

12강 이체하기

1. DTO 만들기

```
<form action="/account/transfer" method="post">
    <input type="text" name="amount" placeholder="Enter 이체금" />
    <input type="text" name="wAccountNumber" placeholder="Enter 원금계좌번호" />
    <input type="text" name="dAccountNumber" placeholder="Enter 입금계좌번호" />
    <input type="text" name="wAccountPassword" placeholder="Enter 비밀번호" />
    <button>이체</button>
</form>
```

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

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class AccountTransferReqDto {
    private Long amount;
    private String wAccountNumber;
    private String dAccountNumber;
    private String wAccountPassword;
}
```

2. 컨트롤러 만들기

AccountController.java

```
@PostMapping("/account/transfer")
public String transfer(AccountTransferReqDto accountTransferReqDto) {
    // 1. 인증 필요
    User principal = (User) session.getAttribute("principal");
    if (principal == null) {
        throw new CustomException("로그인을 먼저 해주세요", HttpStatus.UNAUTHORIZED);
    }

    // 2. 유효성 검사
    if (accountTransferReqDto.getWAccountNumber().equals(accountTransferReqDto.getDAccountNumber())) {
        throw new CustomException("출금계좌와 입금계좌가 동일할 수 없습니다", HttpStatus.BAD_REQUEST);
    }
    if (accountTransferReqDto.getAmount() == null) {
```

```

        throw new CustomException("amount를 입력해주세요", HttpStatus.BAD_REQUEST);
    }
    if (accountTransferReqDto.getAmount().longValue() <= 0) {
        throw new CustomException("이체액이 0원 이하일 수 없습니다",
HttpStatus.BAD_REQUEST);
    }
    if (accountTransferReqDto.getWAccountNumber() == null ||
accountTransferReqDto.getWAccountNumber().isEmpty()) {
        throw new CustomException("출금 계좌번호를 입력해주세요",
HttpStatus.BAD_REQUEST);
    }
    if (accountTransferReqDto.getDAccountNumber() == null ||
accountTransferReqDto.getDAccountNumber().isEmpty()) {
        throw new CustomException("입금 계좌번호를 입력해주세요",
HttpStatus.BAD_REQUEST);
    }
    if (accountTransferReqDto.getWAccountPassword() == null
        || accountTransferReqDto.getWAccountPassword().isEmpty()) {
        throw new CustomException("출금 계좌비밀번호를 입력해주세요",
HttpStatus.BAD_REQUEST);
    }

    // 3. 서비스 호출
    int accountId = accountService.이체하기(accountTransferReqDto,
principal.getId());

    return "redirect:/account/" + accountId;
}

```

3. 서비스 로직 생각해보고, Account.java 코드 수정

- 출금 계좌존재 여부 확인
- 입금 계좌존재 여부 확인
- 출금 계좌패스워드 확인
- 출금 잔액확인
- 출금계좌 소유주 확인
- 출금
- 입금
- 거래내역 남기기

Account.java

```

public void checkOwner(Integer principalId) {
    if (userId != principalId) {
        throw new CustomException("계좌 소유자가 아닙니다", HttpStatus.FORBIDDEN);
    }
}

```

4. 서비스 만들기 (이체)

AccountService.java

```

@Transactional
public int 이체하기(AccountTransferReqDto accountTransferReqDto, Integer principalId)
{
    // 1. 출금 계좌존재 여부
    Account wAccountPS =
accountRepository.findByNumber(accountTransferReqDto.getWAccountNumber());
    if (wAccountPS == null) {
        throw new CustomException("출금 계좌가 없는데?", HttpStatus.BAD_REQUEST);
    }
    // 2. 입금 계좌존재 여부
    Account dAccountPS =
accountRepository.findByNumber(accountTransferReqDto.getDAccountNumber());
    if (dAccountPS == null) {
        throw new CustomException("입금 계좌가 없는데?", HttpStatus.BAD_REQUEST);
    }

    // 3. 출금 계좌패스워드 확인
    wAccountPS.checkPassword(accountTransferReqDto.getWAccountPassword());

    // 4. 출금 잔액확인
    wAccountPS.checkBalance(accountTransferReqDto.getAmount());

    // 5. 출금계좌 소유주 확인 (로그인 한 사람)
    wAccountPS.checkOwner(principalId);

    // 6. 출금
    wAccountPS.withdraw(accountTransferReqDto.getAmount());
    accountRepository.updateById(wAccountPS);

    // 7. 입금
    dAccountPS.deposit(accountTransferReqDto.getAmount());
    accountRepository.updateById(dAccountPS);

    // 8. 히스토리 (거래내역)
    History history = new History();
    history.setAmount(accountTransferReqDto.getAmount());
    history.setWAccountId(wAccountPS.getId());
    history.setDAccountId(dAccountPS.getId());
    history.setWBalance(wAccountPS.getBalance());
}

```

```
history.setDBalance(dAccountPS.getBalance());
historyRepository.insert(history);

// 9. 해당 계좌의 id를
return wAccountPS.getId();
} // 서비스 메서드 종료시에 커밋됩니다. 서비스 실행하다가 예외터지면 롤백
```

5. 실행 (테스트)

이체

localhost:8080 내용:

잔액이 부족한데?

이체

500

2222

3333

1234

이체

localhost:8080 내용:

계좌 소유자가 아닙니다

확인

이체

500

1111

1111

1234

이체

localhost:8080 내용:

출금계좌와 입금계좌가 동일할 수 없습니다

확인

이체

500

1111

2222

1234

이체

메인페이지

계좌번호	잔액
<u>1111</u>	400원
<u>3333</u>	1000원