

## Formularios y parámetros con Spring MVC

Como hemos visto al utilizar Servlets, para poder obtener los parámetros que vienen de un formulario, utilizamos la clase **HttpServletRequest**, el cual es definido como parámetro, ya que sea en el método **doGet** o **doPost**.

Con Spring también podemos utilizar dichos objetos para obtener los parámetros de un formulario, definiéndolos siempre en el método controlador.

Vamos a crear un formulario en la pagina "index.html" en la carpeta **clases/clase10**, el cual quedará de la siguiente manera (desde la línea 62):

```
<h1 class="mt-4">Formulario con HttpServletRequest</h1>
<div class="card mb-4">
  <div class="card-body">
    <p class="mb-0">Ingrese sus credenciales:</p><br/>
    <form method="post" th:action="@{/parametros1}">
      <div class="form-group"><label class="small mb-1" for="usuario">Usuario</label>
      <input class="form-control py-4" id="usuario" type="text" placeholder="Ingrese Usuario" style="width: 400px" name="usuario" /></div>
      <div class="form-group"><label class="small mb-1" for="password">Contrase&ntilde;a</label>
      <input class="form-control py-4" id="password" type="password" placeholder="Ingrese Contraseña" style="width: 400px" name="password" /></div>
      <div class="form-group">
        <input type="submit" class="btn btn-primary" value="Iniciar Sesi&oacute;n"></div>
      </form>
    </div>
  </div>
</div>
```

Como vemos, hemos definido un formulario con el tag **form**, sin embargo, recordemos que estamos utilizando Thymeleaf como gestor de nuestros archivos HTML, por lo que la propiedad **action** del formulario estará de la siguiente manera:

```
<form method="post" th:action="@{/parametros1}">
```

De ahora en adelante, para definir la URL que se ejecutará al enviar el formulario, utilizaremos el tag **th:action** cuyo valor será la URL formada de la forma **@{/url}**, la @ se utiliza para indicar el contexto de la aplicación, mientras que la URL a ejecutar se encontrará definida dentro de las llaves { }.

Los inputs del formulario son definidos tal cual se hace en HTML normal (no hay sintaxis especial de Thymeleaf), en este caso se les ha agregado estilo con las clases css definidas en el atributo **class**, así como un ID, y lo mas importante, la propiedad **name**, la cual se utiliza para identificar este input con el parámetro correspondiente.

Entonces, si levantamos el proyecto e ingresamos a la URL <http://localhost:8080/index10> veremos lo siguiente:



Ahora tenemos que crear el controlador que servirá para procesar la petición que se generará al enviar el formulario.

Para ello, ingresamos a la clase **Clase10Controller** y vemos el siguiente método:

```
24 @RequestMapping("/parametros1")
25 public ModelAndView parametros1(HttpServletRequest request) {
26     ModelAndView mav = new ModelAndView();
27     String usuario = request.getParameter("usuario");
28     String password = request.getParameter("password");
29
30     mav.addObject("user", usuario);
31     mav.addObject("pass", password);
32     mav.setViewName("clases/clase10/resultado");
33     return mav;
34 }
35 }
```

Como vemos, dicho método está asociado a la URL **/parametros1**, la cual es la URL definida en nuestro atributo **action** del formulario anterior.

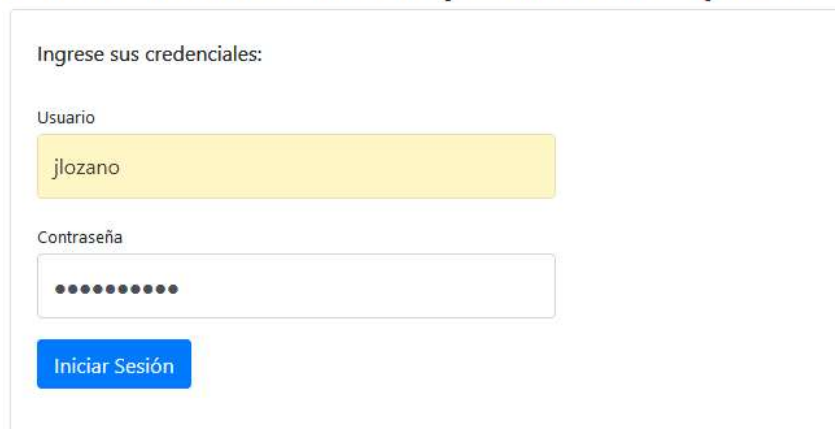
El parámetro que recibe es, precisamente, el objeto **HttpServletRequest**, el cual es provisto automáticamente por el contenedor (Tomcat). Lo que este método hará es obtener los parámetros enviados por el formulario, guardarlos en dos variables, y luego crear un objeto **ModelAndView**, al cual le añadiremos esos dos objetos (usuario y password) al modelo con el método **addObject** (líneas 31 y 31) y luego los enviaremos a la página que se encuentra en la ruta **clases/clase10/resultado**.

