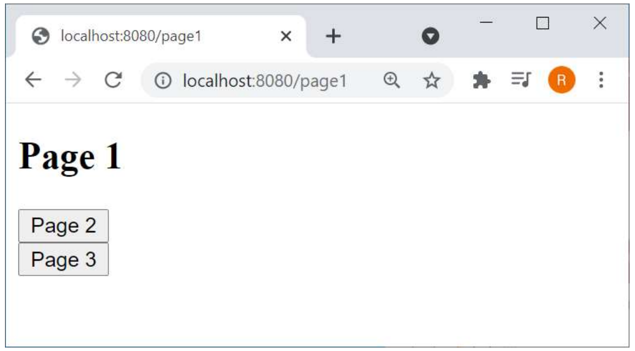
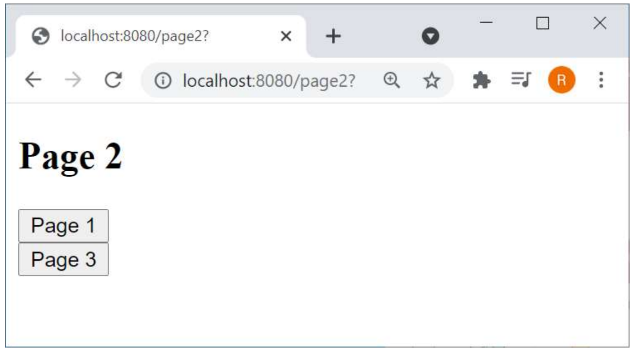
LESSON 2 SPRING MVC

NAVIGATION

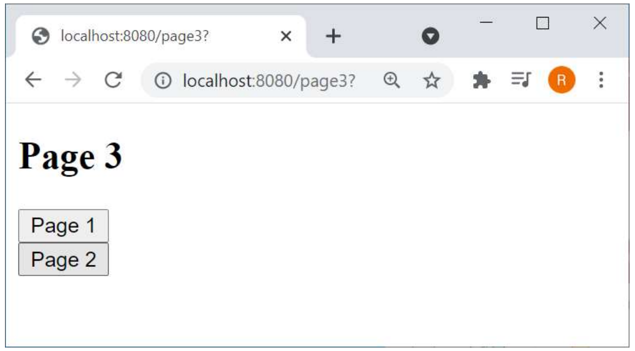
Navigation


localhost:8080/page1

Page 1


localhost:8080/page2?

Page 2


localhost:8080/page3?

Page 3

Navigation

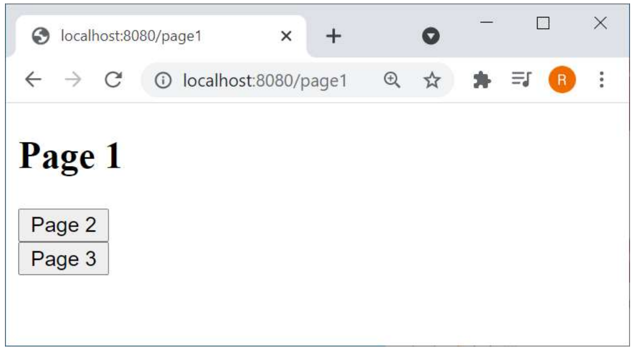
```html
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h2>Page 1</h2>
<form action="page2" method="get">
   <input type="submit" value="Page 2" />
</form>
<form action="page3" method="get">
   <input type="submit" value="Page 3" />
</form>
</body>
</html>
```

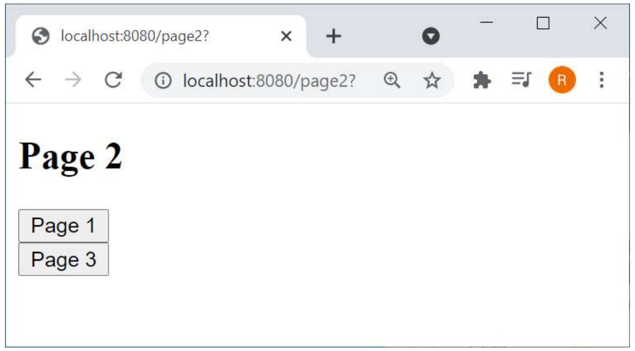
```html
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h2>Page 2</h2>
<form action="page1" method="get">
   <input type="submit" value="Page 1" />
</form>
<form action="page3" method="get">
   <input type="submit" value="Page 3" />
</form>
</body>
</html>
```

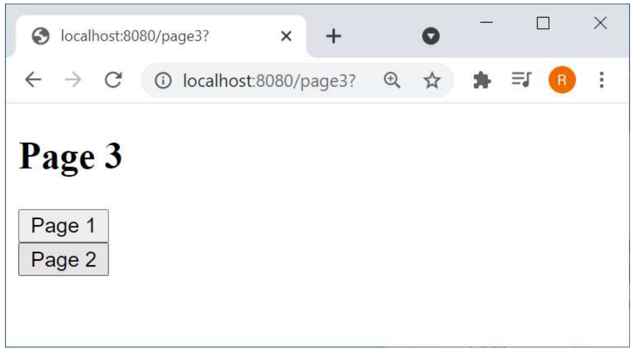
```html
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h2>Page 3</h2>
<form action="page1" method="get">
   <input type="submit" value="Page 1" />
</form>
<form action="page2" method="get">
   <input type="submit" value="Page 2" />
</form>
</body>
</html>
```

The controller

```
@Controller
public class NavigationController {

   @GetMapping("/page1")
   public ModelAndView page1() {
      Map<String, Object> params = new HashMap<>();
      return new ModelAndView("page1", params);
   }
   @GetMapping("/page2")
   public ModelAndView page2() {
      Map<String, Object> params = new HashMap<>();
      return new ModelAndView("page2", params);
   }
   @GetMapping("/page3")
   public ModelAndView page3() {
      Map<String, Object> params = new HashMap<>();
      return new ModelAndView("page3", params);
   }
}
```

