8강 계좌생성하기

1. DTO만들기

```
package shop.mtcoding.bankapp.dto.account;

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class AccountSaveReqDto {
    private String number;
    private String password;
}
```

DTO만들때 조건

- FORM태그의 name 값확인
- 데이터베이스 타입 확인

2. 다운캐스팅 확인

```
Object javax.servlet.http.HttpSession.getAttribute(String name)
Returns the object bound with the
specified name in this session, or
null if no object is bound under
the name.
    User principal =  session.getAttribute(name: "principal");
```

이게 왜 오류가 나지?

마우스를 올려봤더니, 가장 오른쪽 상단에 있는게 리턴 타입이더라!!

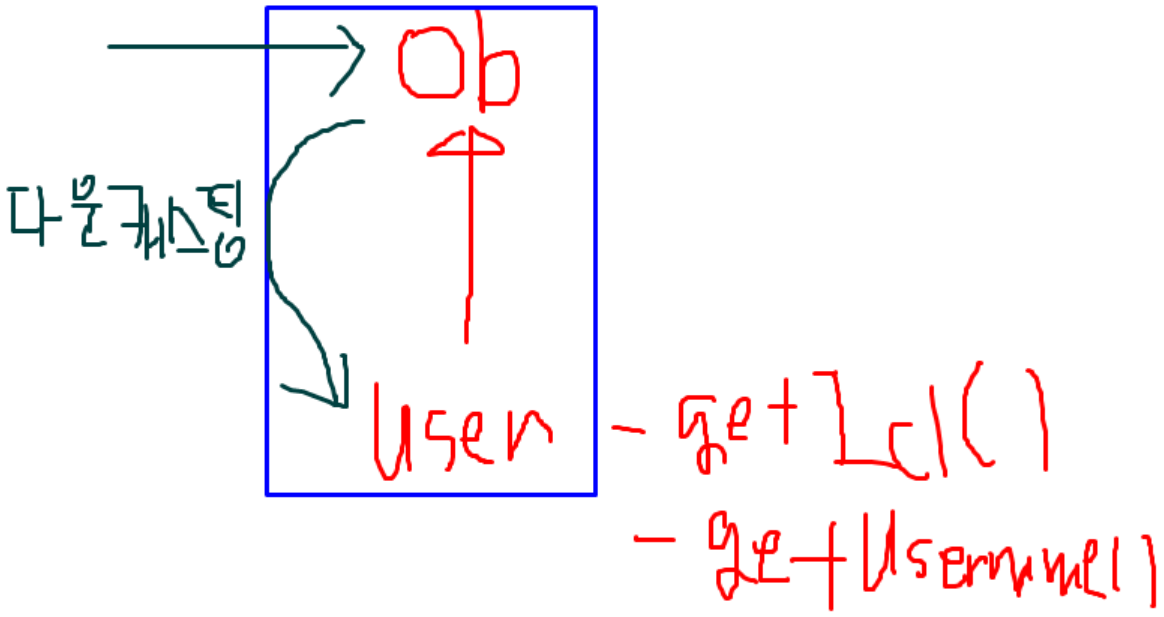
그래서 Object로 받아봤다.

```
    @PostMapping("/account")
    public String save(AccountSaveReqDto accountSaveReqDto){
        Object principal =  session.getAttribute(name: "principal");
```

```
public String save(AccountSaveReqDto accountSaveReqDto){
    Object principal =  session.getAttribute(name: "principal");
    principal.getUsername();
    principal.getId();
```

X 오류

값을 못 꺼내온다.



