Spring 4 MVC REST Controller Example (JSON CRUD Tutorial)

BY VIRAL PATEL · JULY 6, 2016

Spring 4 MVC REST Controller Service Example (JSON CRUD Tutorial) – A step by step tutorial to understand Spring 4 MVC REST API and to create RESTful service using Spring 4.

RECENT POSTS

Spring 4 MVC REST Controller Example (JSON CRUD Tutorial)

Spring 4 MVC Tutorial Maven Example – Spring Java Configuration

Find Process ID of Process using a Port in Windows

How to Embed Tomcat within Maven Project – Run Tomcat with Maven

Bootstrap Navbar Menu without JavaScript

Spring 4 MVC REST provides powerful APIs to built complete RESTful services. Let us understand the core concept and create simple web application using Maven and make our CRUD based REST service.

Getting started with Spring 4 MVC REST Controller

For this project we will use following tools and technologies.

- Java 1.7
- Spring MVC 4.3.0.RELEASE
- Tomcat 7
- Maven 3
- POSTMan (optional)

The demo REST application will have Customer resource. This customer resource can be accessed using standard `GET`, `POST`, `PUT`, `DELETE` http methods. We will create below REST endpoints for this project.

REST Endpoint	HTTP Method	Description
/customers	GET	Returns the list of customers
/customers/{id}	GET	Returns customer detail for given customer {id}
/customers	POST	Creates new customer from the post data
/customers/{id}	PUT	Replace the details for given customer {id}
/customers/{id}	DELETE	Delete the customer for given customer {id}

1. Create a new Maven Project

If you are using Eclipse then you can use M2Eclipse plugin. Check the tutorial Spring 4 MVC Hello World and follow the section 1.

Alternatively if you want to generate Maven webapp using `mvn` command then follow these steps.

```
mvn archetype:create
  -DgroupId=net.viralpatel.spring
  -DartifactId=SpringRest
  -DarchetypeArtifactId=maven-archetype-webapp
```

This will generate maven application with default project directory structure. You can then run following command and convert the project in Eclipse project.

```
mvn eclipse:eclipse
```

And then simply import the project in Eclipse.

2. Add Spring 4 MVC Maven dependencies (Update pom.xml)

Project structure is created. Now let's start and add first the maven dependencies for Spring 4 MVC REST in our `pom.xml` file.

Update pom.xml file and add following dependencies.

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xs
    <modelVersion>4.0.0</modelVersion>
    <groupId>net.viralpatel.spring</groupId>
    <artifactId>Spring4Rest</artifactId>
    <packaging>war</packaging>
    <version>0.0.1-SNAPSHOT</version>
    <name>Spring 4 Rest Service CRUD Example</name>
    <url>http://maven.apache.org</url>
    <properties>
        <java-version>1.7</java-version>
        <springframework.version>4.3.1.RELEASE</springframework.version>
        <jackson.version>2.7.5</jackson.version>
    </properties>
    <dependencies>
16
25
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.0.1</version>
            <scope>provided</scope>
        </dependency>
```

```
        </dependencies>
        <build>
            <finalName>HelloWorld</finalName>
            <pluginManagement>
                <plugins>
                    <plugin>
                        <groupId>org.apache.tomcat.maven</groupId>
                        <artifactId>tomcat7-maven-plugin</artifactId>
                        <version>2.2</version>
                        <configuration>
                            <path>/springrest</path>
                        </configuration>
                    </plugin>
                </plugins>
            </pluginManagement>
        </build>
    </project>
```

After updating pom.xml, Eclipse's maven plugin should start resolving the dependencies.

3. Set Annotation based Configuration for Spring 4 MVC REST

For this Spring 4 MVC REST tutorial we are going to use Spring's Java based configuration or **annotation based configuration** instead of old XML configuration. So now let us add the Java Configuration required to bootstrap Spring 4 MVC REST in our webapp.

Create `AppConfig.java` file under `/src` folder. Give appropriate package name to your file. We are using `@EnableWebMvc`, `@ComponentScan` and `@Configuration` annotations. These will bootstrap the spring mvc application and set package to scan controllers and resources.

