

La API JDBC

Clases DAO (Data Access Object)

El patrón arquitectónico Data Access Object (DAO), permite separar la capa de lógica de negocios de la capa de acceso a datos, de tal forma que el DAO encapsule toda la lógica de acceso de datos del resto de la aplicación.

Implementar la lógica de acceso a datos en la capa de lógica de negocio puede hacer el código complejo y no extensible. Se recomienda siempre usar DAO para abstraer y encapsular todos los accesos a los datos.

El DAO maneja la conexión con las fuentes de datos para obtener y almacenar datos.

La API JDBC

Clases DAO (Data Access Object)

Para ejemplificar un DAO, supongamos que tenemos las clases User y Message como parte del dominio de una aplicación. También tenemos una clase que recupera el DataSource de la aplicación.

```
package chatbot.model;

public class Message {
    private String contenido;
    private User user;
    private Data data;
    public Message(String contenido, User user, Data da
        ..... contenido = contenido; .....
package chatbot.dao;

import javax.naming.*; import javax.sql.DataSource;

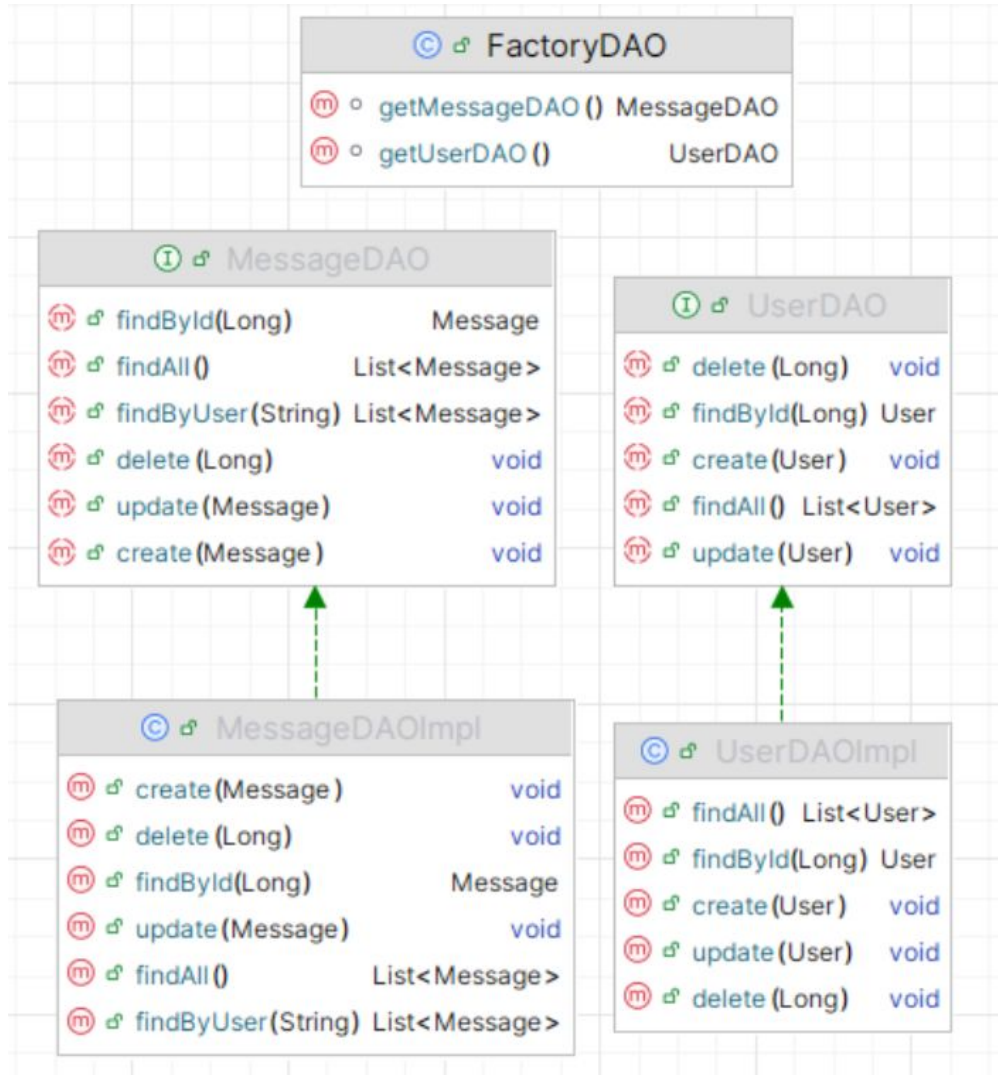
public class MyDataSource {
    private static DataSource dataSource = null;
    static {
        try {
            dataSource = (DataSource) new InitialContext().lookup("java:comp/env/jdbc/sbarra");
        } catch (NamingException e) {
            e.printStackTrace();
        }
    }
    public static DataSource getDataSource() {
        return dataSource;
    }
    private MyDataSource() {
    }
}
```

```
package chatbot.model;

public class User {
    private String email;
    private String password;
    private String nickname;
    private Profile profile;
    public User(String email, String password) {
        this.email = email;
    }
}
```

