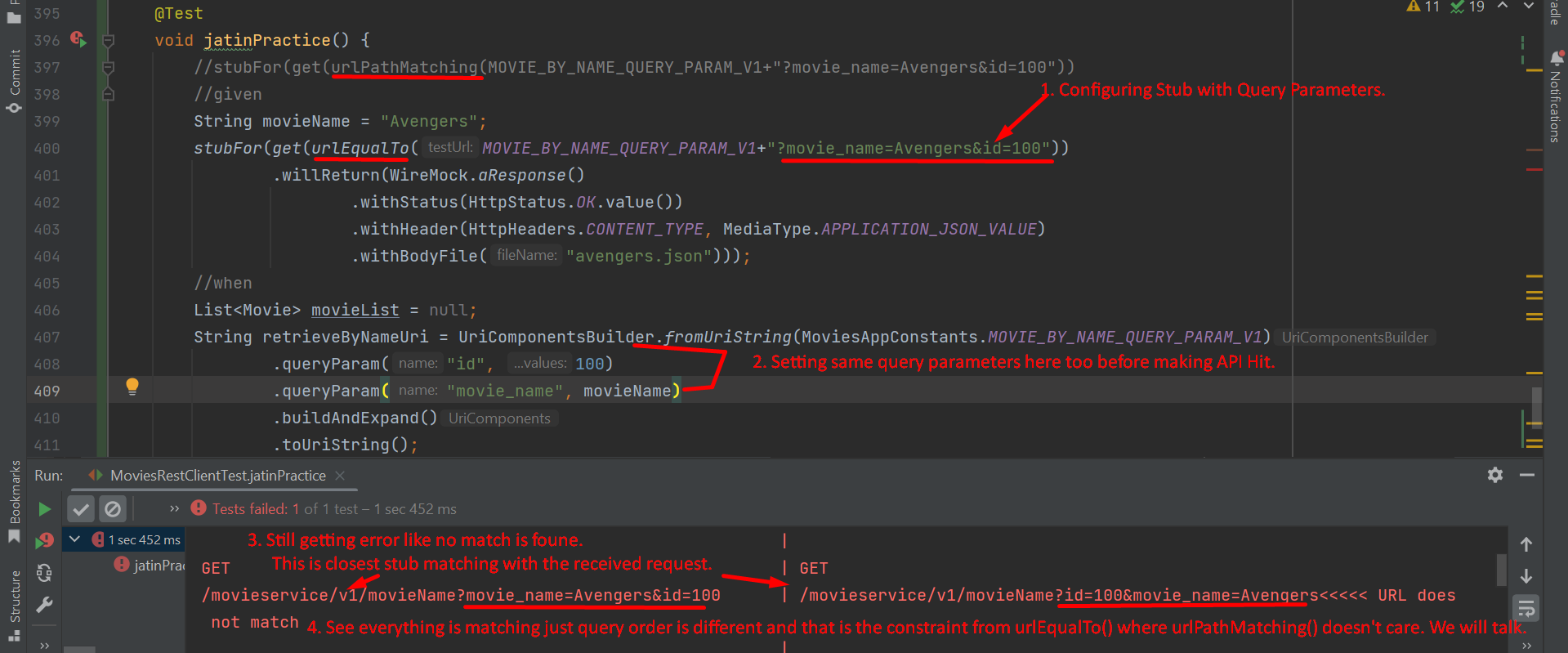
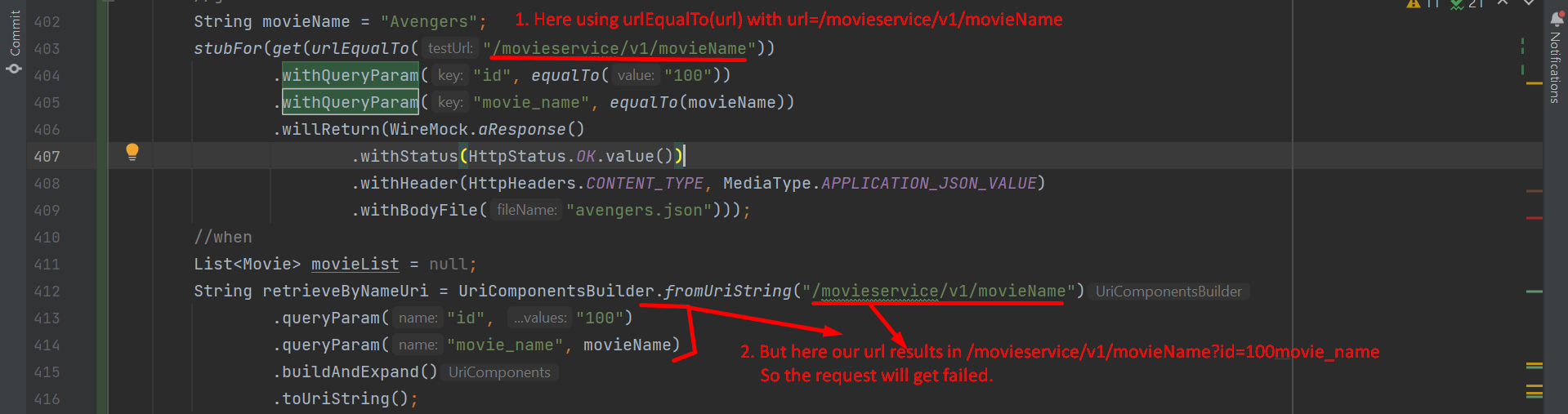
1. **Agenda:**
   1. How to map a stub with a request having query parameter in the URL.
2. 
3. Let’s correct the query order.  
   
