

Chapter-0: Introduction. DAO Pattern

Upcode Software
Engineer Team



CONTENT

1. DAO(Data Access Object) pattern
2. JDBCTemplate
3. DAO with example
4. Conclusion
5. Reference

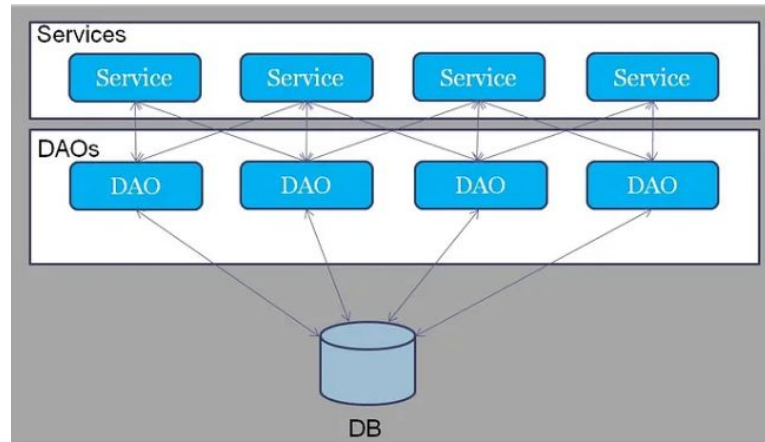
The DAO(Data Access Object) pattern(1/4)

- DAO (Data Access Object)
- In between the database and the business layer, there is a layer called the DAO layer.
- The DAO layer is mainly used to perform the **Create-Retrieve-Update-Delete (CRUD)** operation.