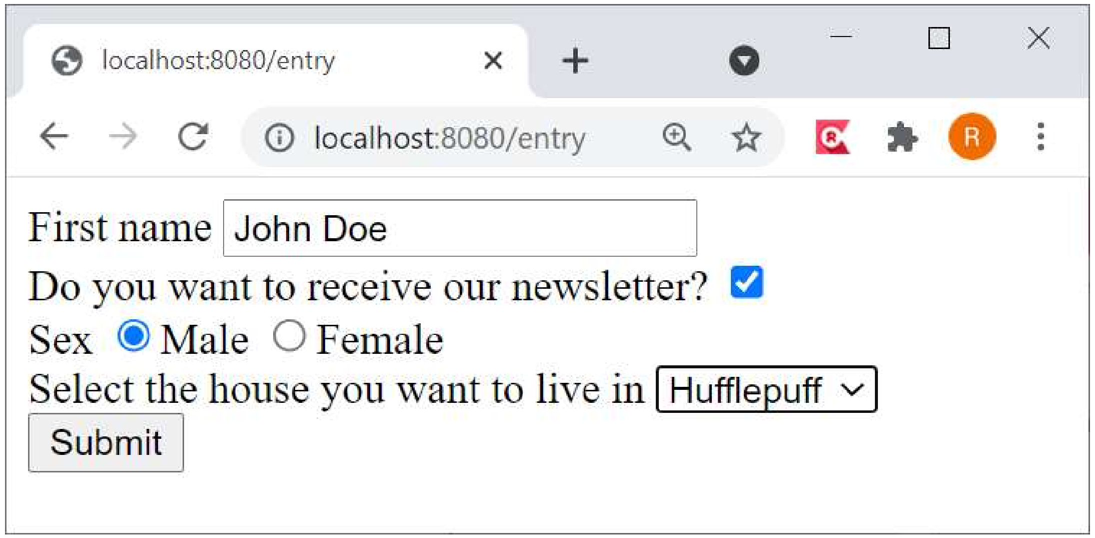






FORMS

Spring forms example

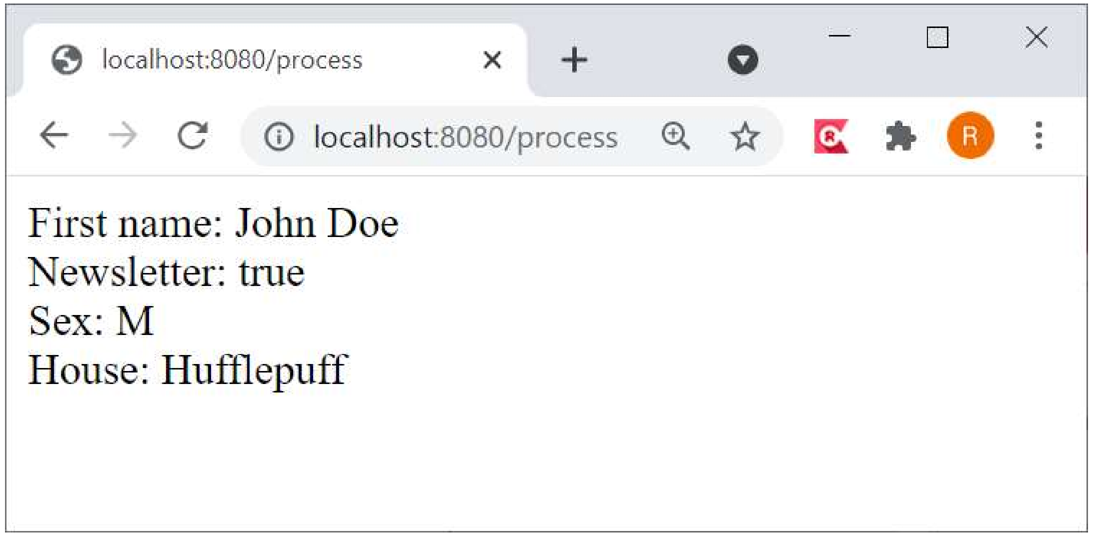


First name John Doe

Do you want to receive our newsletter? ✓

Sex ● Male ○ Female

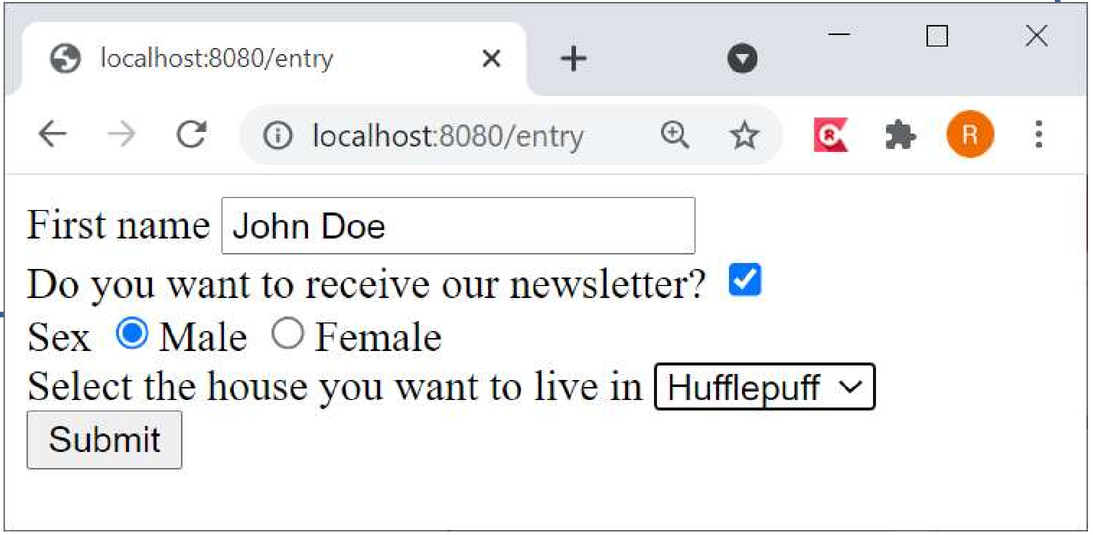
Select the house you want to live in Hufflepuff



```
First name: John Doe
Newsletter: true
Sex: M
House: Hufflepuff
```

entry.html

```html
<html>
<body>
<form action="process" method="post" >
   First name <input type="text" name="firstName" /> <br/>
   Do you want to receive our newsletter? <input type="checkbox" name="receiveNewsletter"/> <br/>
   Sex <input type="radio" name="sex" value="M" />Male
   <input type="radio" name="sex" value="F" />Female<br/>
   Select the house you want to live in
   <select name="house">
      <option value="Gryffindor">Gryffindor</option>
      <option value="Hufflepuff">Hufflepuff</option>
   </select><br/>
   <input type="submit" value="Submit">
</form>
</body>
</html>
```



First name John Doe

Do you want to receive our newsletter? ✓

Sex ● Male ○ Female

Select the house you want to live in Hufflepuff

Submit

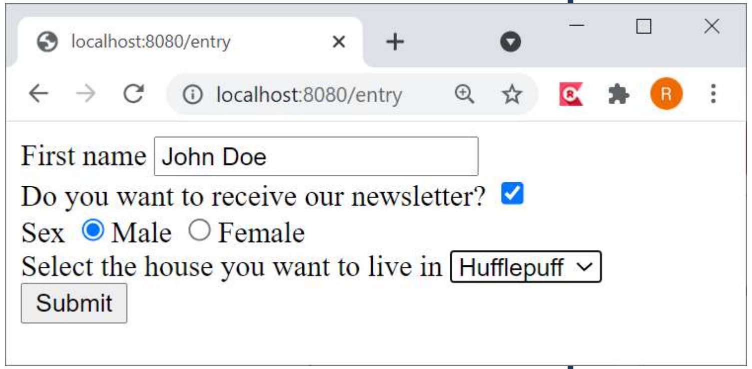
The controller

```
@Controller
public class StudentController {
  @PostMapping("/process")
  public ModelAndView processEntry(@RequestParam(value="firstName") String firstname,
                                  @RequestParam(value="receiveNewsletter", required = false) boolean receiveNewsletter,
                                  @RequestParam(value="sex") String sex,
                                  @RequestParam(value="house") String house
                                  ){
    ModelAndView modelandview = new ModelAndView();
    modelandview.addObject("firstname", firstname);
    modelandview.addObject("newsletter", receiveNewsletter);
    modelandview.addObject("sex", sex);
    modelandview.addObject("house", house);
    modelandview.setViewName("result");
    return modelandview;
  }

  @GetMapping("/entry")
  public ModelAndView showEntry(){
    ModelAndView modelandview = new ModelAndView();
    modelandview.setViewName("entry");
    return modelandview;
  }
}
```

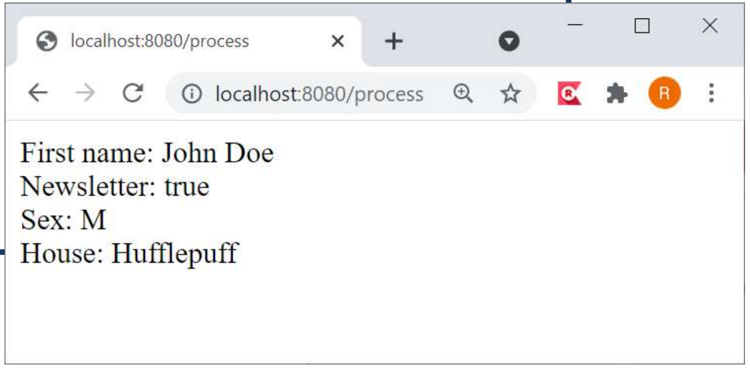
POST

Get all parameters

GET



First name John Doe
Do you want to receive our newsletter? ☑
Sex ● Male ○ Female
Select the house you want to live in Hufflepuff
Submit



