Veritabanında üç tablo olduğuna göre ve bu tablolardan veri transferi yapmayı planlıyorsanız, ihtiyacınız olan işlem API'leri oluşturmak olacaktır. Bu tür bir işlem için Java ve özellikle JSP (Java Server Pages) kullanarak, veritabanına bağlanabilir, verileri çekebilir, güncelleyebilir ve dışa aktarabilirsiniz.

Aşağıda, PostgreSQL veritabanınızdan verileri almak için kullanılabilecek örnek bir API yapısının temel adımlarını anlatacağım. Bu örnekte, PostgreSQL veritabanıyla etkileşime girecek olan Java JDBC kullanarak, **JSP** ile veritabanı işlemlerini gerçekleştireceğiz.

**1. JDBC Bağlantısı ve API Yapısı**

**Adımlar:**

* Veritabanına bağlantı kurmak için JDBC'yi kullanacağız.
* JSP sayfasında API'ye bağlanacak olan servletleri oluşturacağız.
* Tablolardaki verilere yönelik CRUD (Create, Read, Update, Delete) işlemleri gerçekleştireceğiz.

**2. PostgreSQL JDBC Bağlantısı**

İlk olarak, **JDBC** bağlantısını kurmamız gerekecek. PostgreSQL veritabanına bağlanmak için JDBC sürücüsüne ihtiyacımız var. Bu sürücüyü pom.xml dosyasına dahil etmeniz gerekecek (Maven kullanıyorsanız).

**2.1 pom.xml'ye PostgreSQL JDBC Bağlantısı Ekleyin:**

xml

Copy code

<dependencies>

<!-- PostgreSQL JDBC Driver -->

<dependency>

<groupId>org.postgresql</groupId>

<artifactId>postgresql</artifactId>

<version>42.5.0</version>

</dependency>

</dependencies>

**2.2 JDBC Bağlantı Kodunu Yazalım:**

PostgreSQL veritabanına bağlanmak için aşağıdaki sınıfı kullanabiliriz:

java

Copy code

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DatabaseConnection {

private static final String URL = "jdbc:postgresql://dpg-ct4cdttds78s73bhu7s0-a.frankfurt-postgres.render.com:5432/odev1";

private static final String USER = "sa";

private static final String PASSWORD = "qNgqTkPT8lnyPhO92vofBAiaQcHZpAsk";

public static Connection connect() throws SQLException {

try {

// PostgreSQL veritabanına bağlan

Connection connection = DriverManager.getConnection(URL, USER, PASSWORD);

return connection;

} catch (SQLException e) {

e.printStackTrace();

throw new SQLException("Veritabanına bağlanırken bir hata oluştu: " + e.getMessage());

}

}

}

**3. Tablo Verilerine Erişim (JSP ve Servlet ile)**

Veritabanından veri almak için JDBC kullanarak sorgular yapacağız. İlgili tablolar Users, Comment ve Messages ile veri alacağız. Her tablo için basit sorgu örnekleri:

**3.1 User Verilerini Çekmek İçin Sorgu Örneği:**

java

Copy code

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class UserDAO {

public List<User> getAllUsers() throws SQLException {

List<User> users = new ArrayList<>();

String query = "SELECT \* FROM public.\"Users\"";

try (Connection connection = DatabaseConnection.connect();

PreparedStatement statement = connection.prepareStatement(query);

ResultSet resultSet = statement.executeQuery()) {

while (resultSet.next()) {

User user = new User();

user.setId(resultSet.getString("Id"));

user.setEmail(resultSet.getString("Email"));

user.setName(resultSet.getString("Name"));

user.setGender(resultSet.getInt("Gender"));

user.setCity(resultSet.getString("City"));

user.setPassword(resultSet.getString("Password"));

user.setRole(resultSet.getString("Role"));

users.add(user);

}

}

return users;

}

}

**3.2 User Nesnesi:**

JDBC ile çekilen verileri taşımak için bir Java Bean oluşturacağız.

java

Copy code

public class User {

private String id;

private String email;

private String name;

private int gender;

private String city;

private String password;

private String role;

// Getter ve Setter metodları

}

**4. JSP Sayfası ile Veritabanı Sonuçlarını Görüntüleme:**

Veritabanından çektiğimiz verileri **JSP** sayfasında gösterebiliriz. Örnek olarak, kullanıcıların listelendiği bir JSP sayfası aşağıda verilmiştir.

**4.1 Users.jsp**

jsp

Copy code

<%@ page import="java.util.\*, java.sql.\*" %>

<%@ page import="com.example.UserDAO" %>

<%@ page import="com.example.User" %>

<%

UserDAO userDAO = new UserDAO();

List<User> users = userDAO.getAllUsers();

%>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Users List</title>

</head>

<body>

<h2>Users List</h2>

<table border="1">

<thead>

<tr>

<th>ID</th>

<th>Email</th>

<th>Name</th>

<th>Gender</th>

<th>City</th>

</tr>

</thead>

<tbody>

<% for (User user : users) { %>

<tr>

<td><%= user.getId() %></td>

<td><%= user.getEmail() %></td>

<td><%= user.getName() %></td>

<td><%= user.getGender() %></td>

<td><%= user.getCity() %></td>

</tr>

<% } %>

</tbody>

</table>

</body>

</html>

**5. API Endpoint Oluşturma:**

Eğer bir RESTful API üzerinden bu verileri almak istiyorsanız, Java'da Servlet veya Spring Boot gibi bir framework kullanabilirsiniz. Örneğin:

**5.1 Servlet Örneği (Get Kullanıcılar):**

java

Copy code

@WebServlet("/getUsers")

public class GetUsersServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

UserDAO userDAO = new UserDAO();

try {

List<User> users = userDAO.getAllUsers();

request.setAttribute("users", users);

request.getRequestDispatcher("/Users.jsp").forward(request, response);

} catch (SQLException e) {

e.printStackTrace();

response.getWriter().write("Veritabanına bağlanırken bir hata oluştu: " + e.getMessage());

}

}

}

**6. Veritabanı Tablo Yapılarına Göre API'yi Geliştirme**

Yukarıdaki örnek, sadece Users tablosunu çekmeye yönelik. Diğer tablolara (Comments ve Messages) yönelik de benzer işlemleri gerçekleştirebilirsiniz.

Her tablo için aynı mantıkla veri çekebilir ve API'lerinizi geliştirebilirsiniz. API'leri JSON formatında döndürmek istiyorsanız, Servlet veya Spring Boot ile JSON çıktısı döndürebilirsiniz.

Örneğin, Comments tablosu için:

java

Copy code

@WebServlet("/getComments")

public class GetCommentsServlet extends HttpServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

CommentDAO commentDAO = new CommentDAO();

try {

List<Comment> comments = commentDAO.getAllComments();

response.setContentType("application/json");

response.getWriter().write(new Gson().toJson(comments)); // Gson ile JSON'a çevirme

} catch (SQLException e) {

e.printStackTrace();

response.getWriter().write("Veritabanına bağlanırken bir hata oluştu: " + e.getMessage());

}

}

}

**7. Sonuç:**

Bu adımları takip ederek, JSP ile veritabanınızdaki verileri çekebilir ve API aracılığıyla dışa aktarabilirsiniz. Eğer API ile işlem yapmak istiyorsanız, Java'nın Servlet veya Spring Boot gibi framework'leri üzerinden bu işlemleri kolayca gerçekleştirebilirsiniz.