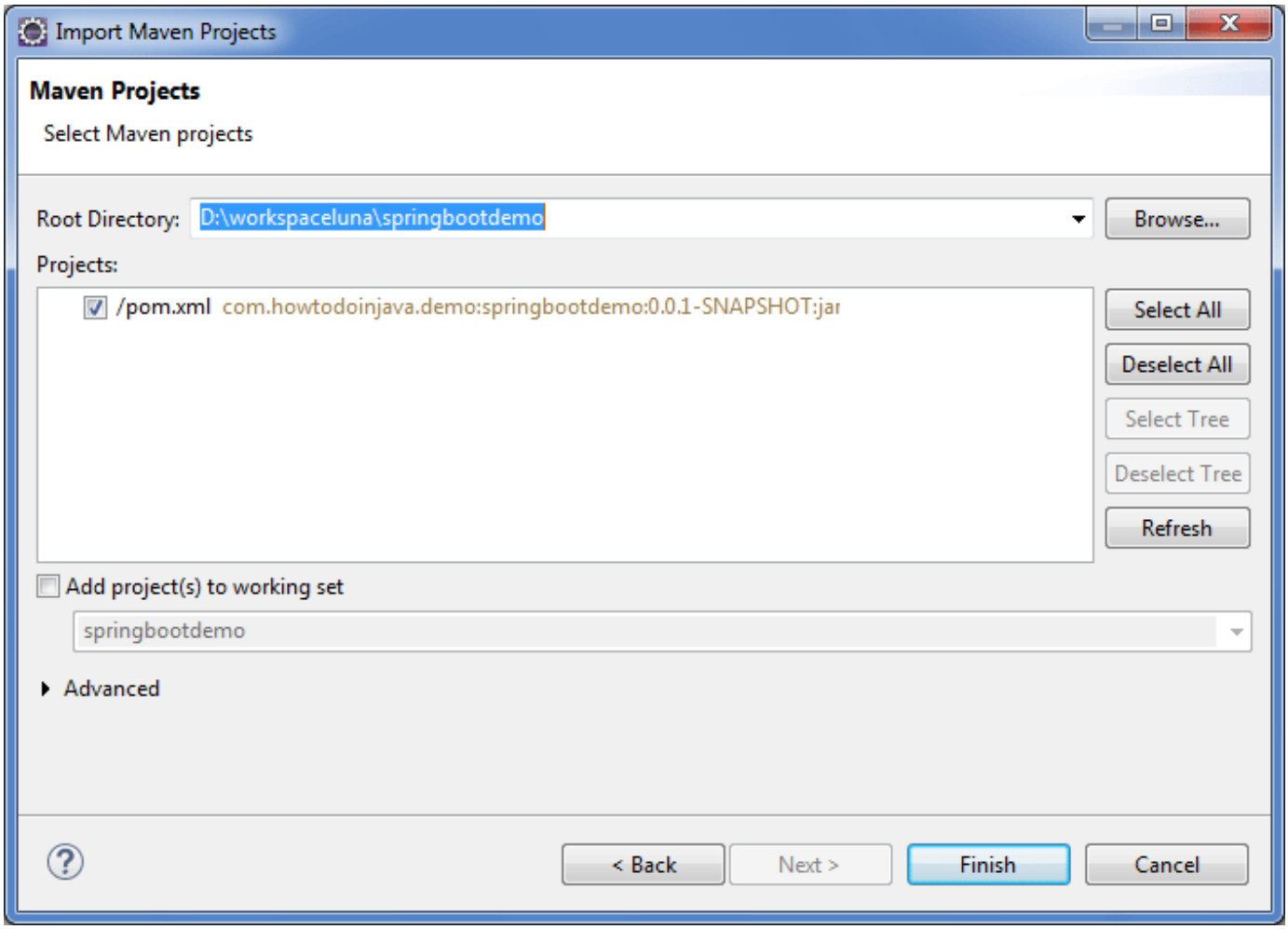
Next step is to import the generated project into your IDE. I have used eclipse for this purpose.

1) Import the spring boot project as existing maven project.



