WebSystique
learn together

Spring 4 MVC REST Service Example using @RestController

August 3, 2014 websystiqueadmin

Spring provides first class support for developing REST services. In this article, we will be developing a Spring 4 MVC based `RESTful JSON service` & `RESTful XML service` using Spring 4 `@RestController` annotation. Spring, behind the scenes, uses HttpMessageConverters to convert the response into desired format [JSON/XML/etc..] based on certain libraries available on the classpath and optionally, **Accept** Headers in request.

In order to serve JSON, we will be using `Jackson library` [jackson-databind.jar]. For XML, we will use `Jackson XML extension` [jackson-dataformat-xml.jar]. Mere presence of these libraries in classpath will trigger Spring to convert the output in required format. Additionally, We will go a step further by annotating the domain class with `JAXB annotations` to support XML in case Jackson's XML extension library is not available for some reason.

Note: If you are sending the request by just typing the URL in browser, you may add the suffix [.xml/.json] which help spring to determine the type of content to be served.

In case you want to dive deeper in details, have a look at Spring MVC 4 RESTFul Web Services CRUD Example+RestTemplate post.

Other interesting posts you may like

- Spring 4 MVC+JPA2+Hibernate Many-to-many Example
- Spring 4 Caching Annotations Tutorial

WebSystiqu
G+ Follow
+ 234

Recent Posts

Spring 4 Caching Annotations Tutorial
Spring 4 Cache Tutorial with EhCache
Spring 4 MVC+JPA2+Hibernate Many-to-many-Example
Spring 4 Email using Velocity,Freemarker Template library
Spring 4 Email With Attachment Tutorial

- Spring 4 Cache Tutorial with EhCache
- Spring 4 Email Template Library Example
- Spring 4 Email With Attachment Tutorial
- Spring 4 Email Integration Tutorial
- Spring MVC 4+JMS+ActiveMQ Integration Example
- Spring 4+JMS+ActiveMQ @JmsLister @EnableJms Example
- Spring 4+JMS+ActiveMQ Integration Example
- Spring MVC 4+Apache Tiles 3 Integration Example
- Spring MVC 4+Spring Security 4 + Hibernate Integration Example
- Spring MVC 4+AngularJS Example
- Spring MVC 4+AngularJS Routing with ngRoute Example
- Spring MVC 4+AngularJS Routing with UI-Router Example
- Spring MVC 4 RESTFul Web Services CRUD Example with Full REST support + RestTemplate
- Spring MVC @RequestBody, @ResponseBody,HttpMessageConverters Example

Following technologies being used:

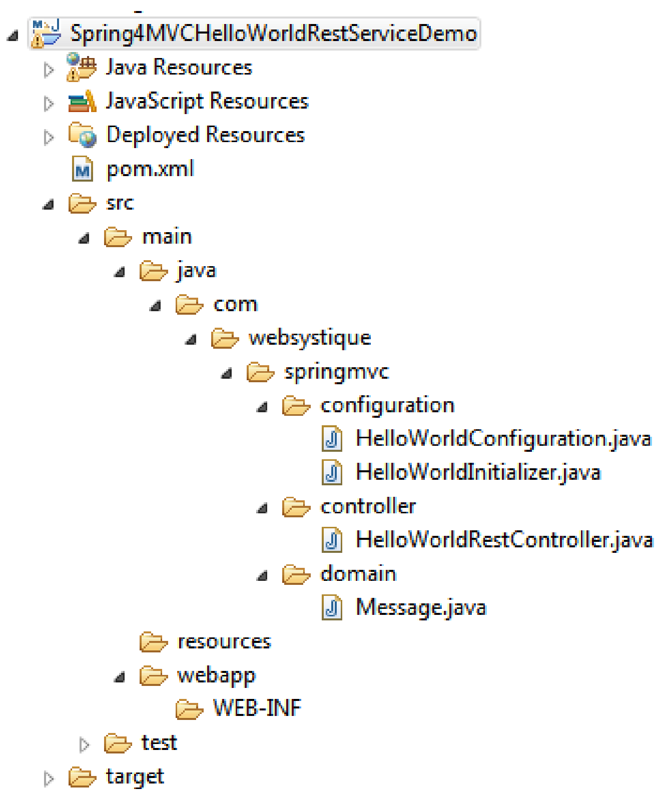
- Spring 4.3.0.RELEASE
- jackson-databind 2.7.5
- jackson-dataformat-xml 2.7.5
- Maven 3
- JDK 1.7
- Tomcat 8.0.21
- Eclipse MARS.1

Let’s begin.

Step 1: Create the directory structure

Post Creating a maven web project with eclipse contains step-by-step instruction to create a `maven` project with eclipse.

