Informe final

Alumno/a: Marc Baulenas Gallego

Alumno/a: Genís Nin Ramírez

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Facultat d'Informàtica de Barcelona

FIB

Contenido

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Informe final prácticas 2 a 5

Contesta a las siguientes cuestiones referentes a las prácticas.

Práctica 2

1. Copia en el cuadro el código del servlet que recoge los datos del formulario para registrar una imagen, guardarlos en la base de datos y almacenar el fichero con la imagen en disco. Si el servlet llama a otras clases (DAO, etc.) no hace falta que copies el código, sólo indica el nombre de la(s) clase(s).

```
  protected void processRequest(HttpServletRequest request, HttpServletResponse
response)

        throws ServletException, IOException {


      String nombreF = new String();

      int aux2 = 1;

       for (Part part : request.getParts()) {

         logger.info(part.getName());

       InputStream is = request.getPart(part.getName()).getInputStream();

         int i = is.available();

         byte[] b = new byte[i];

         is.read(b);

         logger.info("Length : " + b.length);

         String fileName = getFileName(part);


         logger.info("File name : " + fileName);

         FileOutputStream os = new
FileOutputStream("/home/alumne/NetBeansProjects/lab2AD/src/main/webapp/imagenes/"
+ fileName); //


         if(aux2==1){

            nombreF = fileName;

            aux2=0;

         }
```

```
            os.write(b);
            is.close();
          }
        //response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            loginsql aux = new loginsql();

            HttpSession misesion = (HttpSession) request.getSession();
            String username = (String) misesion.getAttribute("username");


            String pattern = "dd/MM/yyyy";
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
            String date = simpleDateFormat.format(new Date());

            aux.AfegirEntrada(request.getParameter("titulo"),
request.getParameter("descripcion"), request.getParameter("keywords"),
request.getParameter("autor"), username , request.getParameter("fechaC"), date,
nombreF);
            //response.sendRedirect("menu.jsp");


            out.write("<h2> Imagen registrada correctamente</h2>");


            out.write("<a href='registrarImagen.jsp'>");
            out.write("<button>Registrar otra imagen</button></a>");

            out.write("<a href='menu.jsp'>");
            out.write("<button>Volver al menú</button></a>");
```

```
    }catch(Exception e){
        System.err.println(e.getMessage());
    }
  }

  private String getFileName(Part part) {
  String partHeader = part.getHeader("content-disposition");
  logger.info("Part Header = " + partHeader);
  for (String cd : part.getHeader("content-disposition").split(";")) {
   if (cd.trim().startsWith("filename")) {
    return cd.substring(cd.indexOf('=') + 1).trim()
        .replace("\"", "");
   }
  }
  return null;

}
```

2. Copia en el cuadro el código del formulario html que pide al usuario los datos de una imagen para registrarla.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Registrar Imagen</title>
  </head>
  <body>
    <div>
        <a href="menu.jsp">
        <button>Volver al menú</button>
         </a>
    </div>

    <div>Formulari image</div>

    <form action = "registrarImagen" method = "POST" enctype="multipart/form-data">
      <!-- Campos del formulario -->
      <div>
        <h3>File:</h3>
      <input type="file" name="file">
      </div>

      <div>
        <h3>Title:</h3>
      <input type="text" name="titulo">
      </div>

      <div>
```

```
            <h3>Description:</h3>
        <input type="text" name="descripcion">
        </div>

        <div>
            <h3>Keyword:</h3>
        <input type="text" name="keywords">
        </div>

        <div>
            <h3>Author:</h3>
        <input type="text" name="autor">
        </div>

        <div>
            <h3>Capture date(dd/MM/yyyy):</h3>
        <input type="text" name="fechaC">
        </div>

        <input type="submit">
      </form>
   </body>
</html>
```

Práctica 3

1. Copia en el cuadro la operación de registro de una imagen en SOAP.

```
private int RegisterImage(Image image){
    try { // Call Web Service Operation
        ClientServlet.ServeiWeb port = service.getServeiWebPort();

        // TODO process result here
        int result = port.registerImage(image);
        return result;

    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    return 0;

}
```

2. Copia en el cuadro la operación de búsqueda por título en SOAP.