```
/src/main/java/net/viralpatel/spring/config/AppConfig.java

package net.viralpatel.spring.config;

import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.EnableWebMvc;

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "net.viralpatel.spring")
public class AppConfig {

}
```

4. Set Servlet 3 Java Configuration

Create `AppInitializer` class under config package. This class will replace web.xml and it will map the spring's dispatcher servlet and bootstrap it.

/src/main/java/net/viralpatel/spring/config/AppInitializer.java

```
package net.viralpatel.spring.config;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class AppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

    @Override
    protected Class[] getRootConfigClasses() {
        return new Class[] { AppConfig.class };
    }

    @Override
    protected Class[] getServletConfigClasses() {
        return null;
    }

    @Override
    protected String[] getServletMappings() {
        return new String[] { "/" };
    }

}
```

We have configured the dispatcher servlet using standard Java based configuration instead of the older web.xml. Thus web.xml is no longer required and we can simply delete it.

5. Create the Customer Model

Next let us create `Customer` model class that will have few properties such as firstName, lastName, email etc. This bean will hold customer information.

/src/main/java/net/viralpatel/spring/model/Customer.java

```
package net.viralpatel.spring.model;

import java.util.Date;

public class Customer {

    private Long id;
    private String firstName;
```

```
    private String lastName;
    private String email;
    private String mobile;
    private Date dateOfBirth;

    public Customer(long id, String firstName, String lastName, String email, String mobile) {
        this.id = id;
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.mobile = mobile;
        this.dateOfBirth = new Date();
    }

    public Customer() {
    }

    //..Getter and setter methods

}
```

6. Create the Dummy Customer Data Access Object (DAO)

Instead of storing the customer data in database and to make this example simple, we will create a dummy data access object that will store customer details in a list. This DAO class can be easily replaced with Spring Data DAO or custom DAO. But for this example we will keep it easy.

The CustomerDAO contains methods `list()`, `get()`, `create()`, `update()` and `delete()` to perform CRUD operation on customers.

```
/src/main/java/net/viralpatel/spring/dao/CustomerDAO.java

package net.viralpatel.spring.dao;

import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Component;

import net.viralpatel.spring.model.Customer;

@Component
public class CustomerDAO {

    // Dummy database. Initialize with some dummy values.
    private static List customers;
    {
```

```
        customers = new ArrayList();
        customers.add(new Customer(101, "John", "Doe", "djohn@gmail.com", "121-232-3435"));
        customers.add(new Customer(201, "Russ", "Smith", "sruss@gmail.com", "343-545-2345"));
        customers.add(new Customer(301, "Kate", "Williams", "kwilliams@gmail.com", "876-237-2987"));
    }

    /**
     * Returns list of customers from dummy database.
     *
     * @return list of customers
     */
    public List list() {
        return customers;
    }

    /**
     * Return customer object for given id from dummy database. If customer is
     * not found for id, returns null.
     *
     * @param id
     *            customer id
     * @return customer object for given id
     */
    public Customer get(Long id) {

        for (Customer c : customers) {
            if (c.getId().equals(id)) {
                return c;
            }
        }
        return null;
    }

    /**
     * Create new customer in dummy database. Updates the id and insert new
     * customer in list.
     *
     * @param customer
     *            Customer object
     * @return customer object with updated id
     */
    public Customer create(Customer customer) {
        customer.setId(System.currentTimeMillis());
        customers.add(customer);
        return customer;
    }

    /**
     * Delete the customer object from dummy database. If customer not found for
     * given id, returns null.
     *
```

