ESCUELA POLITÉCNICA SUPERIOR DE CÓRDOBA
Universidad de Córdoba

EPSC

UNIVERSIDAD D CÓRDOBA

Programación web

Prácticas

Semana 5: HTML (I)

Aurora Ramírez Quesada (aramirez@uco.es)

Departamento de Ciencia de la Computación e Inteligencia Artificial

Universidad de Córdoba

Índice de contenido

Etiquetas básicas
Estructura general

- **Marcado estructural**: elementos que definen la estructura de una página (encabezados, párrafos, etc.)
- Todas las páginas tienen los siguientes elementos principales:
  - **<html>**: Engloba a la página completa
  - **<head>**: Detalla información sobre la página, por ejemplo, el título y metainformación
  - **<body>**: Incluye el contenido de la página (texto, imágenes, tablas, etc.)

```
<!DOCTYPE html>
<html>
      <head>
            Información sobre la página
      </head>
      <body>
            Contenido de la página
      </body>
</html>
```

! **HTML** es un lenguaje de marcas **muy extenso**. Para aprender sobre elementos concretos y ver más ejemplos: https://www.w3schools.com/html

Etiquetas básicas
Estructura general

https://www.w3schools.com/tags/tag_meta.asp

- Dentro de la etiqueta **<head>** es habitual definir los siguientes elementos:
  - **<title>**: Muestra el título de la página en la barra de título del navegador y al marcar la página
  - **<meta>**: Contiene información sobre el propio documento HTML. No es visible a los usuarios, pero es utilizada por navegadores y motores de búsqueda. Se compone de atributos como:
    - "*charset*": Para indicar la codificación de caracteres utilizada (recomendable utilizar **UTF-8**)
    - Pares atributo-valor (*name*/*content*) para dar información sobre el contenido, autoría de la página o establecer cómo debe visualizarse según el dispositivo desde el que se navega

```
<head>
  <title>This is the title of the page</title>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
```

Etiquetas básicas
Estructura general

- Aunque dentro de la etiqueta **<body>** se pueden ubicar distintos tipos de contenedores, por el momento estudiaremos solo los encabezados y elementos de bloque
- Los elementos de bloque contienen una región completa de contenido, y el navegador coloca un margen de espacio en blanco entre ellos
  - Los encabezados <**h**> se estructuran en niveles y ocupan todo el ancho disponible. Hay hasta 6 niveles:

```
<body>
  <h1>Encabezado de nivel 1</h1>
  <h2>Encabezado de nivel 2</h2>
  <h3>Encabezado de nivel 3</h3>
  <h4>Encabezado de nivel 4</h4>
  <h5>Encabezado de nivel 5</h5>
  <h6>Encabezado de nivel 6</h6>
</body>
```

  - Los párrafos de texto utilizan la etiqueta <**p**> y también ocupan todo el ancho (introducen salto de línea después)

```
<body>
  <h1>Encabezado de nivel 1</h1>
  <p>Esto es el primer párrafo</p>
  <p>Esto es el segundo párrafo</p>
</body>
```

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_headers

http://www.w3schools.com/tags/tryit.asp?filename=tryhtml_paragraphs2

Etiquetas básicas
Listas

http://www.w3schools.com/html/html_lists.asp

- Las listas también son elementos de bloque que el navegador muestra con un sangrado por defecto
- HTML nos proporciona diferentes tipos de listas (listados de elementos):
  - Listas de definición
  - Listas desordenadas
  - Listas ordenadas
- Listas de definición: **<dl>**
  - Consiste en una serie de términos y sus definiciones
  - Se escriben en pareja, aunque no es obligatorio:
    - **<dt>**: contiene el término a definir
    - **<dd>**: contiene la definición del término

```
<dl>
    <dt>Sashimi</dt>
    <dd>Sliced raw fish that is served with condiments such
        as shredded daikon radish or ginger root, wasabi and
        soy sauce</dd>
    <dt>Scale</dt>
    <dd>A device used to accurately measure the weight of
        ingredients</dd>
    <dd>A technique by which the scales are removed from the
        skin of a fish</dd>
    <dt>Scamorze</dt>
    <dt>Scamorzo</dt>
    <dd>An Italian cheese usually made from whole cow's milk
        (although it was traditionally made from buffalo milk)</dd>
</dl>
```


