Javarevisited
Blog about Java, Programming, Spring, Hibernate, Interview Questions, Books and Online Course Recommendations from Udemy, Pluarlsight etc

Home core java spring hibernate collections multithreading design patterns interview questions coding data structure OOP java 8 books About Me
Java Certifications JDBC jsp-servlet JSON SQL Linux Courses online resources jvm-internals REST Eclipse jQuery Java IO Java XML

MONDAY, JANUARY 29, 2018

7 Reasons to Use Spring to develop RESTful Web Services in Java

REST has now become a standard way to develop web services and when it comes to Java, there are many frameworks and library available e.g. JAX-RS, Restlet, Jersey, RESTEasy, Apache CFX etc, but I encourage Java developers to use Spring framework to develop RESTful web services. But, some of you might ask, *why use Spring Framework to develop RESTful web services in Java?* What is the advantage and why it's better than other frameworks and libraries available out there? Well, the most important reason I think to use Spring for developing RESTful web service is that you can use your Spring MVC experience to develop RESTful web services.

This is one of the biggest advantage i.e. leveraging your years of experience on Spring MVC to expose your application as REST APIs. Another reason is that *Spring has excellent support for developing RESTful web services.*

In last a couple of versions, starting from Spring version 3.0, it has provided a lot of enhancements to Spring MVC to provide first-class REST support. It has provided dedicated annotations e.g. `@RestController` and `@ResponseStatus` to make the development of RESTful resources even easier in Spring 4.0.

It's also not only help you to create RESTful web services but also provides classes to consume REST resources e.g. you can use `RestTemplate` class to consume RESTful resources.

There are many more utility classes and annotations which makes the development of RESTful web services in Spring easier and seamless and I'll share a couple of them in this article to prove my point that using Spring to develop RESTful Web service is the right decision.

How Spring Supports RESTful Web Services?

As I told you in the first paragraph that we can use Spring MVC to create and consume RESTful web services. Now, let's see those supports in a little bit more details so that you can make the best use of them and quickly develop the RESTful services you always wanted to.

1. In Spring MVC, a controller can handle requests for all `HTTP` methods, which is a backbone of RESTful web services. For example, you can handle `GET` method to perform read operation, POST method to create resources, PUT method to update resources, and `DELETE` method to remove resources from the server. From Spring 3.2 onwards, you can also handle `PATCH` requests. Btw, if you are not familiar with Spring MVC framework then Spring MVC For Beginners on Udemy is a good place to start.

2. In case of REST, the representation of data is very important and that's why Spring MVC allows you to bypass View-based rendering altogether by using the `@ResponseBody` annotation and various `HttpMessgeConverter` implementations.

By using this two you can directly send a response to client e.g. the resource clients wants and also in the format they want.

I'll write more about `@ResponseBody` annotations and `HttpMessageConverter` in this blog in

Follow by Email
Email address... Submit

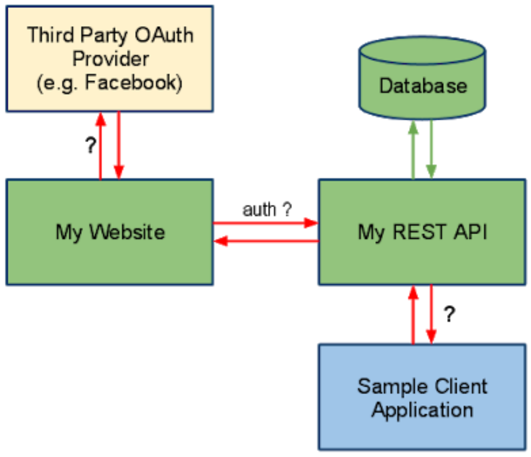
Interview Questions
core java interview question (168)
Coding Interview Question (72)
data structure and algorithm (68)
interview questions (46)
object oriented programming (31)
SQL Interview Questions (30)
design patterns (30)
thread interview questions (30)
collections interview questions (25)
spring interview questions (17)
database interview questions (16)
servlet interview questions (15)
Programming interview question (6)
hibernate interview questions (6)

Best of Javarevisited
How Spring MVC works internally?
How to design a vending machine in Java?
How HashMap works in Java?
Why String is Immutable in Java?
10 Articles Every Programmer Must Read
How to convert lambda expression to method reference in Java 8?
10 Tips to improve Programming Skill
10 OOP design principles programmer should know
How Synchronization works in Java?
10 tips to work fast in Linux
5 Books to improve Coding Skills

Java Tutorials
date and time tutorial (21)
FIX protocol tutorial (15)
Java Certification OCPJP SCJP (23)
java collection tutorial (73)
java IO tutorial (28)
Java JSON tutorial (12)
Java multithreading Tutorials (55)
Java Programming Tutorials (18)
Java xml tutorial (16)
JDBC (29)
jsp-servlet (37)
online resources (96)

Followers

coming articles, but if you can't wait, I suggest you go through **REST with Spring Certification class** by Eugen. He not only explains the basic details of developing RESTful web services but also advanced details like versioning and securing your REST APIs using Spring Security.


