

Architecture

Daum Corp.
백명석

발표목차

- What is Architecture
- What is Use Case
- Architecture Exposes Usage
- Deferring Decisions
- Deferring Decisions 예. Fitness
- Central Abstraction
- Conclusion

What is Architecture

- 전체적인 시스템 개발에 기반을 제공하는 변경 불가한 초기 결정사항의 집합
 - Set of irrevocable early decisions that raised the foundation for the entire system development
 - Java: Language
 - Eclipse: development environment
 - Spring, Tomcat, Hibernate: FW
 - MySQL, MVC
 - 건축의 아키텍처가 해머, 못 등이 아니듯
 - 툴, Building Material 등이 아니라 사용법(Usage)에 대한 것

What is Use Case

- a list of steps, typically defining interactions between a role(actor) and a system, to achieve a goal
- 게시글 작성하기

사용자	시스템
제목을 입력한다.	제목의 유효성을 조사한다.
본문을 입력한다.	본문의 유효성을 조사한다.
글 작성하기를 요청한다.	글을 저장한다.

Architecture Exposes Usage

- 아키텍처는 사용법에 대해서 설명
- MVC 구조만 있는 웹시스템
 - Use case를 숨기고 Delivering 메커니즘만 노출
 - 중요한 것은 Delivering 메커니즘이 아니라 Use case
 - Use case는 Delivering 메커니즘에 decoupled되어야
 - UI, DB, F/W, Tools 등에 대한 결정들이 Use case와는 완전히 decouple되어야 한다.
- Use case should stand alone

Deferring Decisions

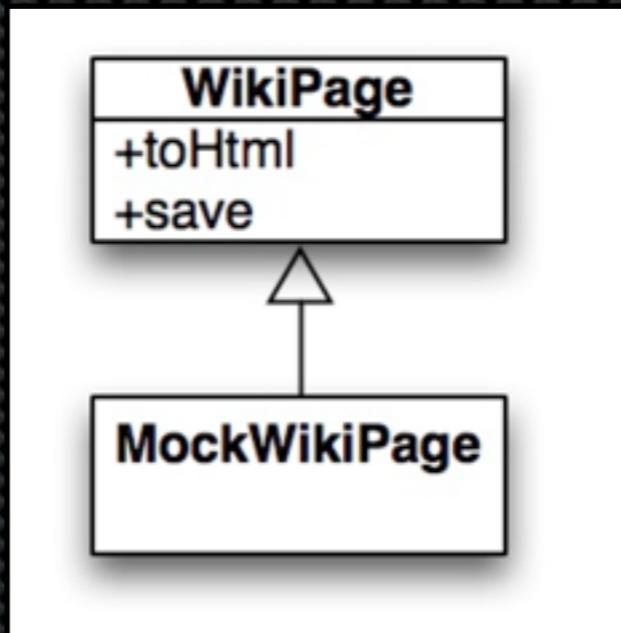
- 좋은 아키텍처는 FW, WAS, UI 등과 같은 stuff들에 대한 결정을 연기하는 것을 허용
 - Stuff에 대한 결정은 연기될 수 있어야하고,
 - 연기되어야만 한다.
 - 이게 좋은 아키텍처의 주요한 목적 중 하나
- 시간이 지날 수록 결정을 위한 정보가 풍부해짐

Deferring Decisions 예. Fitness

- 위키 저장을 위해 MySQL을 생각
- 무언가를 하기 전에 DB를 먼저 기동하고 스키마를 개발해야 한다고 생각
- But 사실 이게 바로 필요하진 않았다.
- 위키 텍스트를 html로 변환하는 것에 초점
 - 파싱, 번역하는 코드는 DB 없이도 개발할 수 있다