```
     * @param id
     *            the customer id
     * @return id of deleted customer object
     */
    public Long delete(Long id) {

        for (Customer c : customers) {
            if (c.getId().equals(id)) {
                customers.remove(c);
                return id;
            }
        }

        return null;
    }

    /**
     * Update the customer object for given id in dummy database. If customer
     * not exists, returns null
     *
     * @param id
     * @param customer
     * @return customer object with id
     */
    public Customer update(Long id, Customer customer) {

        for (Customer c : customers) {
            if (c.getId().equals(id)) {
                customer.setId(c.getId());
                customers.remove(c);
                customers.add(customer);
                return customer;
            }
        }

        return null;
    }

}
```

7. Create the Customer REST Controller

Now let us create `CustomerRestController` class. This class is annotated with `@RestController` annotation. Also note that we are using new annotations @GetMapping, @PostMapping, @PutMapping and @DeleteMapping instead of standard @RequestMapping. These annotations are available since Spring MVC 4.3 and are standard way of defining REST endpoints. They act as wrapper to @RequestMapping. For example @GetMapping is a

composed annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).

/src/main/java/net/viralpatel/spring/controller/CustomerRestController.java

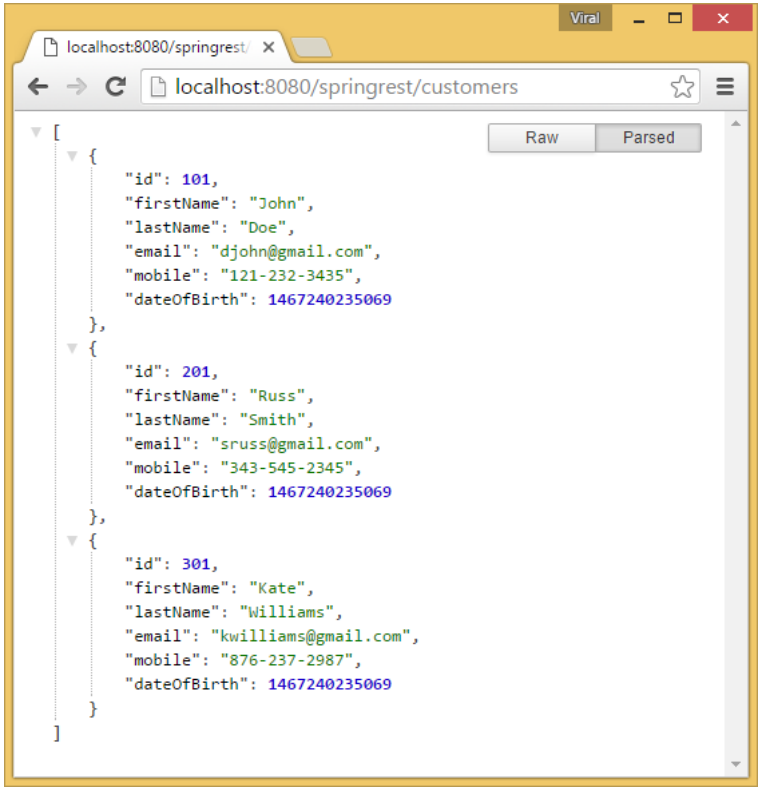
```
package net.viralpatel.spring.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import net.viralpatel.spring.dao.CustomerDAO;
import net.viralpatel.spring.model.Customer;

@RestController
public class CustomerRestController {


    @Autowired
    private CustomerDAO customerDAO;


    @GetMapping("/customers")
    public List getCustomers() {
        return customerDAO.list();
    }

    @GetMapping("/customers/{id}")
    public ResponseEntity getCustomer(@PathVariable("id") Long id) {

        Customer customer = customerDAO.get(id);
        if (customer == null) {
            return new ResponseEntity("No Customer found for ID " + id, HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity(customer, HttpStatus.OK);
    }

    @PostMapping(value = "/customers")
    public ResponseEntity createCustomer(@RequestBody Customer customer) {
```

