Informe final

Alumno/a: Marcel Calleja

Alumno/a: Max Pasten

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, 3 y 4

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).

El servlet una la clase de dbConnection para hacer la conexión a la base de datos y registrar la imagen, también usa la clase constans que tiene la ruta donde se guardará la imagen

```java
import utils.dbConnection;
import utils.constants;
```

Este es el código del servlet:

```java
@MultipartConfig
@WebServlet(name = "registro_imagen", urlPatterns = {"/registro_imagen"})
public class registro_imagen extends HttpServlet {

  /**
     * Processes requests for both HTTP <code>GET</code> and
<code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */

  private static final long serialVersionUID = 1L;

  public registro_imagen() {
    super();
  }

  protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
      throws ServletException, IOException, ClassNotFoundException,
SQLException{
```

```
    // Get registro_imagen form parameters
    String title = request.getParameter("title");
    String description = request.getParameter("description");
    String keywords = request.getParameter("keywords");
    String author = request.getParameter("author");
    String creator = request.getParameter("creator");
    String capture_date = request.getParameter("capture_date");
    String storage_date = request.getParameter("storage_date");
    //String filename = request.getParameter("filename");

    System.out.println("Fecha: "+ storage_date);
    System.out.println("Creator:" + creator);


    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

      dbConnection db = new dbConnection();

      Part file=request.getPart("filename");

      // Nombre de Archivo
      String imageFileName=file.getSubmittedFileName(); // get selected
image file name

      out.println("Selected Image File Name : "+imageFileName);

      //String uploadPath="C:\\Users\\Max
Pasten\\lab1\\images\\"+imageFileName;
      String uploadPath=constants.IMAGESDIR+imageFileName;
      out.println("Upload Path : "+uploadPath);

      // Uploading our selected image into the images folder

      try{

        FileOutputStream fos=new FileOutputStream(uploadPath);
        InputStream is=file.getInputStream();

        byte[] data=new byte[is.available()];
        is.read(data);
        fos.write(data);
        fos.close();


        db.registerImage(title, description, keywords, author, creator,
capture_date, storage_date, imageFileName);

      }
```

```
    catch(Exception e){
      out.println("--ERROR-- : "+e);
      e.printStackTrace();
      response.sendRedirect("/lab1/error.jsp");
    }


    db.closeDb();

    response.sendRedirect("/lab1/menu.jsp");
    /* TODO output your page here. You may use following sample code. */



   } catch (SQLException ex) {
     Logger.getLogger(registro_imagen.class.getName()).log(Level.SEVERE,
null, ex);
   } catch (Exception e) {
     System.err.println(e.getMessage());
   }
  }
```

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

Este es el formulario que pide los datos al usuario para registrar una imagen, tenemos un script en javascript para mostrar una previsualización de la imagen que se esta registrando.

Java

```
<form action = "registro_imagen" method="POST"
enctype="multipart/form-data" onsubmit="return previewImage();>
                    <label for="titulo">Título:</label>
        <input type="text" id="title" name="title" required><br><br>

        <label for="descripcion">Descripción:</label><br>
        <textarea id="descripcion" name="description" rows="4" cols="50"
required></textarea><br><br>

        <label for="palabras_clave">Palabras Clave:</label>
        <input type="text" id="palabras_clave" name="keywords"
required><br><br>

        <label for="autor">Autor:</label>
        <input type="text" id="autor" name="author" required><br><br>
```

```
        <label for="creador">Creador:</label>
        <input type="text" id="creador" name="creator" value="${creator}"
readonly><br><br>

        <label for="fecha_captura">Fecha Captura:</label>
        <input type="date" id="fecha_captura" name="capture_date"
required><br><br>

        <label for="fecha_ingreso">Fecha Ingreso:</label>
        <input type="date" id="fecha_ingreso" name="storage_date"
value="${fechaActual}" readonly><br><br>

        <label for="nombre_imagen">Nombre Archivo:</label>
        <input type="file" id="imagen" name="filename" accept="image/*"
onchange="previewImage(event)">

        <br>
        <br>
        <input type="submit" value="Enviar">
        <input type="reset" value="Limpiar">

        <div id="imagePreview" style="display:none;">
          <h4>Imagen:</h4>
          <div align="center">
            <img id="preview" src="" alt="Image Preview" style="max-height:
300px;">
          </div>

        </div>
      </form>

    </div>
  </div>
  <script>
    function previewImage(event) {
      var input = event.target;
      var preview = document.getElementById('preview');
      var previewContainer = document.getElementById('imagePreview');

      var file = input.files[0];
      var reader = new FileReader();

      reader.onload = function() {
        preview.src = reader.result;
        previewContainer.style.display = 'block';
      }

      if (file) {
        reader.readAsDataURL(file);
```

```
      }
    }
  </script>
```

