# ✅ **How Data is Sent from UI to Controller in Spring**

In a Spring Web MVC application, when a **user performs an action on the UI (browser)** — like clicking a link, submitting a form, or entering something in the URL — that data is sent to the **Controller** on the server.

The Controller is like a **Java class that listens for requests** and processes them.

There are **3 main ways** to send data from the UI (browser) to the Controller:

## 📌 3 Common Methods to Send Data to Controller

| **Method** | **Description** | **When to Use** |
| --- | --- | --- |
| **1. Query Parameter** | Data is passed in the URL after a ?, in key-value format | For search, filters, small inputs |
| **2. Path Parameter** | Data is passed directly inside the URL path | When data is part of the resource |
| **3. Form Submission** | Data is sent through an HTML form using GET or POST methods | For submitting user inputs (e.g., login, registration) |

## 🔹 1. Query Parameter (Detailed)

### ✅ What is it?

A **query parameter** is a small piece of data that is sent in the **URL itself**, after a ? symbol.

It works like this:

http://localhost:9090/service?name=Najafi

Here:

* service is the endpoint
* ?name=Najafi is the query parameter
  + name is the key
  + Najafi is the value

You can also pass **multiple** parameters like this:

/service?name=Najafi&age=25

### 🧾 Controller Code Example:

@GetMapping("/service")

public String greet(@RequestParam String name) {

return name + ", Welcome to iNeuron .ai";

}

* @GetMapping("/service"): listens for requests at /service.
* @RequestParam String name: tells Spring to extract the name value from the URL.

### 🎯 Sample Inputs & Outputs:

**URL:**

http://localhost:9090/service?name=pavankalyan

**Output:**

pavankalyan, Welcome to iNeuron .ai

**URL:**

http://localhost:9090/service?name=pavan

**Output:**

pavan, Welcome to iNeuron .ai

**🔹 How to Send Multiple Query Parameters?**

**🧾 Syntax Format:**

?key1=value1&key2=value2&key3=value3

The & (ampersand) symbol is used to **separate each key-value pair**.

**✅ Example:**

http://localhost:9090/service?id=1&name=Najafi

Here:

* id is a query parameter with value 1
* name is another query parameter with value Najafi

**🔹 How to Receive Multiple Query Parameters in Controller?**

In Spring, we use the @RequestParam annotation for each parameter **individually** to receive its value.

**🧾 Example Controller Code:**

@GetMapping("/service")

public String getDetails(@RequestParam int id, @RequestParam String name) {

return "ID: " + id + ", Name: " + name;

}

**🎯 Sample Request and Output:**

**Request URL:**

http://localhost:9090/service?id=1&name=Najafi

**Output:**

ID: 1, Name: Najafi

**🔍 Notes on Using @RequestParam for Multiple Parameters:**

| **Point** | **Explanation** |
| --- | --- |
| ✅ **One annotation per parameter** | Each query parameter must have its own @RequestParam |
| ✅ **Type-safe** | Spring will convert the value to the required data type (like int, String) |
| ❗ **Missing parameter causes error** | If the parameter is not optional, and it's not present, Spring throws an error |
| ✅ **Make parameters optional** | Use required = false and/or defaultValue to avoid errors |

**🧪 Optional Parameters Example:**

@GetMapping("/service")

public String getDetails(

@RequestParam(required = false, defaultValue = "0") int id,

@RequestParam(required = false, defaultValue = "Guest") String name

) {

return "ID: " + id + ", Name: " + name;

}

**Request URL:**

http://localhost:9090/service

**Output:**

ID: 0, Name: Guest

**🧠 Summary:**

* You can pass **multiple query parameters** by separating them with & in the URL.
* Use @RequestParam for each key in the Controller.
* Handle optional values using required = false and defaultValue.

**✅ Real-Life Use Case:**

**URL:**

http://localhost:9090/search?category=books&page=2&limit=10

**Controller:**

@GetMapping("/search")

public String search(

@RequestParam String category,

@RequestParam int page,

@RequestParam int limit

) {

return "Category: " + category + ", Page: " + page + ", Limit: " + limit;

}

### 🧠 Notes:

* Query parameters are **publicly visible** in the browser URL.
* Use only for **small, safe, non-sensitive** data.
* You can make the parameter optional using:

@RequestParam(required = false, defaultValue = "Guest")

**✅ @RequestParam(required = false, defaultValue = "Guest")**

This annotation is used to **safely handle query parameters** when:

* The user **might not send the parameter** in the request.
* You want to set a **default value** if the parameter is missing.

