BoostCourse

(Full-Stack Developer)

SeokRae

예약서비스

예약서비스?

- 예약서비스는 네이버웹서비스에서 사용됐던 서비스다.
- 실제 네이버 서비스의 동작과는 다르며, 본 기획서 내용을 토대로 개발해야 한다.
- 모바일웹과 PC웹을 하나의 UI로 구성한다. 따라서 PC에서도 동작하고, 모바일웹에서도 동작되는 UI이며, 예약서비스는 PC보다는 모바일웹에 어울리는 UI다.

- 크게 5가지 서비스 기능을 개발해야 한다.
  - 메인페이지
  - 각 항목의 상세페이지
  - 예약하기
  - 나의 예매 내역 확인
  - 한줄평등록

기능 요구사항 정의서

메인번호	기능	기능명	기능번호	설명	진행사항
1	예매	예매내역확인 (예약확인) - 예약취소기능	1.1	- 아래 기획서 내용대로 인터랙션이 동작한다.	
			1.2	- 취소를 누르면 취소하겠냐는 메시지 레이어가 화면 가운데 뜨고 확인/취소를 통해 즉시 반영된다. - 취소된 이후에는 '취소된 예약' 으로 새롭게 화면에 결과가 나온다.	
2	소스코드	JAVA - 이름규칙	2.1	- JAVA Naming Conventions 을 지킨다.	
			2.2	- 클래스의 이름과 메소드의 이름은 직관적으로 작성하도록 한다.	
			2.3	- 클래스의 이름과 메소드의 이름만 보아도, 어떤 기능을 가지고 있을지 어떤 내용이 구현되어 있을지 짐작할 수 있어야 한다.	
			2.4	- 코드를 읽는 사람이 개념을 쉽게 파악할 수 있도록 읽기 쉬운 코드를 작성하도록 한다. (예를 들어 변수이름을 구체적으로 작성하도록 한다.)	
3		JAVA - 중복된 코드 제거 및 코드 구조화	3.1	- 중복된 코드가 있다면, 별도의 메소드나 클래스로 분리하도록 한다.	
			3.2	- 하나의 메소드가 너무 많은 코드를 담지 않는다.	
			3.3	- 코드의 양이 많을 경우 private한 메소드를 이용해서 메소드를 분리하거나 별도의 객체를 만들어 사용하도록 한다.	
			3.4	- 클래스의 코드 길이가 너무 길어진다면, 해당 클래스가 몇개의 클래스로 분리될 수 있는지 고민한다.	
			3.5	- 변수는 최대한 덜 사용하고, 최대한 가볍게 만들어 가독성을 높이도록 한다.	
4		JAVA - 가독성	4.1	- 조건문의 경우 긍정적이고 흥미로운 (주 흐름에 해당하는) 경우가 앞쪽에 위치하도록 한다.	
			4.2	- 삼항연산자, do-while문장은 코드 가독성을 떨어트리니 되도록 사용하지 않는다.	
			4.3	- 블럭이 너무 많이 중첩되면 코드를 읽기 어려워진다. 블록을 private메소드로 추출할 수 있는지 고민한다.	
			4.3	- 코드는 빈줄을 이용해 커다란 블록을 논리적인 문단으로 구분한다.	
			4.4	- 코드는 들여쓰기를 잘 지키도록 한다.	
			4.5	- 필요하지 않은 코드는 제거한다.	
5		JAVA - 프로젝트 구조	5.1	- Controller, Service, Repository를 사용하여 구현되어 있어야 한다.	
			5.2	- Controller에서 Service를 Service에서 Repository의 기능을 호출할 수 있지만, 그 반대는 허용되지 않는다.	
6		Java - Web API	6.1	- 티켓 정보, 예약자정보를 전달받아 저장하는 Web API를 작성한다.	
			6.2	- email을 입력받아, 해당 email로 예약된 예약정보 목록을 보여주는 Web API를 작성한다.	
			6.3	- email로 검색된 결과가 1건 이상이 있을 경우 세션에 email정보를 저장한다.	
7		Java - JSP	7.1	- 세션에 email이 있을 경우 메인 페이지의 우측 상단의 "예약확인" 은 세션의 email값으로 변경된다.	
			7.2	- 메인 페이지의 email링크를 클릭하면 해당 email로 예약된 예약 정보 목록을 보여주는 Web API를 호출한 결과가 보여진다.	

기능 요구사항 정의서

- Project 5 기능 요구사항

기능 요구사항 정의서

메인번호	기능	기능명	기능번호	설명	진행사항
1	화면 레이아웃	예약하기	1.1	- HTML/CSS가 제공되는 것을 기본으로 작성해야 하며, 화면구성은 아래 기획서대로 보여야 한다.	
		예약확인하기	1.2	- HTML/CSS가 제공되는 것을 기본으로 작성해야 하며, 화면구성은 아래 기획서대로 보여야 한다.	
2	기능 - 예약하기	티켓입력	2.1	- 아래 기획서 내용대로 인터랙션이 동작한다.	
			2.2	- +,- 버튼을 누르면 숫자가 증가/감소되고, 금액이 변경된다.	
			2.3	- 0명이면 버튼이 비활성화 된다.	
3		예매자정보 입력	3.1	- 아래 기획서 내용대로 인터랙션이 동작한다.	
			3.2	- 총 매수는 예약상황을 합쳐서 갯수가 변경될 때 바로 변경되어 노출된다.	
			3.3	- 연락처 정보는 전화번호 포맷(휴대폰/일반전화 모두 가능)만 입력이 가능해야 한다.	
			3.4	- 그 외 값이 입력 되면 입력된 값의 형식이 틀렸다는 메시지를 자유롭게 만들어서 화면에 노출해야 한다.	
4		약관동의 및 예약	4.1	- 아래 기획서 내용대로 인터랙션이 동작한다.	
			4.2	- 약관은 접기/보기로 토글된다.	
			4.3	- 약관정보 동의에 체크하면 활성화된 상태로 보여진다.	
			4.4	- 예매갯수가 1개이상, 예매자입력,연락처입력,약관동의 된 상태라면 모든 값이 유효한 상태이다.	
			4.5	- 모든 정보가 유효하다면 '예약하기' 버튼이 활성화된 상태로 노출된다.	
5	기능 - 예매확인	예매내역확인 (예약확인) - 이메일로그인	5.1	- 아래 기획서 내용대로 인터랙션이 동작한다.	
			5.2	- 이메일 규칙검사를 한다.	
		예매내역확인 (예약확인) - 예약취소기능	5.3	- 아래 기획서 내용대로 인터랙션이 동작한다.	
			5.4	- 취소를 누르면 취소하겠냐는 메시지 레이어가 화면 가운데 뜨고 확인/취소를 통해 즉시 반영된다.	
			5.5	- 취소된 이후에는 '취소된 예약' 으로 새롭게 화면에 결과가 나온다.	
6	소스코드	JavaScript - 객체지향개발	6.1	- 자주 사용되는 함수를 객체형태로 묶어서 사용해야 한다.	
			6.2	- UI 별로 기능을 묶어서 객체화된 모듈을 만들어야 하며, prototype방식을 적용해야 한다.	
7		JavaScript - 정규표현식기반 유효성검증	7.1	- form에 입력된 값을 체크를 할 때는 값의 유효성(validation)을 체크해야 하며, 정규표현식을 써서 구현해야 함. (이메일 필드는 반드시 유효성검증해야 함)	
8		JavaScript 클린코드	8.1	- 함수 하나에 여러개의 기능을 넣지 않고, 함수를 여러개로 분리한다.	
			8.2	- 함수이름은 동사+명사이며 함수의 의도를 충분히 반영하고 있는가?	
			8.3	- 함수안의 내용은 이름에 어울리게 하나의 로직을 담고 있는가?	
			8.4	- 중복코드를 제거하고 공통부분을 공통함수로 만든다.	
			8.5	- 의도가 드러난 구현패턴. 의도가 표현되지 않은 코드를 최소한으로 줄이도록 한다.	
			8.6	- 전역변수를 없애려고 노력하자. (지역변수로 넣으면 될 걸 전역공간에 두지 말기 등)	