Dicha página mostrará el contenido de los parámetros enviados.

Ejemplo:

1. Ingreso los valores en el formulario

## Formulario con HttpServletRequest



Ingrese sus credenciales:

Usuario

jlozano

Contraseña

.....

Iniciar Sesión

2. Dar clic al botón "Iniciar Sesión" (que no es que inicie sesión, simplemente se envía el formulario al controlador.

- Luego, nos mandará a la página **resultado.html** que mostrará los parámetros enviados.



Ahora bien, Spring nos provee de una forma de recibir los parámetros de un formulario como parámetros de un método normal, para esto utilizaremos la anotación **@RequestParam**.

Dicha anotación afectará a los parámetros del método que queremos utilizar para recibir del formulario.

Entonces, para el ejemplo anterior, como son dos parámetros los que se envían, tendremos que definir dos parámetros en el método del controlador (en este caso de tipo String) de la siguiente manera:

```
@RequestMapping("/parametros2")
public ModelAndView parametros2(String usuario, String password) {
    ModelAndView mav = new ModelAndView();
    mav.addObject("user", usuario);
    mav.addObject("pass", password);
    mav.setViewName("clases/clase10/resultado");
    return mav;
}
```

Ahora, estos dos parámetros serán utilizados para recibir los dos parámetros del formulario. Sin embargo, nos falta decirle a Spring que queremos que los utilice para ese fin. Para esto, debemos anteponer la anotación **@RequestParam** a cada parámetro, quedando de la siguiente manera:

```
@RequestMapping("/parametros2")
public ModelAndView parametros2(@RequestParam String usuario, @RequestParam String password) {
    ModelAndView mav = new ModelAndView();
    mav.addObject("user", usuario);
    mav.addObject("pass", password);
    mav.setViewName("clases/clase10/resultado");
    return mav;
}
```

Con esto, Spring sabrá que queremos que dichos parámetros serán asociados a los parámetros **usuario** y **password** del formulario.

---

*En este caso, los nombres de los parámetros deben ser iguales a los de la propiedad **name** del input al que queremos hacer referencia, si, por ejemplo, ponemos en el parámetro del método **String user** (en vez de **String usuario**) y el input se llama **usuario**, Spring lanzará una excepción mencionando que se espera que venga un parámetro de nombre **user***

---

Entonces, al ingresar los valores al formulario:

## Formulario con @RequestParam

Ingrese sus credenciales:

Usuario

Contraseña

Iniciar Sesión

Y dar clic al botón “Iniciar Sesión” (submit), veremos el mismo resultado que el anterior:

Spring MVC / Thymeleaf

[Dashboard](#)[Código Clases](#)

### Parámetros enviados

Usuario: **juan86**  
Contraseña: **juan123**

### Propiedad “value”

Ahora bien, si quisiéramos nombrar la variable de otra manera que no sea el del **input** entonces para eso utilizaremos la propiedad **value** de la anotación **@RequestParam** de la siguiente manera:

```
@RequestMapping("/parametros3")
public ModelAndView parametros3(@RequestParam(value="user") String usuario,
    @RequestParam(value="pass") String password) {
    ModelAndView mav = new ModelAndView();
    mav.addObject("user", usuario);
    mav.addObject("pass", password);
    mav.setViewName("clases/clase10/resultado");
    return mav;
}
```

Para estos casos, utilizamos la propiedad **value** decirle a Spring que queremos que a la variable **usuario** le asigne el valor que venga del input con nombre **user**. De igual forma, para el segundo parámetro, a la variable **password** queremos que le asigne el valor del input con nombre **pass**.

