Service Platform Aradon

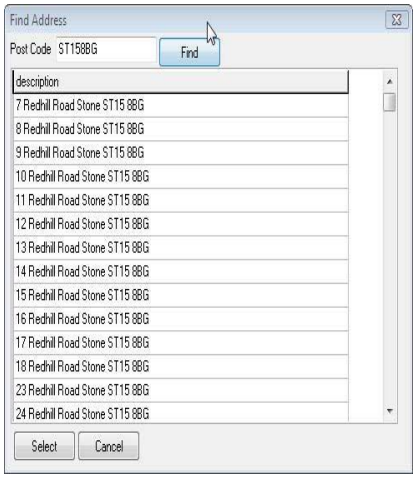
Aradon Core Concept

- Aradon is the framework & middleware platform for the next generation of asynchronous, effortlessly scalable, concurrent applications

- 비슷한 목적의 잘 알려진 프로젝트
  - Play
  - Node.js
  - Vert.x
  - BASS.io

Aradon을 만들기 전에

- 1998년 : I모 쇼핑사이트 제작 때
  배달 주소 등록 때 우편번호 검색이 필요함
  해야 하는 작업
  - 우편번호 테이블 만듦
  - 데이터 Batch로 입력
  - SQL과 프로그램 작성
  - View 화면 제작

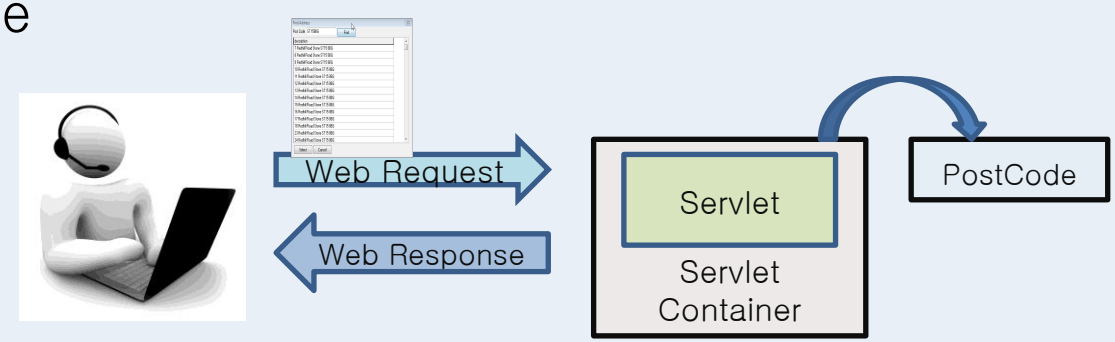


- 그런데 이게 끝이 아님…
  - 우편번호 관리(우편번호가 수정됨), 잘못 등록된 우편번호 발견
  - 새로운 우편번호가 지정됨, 과거 우편번호와 Data 충돌
  - 이걸 매번 사이트 만들 때마다 고려 ?

- 2012년 현재 : 우편번호 컴포넌트 ?
- 컴포넌트나 프레임워크가 아닌 서비스가 필요함

Aradon을 만든 후에

- Before



- After

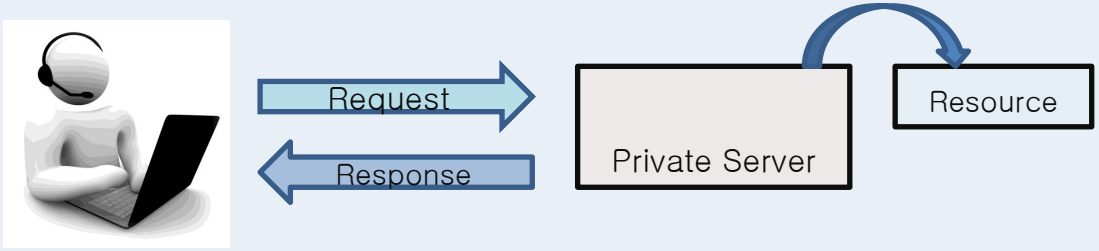
Aradon을 만들기 전에

- 2002년 : Java Applet 게임 만들 때
  - 게임마다 Hall Of Fame이나 기타 정보 등록이 필요함
  - 게임마다 서버를 구현?
  - 게임을 팔 때마다 서버를 구축?
  - 게임은 500원, 얼마나 팔릴지 모르는 상태에서 서버구입은 무리

