

TDD 실습

TDD 실습 #3
Test Dependency
Trip Service

Test Dependency

- Seam
 - 코드를 편집하지 않고도 프로그램의 동작을 변경할 수 있는 위치
 - 의존 관계를 제거하기 위한 단서
 - seam 의 동작을 대체하면 테스트할 때 의존 관계를 배제할 수 있다.
 - seam 이 활성화되는 지점에서는 어느 동작을 사용할 지 선택할 수 있다.
- Dependency 테스트하기
 - 오버라이딩 주입
 - 루프 조건 캡슐화
 - 에러 주입
 - 협업자(Collaborator) 대체
 - 무연산 클래스 사용 : 널 객체나 스텝을 테스트

Dependency 테스트 – 오버라이딩 주입

```
public class CustomSpreadSheet extends Spreadsheet{  
    public Spreadsheet buildMartSheet(Cell cell){  
        ...  
        recalculate(cell);  
        ...  
    }  
    private static void recalculate(Cell cell) {  
        ...  
    }  
    ...  
}
```

- 오버라이딩 주입을 사용한 테스트
- private static recalculate() 를 protected recalculate() 변경하고, 테스트 코드에서 서브클래스를 작성해 메소드 오버라이드하였다.
- 테스트 코드를 작성할 때 코드 수정을 최소화하면서 점진적으로 접근할 수 있는 방법

```
public class CustomSpreadSheet extends Spreadsheet{  
    public Spreadsheet buildMartSheet(Cell cell){  
        ...  
        recalculate(cell);  
        ...  
    }  
    protected void recalculate(Cell cell) {  
        ...  
    }  
    ...  
}
```

```
public class TestCustomSpreadsheet extends  
CustomSpreadsheet{  
    protected void recalculate(Cell cell){  
        ...  
    }  
}
```

Dependency 테스트 – 조건 캡슐화

```
while(underSetTemperature()){  
    keepHeating();  
}
```

- 무한히 계속되거나 종료 조건이 단순 조건문으로 나타내기 복잡한 경우 – 무한 루프의 형태
- 테스트하기 까다롭고 테스트가 무한히 실행될 가능성이 있다.
- 루프 조건을 캡슐화
- shouldContinueLoop() 를 테스트 용도로 세세하게 조절해 줄 수 있다.
- 0 번 수행하려면 false 리턴하거나, 정해진 횟수만큼 반복한 후 false 를 리턴하는 방법으로 테스트한다.

```
boolean shouldContinueLoop(){  
    return true;  
}
```

```
void infiniteLoop(){  
    while(shouldContinueLoop()){  
        //어떤 일을 수행  
        if(/*어떤 조건 */){  
            break;  
        }  
    }  
}
```

Dependency 테스트 – 에러 주입 1/2

```
public class NetRetriever{
    public NetRetriever(){
    }
    public Response retrieveResponseFor(Request request)
        throws RetrieveException{
        try {
            openConnection();
            return makeRequest(request);
        } catch (RemoteException re) {
            logError("Error making request", re);
            throw new RetrievalException(re);
        } finally {
            closeConnection();
        }
    }
}
```

- Network, DB 연결과 같은 형태의 코드
- 예외 시의 여러 결과를 검증하기 어렵다

Dependency 테스트 – 에러 주입 2/2

```
@Test(expected=RetrievalException.class) //JUnit4
public void testRetrievalResponseFor_Exception() throws Retrievalxception{
    NetRetriever sut = new NetRetriever(){
        @Override
        public Response makeRequest(Request request) throws RemoteException {
            throw new RemoteException();
        }
    };
    sut.retrieveResponseFor(null);
}
```

- JUnit4 기준 테스트 코드
- 메소드가 request 처리할 때 예외를 발생시키도록 오버라이딩.
- 오버라이딩하여 에러를 주입하면 테스트에 대한 가능성이 넓어진다.