La API JDBC

Clases DAO



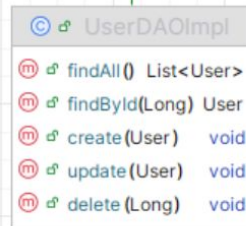
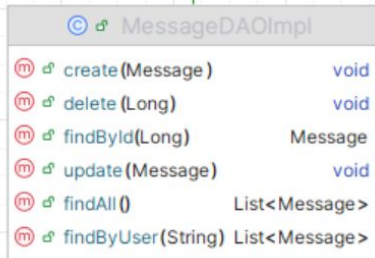
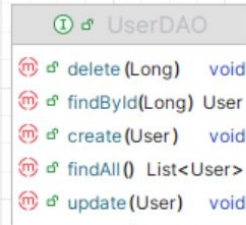
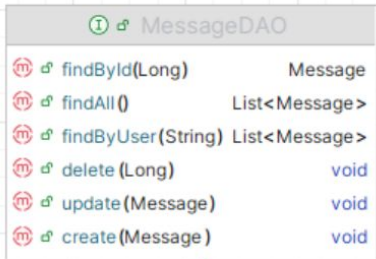
Esta clase crea objetos xxxDAO. Nos provee de objetos que implementan las distintas interfaces xxxDAO

Las interfaces xxxDAO tienen operaciones comunes de acceso a datos.

Implementaciones de las interfaces xxxDAO usando JDBC

La API JDBC

Clases DAO



```

public class FactoryDAO {
    public static UserDao getUserDAO() {
        return new UserDaoJdbc();
    }
    . . .
}
  
```

```
package dao.implJDBC;
```

```
public class UserDaoJdbc implements UserDao {
```

```
    public Usuario findById(Long id) {
```

```
        User usr = null;
```

```
        try{
```

```
            Connection con = MyDataSource.getDataSource().getConnection();
```

```
            Statement st = con.createStatement();
```

```
            ResultSet rs= st.executeQuery("Select u from User where
                u.id="+id);
```

```
            if (rs.next()==true) {
```

```
                usr = new User();
```

```
                usr.setProfile(rs.getProfile());
```

```
                usr.setEmail(rs.getEmail());
```

```
                // más setters
```

```
            }
```

```
            rs.close();
```

```
            st.close();
```

```
            con.close();
```

```
        } catch (java.sql.SQLException e) {
```

```
            System.out.println("Error de SQL: "+e.getMessage());
```

```
        }
```

```
        return usr;
```

```
    }
```

```
    public List<User> findAll() {...}
```

```
    public void create(User u) {...}
```

```
    public void update(User u) {...}
```

```
    public void delete(Long id) {...}
```

```
}
```