기능 요구사항 정의서

- Project 5 기능 요구사항

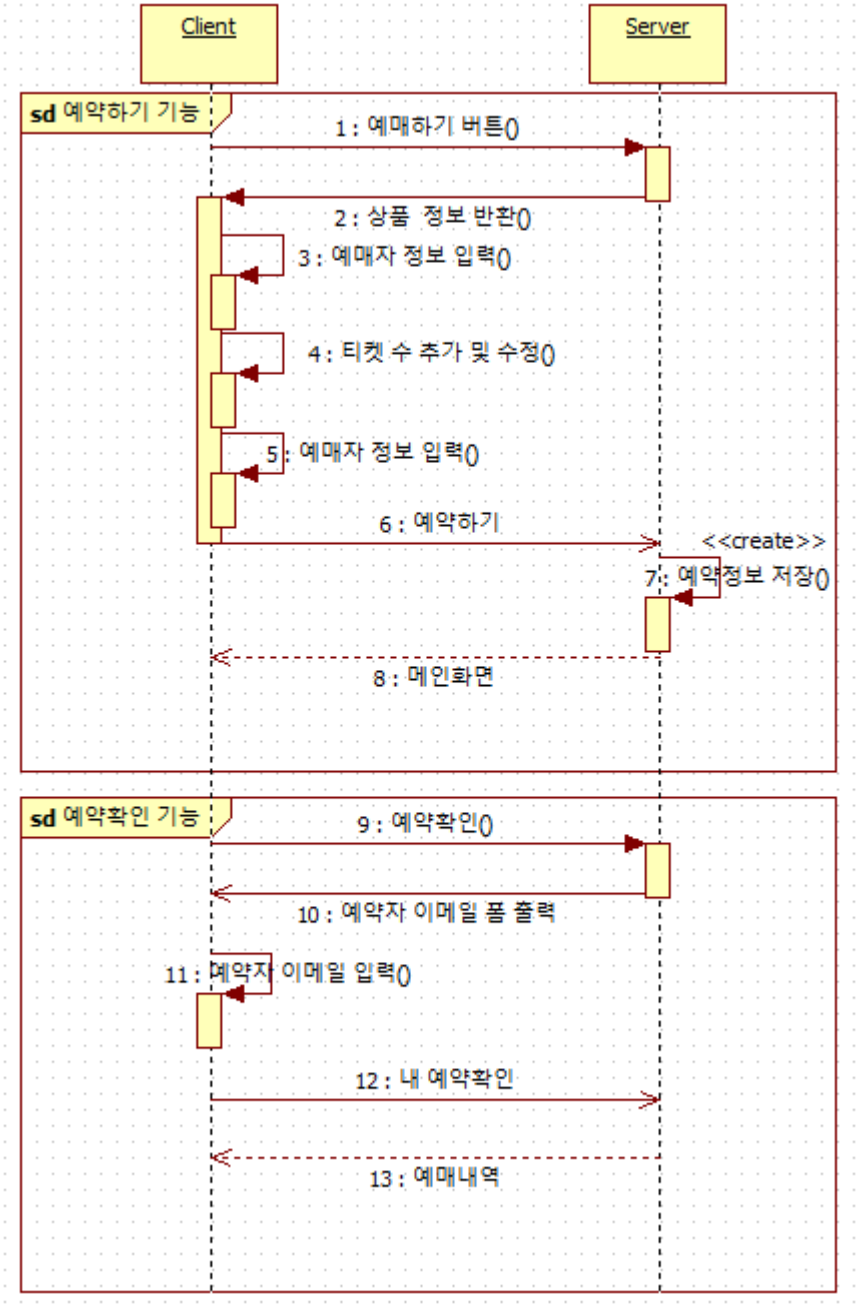
기술 요구사항 정의서

영역	기술명	규칙번호	규칙 설명
웹 프론트 기술 요구사항	개발규칙	1.1	- 자주 사용되는 함수를 객체형태로 묶어서 사용해야 합니다.
		1.2	- UI 별로 기능을 묶어서 객체화된 모듈을 만들어야 하며, prototype 방식을 적용해야 합니다.
		1.3	- 이전 프로젝트와 같이 클린코드의 규칙대로 코드를 구현합니다.
	유효성 검사	2.1	- form에 입력된 값을 체크를 할 때는 값의 유효성(validation)을 체크해야 하며, 정규표현식을 써서 구현해야 합니다. (이메일 필드는 반드시 유효성 검증해야 합니다.)
웹 백엔드 기술 요구사항	세션	1.1	- 예매확인을 위해 이메일을 입력 후 예약확인을 눌렀을 때 예매 내역이 있다면 이메일 정보를 세션에 저장합니다.
	메인화면	2.1	- 메인화면의 경우 세션에 이메일 정보가 있을 경우 예매 확인 버튼 대신 이메일을 보여주고, 이메일을 클릭하면 해당 이메일로 예약된 예매 내역이 보입니다.

기능 요구사항 정의서

- Project 4 기술 요구사항
  1. 웹 프론트 기술 요구사항
     - ① 객체 리터럴
     - ② 라이브러리 사용
     - ③ 클린코드
     - ④ 레이아웃
  2. 웹 백엔드 기술 요구사항
     - ① 레이어드 아키텍쳐
     - ② JDBC
     - ③ Web API

Sequence Diagram

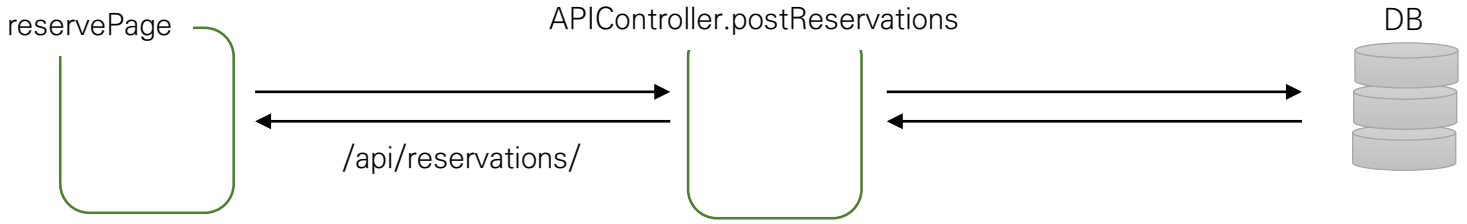


Sequence Diagram

- 예약하기
    1. Client의 예매하기
    2. Server에서 상품정보, 티켓 수 선택, 예매자 정보 입력, 약관 폼 노출
    3. Client는 예매자 정보 입력
    4. Client는 약관에 동의
    5. Client는 예약하기 버튼을 누른다.
    6. Server는 예약정보를 저장
    7. Server는 redirect로 메인화면을 Client에게 출력

- 예약확인
    1. Client의 예약확인
    2. Server는 예약자 이메일 폼을 Client에게 출력
    3. Client는 예약자 이메일을 입력
    4. Client의 예약확인으로 Server에게 데이터를 전송
    5. Server는 예약내역을 Client에게 출력

API Data Flow Diagram



View로부터 전달 받은 reservationParam

```
{
  "displayInfoId": 0,
  "prices": [
    {
      "count": 0,
      "productPriceId": 0,
      "reservationInfoId": 0,
    }
  ],
  "productId": 0,
  "reservationEmail": "string",
  "reservationName": "string",
  "reservationTelephone": "string",
  "reservationYearMonthDay": "string"
}
```

1. 예약 정보 등록 (reservation_Info)

```
{
  displayInfoId: 0
  , productId: 0
  , reservationEmail: “string”
  , reservationName: “string”
  , reservationTelephone: “string”
  , reservationYearMonthDay: “string”
}
```

2. 예약 정보 등록 PK 정보 가져오기 (reservation_info)

```
{
  reservationInfoId: 0
}
```

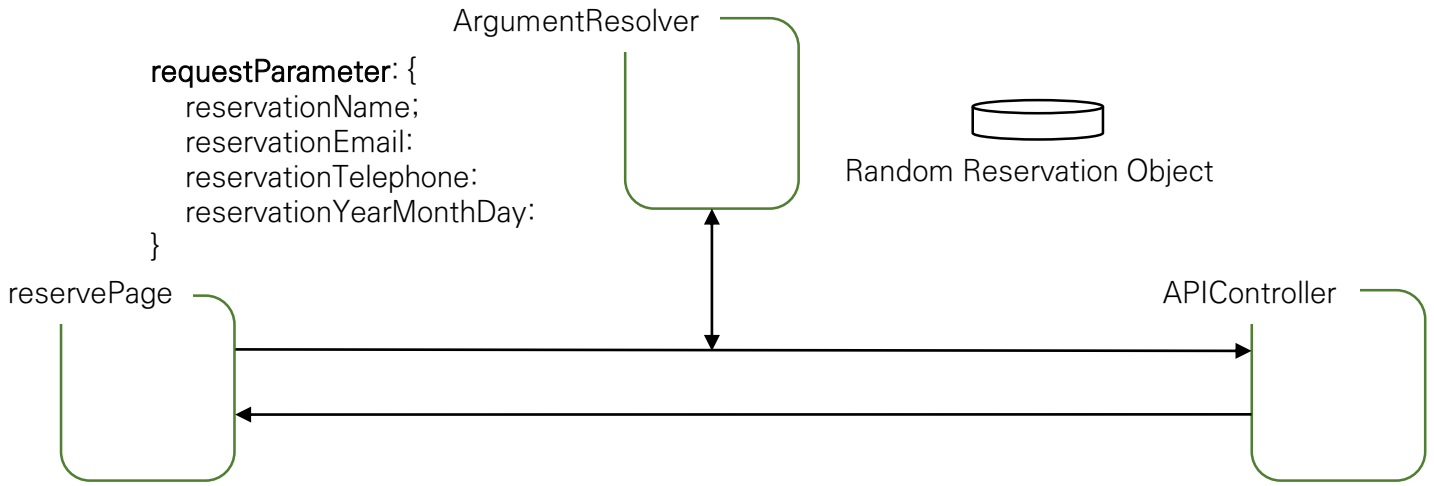
3. 예약 티켓 정보 등록 (reservation_info_price)

```
{
  reservationInfoId: 0
  , prices: [
    productPriceId: 0
    , count: 0
  ]
}
```