Sashimi
: Sliced raw fish that is served with condiments such as shredded daikon radish or ginger root, wasabi and soy sauce

Scale
: A device used to accurately measure the weight of ingredients
: A technique by which the scales are removed from the skin of a fish

Scamorze

Scamorzo
: An Italian cheese usually made from whole cow's milk (although it was traditionally made from buffalo milk)

Etiquetas básicas
Listas

http://www.w3schools.com/html/html_lists.asp

- Listas ordenadas: **<ol>**
  - Para cada elemento de la lista: **<li>**
  - El atributo *type* permite modificar el tipo de numeración (números, letras, números romanos, etc.)

```
<ol>
    <li>Chop potatoes into quarters</li>
    <li>Simmer in salted water for 15-20
        minutes until tender</li>
    <li>Heat milk, butter and nutmeg</li>
    <li>Drain potatoes and mash</li>
    <li>Mix in the milk mixture</li>
</ol>
```


1. Chop potatoes into quarters
2. Simmer in salted water for 15-20 minutes until tender
3. Heat milk, butter and nutmeg
4. Drain potatoes and mash
5. Mix in the milk mixture

- Listas no ordenadas: **<ul>**
  - Para cada elemento de la lista: **<li>**
  - El atributo *type* permite modificar el tipo de viñeta (círculos, cuadrado, diamante)
  - Se recomienda usar la propiedad *list-style-type* de **CSS** en su lugar: "*disc*", "*circle*", "*square*", "*none*"

```
<ul style="list-style-type:disc;">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Etiquetas básicas
Enlaces

- La característica definitoria de la Web:
  - Permiten pasar de una página web a otra
  - Permiten la idea misma de explorar y navegar
- Tipos de enlaces que comúnmente se encuentran:
  - De un sitio web a otro
  - De una página a otra en el mismo sitio web
  - Desde una parte del sitio web a otra del mismo sitio
  - Que se abre en una nueva ventana del navegador
  - Que inicie el programa de correo electrónico y dirija un nuevo correo electrónico a alguien
- Por defecto, los navegadores muestran los enlaces en azul con subrayado
- El texto del enlace debe explicar a dónde se llevará al usuario si hace clic en él

Etiquetas básicas
Enlaces

- Sintaxis de la etiqueta **<a>**
  - El texto entre la etiqueta y su cierre es el texto que se mostrará en el navegador para que el usuario haga clic en él, y debe ser informativo
  - El atributo *href* indica la dirección del enlace:
    - Ruta absoluta (vínculo al exterior): debe ser la URL completa
    - Ruta relativa (vínculo interno): enlace a documentos internos de la aplicación, que pueden estar ubicados en diferentes carpetas (escribir ruta en relación con la página actual)
    - Ancla (#): enlaza a una sección para desplazarse por la página actual. La sección tendrá un atributo *id* con el mismo nombre
  - El atributo *target* permite abrir el enlace en una nueva pestaña usando el valor "*_blank*"

