10강 출금하기

1. DTO 만들기

```
<h1>ATM 출금</h1>
<hr />
<form action="/account/withdraw" method="post">
    <input type="text" name="amount" placeholder="Enter 출금금액" /><br />
    <input type="text" name="wAccountNumber" placeholder="Enter 출금계좌번호" /><br
    <input type="password" name="wAccountPassword" placeholder="Enter 출금계좌비밀번
    <button>출금</button>
</form>
</body>
```

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

import lombok.Getter;
import lombok.Setter;

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

2. 컨트롤러 만들기

AccountController.java

```
@PostMapping("/account/withdraw")
public String withdraw(AccountWithdrawReqDto accountWithdrawReqDto) {
    if (accountWithdrawReqDto.getAmount() == null) {
        throw new CustomException("amount를 입력해주세요", HttpStatus.BAD_REQUEST);
    }
    if (accountWithdrawReqDto.getAmount().longValue() <= 0) {
        throw new CustomException("출금액이 0원 이하일 수 없습니다",
HttpStatus.BAD_REQUEST);
    }
    if (accountWithdrawReqDto.getWAccountNumber() == null ||
accountWithdrawReqDto.getWAccountNumber().isEmpty()) {
        throw new CustomException("계좌번호를 입력해주세요", HttpStatus.BAD_REQUEST);
    }
    if (accountWithdrawReqDto.getWAccountPassword() == null
```

```
            || accountWithdrawReqDto.getWAccountPassword().isEmpty()) {
        throw new CustomException("계좌비밀번호를 입력해주세요",
HttpStatus.BAD_REQUEST);
    }

    int accountId = accountService.계좌출금(accountWithdrawReqDto);

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

3. 서비스에서 계좌출금시 해야할 일

- 계좌존재 여부
- 계좌 패스워드 확인
- 잔액 확인
- 출금
- 히스토리 (거래내역)
- 해당 계좌 ID 리턴

4. 서비스 만들기

account.xml

```
<select id="findByNumber" resultType="shop.mtcoding.bankapp.model.account.Account">
    SELECT * FROM ACCOUNT_TB WHERE number = #{number}
</select>
```

AccountRepository.java

```
public Account findByNumber(String number);
```

AccountService.java

```
@Autowired
private HistoryRepository historyRepository;

@Transactional
public int 계좌출금(AccountWithdrawReqDto accountWithdrawReqDto) {
    // 1. 계좌존재 여부
    Account accountPS =
accountRepository.findByNumber(accountWithdrawReqDto.getWAccountNumber());
    if (accountPS == null) {
        throw new CustomException("계좌가 없는데?", HttpStatus.BAD_REQUEST);
```

```
    }

    // 2. 계좌패스워드 확인
    if (!accountPS.getNumber().equals(accountWithdrawReqDto.getWAccountPassword()))
{
        throw new CustomException("출금계좌 비밀번호 틀렸는데?",
HttpStatus.BAD_REQUEST);
    }

    // 3. 잔액확인
    if (accountPS.getBalance() < accountWithdrawReqDto.getAmount()) {
        throw new CustomException("잔액이 부족한데?", HttpStatus.BAD_REQUEST);
    }

    // 4. 출금(balance - 마이너스)
    accountPS.setBalance(accountPS.getBalance() -
accountWithdrawReqDto.getAmount());
    accountRepository.updateById(accountPS);

    // 5. 히스토리 (거래내역)
    History history = new History();
    history.setAmount(accountWithdrawReqDto.getAmount());
    history.setWAccountId(accountPS.getId());
    history.setDAccountId(null);
    history.setWBalance(accountPS.getBalance());
    history.setDBalance(null);

    historyRepository.insert(history);

    // 6. 해당 계좌의 id를 return
    return accountPS.getId();
}
```

5. 서비스 코드 리팩토링

Account.java

```
@Transactional
public int 계좌출금(AccountWithdrawReqDto accountWithdrawReqDto) {
    // 1. 계좌존재 여부
    Account accountPS =
accountRepository.findByNumber(accountWithdrawReqDto.getWAccountNumber());
    if (accountPS == null) {
        throw new CustomException("계좌가 없는데?", HttpStatus.BAD_REQUEST);
    }

