



Initial page

Cache

week-9



week-7



week-8



Forms, post and redirect

Thymeleaf

week-10



Forms, post and redirect



Go here for online version: <https://behu.gitbook.io/kea/week-8/forms-post-redirect>

Why are we even talking about forms and post?

Sending data to a server is essential for interacting with a website user. Create a new user, booking online flight tickets, ordering a product online.

Learning goals

- HTML forms
- @PostMapping
- forward , redirect
- PRG pattern

HTML Forms

HTML forms is used for sending data to the server. it comes from physical forms like these:



PHYSICAL EXAMINATION CLEARANCE FORM

This form must be on file in the school before practicing with any athletic team

Student Name: _____ Birth Date: _____ Age: _____ Gender: M / F
Address: _____
Home Telephone: _____ - _____ - _____
School: _____ Grade: _____ Sports: _____

I certify that the above student has been medically evaluated and is deemed to be physically fit to: (Check One Box)

☐ (1) Participate in all school interscholastic activities without restrictions.

☐ (2) Not cleared for: ☐ All Sports ☐ Specific Sports _____

Cross out specific sports below not cleared for participation.

Sport classification based on contact:

Collision Contact Sports		Limited Contact Sports		Non-contact Sports	
Basketball	Ice Hockey	Baseball	Alpine Skiing	Bowling	Track Running
Boys Lacrosse	Soccer	Competitive Cheer	Girls Softball	Cross Country	Track Field Events
Diving	Wrestling	Girls Lacrosse	Pole Vault	Golf	Discus
Football		Girls Gymnastics	Girls Volleyball	Swimming	Shot Put
				Tennis	

Sport classification based on intensity and strenuousness:

High Intensity High-to-Moderate Dynamic High-to-Moderate Static		High Intensity High-to-Moderate Dynamic Low Static		High Intensity Low Dynamic High-to-Moderate Static	Low Intensity Low Dynamic Low Static
Alpine Skiing	Track Events - Distance	Baseball	Swimming	Girls Competitive	Bowling
Cross Country	Track Events - Sprint	Lacrosse (Boys and Girls)	Tennis	Cheer	Golf
Football	Wrestling	Soccer	Girls Volleyball	Diving	
Ice Hockey		Girls Softball		Field Events	
				Girls Gymnastics	

☐ (3) Requires further evaluation before a final recommendation can be made.

Additional recommendations for the school or parents: _____

I have examined the above named student and completed the preparticipation physical evaluation. The athlete does not present apparent clinical contraindications to practice and participate in the sport(s) as outlined above. A copy of the physical exam is on record in my office and can be made available to the school at the request of the parents. If conditions arise after the athlete has been cleared for participation, the provider may rescind the clearance until the problem is resolved and the potential consequences are completely explained to the athlete (and parents/guardians).

Examiner Signature: _____ Date of Exam: _____

Print Examiner Name: _____

Address: _____

Office Telephone: _____ - _____ - _____

COPY BOTH SIDES OF THIS SHEET FOR THE STUDENT TO RETURN TO THE SCHOOL AND KEEP THE ENTIRE FORM IN THE STUDENTS MEDICAL RECORD

EMERGENCY INFORMATION FOR: _____ Grade: _____

Allergies - Drug Reactions - Current Medications: _____

Other Special Medical Information: _____

Emergency Contact: _____ Relationship: _____

Telephone: (H) _____ - _____ - _____ (W) _____ - _____ - _____ (C) _____ - _____ - _____

Personal Physician _____ Office Telephone _____ - _____ - _____

PHYSICAL EXAMINATION CLEARANCE FORM This form must be on file in the school before practicing with...