Following will be the final project structure:



```
Spring4MVCHelloWorldRestServiceDemo
    Java Resources
    JavaScript Resources
    Deployed Resources
    pom.xml
    src
        main
            java
                com
                    websystique
                        springmvc
                            configuration
                                HelloWorldConfiguration.java
                                HelloWorldInitializer.java
                            controller
                                HelloWorldRestController.java
                            domain
                                Message.java
            resources
            webapp
                WEB-INF
        test
    target
```

We will be using Spring Java configuration with no xml. Now let's add/update the content mentioned in above project structure.

Step 2: Update pom.xml with required dependencies

```
<?xml version="1.0"?>
<project
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.ap
    xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.or

    <modelVersion>4.0.0</modelVersion>
    <groupId>com.websystique.springmvc</groupId>
    <artifactId>Spring4MVCHelloWorldRestServiceDemo</artifactId>
    <packaging>war</packaging>
    <version>1.0.0</version>
    <name>Spring4MVCHelloWorldRestServiceDemo Maven Webapp</name>

    <properties>
        <springframework.version>4.3.0.RELEASE</springframework.version>
        <jackson.library>2.7.5</jackson.library>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-core</artifactId>
            <version>${springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-web</artifactId>
            <version>${springframework.version}</version>
        </dependency>
        <dependency>
```

```
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${springframework.version}</version>
        </dependency>

        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>3.1.0</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>${jackson.library}</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.dataformat</groupId>
            <artifactId>jackson-dataformat-xml</artifactId>
            <version>${jackson.library}</version>
        </dependency>
    </dependencies>


    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.2</version>
                    <configuration>
                        <source>1.7</source>
                        <target>1.7</target>
                    </configuration>
                </plugin>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-war-plugin</artifactId>
                    <version>2.4</version>
                    <configuration>
                        <warSourceDirectory>src/main/webapp</warSourceDir
                        <warName>Spring4MVCHelloWorldRestServiceDemo</war
                        <failOnMissingWebXml>false</failOnMissingWebXml>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>

        <finalName>Spring4MVCHelloWorldRestServiceDemo</finalName>
    </build>
</project>
```

Main dependencies to be noticed here are `Jackson library` (jackson-databind) which will be used to convert the response data into `JSON` string, and `Jackson XML Extension library` (jackson-dataformat-xml) which will help to provide XML converted response. Again, if the jackson-dataformat-xml is not included, only JSON response will be served, unless the domain object is annotated explicitly with JAXB annotations.

Step 3: Add a Pojo/domain object

```
package com.websystique.springmvc.domain;
```

```
public class Message {

    String name;
    String text;

    public Message(String name, String text) {
        this.name = name;
        this.text = text;
    }

    public String getName() {
        return name;
    }

    public String getText() {
        return text;
    }

}
```

Above object will be returned from controllers and converted by Jackson into JSON format. If the jackson-dataformat-xml is present, then it can also be converted into XML.

Step 4: Add a Controller

Add a controller class under `src/main/java` with mentioned package as shown below.

```
package com.websystique.springmvc.controller;

import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.websystique.springmvc.domain.Message;

@RestController
public class HelloWorldRestController {

    @RequestMapping("/")
    public String welcome() {//Welcome page, non-rest
        return "Welcome to RestTemplate Example.";
    }

    @RequestMapping("/hello/{player}")
    public Message message(@PathVariable String player) {//REST Endpoint.

        Message msg = new Message(player, "Hello " + player);
        return msg;
    }
}
```

`@PathVariable` indicates that this parameter will be bound to variable in URI template. More interesting thing to note here is that here we are using `@RestController` annotation, which marks this class as a controller where every method returns a domain object/pojo instead of a view. It means that we are no more using view-resolvers, we are no more directly sending the html in response but we are sending domain object converted into format understood by the consumers. In our case, due to jackson library included in class path, the Message object will be converted into `JSON format` [or in XML if either the jackson-dataformat-xml.jar is present in classpath or Model class i annotated with JAXB annotations].

Step 5: Add Configuration Class

```
package com.websystique.springmvc.configuration;

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

@Configuration
@EnableWebMvc
@ComponentScan(basePackages = "com.websystique.springmvc")
public class HelloWorldConfiguration {


}
```

Here this class is mainly providing the component-scanning and annotation support.Note that we don’t have any view-resolvers configured as we don’t need one in Rest case.

Step 6: Add Initialization class

Add an initializer class as shown below(which in this case acts as replacement of any spring configuration defined in web.xml). During Servlet 3.0 Container startup, this class will be loaded and instantiated.