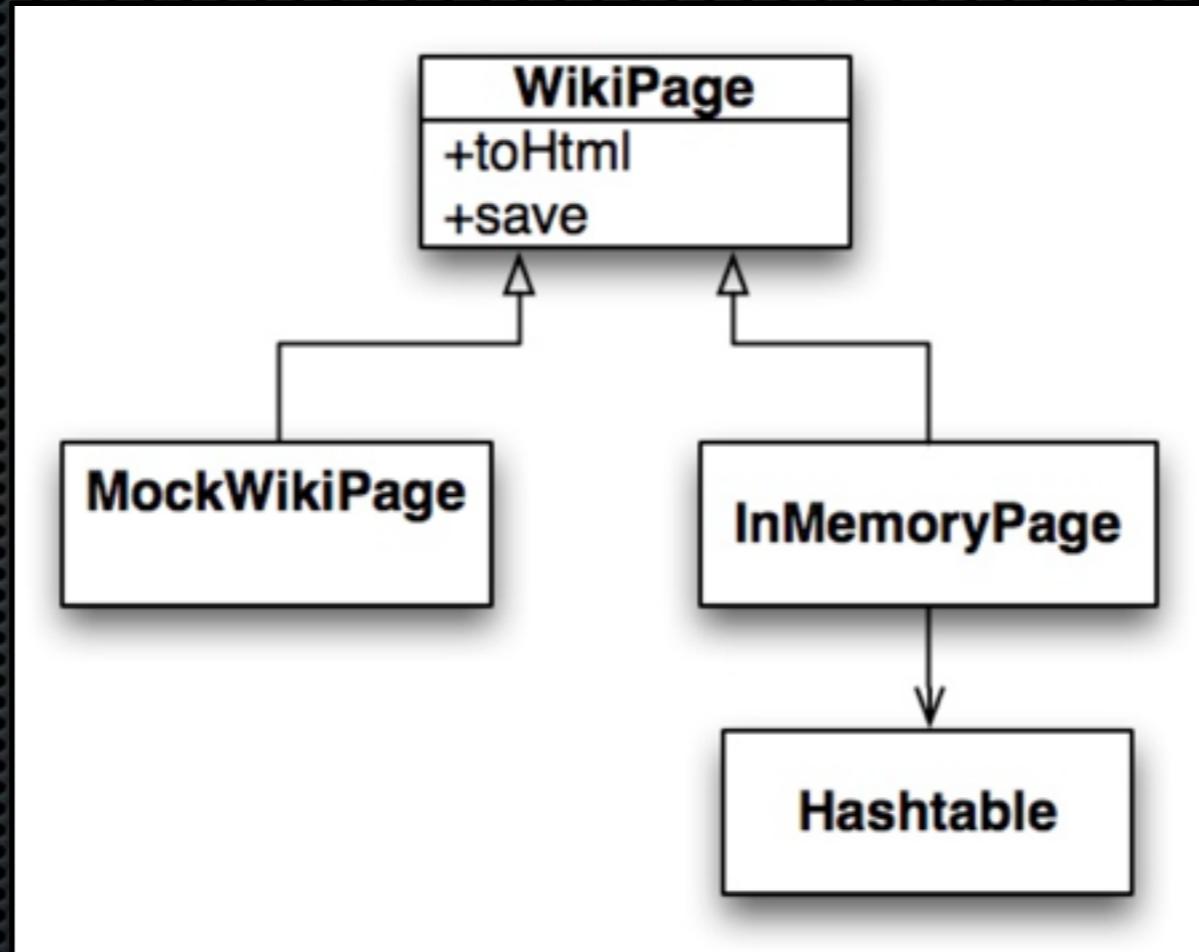
Deferring Decisions 예. Fitness

- WikiPage라는 중요한 추상화
 - wiki text를 가짐
 - toHtml: wiki text를 html로 parse
 - save: wiki text를 DB에 저장
- mocking
 - override save
 - do nothing