4.   
   Above, it will try to match the url exactly we have defined in Stub then query parameters.   
   So, if you remove query parameters from request then URL will match but query parameters will be missing.   
   So better use **urlPathMatching**() it will match first only path then query parameters without query parameters’ order.   
   We can also use **urlPathMatching**() when we have dynamic url path as it provides regexp in the url like   
   /movieservice/v1/movieName/100, /movieservice/v1/movieName/200, /movieservice/v1/movieName/300 etc. where 100, 200, 300 are id. Then   
   using **urlPathMatching(“**/movieservice/v1/movieName/[0-9]**+”)**

Here is a **clear tabular comparison** of **urlPathMatching()**, **urlPathEqualTo()**, and **urlEqualTo()**, along with their **use cases** and **examples** to help you decide when and how to use each one effectively.

## **Comparison Table**

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature** | urlEqualTo() | urlPathEqualTo() | urlPathMatching() |
| **Matching Scope** | Matches the **entire URL**, including **path** and **query parameters (values + order)**. | Matches only the **path** and **ignores query parameters**. | Matches the **path** using **regular expressions (regex)**, ignores query params. |
| **Query Parameter Handling** | Requires **exact query parameters** in **order** and **values**. | Query parameters are **validated separately** using **withQueryParam()**. | Query parameters are **validated separately** using **withQueryParam()**. |
| **Order of Query Parameters** | **Order matters**—query parameters must be in **exact order** as defined. | **Order doesn’t matter**—parameters are validated **individually**. | **Order doesn’t matter**—parameters are validated **individually**. |
| **Extra Query Parameters** | **Fails** if there are **extra query parameters** not defined in the stub. | **Ignored** if extra query parameters are present. | **Ignored** if extra query parameters are present. |
| **Path Matching** | Matches **exact path** (case-sensitive). | Matches **exact path** (case-sensitive). | Matches **path with regex**—allows dynamic segments like /users/[0-9]+. |
| **Dynamic Paths (Variables)** | **Not supported**—requires exact paths. | **Not supported**—requires exact paths. | **Supported**—handles dynamic paths like /users/{id} using regex. |
| **Use Cases** | Use for **strict validation** where query parameters and their order **must match exactly**. | Use for APIs where query parameters **may vary in order** or where **extra parameters** are allowed. | Use for **dynamic or variable paths** (e.g., /users/{id}) with **pattern matching**. |
| **Example Stub** | urlEqualTo("/movieservice/v1/movieName?id=100&movie\_name=Avengers") | urlPathEqualTo("/movieservice/v1/movieName") | urlPathMatching("/movieservice/v1/movieName/.\*") |
| **Matching Request** | GET /movieservice/v1/movieName?id=100&movie\_name=Avengers | GET /movieservice/v1/movieName?id=100&movie\_name=Avengers or ?movie\_name=Avengers&id=100 | GET /movieservice/v1/movieName/101 or /movieservice/v1/movieName/Avengers |
| **Non-Matching Request** | Fails if query params are **unordered** or if **extra params** are added. | Passes even if query params are **unordered** or **extra params** are added. | Fails if the **path doesn’t match the regex pattern** defined. |
| **Flexible Handling** | ❌ Rigid—does not tolerate minor differences. | ✅ Flexible—handles unordered or extra query params. | ✅ Highly Flexible—handles dynamic paths with regex. |