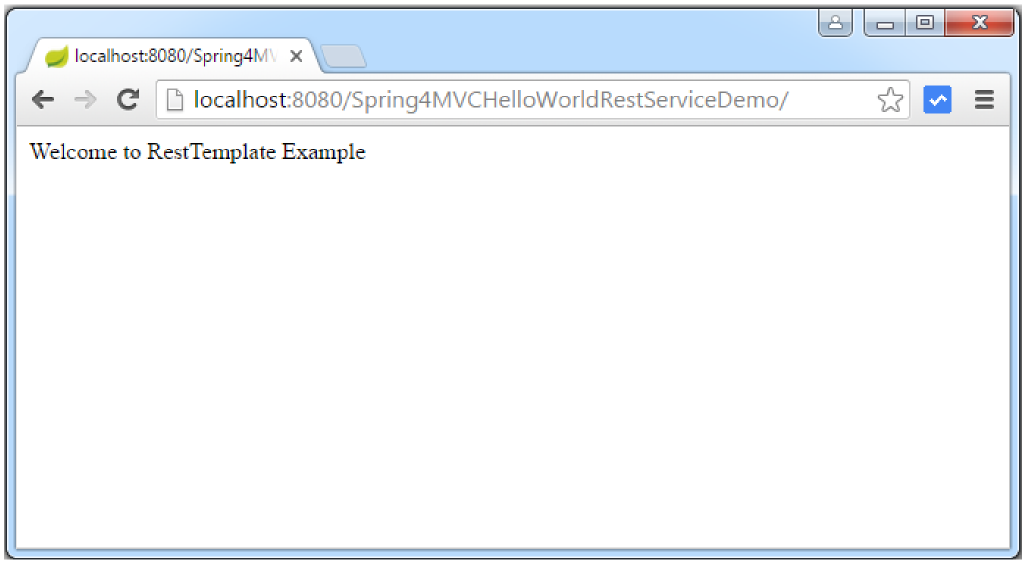
```
package com.websystique.springmvc.configuration;

import org.springframework.web.servlet.support.AbstractAnnotationConfigDi

public class HelloWorldInitializer extends AbstractAnnotationConfigDispat

    @Override
    protected Class<?>[] getRootConfigClasses() {
        return new Class[] { HelloWorldConfiguration.class };
    }

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

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

}
```

Step 7: Build and Deploy the application

Now build the war (either by eclipse using m2e plugin) or via maven command line(mvn clean install). Deploy the war to a Servlet 3.x container . Since here i am using Tomcat, i will simply put this war file into `tomcat webapps` folder and click on `startup.bat` inside tomcat bin directory.

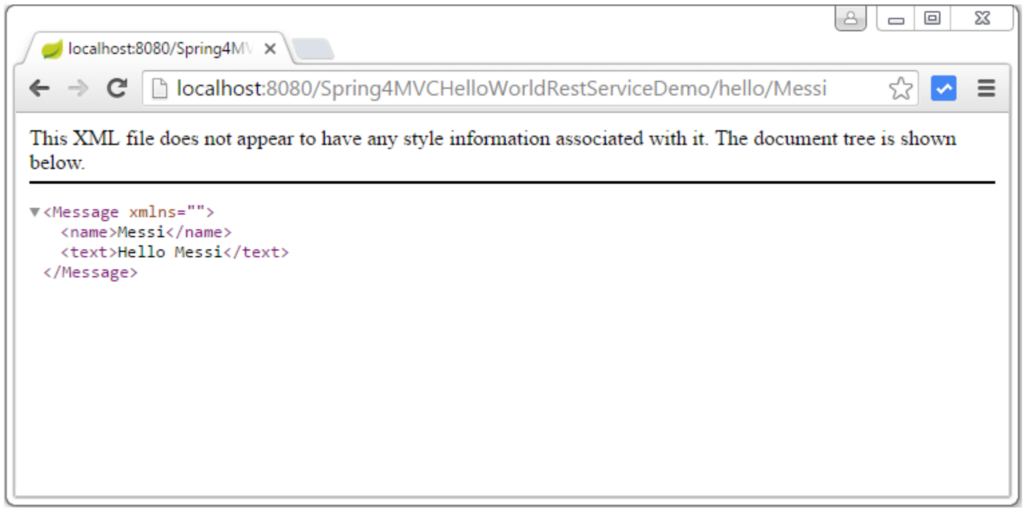
In order to test it, you can use either the browser or a true-client.POSTMAN is a nice tool to test your REST Endpoints as in a real scenario. Advantage of Postman is that you can send the "Accept" header along with request which will then be used by Spring while sending the response in required format. With browsers it is not so straight-forward to send the "Accept" Header but you can suffix the URL with format[.json/.xml] to get similar results.

Let's start with the browser.



Welcome to RestTemplate Example

Now let's access the REST Endpoint. Please note that since we have included the jackson-dataformat-xml.jar in classpath, the response you will get will be XML.


localhost:8080/Spring4M
localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/Messi

