Spring Framework

10. 사용자 관리 프로젝트

Tacademy

CONTENTS

학습 목표

- 사용자 관리 프로젝트 아키텍쳐에 대하여 이해할 수 있습니다.
- 사용자 관리 프로젝트 클래스 설계에 대하여 이해할 수 있습니다.
- 사용자 관리 프로젝트 클래스 Code에 대하여 이해할 수 있습니다.

1. 사용자 관리 프로젝트 아키텍쳐

아키텍쳐 개요

- 대부분의 중·대규모 웹 애플리케이션은 효율적인 개발 및 유지보수를 위하여 **계층화(Layering)**하여 개발하는 것이 일반적이다.

- 사용자관리 프로젝트 아키텍쳐에서 기본적으로 가지는 계층은 **프리젠테이션 계층(Presentation Layer)**, **서비스 계층(Service Layer)**, **데이터액세스 계층(Data Access Layer)** 3계층과 모든 계층에서 사용되는 **도메인 모델 클래스**로 구성되어 있다.

- 각각의 계층은 계층마다 **독립적으로 분리하여 구현**하는 것이 가능해야 하며, 각 계층에서 담당해야 할 기능들이 있다.

아키텍쳐 개요

- 위의 세 가지 계층은 독립적으로 분리할 수 있도록 구현해야 하며, 일반적으로 각 계층 사이에서는 인터페이스(Interface)를 이용하여 통신하는 것이 일반적이다.

프리젠테이션 계층

- 브라우저상의 웹클라이언트의 요청 및 응답을 처리
- 상위계층(서비스계층, 데이터 액세스계층)에서 발생하는 Exception에 대한 처리
- 최종 UI에서 표현해야 할 도메인 모델을 사용
- 최종 UI에서 입력한 데이터에 대한 유효성 검증(Validation) 기능을 제공
- 비즈니스 로직과 최종 UI를 분리하기 위한 컨트롤러 기능을 제공
- @Controller 어노테이션을 사용하여 작성된 Controller 클래스가 이 계층에 속함

서비스 계층

- 애플리케이션 **비즈니스 로직 처리**와 비즈니스와 관련된 도메인 모델의 적합성 검증
- **트랜잭션(Transaction) 처리**
- 프리젠테이션 계층과 데이터 액세스 계층 사이를 연결하는 역할로서 두 계층이 직접적으로 통신하지 않게 하여 **애플리케이션의 유연성을 증가**
- 다른 계층들과 통신하기 위한 **인터페이스를 제공**
- Service 인터페이스와 @Service 어노테이션을 사용하여 작성된 Service 구현 클래스가 이 계층에 속함

데이터 액세스 계층

- 영구 저장소(관계형 데이터베이스)의 데이터를 조작하는 **데이터 액세스 로직을 객체화**
- 영구 저장소의 **데이터를 조회, 등록, 수정, 삭제** 함
- **ORM(Object Relational Mapping) 프레임워크**(MyBatis, Hibernate)를 주로 사용하는 계층
- DAO 인터페이스와 @Repository 어노테이션을 사용하여 작성된 DAO 구현 클래스가 이 계층에 속함

도메인 모델 클래스

- 관계형 데이터 베이스의 엔티티와 비슷한 개념을 가지는 것으로 실제 VO(Value Object) 혹은 DTO(Data Transfer Object) 객체에 해당
- 도메인 모델 클래스는 3개의 계층 전체에 걸쳐 사용
- private으로 선언된 멤버변수가 있고, 그 변수에 대한 getter와 setter 메서드를 가진 클래스를 말함