Práctica 3

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

Esta es nuestra operación de modificar una imagen, la operación usa la clase de conexión a la base de datos y envía los parámetros para modificar la imagen:

Java

```
/**
    * POST method to modify an existing image
    * @param id
    * @param title
    * @param description
    * @param keywords
    * @param author
    * @param creator, used for checking image ownership
    * @param capt_date
        * @param filename
    * @return
    return*/
 @Path("modify")
 @POST
 @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
 @Produces(MediaType.APPLICATION_JSON)
 public Response 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,
 @FormParam("filename") String filename)
 {
   String error;
   int code;

   try {

    db.modifyImage(Integer.parseInt(id), title, description, keywords,
author, capt_date, filename);
```

```
      return Response.ok().build();
      //}
    } catch (ClassNotFoundException | SQLException ex) {
       error="Error sentencia sql";
       code=502;
    }
    JsonObject json =
Json.createObjectBuilder().add("error",error).build();
    return Response.status(code).entity(json).build();
  }
```

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

Nuestra operación de búsqueda en una combinada que usa los campos de fecha (obligatorios) y los campos de autor y palabras clave (opcionales):

Java

```
@Path("searchCombined")
  @POST
  @Consumes(MediaType.APPLICATION_FORM_URLENCODED)
  @Produces(MediaType.APPLICATION_JSON)
  public Response searchCombined (@FormParam("date_ini") String date_ini,
     @FormParam("date_fin") String date_end,
     @FormParam("keywords") String keywords,
     @FormParam("author") String author){

     System.out.println("title=" + date_ini);
     System.out.println("description=" +date_end);
     System.out.println("keywords=" +keywords);
     System.out.println("author=" +author);


     List<Image> images = new ArrayList<>();
     try {

       if (author.length() == 0 && keywords.length() == 0){
         images = db.searchImageDate(date_ini, date_end);
       } else if (author.length() != 0 && keywords.length() != 0){
        images = db.searchImageDateAuthorKeywords(date_ini, date_end,
author, keywords);
       } else if (keywords.length() != 0 && author.length() == 0){
         images = db.searchImageDateKeywords(date_ini, date_end, keywords);
       } else if (keywords.length() == 0&& author.length() != 0){
         images = db.searchImageDateAuthor(date_ini, date_end, author);
```

```
        }


        JsonArrayBuilder jsonArrayBuilder = Json.createArrayBuilder();

        for (Image image : images) {
          jsonArrayBuilder.add(Json.createObjectBuilder()
              .add("id", image.getId())
              .add("title", image.getTitle())
              .add("description", image.getDescription())
              .add("keywords",
convertKeywordsToJsonArray(image.getKeywords()))
              .add("author", image.getAuthor())
              .add("creator", image.getCreator())
              .add("captureDate", image.getCaptureDate())
              .add("storageDate", image.getStorageDate())
              .add("filename", image.getFilename())
              .build());
        }

        JsonArray json = jsonArrayBuilder.build();
        System.out.println("ENVIA BUSQUEDA");
        return Response.ok().entity(json).build();

      } catch (SQLException ex) {
        return Response.status(Response.Status.BAD_GATEWAY).build();
      }

  }
```

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

Este es el código que llama a la operación para registrar una imagen, se encuentra en el servlet de registrar imagen:

```
Java
// Get registro_imagen form parameters
    String title = request.getParameter("title");
    String description = request.getParameter("description");
    String keywords = request.getParameter("keywords");
    String author = request.getParameter("author");
    String creator = request.getParameter("creator");
    String capture_date = request.getParameter("capture_date");
```

```
    String storage_date = request.getParameter("storage_date");
    Part file=request.getPart("filename");
    // Nombre de Archivo
    String filename=file.getSubmittedFileName();


    response.setContentType("text/html;charset=UTF-8");

    URL url = new
URL("http://localhost:8080/RestAD/resources/jakartaee9/register");
    // Conectar URL
    try {
      URLConnection myURLConnection = url.openConnection();
      myURLConnection.connect();
    }
    catch (MalformedURLException e) {
      // Fallo de URL
      Logger.getAnonymousLogger().log(Level.SEVERE,"URL error", e);
    }
    catch (IOException e){
      // fallo de la conexion
      Logger.getAnonymousLogger( ).log(Level.SEVERE, "I0 error",e);
    }

    //Abrir una conexión HTTP
    HttpURLConnection connection =(HttpURLConnection)url.openConnection();
    // Configurar el método de la petición
    connection.setDoOutput(true);
    connection.setRequestMethod("POST");

connection.setRequestProperty("Content-Type","application/x-www-form-url
encoded");
    // Escribir los parámetros
    try (OutputStream output = connection.getOutputStream()){
      // Construye la cadena de datos
      String data = "title=" + title +
            "&description=" +description +
            "&keywords=" +keywords +
            "&author=" +author +
            "&creator=" + creator+
            "&capture=" + capture_date +
            "&filename=" + filename;

      output.write(data.getBytes("UTF-8"));
      output.close();

      System.out.println("ENVIADO REGISTRO");
      // Recibe la respuesta del servidor
      int responsecode = connection.getResponseCode();
```

```
      if (responsecode == HttpURLConnection.HTTP_OK){
        // La conexión fue exitosa

        String uploadPath=constants.IMAGESDIR+filename;
        FileOutputStream fos=new FileOutputStream(uploadPath);
        InputStream is=file.getInputStream();

        byte[] dataImg=new byte[is.available()];
        is.read(dataImg);
        fos.write(dataImg);
        fos.close();


        // Redirect
        response.sendRedirect("/Client/menu.jsp");
      } else {
        response.sendRedirect("/Client/error.jsp");
      }
    } catch (Exception e) {
      e.printStackTrace();
      response.sendRedirect("/Client/error.jsp");
    }
```

