*Use-case 모델링 - 요구사항 식별

시스템

actor → Use-case
Use-case
actor →

*Actor 식별 ↙시스템
└ 시스템을 사용하는 사람, 프로세스를 가르킨다.

사람
매니저 →
강사 →
학생 →

학생관리 시스템 ← 출입 관리 시스템
← 타이머 프로세스

*시스템을 사용하는 액터 = "Primary Actor"

학생→	학원시스템	→결제시스템
Primary Actor		Secondary Actor
		(Supplementary)
주액터		보조액터
		↓
		>우리가 개발할 시스템이 사용하는 외부 시스템

*Actor의 관계

관리자
Sub Actor → 회원 ← Super Actor
학생 강사

↗ 로그인
→ 로그아웃
↘ 회원탈퇴

← 상속관계(서브)
> 서브액터는 수퍼액터의 역할을 포함한다.
> 학 수퍼액터가 사용할 수 있는 기능을
> 서브액터도 사용할 수 있다.

*Use - case 식별
└ 액터가 시스템을 사용하여 처리하고자 하는 업무(사용사례)

액터 → 유스케이스

*Use-case 식별 지침서(권고사항)
1. 시스템을 통해 수행하는 업무
2. 한 액터가 수행하는 업무
3. 한번에 한 순간에 수행하는 업무
↓
쉴 수 있는 업무
↓
1주~4주 사이에 개발할 수 있는 규모로
시스템의 기능을 조결 수 있다.
↓
프로젝트 관리가 용이한 규모로
개발할 시스템의 기능을 정의할 수 있다.

"상품구매" → 2. 결제 ← 사용자
↑ 3. 배송 ← 배송업체,택배사
너무 Use_case가 크다.

예) 게시글 등록하기
게시글 목록 조회하기
게시글 상세 조회하기
게시글 변경하기
게시글 삭제하기

↙ 쪼개는게 낫다.
1. 상품조회
로그인/로그아웃 -> 업무가 아니기때문에 use-case 작성할 필요없음.

Use-case = 시스템의 동작을 유저 입장에서 표현한 시나리오.
1. 한 액터가 하는 게 아니다.
2. 한번에 한 순간 수행하는 업무로 쪼개기.

Use-case 식별
└ 액터가 시스템을 사용하여 처리하고자 하는 업무 (사용사례)

액터 → 유스케이스

*Use-case 실무
1. 서로 관련된 Use-case인 경우 한계의 use-case로 합칠 수 있다.

너무 use-case를 잘게 쪼개면
오히려 관리하기 번거롭다.
↓
단, 합쳤을때
use-case의 개발 기간이
4주 안팎인 것이 좋다.

제품 주문하기
제품 배송하기
제품 구매하기
로그인하기
로그아웃하기
사용자 인증

공통
주소입력

2. CRUD Use-case 들을 하나의 Use-case로 합칠 수 있다.
- 게시글 등록하기
- 게시글 목록 조회하기
- 게시글 상세 조회하기
- 게시글 변경하기
- 게시글 삭제하기

게시글 관리하기

3. 여러 Use_case의 공통 시나리오인 경우 별도의 use-case로 뽑을 수 있다.
게시글 등록하기 ↘
게시글 변경하기 →
게시글 삭제하기 ↗
(include)
로그인하기 → 로그인하기는 비록 업무는 아니지만
여러Use-case에 포함된 작업이다.
공통기능으로 분리하여 개발할 수 있게 별도의 use-case로 뽑아낸다.

기능을 나눌때도 합리적으로 관리하고 개발하기 좋도록 Use-case 식별

*Use-case 관계
> 포함 / 확장 관계를 명시적으로 표시하는게 이해에 도움이 될 때 그려라
>굳이 그리지 않아도 관계를 유추할 수 있다면 그리지 말라!

1. 포함관계(include) = 필수 → <<include>>
>> 회원가입하기 → 이메일 중복검사
>> 회원정보 변경하기 <<include>>

회원가입

* Use-case 관계

① 포함관계(include) : 필수
Base UC
<<include>>

② 확장관계(extend) : 선택
Base UC
<<extend>>

✓이메일 중복검사를 수행하지 않고는 회원가입 업무를 완료할 수 없다.

이메일 중복검사

1. 포함관계(Include) : 필수

Base UC <<include>>
회원가입하기 ↘
이메일 중복검사

이메일 중복검사를 수행하지 않고는 회원가입 업무를 완료할 수 없다.