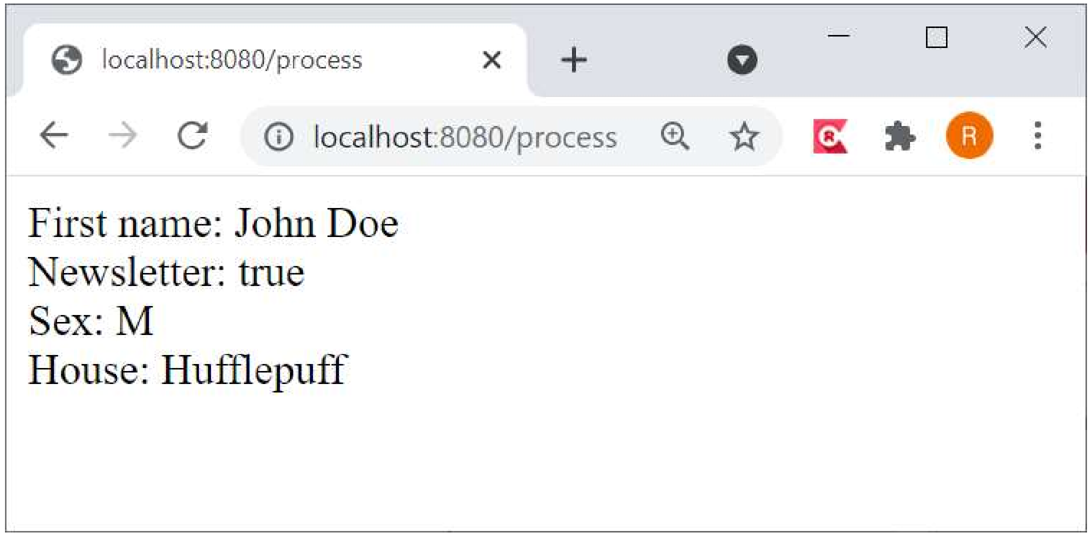
First name: John Doe
Newsletter: true
Sex: M
House: Hufflepuff

result.html

```
<html xmlns:th="http://www.thymeleaf.org">
<body>
First name: <span th:text="${ firstname }" /><br/>
Newsletter: <span th:text="${ newsletter }" /><br/>
Sex: <span th:text="${ sex }" /><br/>
House: <span th:text="${ house }" /><br/>
</body>
</html>
```

Thymeleaf namespace

th:text

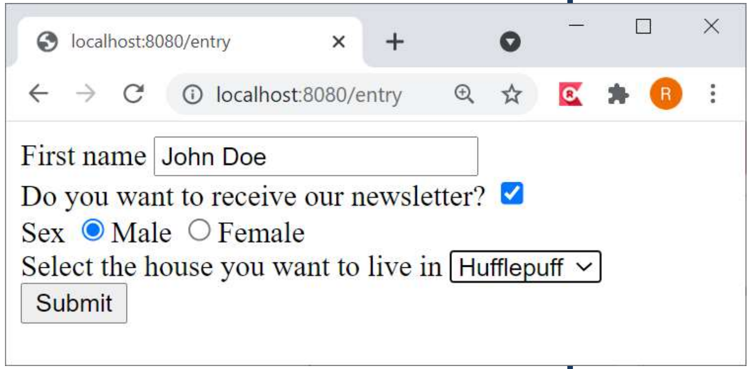


```
First name: John Doe
Newsletter: true
Sex: M
House: Hufflepuff
```

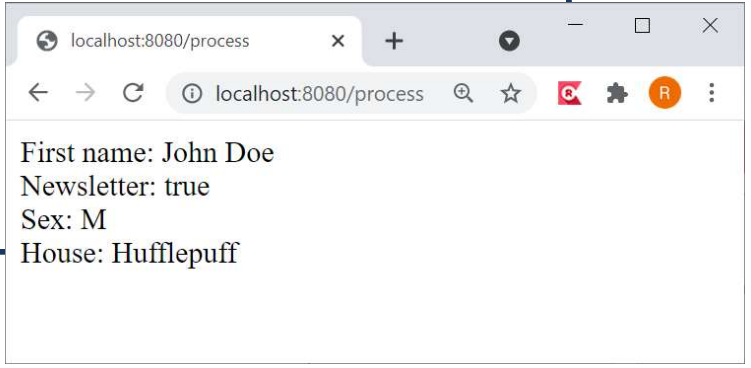
Form parameters

```
@Controller
public class StudentController {
  @PostMapping("/process")
  public ModelAndView processEntry(@RequestParam(value="firstName") String firstname,
                                  @RequestParam(value="receiveNewsletter", required = false) boolean receiveNewsletter,
                                  @RequestParam(value="sex") String sex,
                                  @RequestParam(value="house") String house
                                  ){
    ModelAndView modelandview = new ModelAndView();
    modelandview.addObject("firstname", firstname);
    modelandview.addObject("newsletter", receiveNewsletter);
    modelandview.addObject("sex", sex);
    modelandview.addObject("house", house);
    modelandview.setViewName("result");
    return modelandview;
  }

  @GetMapping("/entry")
  public ModelAndView showEntry(){
    ModelAndView modelandview = new ModelAndView();
    modelandview.setViewName("entry");
    return modelandview;
  }
}
```

What if we have many parameters?



First name John Doe
Do you want to receive our newsletter? ☑
Sex ● Male ○ Female
Select the house you want to live in Hufflepuff
Submit



```
First name: John Doe
Newsletter: true
Sex: M
House: Hufflepuff
```

Command Object

```
<html xmlns:th="http://www.thymeleaf.org">
<body>
<form action="process" method="post" th:object="${student}">
   First name <input type="text" th:field="*{firstName}" /> <br/>
   Do you want to receive our newsletter? <input type="checkbox" th:field="*{receiveNewsletter}"/>
<br/>
   Sex <input type="radio" name="sex" value="M" />Male
   <input type="radio" th:field="*{sex}" value="F" />Female<br/>
   Select the house you want to live in
   <select th:field="*{house}">
      <option value="Gryffindor">Gryffindor</option>
      <option value="Hufflepuff">Hufflepuff</option>
   </select><br/>
   <input type="submit" value="Submit">
</form>
</body>
</html>
```

th:object

th:field

Command object

```
public class Student {
   private String firstName;
   private boolean receiveNewsletter;
   private String sex;
   private String house;
...
```

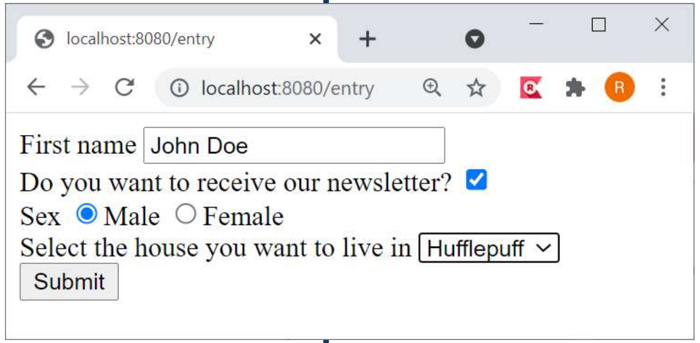
Attribute names are the same as the form element field names

The controller

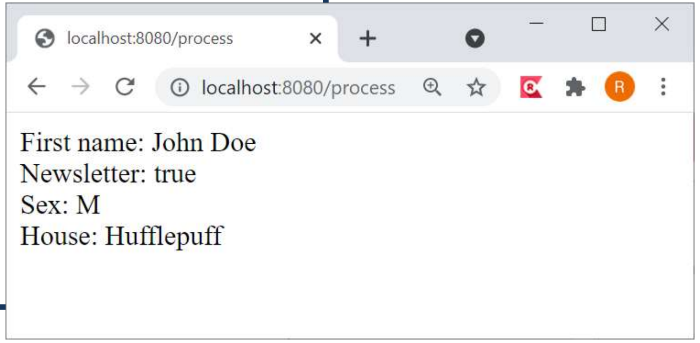
```
@Controller
public class StudentController {
  @PostMapping("/process")
  public ModelAndView processEntry(@ModelAttribute("student") Student student){
    ModelAndView modelandview = new ModelAndView();
    modelandview.addObject("firstname", student.getFirstName());
    modelandview.addObject("newsletter", student.isReceiveNewsletter());
    modelandview.addObject("sex", student.getSex());
    modelandview.addObject("house", student.getHouse());
    modelandview.setViewName("result");
    return modelandview;
  }

  @GetMapping("/entry")
  public ModelAndView showEntry(){
    Student student = new Student();
    ModelAndView modelandview = new ModelAndView();
    modelandview.addObject("student", student);
    modelandview.setViewName("entry");
    return modelandview;
  }
}
```