Dependency 테스트 – 협업자(Collaborator) 대체 1/2

```
public class NetRetriever{
    private Connection connection;

    public NetRetriever(){
        this.connection = connection;
    }
    public Response retrieveResponseFor(Request request){
        throws RetrieveException{
        try {
            connection.open();
            return makeRequest(request);
        } catch (RemoteException re) {
            logError("Error making request", re);
            throw new RetrievalException(re);
        } finally {
            if(connection.isOpen()){
                connection.close();
            }
        }
    }
}
```

- 연결을 NetRetriever 와 분리

Dependency 테스트 – 협업자(Collaborator) 대체 2/2

```
@Test(expected=RetrievalException.class) //JUnit4
public void testRetrievalResponseFor_Exception() throws RetrievalException{
    Connection connection = new Connection(){
        @Override
        public void open() throws RemoteException {
            throw new RemoteException();
        }
    };
    NetRetriever sut = new NetRetriever(connection);
    sut.retrieveResponseFor(null);
}
```

- JUnit4 기준 테스트 코드
- NetRetriever 클래스 메소드의 오버라이딩 대신
협업자의 메소드를 오버라이딩해서 테스트에 에러 주입
- 캡슐화와 분업 설계를 향상시킴.

실습 #3 준비 – Trip Service

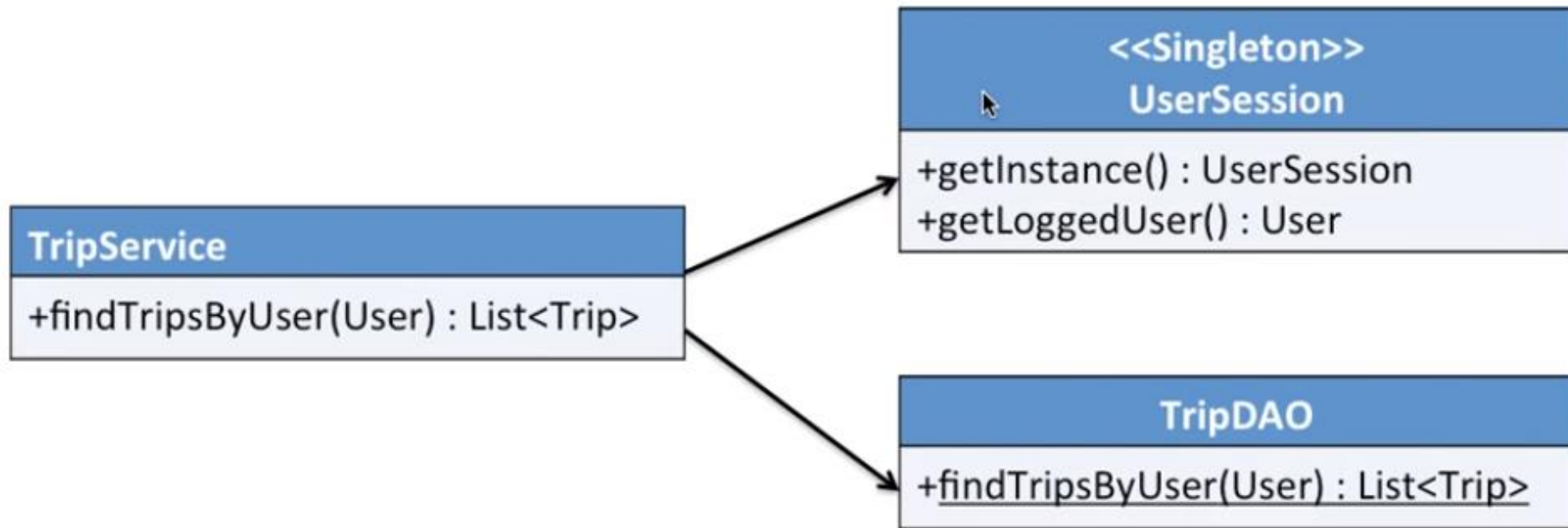
- 실습목적 : Dependency 를 갖는 Legacy code 의 테스트 코드 작성
- 실습내용 : 기존 코드의 dependency 를 분리하여 테스트 코드를 작성해 보세요.
- 삼성 GitHub Organization
 - Organization 이름 : Best-Reviewer-2-13
 - <https://github.ecodesamsung.com/Best-Reviewer-2-13>
 - Repository : TDD-TripService
- 조별, Pair별 실습 코드는 Org 내 Repo 생성, 실습 진행
 - Repository : 과정명_실습명_개인번호 (ex: TDD_TripService_01, TDD_TripService_02,...)
 - description : Author, Reviewer 성명 -> ex: Author : 고단수, Reviewer : 안단순

실습 #3 설명 – Trip Service

1. UserSession, TripDAO 를 getTripsByUser() 에서 적당한 수준으로 분리하여 TripService 에 대한 Test를 만들어 주세요.
2. getTripsByUser 의 signature 를 변경해 보세요.
3. TripDAO를 Mock 으로 구성하여 Test 를 변경해 보세요.