```
        customerDAO.create(customer);

        return new ResponseEntity(customer, HttpStatus.OK);
    }

    @DeleteMapping("/customers/{id}")
    public ResponseEntity deleteCustomer(@PathVariable Long id) {

        if (null == customerDAO.delete(id)) {
            return new ResponseEntity("No Customer found for ID " + id, HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity(id, HttpStatus.OK);

    }

    @PutMapping("/customers/{id}")
    public ResponseEntity updateCustomer(@PathVariable Long id, @RequestBody Customer customer) {

        customer = customerDAO.update(id, customer);

        if (null == customer) {
            return new ResponseEntity("No Customer found for ID " + id, HttpStatus.NOT_FOUND);
        }

        return new ResponseEntity(customer, HttpStatus.OK);
    }

}
```

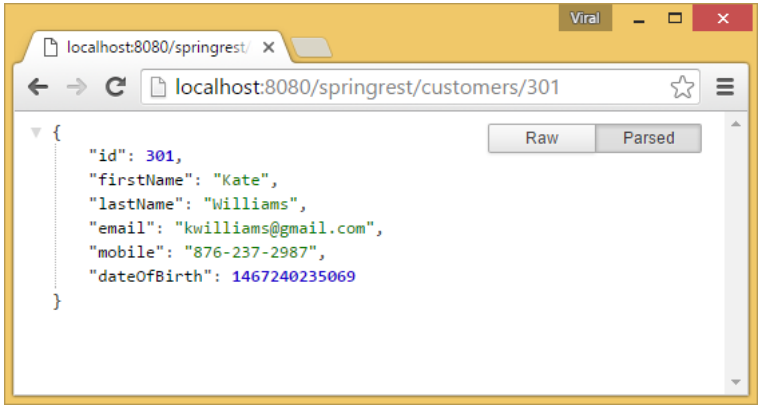
That's All Folks

Let us execute the Spring REST project. In Eclipse you can start Tomcat and run the project inside it. Or you can run the project using Embedded Tomcat using Maven.

Once the application starts successfully, launch the browser and open http://localhost:8080/springrest/customers . This hit the /customers end point and will list down all the customers.

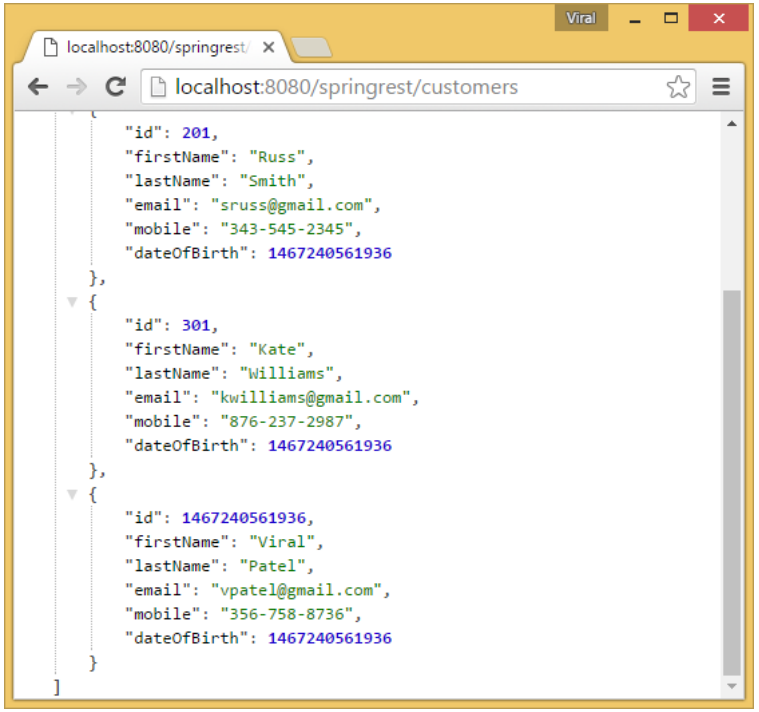


Next we can try GET method by hitting `http://localhost:8080/springrest/customers/{id}` endpoint. This will display details of customer for given id.



Also you can POST the customer details to `http://localhost:8080/springrest/customers` using POSTMan extension. Once you do that, the new customer will be created and same can be viewed under

/customers endpoint.



Download Source Code – Spring 4 REST Tutorial

Source code of this Spring 4 REST Controller tutorial is available in Github.

Download – spring4-rest-example.zip (6.54 KB)
Github – spring4-rest-example.git

Related Articles

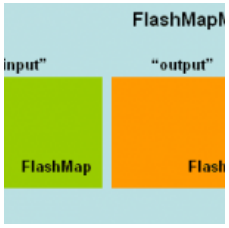
1. **Spring 4 MVC Tutorial Maven Example – Spring Java Configuration**
2. **Spring MVC Flash Attribute tutorial with example**
3. **Spring 3 MVC – Autocomplete with JQuery & JSON example**
4. **Spring @RequestHeader Annotation example**
5. **Spring 3 MVC Interceptor tutorial with example**
6. **Spring MVC Exception Handling using @ControllerAdvice annotation**
7. **Spring MVC Cookie example**

Tags: rest restful

PREVIOUS STORY
Spring 4 MVC Tutorial Maven Example – Spring Java Configuration

YOU MAY ALSO LIKE...

Spring MVC Multiple File Upload example



Spring MVC Flash Attribute tutorial with example

Change spring-servlet.xml Filename (Spring Web Contenxt Configuration Filename)

Tutorial:Struts Spring framework example in Eclipse.

41 COMMENTS

Vimalkumar Patel 6 July, 2016, 7:25

1. You can use a map since you are using ID for customer model, this is to make it efficient.
2. It would be interesting to see Jpa Repository integrated in this demo app. That would take dummy code out of picture altogether.
3. Tweet again when it works in the first attempt :)

Reply

Viral Patel 8 July, 2016, 7:35