@ModelAttribute: The command object is the input parameter

Add a student (command object) to the model



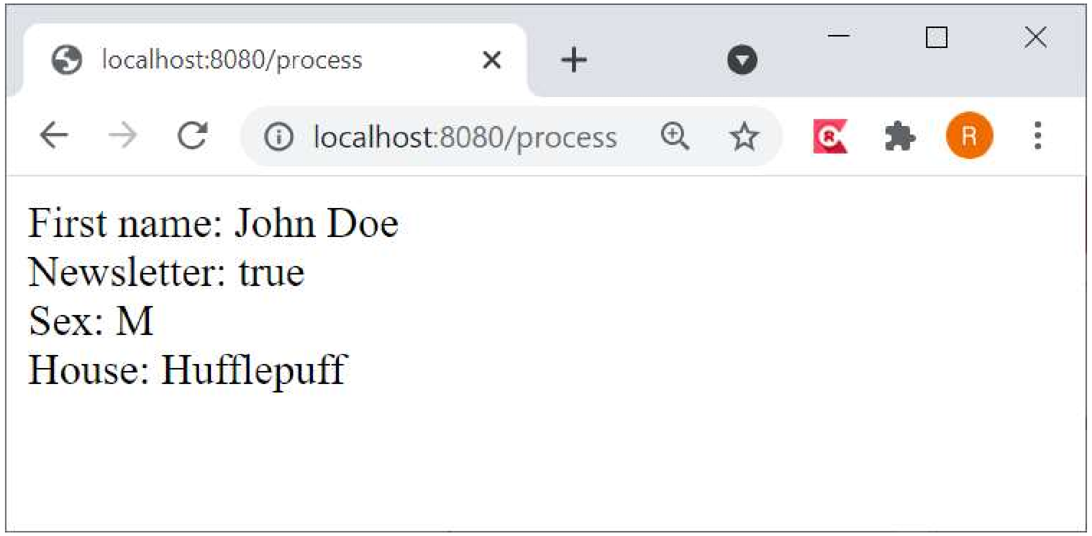
First name John Doe
Do you want to receive our newsletter? ✓
Sex ● Male ○ Female
Select the house you want to live in Hufflepuff
Submit



First name: John Doe
Newsletter: true
Sex: M
House: Hufflepuff

result.html

```
<html xmlns:th="http://www.thymeleaf.org">
<body>
First name: <span th:text="${ firstname }" /><br/>
Newsletter: <span th:text="${ newsletter }" /><br/>
Sex: <span th:text="${ sex }" /><br/>
House: <span th:text="${ house }" /><br/>
</body>
</html>
```



```
First name: John Doe
Newsletter: true
Sex: M
House: Hufflepuff
```

Main point

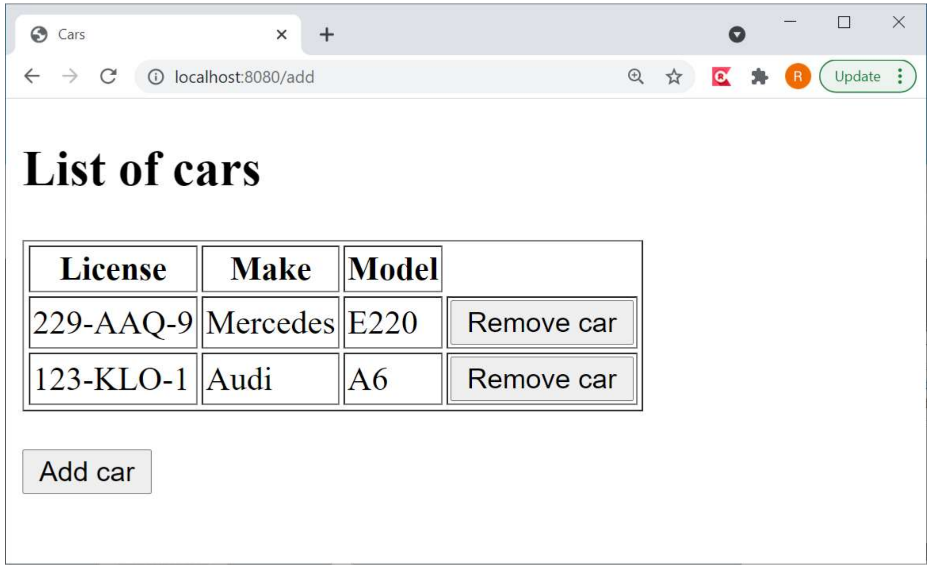
- The command object makes it easy to pass form data to the controller. *The Unified field is the source of all relative aspects of creation.*

SESSION SCOPE

3 scopes

- Request
  - Data in request scope is available during one request-reply call
- Session
  - Data in session scope is available during one browser session
  - It is available only for one user
- Context
  - Data in context scope is available during the lifetime of the application
  - It is available for all users

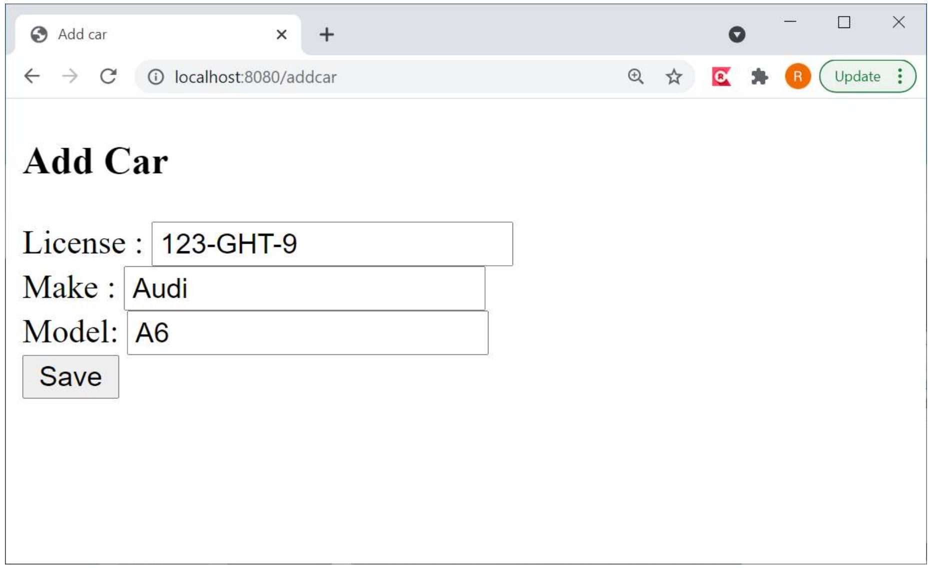
Session example



List of cars

License	Make	Model	
229-AAQ-9	Mercedes	E220	Remove car
123-KLO-1	Audi	A6	Remove car