Data Flow Diagram

- 예약하기 기능
  - View로부터 예약 정보, 전시 티켓 정보 전달 받기
  - Controller에서 예약 등록 하기
    –〉 예약자 정보 등록 시 generated key 리턴
    –〉 generated key 값과 함께 티켓 가격 정보 등록
    –〉 티켓 정보 정상 등록 시 이메일 세션에 저장
    –〉 예약 확인 페이지 이동
- 예약확인

Session management



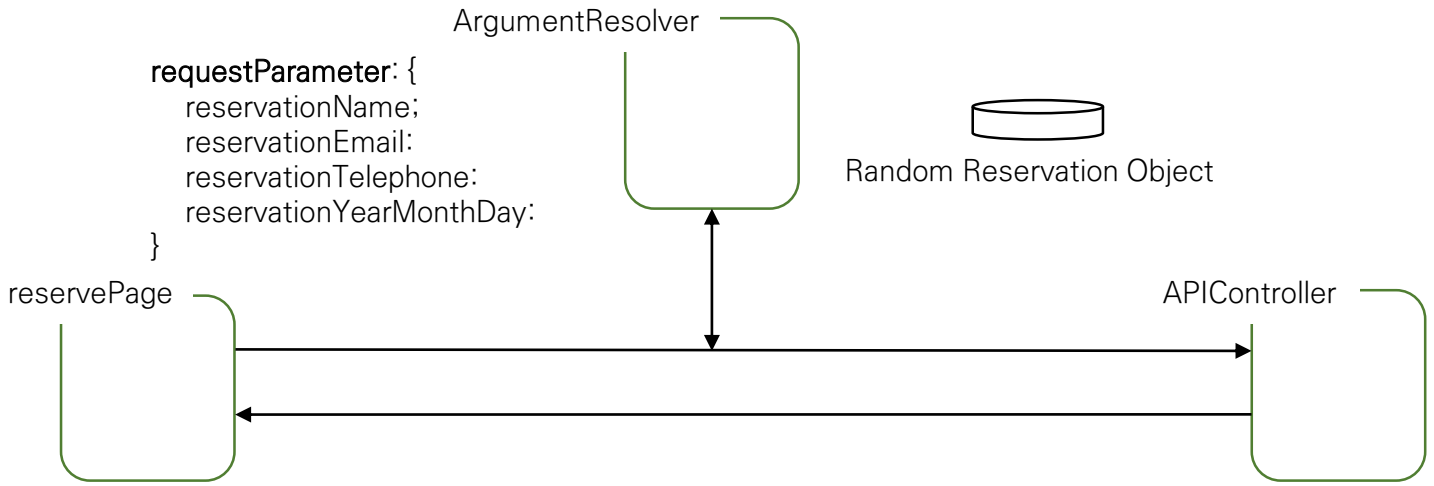
ReservationParam: {
reservationName: String
reservationEmail: String
reservationDate: String
reservationTelephone: String
reservationYearMonthDay: String
productId: 0
displayInfoId: 0
createDate: DATE
modifyDate: DATE
cancleYn: true
prices: [
count: 0
productPriceId: 0
reservationInfoId: 0
reservationInfoPriceId: 0
]
}

ReservationParam: {
reservationName: String
reservationEmail: String
reservationTelephone: String
reservationYearMonthDay: String
productId: 0
displayInfoId: 0
prices: [
count: 0
productPriceId: 0
reservationInfoId: 0
reservationInfoPriceId: 0
]
}

Data Flow Diagram

- 예매하기 페이지
- 예약하기 기능
  1. Client의 예약정보를 받는다. (name, Email, Telephone)
  2. Client의 예약정보를 토대로 ArgumentResolver를 통하여 예약 정보를 생성하여 DB에 적재
- 예약확인

Interceptor



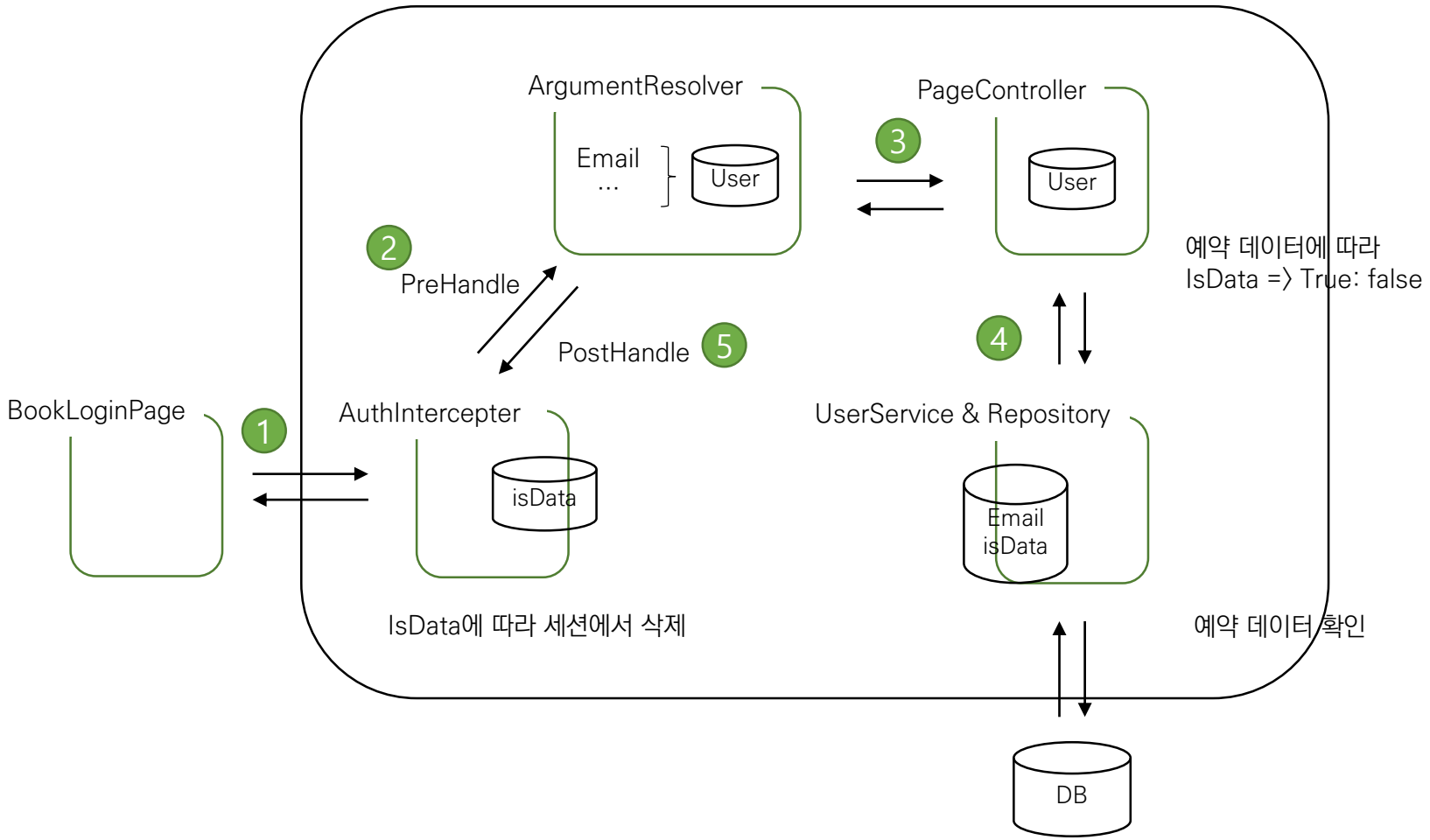
ReservationParam: {
reservationName: String
reservationEmail: String
reservationDate: String
reservationTelephone: String
reservationYearMonthDay: String
productId: 0
displayInfoId: 0
createDate: DATE
modifyDate: DATE
cancleYn: true
prices: [
count: 0
productPriceId: 0
reservationInfoId: 0
reservationInfoPriceId: 0
]
}