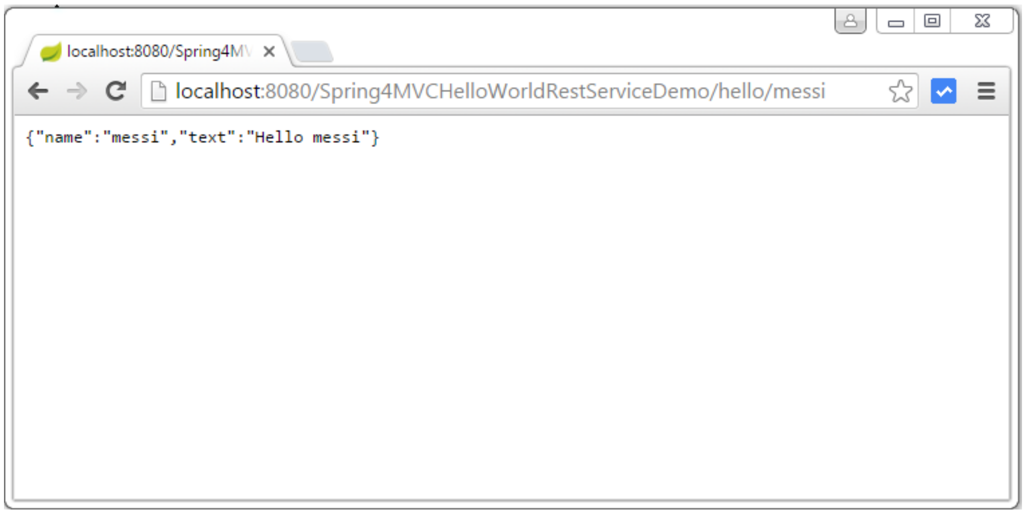
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<Message xmlns="">
   <name>Messi</name>
   <text>Hello Messi</text>
 </Message>
```

If you want Spring to serve JSON response instead, you can either

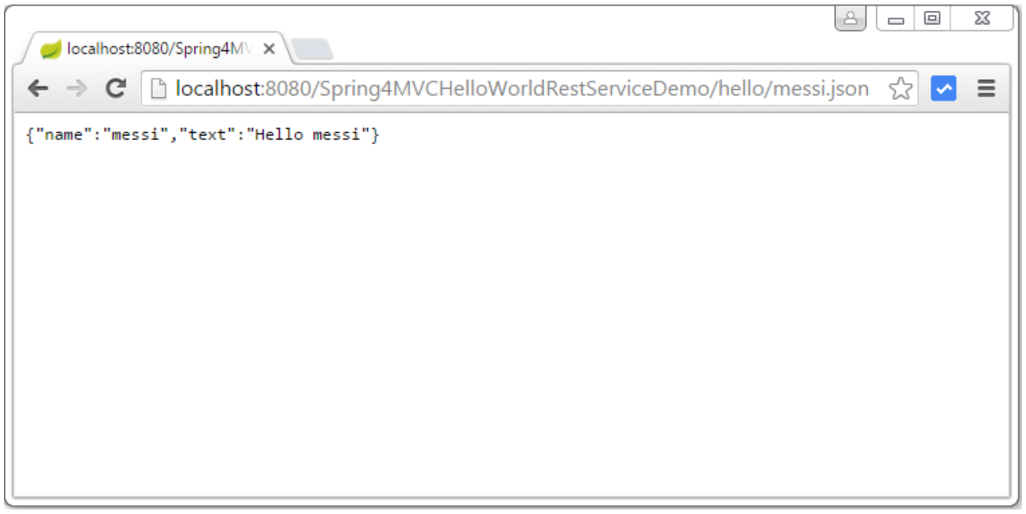
– remove the jackson-dataformat-xml.jar [comment it in pom.xml, build and deploy it again].
– Or Suffix the URL with .json

Had you redeployed the app with removing the dataformat dependency, you would have seen the following:


localhost:8080/Spring4M
localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi

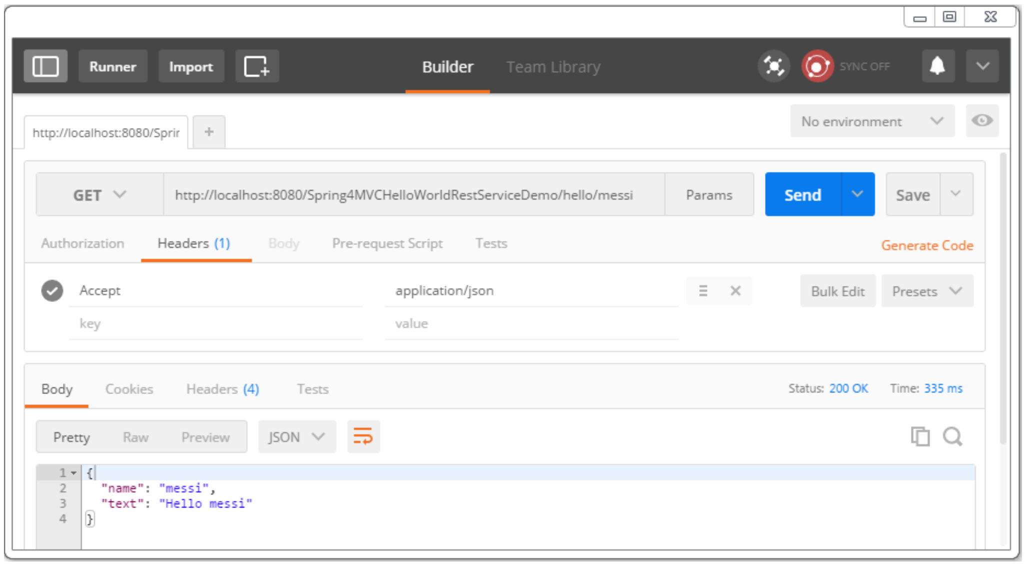
```
{"name":"messi","text":"Hello messi"}
```

Without redeployment, you can get the same result by suffixing url with format.


localhost:8080/Spring4M
localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi.json

