What’s the difference between @PathVariable, @PathParam, and @RequestParam?

Even though @PathVariable and @RequestParam are both used to extract values from the URL, their usage is largely determined by how a site is designed.

The @PathVariable annotation is used for data passed in the URI (e.g. RESTful web services) while @RequestParam is used to extract the data found in query parameters.

If we had chosen to use query parameters, our URL would be:

http://localhost:8080/orders?id=100

This would be implemented in a controller method like the following:

@GetMapping("/orders")

@ResponseBody

public String getOrder(@RequestParam(value = "id", required = true) String id) {

  return "Order ID: " + id;

}

These annotations can be mixed together inside the same controller.

@PathParam is a JAX-RS annotation that is equivalent to @PathVariable in Spring.

Missing URI template variable 'X' for method parameter of type Y

Spring will throw this exception if it cannot match up the template variable with the name of our path variable.

Let's take a look at an example causes this error:

@GetMapping("/orders/{id}")

public String getOrder(@PathVariable String orderId) {}

Under the hood, Spring looks at the name of our path variable orderId and tries to match it up with a template variable.  It cannot because our template variable is defined as id.

We can fix this by changing the path variable name to id.

Alternatively, @PathVariable accepts a value attribute:

@GetMapping("/orders/{id}")

public String getOrder(@PathVariable("id") String orderId) {}

# [Differences between @RequestParam and @PathVariable annotations in Spring MVC?](https://javarevisited.blogspot.com/2017/10/differences-between-requestparam-and-pathvariable-annotations-spring-mvc.html)

The Spring MVC framework, one of the most popular frameworks for developing a web application in Java world also provides several useful annotations to extract data from the incoming request and mapping the request to the controller, e.g., @RequestMapping, @RequestParam, and @PathVariable. Even though both @RequestParam and @PathVariable is used to extract values from the HTTP request, there is a subtle difference between them, which makes them a useful question from an interview and [spring certification](http://javarevisited.blogspot.sg/2017/06/2-books-to-prepare-for-spring-certification-exam.html#axzz4pbqSY8Ua) point of view. We'll examine the subtle difference between @RequestParam and @PathVaraible in this article. As the name suggests, @RequestParam is used to get the request parameters from URL, also known as query parameters, while @PathVariable extracts values from URI.  
  
For example, if the incoming HTTP request to retrieve a book on topic "Java" is http://localhost:8080/shop/order/1001/receipts?date=12-05-2017, then you can use the @RequestParam annotation to retrieve the query parameter date and you can use @PathVariable to extract the orderId i.e. "1001" as shown below:  
  
@RequestMapping(value="/order/{orderId}/receipts", method = RequestMethod.GET)  
public List listUsersInvoices(                               @PathVariable("orderId") int order,  
 @RequestParam(value = "date", required = false) Date dateOrNull) {  
...  
}  
  
The required=false denotes that the query parameter can be optional, but the [URL](http://www.java67.com/2013/01/difference-between-url-uri-and-urn.html) must have the same URI.

## How to extract Query Parameters in Spring MVC using @RequestParam

Spring MVC is a rich framework to develop both web applications and [RESTful web services](http://javarevisited.blogspot.sg/2017/02/how-to-consume-json-from-restful-web-services-Spring-RESTTemplate-Example.html) in Java. It provides several ways to retrieve data from the incoming HTTP requests e.g.

* Request Parameters
* Path Variables
* Form inputs

Now that you know the difference between @RequestParam and @PathVariable in Spring MVC let's see how to write Spring controllers that can handle request parameters and path variables.

### Using @RequestParam to get Query parameters

In a Spring MVC application, you can use the @RequestParam annotation to accept query parameters in Controller's handler methods.  
  