    // 2. 계좌패스워드 확인
    accountPS.checkPassword(accountWithdrawReqDto.getWAccountPassword());
```

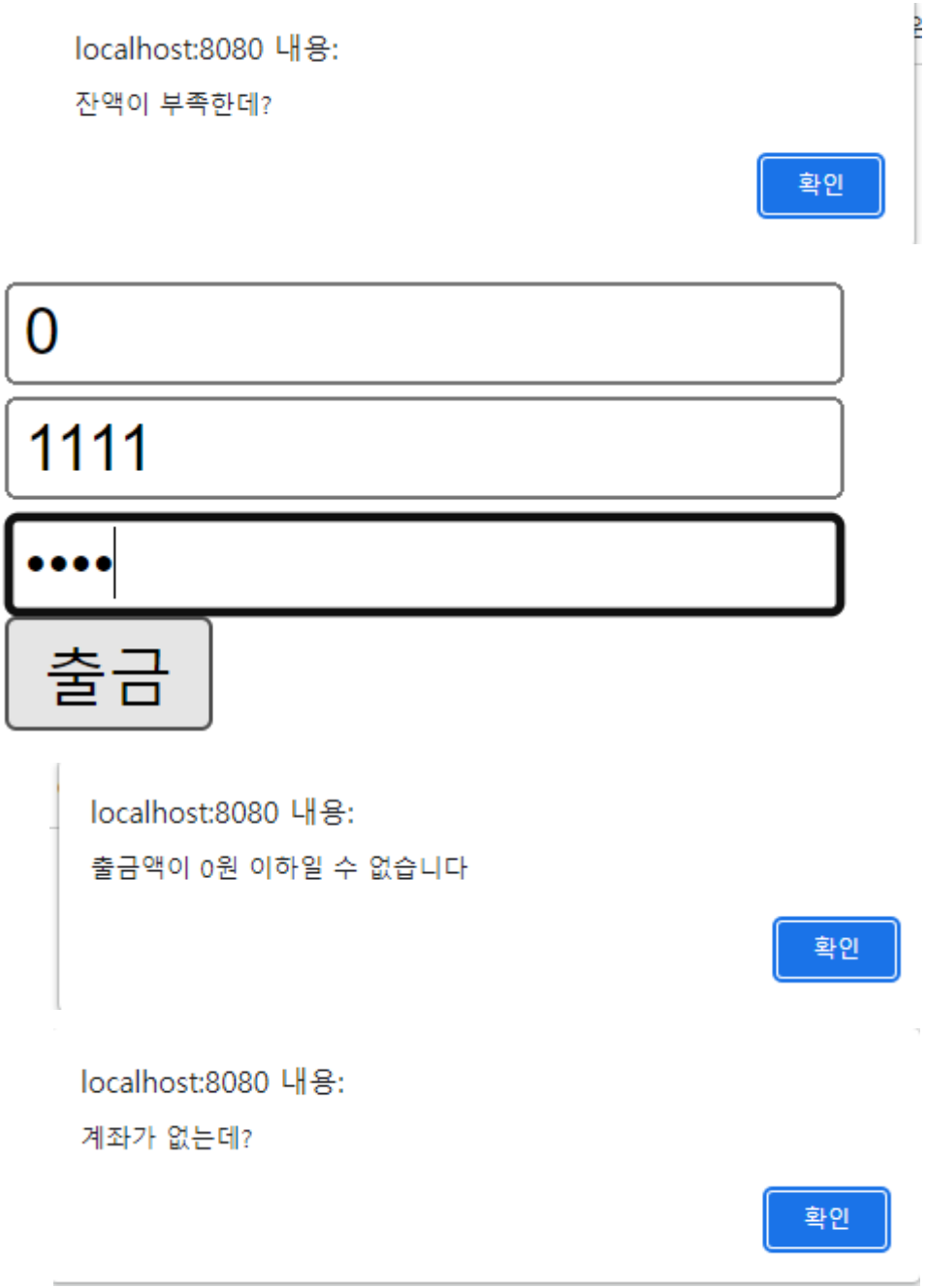
```
    // 3. 잔액확인
    accountPS.checkBalance(accountWithdrawReqDto.getAmount());

    // 4. 출금(balance - 마이너스)
    accountPS.withdraw(accountWithdrawReqDto.getAmount());
    accountRepository.updateById(accountPS);

    // 5. 히스토리 (거래내역)
    History history = new History();
    history.setAmount(accountWithdrawReqDto.getAmount());
    history.setWAccountId(accountPS.getId());
    history.setDAccountId(null);
    history.setWBalance(accountPS.getBalance());
    history.setDBalance(null);

    historyRepository.insert(history);

    // 6. 해당 계좌의 id를 return
    return accountPS.getId();
}
```

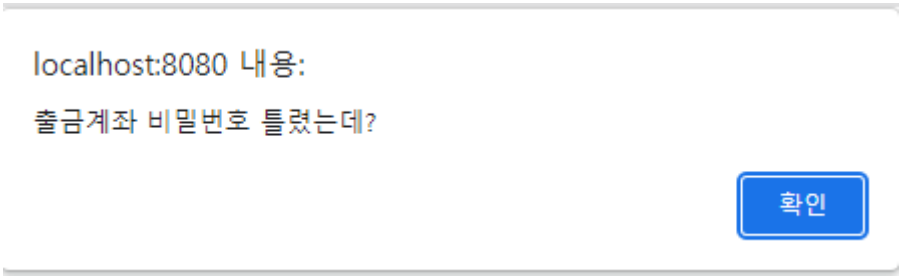
AccountService.java

```
 @Transactional
public int 계좌출금(AccountWithdrawReqDto accountWithdrawReqDto) {
    // 1. 계좌존재 여부
    Account accountPS =
accountRepository.findByNumber(accountWithdrawReqDto.getWAccountNumber());
    if (accountPS == null) {
        throw new CustomException("계좌가 없는데?", HttpStatus.BAD_REQUEST);
    }

    // 2. 계좌패스워드 확인
    accountPS.checkPassword(accountWithdrawReqDto.getWAccountPassword());

    // 3. 잔액확인
    accountPS.checkBalance(accountWithdrawReqDto.getAmount());

    // 4. 출금(balance - 마이너스)
    accountPS.withdraw(accountWithdrawReqDto.getAmount());
    accountRepository.updateById(accountPS);

    // 5. 히스토리 (거래내역)
    History history = new History();
    history.setAmount(accountWithdrawReqDto.getAmount());
    history.setWAccountId(accountPS.getId());
    history.setDAccountId(null);
    history.setWBlanace(accountPS.getBalance());
    history.setDBalance(null);
```

```
        historyRepository.insert(history);

        // 6. 해당 계좌의 id를 return
        return accountPS.getId();
    }
```

6. 실행 (테스트)





메인페이지

계좌번호	잔액
1111	900원
3333	1000원