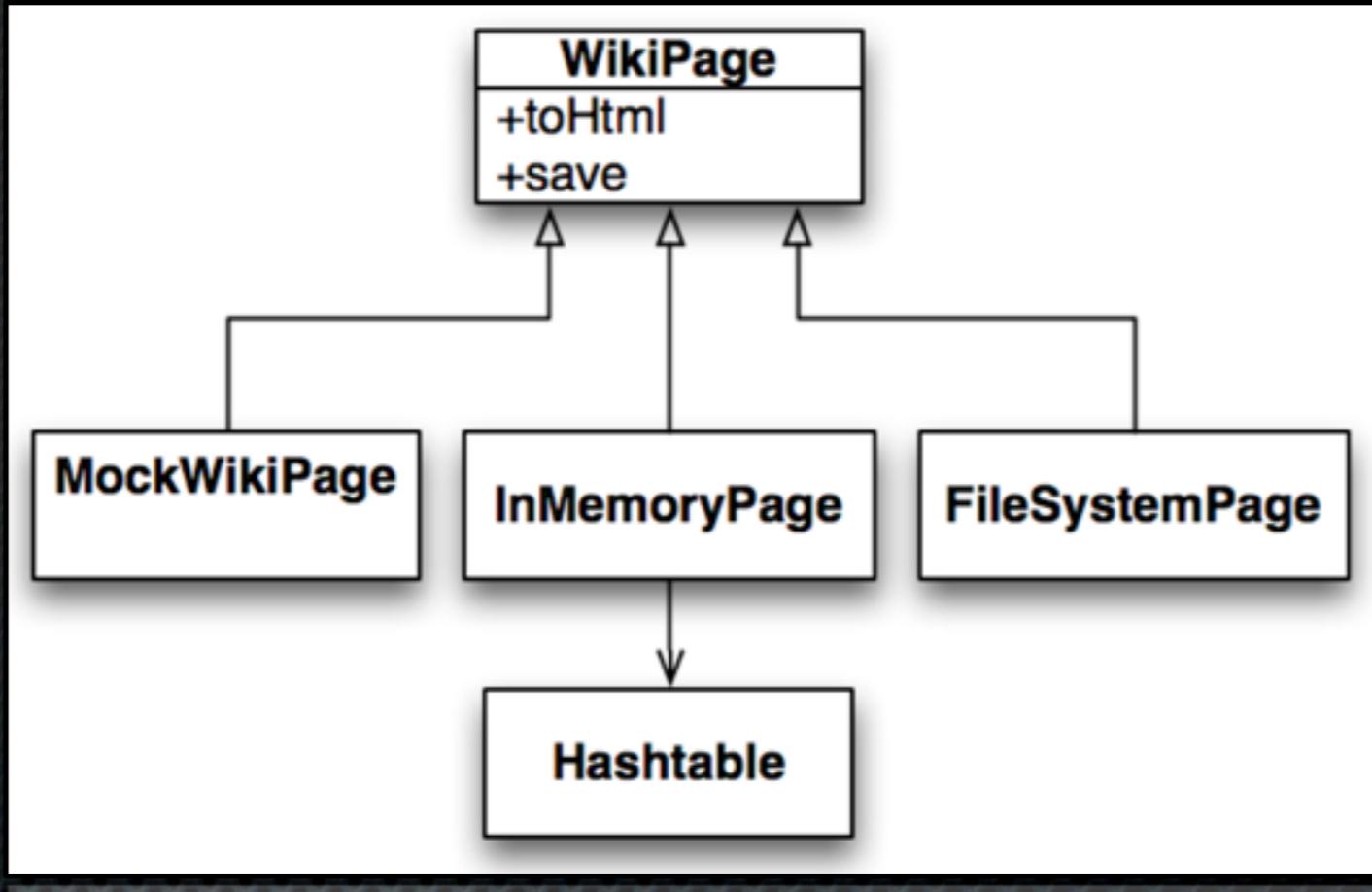
Deferring Decisions 예. Fitness

- 하나 이상의 페이지를 사용해야 하는 경우 발생
 - DB가 실제로 필요한 상황
- InMemoryPage Test Double



Deferring Decisions 예. Fitness

- 결국에는 persistence 없이는 구현할 것이 없어졌다.
 - 생각해 낼 수 있는 새로운 테스트는 항상 DB가 필요
 - DB를 기동하고, 스키마를 만들고, DB에 위키 페이지를 저장하고 읽는 기능을 구현
- FileSystemPage Test Double