For examples, suppose you have a web application which returns details of orders and trades, and you have the following URLs:  
  
http://localhost:8080/eportal/orders?id=1001  
  
To accept the query parameters in the above URLs, you can use the following code in the Spring MVC controller:  
  
@RequestMapping("/orders")  
public String showOrderDetails(@RequestParam("id") String orderId, Model model){  
   model.addAttribute("orderId", orderId);  
   return "orderDetails";  
}  
  
  
If the name of the query parameter is same as the name of the variable in handler's @RequestParam annotated argument then you can simply use @RequestParam without specifying the name of a query parameter, Spring will automatically derive the value (see [Introduction to Spring MVC](https://pluralsight.pxf.io/c/1193463/424552/7490?u=https%3A%2F%2Fwww.pluralsight.com%2Fcourses%2Fspring-mvc4-introduction)).

Also, here is the code to prove the point:  
  
URL: http://localhost:8080/eportal/trades?tradeId=2001  
  
@RequestMapping("/trades")  
public String showTradeDetails(@RequestParam String tradeId,  
                               Model model){  
  model.addAttribute("tradeId", tradeId);  
  return "tradeDetails";  
}  
  
You can see that we have just annotated the method parameter tradeId with @RequestParam without specifying the name of the query parameter because the name of both request parameter and argument name is same, i.e., "tradeId."  
  
If you are interested in learning more about the @RequestParam annotations and other useful Spring MVC annotations, I suggest you check out [Spring Framework 5: Beginner to Guru](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-framework-5-beginner-to-guru%2F), one of the valuable resource for learning Spring MVC 5.

### Using @PathVariable annotation to extract values from URI

You can use Spring MVC's @PathVaraible annotation to extract any value which is embedded in the URL itself. Spring call it a URI template, where @PathVariable is used to obtain some placeholders from the URI itself.  
  
If you have worked in RESTful Web services, you might know that the [REST URIs](http://www.java67.com/2016/09/when-to-use-put-or-post-in-restful-web-services.html) contains values, e.g. a REST API to retrieve a book using ISBN number looks like the following:  
  
URL: http://localhost:8080/book/9783827319333  
  
Now, to extract the value of ISBN number from the URI in your Spring MVC Controller's handler method, you can use @PathVariable annotation as shown in following code:  
  
@RequestMapping(value="/book/{ISBN}", method= RequestMethod.GET)  
public String showBookDetails(@PathVariable("ISBN") String id,  
                              Model model){  
   model.addAttribute("ISBN", id);  
   return "bookDetails";  
}  
  
Similar to @RequestParameter annotation, you also can also omit the value attribute in @PathVariable annotation, if the name of path variable's placeholder in the @RequestMapping annotation is same to the variable name in the handler method's @PathVariable annotated parameter (see [REST with Spring](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa)).  
  
For example, you can rewrite above code as shown below:  
  
@RequestMapping(value="/book/{ISBN}", method= RequestMethod.GET)  
public String showBookDetails(@PathVariable String ISBN,   
                              Model model){  
   model.addAttribute("ISBN", ISBN);  
   return "bookDetails";  
}  
  
  
Spring MVC provides several useful annotations to map and extract data from the HTTP request, and as of Spring developer, you should be aware of these, e.g. @RequestMapping, @RequestParam, and @PathVariable.  
  
These concepts and annotations are fundamental from both Spring MVC interview prospect as well as Spring certifications. You will always find a couple of questions based on these concepts.  
  
Btw, if you are preparing for Spring Web Certifications, you can also check out [David Mayer's free Spring Mock Questions](https://www.certification-questions.com/spring-pdf-download/spring-core-v4.2-pdf.html?affiliateCode=fcff36fd-557a-4713-abf6-973e9924770f&utm_source=Javin&utm_medium=affiliate&utm_campaign=affiliate) to get an idea about the level of questions and format of questions you can expect on real Spring Web Certification.

## Difference between @PathVariable and @RequestParam in Spring

Now that we understand both the difference as well as how to use both @RequestParam and @PathVariable in Spring MVC application, let's revise the critical difference between them from an interview perspective.  
  
1) The @RequestParam is used to extract query parameters while @PathVariable is used to extract data right from the URI.  
  
2) @RequestParam is more useful on a traditional web application where data is mostly passed in the query abatements while @PathVariable is more suitable for RESTful web services where URL contains values, e.g. http://localhost:8080/book/9783827319333, here data, which is ISBN number is part of URI.  
  