Add car



Add Car

License :	123-GHT-9
Make :	Audi
Model:	A6

Save

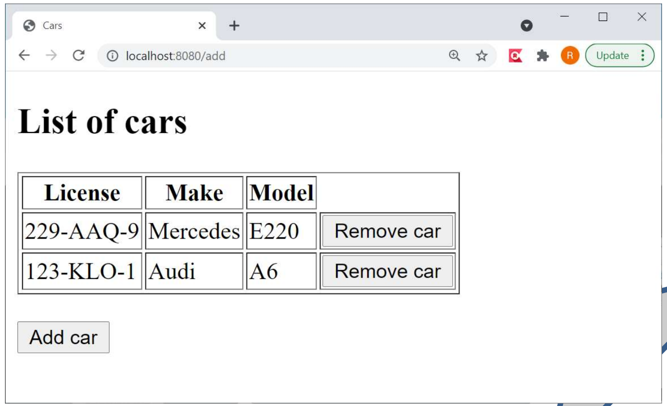
cars.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Cars</title></head>
<body>
<div id="header"><h2>List of cars</h2></div>
  <table class="datatable" border="1">
    <tr><th>License</th><th>Make</th><th>Model</th></tr>
    <tr th:each="car : ${carList}">
      <td th:text="${car.license}">license</td>
      <td th:text="${car.make}">make</td>
      <td th:text="${car.model}">model</td>
      <td>
        <form action="removecar" method="post">
          <button type="submit" name="licence" th:value="${car.license}">Remove car</button>
        </form>
      </td>
    </tr>
  </table>
  <br/>
  <form action="addcar" method="post">
    <button type="submit">Add car</button>
  </form>
</body>
</html>
```

th:each

th:text

th:value



List of cars

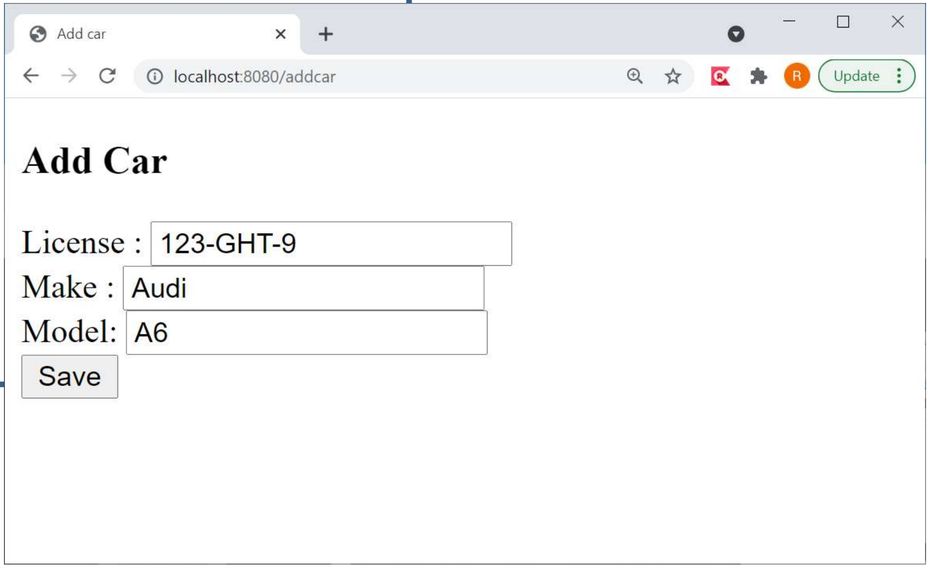
License	Make	Model	
229-AAQ-9	Mercedes	E220	Remove car
123-KLO-1	Audi	A6	Remove car

Add car

addcar.html

```
<html xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>Add car</title></head>
<body>
</div>
    <h3>Add Car</h3>
    <form th:object="${car}" action="add" method="post">
      License : <input type="text" th:field="*{license}" /><br />
      Make : <input type="text" th:field="*{make}" /><br />
      Model: <input type="text" th:field="*{model}" /><br />
      <input type="submit" value="Save" />
    </form>
  <br />

</body>
</html>
```

Command object



Add Car

License : 123-GHT-9
Make : Audi
Model: A6
Save

CarController (1/3)

```
@Controller
public class CarController {
@GetMapping("/cars")
public ModelAndView init(HttpSession session) {
   //get the carlist from the session
   Map<String, Car> carList = (Map<String, Car>) session.getAttribute("carList");
   //if there is no carlist in the session, create one.
   if (carList == null) {
      carList = new HashMap<String, Car>();
      session.setAttribute("carList", carList);
   }
   Map<String, Object> params = new HashMap<>();
   params.put("carList", carList.values());
   return new ModelAndView("cars", params);
}
...
```

Get session as parameter

Put list of cars in the model

Show cars page

```
public class Car {
   private String license;
   private String make;
   private String model;
. . .
```

CarController (2/3)

```
@PostMapping("/addcar")
public ModelAndView addcar(HttpSession session) {
   Map<String, Object> params = new HashMap<>();
   params.put("car", new Car());
   return new ModelAndView("addcar", params);
}

@PostMapping("/add")
public ModelAndView add( HttpSession session, @ModelAttribute("car") Car car) {
   Map<String, Object> params = new HashMap<>();
   if (car != null) {
      //get the carlist from the session
      Map<String, Car> carList = (Map<String, Car>) session.getAttribute("carList");
      //if there is no carlist in the session, create one.
      if (carList == null) {
         carList = new HashMap<String, Car>();
         session.setAttribute("carList", carList);
      }
      //add the car to the carlist
      carList.put(car.getLicense(), car);
      params.put("carList", carList.values());
   }
   return new ModelAndView("cars", params);
}
```

Navigate to the addcar page

Put car in the model

Get the command object

Put list of cars in the model

Navigate to the cars page

CarController (3/3)

```
@PostMapping("/removecar")
public ModelAndView removecar(@RequestParam("licence") String license, HttpSession session) {
   Map<String, Object> params = new HashMap<>();
   if (license != null) {
      //get the carlist from the session
      Map<String, Car> carList = (Map<String, Car>) session.getAttribute("carList");
      //if there is no carlist in the session, create one.
      if (carList == null) {
         carList = new HashMap<String, Car>();
         session.setAttribute("carList", carList);
      }
      //add the car to the carlist
      carList.remove(license);
      params.put("carList", carList.values());
   }
   return new ModelAndView("cars", params);
}
```

FORM VALIDATION

JSR 303 standard

- Standard set of constraints annotations
- Can be used in all layers of the application

```
@Entity
public class Customer {
    @NotNull
    private String firstname;
    @NotNull
    private String lastname;
    @NotNull
    private String street;
    @NotNull
    private String city;
    @Length(max=5)
    @Pattern(regex="[0-9]+")
    @NotNull
    private String zip;
    @NotNull
    private String state;
    @Length(max=20)
    @NotNull
    private String country;
    @Past
    private Date dateofBirth;

    …
}
```

Constraints annotations

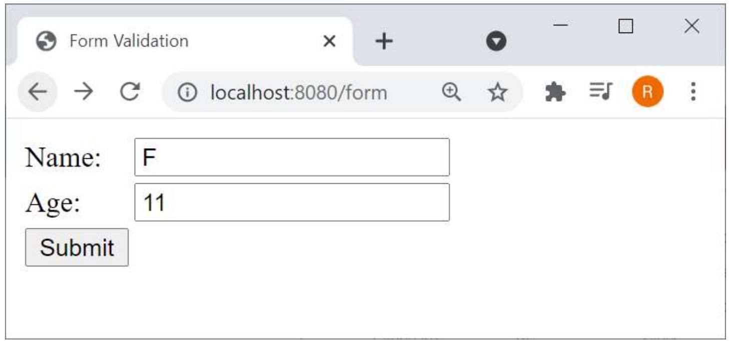
annotatie	toepasbaar op	runtime checking	DDL generatie impact
@Length(min=, max=)	String	check if the string length match the range	Column length will be set to max
@Max(value=)	numeric or string representation of a numeric	check if the value is less than or equals to max	Add a check constraint on the column
@Min(value=)	numeric or string representation of a numeric	check if the value is more than or equals to min	Add a check constraint on the column
@NotNull	property	check if the value is not null	Column(s) are not null
@NotEmpty	property	check if the string is not null nor empty. Check if the connection is not null nor empty	Column(s) are not null (for String)
@Past	date or calendar	check if the date is in the past	none
@Future	date or calendar	check if the date is in the future	none
@Pattern(regex="regexp", flag=) or @Patterns({@Pattern(...)})	String	check if the property match the regular expression given a match flag (see java.util.regex.Pattern)	none
@Range(min=, max=)	numeric or string representation of a numeric	check if the value is between min and max (included)	Add a check constraint on the column

Constraints annotations

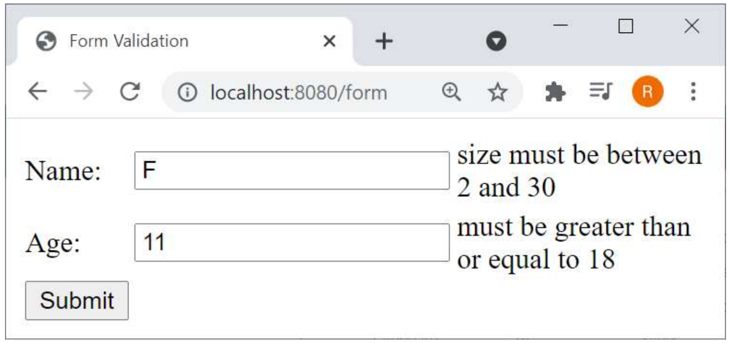
annotatie	toepasbaar op	runtime checking	DDL generatie impact
@Size(min=, max=)	array, collection, map	check if the element size is between min and max (included)	none
@AssertFalse	property	check that the method evaluates to false	none
@AssertTrue	property	check that the method evaluates to true	none
@Valid	object	perform validation recursively on the associated object	none
@Email	string	check whether the string is conform to the email address specification	none
@CreditCardNumber	string	check whether the string is a well formated credit card number	none
@Digits	numeric or string representation of a numeric	check whether the property is a number having up to integerDigits integer digits and fractionalDigits fractonal digits	define column precision and scale

Form validation

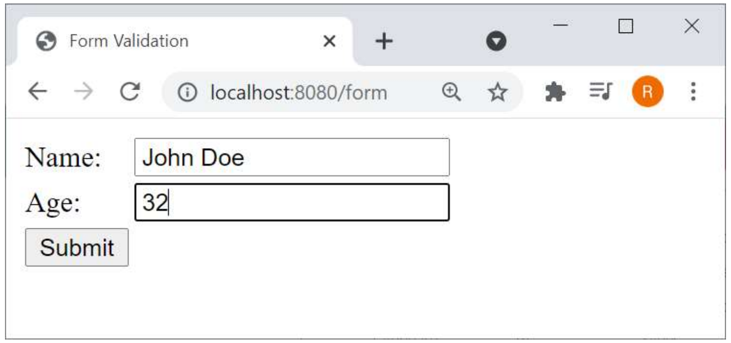
```
public class Person {
   @NotNull
   @Size(min=2, max=30)
   private String name;