Thanks Vimal.. Integration with Spring Data is coming next.

Reply

sachin 28 September, 2016, 12:31

maybe this is a bit late but i have modified the code to include a HashMap

private static HashMap customers;
{

customers = new HashMap();
customers.put(101, new Customer(101, “John”, “Doe”, “djohn@gmail.com”, “121-232-3435”));
customers.put(201, new Customer(201, “Russ”, “Smith”, “sruss@gmail.com”, “343-545-2345”));
customers.put(301, new Customer(301, “Kate”, “Williams”, “kwilliams@gmail.com”, “876-237-2987”));
}

Reply

Ssingh 7 March, 2017, 12:20

I am facing this issue :(
Mar 07, 2017 12:02:17 PM org.apache.catalina.core.StandardContext listenerStart
SEVERE: Exception sending context initialized event to listener instance of clas
s org.springframework.web.context.ContextLoaderListener
java.lang.IllegalStateException: Cannot initialize context because there is alre
ady a root application context present – check whether you have multiple Context
Loader* definitions in your web.xml!

Reply

Krishnan Narayanan 8 July, 2016, 1:41

The blog looks good when you read it. However,wWhen you click on Download link (spring-rest-example.zip) you get “spring4-mvc-example-master (1).zip” and when you go to github and download the zip file you get “spring4-restful-example-master.zip”. Regardless both of them does not work.
I have changed the jva version to 1.8 and tried without anysuccess.

Reply

Viral Patel 8 July, 2016, 7:22

Hi Krishnan, Thanks for your feedback. I have fixed the Download link. Now you should be able to download correct source. Did you try running mvn tomcat7:run and execute the project? Let me know what errors are you getting while running this example.

Reply

kuldip h soni 2 August, 2016, 9:53

Hi Virat,
Thanks for such a nice blog
I have followed all the steps, and when i tried accessing below url
http://localhost:8080/SpringRest/
i can see o/p as “Hello World”
but when i am trying below url:
http://localhost:8080/SpringRest/customers
o/p is “404-page not found”
re verified all the steps but same o/p.
could you kindly help

Reply

gaurav 21 August, 2016, 23:39

Build failing
Error assembling WAR: webxml attribute is required (or pre-existing WEB-INF/web.xml if executing in update mode) –

Reply

ravi 29 September, 2016, 20:15
did you get this resolved. please let me know

Reply

Sarang 25 October, 2016, 8:55
You can add this in the plugin section of pom.xml to avoid this error:

maven-war-plugin
2.4

false

Reply

Carmen Stira 1 September, 2016, 18:12
Hi
Thanks for your guide.
I tried it on Tomcat 7 with JRE 7 and it works fine.

I modified the pom.xml adding the maven-war-plugin 2.6 with the failOnMissingWebXml=false to deploy the war on JBoss with JRE 7.

On JBoss 7 or JBoss EAP 6 the module is deployed and the welcome page works but the /customers or /customers/{id} call doesn't work.

>JBWEB000069: description JBWEB000124: The requested resource is not available.

Any idea ?
Thanks

Carmen

Reply

ravi 30 September, 2016, 2:25
did you figure it out?

Reply