3. 계좌생성 컨트롤러 만들기

1. 계좌생성 컨트롤러

AccountController.java

```
@Autowired
private HttpSession session;

@PostMapping("/account")
public String save(AccountSaveReqDto accountSaveReqDto) {
    User principal = (User) session.getAttribute("principal");
    if (principal == null) {
        throw new CustomException("로그인을 먼저 해주세요", HttpStatus.UNAUTHORIZED);
    }
```

```
    if (accountSaveReqDto.getNumber() == null ||
accountSaveReqDto.getNumber().isEmpty()) {
        throw new CustomException("number를 입력해주세요", HttpStatus.BAD_REQUEST);
    }
    if (accountSaveReqDto.getPassword() == null ||
accountSaveReqDto.getPassword().isEmpty()) {
        throw new CustomException("password를 입력해주세요", HttpStatus.BAD_REQUEST);
    }

    // 서비스에 계좌생성() 호출
    return "redirect:/";
}
```

2. AccountSaveReqDto -> Account 모델로 변환

> 한번에 AccountRepository에게 데이터를 전달할려고!!
>
> 현재 전달해야 할 데이터 AccountSaveReqDto + principalId 를 전달해야 하기 때문에 하나의 오브젝트로 변경해서 전달

```
package shop.mtcoding.bankapp.dto.account;

import lombok.Getter;
import lombok.Setter;
import shop.mtcoding.bankapp.model.account.Account;

@Setter
@Getter
public class AccountSaveReqDto {
    private String number;
    private String password;

    // insert, update 할 때 이 메서드가 DTO에 필요하다.
    public Account toModel(int principalId) {
        Account account = new Account();
        account.setNumber(number);
        account.setPassword(password);
        account.setUserId(principalId);
        account.setBalance(1000L);
        return account;
    }
}
```

3. 계좌생성 서비스

AccountService.java

```
```

```
package shop.mtcoding.bankapp.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import shop.mtcoding.bankapp.dto.account.AccountSaveReqDto;
import shop.mtcoding.bankapp.model.account.Account;
import shop.mtcoding.bankapp.model.account.AccountRepository;

@Service
public class AccountService {

    @Autowired
    private AccountRepository accountRepository;

    @Transactional
    public void 계좌생성(AccountSaveReqDto accountSaveReqDto, int principalId) {
        Account account = accountSaveReqDto.toModel(principalId);
        accountRepository.insert(account);
    }
}
```

4. 회원가입 코드 리팩토링

4.1 JoinReqDto -> User 모델로 변환

```
package shop.mtcoding.bankapp.dto.user;

import lombok.Getter;
import lombok.Setter;
import shop.mtcoding.bankapp.model.user.User;

@Setter
@Getter
public class JoinReqDto {
    private String username;
    private String password;
    private String fullname;

    public User toModel() {
        User user = new User();
        user.setUsername(username);
        user.setPassword(password);
        user.setFullname(fullname);
        return user;
    }
```

```
}
```

4.2 UserRepository 수정

```
public int insert(User user);
```

4.3 UserService 의 회원가입 메서드 수정

```
int result = userRepository.insert(joinReqDto.toModel());
```

5. 로그인 코드 리팩토링 (할필요 없음)

> 왜냐하면 이 친구는 SELECT 하는 친구니까 (toModel은 Insert, Update 할 때만 변환시켜서 만들어주라더라!!)

6. AccountController 코드 수정

```
// ... 생략

@Autowired
private AccountService accountService;


@PostMapping("/account")
public String save(AccountSaveReqDto accountSaveReqDto) {
    // ... 생략

    accountService.계좌생성(accountSaveReqDto, principal.getId());

    return "redirect:/";
}
```

```
SELECT * FROM ACCOUNT_TB
```

```
SELECT * FROM ACCOUNT_TB;
```

ID	NUMBER	PASSWORD	BALANCE	USER_ID	CREATED_AT
1	1111	1234	1000	1	2023-02-16 12:35:30.888194
2	2222	1234	1000	2	2023-02-16 12:35:30.888194
3	5555	1234	1000	1	2023-02-16 12:38:45.839537

(3 rows, 1 ms)

7.정리

DTO로 받아서 Model로 변경해서 Repository에게 전달하자!! 명심하자