De esta manera evitamos tener que utilizar el mismo nombre del input al de la variable.

## Propiedad "Required"

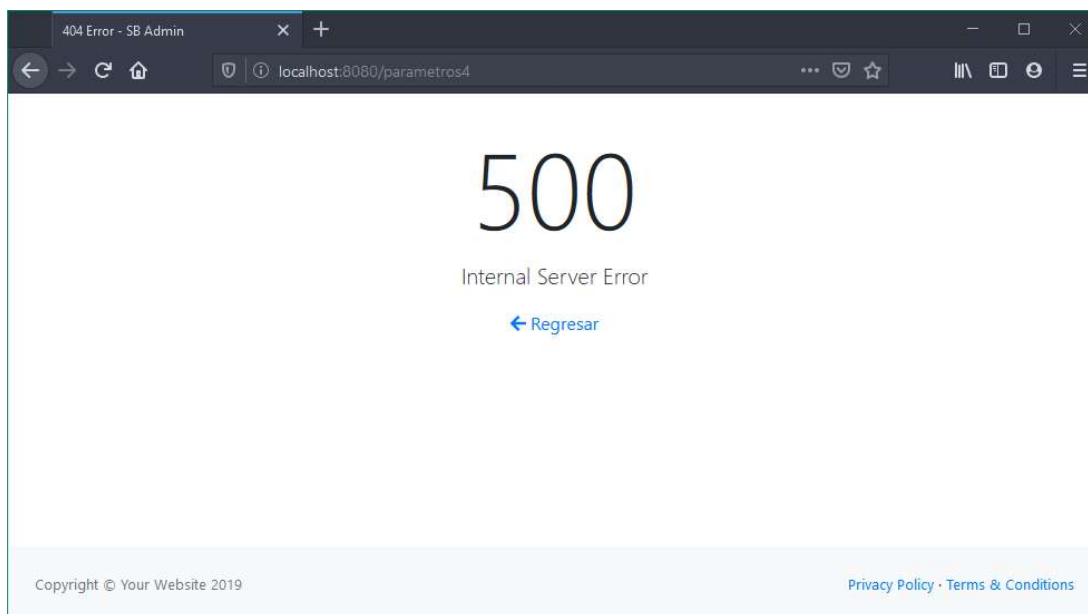
Cuando definimos los parámetros en el método del controlador, es obligación que estos vengan en la petición que se hace en el formulario, si esto no es así (ya sea que falte alguno), Spring lanzará una excepción mencionando dicho problema.

Para evitar este tipo de situaciones, podemos utilizar otra propiedad de la anotación `@RequestParam`, la cual es **required**, esta recibe un booleano **true** o **false**, si queremos que dicho parámetro sea mandatorio o no.

En el siguiente ejemplo:

```
46 @RequestMapping("/parametros3")
47 public ModelAndView parametros3(@RequestParam(value="user") String usuario,
48     @RequestParam(value="pass") String password,
49     @RequestParam(value="rol", required=false) String rol) {
50     ModelAndView mav = new ModelAndView();
51     mav.addObject("user", usuario);
52     mav.addObject("pass", password);
53     mav.setViewName("clases/clase10/resultado");
54     return mav;
55 }
```

Si vemos la línea 49, hemos definido otro parámetro de nombre `rol`, al cual le hemos puesto la propiedad **required=false**, esto quiere decir que al ejecutar el submit del formulario, como solo viajarán los parámetros **user** y **password**, dicho parámetro **rol** vendrá **null**. Si le quitáramos la propiedad **required=false**, al hacer el submit veríamos la siguiente página:



Esto quiere decir que ha ocurrido un error en el servidor, el cual es la excepción que lanza Spring al no encontrar algún parámetro que él espera (por estar definido en el método controlador). Si revisamos la consola, efectivamente vemos lo siguiente:

Resolved [\[org.springframework.web.bind.MissingServletRequestParameterException: Required String parameter 'rol' is not present\]](#)