Creating a form

Here is an example of a form

```
1  <form action="/sign-up" method="POST">
2    <label for="name">Write your name</label>
3    <input type="text" name="name" id="name" />
4
5    <label for="mobile">Write your mobile</label>
6    <input type="tel" name="mobile" id="mobile" />
7
8    <label for="formal-name">Use formal name</label>
9    <input type="checkbox" name="formal-name" id="formal-name"/>
10
11   <label for="gender">Gender</label>
12   <input type="radio" name="gender" id="gender" value="Female"/>
13   <input type="radio" name="gender" value="Male" />
14   <input type="radio" name="gender" value="Other" />
15
16   <label for="vehicle">Select your favorite car brand</label>
17   <select name="vehicle" id="vehicle">
18     <option value="volvo">Volvo</option>
19     <option value="fiat">Fiat</option>
20     <option value="bmw">Bmw</option>
21   </select>
22
23   <label for="description">Describe yourself</label>
24   <textarea name="description" cols="30" rows="3" id="description"></textarea>
25
26   <label for="driver-license">Image of driver license</label>
27   <input type="file" name="driver-license" id="driver-license"/>
28   <button type="submit">Submit</button>
29 </form>
```

There are a few things going on. Lets dissect it:

`action="https://telmore.dk"` - The `action` attribute decides what url the form data should be send to.

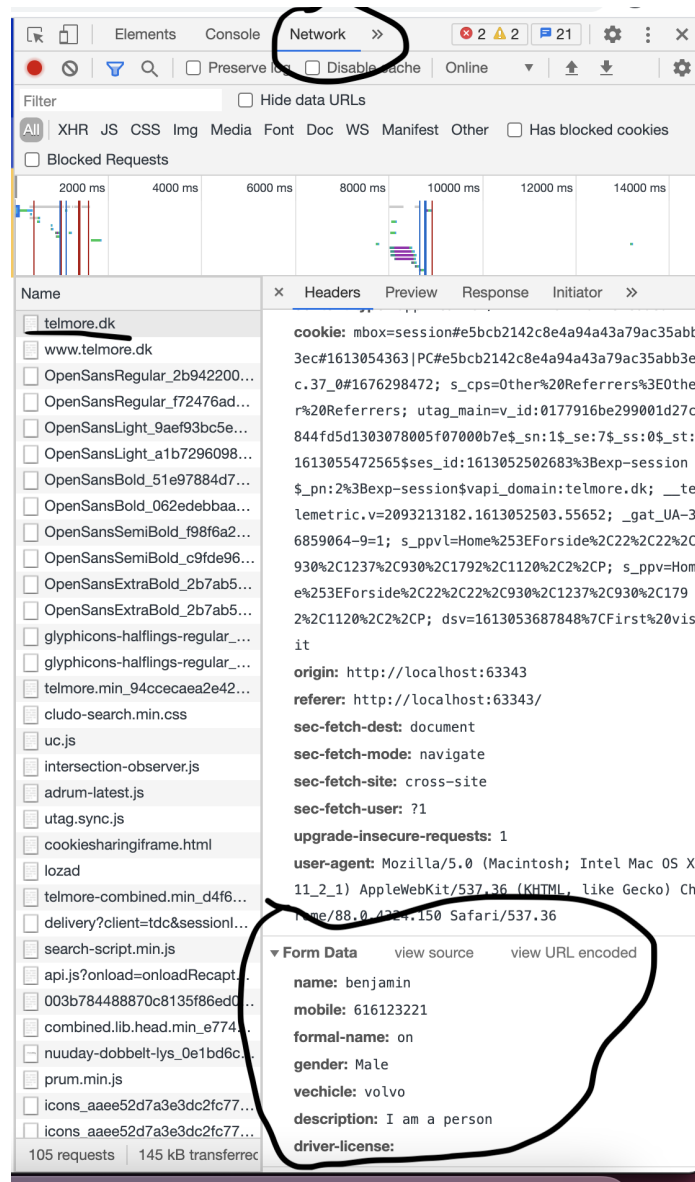
`method="GET"` - The `method` attribute decides what kind of request to make. When posting we will mostly be using a `POST` request. Here is quickly about the two most important request types (there are a lot more but that's not relevant for now)

- **GET request** - Getting information. Fx get all the information we have on the user with id 1. Or simply get the html at the `/about` url.
- **POST request** - Creating new information. Fx creating a new user, making a new order, creating new facebook post.