ReservationParam: {
reservationName: String
reservationEmail: String
reservationTelephone: String
reservationYearMonthDay: String
productId: 0
displayInfoId: 0
prices: [
count: 0
productPriceId: 0
reservationInfoId: 0
reservationInfoPriceId: 0
]
}

Data Flow Diagram

- 예매하기 페이지
- 예약하기 기능
  1. Client의 예약정보를 받는다. (name, Email, Telephone)
  2. Client의 예약정보를 토대로 ArgumentResolver를 통하여 예약 정보를 생성하여 DB에 적재
- 예약확인

Spring MVC Session



Session 처리하기

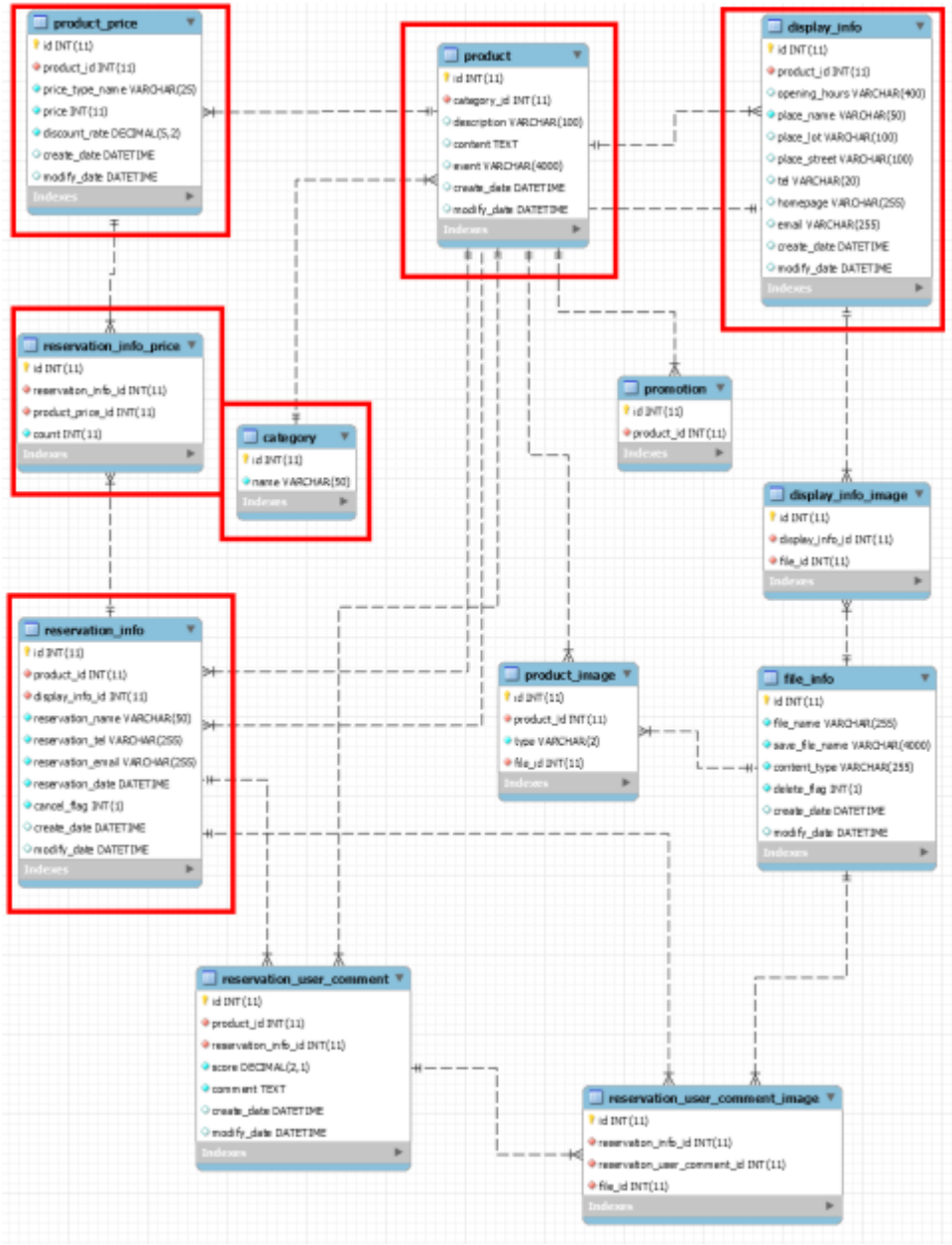
- **사전 조건**
  1. Argument Resolver
     - View에서 넘어오는 파라미터를 받아 User 객체로 만드는 역할
  2. Interceptor
     - 예약데이터 유무의 개념을 권한으로 치환하여 세션에 저장 판단
     - 권한 로그 관리

- **세션을 통한 로그인 처리 순서**
  1. View
     - bookLogin에서 email로 로그인
  2. Interceptor preHandle
     - Controller 전 파라미터 확인
  3. ArgumentResolver
     (각 파라미터들을 객체로 변환)
     - bookLogin페이지는 Email
     - Reserve페이지는 Email, Name, Phone
  4. Controller
     - DB 호출을 통한 예약데이터 존재여부 확인
     - 세션여부 값 저장
     - 기본적으로 User 객체가 세션에 저장
  5. Interceptor postHandle
     - 예약 데이터 존재여부 값에 따른 세션삭제

- 현재 PageController에서 @SessionAttributes, @ModelAttribute 사용으로 자동으로 저장되고 있는 데, 이를 Service 로직 이후 isData 값에 따라 세션에 저장할 수 있도록 하고 싶음