```
<a href="https://www.uco.es" target="_blank">UCO</a>
```

```
<a href="#top">Ir arriba</a>
```

https://www.w3schools.com/tags/tag_a.asp

Formularios
Introducción

- Colección de elementos que permiten al visitante introducir información con diferentes propósitos:
  - Agregar un cuadro de búsqueda simple
  - Crear cuadros de seguridad de acceso a aplicaciones
  - Recopilar información del visitante
- **Funcionamiento**:
  - El usuario completa el formulario y presiona un botón para enviar la información al servidor
  - Cada campo del formulario tiene asociado un nombre que se envía junto con el valor introducido
  - El servidor procesa la información y envía una respuesta de vuelta al navegador, según proceda

https://www.w3schools.com/html/html_forms.asp

Formularios
Introducción

- **Estructura**:
  - Cada control de formulario se sitúa dentro de un elemento **<form>**
  - En el atributo *action*=”*URL*” se indica la URL del servidor para enviar el formulario
  - En el atributo *method*=”*httpVerb*” se especifica la operación HTTP (GET, POST)
  - El atributo *id*=”*someId*” identifica de forma exclusiva el formulario en la página

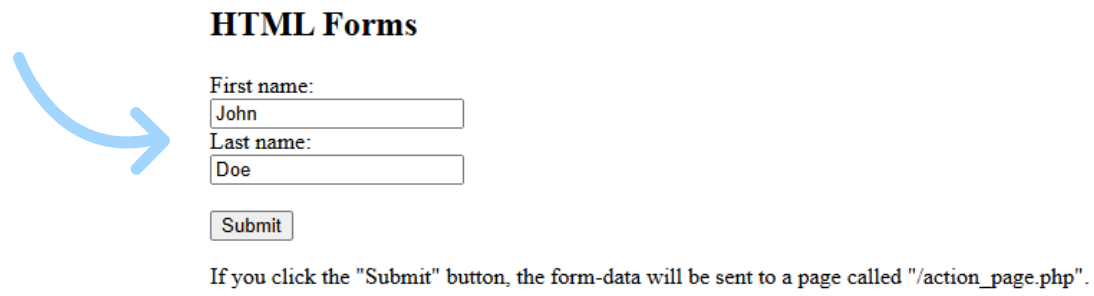
```
<!DOCTYPE html>
<html>
<body>

<h2>HTML Forms</h2>

<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>

</body>
</html>
```



! **GET** añade los datos a la URL (longitud limitada), por lo que no debe utilizarse para datos sensibles. **POST** no tiene limitación de tamaño y los datos se envían dentro del cuerpo de la HTTP *request*

https://www.w3schools.com/html/html_forms_attributes.asp
https://www.w3schools.com/html/tryit.asp?filename=tryhtml_form_submit

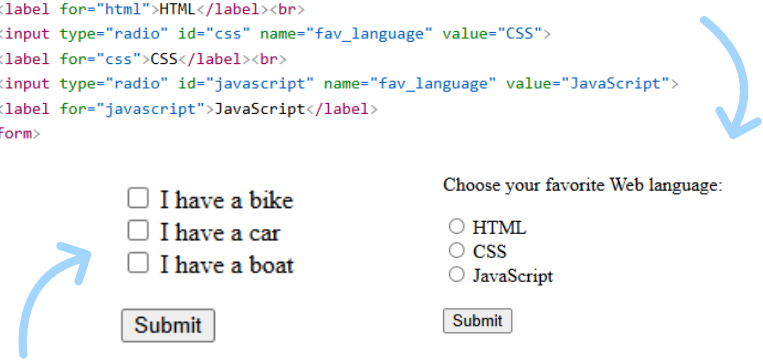
Formularios
Controles de formulario

- Cada elemento **<form>** incluye uno o varios elementos según el tipo de información a recopilar: texto, selección de ítems, botones, etc.
- El elemento más utilizado es **<input>**, que se muestra de diferente manera según su atributo "*type*"
  - Texto: <input type="*text*"> para campo de texto en una línea
  - Contraseña: <input type="*password*"> para ocultar caracteres
  - Botón de selección única: <input type="*radio*">
  - Botón de selección múltiple (0 o más): <input type="*checkbox*">
  - Fecha: <input type="*date*"> para elegir sobre un calendario
  - Botón de envío: <input type="*submit*"> para enviar los datos
  - Botón de reseteo: <input type="*reset*"> para resetear los campos
  - Otros para formato email, teléfono, rangos de números, etc.

https://www.w3schools.com/html/html_form_input_types.asp