```
private List SearchbyTitle(String title){

    try { // Call Web Service Operation
        ClientServlet.ServeiWeb port = service.getServeiWebPort();
        // TODO process result here
        List result = port.searchbyTitle(title);
        return result;
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
```

```
    return null;
}
```

3. Copia en el cuadro el código que llama a una de las operaciones del servicio web de imágenes en SOAP.

```
private List SearchbyTitle(String title){


    try { // Call Web Service Operation
        ClientServlet.ServeiWeb port = service.getServeiWebPort();
        // TODO process result here
        List result = port.searchbyTitle(title);
        return result;
    } catch (Exception ex) {
        // TODO handle custom exceptions here
    }
    return null;
}
```

Práctica 4

1. Copia en el cuadro la operación para modificar una imagen ya existente en REST.

```
/**
* POST method to modify an existing image
* @param title
* @param description
* @param keywords
* @param author
* @param creator
* @param capt_date
* @return
*/
@Path("modify")
@POST
```

```
@Consumes(MediaType.APPLICATION_FORM_URLENCODED)
@Produces(MediaType.TEXT_HTML)
public String modifyImage (@FormParam("id") String id,
@FormParam("title") String title,
@FormParam("description") String description,
@FormParam("keywords") String keywords,
@FormParam("author") String author,
@FormParam("creator") String creator,
@FormParam("capture") String capt_date){

    Image im= new Image();

    im.setId(Integer.parseInt(id));
    im.setTitulo(title);
    im.setDescripcion(description);
    im.setKeywords(keywords);
    im.setAutor(author);
    im.setCreator(creator);
    im.setFechaC(capt_date);

    String pattern = "yyyy-MM-dd";
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat(pattern);
    String date = simpleDateFormat.format(new Date());

    im.setFechaA(date);

    int aux = funcionsBD.ModifyImage(im);

    if(aux!=0) return "OK";
    else return "KO";
```

```
}
```

2. Copia en el cuadro la operación para buscar una imagen por palabra clave en REST.

```
/*** GET method to search images by keyword
 * @param keywords
 * @return
 */
@Path("searchKeywords/{keywords}")
@GET
@Produces(MediaType.APPLICATION_JSON)
public String searchByKeywords (@PathParam("keywords") String keywords){

    List<Image> list = funcionsBD.SearchbyKeywords(keywords);

    Gson gson = new Gson();

    String res = gson.toJson(list);

    return res;
}
```

3. Copia en el cuadro el código que llama a una de las operaciones del servicio web de imágenes en REST.

```
private Image SearchbyId(int id){
  try{

    String aux = "http://localhost:8080/lab4AD/resources/javaee8/searchID/"+id;

    URL url = new URL(aux);
```

```
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
          //conn.setRequestProperty("Accept","");

        InputStreamReader in = new
InputStreamReader(conn.getInputStream());
        BufferedReader br = new BufferedReader(in);
        Image im = new Image();
        String output;
        String aux2 = new String();

          if(conn.getResponseCode() == 200){

            while((output=br.readLine()) != null){
              if(output.length()>0){
                  aux2+=output;
               }
            }
            Gson gson = new Gson();
            im = gson.fromJson(aux2, Image.class);
          }
        conn.disconnect();
        return im;