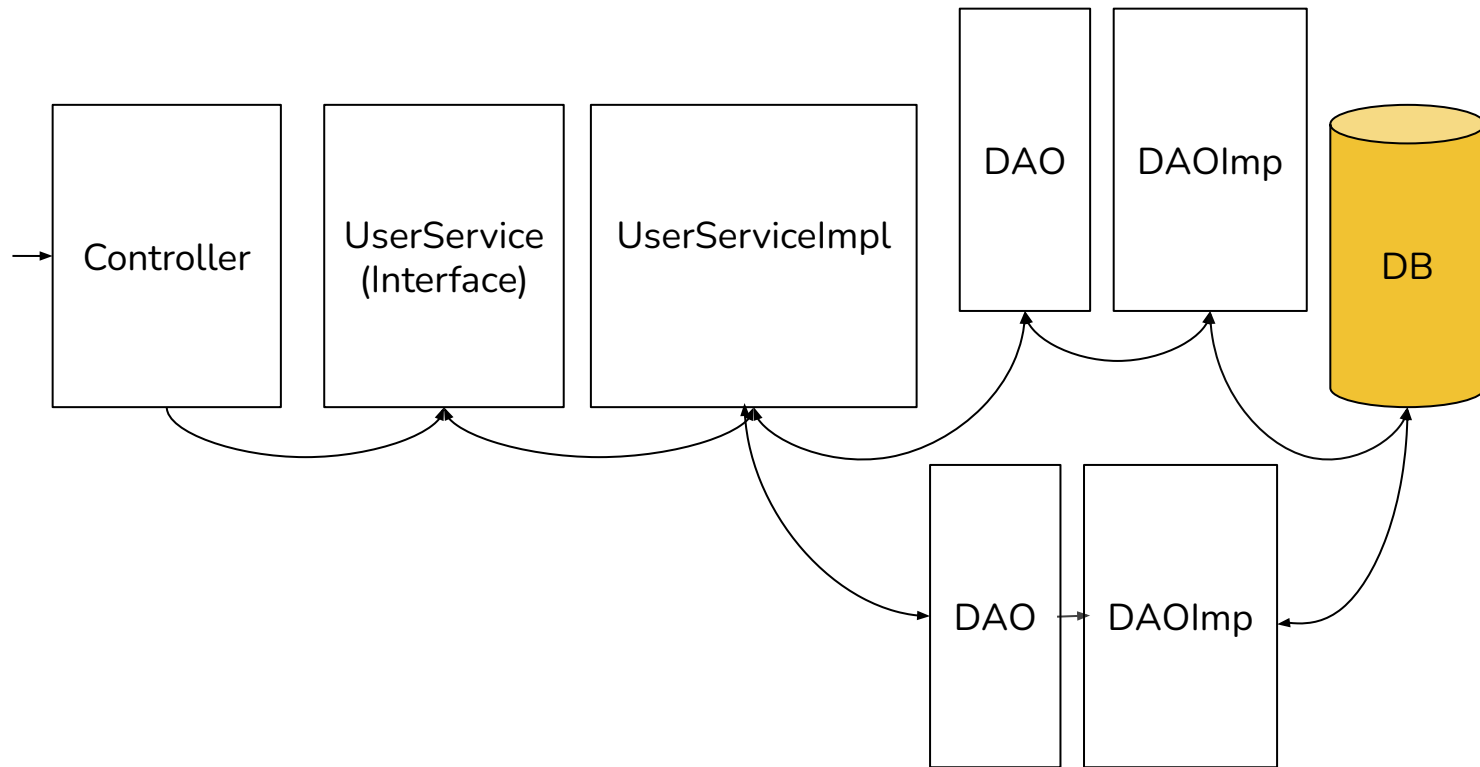


The DAO(Data Access Object) pattern(2/4)

- The DAO layer is responsible for creating, obtaining, updating, or deleting records in the database table.
- To perform this CRUD operation, DAO uses a low-level API, such as the JDBC API or the Hibernate API.



The DAO(Data Access Object) pattern(3/4)





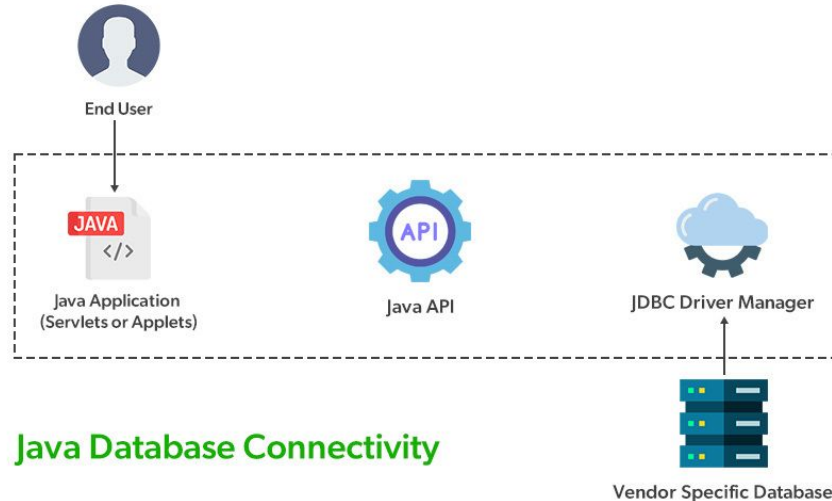
The DAO(Data Access Object) pattern (4/4)

The DAO pattern typically consists of the following components:

- **Entity:** Represents the data model or domain object that is persisted in the data storage system.
- **DAO Interface:** Defines the contract or interface that specifies the operations to be performed on the entity.
- **DAO Implementation:** Provides the actual implementation of the DAO interface, handling the low-level data access operations.
- **Service Layer:** Acts as an intermediary between the DAO layer and the business logic components, coordinating the interactions between them.