   @NotNull
   @Min(18)
   private Integer age;
```



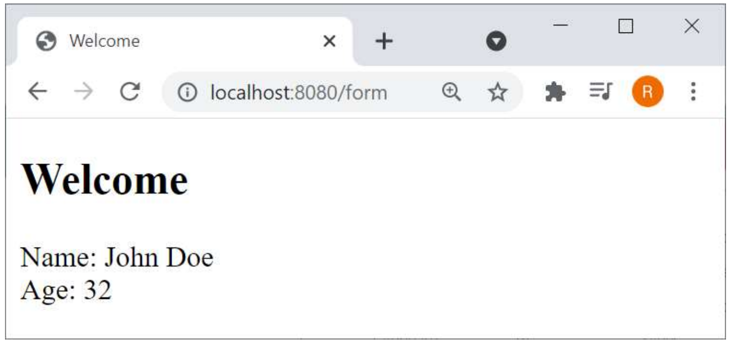
Name: F
Age: 11
Submit



Name: F size must be between 2 and 30
Age: 11 must be greater than or equal to 18
Submit



Name: John Doe
Age: 32
Submit



Welcome

Name: John Doe
Age: 32

Form validation

```
@Controller
public class FormController {
   @GetMapping("/form")
   public ModelAndView form() {
      Person person = new Person();

      ModelAndView mav = new ModelAndView();
      mav.addObject("person", person);
      mav.setViewName("form");

      return mav;
   }

   @PostMapping("/form")
   public ModelAndView formSubmit(@Valid Person person, BindingResult bindingResult) {
      ModelAndView mav = new ModelAndView();
      if (bindingResult.hasErrors()) {
         mav.setViewName("form");
         return mav;
      }
      //if there are no errors, show form success screen
      mav.addObject("person", person);
      mav.setViewName("success");
      return mav;
   }
}
```

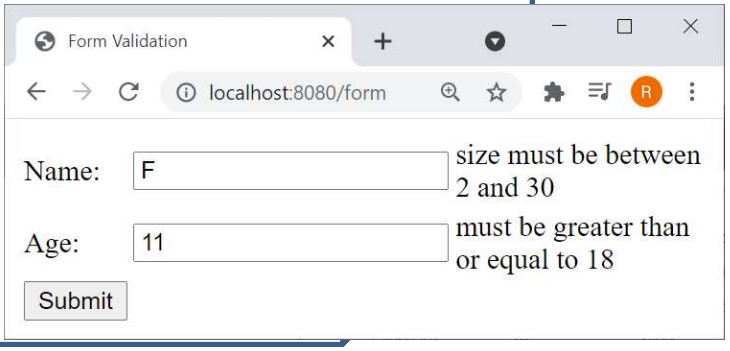
Add the person to the model

Validate the person

Form validation

```
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <title>Form Validation</title>
</head>
<body >
<form action="form" th:object="${person}" method="post">
  <table>
    <tr>
      <td>Name:</td>
      <td><input type="text" th:field="*{name}" /></td>
      <td th:if="${#fields.hasErrors('name')}" th:errors="*{name}">Name Error</td>
    </tr>
    <tr>
      <td>Age:</td>
      <td><input type="text" th:field="*{age}" /></td>
      <td th:if="${#fields.hasErrors('age')}" th:errors="*{age}">Age Error</td>
    </tr>
    <tr>
      <td><button type="submit">Submit</button></td>
    </tr>
  </table>
</form>
</body>
</html>
```

Show error message



Name:	F	size must be between 2 and 30
Age:	11	must be greater than or equal to 18

Main point

- The JSR 303 constraints can be used for form validation. *Daily access to pure consciousness leads more happiness and fulfilment in life.*

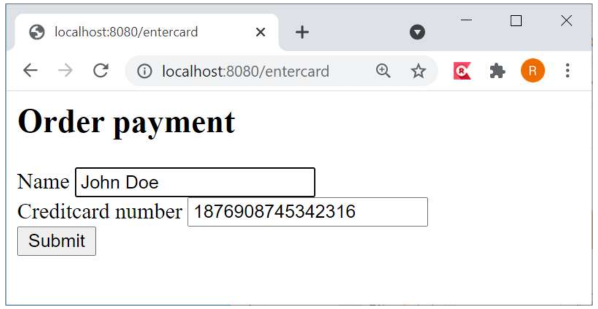
PRG PATTERN

POST and GET

- GET requests are idempotent
  - It does not matter how often you send a GET request, the state on the sever is always the same
- POST requests are not idempotent
  - It does matter how often you send a POST request, the state on the sever will be changed

payment.html and thankyou.html

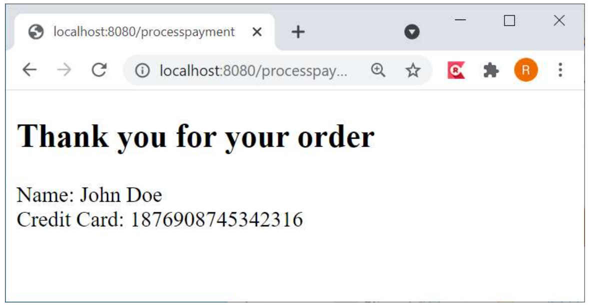
```
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h2>Order payment</h2>
<form action="processpayment" method="post" >
   Name <input type="text" name="name" /> <br/>
   Creditcard number <input type="text" name="creditcardnumber" /> <br/>
   <input type="submit" value="Submit">
</form>
</body>
</html>
```



Order payment

Name John Doe
Creditcard number 1876908745342316
Submit

```
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h2>Thank you for your order</h2>
Name: <span th:text="${ name }" /><br/>
Credit Card: <span th:text="${ creditcardnumber }" /><br/>
</body>
</html>
```



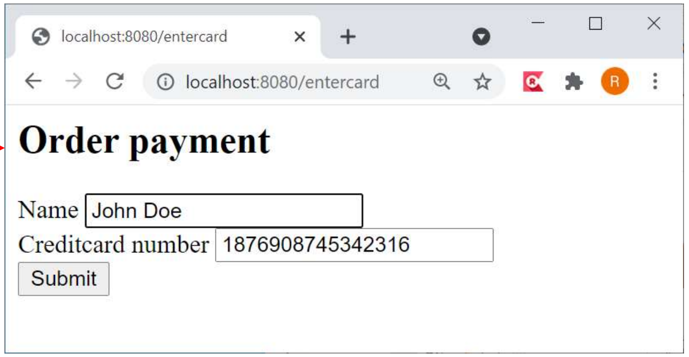
Thank you for your order

Name: John Doe
Credit Card: 1876908745342316

PaymentController

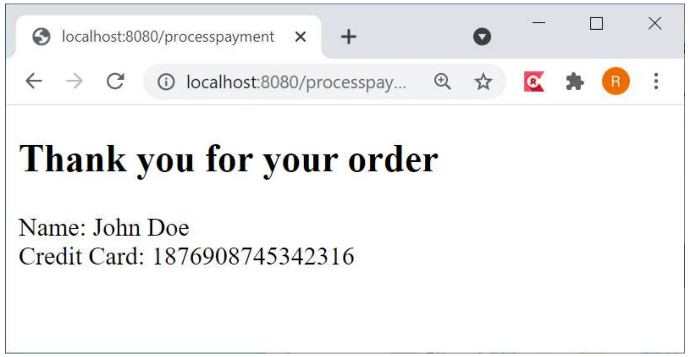
```
@Controller
public class PaymentController {