```
{"name":"messi","text":"Hello messi"}
```

Using Postman:
Set the ‘Accept’ request header [to appliction/json or application/xml] and send the request. Check the response body:

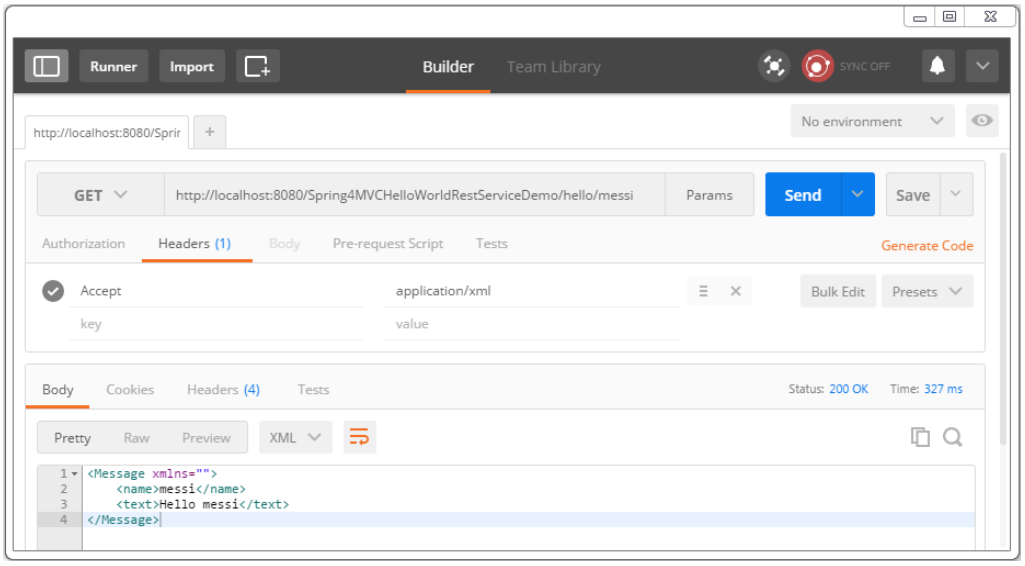


GET http://localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi

Accept: application/json

Status: 200 OK Time: 335 ms

```
{
  "name": "messi",
  "text": "Hello messi"
}
```



GET http://localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi

Accept: application/xml

Status: 200 OK Time: 327 ms

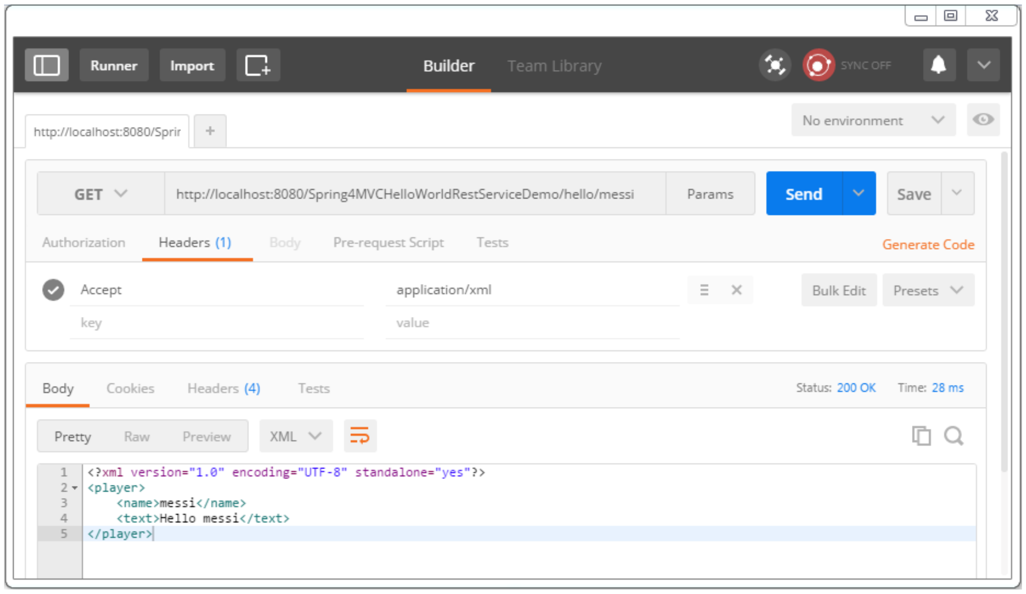
```
<Message xmlns="">
    <name>messi</name>
    <text>Hello messi</text>
</Message>
```

With JAXB

In case jackson-dataformat-xml.jar is not available, and you still want to get the XML response, just by adding JAXB annotations on model class (Message), we can enable XML output support. Below is the demonstration of same :