If you want to learn more about how to effectively develop REST applications using Spring, I suggest you join Eugen Paraschiv's [**REST with Spring**](http://courses.baeldung.com/p/rest-with-spring-the-master-class?affcode=22136_bkwjs9xa) course.  
  
It's an online course which will teach you the nitty-gritty of developing both REST API and services for real-world scenarios.  
  
Eugen has extensive experienced in developing RESTful web services using Spring, and this course is a great way to tap into his knowledge.  
  
3) @RequestParam annotation can specify default values if a query parameter is not present or empty by using a defaultValue attribute, provided the required attribute is false.  
  
4) Spring MVC allows you to use multiple @PathVariable annotations in the same method, provided, no more than one argument has the same pattern.  
  
  
That's all about the **difference between @PathVariable and @RequestParam in Spring MVC**. Even though both are used to extract data from URL, @RequestParam is used to retrieve query parameters, anything after ? in the URL, while @PathVariable is used to retrieve values from URI itself.  This concept is essential for both traditional web application development as well as developing RESTful Web Services using Spring, so you must spend some time to understand it better.  
  
Other **Spring MVC articles** you may like to explore  
[Spring Framework 5: Beginner to Guru](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-framework-5-beginner-to-guru%2F)  
[Spring MasterClass from Beginner to Expert](https://click.linksynergy.com/fs-bin/click?id=JVFxdTr9V80&subid=0&offerid=323058.1&type=10&tmpid=14538&RD_PARM1=https%3A%2F%2Fwww.udemy.com%2Fspring-tutorial-for-beginners%2F)  
[Difference between @RestController and @Controller in Spring MVC?](http://javarevisited.blogspot.sg/2017/08/difference-between-restcontroller-and-controller-annotations-spring-mvc-rest.html)  
[How Spring MVC works internally?](http://javarevisited.blogspot.sg/2017/06/how-spring-mvc-framework-works-web-flow.html)  
[What is the use of DispatcherServlet in Spring MVC?](http://www.java67.com/2017/06/what-is-use-of-dispatcherservlet-in-spring-mvc.html)  
[How to enable Spring security in a Java web application?](http://javarevisited.blogspot.sg/2017/05/how-to-enable-spring-security-in-java-web-application.html)  
[Spring in Action by Craig Walls](https://www.amazon.com/Spring-Action-Covers-4/dp/161729120X?tag=javamysqlanta-20)  
  
Thanks for reading this article so far. If you like this article, then please share it with your friends and colleagues. If you have any questions, please drop me a note.

Read more: <https://javarevisited.blogspot.com/2017/10/differences-between-requestparam-and-pathvariable-annotations-spring-mvc.html#ixzz5yCuKu2li>

@Param : By default, Spring Data JPA uses position-based parameter binding, as we saw in our previous tutorials. We can also use named parameter with @Param annotation to give a method parameter a concrete name and bind the name in the query. That makes refactoring of code easier in case we have to add/remove additional parameters.

@Param works with both @Query and @NamedQuery.

HttpPooling

The @PathVariable annotation is used for data passed in the URI (e.g. RESTful web services) while @RequestParam is used to extract the data found in query parameters.

**@XmlAccessorType**

@XmlAccessorType sets default field and property serializability. By default, JAXB serializes public fields and properties. By setting @XmlAccessorType, the bean can choose to only allow annotated fields to be serialized.

@XmlAccessorType works with the other annotations and [@XmlTransient](http://www.caucho.com/resin-3.1/doc/jaxb-annotations.xtp#@XmlTransient) to serialize fields and properties. @XmlTransient prevents serialization, overriding the @XmlAccessorType. The presense of any other annotation will force serialization, overriding the @XmlAccessorType.

|  |  |
| --- | --- |
| @XmlAccessorType values | |
| NONE | Only annotated fields and properites should be serialized> |
| FIELD | All fields (public or private) should be serialized> |
| PROPERTY | All properties (public or private) should be serialized> |
| PUBLIC\_MEMBER | Public fields and properties> |