```
<p>Choose your favorite Web language:</p>

<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
```

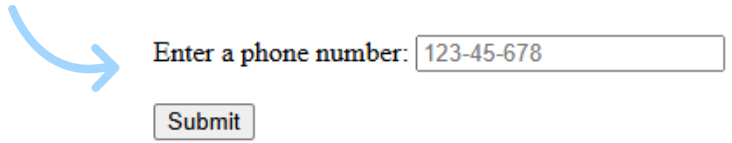



```
<form action="/action_page.php">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
</form>
```

Formularios
Controles de formulario

- Los elementos <input> pueden tener otros atributos que ayudan a validar la información:
  - ***value***: Establece un valor inicial
  - ***readonly***: Valor de solo lectura
  - ***size***: Número de caracteres visibles
  - ***maxlength***: Número máximo de caracteres permitidos
  - ***min***/***max***: Valor mínimo y máximo permitido para numéricos o fechas
  - ***multiple***: El usuario puede indicar más de un valor (email o ficheros)
  - ***pattern***: El valor introducido debe encajar con una expresión regular
  - ***placeholder***: Ejemplo del formato esperado que se visualiza mientras que el usuario no introduzca ningún valor
  - ***required***: El campo debe rellenarse obligatoriamente

```
<form action="/action_page.php">
  <label for="phone">Enter a phone number:</label>
  <input type="tel" id="phone" name="phone" placeholder="123-45-678" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}"><br>
<br>
  <input type="submit" value="Submit">
</form>
```



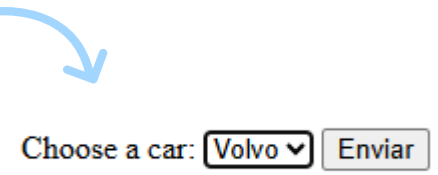

```
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" size="50"><br>
  <label for="pin">PIN:</label><br>
  <input type="password" id="pin" name="pin" maxlength="4" size="4"><br><br>
  <input type="submit" value="Submit">
</form>
```

https://www.w3schools.com/html/html_form_attributes.asp

Formularios
Controles de formulario

- Otros atributos generales para controles de formulario:
  - ***name***: Nombre del campo que se enviará al servidor junto con el valor (obligatorio)
  - ***value***: Valor que se enviará al servidor, por si queremos que sea diferente al texto mostrado en el campo (*checkbox* o listas)
- Otros elementos propios de los formularios:
  - ***label***: Asigna una etiqueta al campo para ayudar a su interpretación (p.ej. Lectores de pantalla)
  - ***select***: Lista desplegable de opciones (<*option*>)
  - ***textarea***: campo de texto con varias líneas
  - ***fieldset***: Agrupa un conjunto de elementos

```
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <input type="submit">
</form>
```



https://www.w3schools.com/html/html_form_elements.asp

Thymeleaf
Introducción

- Es un motor de plantillas Java para lado servidor que permite integrar datos dinámicos en HTML
- Módulo dentro de Spring Framework que se integra de forma sencilla con HTML 5
- Las plantillas en Thymeleaf siguen la estructura de las páginas HTML, incorporando etiquetas especiales para gestionar contenido
- Su motor puede procesar lenguajes como HTML, XML, JavaScript y CSS
- El "*Standard Dialect*" agrupa una serie de procesadores lógicos que permite aplicar acciones sobre artefactos como etiquetas, texto, etc.
- La mayoría de los procesadores actúan a nivel de atributo sobre las etiquetas HTML, por ejemplo, para asignar el valor de un campo de texto en base a una variable
- Es extensible por parte del desarrollador
- Debe indicarse junto a la etiqueta **html**