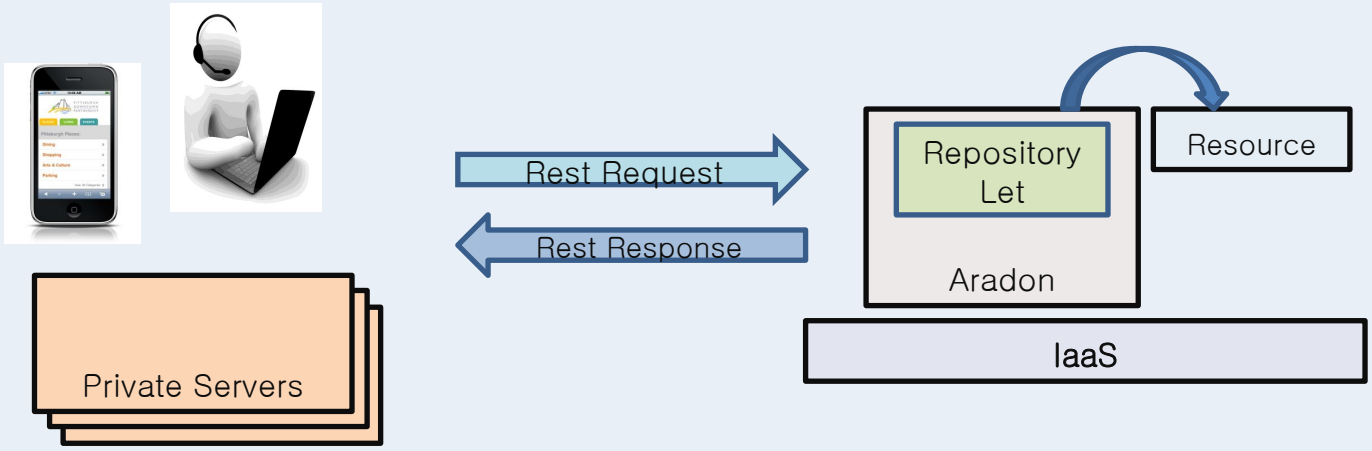
- 2012년 : Mobile App 게임 만들 때
  - 위의 문제 고스란히 상속
  - 별도의 서버 관리 비용이 부담
  - 안정적인 서비스 & 모니터링이 필요함
  - 유연한 확장이 가능해야 함.

Aradon을 만든 후에

- Before



- After



Aradon을 만들기 전에

- 2005년 : KTF WIPI 프로그램 관리툴 개발 때
  사용자 정보는 타 시스템에서 관리
  사용자 정보를 제공받을 수 있는 서비스가 없음,
  DBA는 보안상의 이유로 DB 계정 공유 불가 -> Scheduling Copy
  DW?(Data Warehousing)

- DB는 서비스가 아니라 Resource

- Resource를 다루기 위한 Service 필요
  별도의 Security 관리 정책(인증과 허가)
  서비스 제어 및 통제 필요
  그러나 DB처럼 사용할 수 있어야 한다.
  Multi-Tenancy

프로그래밍의 성배

- DRY(Don't Repeat, Yourself)

  Library(코드 중복)
  - 절차 지향 프로그램 언어(포트란 등의 고급언어)

  Component
  - 객체 지향 프로그램 언어
  - High cohesion & Low coupling

  Framework(구조적 중복 해결)
  - Request와 Response 사이에 일반적인 구조를 찾아낸다

  WebService(서비스 중복 해결)