**🔹 Problem Without It**

Let’s say you write:

@GetMapping("/service")

public String greet(@RequestParam String name) {

return name + ", Welcome to iNeuron .ai";

}

If someone opens this URL:

http://localhost:9090/service

➡️ It will throw an **error** like:

400 Bad Request: Required request parameter 'name' is not present

Because Spring is **expecting** the name parameter but it’s **missing**.

**🔹 Solution: required = false**

@RequestParam(required = false)

This tells Spring:

“It’s okay if the parameter is not sent. Don’t throw an error.”

But if the user doesn't send anything, the value will be null.

Example:

@GetMapping("/service")

public String greet(@RequestParam(required = false) String name) {

return name + ", Welcome to iNeuron .ai";

}

➡️ If the user skips the name:

http://localhost:9090/service

Output:

null, Welcome to iNeuron .ai

So it’s safe from errors, but **null is not a friendly response**.

**🔹 Solution: defaultValue = "Guest"**

To avoid showing null, we add a default value:

@RequestParam(required = false, defaultValue = "Guest")

Now:

* If the user sends ?name=Hyder, it uses Hyder
* If the user **skips name**, it uses "Guest"

**✅ Final Example:**

@GetMapping("/service")

public String greet(@RequestParam(required = false, defaultValue = "Guest") String name) {

return name + ", Welcome to iNeuron .ai";

}

| **URL** | **Output** |
| --- | --- |
| /service?name=Najafi | Najafi, Welcome to iNeuron .ai |
| /service | Guest, Welcome to iNeuron .ai |

**🧠 Summary:**

| **Property** | **Purpose** |
| --- | --- |
| required=false | Tells Spring not to throw error if missing |
| defaultValue | Provides a fallback value when input is null |

**🔍 Your Sir’s Version:**

@RequestParam("firstname") String fname

**✅ What it means:**

* It tells Spring:

“Take the value of the query parameter named firstname and put it into the variable fname.”

* This is **explicit mapping** between the **request parameter name** and the **Java variable name**.

**✅ When to use:**

* When your **Java variable name is different** from the parameter name in the URL.

**🌐 Example:**

**Request URL:**

http://localhost:9090/hello?firstname=Najafi

**Controller:**

@GetMapping("/hello")

public String greet(@RequestParam("firstname") String fname) {

return "Hello, " + fname;

}

**Output:**

Hello, Najafi

**🆚 Compared with:**

@RequestParam(required = false, defaultValue = "0") int id

**✅ What this means:**

* This version is more **robust**. It tells Spring:

“This id parameter is optional. If it’s not given, use the default value 0.”

* You are not changing the name; here, id is the same in the URL and the variable.

**🌐 Example:**

**Controller:**

@GetMapping("/service")

public String getId(@RequestParam(required = false, defaultValue = "0") int id) {

return "ID: " + id;

}

**Request 1:**

http://localhost:9090/service?id=5

**Output:**

ID: 5

**Request 2 (no parameter):**

http://localhost:9090/service

**Output:**

ID: 0

**✅ Side-by-side Comparison**

| **Aspect** | **Your Sir’s Version** | **required = false, defaultValue Version** |
| --- | --- | --- |
| **Purpose** | Map custom query param names | Handle optional/missing parameters |
| **Allows Renaming?** | ✅ Yes (firstname → fname) | ❌ No (name must match variable) |
| **Handles Missing Values?** | ❌ No, will throw error if missing | ✅ Yes, uses default |
| **Example Use** | @RequestParam("firstname") String fname | @RequestParam(required = false, defaultValue = "0") int id |

**💡 Can We Combine Both?**

Yes!

@RequestParam(value = "firstname", required = false, defaultValue = "Guest") String fname

* Maps firstname to fname
* Makes it optional
* Uses "Guest" if not provided

**🧠 Final Takeaway:**

* Use @RequestParam("paramName") to **rename or match differently named parameters**.
* Use required = false and defaultValue = "..." to **make parameters optional and safe**.
* Combine them if needed for **flexibility**.

Eg: SpringWebMvcUiToControllerDataTransfer