PONTIFICIA UNIVERSIDAD CATÓLICA MADRE Y MAESTRA
FACULTAD DE CIENCIAS DE LAS INGENIERÍAS
DEPARTAMENTO DE INGENIERÍA EN SISTEMAS Y COMPUTACIÓN

Aplicación Web – CRUD - JDBC

Materia: ISC-415.

Profesor: Carlos Camacho

Por
Ernesto Rodríguez

Matrícula:
2012-0201

Código en Github:
https://github.com/er12/Prog-Web/tree/master/Practica2

Santiago de los Caballeros, República Dominicana
Mayo 2016

Introducción

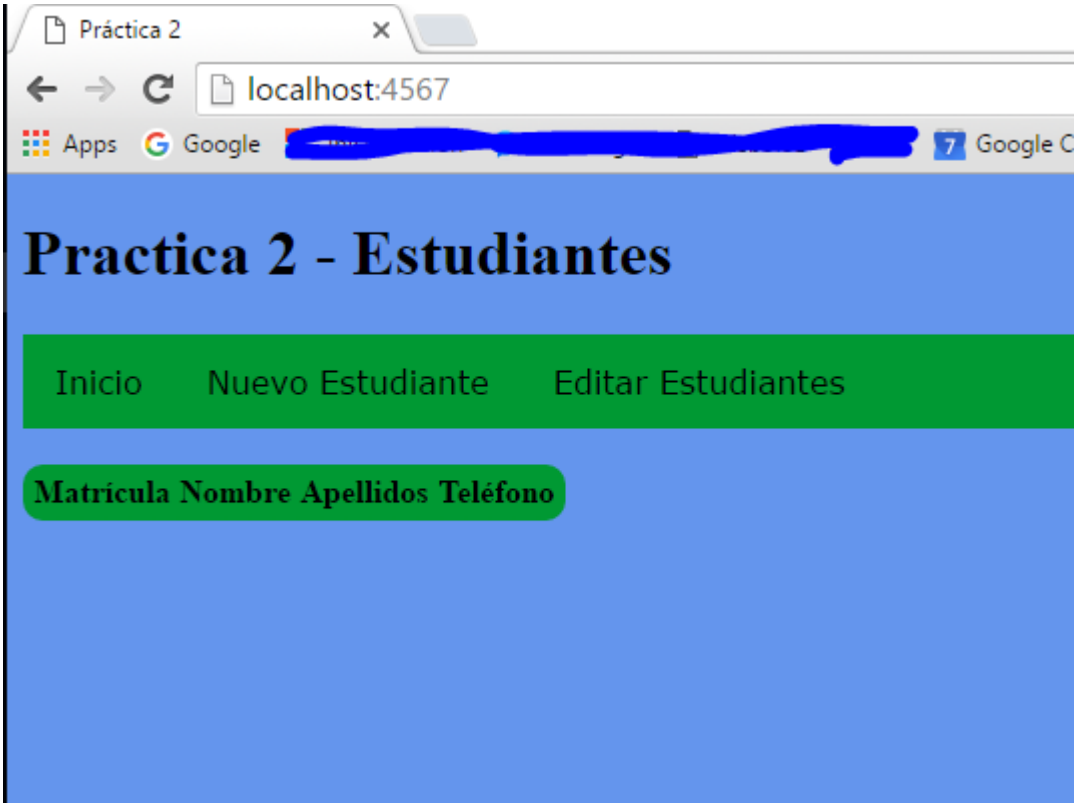
La práctica consiste en integrar cuatro grandes tecnologías para hacer un sistema simple de CRUD(Create Read Update Delete) de estudiantes. Las tecnologías fueron: SparkJava para el manejo de Servlets, Gradle para gestor de dependencias, FreeMarker para la vista y un poco de control del html y por último, H2, un manejador de bases de datos basado en Java para facilitaar la tarea. La interfaz será diseñada con la ayuda de cursos tomados en CodeAcademi.com.y blogs en internet.

Desarrollo

```
get("/", (request, response) -> {}
```

Es la expresión Lambda utilizada desde Java 7 para simplificar la forma de crear servlets web, ya en ella podemos consultar los parámetros request y alterar los atributos post que queramos de manera no muy compleja e intuitiva.

Para comenzar, tenemos nuestra página corriendo según la expresión Lambda especificada. SparkJava utiliza el puerto 4567 como "dominio" de la aplicación.