Tranditional Architecture

Rigidity
- System is hard to change

Fragility
- Changes cause system to break and require other change

Immobility
- Difficult to entangle components that can be reused in other system

Viscosity
- Doing things right is harder than doing things wrong

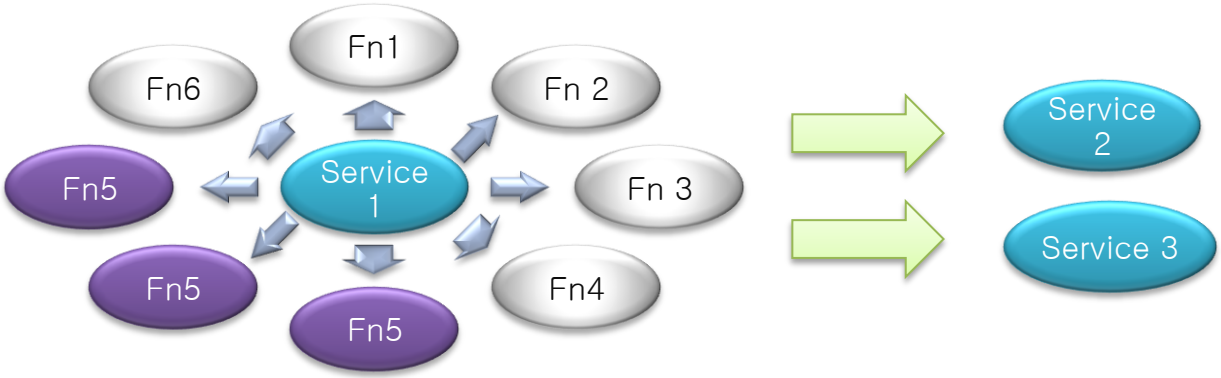
Needless Complexity
- System contains infrastructure that has no direct benefit

Needless Repetition
- Repeated structures that should have a single abstraction

Opacity
- Code is hard to understand(Not Easy, Must be Simple)

서비스를 만들자

- Multi-Tenancy가 되는 서비스
- 다양한 Device에서 접속 가능한 서비스
- 다른 서비스와 연계가 가능한 서비스(다양한 Protocol)
- 안정적인 쓰루풋과 확장이 가능한 서비스
- SNS -> Service aNd Service

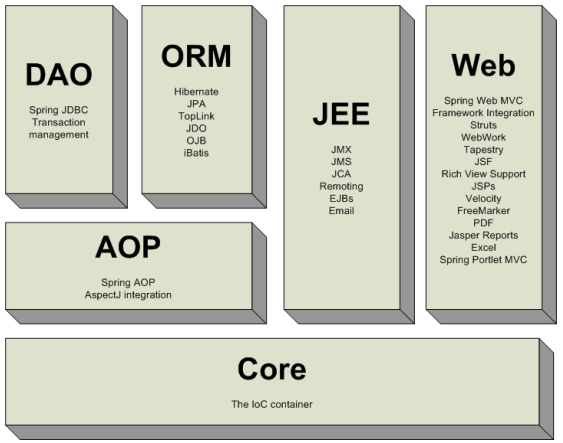


왜 Based 스프링이 아닌가?

- 불필요한 복잡함
  - 잡으려고 하는 몬스터(서비스)보다 잡을때 사용할 도구 사용이 더 복잡함
  - 자바의 복잡성은 문화적인 것이지 강제된 것은 아니다.

- 서비스의 비기능적 요구사항의 증가
  - 스프링은 비기능적 요구사항에 도움이 못됨

- 서비스를 사용하는 것은 사람이 아니라 다른 서비스임
  User View는 거의 없거나 별로 고려할 사항이 아님
  Multi-Tenancy는 중요함
  다양한 Device 고려



왜 Framework이 아닌가?

- Framework의 단점
  배우는 데 시간이 걸린다. (언제나 도움말은 충분치 못하다.)
  문제영역에 맞춘 코드가 아니라 프레임워크에 맞춰진 코드가 탄생한다.
  여러 프레임워크의 혼용 시 복잡도가 증가한다.(단일지점 제어의 혼란)

- Framework는 정글에 난 길과 같다.

- Framework와 Platform이 분리되면 Testability가 떨어지며 Deploy가 복잡해진다.
  개발, 배포, 그리고 운영을
  하나의 스펙트럼 안에서 처리하고 싶었다.
  (Aradon은 Framework인 동시에 Platform이다.)