Base use-case

inclusion Uc

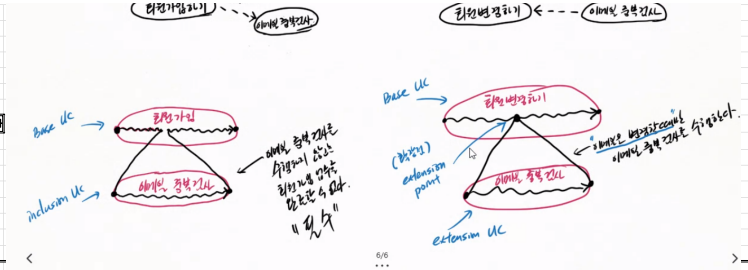
2. 확장관계(Extend) : 선택

Base UC <<extent>>
회원변경하기 ←- - - - - - - - - 이메일 중복검사

이메일을 변경할때만 이메일 중복 검사를 수행한다.

Base UC 회원변경하기

확장점
Extension point 이메일중복검사



*Use-case와 서블릿

			<<extent>>				
비회원	→	회원가입하기	←- - - - - -	주소검색하기			
		↓		↓			
		SignupServlet		SearchAddrServlet			
회원	→	주문하기	→	OrderAddServlet	Usecase	↗	1 . . * Servlet
			↘			→	Servlet
				PaymentServlet		↘	Servlet

*한 개의 Usecase는
한 개 이상의 서블릿으로 처리
[서블릿 = 컨트롤러]

*요구사항 정의 → 설계 → 구현

1. 요구사항(분석) 정의 → 2. 설계 → 3. 구현

Class Diagram

er Diagram

use-case 명세서(text)
기능/비기능 명세서
UI Prototype

시스템분석가
요구사항 정리

*프로젝트에 use-case Modeling 적용

				각 use-case에 대해
1. actor 식별	→	use-case 식별	→	3.ui-prototype
		로그인하기	→	로그인 id/암호
		회원탈퇴하기		
		게시글등록하기	→	게시글/제목/내용

*JSP → 서블릿 클래스

directive element	<%@ page contentType = "text/html;charset=utf-8">	그대로 복사	
template data	<html>		
	<head>	작성된 순서대로 출력문으로 변환	
scriptlet	<% int a = 20;		
	int b = 30 ; //if() { - }		
	%>		
	<%!		
	int x = 300;		
	void m() { - }	선언된 순서대로 그대로 복사	
	declaration element		

class xxx extends HttpServlet{

int x = 300;
void m () { - }

void -jspService(- - -) {
resp.setContentType("— —");
out.write("<html>\n");
out.write("<head>\n");

int a = 20;
int b = 30;

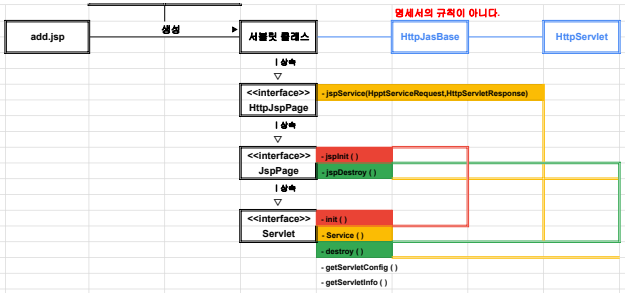
web.xml = DD파일

<welcome-file-list>	
<welcome-file>index.html</welcome-file>	해당되는 폴더에 있는 index.html이 없으면
<welcome-file>index.jsp</welcome-file>	해당되는 폴더에 있는 index.jsp이 없으면
<welcome-file>welcome.jsp</welcome-file>	해당되는 폴더에 있는 welcome.jsp이 없으면
</welcome-file-list>	
</web-app>	

자바 소스를 수정할때만 배치를 해야하기때문에 퍼블리시를 해야함

*JSP 파일로 만드는 서블릿 클래스를 만들때 지켜야할 조건

JSP 엔진


명세서의 규칙이 아니다.
add.jsp
생성
서블릿 클래스
HttpJasBase
HttpServlet
상속
<<interface>>
HttpJspPage
- jspService(HpptServiceRequest,HttpServletResponse)
상속
<<interface>>
JspPage
- jspInit ( )
- jspDestroy ( )
상속
<<interface>>
Servlet
- init ( )
- Service ( )
- destroy ( )
- getServletConfig ( )
- getServletInfo ( )