Anil Samuel 27 March, 2017, 9:43
I have same issue; works in Tomcat 8.

Reply

Will 2 September, 2016, 21:28
Hi Viral, nice REST server.

Any chance of seeing this code integrated with an AngularJS front-end/UI?

Will

Reply

Ishan Sharma 16 September, 2016, 22:31

Excellent thanks…. helped me to understand many points which were creating error in my project :-)

Reply

D Prasad 19 September, 2016, 17:13
Hi Viral, nice REST server.

Any chance of seeing this code integrated with an AngularJS front-end/UI?

Will

Reply

> **Ravikumar** 28 October, 2016, 12:51
> its vary simple, make a ajax call and store the data into a variable. then all your data will be at front end then you can play with ng-repeat and ng-bind. That's all..!
>
> Reply

D Prasad 19 September, 2016, 17:14
i have the same question as will

Reply

ravi 29 September, 2016, 21:04
Build failing
Error assembling WAR: webxml attribute is required (or pre-existing WEB-INF/web.xml if executing in update mode) –

Reply

> **ravi** 30 September, 2016, 2:26
> figured it out.
>
> org.apache.maven.plugins
> maven-war-plugin
>
> false
>
> Reply

java blogger 1 October, 2016, 17:41
Hi,

Is there a way we can have both Spring MVC and Struts2 in a single web app project?
If so do you have any samples of such project.

Reply

Dinesh Krishnan 14 October, 2016, 18:54
Thanks for sharing it was very useful to under the concept.

Reply

Abbas 17 October, 2016, 17:54
i am getting this error when i configure project java based ,Please help
Oct 17, 2016 5:19:31 PM org.apache.catalina.core.AprLifecycleListener init
INFO: The APR based Apache Tomcat Native library which allows optimal performance in production

environments was not found on the java.library.path: C:\Program
Files\Java\jre1.8.0_45\bin;C:\Windows\Sun\Java\bin;C:\Windows\system32;C:\Windows;C:/Program
Files/Java/jre1.8.0_45/bin/server;C:/Program Files/Java/jre1.8.0_45/bin;C:/Program
Files/Java/jre1.8.0_45/lib/amd64;C:\oracle\product\10.2.0\client_1\bin;C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\Syste
Files (x86)\Fuse\;C:\Program Files (x86)\Skype\Phone\;C:\Program
Files\nodejs\;C:\Users\ghulam.abbas\AppData\Roaming\npm;D:\eclipse;;.
Oct 17, 2016 5:19:31 PM org.apache.tomcat.util.digester.SetPropertiesRule begin
WARNING: [SetPropertiesRule]{Server/Service/Engine/Host/Context} Setting property 'source' to
'org.eclipse.jst.jee.server:Spring4Rest' did not find a matching property.
Oct 17, 2016 5:19:31 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["http-bio-8080"]
Oct 17, 2016 5:19:31 PM org.apache.coyote.AbstractProtocol init
INFO: Initializing ProtocolHandler ["ajp-bio-8009"]
Oct 17, 2016 5:19:31 PM org.apache.catalina.startup.Catalina load
INFO: Initialization processed in 328 ms
Oct 17, 2016 5:19:31 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service Catalina
Oct 17, 2016 5:19:31 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet Engine: Apache Tomcat/7.0.47
Oct 17, 2016 5:19:31 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-bio-8080"]