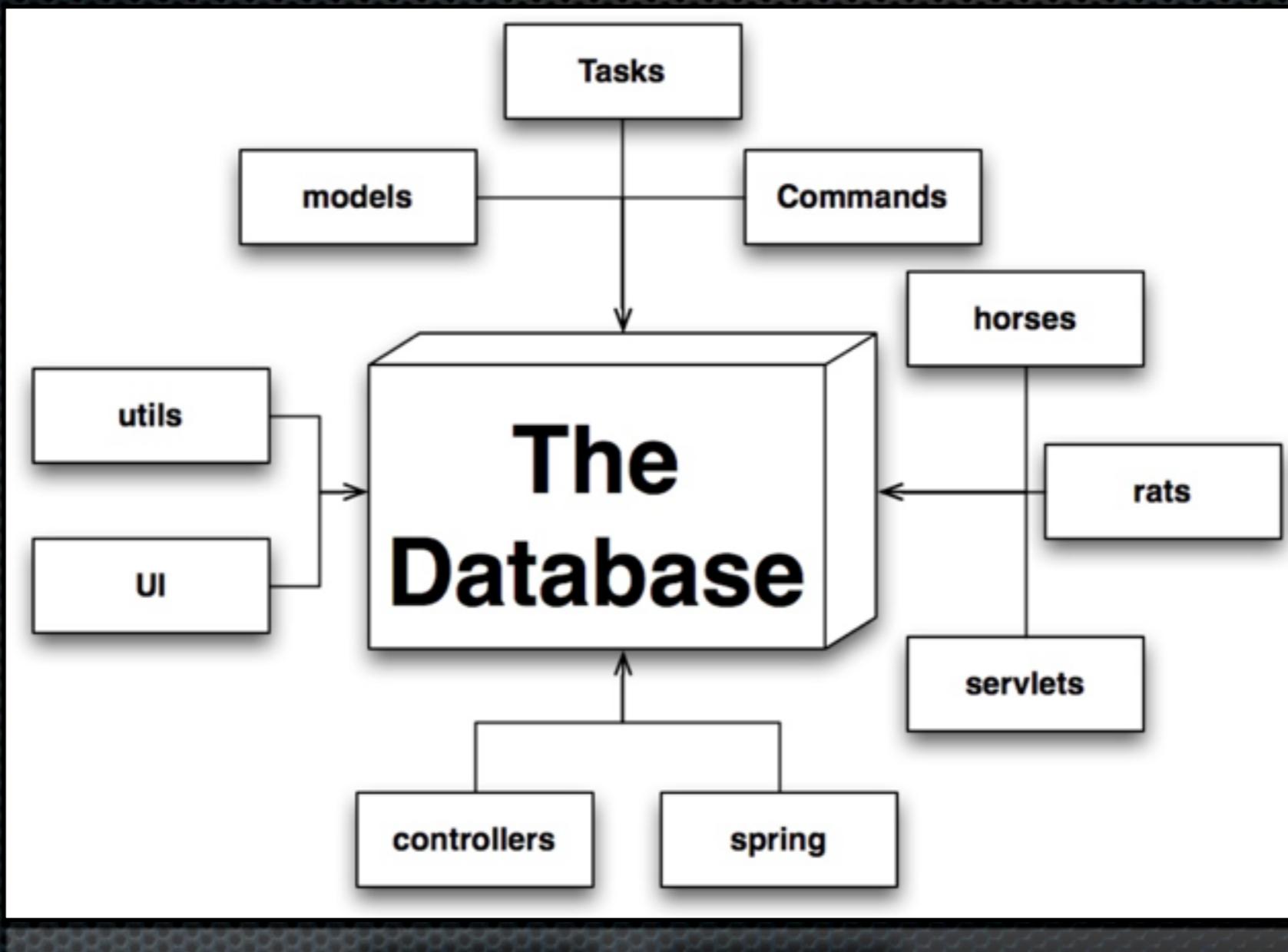


Deferring Decisions 예. Fitness

- Flat file system이 충분히 좋았기 때문에 MySQL은 사용하지 않았다.
- 주요한 아키텍처 결정을 연기했다.
 - 실제 DB는 사용 안 했다.
 - 더 간단하고 충분히 좋은 해결책이 있었기에.
- 최종적으로는 고객사에서 모든 데이터는 실제 DB에 넣어야 한다는 정책이 있어서 그렇게 했다.
 - MySqlPage를 추가

Central Abstraction

- 많은 아키텍트는 DB를 core abstraction이라고 생각
 - DB가 동작하고 스키마가 준비되기 전에는 어떠한 생각도, 작업도 시작하지 않는다.



Central Abstraction

- Fitness 프로젝트의 Central Abstraction - WikiPage
 - DB를 연기할 수 있는 Detail로 간주
 - 좋은 아키텍처는 Tool, FW로 구성되는 것이 아니다.
 - 좋은 아키텍처는 UI, WAS, DI FW 등과 같은 Detail에 대한 결정을 연기할 수 있도록 해준다.
- 어떻게 하면 이렇게 연기할 수 있나?
 - 연기하고 싶은 것에서 구조를 decouple, irrelevant하게 설계
- 어떻게 하면 Tool, FWs, DB에서 decouple하나?
 - Focus your architecture on the use cases not on the SW environments.

Conclusion

- 아키텍처를 Use case에 집중
 - UI, FWs, WAS, DB 등과 같은 다른 시스템 컴포넌트에 대한 결정을 미룰 수 있다.
 - 이러한 연기는 우리의 선택을 최대한 오래 열어 둘 수 있다.
 - 이 말은 필요에 따라 결정을 변경할 수 있다는 것이다.
 - 프로젝트 진행 중에 충분한 정보가 생김에 따라 undo에 대한 비용 없이 여러번 변경할 수도 있다.