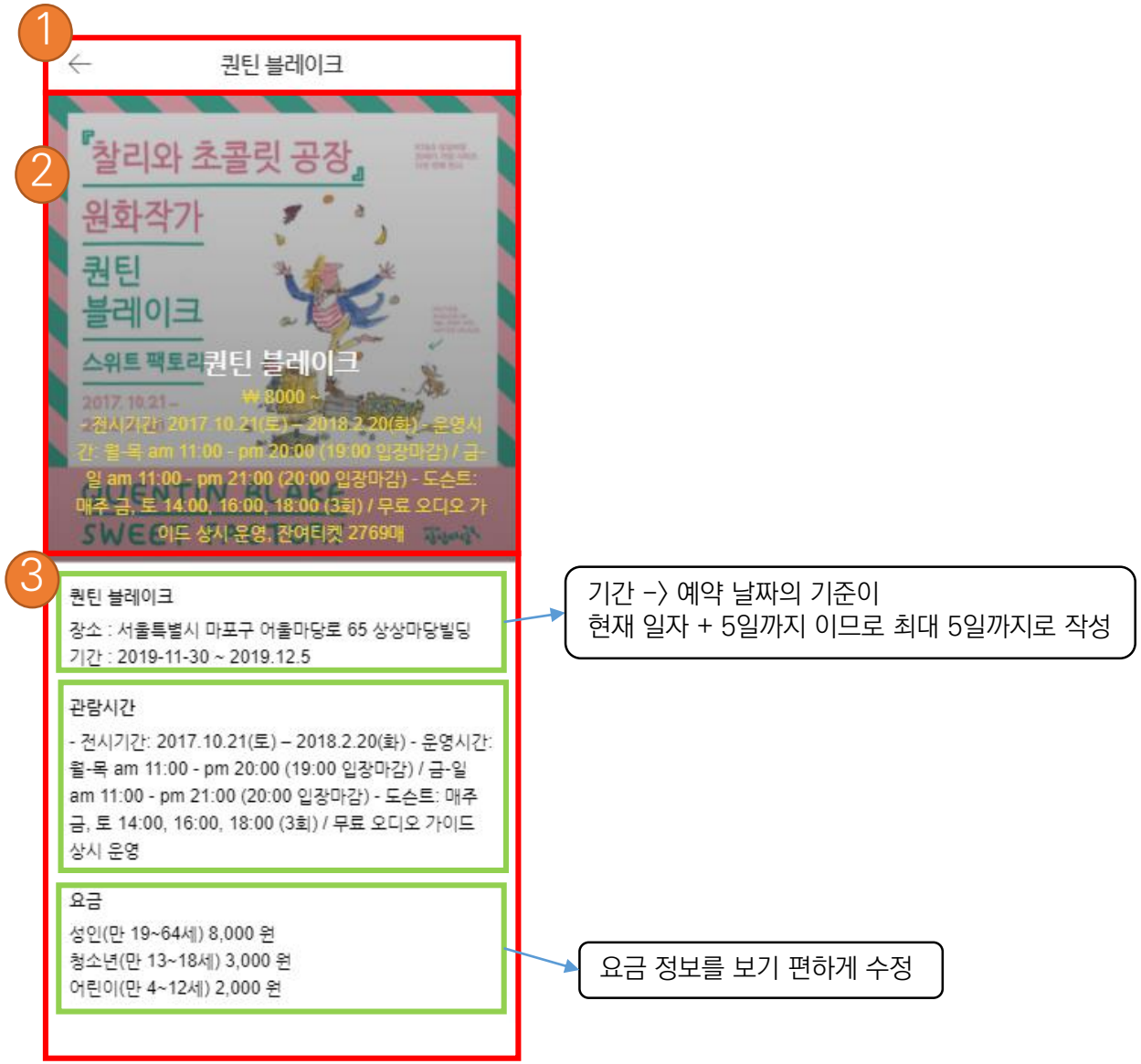
*** 필수로 만들어져야 하는 구조가 아님**

E-R Diagram



Database Table

예약하기 페이지 기능 구현



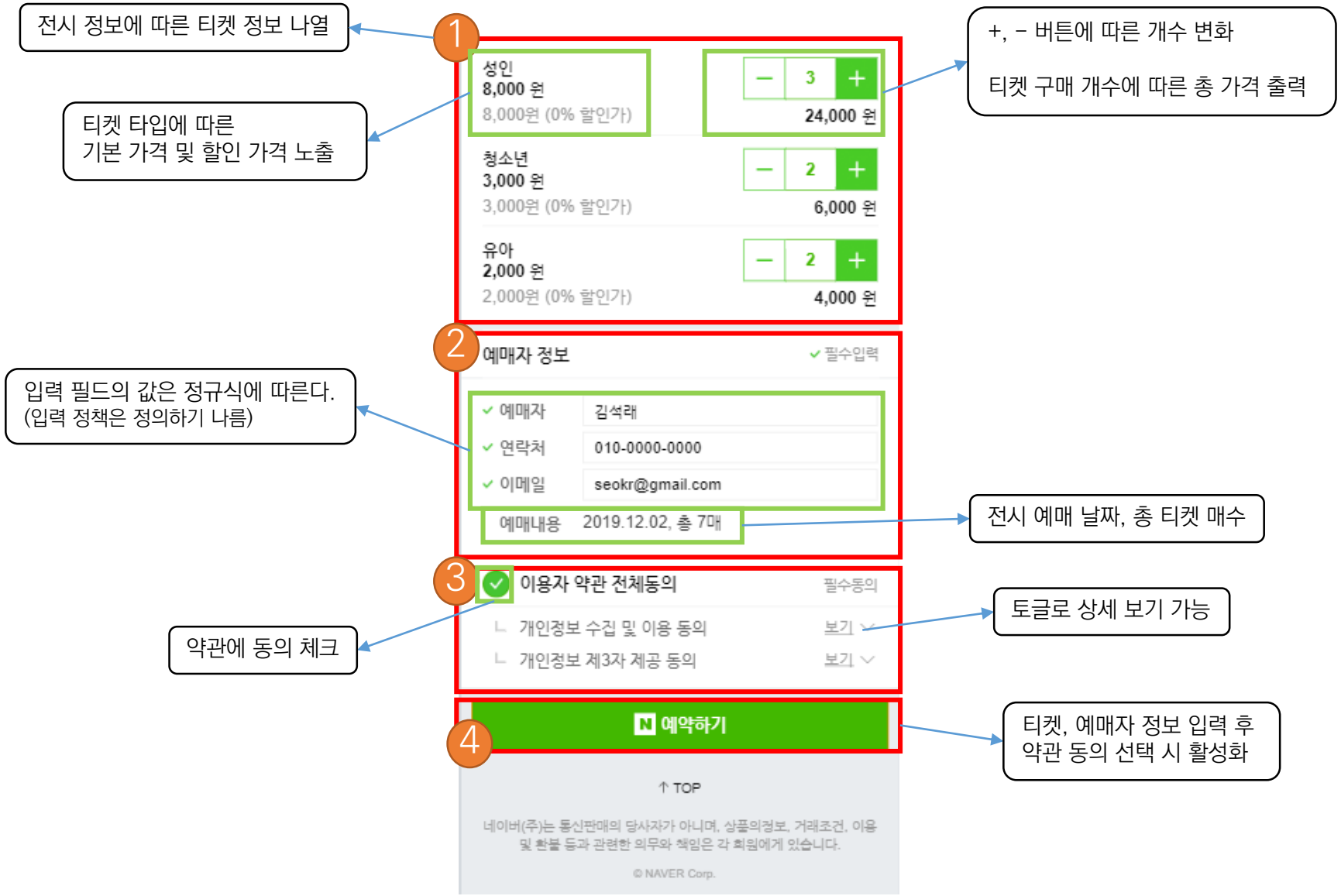
기간 -> 예약 날짜의 기준이
현재 일자 + 5일까지 이므로 최대 5일까지로 작성

요금 정보를 보기 편하게 수정

전시 정보

1. 상품정보
    - ① 타이틀
    - ② 전시이미지(ma)
    - ③ 전시 상세 정보
        - 타이틀
        - 장소
        - 기간
        - 관람시간
        - 요금 정보
2. 티켓 수 선택
3. 예매자 정보
4. 약관 동의 영역
5. 예약하기 버튼

예약하기 페이지 기능 구현



티켓, 예매자 정보 등록

1. **상품정보**
2. **티켓 수 선택**
   - ① 화면
   - ② 동적 티켓 타입 명, 가격, 할인가
3. **예매자 정보**
   - ① 이름
   - ② 연락처 (정규식)
   - ③ 이메일 (정규식)
   - ④ 예매 내용 정보 (날짜, 티켓 총 개수)
4. **약관 동의 영역**
   - ① 이용약관 동의
     위 티켓, 예매자 정보 유효성 검사
5. **예약하기 버튼**
   - ① 티켓, 예매자 정보, 약관 동의 조건 만족 시 활성화

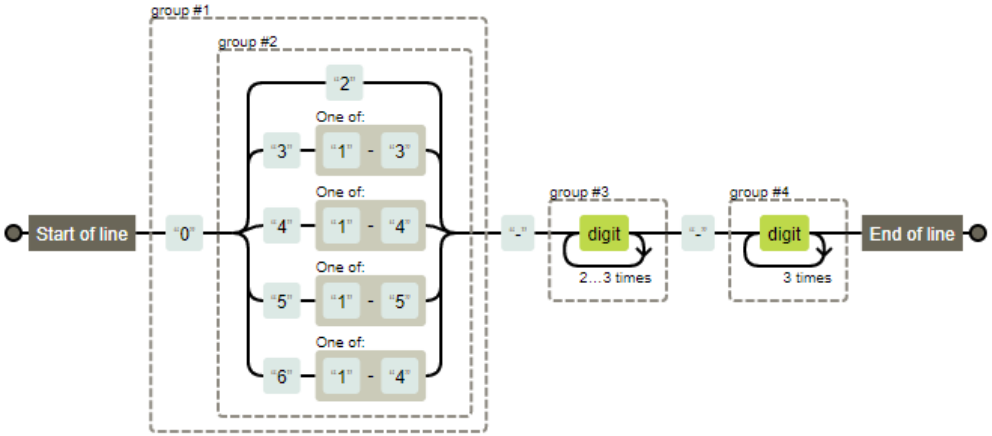
정규표현식

전화번호 체계

1. 지역별 전화번호 체계
2. 용도별 전화번호 체계

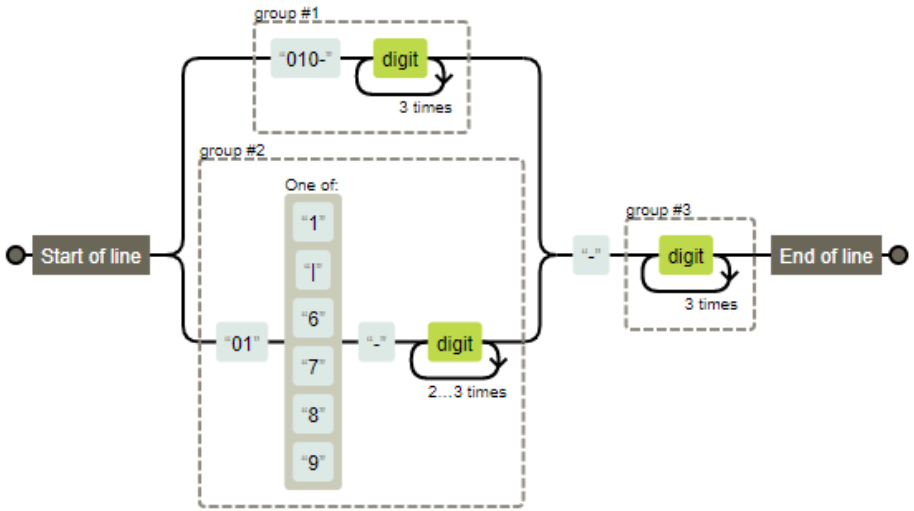
/^(0(2|3[1-3]|4[1-4]|5[1-5]|6[1-4]))-(₩d{3,4})-(₩d{4})$/