Oct 17, 2016 5:19:31 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["ajp-bio-8009"]
Oct 17, 2016 5:19:31 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 304 ms

Reply

Fin 5 November, 2016, 21:23
How can I handle requestparam with bracket :
myhost.com?film&sort[title]=desc

Reply

> **kunal** 22 December, 2016, 11:04
> myhost.com?==:film&sort[webname]=desc
>
> Reply

bala 6 December, 2016, 18:16
Awesome Post Sir Very Clean Explanation. Thanks For Posting

Reply

Dhruv 7 December, 2016, 18:39
can u please tell how to pass list/data to jsp using getmapping method.

Reply

Manoj Dhanji 8 December, 2016, 1:45
It should be mvn archetype:generate for creating the project. The tutorial, otherwise is a good starting
point. Thanks

Reply

Nikhil 11 December, 2016, 15:28
How can I generate response in XML ? Response getting is in JSONArray format.Can anyone please help.
Reply

Nikhil Pate 12 December, 2016, 15:23
Very nice tutorial.But one question,this code is always returning response in JSON format.How can I change it to XML ? Can any one please suggest ?
Reply

Rakesh 18 December, 2016, 19:04
Very nice blog . Its working for me.Patel Jee.
Reply

ramesh 20 December, 2016, 14:24
org.apache.tomcat.maven
tomcat7-maven-plugin
2.2
/springrest
maven-war-plugin
2.4
false
Reply

Dinesh Krishnan 20 December, 2016, 15:15
Very Good Post, Thanks for sharing.
Reply

pink 3 February, 2017, 18:11
Hi ,
Iam getting 404 error while accessing : http://localhost:8080/spring/customers
please let me know what to do ? I have followed the same as in documentation.
Reply

pink 3 February, 2017, 18:12
Hi ,
Iam getting page not found error while accessing : http://localhost:8080/spring/customers
please let me know what to do ? I have followed the same as in documentation.
Reply

pink 6 February, 2017, 12:40
Hi,

any one please give me the solution for this error? Please??

Reply

song lei 13 February, 2017, 14:21

for the error http://localhost:8080/spring/customers not found.
firstly : add the old web.xml back.
secondly: in pom.xml , remove the only leave as below:

org.apache.tomcat.maven
tomcat7-maven-plugin
2.2

thirdly : access it like : http://localhost:8080/webService/customers eg: webService is my project name

Reply

> **song lei** 13 February, 2017, 14:23
>
> secondly: in pom.xml , remove the one line “configutation” at the bottom part plugin area
>
> Reply

Caíque 9 March, 2017, 4:14

If you are trying to do a “Maven Build” from eclipse, you must add the goal “spring-boot:run”, otherwise it won’t initialize the spring boot and you won’t be able to see the pages.

Reply

Bala 15 March, 2017, 19:00

HI
Good article, but I am facing issues when I try to http://localhost:8080/spring/customers

WARNING: No mapping found for HTTP request with URI [/SpringRest/] in DispatcherServlet with name ‘dispatcher’

I am using inteliJ IDE

thanks,
Bala

Reply

Dinesh Krishnan 22 March, 2017, 20:52

Useful website…. keep up the good work

Reply

LEAVE A REPLY

Comment

Name *

Email *

Website

Post Comment