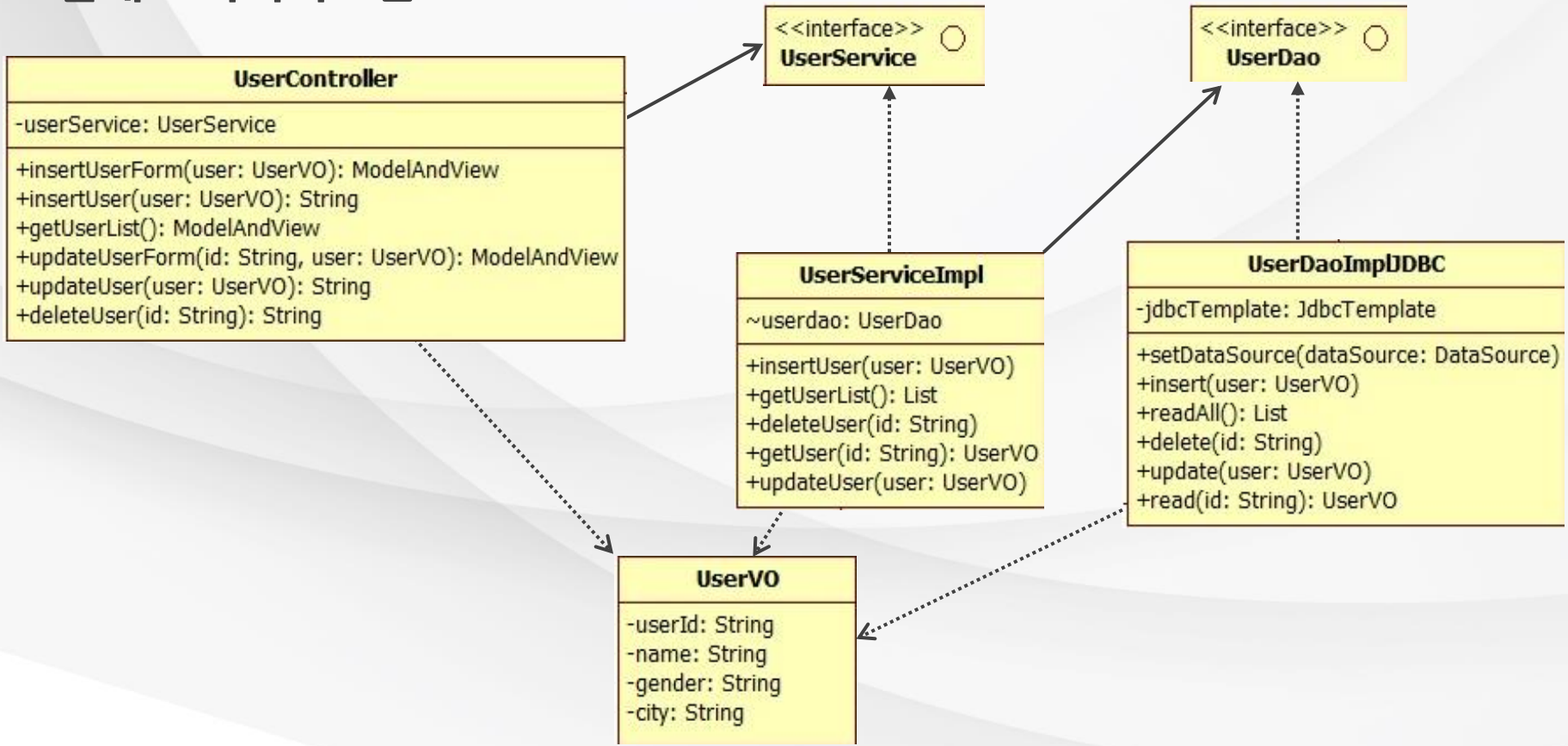
테이블 설계

```
CREATE TABLE USERS
(
    userid      VARCHAR2(30)  NOT NULL PRIMARY KEY,
    name        VARCHAR2(100) NOT NULL,
    gender      VARCHAR2(10),
    city        VARCHAR2(30)
);
```

userid : 아이디, name : 이름, gender : 성별, city : 도시명

2. 사용자 관리 프로젝트 클래스 설계

클래스 다이어그램


UserController
-userService: UserService
+insertUserForm(user: UserVO): ModelAndView
+insertUser(user: UserVO): String
+getUserList(): ModelAndView
+updateUserForm(id: String, user: UserVO): ModelAndView
+updateUser(user: UserVO): String
+deleteUser(id: String): String
<<interface>>
UserService
<<interface>>
UserDao
UserServiceImpl
~userdao: UserDao
+insertUser(user: UserVO)
+getUserList(): List
+deleteUser(id: String)
+getUser(id: String): UserVO
+updateUser(user: UserVO)
UserDaoImplJDBC
-jdbcTemplate: JdbcTemplate
+setDataSource(dataSource: DataSource)
+insert(user: UserVO)
+readAll(): List
+delete(id: String)
+update(user: UserVO)
+read(id: String): UserVO
UserVO
-userId: String
-name: String
-gender: String
-city: String

클래스의 역할

❖ 프리젠테이션 계층

UserController 클래스

- UI 계층과 서비스 계층을 연결하는 역할을 하는 클래스
- JSP에서 UserController를 통해서 서비스 계층의 UserService를 사용하게 된다.
- 서비스 계층의 UserService 인터페이스를 구현하나 객체를 IoC 컨테이너가 주입해준다.