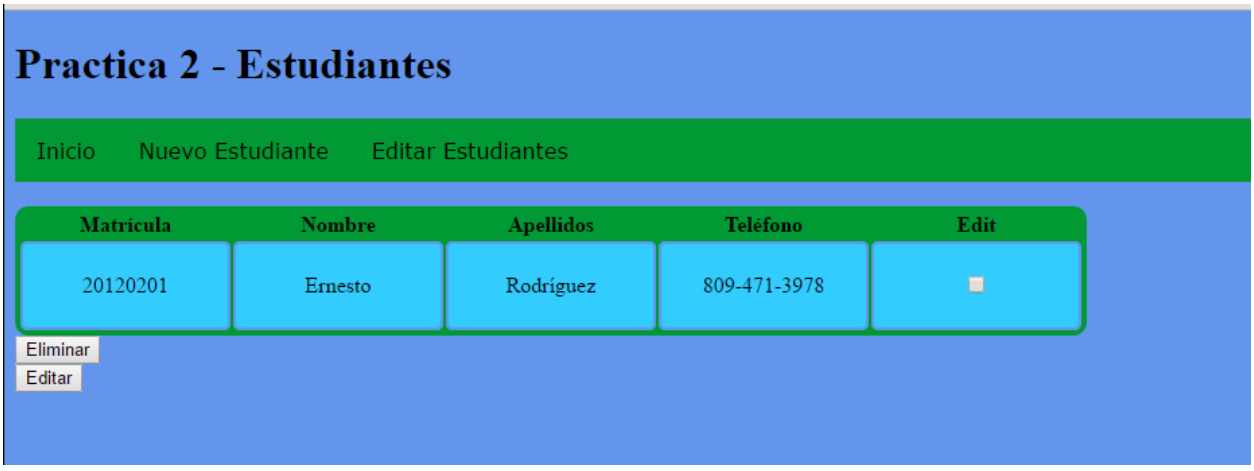


En un principio, la tabla está vacía, el código hace un "Select * from estudiantes" para llenar la tabla con el ciclo <#list l as x > </#list> , donde l es una lista y x es la variable iterativa.

Esta es la ventana de edición, la cual usa html forms para tomar los datos. Después l botón genera una acción Post al “/” para que inserte los la información a la base de datos H2 mediante JDBC.

Por último, se inserta el studiante y la tabla es refrescada.

La ventana de editar funciona seleccionando con el checkbox del estudiante a modificar.



Iteracion con Freemarker

```
</thead>
<tbody>
    <#list estudiantes as estudiante>
    <tr>
        <td>${estudiante.getMatricula()} </td>
        <td>${estudiante.getNombre()} </td>
        <td>${estudiante.getApellidos()} </td>
        <td>${estudiante.getTelefono()} </td>

    </tr>
    </#list>
```

Dependencias

```
dependencies {
    def freemarkerVersion = "2.3.21"

    compile group: 'com.h2database', name: 'h2', version: '1.4.192'

    compile group: 'com.sparkjava', name: 'spark-core', version: '2.5'
    compile("org.freemarker:freemarker:${freemarkerVersion}")
    testCompile group: 'junit', name: 'junit', version: '4.11'
    compile group: 'org.slf4j', name: 'slf4j-simple', version: '1.7.21'
    compile group: 'org.slf4j', name: 'slf4j-simple', version: '1.6.1'
```

JDBC

Se creó una clase Gestor para mantener todo el trámite de la base de datos en una sola clase.

```
public ArrayList<Estudiante> getAllStudents()
{
    ArrayList<Estudiante> estudiantes = new ArrayList<>();
    Connection conn = null;
    JdbcConnectionPool cp = JdbcConnectionPool.
            create("jdbc:h2:~/Pracica2", "sa", "");
    try {
        conn = cp.getConnection();
        Statement stmt = conn.createStatement();


        String sql = "SELECT * FROM ESTUDIANTES";
        ResultSet rs = stmt.executeQuery(sql);
        //STEP 5: Extract data from result set
        while(rs.next()){

            String id  = rs.getString("Matricula");
            String first = rs.getString("Nombre");
            String last = rs.getString("Apellidos");
```

Método post que recibe los datos insertados.

```
post("/", (request, response) -> {
    Map<String, Object> attributes = new HashMap<>();
    String mat = request.params("matricula");
    String nom = request.params("nombre");
    String apl = request.params("apellidos");
    String tel = request.params("telefono");
    System.out.println(mat +" "+nom+" "+apl+" "+tel);



    Estudiante e = new Estudiante(mat,nom,apl,tel);
    basedato.insertarEstudiante(e);
    attributes.put("estudiantes", basedato.getAllStudents());


    return new ModelAndView(attributes, "hello.ftl");
}, new FreeMarkerEngine());
```

Conclusión

Hubo muchos errores a partir del IDE y de parte de la configuración que debían de tener la bases de datos en modo de Gradle. Es recomendable buscar mas allá de la documentación de cada Framewirk.

A groso modo, fue tedioso pero no difícil, FreemMarker ayudó aentender como funcionan los controladores en la Web y Spark nos simuló un servidor para entender como funcionaban las acciones Get y Post.

El siclo de trabajo simuló una aplicación casi completa, haciendo de esta práctica una de las mas importantes en la carrera universitaria.

Bibliografía

http://www.h2database.com/html/features.html

http://www.stackoverflow.com

http://www.tutorialspoint.com/jdbc/jdbc-create-tables.htm

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