Third Party OAuth Provider (e.g. Facebook)
Database
?
My Website
auth ?
My REST API
?
Sample Client Application

3. The Spring 4.0 release added a dedicated annotation **`@RestController`** to make the development of RESTful web services even easier.

If you annotate your controller class using `@RestController` instead of `@Controller` then Spring applied message conversations to all handler methods in the controller.

This means you don't need to annotate each method with the **`@ResponseBody`** annotation. This also makes your code much cleaner. You can read more about it on my post difference between @Conroller and @RestController in Spring.

4. One of the main difference between REST web services and a normal web application is that REST pass resource identifier data in URI itself e.g. `/messages/101` while web application normally uses a query parameter e.g. `/messages?Id=101`.

If you remember, we use `@RequestParam` to get the value of those query parameter but not to worry, Spring MVC also provides a `@PathVariable` annotation which can extract data from URL. It allows the controller to handle requests for parameterized URLs.

You can learn more about **`@PathVariable`** in my post difference between @RequestParam and @PathVaraible in Spring.

5. Another key aspect of RESTful web services is Representation e.g. the same resource can be represented in different formats e.g. JSON, XML, HTML etc. Thankfully Spring provides several view implementations and views resolvers to render data as JSON, XML, and HTML.

For example, `ContentNegotiatingViewResolver` can look at the file extension of requests or Accept header to find out the correct representation of a resource for the client.

6. Similar to **`@ResponseBody`** annotation, which is used for converting the response to the format client wants (by using HttpMessageConverts), Spring MVC also provides **`@RequestBody`** annotation, which uses `HTtpMethodConverter` implementations to convert inbound HTTP data into Java objects passed into a controller's handler method.

You can further see **REST with Spring MasterClass** to learn more about `@RequestBody`

Categories

- SQL (53)
- courses (41)
- linux (36)
- database (30)
- Eclipse (28)
- Java Certification OCPJP SCJP (23)
- JVM Internals (22)
- JQuery (16)
- REST (16)
- Testing (11)
- Maven (10)
- general (10)

Blog Archive

- ▼ 2018 (131)
  - ► September (26)
  - ► August (21)
  - ► July (24)
  - ► June (28)
  - ► May (2)
  - ► April (6)
  - ► March (10)
  - ► February (9)
  - ▼ January (5)
    - 7 Reasons to Use Spring to develop RESTful Web Ser...
    - How to deal with ConcurrentModificationException i...
    - 10 Unit Testing and Integration Tools and Librarie...
    - 10 reasons to Learn Scala Programming Language
    - Top 20 Libraries and APIs Java Developer should kn...
- ► 2017 (831)
- ► 2016 (131)

annotation and how to effectively use it to develop RESTful web services in Java with Spring.

7. Spring Framework also provides a Template class, `RestTemplate`, similar to `JdbcTemplate`, and `JmsTemplate`, which can consume REST resources. You can use this class to test your RESTful web service or develop REST clients.

I have already talked about this class in my earlier blog posts and you can see this tutorial for a live example of using RestTemplate to consume JSON from a RESTful web service in Java.

These were some of the **important features of Spring MVC framework** which assist in developing RESTful web services. As I told the most important reason for me to choose Spring for developing RESTful resources is that I can use my existing knowledge of framework, which means no steep learning curve. If you look at from high level, developing RESTful services is not very different from developing a web application.

The fundamental difference is that in case of former, we mostly deal with human users where in case of REST, you have to deal with non-human users mostly rich JavaScript clients and mobile applications. This key difference then derives other differences e.g. representing data in JSON or XML instead of HTML which is suitable for human users but not for non-human systems.

If you are still not convinced, I suggest you go through **REST with Spring Certification class** on Baeldung, probably the best resource on Spring with REST at the moment, which will give you enough reason to stick with Spring for developing both RESTful resources and REST clients.

Other **Spring** and **REST Resources** you may like
How to build Microservices with Spring Cloud?
Microservices With Spring Boot and Spring Cloud
Difference between Restlet, Jersey, and RESTEasy in Java?
What is the use of DispatcherServlet in Spring MVC?
How to enable Spring security in a Java web application?
Spring in Action by Craig Walls
Spring Framework 5: Beginner to Guru

Thanks for reading this article, if you like these reasons for developing RESTful web services using Spring then please share with your friends and colleagues. If you have any questions or feedback, then please drop a note.

Javarevisited **3 Ways to convert String to Boolean in Java? Examples**

You can convert a String object to Boolean object or boolean primitive by using the Boolean.valueOf() and Boolean.parseBoolean() method. ...

References

1. Oracle's Java Tech Network
2. jQuery Documentation
3. Microsoft SQL Server Documentation
4. Java SE 8 API Documentation
5. Spring Documentation
6. Oracle's JAva Certification
7. Spring Security 5 Documentation

Pages

Privacy Policy