실습 #3 설명 – Trip service 클래스 관계도

Trip Service - Problems



실습 #3 설명 – TripService 의 테스트 작성



```
public List<Trip> getTripsByUser(User user) throws UserNotLoggedInException {
    List<Trip> tripList = new ArrayList<Trip>();
    User loggedInUser = UserSession.getInstance().getLoggedInUser();
    boolean isFriend = false;
    if (loggedInUser != null) {
        for (User friend : user.getFriends()) {
            if (friend.equals(loggedInUser)) {
                isFriend = true;
                break;
            }
        }
        if (isFriend) {
            tripList = TripDAO.findTripsByUser(user);
        }
        return tripList;
    } else {
        throw new UserNotLoggedInException();
    }
}
```

실습 #3 설명 – TripService 의 테스트 작성

- 로그인하지 않은 user 의 경우 exception 발생 테스트

```
@Test
void 익셉션_로그인하지않은유저() {
    TripService tripService = new TripService();
    assertThrows(UserNotLoggedInException.class, () -> {
        tripService.getTripsByUser(null);
    });
}
```

- UserSession 을 분리하기 위해 Extract Method

```
protected User getLoggedInUser() {
    return UserSession.getInstance().getLoggedInUser();
}
```

실습 #3 설명 – TripService 의 테스트 작성



```
public List<Trip> getTripsByUser(User user) throws UserNotLoggedInException {
    List<Trip> tripList = new ArrayList<Trip>();
    User loggedInUser = UserSession.getInstance().getLoggedInUser();
    boolean isFriend = false;
    if (loggedInUser != null) {
        for (User friend : user.getFriends()) {
            if (friend.equals(loggedInUser)) {
                isFriend = true;
                break;
            }
        }
        if (isFriend) {
            tripList = TripDAO.findTripsByUser(user);
        }
        return tripList;
    } else {
        throw new UserNotLoggedInException();
    }
}
```

실습 #3 설명 – TripService 의 테스트 작성

- 로그인한 user의 경우 테스트는 어떻게 할 것인가?
=> Override 주입

```
class TestableTripService extends TripService {  
    @Override  
    protected User getLoggedInUser() {  
        return null;  
    }  
}
```

실습 #3 설명 – TripService 의 테스트 작성

```
@Test
void 빈리스트_친구아닌_로그인유저() {
    loginUser = new User();
    User userWithNoFriend = new User();
    userWithNoFriend.addTrip(TO_BUSAN);

    TestableTripService tripService = new TestableTripService();
    List<Trip> trips = tripService.getTripsByUser(userWithNoFriend);

    assertEquals(0, trips.size());
}
```

```
class TestableTripService extends TripService {
    @Override
    protected User getLoggedInUser() {
        return loginUser;
    }
}
```


실습 #3 설명 – TripService 의 테스트 작성

- TripDAO 를 호출한 부분에 대해 테스트는 어떻게 할 것인가?
- Override method 는 어떻게?

< Class TestableTripService >

```
@Override
protected List<Trip> TripBy(User user) {
    return ??????????;
}
```

- Extract method and override for test
 - TripDAO.findTripsByUser(user);

< Class TripService >

```
protected List<Trip> TripBy(User user) {
    return TripDAO.findTripsByUser(user);
}
```

< Class TestableTripService >

```
@Override
protected List<Trip> TripBy(User user) {
    return new ArrayList<Trip>();
}
```

실습 #3 설명 – TripService 의 테스트 작성

- 친구 확인 루틴이 TripService Class 의 getTripByUser 에 있는 것보다는 User Class 에 있는 것이 더 좋아 보입니다.

< TripService Class getTripByUser >

```
for (User friend : user.getFriends()) {  
    if (friend.equals(loggedUser)) {  
        isFriend = true;  
        break;  
    }  
}
```



??????

< User Class ??? Method >

??????

실습 #3 설명 – TripService 의 테스트 작성

- getTripByUser 의 signature 를 변경하여 UserSession 을 분리
 - UserSession를 확인 책임

TripService 내부		TripService를 호출하는 UI
getTripsByUser(User user)	vs	getTripsByUser(User user, User loginUser)

-
- Test Double을 이용하여, TestableTripService 를 사용하지 않고, 테스트를 변경