지역별 전화번호 체계	
번호	지역
02	서울특별시
031	경기도
032	인천광역시
033	강원도
041	충청남도
042	대전광역시
043	충청북도
044	세종특별자치시
051	부산광역시
052	울산광역시
053	대구광역시
054	경상북도
055	경상남도
061	전라남도
062	광주광역시
063	전라북도
064	제주특별자치도



/^(?:(010-₩d{4})|(01[1|6|7|8|9]-₩d{3,4}))-(₩d{4})$/

용도별 전화번호 체계	
번호	용도
00Y	국제전화
01Y	무선전화, 무선호출, 부가통신망
0N0	공통 서비스 (각종 지능망 등)
02	지역번호
03K ~ 06K	지역번호
08Y	시외전화
07Y, 09Y	(예비)
10Y	통신 사업자 민원안내 등
11Y·12Y	긴급 민원 신고
13Y·18Y	긴급 민원 신고
13YY	공공기관 생활정보, 안내, 상담
14YY	(예비; 기간통신사업자 부가서비스)
15YY	기간통신사업자 공통부가서비스 및 자율부가서비스 (전국대표번호 등)
16YY	기간통신사업자 공통부가서비스 (전국대표번호 등)
18YY	기간통신사업자 공통부가서비스 (전국대표번호 등)
17YY·19YY	(예비)

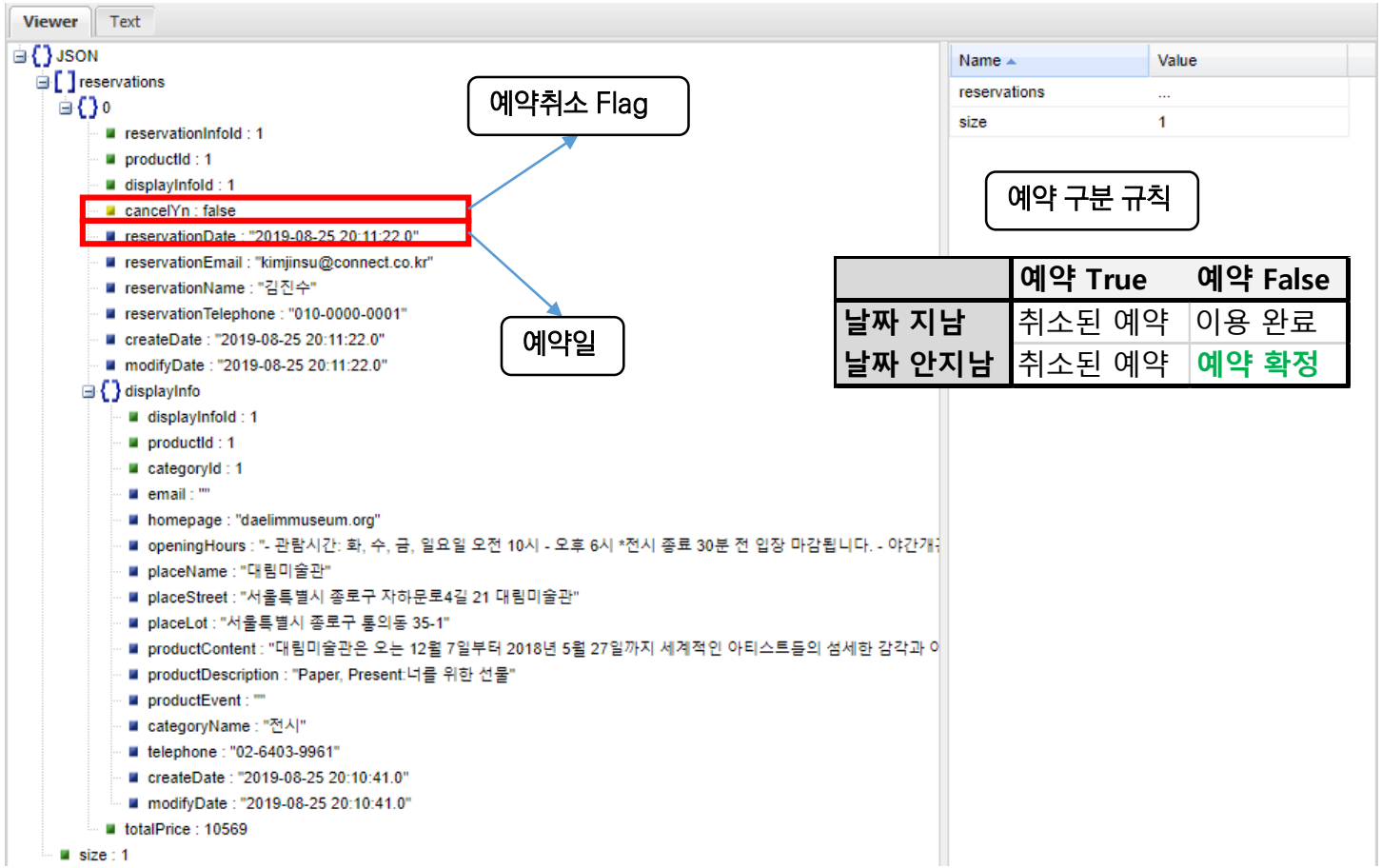


예약등록 후 세션 저장

예약 등록 후 세션 관리

1. 세션 사용 페이지
   - ① bookingLogin
     - 로그인 시 예약 정보 존재 시 세션 등록
   - ② reservePage
     - 예약 정보 정상 등록 시, 예약 정보가 존재하므로 세션 등록

예약확인 페이지 데이터 확인



예매 정보 확인

1. 예약 데이터 확인
   1) 예약구분 규칙 만들기
      - 예약 취소 구분값: cancelYn
   2) 예약정보 값
      ① No
      ② 서비스명/상품명
      ③ 일정
      ④ 내역
      ⑤ 장소
      ⑥ 업체명
      ⑦ 결제 예정 금액
   3) 정렬 순서 정하기
      ① 예약 신청 중
      ② 예약 확정
      ③ 이용완료
      ④ 취소된 예약

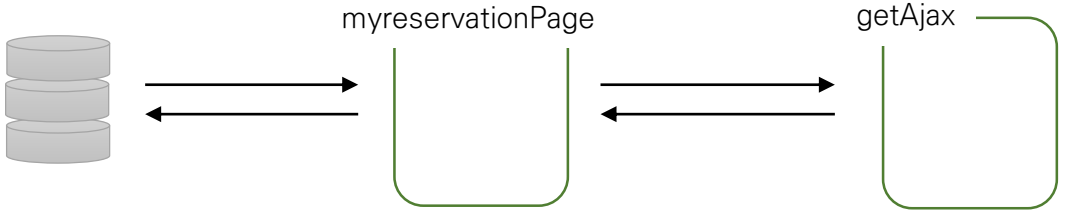
* 전시 데이터가 정적 데이터라는 것을 감안하여 개발자 임의대로 정의

*** '예약일자': 당일 + 최대 5일 인 것을 기준**
if(현재 날짜 – 예약 일자 〉 5) 이용완료
else 예약 확정

* 예약 신청 중은 어떻게 구분하지 ?..

예약확인 페이지 구현

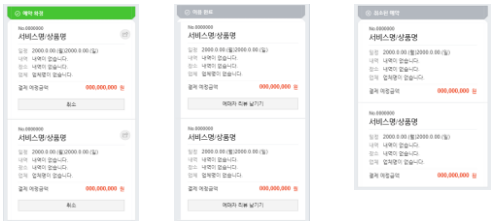
GET: /api/reservations?resrvEmail='메일주소'