  @PostMapping("/processpayment")
  public ModelAndView processPayment(@RequestParam(value="name") String name,
                                  @RequestParam(value="creditcardnumber") String creditCardNumber) {
    System.out.println("process order from "+name+" with credit card: "+creditCardNumber);
    Map<String, Object> params = new HashMap<>();
    params.put("name", name);
    params.put("creditcardnumber", creditCardNumber);
    return new ModelAndView("thankyou", params);
  }
  @GetMapping("/entercard")
  public ModelAndView page2() {
    Map<String, Object> params = new HashMap<>();
    return new ModelAndView("payment", params);
  }
}
```

The problem


localhost:8080/entercard

Order payment

Name John Doe
Creditcard number 1876908745342316
Submit

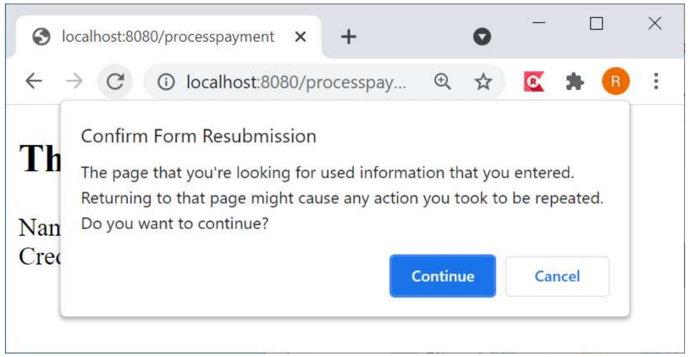

localhost:8080/processpayment

Thank you for your order

Name: John Doe
Credit Card: 1876908745342316

```
process order from John Doe with credit card: 1876908745342316
```

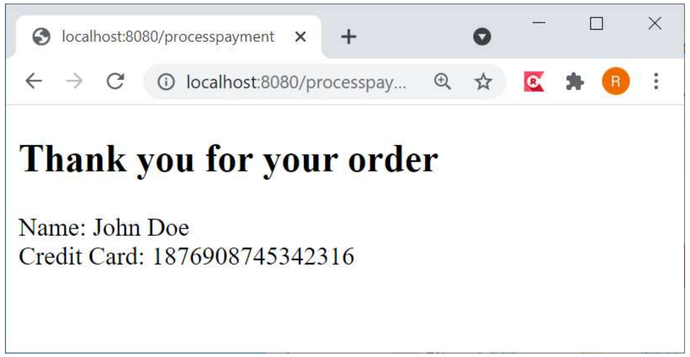
Click refresh button


localhost:8080/processpayment

Confirm Form Resubmission

The page that you're looking for used information that you entered. Returning to that page might cause any action you took to be repeated. Do you want to continue?

Continue Cancel


localhost:8080/processpayment

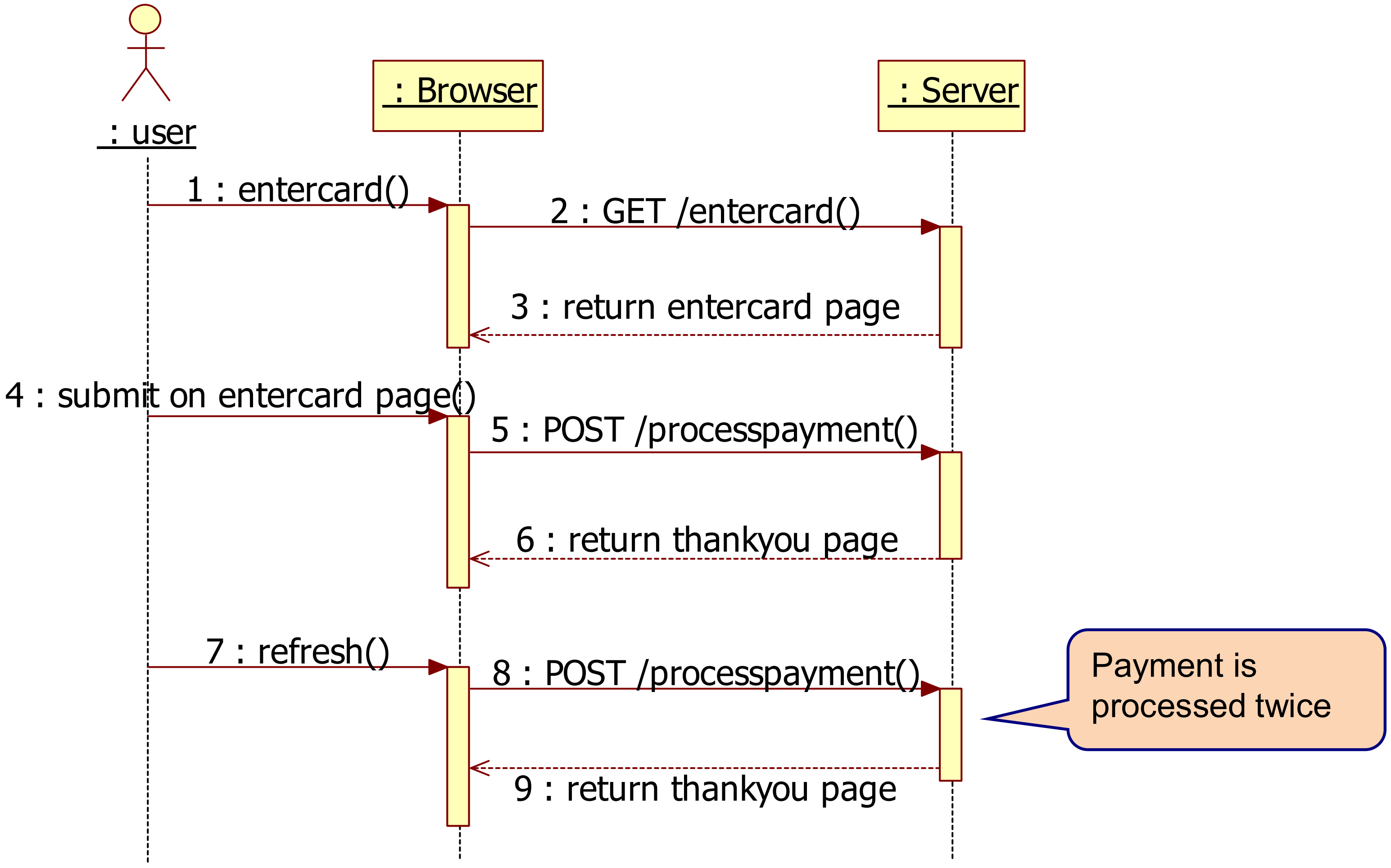
Thank you for your order

Name: John Doe
Credit Card: 1876908745342316

Payment is processed twice

```
process order from John Doe with credit card: 1876908745342316
process order from John Doe with credit card: 1876908745342316
```

The problem


: user
: Browser
: Server
1 : entercard()
2 : GET /entercard()
3 : return entercard page
4 : submit on entercard page()
5 : POST /processpayment()
6 : return thankyou page
7 : refresh()
8 : POST /processpayment()
9 : return thankyou page
Payment is processed twice

Solution: Post-Redirect-Get (PRG)

- Never show pages in response to POST
- Always load pages using GET
- Navigate from POST to GET using REDIRECT

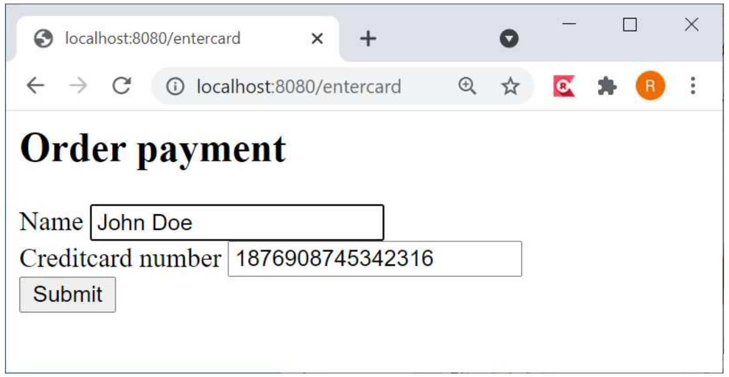
PRG pattern