```
package com.websystique.springmvc.domain;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "player")
public class Message {

    String name;
    String text;

    public Message(){

    }

    public Message(String name, String text) {
        this.name = name;
        this.text = text;
    }
    @XmlElement
    public String getName() {
        return name;
    }

    @XmlElement
    public String getText() {
        return text;
    }

}
```

Remove dataformat dependency[jackson-dataformat-xml.jar], Compile, deploy and run it again, Send the request, you should see following response:

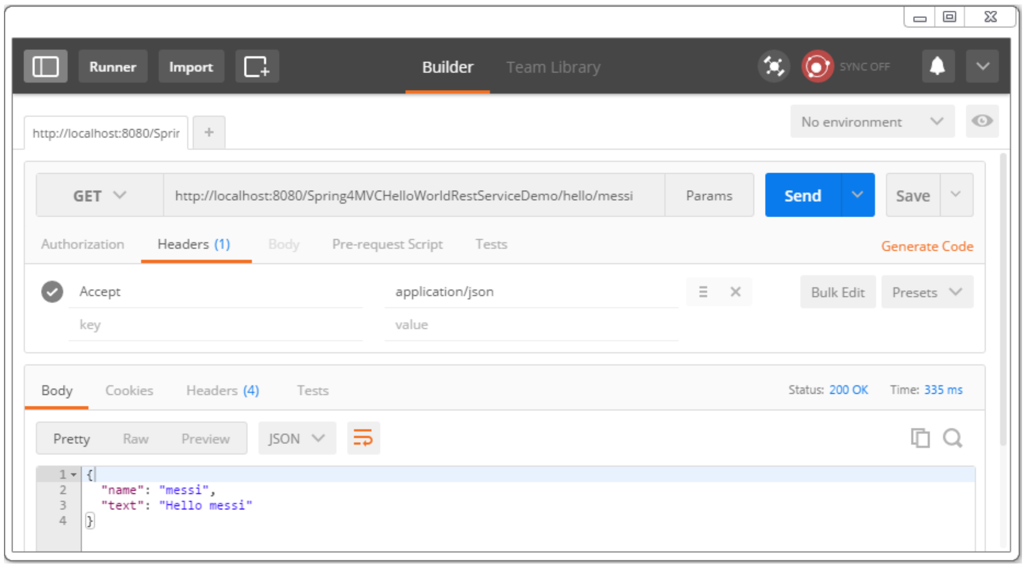


GET http://localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi

Accept application/xml

Status: 200 OK Time: 28 ms

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<player>
    <name>messi</name>
    <text>Hello messi</text>
</player>
```