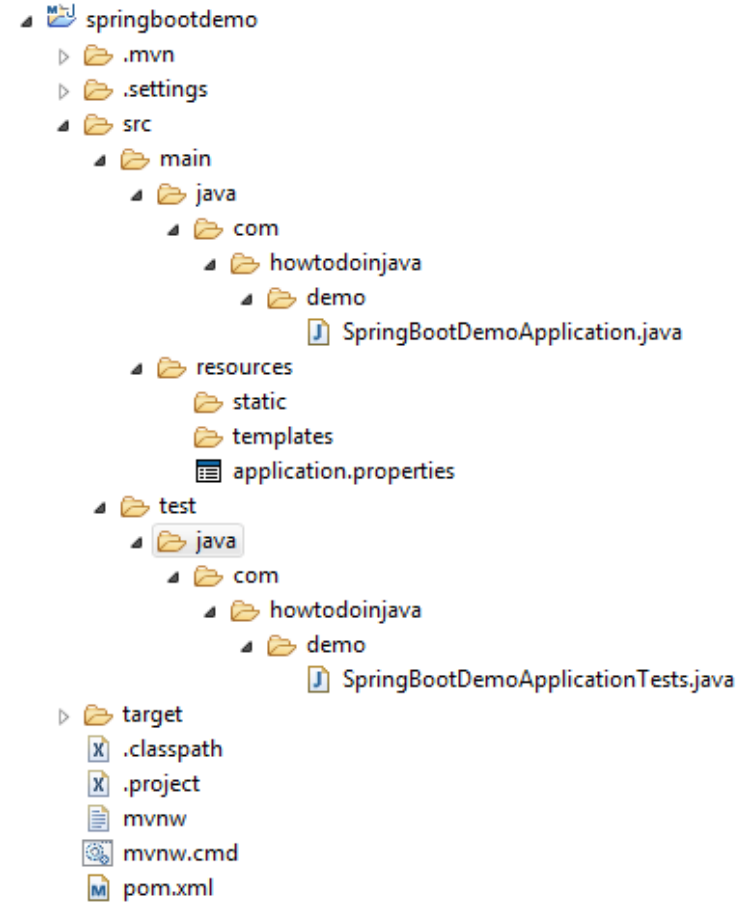
Import Existing Maven Project into Eclipse

2) Select the `pom.xml` file to import it.



Select pom.xml file to import maven project

3) Project will be imported and the dependencies you added while generating zip file, will be automatically downloaded and added into classpath.



Imported Spring Boot Project Structure

You have now successfully imported spring boot application. Now let's see what it has already configured for you.

3. Spring boot auto configuration

With spring boot, good thing is when you add a dependency (e.g. *Spring security*), it make fair assumptions and automatically configure some defaults for you. So you can start immediately.

Spring Boot uses convention over configuration by scanning the dependent libraries available in the class path. For each `spring-boot-starter-*` dependency in the POM file, Spring Boot executes a default `AutoConfiguration` class. `AutoConfiguration` classes use the `*AutoConfiguration` lexical pattern, where * represents the library. For example, the autoconfiguration of spring security is done through `SecurityAutoConfiguration`.

At the same time, if you don't want to use auto configuration for any project, it makes it very simple. Just use `exclude = SecurityAutoConfiguration.class` like below.

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

It is also possible to override default configuration values using the `application.properties` file in `src/main/resources` folder.

4. Spring boot annotations

Now look at `@SpringBootApplication` annotation what it actually does.

4.1. @SpringBootApplication annotation

`SpringBootApplication` is defined as below:

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Inherited
@SpringBootConfiguration
@EnableAutoConfiguration
@ComponentScan(excludeFilters = @Filter(type = FilterType.CUSTOM, classes = TypeEx
public @interface SpringBootApplication
{
   //more code
}
```

It adds 3 important annotations for application configuration purpose.

1. **@SpringBootConfiguration**

```
@Configuration
public @interface SpringBootConfiguration
{
   //more code
}
```

This annotation adds `@Configuration` annotation to class which **mark the class a source of bean definitions for the application context.**

2. **@EnableAutoConfiguration**

   This tells spring boot to auto configure important bean definitions based on added dependencies in `pom.xml` by start adding beans based on classpath settings, other beans, and various property settings.

3. **@ComponentScan**

   This annotation tells spring boot to scan base package, find other beans/components and configure them as well.

5. How to verify auto-configured beans by spring boot

If you ever want to know what all beans have been automatically configured into your **spring boot hello world application**, then use this code and run it.

```
SpringBootDemoApplication.java

import java.util.Arrays;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.security.SecurityAutoConfiguration;
import org.springframework.context.ApplicationContext;

@SpringBootApplication (exclude = SecurityAutoConfiguration.class)
public class SpringBootDemoApplication {

   public static void main(String[] args)
   {
      ApplicationContext ctx = SpringApplication.run(SpringBootDemoApplication.cla
```

```
        String[] beanNames = ctx.getBeanDefinitionNames();

        Arrays.sort(beanNames);

        for (String beanName : beanNames)
        {
            System.out.println(beanName);
        }
    }
}
```

With my **pom.xml** file, it generates following beans names along with plenty of other **springframework.boot.autoconfigure** dependencies.