- Framework vs Platform

왜 Based Servlet이 아닌가?

- 동적인 웹 페이지를 만들기 위해 나온 Servlet, JSP의 한계
  - Servlet Arch Model : State Model, Disconnected Web
  - 서블릿이 하기 어려운 문제들 :
    Connected Web, Streaming
    Distribute & Split Service(Not Cluster)
    데이터 직렬화(AVRO, Apache Thrift)

- Mobile Device의 확장에 따른 시장 환경
  - Required high scalability

- 배포와 테스트 곤란
  - Servlet Model은 Test 친화적이지 않다.
  - View와 너무 밀접하게 연관되어 있다.
  - Not Fun and Not Productive
  - WANT to reload service

왜 Node.js가 아닌가?

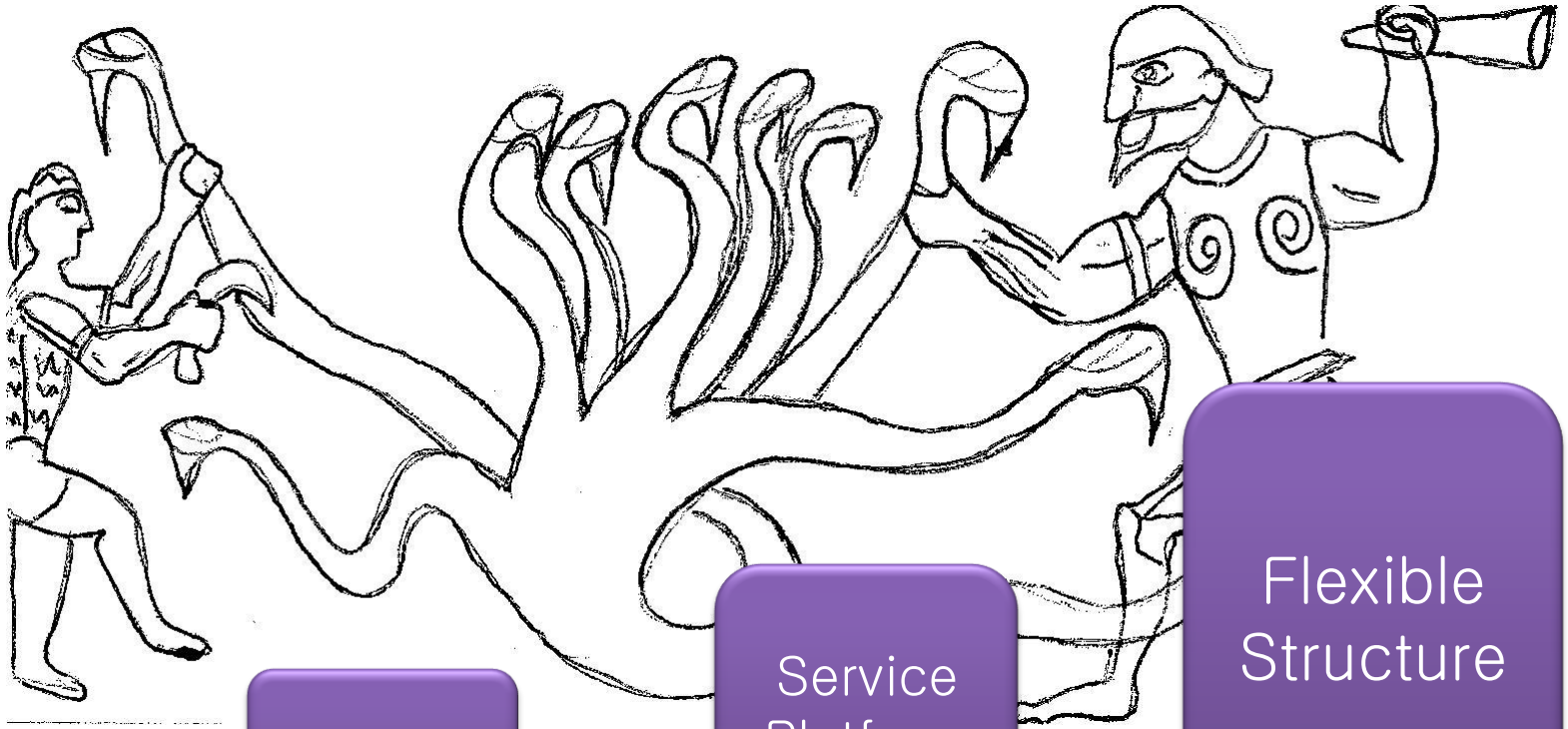
- I Need Control
  - Filter
  - Context
  - Multi Protocol on Port
  - Dynamic Handling

