11강 입금하기

1. DTO만들기

```
<h1>ATM 입금</h1>
<hr />
<form action="/account/deposit" method="post">
    <input type="text" name="amount" placeholder="Enter 입금금액" /><br />
    <input type="text" name="dAccountNumber" placeholder="Enter 입금계좌"
    <button>입금</button>
</form>
</body>
```

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

import lombok.Getter;
import lombok.Setter;

@Setter
@Getter
public class AccountDepositReqDto {
    // DTO는 똑같은게 존재해도 공유해서 쓰지 않는다.
    // 이유는 DTO는 화면에 나타나는 데이터 (자주 변경될 수 있음)
    private Long amount;
    private String dAccountNumber;
}
```

2. 컨트롤러 만들기

AccountController.java

```
 @PostMapping("/account/deposit")
public String deposit(AccountDepositReqDto accountDepositReqDto) {
    if (accountDepositReqDto.getAmount() == null) {
        throw new CustomException("amount를 입력해주세요", HttpStatus.BAD_REQUEST);
    }
    if (accountDepositReqDto.getAmount().longValue() <= 0) {
        throw new CustomException("입금액이 0원 이하일 수 없습니다",
HttpStatus.BAD_REQUEST);
    }
    if (accountDepositReqDto.getDAccountNumber() == null ||
accountDepositReqDto.getDAccountNumber().isEmpty()) {
```

```
        throw new CustomException("계좌번호를 입력해주세요", HttpStatus.BAD_REQUEST);
    }

    accountService.입금하기(accountDepositReqDto);
    return "redirect:/";
}
```

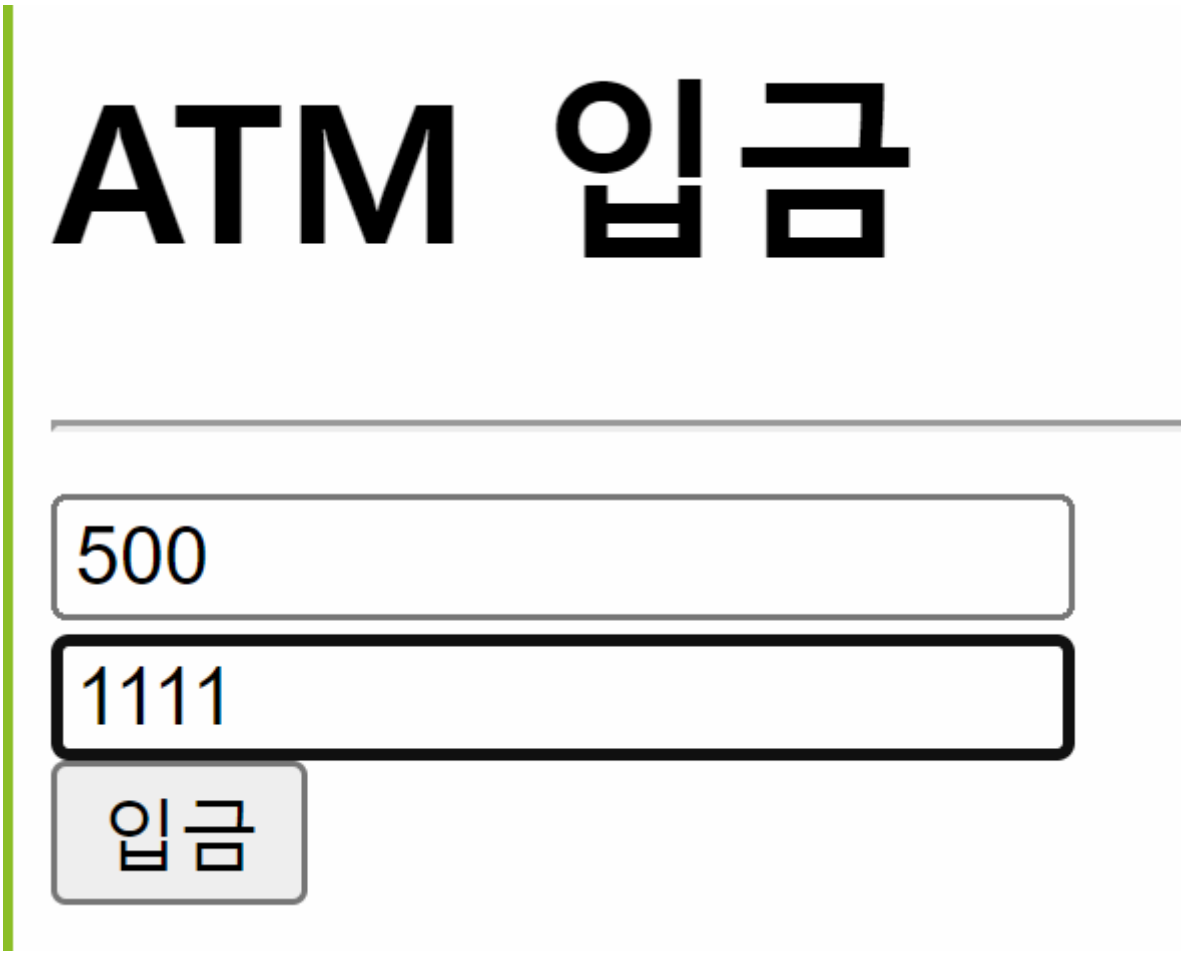
3. 서비스 만들기

AccountService.java

```
@Transactional
public void 입금하기(AccountDepositReqDto accountDepositReqDto) {
    // 1. 입금계좌 존재 여부
    Account accountPS =
accountRepository.findByNumber(accountDepositReqDto.getDAccountNumber());
    if (accountPS == null) {
        throw new CustomException("계좌가 없는데?", HttpStatus.BAD_REQUEST);
    }

    // 2. 입금하기 (의미 있는 메서드를 호출)
    accountPS.deposit(accountDepositReqDto.getAmount()); // 모델에 상태 변경
    accountRepository.updateById(accountPS); // 디비에 commit

    // 3. 입금 트랜잭션 만들기 (히스토리)
    History history = new History();
    history.setAmount(accountDepositReqDto.getAmount());
    history.setWAccountId(null);
    history.setDAccountId(accountPS.getId());
    history.setWBalance(null);
    history.setDBalance(accountPS.getBalance());

    historyRepository.insert(history);
}
```

4. 더미 데이터 트랜잭션 히스토리 동기화 시키기

```
INSERT INTO user_tb(username, password, fullname, created_at) values('ssar', '1234',
'쌀', now());
INSERT INTO user_tb(username, password, fullname, created_at) values('cos', '1234',
'코스', now());
INSERT INTO account_tb(number, password, balance, user_id, created_at)
values('1111', '1234', 900, 1, now());
INSERT INTO account_tb(number, password, balance, user_id, created_at)
values('2222', '1234', 1100, 2, now());
INSERT INTO account_tb(number, password, balance, user_id, created_at)
values('3333', '1234', 1000, 1, now());

INSERT INTO history_tb(amount, w_balance, d_balance, w_account_id, d_account_id,
created_at) values(100, 900, 1100, 1, 2, now());
INSERT INTO history_tb(amount, w_balance, d_balance, w_account_id, d_account_id,
created_at) values(100, 800, null, 1, null, now());
INSERT INTO history_tb(amount, w_balance, d_balance, w_account_id, d_account_id,
created_at) values(100, null, 900, null, 1, now());

commit;
```



메인페이지

계좌번호	잔액
1111	1400원
3333	1000원