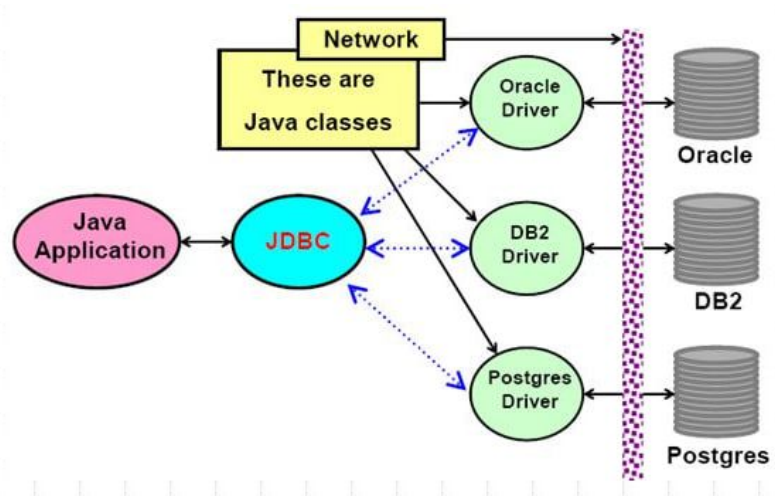
JdbcTemplate(1/2)

- **JdbcTemplate** is a class provided by the Spring Framework that simplifies JDBC (Java Database Connectivity) operations in Java applications.
- Using JdbcTemplate in a Spring application allows you to interact with a relational database in a straightforward manner.



JdbcTemplate(2/2)

- It handles resource creation and release, which helps avoid common mistakes such as forgetting to close a connection.
- It performs the basic tasks of the main JDBC workflow (such as creating and executing statements), leaving the application code to provide SQL and retrieve results. JdbcTemplate class:
- Executes SQL queries
- Updates statements and stored procedure calls



DAO with example(1/4)

Department JPA Entity

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Entity
@Table(name = "person")
public class User implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "username")
    private String username;
    @Column(name = "email")
    private String email;
}
```

```
public interface UserDAO
{
    void save(User user);
    List<User> findAll();
    User findById(Long id);
    void update(User user);
    void delete(Long id);
}
```

Create DAO Interface



DAO with example(2/4)

```
@Repository
public class UserDaoImpl implements UserDao {
    @Resource
    private JdbcTemplate jdbcTemplate;
    @Override
    public void save(User user) {
        String sql = "insert into \"person\" (username, email) VALUES (?, ?)";
        jdbcTemplate.update(sql, user.getUsername(), user.getEmail());
    }

    @Override
    public List<User> findAll() {
        String sql = "select * from user";
        return Collections.singletonList(jdbcTemplate.queryForObject(sql, new
Object[] {},
                                new BeanPropertyRowMapper<>(User.class)));
    }
}
```



DAO with example(3/4)

```
@Override
public User findById(Long id) {
    String sql = "SELECT * FROM person WHERE id = ? ";
    return jdbcTemplate.queryForObject(sql, new Object[]{id},
        (resultSet, i) -> new User(resultSet.getLong("id"),
            resultSet.getString("username"), resultSet.getString("email")));
}
```

```
@Override
public void update(User user) {
    String sql = "update person set username=?,password=? where id=? ";
    jdbcTemplate.update(sql, user.getUsername(), user.getEmail(), user.getId());
}
```

```
@Override
public void delete(Long id) {
    String sql = "delete from person where id = ? ";
    jdbcTemplate.update(sql, id);
}
```

DAO with example(4/4)

```
public interface UserService {  
    void save(User user);  
    void update(Long id);  
    void delete(Long id);  
    User getUserById(Long id);  
}
```

1

```
@Service  
public class UserService implements UserService {  
    @Autowired  
    private UserDAO userDAO;  
  
    public void save(User user) {  
        userDAO.save(user);  
    }  
    public User getUserById(Long id) {  
        return userDAO.findById(id);  
    }  
    public List<User> getAllUsers() {  
        return userDAO.findAll();  
    }  
    public void deleteUserById(Long id) {  
        userDAO.delete(id);  
    }  
}
```

2



REFERENCE

1: [Medium .org](#)

2. [Java Guides\(DAO\)](#)



Thank you!

Presented by

Tokhirjon Sadullaev

(tohirjonsadullayev387@gmail.com)