```
<!DOCTYPE html>
<html    xmlns="http://www.w3.org/1999/xhtml"
         xmlns:th="http://www.thymeleaf.org">
    <head>
```

https://www.thymeleaf.org/
https://www.thymeleaf.org/documentation.html
https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html

Thymeleaf
Introducción

- Funcionamiento básico: Externalización de texto
  - La etiqueta ***th:text=“”*** permite reemplazar el texto de una etiqueta HTML, por ejemplo, <p> por el valor entre comillas
  - La asignación puede corresponderse con una propiedad alojada en un fichero de propiedades. Para este caso, se utiliza al inicio el símbolo **#** seguido del nombre de la propiedad
    - Ejemplo: <p th:text=“#home.welcome> para adaptar un mensaje según el idioma detectado por el navegador
  - En nuestro contexto, el caso habitual será mostrar un objeto creado por el controlador (`ModelAndView`):
    - Se usa el símbolo **$** seguido de llaves para indicar que es una variable
    - Para acceder a las propiedades, utilizamos el punto (accede a *`get`*)

```
<body>
    <h1>Find Professor by ID</h1>
    <p th:text="${professor.name}">NULL</p>
    <p th:text="${professor.surname}">NULL</p>
    <p th:text="${professor.department}">NULL</p>
</body>
```


Find Professor by ID

Rodrigo

Redondo

Computer Science and AI

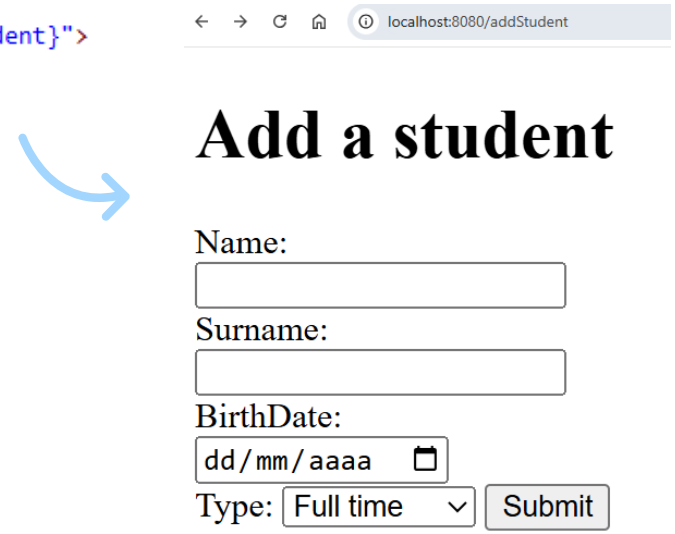
https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html#using-texts

Thymeleaf
Manejo de formularios

- La etiqueta ***th:action="@{}"*** realiza el mapeo al controlador mediante la URL indicada
- La etiqueta ***th:object="${}"*** indica el nombre del objeto donde se guardarán los valores del formulario
- La etiqueta ***th:field="*{}"*** asocia el valor del control del formulario a la propiedad del objeto `Student`

- Es importante elegir el tipo de "*input*" adecuado, ya que Spring realizará la conversión al dato Java automáticamente
- Como el constructor de `Student` necesita todos los campos, podemos usar el tipo "*hidden*" para evitar que el id (valor a asignar internamente) se muestre en el formulario

```
<h1>Add a student</h1>
<form method="POST" th:action="@{/addStudent}" th:object="${newStudent}">
    <input type="hidden" th:field="*{id}">
    <label for="name">Name:</label><br>
    <input type="text" th:field="*{name}"><br>
    <label for="surname">Surname:</label><br>
    <input type="text" th:field="*{surname}"><br>
    <label for="birthDate">BirthDate:</label><br>
    <input type="date" th:field="*{birthDate}"><br>
    <label for="type">Type:</label>
    <select th:field="*{type}">
        <option value="FULL_TIME">Full time</option>
        <option value="PARTIAL_TIME">Partial time</option>
        <option value="ERASMUS">Erasmus</option>
    </select>
    <input type="submit" value="Submit">
</form>
```



https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html#inputs

Thymeleaf
Ejemplo MVC

- Redefinición del controlador `AddStudentController`: /addStudent
  1. Si llega una petición **GET**:
     a. Crea el objeto vacío *newStudent*, para lo que añadimos el constructor vacío en `Student`
     b. Redirige a la vista addStudentView.html donde está el formulario para introducir los datos