- Testability
  - Javascript 가독성
  - 하위 구조까지 접근할 수 없다.

- Javascript의 domain 한계
  - 멀티 쓰레딩
  - Closure 구조의 복잡성

- Aradon의 경우 EngineType에 따라 동기/비동기 선택

Aradon



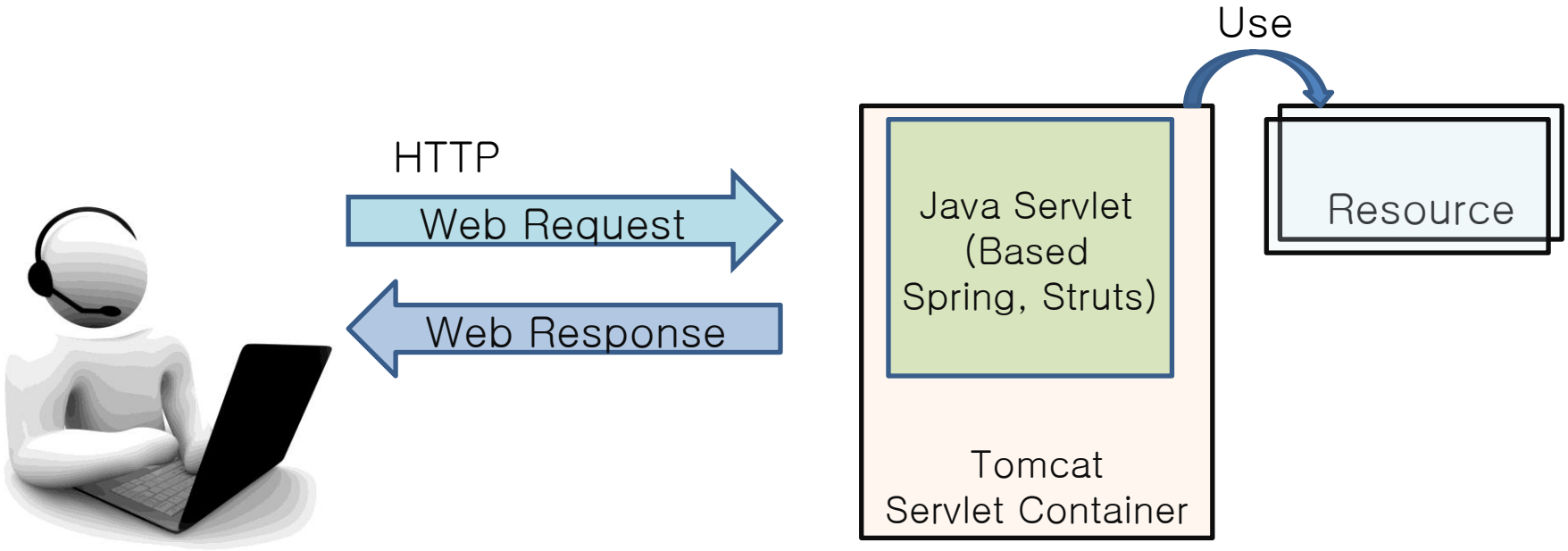
Easy Call

Service Platform

Flexible Structure

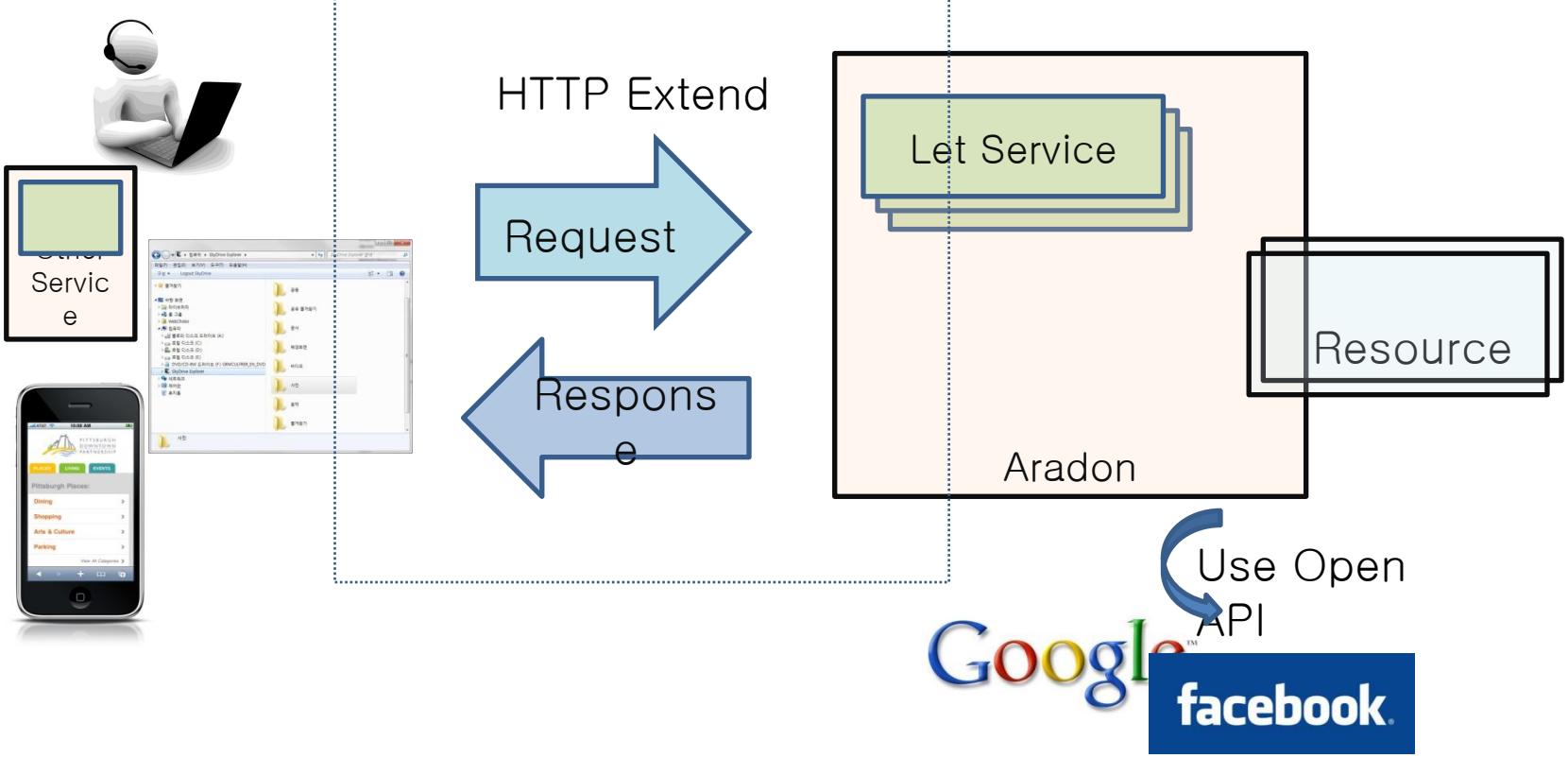
What Aradon ?