```
Console

simpleControllerHandlerAdapter
sortResolver
spring.datasource-org.springframework.boot.autoconfigure.jdbc.DataSourceProperties
spring.hateoas-org.springframework.boot.autoconfigure.hateoas.HateoasProperties
spring.http.encoding-org.springframework.boot.autoconfigure.web.HttpEncodingProper
spring.http.multipart-org.springframework.boot.autoconfigure.web.MultipartProperti
spring.info-org.springframework.boot.autoconfigure.info.ProjectInfoProperties
spring.jackson-org.springframework.boot.autoconfigure.jackson.JacksonProperties
spring.jpa-org.springframework.boot.autoconfigure.orm.jpa.JpaProperties
spring.jta-org.springframework.boot.autoconfigure.transaction.jta.JtaProperties
spring.mvc-org.springframework.boot.autoconfigure.web.WebMvcProperties
spring.resources-org.springframework.boot.autoconfigure.web.ResourceProperties
springBootDemoApplication
standardJacksonObjectMapperBuilderCustomizer
stringHttpMessageConverter
tomcatEmbeddedServletContainerFactory
tomcatPoolDataSourceMetadataProvider
transactionAttributeSource
transactionInterceptor
transactionManager
transactionTemplate
viewControllerHandlerMapping
viewResolver
websocketContainerCustomizer
```

6. Spring boot REST API example

Now it's time to build any functionality into hello world application. You can add functionality as per your need, I am adding a REST API.

6.1. Create REST Controller

Create a package `com.howtodoinjava.demo.controller` and create rest controller inside it.

```
EmployeeController.java

import java.util.ArrayList;
import java.util.List;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import com.howtodoinjava.demo.model.Employee;

@RestController
public class EmployeeController
{
   @RequestMapping("/")
    public List<Employee> getEmployees()
    {
      List<Employee> employeesList = new ArrayList<Employee>();
      employeesList.add(new Employee(1,"lokesh","gupta","howtodoinjava@gmail.com")
      return employeesList;
    }
}
```

6.2. Create Model

Create model class `Employee`.

```
Employee.java

public class Employee {

   public Employee() {

   }
   public Employee(Integer id, String firstName, String lastName, String email) {
      super();
      this.id = id;
      this.firstName = firstName;
```

```
        this.lastName = lastName;
        this.email = email;
    }

    private Integer id;
    private String firstName;
    private String lastName;
    private String email;

    //getters and setters

    @Override
    public String toString() {
        return "Employee [id=" + id + ", firstName=" + firstName
                + ", lastName=" + lastName + ", email=" + email + "]";
    }
}
```

7. Spring boot hello world example demo

Now start the application by running `main()` method in `SpringBootDemoApplication`. It will start the embedded tomcat server on port `8080`.

As we have configured the demo REST API URL to root URL, you can access it on `http;//localhost:8080/` itself.

Verify Spring Boot REST API

You will get the below response in testing tool or browser.

```
[{"id":1,"firstName":"lokesh","lastName":"gupta","email":"howtodoinjava@gmail.com"}]
```

That's all for this **spring boot rest hello world example** with simple **rest api** example.

Drop me your questions related to how to create spring boot project in eclipse using maven.

Happy Learning !!

Sourcecode Download

Resources:

Spring Boot Project
http://start.spring.io/

@SpringBootApplication Annotation

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. REST API Request Validation with Spring Boot
2. Securing Spring Boot REST API with Basic Auth
3. REST API Security Guide
4. Jersey REST API Security Example
5. Jersey – How to set Cookie in REST API Response
6. Generate REST API Docs with Swagger
7. Java REST API Tutorials
8. Create Jersey REST APIs with Spring Boot
9. Adding Role Based Security with Spring Boot REST APIs
0. Bootstrapping ValidationFactory with Hibernate Validator CDI

27 thoughts on “Bootstrapping a REST API with Spring Boot”

Mark

April 6, 2020 at 4:04 am

This example provides novices with steps to follow at ease. However, these two things may need to mention to get desired result:
1) In initial creating template phase, picking Spring Web only as dependency would suffice;
2) In Employee class, one needs to add getters and setters for all private attributes defined there (all such methods were put in the associated file from downloadable)
Thanks

Reply

Ajit Shinde

January 17, 2020 at 6:52 pm

i use below line of code to avoid the run time exception, its working fine

```
@SpringBootApplication(exclude=
{DataSourceAutoConfiguration.class,
SecurityAutoConfiguration.class})
```

But am getting below response