클래스의 역할

❖ 서비스 계층

UserService 인터페이스

- 서비스 계층에 속한 상위 인터페이스

UserServiceImpl 클래스

- UserService 인터페이스를 구현한 클래스
- 복잡한 업무 로직이 있을 경우에는
  이 클래스에서 업무 로직을 구현하면 된다.
- 데이터 액세스 계층의 UserDao 인터페이스를 구현한 객체를
  IoC 컨테이너가 주입해준다.

클래스의 역할

❖데이터 액세스 계층

UserDao 인터페이스

- 데이터 액세스 계층에 속한 상위 인터페이스

UserDaoImplJDBC 클래스

- UserDao 인터페이스를 구현한 클래스로 이 클래스에서는 데이터 액세스 로직을 구현하면 된다.
- SpringJDBC를 사용하는 경우에는 DataSource를 IoC 컨테이너가 주입해준다.
- MyBatis를 사용하는 경우에는 SqlSession을 IoC 컨테이너가 주입해준다.

3. 사용자 관리 프로젝트 클래스 Code

3. 사용자 관리 프로젝트 클래스 Code

UserVO.java

*UserVO.java

```
1  package myspring.user.vo;
2
3  public class UserVO {
4
5      private String userId;
6      private String name;
7      private String gender;
8      private String city;
9
10     public UserVO() {}
11
12     public UserVO(String userId, String name, String gender, String city) {
13         this.userId = userId;
14         this.name = name;
15         this.gender = gender;
16         this.city = city;
17     }
18
19     public String getUserId() {
20         return userId;
21     }
22
23     public void setUserId(String userId) {
24         this.userId = userId;
25     }
```

UserService.java

UserService.java

```
1  package myspring.user.service;
2
3  import java.util.List;
6
7  public interface UserService {
8
9      public void insertUser(UserVO user);
10
11     public List<UserVO> getUserList();
12
13     public void deleteUser(String id);
14
15     public UserVO getUser(String id);
16
17     public void updateUser(UserVO user);
18 }
```

3. 사용자 관리 프로젝트 클래스 Code

UserServiceImpl.java

UserServiceImpl.java

```
1  package myspring.user.service;
2
3  import java.util.List;
12
13 @Service("userService")
14 public class UserServiceImpl implements UserService {
15
16     @Autowired
17     UserDao userdao;
18
19     @Override
20     public void insertUser(UserVO user) {
21         userdao.insert(user);
22     }
23
24     public List<UserVO> getUserList() {
25         return userdao.readAll();
26     }
```

UserDao.java

UserDao.java

```
1  package myspring.user.dao;
2
3  import java.util.List;
6
7  public interface UserDao {
8      public void insert(UserVO user);
9
10     public List<UserVO> readAll();
11
12     public void update(UserVO user);
13
14     public void delete(String id);
15
16     public UserVO read(String id);
17
18 }
```

3. 사용자 관리 프로젝트 클래스 Code

UserDaoImplJDBC.java

UserDaoImplJDBC.java

```
1  package myspring.user.dao;
2
3  import java.sql.ResultSet;
16
17 @Repository("userDao")
18 public class UserDaoImplJDBC implements UserDao {
19     private JdbcTemplate jdbcTemplate;
20
21     @Autowired
22     public void setDataSource(DataSource dataSource) {
23         this.jdbcTemplate = new JdbcTemplate(dataSource);
24     }
25
26     class UserMapper implements RowMapper<UserVO> {
27         public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
28             UserVO user = new UserVO();
29             user.setUserId(rs.getString("userid"));
30             user.setName(rs.getString("name"));
31             user.setGender(rs.getString("gender"));
32             user.setCity(rs.getString("city"));
33             return user;
```

학습정리

지금까지 [사용자 관리 프로젝트]에 대해서 살펴보았습니다.

사용자 관리 프로젝트 아키텍쳐

프리젠테이션 계층, 서비스 계층, 데이터액세스 계층, 도메인 클래스

사용자 관리 프로젝트 클래스 설계

클래스 다이어그램, 각 클래스들의 역할

사용자 관리 프로젝트 클래스 Code

각 클래스들의 Code 살펴보기