## **When to Use Each Matcher?**

|  |  |  |
| --- | --- | --- |
| **Matcher** | **When to Use It?** | **Example Use Case** |
| urlEqualTo() | - Use when the **entire URL** (path + query parameters) **must match exactly** (e.g., query parameters and order). | **Strict Validation APIs** (e.g., payment gateways, authentication tokens, or query-sensitive APIs). |
| urlPathEqualTo() | - Use when query parameters are **dynamic** or **unordered** but the **path must match exactly**. | **Search APIs** (e.g., /movies?genre=action&page=1&size=10) where query parameters can be **unordered or dynamic**. |
| urlPathMatching() | - Use when **dynamic paths** or **pattern matching** is required (e.g., path variables or RESTful APIs). | **RESTful APIs** (e.g., /users/{id} or /orders/{orderId} with variable segments). |

## **Examples for Each Matcher**

### **1.** urlEqualTo() **– Exact Matching Example**

stubFor(get(urlEqualTo("/movieservice/v1/movieName?id=100&movie\_name=Avengers"))

.willReturn(aResponse()

.withStatus(200)

.withBody("{\"message\": \"Movie Found!\"}")));

**Matches:**

GET /movieservice/v1/movieName?id=100&movie\_name=Avengers

**Fails:**

GET /movieservice/v1/movieName?movie\_name=Avengers&id=100 // Order mismatch

GET /movieservice/v1/movieName?id=100 // Missing param

### **2.** urlPathEqualTo() **– Flexible Query Parameters Example**

stubFor(get(urlPathEqualTo("/movieservice/v1/movieName"))

.withQueryParam("id", equalTo("100"))

.withQueryParam("movie\_name", equalTo("Avengers"))

.willReturn(aResponse()

.withStatus(200)

.withBody("{\"message\": \"Movie Found!\"}")));

**Matches:**

GET /movieservice/v1/movieName?id=100&movie\_name=Avengers

GET /movieservice/v1/movieName?movie\_name=Avengers&id=100 // Query order doesn't matter

GET /movieservice/v1/movieName?id=100&movie\_name=Avengers&type=action // Extra params ignored

### **3.** urlPathMatching() **– Dynamic Path Example**

stubFor(get(urlPathMatching("/movieservice/v1/movieName/.\*")) // Regex for dynamic path matching

.withQueryParam("id", equalTo("100"))

.willReturn(aResponse()

.withStatus(200)

.withBody("{\"message\": \"Movie Found!\"}")));

**Matches:**

GET /movieservice/v1/movieName/101?id=100

GET /movieservice/v1/movieName/Avengers?id=100

**Fails:**

GET /movieservice/v1/movieSearch/101?id=100 // Path mismatch

## **Key Takeaways:**

1. **Use urlEqualTo()** for **strict URL validation**—both path and query params must match **exactly** (including order).
2. **Use urlPathEqualTo()** when query parameters are **flexible**—allowing unordered or extra params.
3. **Use urlPathMatching()** for **dynamic paths with regex**—ideal for RESTful APIs with variable path segments.

Let me know if you'd like me to elaborate further on any of these scenarios! 😊