```
{
  "timestamp": "2020-01-17T13:06:12.158+0000",
  "status": 404,
  "error": "Not Found",
  "message": "No message available",
  "path": "/"
}
```

Reply

Jackie Zheng

October 10, 2019 at 2:09 am

Hi Lokesh,

I have removed the Security dependency from pom.xml and it says the webpage cannot be found on http://localhost:8080/
Would you know what I have done wrong?

Reply

Jackie Zheng

October 9, 2019 at 9:56 pm

Hi Lokesh,

I hope you are well. I am a very beginner.

I am using STS 4 doing this exercise.

I have experienced two problems when doing this exercise:

1. at this step from your work: 5. How to verify auto-configured beans by spring boot.

I use your code and there are two error message shows:

a: The import org.springframework.boot.autoconfigure.security.SecurityAutoConfiguration cannot be resolved (this is from one of the import)

b: org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration (this is from exclude = SecurityAutoConfiguration.class)

So I didn't run this code. Would you know what have I done wrong?

2. I have followed your work till 7. Spring boot hello world example demo. and I run the springbootdemo on STS Boot Dashboard and right click it and click open web browser. It took me to the http://localhost:8080/login. Then request me to sign in with name and password. I used the name from console which is default and copied the generated security password that also from console but still can't sign in. Is this normal? Thus for now I didn't get the result as what you get above.

Would you please help. I would be much appreciated.

Reply

Jackie Zheng

October 9, 2019 at 10:46 pm

Hi Lokesh,

I guess it is to do with the Security dependencies that's why I need to sign in. I don't know what is the correct sign in detail as I used the name and password created from console and It won't work. So I have tried to delete the Security dependencies and run the application again. It took me to http://localhost:8080/ this time without sign in page, but shows The webpage cannot be found.

Any advise would be help. Thanks.

Reply

Danish

February 14, 2019 at 12:41 pm

After removing "spring-boot-starter-security" dependency from pom.xml also. It is asking for username and password.

Reply

Lokesh Gupta

February 14, 2019 at 1:16 pm

try running `mvn clean install`. Check for dependencies/jars in deployment assembly.

Reply

Susampath

October 7, 2019 at 11:02 am

place following code in the application.properties

security.basic.enabled=false

This will disable the security default security oprions

Reply

andrew

December 8, 2019 at 1:00 pm

It is working thanks

Reply

anupama

February 5, 2019 at 10:44 am

Do I need to update application.properties? I am using Spring boot 2 and on startup get the exception:

ConfigServletWebServerApplicationContext : Exception encountered during context initialization – cancelling refresh attempt: org.springframework.beans.factory.UnsatisfiedDependencyException: Error creating bean with name

'org.springframework.boot.autoconfigure.orm.jpa.HibernateJpaConfiguration': Unsatisfied dependency expressed through constructor parameter 0; nested exception is org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'dataSource' defined in class path resource [org/springframework/boot/autoconfigure/jdbc/DataSourceConfiguration$Hikari.class]: Bean instantiation via factory method failed; nested exception is org.springframework.beans.BeanInstantiationException: Failed to instantiate [com.zaxxer.hikari.HikariDataSource]: Factory method 'dataSource' threw exception; nested exception is org.springframework.boot.autoconfigure.jdbc.DataSourceProperties$DataSourceBeanCreationException: Failed to determine a suitable driver class

Could you please guide?

Reply

Lokesh Gupta

February 5, 2019 at 12:04 pm

Not sure what is your configuration. Can you please share?

Reply

Adolpho

February 4, 2019 at 2:01 am

Hello,

Your site is amazing and post was just what a need.