La API JDBC

Clases DAO

En una aplicación web tradicional, implementada con las componentes estándares de JEE y sin framework, los Servlets podrían implementar la **lógica de negocios**. Estos comúnmente acceden a la capa de datos a través de objetos DAO y utilizan método de la interface DAO para comunicarse con la base de datos.

```
public class ServletLogin extends HttpServlet {

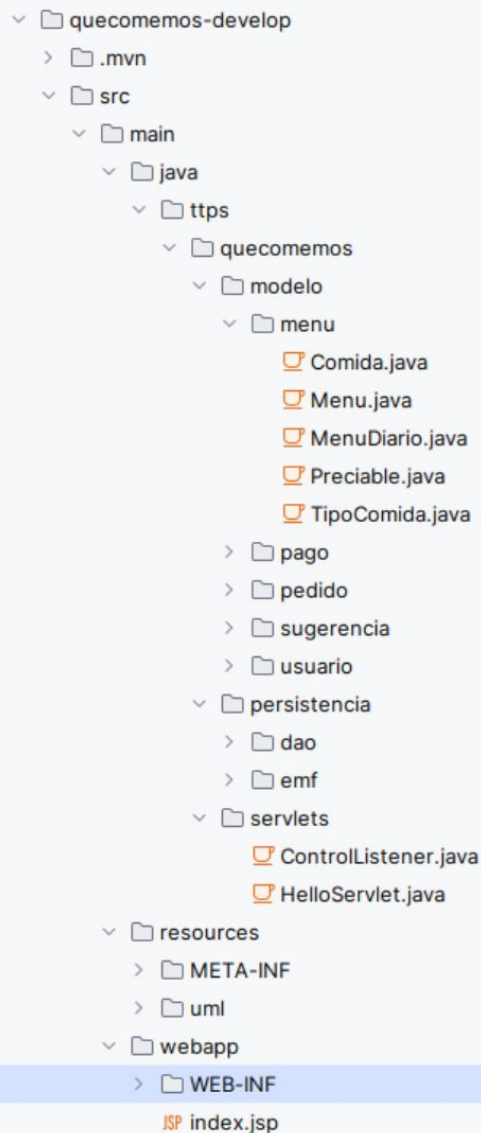
    public void doGet(HttpServletRequest request, HttpServletResponse response) {
        Perfil perfil = null;
        RequestDispatcher rd = null;
        UserDAO uDAO = FactoryDAO.getUserDAO();
        User u = uDAO.findById(Long.parseInt(request.getParameter("id")));
        if (u != null) {
            perfil=u.getProfile();
            HttpSession sesion = request.getSession();
            sesion.setAttribute("perfil", perfil);
            result = "/mostrarMenu";
        } else
            result = "/registracion.jsp";

        rd = getServletContext().getRequestDispatcher(result);
        rd.forward(request, response);

    }
}
```

La API JDBC

Organización de los datos



```
quecomemos-develop
├── .mvn
├── src
│   ├── main
│   │   ├── java
│   │   │   ├── ttps
│   │   │   │   ├── quecomemos
│   │   │   │   │   ├── modelo
│   │   │   │   │   │   ├── menu
│   │   │   │   │   │   │   ├── Comida.java
│   │   │   │   │   │   │   ├── Menu.java
│   │   │   │   │   │   │   ├── MenuDiario.java
│   │   │   │   │   │   │   ├── Preciable.java
│   │   │   │   │   │   │   ├── TipoComida.java
│   │   │   │   │   │   ├── pago
│   │   │   │   │   │   ├── pedido
│   │   │   │   │   │   ├── sugerencia
│   │   │   │   │   │   ├── usuario
│   │   │   │   │   ├── persistencia
│   │   │   │   │   │   ├── dao
│   │   │   │   │   │   ├── emf
│   │   │   │   │   ├── servlets
│   │   │   │   │   │   ├── ControlListener.java
│   │   │   │   │   │   ├── HelloServlet.java
│   │   ├── resources
│   │   │   ├── META-INF
│   │   │   ├── uml
│   │   ├── webapp
│   │   │   ├── WEB-INF
│   │   │   │   └── index.jsp
```

Agrupar los fuentes en paquetes:

- Paquete para las interfaces DAO
- Paquetes para las implementaciones de las interfaces DAO
- Pueden existir más de una implementación.
- Paquetes para los listeners.
- Paquetes para los Servlets (por ahora).