```java
public class Student {

    private int id;
    private String name;
    private String surname;
    private LocalDate birthDate;
    private StudentType type;

    public Student(){

    }
```

```java
@GetMapping("/addStudent")
public ModelAndView getAddStudentView() {
    this.modelAndView.setViewName(viewName:"addStudentView.html");
    this.modelAndView.addObject(attributeName:"newStudent", new Student());
    return modelAndView;
}
```

Thymeleaf
Ejemplo MVC

En la mayoría de los casos tendremos un atributo identificador (p. ej. DNI) con el que comprobar la existencia de la tupla, tratando de insertarla directamente

- Redefinición del controlador `AddStudentController`: /addStudent

  2. Si llega una petición **POST**:

     a. Accede al repositorio para comprobar si existe un estudiante con el mismo nombre y apellido

     b. Si no se devuelve un ID válido, se calcula el ID que le corresponde (simulando un proceso auto-incremental) y se inserta en la base de datos a través del repositorio

     c. En caso contrario, no se inserta

     d. Se redirecciona a una vista según el resultado, manteniendo un objeto `Student` para poder acceder a su ID en la vista

```java
@PostMapping("/addStudent")
public ModelAndView addStudent(@ModelAttribute Student newStudent) {

    String nextPage;
    Integer idStudent = studentRepository.getStudentIdIfExists(newStudent.getName(), newStudent.getSurname());
    if(idStudent != -1){
        newStudent.setId(idStudent);
        nextPage = "addStudentViewFail.html";
    }

    else{
        int nextId = studentRepository.findAllStudents().size() + 1;
        newStudent.setId(nextId);
        boolean success = studentRepository.addStudent(newStudent);
        if(success){
            nextPage = "addStudentViewSucess.html";
        }
        else{
            nextPage = "/addStudentViewFail.html";
        }
    }
    this.modelAndView.setViewName(nextPage);
    this.modelAndView.addObject(attributeName:"student", newStudent);
    return this.modelAndView;
}
```

Thymeleaf
Ejemplo MVC

- Nuevo método en el repositorio `StudentRepository`:
  1. El método `getStudentIdIfExists` incorpora algunas novedades:
     a. Consulta de búsqueda con una cláusula WHERE basada en dos campos (nombre y apellido)
     b. La consulta se invoca mediante `queryForObject`: el resultado se mapea a un objeto (`Integer.class`)
     c. Si no se puede devolver exactamente un objeto (no hay coincidencias o hay más de una), salta la excepción: `IncorrectResultSizeDataAccessException`

```
public int getStudentIdIfExists(String name, String surname){
    try{
        String query = sqlQueries.getProperty(key:"select-findStudentByName");
        int idFound = jdbcTemplate.queryForObject(query, requiredType:Integer.class, name, surname);
        return idFound;
    }catch(IncorrectResultSizeDataAccessException | IncorrectResultSetColumnCountException resultException){
        // If no unique id is found
        return -1;
    }catch(DataAccessException queryException){
        System.err.println(x:"Unable to check student in the database");
        queryException.printStackTrace();
        return -1;
    }
}
```

Thymeleaf
Ejemplo MVC

- Según el resultado, se dirige a una vista u otra:
  1. Si la inserción ha sido correcta (addStudentViewSuccess.html), mostramos el ID asignado
  2. Si la inserción no ha sido correcta (addStudentViewFail.html), se muestra el ID encontrado
     - Para asegurarnos de que un objeto existe, podemos usar la etiqueta **th:if**
  3. En ambos casos, se permite volver al formulario enviando la solicitud **GET** (/addStudent)