```
@PostMapping("/processpayment")
public ModelAndView processPayment(@RequestParam(value="name") String name,
                         @RequestParam(value="creditcardnumber") String creditCardNumber) {
   System.out.println("process order from "+name+" with credit card: "+creditCardNumber);
   Map<String, Object> params = new HashMap<>();
   params.put("name", name);
   params.put("creditcardnumber", creditCardNumber);
   return new ModelAndView("redirect:thankyou", params);
}
@GetMapping("/entercard")
public ModelAndView enterCard() {
   Map<String, Object> params = new HashMap<>();
   return new ModelAndView("payment", params);
}

@GetMapping("/thankyou")
public ModelAndView thankYou() {
   Map<String, Object> params = new HashMap<>();
   return new ModelAndView("thankyou", params);
}
```

redirect

Get mapping for redirect

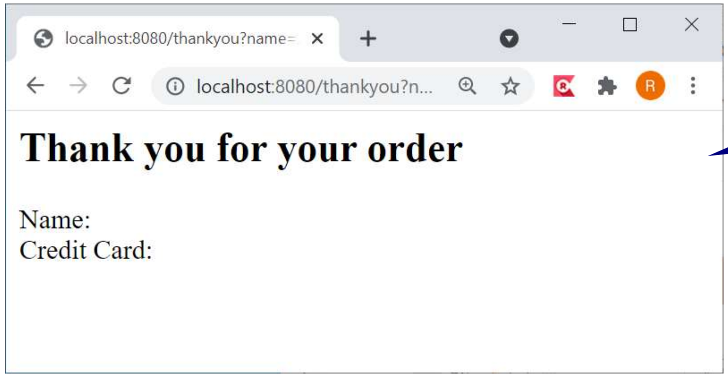
Parameters are lost



localhost:8080/entercard

Order payment

Name John Doe
Creditcard number 1876908745342316
Submit

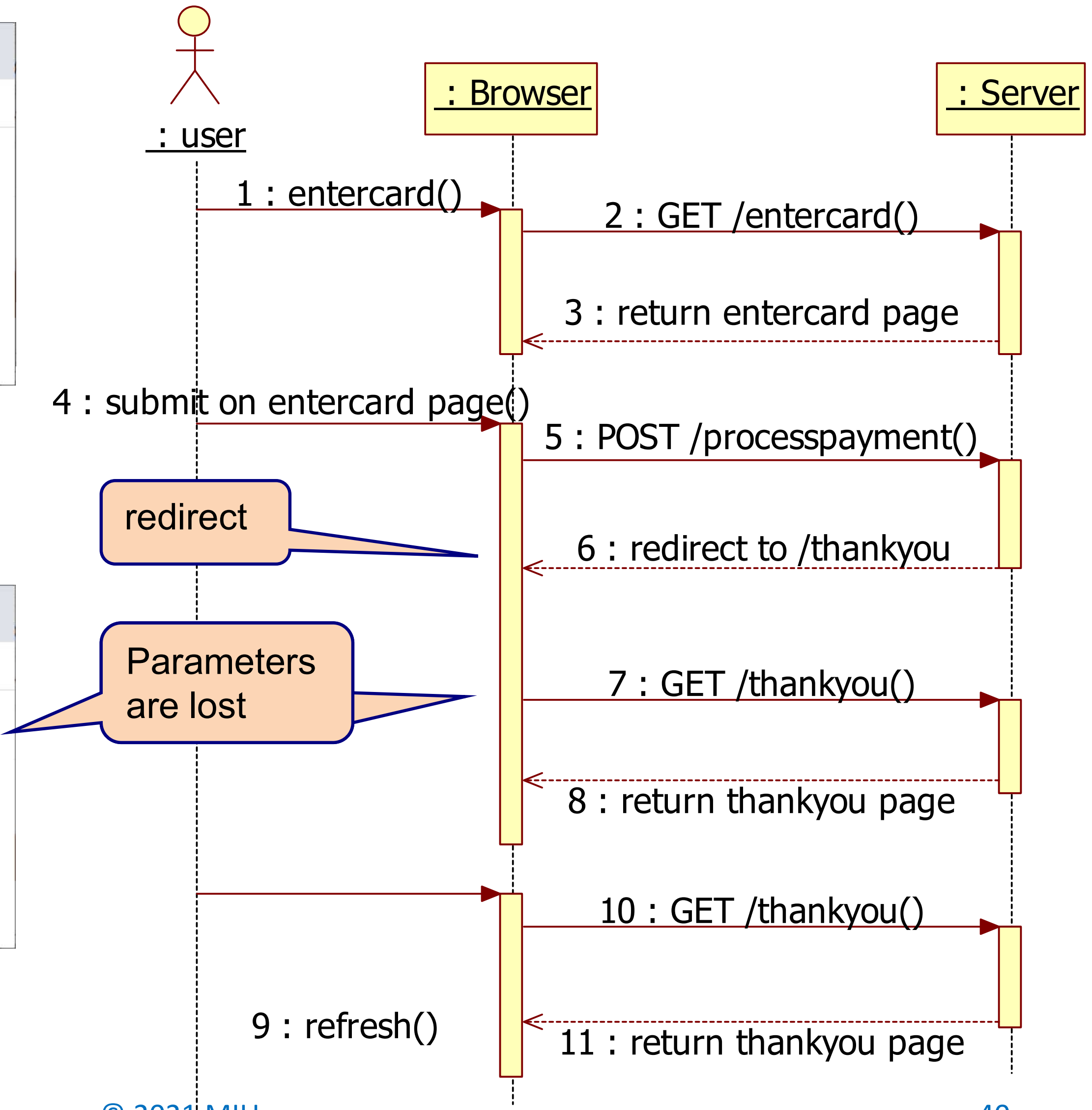


localhost:8080/thankyou?name=

localhost:8080/thankyou?n...

Thank you for your order

Name:
Credit Card:



Flash attributes

```
@Controller
public class PaymentController {
  @PostMapping("/processpayment")
  public ModelAndView processPayment(@RequestParam(value="name") String name,
                                     @RequestParam(value="creditcardnumber") String creditCardNumber,
                                     RedirectAttributes redirectAttributes) {
    System.out.println("process order from "+name+" with credit card: "+creditCardNumber);
    Map<String, Object> params = new HashMap<>();
    redirectAttributes.addFlashAttribute( "name", name);
    redirectAttributes.addFlashAttribute( "creditcardnumber", creditCardNumber);
    return new ModelAndView("redirect:thankyou", params);
  }
  @GetMapping("/entercard")
  public ModelAndView enterCard() {
    Map<String, Object> params = new HashMap<>();
    return new ModelAndView("payment", params);
  }

  @GetMapping("/thankyou")
  public ModelAndView thankYou() {
    Map<String, Object> params = new HashMap<>();
    return new ModelAndView("thankyou", params);
  }
}
```

Flash attributes

Will be stored in the session during the request-reply call. Will be removed from the session after the reply is sent.

Main point

- GET requests are idempotent, POST requests are not idempotent. *The Unified field is the source of all change.*

Connecting the parts of knowledge with the wholeness of knowledge

1. SpringMVC is a server-side web framework that supports all necessary web specific functionality
2. The PRG pattern redirects all responses of a POST request to a GET request

---

3. **Transcendental consciousness** is the direct experience of pure consciousness, the unified field of all the laws of nature.
4. **Wholeness moving within itself:** In unity consciousness, one appreciates the inherent underlying unity that underlies all the diversity of creation.