Práctica 4

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

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

En la parte del cliente de la práctica 2 era más sencillo el hecho de conectarse a la base de datos, ya que en la práctica 3 se añadió la configuración de los recursos y la conexión por rest.

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

Implementando un controlador de tiempo en la práctica 2 y 4 para la función de registro tenemos los siguientes resultados:

Practica 2:
El tiempo de ejecución de registro fue de: 155 milisegundos.|#]
Práctica 4:
El tiempo de ejecución de registro fue de: 164 milisegundos.|#]

Como se observa la diferencia de tiempo de registro de una imagen es poca, se considera que el tanto como el servicio Rest y el cliente están en la misma máquina local, por lo que en este escenario no se contempla mucha diferencia respecto al tiempo, a diferencia de otro caso donde el servicio rest es estaría en otra máquina existirá aun mas diferencia de tiempo.

El código usado para medir el tiempo es el siguiente:

```java
long startTime = System.currentTimeMillis();

// Realiza la funcion de registro...

long endTime = System.currentTimeMillis();
long elapsedTime = endTime - startTime;
System.out.println("El tiempo de ejecución fue de: " + elapsedTime + "
milisegundos.");
```

3. ¿Cómo se realiza el envío de objetos complejos como por ejemplo las listas en REST? Pon un ejemplo.

Se envia mediante objetos json:

```
jsonArrayBuilder.add(Json.createObjectBuilder()
                    .add("id", image.getId())
                    .add("title", image.getTitle())
                    .add("description", image.getDescription())
                    .add("keywords", convertKeywordsToJsonArray(image.getKeywords()))
                    .add("author", image.getAuthor())
                    .add("creator", image.getCreator())
                    .add("captureDate", image.getCaptureDate())
                    .add("storageDate", image.getStorageDate())
                    .add("filename", image.getFilename())
                     .add("image", encodedString)
                    .build());
JsonArray json = jsonArrayBuilder.build();
```

Se añaden todos los campos del objeto complejo y se forma el json.

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

El download consiste en un endpoint que dado un filename produce un archivo el cual se descarga en una carpeta del cliente. El upload en cambio consiste en formar un objeto multipart form data con todos los campos de la imagen y el archivo y recibirlo en un endpoint mediante la configuración multipart, y guardar el archivo en una carpeta del servidor.

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 o 3) y con qué grupo(s).

Hemos integrado la aplicación Rest del grupo de Pol Crespi, concretamente las funcionalidades de login y listar imágenes.

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.

Hemos desarrollado las siguientes funcionalidades extra:

- Visualización de imágenes recientes: En el menú a parte de la redirección, se muestra una lista de imágenes además de sus datos que corresponden a las últimas imágenes registradas en la aplicación.

- Búsqueda de imágenes por fecha: Hemos añadido filtros combinados de búsqueda mediante fechas de inicio y fin, y otros atributos como el título de la imagen o las palabras clave.

Practica 4

En esta práctica implementamos la subida y bajada de imágenes a la práctica 3, dando la opción de descargar la imagen desde el lado del cliente.

También la integración de un nuevo cliente al servicio rest de otro Grupo

Repositorios de código consultados

https://github.com/JordiiBru/AD-FIB

Bibliografía consultada

https://stackoverflow.com/questions/8499633/how-to-display-base64-images-in-html

https://www.delftstack.com/es/howto/java/convert-json-to-java/