Always that a do a Post request my postman got a 403 status. Its only works when security is enable.

How could i work with security in this example?

Thank for your help,
Adolpho

Reply

harish kumar

May 29, 2018 at 1:23 pm

How do it for multiple objects in json and also i tried to consume for existing multiple object from json but i am getting array exception.

please post the demo on consuming rest api for multiple objects

Reply

Girish

April 27, 2018 at 11:42 am

i am getting login page requesting username and password ? What is the username & password to login to application.

Reply

Lokesh Gupta

April 27, 2018 at 12:02 pm

Please remove “spring-boot-starter-security” dependency from `pom.xml`.

For customizing the security, use information given in https://howtodoinjava.com/spring-boot/role-based-security-jaxrs-annotations/

Reply

Hareesh

July 10, 2019 at 6:14 am

yes its working once i commented the spring-boot-starter-security dependency and maven clean and install
thanks

Reply

Jackie Zheng

October 9, 2019 at 11:08 pm

Hi Hareesh,

I have deleted the security dependency from pom.xml and run again, it tells The webpage cannot be found. How did you get yours working? Thanks.

Reply

Dhruv

May 20, 2020 at 4:50 pm

Hi,
I faced the same issue and took me some time to resolve this. I am using Spring boot 2.3.0 version.
Commenting pring-boot-starter-security in pom.xml didn't help me, neither "security.basic.enabled=false"
Anyways, what i did was created a new Configuration class "WebSecurityConfig.java" , like so and allowed all requests.

Also maybe for some of you, this cud help "Spring Boot 2.0 disable default security"- Stack Overflow

@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends WebSecurityConfigurerAdapter{
@Override
protected void configure(HttpSecurity http) throws Exception{
http.authorizeRequests().antMatchers("/").permitAll();
}
}

Reply

srini

November 28, 2017 at 10:40 am

Good One!, when I run it asks for user name and pwd for authentication

Reply

Girish

April 27, 2018 at 11:43 am

Even am being asked. did u get an answer for that ?

Reply

Venu

January 20, 2017 at 9:42 am

Hi Lokesh,
When I importef the project and ran, the application runs but exits.

Reply

Dinesh Krishnan

December 30, 2016 at 5:18 pm

Thanks for sharing this. Can you do tutorials about, Restful webservice security.?
Thanks in advance

Reply

Vikram Hiraman Gore

December 2, 2016 at 8:41 pm

Hi Lokesh ,
Which book or web link is good for becoming expert in spring boot. I having good knowledge of it.
Thanks and Regards,
Vikram
9028163305

Reply

Ankur

November 21, 2016 at 8:21 pm

Hi Lokesh,

I am new to Spring boot thanks for your efforts putting this tutorial.

Can we create parent child modules with spring boot?
e.g. My student project have web-module that creates war file and it uses, created jar from service/data module and business module
or
We create/add them separately and update pom file?

Thanks

Reply

adarsha. Lunia

October 7, 2016 at 9:34 am

Hi Lokesh,

I have requirement in our project that, We need to integrate our spring application with Kibana(ELK) to get total response and request time for API request Method (GET,PUT). Currently we are using Splunk. Can you please let me know if have any idea in this ?, By the way our project is Spring boot.

Reply

rahul

September 25, 2016 at 9:34 am

pls upload angularjs tutorial

Reply

Tobias

January 5, 2020 at 4:46 pm

Hello Likesh,
thank you for the nice and helpfull tutorial!

On sending one request I got this error: “No converter found for return value of type”
To solve this, I needed to add public getters for all fields of the class Employee.
I hope this helps anyone…

BR

Reply

Leave a Comment

Name *

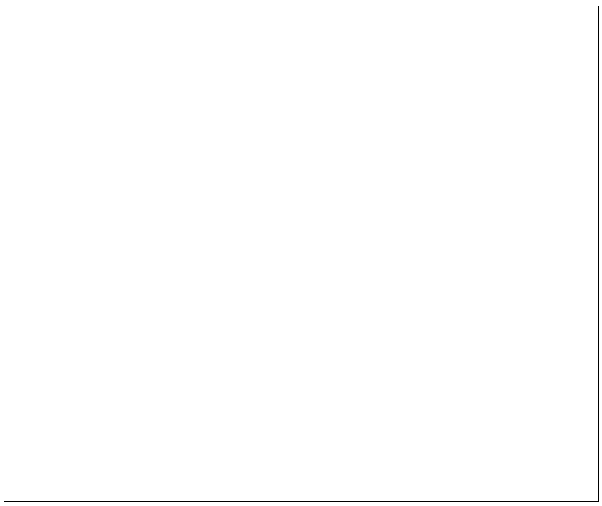
Email *

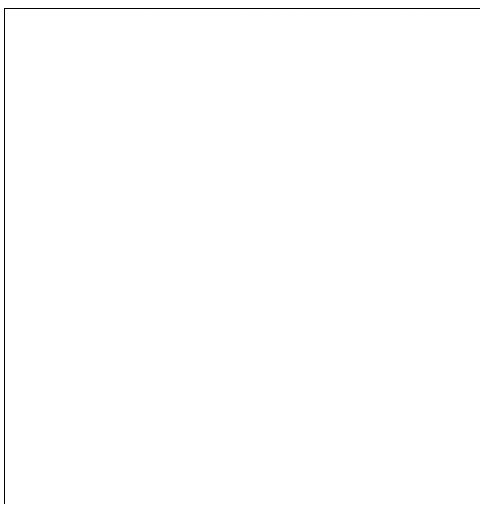
Website

Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment

Search …





HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- About Me
- Contact Us

- Privacy policy
- Advertise
- Guest Posts

Blogs

REST API Tutorial