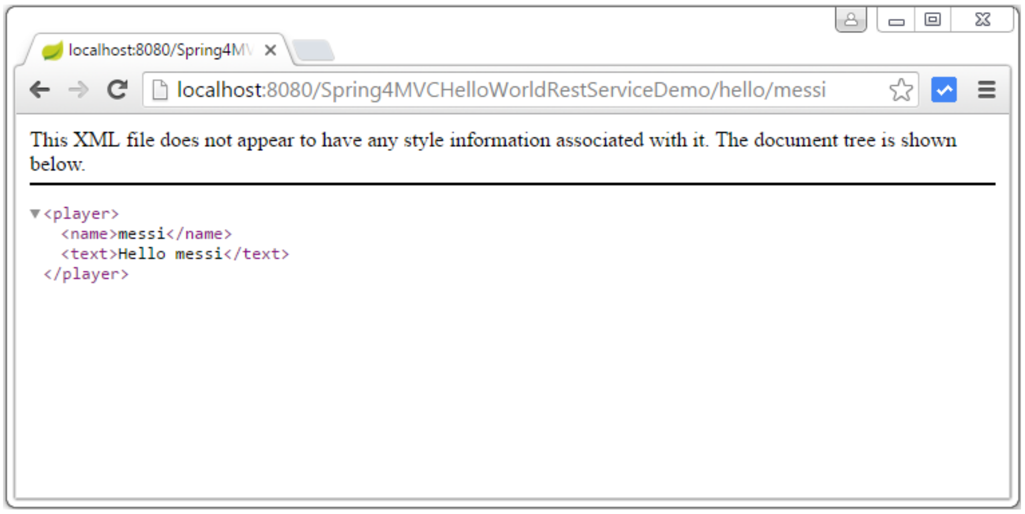


GET http://localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi

Accept application/json

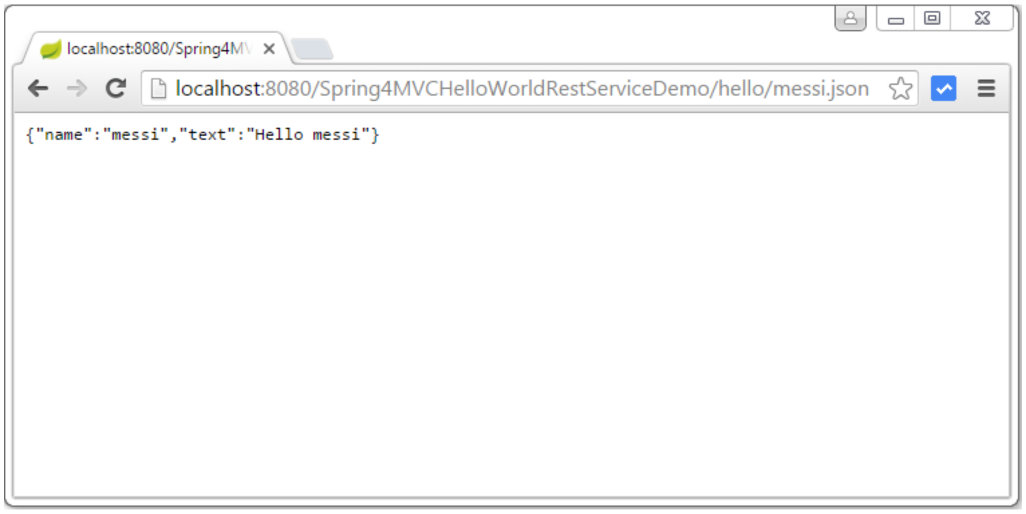
Status: 200 OK Time: 335 ms

```
{
  "name": "messi",
  "text": "Hello messi"
}
```


localhost:8080/Spring4M\
localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<player>
   <name>messi</name>
   <text>Hello messi</text>
 </player>
```


localhost:8080/Spring4M\
localhost:8080/Spring4MVCHelloWorldRestServiceDemo/hello/messi.json