    }catch(Exception e){

    }

    return null;
  }
```

Práctica 5

Compara los siguientes aspectos de la funcionalidad desarrollada en las prácticas 2, 3 y 4.

1. Facilidad de implementación de la parte cliente y la parte servidor.

   Es más fácil implementar la parte del servidor que la del cliente. En la parte del cliente es necesario crear todas la paginas necesarias para disponer de una navegabilidad correcta. Además para implementar el cliente se tiene que observar detenidamente la parte del servidor para ver que nos devuelve cada función y tratarla de forma correcta.

2. Tiempo de respuesta para la funcionalidad de registro de imagen. Para poder realizar la comparación, comenta la parte de upload de la página en la Práctica 2.

Pese a que se usaban dos métodos distintos, y estando en uno de ellos la funcionalidad fuera de la propia máquina, la diferencia de tiempo de respuesta era imperceptible.

3. Compara el formato de las peticiones y las respuestas en SOAP y REST. ¿Cómo se realiza el envío de objetos complejos como por ejemplo las listas en ambos servicios?

El envío de objetos complejos en rest se realiza mediante objetos JSON. Se puede hacer de otras maneras pero esta es la que nosotros hemos implementado. En cambio en SOAP se pueden enviar objetos complejos sin tener que usar objetos del tipo JSON.

4. Describe brevemente la implementación de upload / download en SOAP / REST, en caso de haberla realizado.

SOAP: El upload consiste en transformar la imagen recibida en el formulario, serializarla en Base64 y pasarla como parámetro juntamente con el nombre del fichero a la función upload del servicio web. El servicio web deserializa la imagen y la guarda con el nombre pasado como parámetro.

El download consiste en leer la imagen que tenemos guardada (averiguamos cuál es la imagen a descargar con un SearchById) usando un File. Serializar la imagen en base64 y enviarla al cliente. Este se encara de deserializarla i guardarla en un fichero.

REST: Para realizar el upload nos descargamos los bytes de la imagen y lo transformamos a String codificado en Base64 gracias a la libreria de java.util. Una vez dentro del servicio se descodifica y se escriben los bytes. El mecanismo de transformación es el mismo en el download, devolviendo la imagen adjuntada a una response.

5. Describe brevemente la integración cliente / servidor que hayas realizado con otro grupo de prácticas o entre distintos compañeros de un mismo grupo. En concreto, indica qué práctica(s) has integrado (2, 3 o 4) y con qué grupo(s).

Práctica 2 (Sergi y Isis): Simplemente consistía en conectarnos a la práctica de los compañeros desde nuestro navegador. Poniendo la dirección correcta de su proyecto en el navegador ya se podía usar. Lo mismo han hecho ellos con nuestra práctica.

Práctica 3 (Sergi y Isis): Ha sido la más difícil de integrar. Para que funcionara correctamente hemos tenido que crear un nuevo web service client mediante la URL del wsdl de los compañeros, hemos tenido que cambiar las declaraciones de los servicios para que llamara a su servicio web. Para terminar hemos tenido que importar los packages del servicio de los compañeros. Esta integración ha sido la que más problemas nos ha dado ya que hemos tenido que solucionar algunos errores producidos por las distintas versiones de jdk.

Practica 4 (Sergi y Isis): Hemos tenido que crear un cliente que pudiera usar los servicios de los compañeros. Ha sido difícil ya que ellos no usaban Gson para tratar los metadatos de la imagen como la clase Java «Image», así que hemos tenido que implementar un tratamiento para sus objetos JSON.

Todas las prácticas

1. Detalla las ampliaciones que hayas realizado en cada práctica. Algunos ejemplos de ampliaciones son: funcionalidades extra de gestión de imágenes, jsp para gestión de errores, funciones extra de búsqueda, etc. Puedes copiar el código correspondiente a cada ampliación.

Práctica 2:

La clase loginsql. Esta clase que implementamos en la practica 2 se encarga de todas las funciones que tienen que ver con la base de datos.

ErrorLogin.jsp: Página de error en caso de introducir incorrectamente las credenciales.

Práctica 3:

En el cliente de la práctica 3 añadimos:

errorUsuariOcupat.jsp: Página de error en caso de registrarse con un usuario que ya esta en uso.

En la parte del servidor:

La función de búsqueda multivalor. Permite buscar por más de un valor incluso todos si se desea.

La función comprovarCredencials que nos permite comprobar si las credenciales entrada en el momento del login son correctas o no.

La función AfegirUsuari que nos permite registrar nuevos usuarios.

Práctica 5:

La función FilenameCheck, que comprueba si el nombre del fichero esta en uso para asignarle uno nuevo.

Repositorios de código consultados

Lista de ejemplos de código consultados en github, gitlab, etc.

Bibliografía consultada

Lista de webs consultadas para la realización de la práctica que no sean repositorios de código.