예약 종류 분류

예약 확정 | 이용완료 | 예약 취소

템플릿 작성



Card –〉 예약 상품 요약 정보

〈article〉 예약 확정 관련 아이템 **리스트**

〈li〉 예약확정, 예약 취소, 이용완료

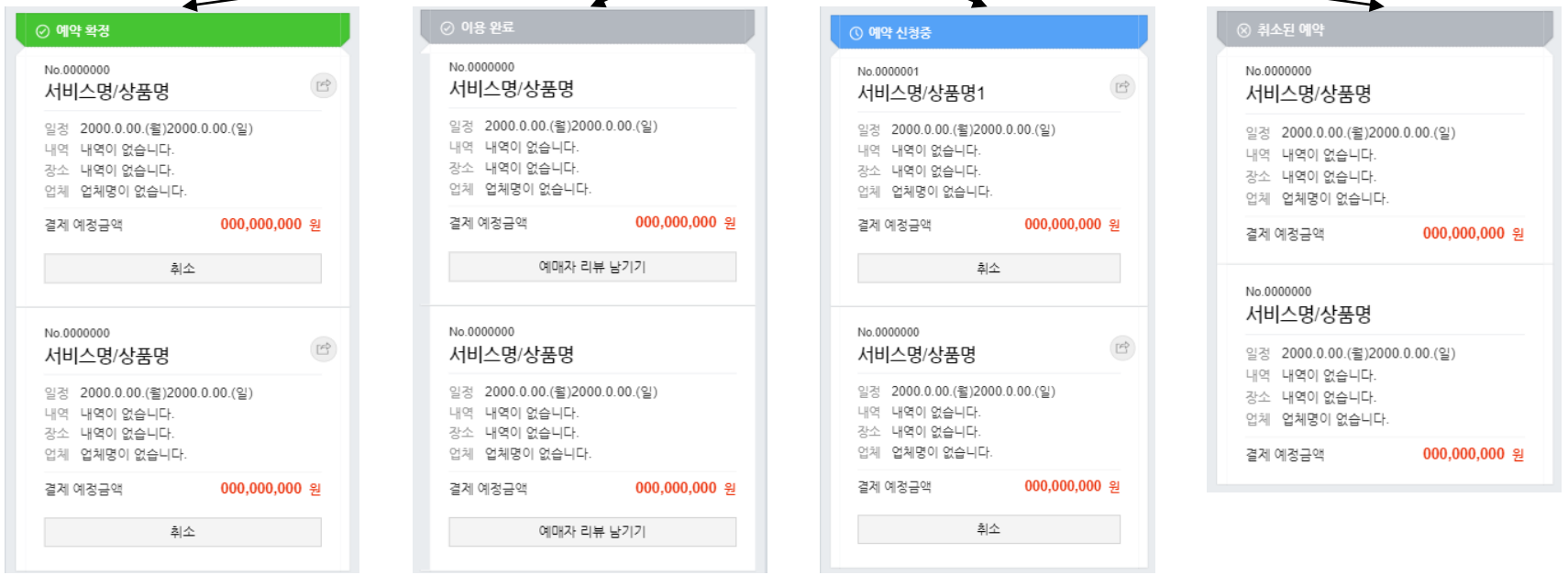
예매 정보 확인

1. Card
   - ① 예약 확정: .confirmed
   - ② 취소된 예약 & 이용 완료: .used
2. spr_book2
   - ① 예약 신청중: .ico_clock
   - ② 예약 확정 & 이용완료: .ico_check2
3. API

   ReservationInfoResponse
   - ① reservations
     (예약 조회 모델)
     a. displayInfo (상품 전시모델)
     b. reservation (예약 정보)
     …
   - ② Size (예약 수)

예약확인 페이지 구현

〈li〉 예약확정, 예약 취소, 이용완료




```
〈i class="spr_book2 ico_check2"〉〈/i〉
〈span class="tit"〉예약 확정〈/span〉
```

```
〈i class="spr_book2 ico_check2"〉〈/i〉
〈span class="tit"〉이용 완료〈/span〉
```

```
〈i class="spr_book2 ico_clock"〉〈/i〉
〈span class="tit"〉예약 신청중〈/span〉
```

```
〈i class="spr_book2 ico_cancel"〉〈/i〉
〈span class="tit"〉취소된 예약〈/span〉
```

```
〈div class="booking_cancel"〉
  〈button class="btn"〉
   〈span〉취소〈/span〉
  〈/button〉
〈/div〉
```

```
〈div class="booking_cancel"〉
 〈a href="/reviewWrite"〉
  〈button class="btn"〉
  〈span〉예매자 리뷰 남기기〈/span〉
  〈/button〉
 〈/a〉
〈/div〉
```

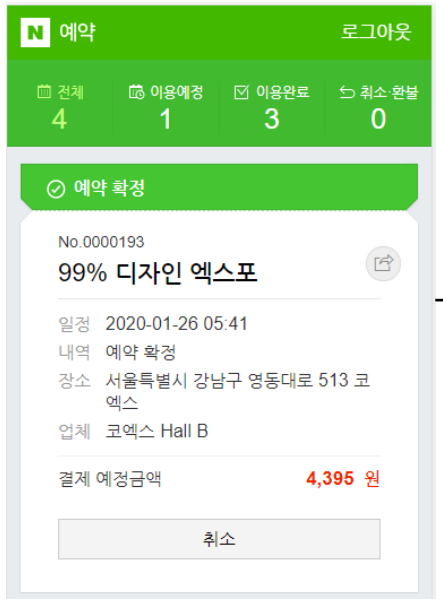
```
〈div class="booking_cancel"〉
  〈button class="btn"〉
   〈span〉취소〈/span〉
  〈/button〉
〈/div〉
```

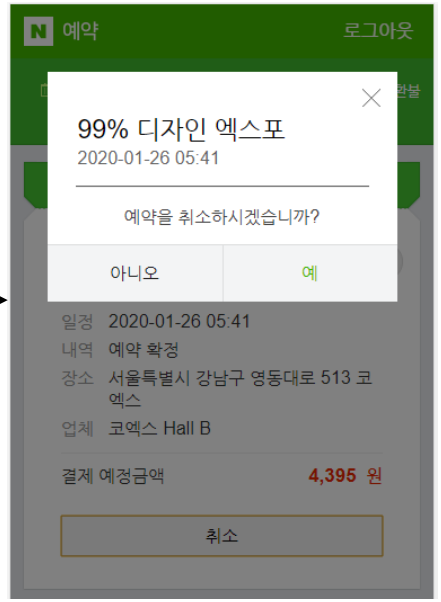
예매 정보 확인

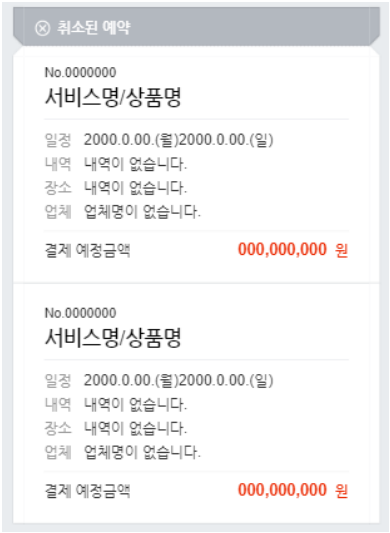
1. 예약 확정
2. 이용 완료
3. 예약 취소
4. 예약 내용 X

* 예약 신청중은 어떻게 구분하지 ?..

예약취소 팝업 구현







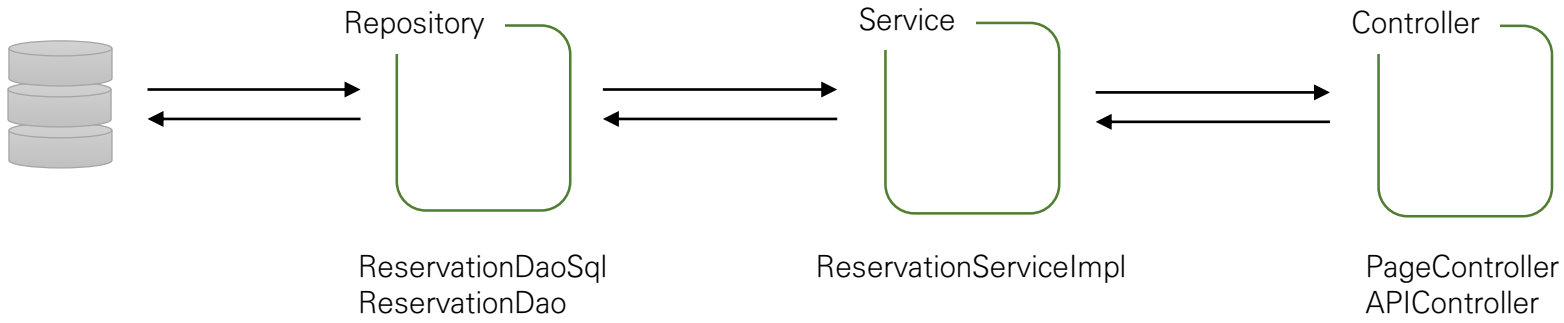
예약 취소 팝업

1. '예약 확정' 카드 클릭
2. 취소 버튼을 누른 특정 카드의 데이터를 팝업이 갖도록 하기
3. '예'를 누를 경우
    1. Ajax를 통해 비동기 호출
    2. '예약확정'데이터에서 삭제
    3. '예약확정' 카드리스트에서 삭제
    4. '예약취소'데이터리스트에 취소된 데이터 추가
    5. '예약취소' 카드리스트에 추가

* 특정 카드의 취소 버튼을 눌렀을 때 팝업은 그 데이터를 어떻게 갖고 갈 것인가?