`<label for="gender">Gender</label>` This is a label that is connected to some field (`input` , `textarea` or `select`). It helps the user figuring out what to put into the connected field. The connection between `label` and field happens with the `for` attribute and the `id` on the field.

`type="text"` - `input` fields can have a type. There are quite a lot of **types**. it can help the user and also do a bit of validation on the frontend. So fx if you specify `type="number"` then the number keyboard will come up on the users mobile.

`name="description"` - When we send the data to a server, then name decides the key of that field. See below. Here is the `POST` request



Screenshot 2021-02-11 at 15.29.05

`button type="submit"` - When the button is clicked submit the form.

If you want to continue your learning

- Form validation

PostMapping

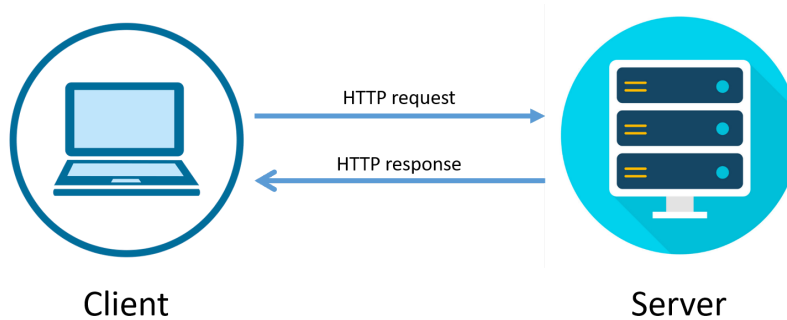


Image for post

Now we have figured out how to send the `POST` request (with data) to the server using forms. Now we need to figure out how to get that data in our `@controller`

```
1 @PostMapping(value = "/sign-up")
2 @ResponseBody
3 public String createNewUser(@RequestParam("name") String name, @RequestParam("mobile")
4     return "User created with name: " + name + " and mobile: " + mobile;
5 }
```

using the `@PostMapping` notation we can use it just like the `@GetMapping` specifying a `value` that will be the endpoint.

To get data out of the `POST` request use `@RequestParam("name") String name`. `@RequestParam` specifies the key you are looking for. Remember that the `name` attribute on the field decided the key!

Forward and redirect

Some times we are interested in making the user go to another website than the one he put in the url or was directed to. For this we use forwards and redirects

Redirect

There are two ways of doing a redirect in spring boot.

RedirectView

Use the `RedirectView` class. To add query parameters use the `RedirectAttributes` class as a parameter to the redirect method.

```
1 @GetMapping("redirect-test")
2 public RedirectView redirectView(RedirectAttributes attributes) {
3     // adding query parameters to the redirected page
4     attributes.addAttribute("name", "Charlotte");
5     return new RedirectView("/sign-up");
6 }
```

Below is how the redirect will work behind the scenes. What does the 302 mean?

Screenshot 2021-02-12 at 13.50.10

So the redirect says : "Hey browser i have actually moved this url by sending the `302` response code".

Now the browser asks: "Sound good server, but where have you moved the url to???".

The server responds: "Just look at the `response header` called `Location`. Thats where the url has been moved to!".

The browser now loads the new url found under the `Location` header!