```
{"name":"messi","text":"Hello messi"}
```

That's it. Check out Spring MVC 4 RESTFul Web Services CRUD Example+RestTemplate post for more advanced example.

Download Source Code

Download Now!

References

- Spring framework

websystiqueadmin

If you like tutorials on this site, why not take a step further and connect me on Facebook , Google Plus & Twitter as well? I would love to hear your thoughts on these articles, it will help me improve further our learning process.

If you appreciate the effort I have put in this learning site, help me improve the visibility of this site towards global audience by sharing and linking this site from within and beyond your network. You & your friends can always link my site from your site on www.websystique.com, and share the learning.

After all, we are here to learn together, aren't we?

Related Posts:

1. **Spring MVC @RequestBody @ResponseBody Example**
2. **Converting JSON to/from Java Maps using JACKSON API**
3. **Jackson Tree Model Example**
4. **Jackson Streaming Api Example**

springmvc. permalink.

← Spring 4 MVC HelloWorld Tutorial – Annotation/JavaConfig Example

Spring Batch- Read a CSV file and write to an XML file →

31 Comments **websystique** 1 **Login**

Recommend **Share** **Sort by Best**

Join the discussion…

Suraj Kadam • 4 months ago

Hi, whenever i run it it shows hello World message, but when i typed hello/messi it shows HTTP Status 404 - /Spring4MVCHelloWorldRestServiceDemo/hello/messi, i dont know what it made wrong , i am following same as u do. still getting this error, by the way your tutorials are very help full, thx for sharing.....

2 • Reply • Share ›

websystique Mod → Suraj Kadam • 2 months ago

Hi Suraj, Sorry I missed your message. In case you could not solve your issue, let me know.

• Reply • Share ›

Suraj Kadam → websystique • 2 months ago thx for replay, The issue has been solved.

• Reply • Share ›

Mack • 10 months ago

I built the project and was able to deploy and test the service on Tomcat. But using the same war file on Jboss 7.1.1 did not work. I wasn't able to hit the service the same way that I did using tomcat. Any idea why?

1 • Reply • Share ›

Balasaheb Karjule • 23 days ago

Thanks for this article.....

• Reply • Share ›

Dariel R. • 2 months ago

I just download the project, without made any modification I run it on Netbeans 8.1 + tomcat 8.0.33 and the browser always show HTTP 404 error.

• Reply • Share ›

websystique Mod → Dariel R. • 2 months ago

Hi Daniel, I don;t have NetBeans setup at my side, but looking at 404 i can tell you it may be related with tomcat setup in your IDE. Look at Setup Tomcat With Eclipse, although this is for Eclipse, you may get idea about how to fix it in your IDE.

• Reply • Share ›

Nico Grossi • 4 months ago

Hello, this is a really good tutorial, I have just 1 question, how I can set a default page like <welcome-file-list> tag, but in the Initialization class??

Thanks

• Reply • Share ›

websystique Mod → Nico Grossi • 2 months ago

Hi Nico, Sorry i missed your message. I am not aware of any official way to represent welcome-file-list with annotations, only the welcome page can be configured to return from controller method using default url mapping.

• Reply • Share ›

Paul C • 8 months ago

Thank you for sharing your knowledge. It's very helpful.
I had some sample Angular code in .jsp and it worked well. But if I change the file name to .html, I would get the HTTP 404 page. I think it must be the view-resolver issue but wasn't able to figure out the correct syntax. I would greatly appreciate if you could post the answers. Thanks.

• Reply • Share ›

websystique Mod → Paul C • 8 months ago

Hey Paul,

If i understood correctly, you are referring to templates and not only the data. In that case you need to configure view resolver.

I would suggest you to visit this & this post. First one shows AngularJS with JSP,and second one shows AngularJS with HTML templates. Note that in both cases backend is Spring based, template serving is done using regular controller while data (JSON e.g.) is served using pure REST API.

Any doubts, let me know.

1 • Reply • Share ›

Lee Namkyu • 9 months ago

Thanks for sharing this article~

• Reply • Share ›

websystique Mod → Lee Namkyu • 9 months ago

Glad you liked it. You might want to go through this post which is bit more realistic.

• Reply • Share ›

康大玮 • 9 months ago

thank you ! idea run well.

• Reply • Share ›

websystique Mod → 康大玮 • 9 months ago

Hi,
Great to see you liked it. You might want to go through this post which is bit more realistic.

• Reply • Share ›

cristian • 9 months ago

I was following you tutorial but when I run the application over Jboss the url localhost:8080/project/hello/something always shows Http get status 404 I'm using AbstractAnnotationConfigDispatcherServletInitializer, do you now why?

• Reply • Share ›

websystique Mod → cristian • 9 months ago

Hi Cristian, i have tested this example on multiple versions of tomcat without any issue. I can try on JBoss as well. Which version of JBOSS are you using? How are you deploying [From within Eclipse or using JBoss Admin console]?

• Reply • Share ›

cristian → websystique • 9 months ago

Thanks for your quickly answer, the JBoss version is 7.1, and I use eclipse to deploy the application.

• Reply • Share ›

• Reply • Share ›

websystique Mod ➦ cristian • 9 months ago

Cristian, as you mentioned using Eclipse to deploy your app, May i ask you if you have successfully deployed any other maven based project till now with your eclipse+JBoss setup? What i want to know is whether your Eclipse+Jboss setup itself is working correctly?

To help with that, I would recommend going through the post Setup Eclipse with Tomcat which details the steps for the setting up tomcat to deploy any project successfully. Although the steps here are for tomcat, somewhat similar setup should be applicable for JBoss. Can you try this please and let me know the findings? In the mean time, i will try to simulate your setup at my end.

• Reply • Share ›

cristian ➦ websystique • 9 months ago

Im trying to run with tomcat and I get the following error: WARNING: No mapping found for HTTP request with URI [/project/] in DispatcherServlet with name 'dispatcher', so where I must to setup the dispatcher?

• Reply • Share ›

websystique Mod ➦ cristian • 9 months ago

Hi Cristian, this is normal because in controller we have only one type mapping defined ['/hello/{something}']. It was sufficient to demonstrate this example [see the screenshots in post] but you can define additional mappings ['/' for example so that you can see something on home page].
Try :
http://localhost:8080/Spring4MVCHelloWorldRestSe

Note: Try to use other browser than Eclipse default one (window->Web browser->..). Sometimes the default browser used by Eclipse does not perform very well.

• Reply • Share ›

cristian ➦ websystique • 9 months ago

Thanks for you tutorial, to summary works over tomcat but not over jboss,maybe because I need and extra configurations.

• Reply • Share ›

Dilip Kumar 9 months ago

Dilip Kumar • 9 months ago

Simple and good example to learn REST on Spring

• Reply • Share ›

RAJ GOPAL • 9 months ago

Hello **@websystique** Can you help me with how should i maintain the session.

• Reply • Share ›

Mack • 10 months ago

By the way, thanks for this post.

• Reply • Share ›

Kishore Saraswathula • a year ago

Can you please help me understand DI concept in the above example? Is there a way to avoid use of "new"

• Reply • Share ›

websystique Mod → Kishore Saraswathula • a year ago

Hey Kishore,

The trivial example shown above is not really about DI concept. It's mainly focused on Spring REST capabilities to serve a resource to a client using default HttpMessageConverters.

And that line with 'new' is not really breaking DI concept. Client asked for a resource and server have prepared it on the fly [using client's input] and served it.

For the DI concept, please have a look on this post.

Let me know if i misunderstood your question.

• Reply • Share ›

Kishore Saraswathula → websystique • a year ago

Well done for the article. You answered my question also.

• Reply • Share ›

websystique Mod → Kishore Saraswathula • a year ago

Just rolled a post over Spring MVC 4 RESTFul Web Services CRUD Example+RestTemplate. This might help you understand Spring REST support in detail.

• Reply • Share ›

Chandra • a year ago

Very helpful - Thanks. One question though - this app works for 4.0.6 Spring release BUT Not for 4.1.6. Unable to find the info in Spring release notes. Do you know the reason?