* 취소 팝업에서 '예'를 누를 경우 예약확정리스트에 카드를 없애고, 예약취소리스트에 카드를 추가하는 방법을 어떤 식으로 구현할 것인가?

Feedback



Advice

해당 테이블 생성 DDL을 확인해보시면
해당 key 컬럼은 auto increment되도록 지정되어 있습니다.
따라서 굳이 insert시 value에서 제외하더라도 id 값 max + 1 된 값이 들어갑니다.

Advice

코드를 깔끔하게 보이기 위해 , 위치를 tab으로 맞춘 것으로 보입니다.
하지만 이는 유지보수에 분리할 수 있습니다.
만일 key 명이 더 긴게 있으면 다른 put도 tab을 늘려야되기 때문에 번거롭습니다.
따라서 ,의 위치를 마주는 것보다 띄어쓰기 하나로 parameter를 구분하도록 합니다.

```
paramMap.put("productId", reservationParam.get("productId"));
paramMap.put("displayInfoId", reservationParam.get("displayInfoId"));
paramMap.put("reservationEmail", reservationParam.get("reservationEmail"));
paramMap.put("reservationName", reservationParam.get("reservationName"));
paramMap.put("reservationTel", reservationParam.get("reservationTelephone"));
paramMap.put("reservationDate", reservationParam.get("reservationYearMonthDay"));
```

Advice

Transactional의 readOnly 멤버변수는 false이기 때문에
생략하셔도 무방합니다.
Transactional 어노테이션의 소스 참고 부탁드립니다.

```
/* TODO 예약 정보 등록 PK (GeneratorKey 같은거 있는지 알아보기) */
```

Advice

TODO 주석으로 남겨주신 것 처럼 insert 시
auto seq (key)를 반환받을 수 있습니다.
자세한 내용은 아래 링크 참고 부탁드립니다.

```
System.out.println("취소 정보 못가져옴");
```

Advice

System.out.println를 사용할 경우 해당 내용이 출력할 때까지 대기하도록 되어 있습니다.
따라서, 사용자가 많아진다면 애플리케이션에 해당 출력문이 출력할 때까지 대기할 수도 있습니다.
따라서 System.out.println보다는 logger를 사용하는 것이 좋습니다.

```
//    @GetMapping(path="/session")
```

Advice

구현을 했다가 만일을 위해 주석처리하신 것으로 보입니다.
가독성을 위해서 불필요한 소스는 지우는 것이 좋습니다.
만일 삭제한 코드가 추후에 필요할 때
현업에선 git과 같은 소스 형상 관리 툴을 사용하기 때문에
언제든 수정이력을 추적할 수 있기 때문입니다.

Advice

아래 그림을 보시면 들여쓰기에 기준이 틀린 것을 보실 수 있습니다.
들여쓰기에 기준을 한가지로 통일하시는 것이 좋습니다.
이유는 다음과 같습니다.
현재 제가 사용하는 IDE의 tab 기준은 스페이스 4 크기로 설정되어 있습니다.
만일 회사의 코드 포맷 표준이 들여쓰기를 tab으로 해야되며 tab의 크기는 스페이스 2로되어 있다면
해당 코드의 들여쓰기는 들쭉날쭉할 것입니다.
실무에서는 각 회사에 대한 코드 포맷 표준 가이드가 존재하며
이를 숙지하고 들여쓰기를 맞춰주는 것이 좋습니다.

```
@PostMapping(path = "/reservations")
public Map<String, Object> postReservations(
        @RequestBody Map<String, Object> reservationParam
        , HttpServletRequest requset) {

    Map<String, Object> reservationResponse = new HashMap<>();

    reservationResponse = reservationService.postReservations(reservationParam);

    if(reservationResponse.size() > 0) {
        requset.getSession().setAttribute( name: "reservationEmail", reservationParam.get("reservationEmail"));
    }

    return reservationResponse;
}
```

이러한 코드 포맷을 맞춰주는 것을 IDE에서 제공해주며
아래 링크 참조 부탁드립니다. (실무에서는 IDE에서 제공하는 코드 스타일 포맷을 사용하거나 회사에서 제공하는 코드 포맷터로 설정합니다.)

Advice

해당 주석은 javadoc 문서를 생성하기 위한 주석입니다.
따라서, parameter 또는 return이 변경되면 해당 주석도 변경해주시는 것이 좋습니다.
java doc 주석에 대해 정리되어 있는 링크를 남깁니다. 참고 부탁드립니다.

```
public class PageContoller {
```

Advice

클래스 명을 오타없이 수정하는 것이 좋을 것 같습니다.
Controller의 역할에 따라 파일명으로 찾기를 할 수 있는데
해당 파일 접미사가 Controller가 아닌 Contoller로 되어 있어 찾기 어려울 수 도 있습니다.
따라서 변수명, 클래스명 등에 오타가 없는지 모두 코드 구현 후에 오타가 없는지 한번더 확인하는 습관을 갖는 것이 좋습니다.

```
if(reservationEmail != null && !"".equals(reservationEmail)) {
```

Advice

String이 null 이거나 공백("")을 체크하기 위해 매번 해당과 같이 조건식을 구현하는 것이 번거로우실 겁니다.
이와같은 번거로움을 해결하기 위해 많은 라이브러리가 존재하며
아파치 재단에서 만든 String에 관련된 Util 라이브러리인 StringUtils가 있습니다.
해당 라이브러리를 사용하면 아래와 같이 간단하면서도 가독성 좋게 구현할 수 있습니다.

```
if (StringUtils.isNotEmpt(reservationEmail)) {
    ....
}
```

StringUtils에 관한 maven 의존성 추가 및 메소드들에 내용은 아래 링크 참고 부탁드립니다.
maven 의존성 : https://mvnrepository.com/artifact/org.apache.commons/commons-lang3/3.0
StringUtils methods : https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/StringUtils.html

Comments

1. ReservationDaoSql
   - ① INSERT_RESERVATION_INFO
     reservationInfoId가 auto_increment임에도 불구하고 가장 최근 pk값을 가져와 + 1 하는 쿼리를 작성함
2. ReservationDao
   - ① Map<String, Object> 데이터 관리
     다수의 값을 넣을 때, 무의미한 tab
3. ReservationServiceImpl
   - ① PostReservations(), PutReservations() 에 @Transactional 옵션
     Transactional은 default 값이 false이기 때문에 항상 명시할 필요 없다.
   - ② PostReservations() 구현 시
     예약 정보 입력 후 > 예약 가격 정보 입력 시 예약 정보에 대한 pk 값을 auto key를 이용할 것
   - ③ System.out.println() 남기지 말기 또는 logger 사용하기
4. APIController
   - ① 사용하지 않는 메서드 주석 처리하지 말고 지울 것
   - ② 들여쓰기 기준을 두고 사용하기
   - ③ 코드와 주석 일치시키기
5. PageController
   - ① 클래스 명 오타 수정
   - ② Null, 공백 체크 시
     라이브러리 사용하기 StringUtils

Feedback

Advice

ajax 호출을 위한 공통 함수로 보이며 잘 활용하고 잘 구현해주셨습니다.
단, get 접두사는 getter의 의미로 ajax를 가져온다라는 의미로 해석됩니다.
좀더 의미에 맞는 메소드명으로 바꾸시는게 좋을 것 같습니다
ex)
callAjax or callApi

Advice

취소 팝업에 경우 jsp 쪽에 분리하여 핸들바를 사용하는 것이
자바스크립트의 가독성이 더 좋아 질 것 같습니다.

Advice

boolean 변수엔 접두사로 is, has should를 많이 붙입니다.
이유는 해당 변수명 처럼 변경시
코드가 문장처럼 읽히기 때문입니다.
isValidName 같이 변경하는 것을 권장드립니다.

Advice

전화번호 유효성 검사를 위한 정규표현식을 잘 구현해주셨습니다.
단, 일반전화의 유효성 감사가 되어 있지 않습니다.
지역번호 02, 031, 032등 일반 전화번호도 유효성 체크를 할 수 있도록 수정해보세요.

Comments

1. JS 코드 명칭 수정
    - ① Ajax 공통호출 함수명 수정
      getAjax() -> callAjax()
    - ② 유효성 검사를 위한 변수명 수정
      boolName -> isName

2. 소스코드 구조 개선
    - ① 예약 취소 템플릿 구현 코드 수정
      현재 취소 기능 안에 템플릿 코드도 함께 있는 것을 handlebar를 이용하여 기능과 템플릿 코드 구분

3. 정규 표현식 개선
    - ① 기본 휴대전화에 대한 정규식 수정
      일반전화에 대한 유효성 검사가 가능하도록 개선