Disadvantages

1. we're now coupled to the Spring API because we're using the `RedirectView` directly in our code.
2. We now need to know from the start, when implementing that controller – that the result will always be a redirect – which may not always be the case. Maybe we have a check. Fx

```
1 if(user.loggedIn()) {
2     return "dashboard"
3 } else {
4     // Redirect to /sign-in
5 }
```

In this example we could not use the `RedirectView` because we return different things based on an `if` sentence

Prefix

The result is **exactly** the same as above! Server sends a `302` with the `Location` header set. But we are not dependent on `RedirectView` !

```
1 // Redirect with prefix redirect
2 @GetMapping("redirect-prefix-test-simple")
3 public String redirectViewPrefixSimple() {
4     // adding query parameters to the redirected page
5     return new String("redirect:/sign-up");
6 }
```

Using query parameters

```
1 // Redirect with prefix redirect
2 @GetMapping("redirect-prefix-test")
3 public ModelAndView redirectViewprefix(ModelMap model) {
4     // adding query parameters to the redirected page
5     model.addAttribute("name", "Louise");
6     return new ModelAndView("redirect:/sign-up", model);
7 }
```

Forward

So far we have used `302` to redirect a page

Now let's try and do a redirect with the server. First a simple version

```
1 // Redirect using forward simple
2 @GetMapping("redirect-forward-test-simple")
3 public String redirectForwardSimple() {
4     // adding query parameters to the redirected page
5     return new String("forward:/sign-up");
6 }
```

Adding query parameters to the forward

```
1 // Redirect using forward
2 @GetMapping("redirect-forward-test")
3 public ModelAndView redirectForward(ModelMap model) {
4     // adding query parameters to the redirected page
5     model.addAttribute("name", "Charlotte");
6     return new ModelAndView("forward:/sign-up", model);
7 }
```

Now keep attention to the url! It does not change and only one request happens. Basically Spring boot just serves the `/sign-up` view and nothing else

Post, redirect, get pattern

Imagine a user submits a form and reloads the page. Now that form request will be sent twice. Resulting in two database instances.

With this new pattern a server receives a request, saves the data (`createProduct`) and then redirects the user to a confirmation page using `GET` not `POST` (`createProductPageSuccess`)

```
1 @Controller
2 public class PostRedirectGet {
3     @GetMapping("create-product")
4     public String createProductPage() {
5         return "create-new-product";
6     }
7
8     @PostMapping("create-product")
9     public String createProduct(@RequestParam String title, @RequestParam int price, RequestAttributes attributes) {
10         attributes.addAttribute("title", title);
```

```
11         attributes.addAttribute("price", price);
12
13         return "redirect:/create-product-success";
14     }
15
16     @GetMapping("create-product-success")
17     @ResponseBody
18     public String createProductPageSuccess(@RequestParam String title, @RequestParam int price) {
19         return "Created product: " + title + " " + price;
20     }
21 }
```

Notice how the `POST` parameters are sent to the `create-product-success` using `RedirectAttributes`.

Exercise time

We would like to create a new social media!

Therefore create a website where users can create a new social media post and see a list of all posts that were created. Use this repo as a starter for your project: <https://github.com/behu-kea/new-social-media>

The site should have these url's:

Url	Description
<code>/dashboard</code>	Return the json for all the public social media posts (Thursday we will render these posts using html templates). In the starter example there is an example of how to return json from a list.
<code>/submit</code>	Is where a user can create a new social media post using a form. You need to create the html <code>form</code> . In the starter there is an example of how to return an html template for a specific route.
	Return the json for the Social media post that was just created!
<code>/success</code>	Thursday we will be showing that the social media post was successfully created. Maybe you want to add the post information. Fx this is the post that you created: title: "I love sunshine", Description... Should contain a link to go to <code>/dashboard</code>

This is what a post should include

- Title
- Content
- Date
- Public/private
- Something that you come up with!

To give this new social media a bit of edge, add something to the social media post.

Maybe it's a site for dog lovers, so you add Dog name to the post


Maybe its a Dice lovers so you add their favorite dice number from 1-6

I would love to see a bit of creativity here :)

Remember to structure your application properly with

- Controllers
- Models
- Services
- Repositories

Focus on creating the post, sending the post and doing the redirects. Thursday we will focus on creating the html templates and rendering the different pages.

 Powered by GitBook

Last updated 2 minutes ago