```
<body>
    <h1>Add student: SUCCESS</h1>
    <p>Student successfully added to the database! :) The assigned ID is:</p>
    <p th:text="${student.id}"></p>

    <a href="/addStudent">Back to the form</a>
</body>
```

Add student: SUCCESS

Student successfully added to the database! :) The assigned ID is:

9

Back to the form

```
<body>
    <h1>Add student: ERROR :(</h1>
    <div th:if="${student != null}"></div>
    <p>Student already exists in the database. ID found:</p>
    <p th:text="${student.id}"></p>
    </div>
    <a href="/addStudent">Back to the form</a>
</body>
```

Add student: ERROR :(

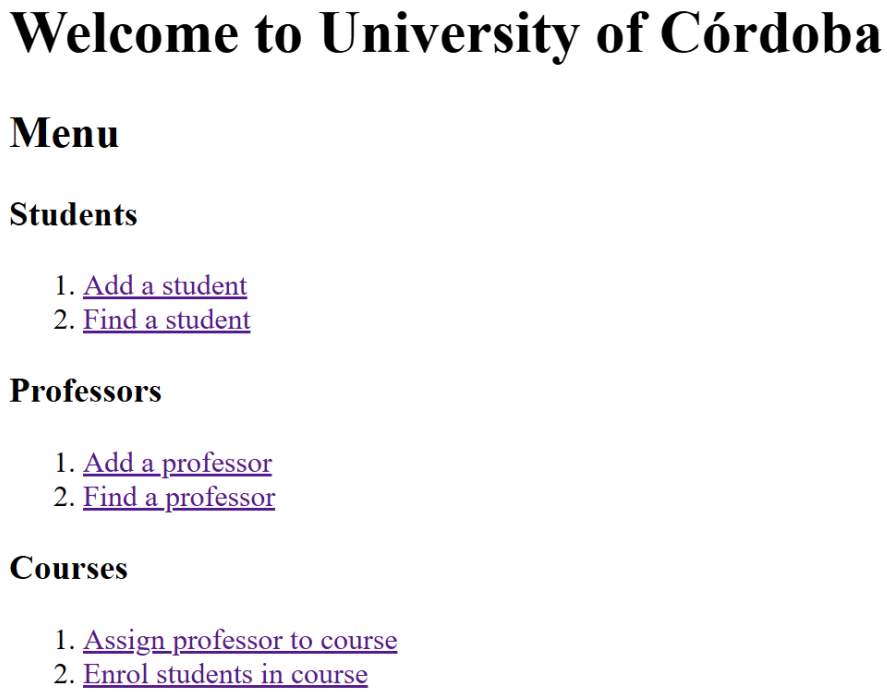
Student already exists in the database. ID found:

9

Back to the form

Thymeleaf
Ejemplo MVC

- Añadiendo menú y enlaces:
  - Se debe diseñar un mapa de navegación adecuado para la aplicación
  - Es importante enlazar correctamente las URL de vistas y controladores para evitar errores de mapeo
  - Permitir la recuperación ante errores del usuario (redireccionar a home, volver al formulario, etc.)



Objetivos de la semana

1. Replica el ejemplo explicado en clase, ejecutándolo sobre tu base de datos
   - Versión completa disponible en el repositorio Github: https://github.com/aramirez-uco/pw-examples
2. Incorpora las vistas necesarias para las funciones de **inserción** en base de datos
   - Define los formularios apropiados según los conceptos del dominio y sus restricciones
   - Conecta adecuadamente controladores y vistas
   - Añade menú y enlaces para acceder a las distintas funciones de forma más cómoda
   - Añade gestión de errores en los controladores y redireccionamiento a vistas según el resultado
3. Consulta la bibliografía recomendada:
   - *HTML forms:* https://www.w3schools.com/html/html_forms.asp
   - *Thymeleaf:* https://www.thymeleaf.org/
   - *Tutorial Thymeleaf + Spring:* https://www.thymeleaf.org/doc/tutorials/3.0/thymeleafspring.html