- Traditional WebService



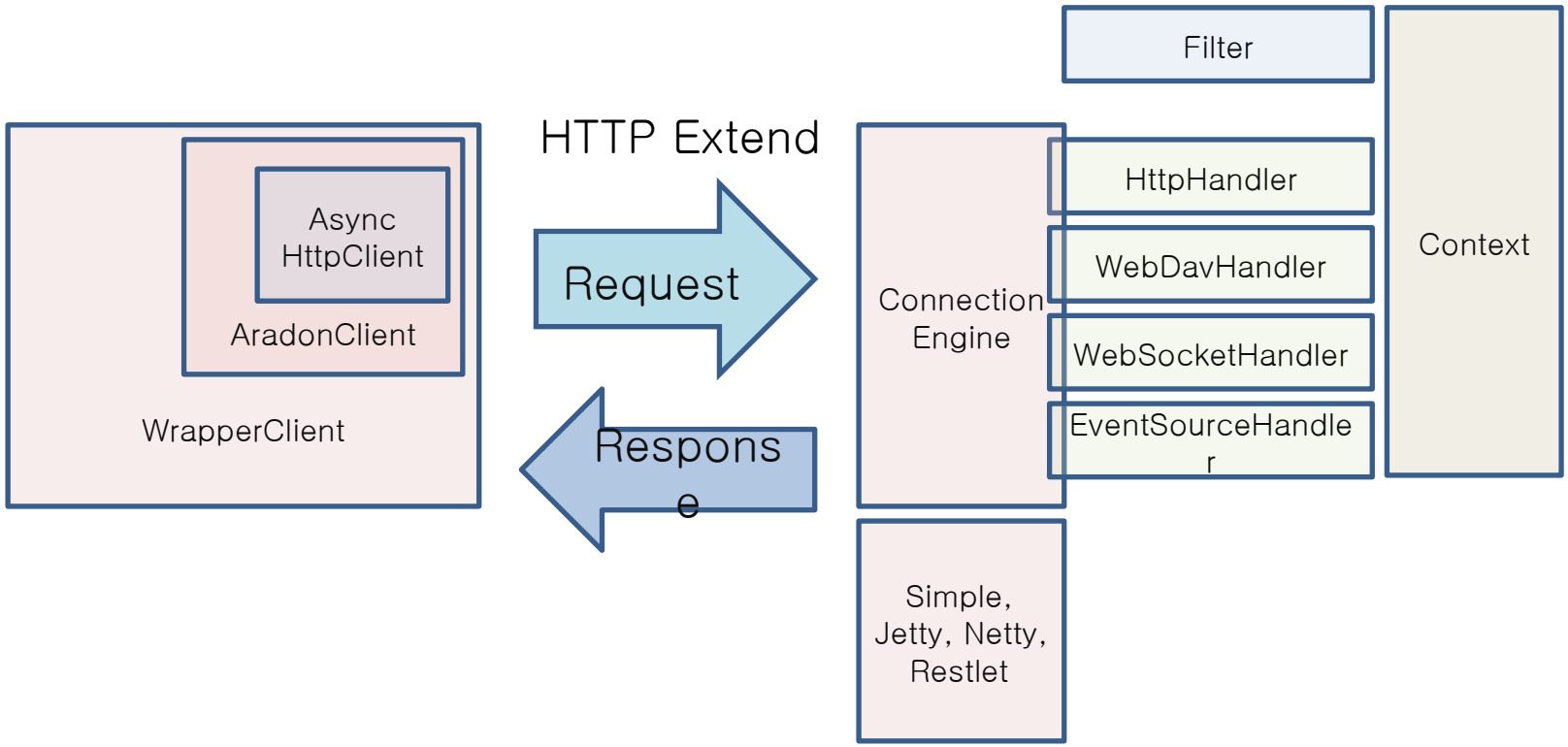
What Aradon ?

- Aradon



What Aradon ?

- Aradon



Aradon Key Feature

- Testability
- Scalability
- Multi tenancy
- From Software To Service

- Why use HTTP Protocol in Aradon
  - Servers and Clients in any language
  - Everyone already knows HTTP
  - Prior knowledge and tools

Aradon Download

- Egit 설치
  http://www.vogella.com/articles/EGit/article.html
  http://download.eclipse.org/egit/updates

- Download From gitHub
  AradonServer : https://github.com/bleujin/aradon
  AradonClient : https://github.com/bleujin/aradonClient
  AradonExtend : https://github.com/bleujin/aradonExtend

Hello World 만들기(실습)

- Let 만들기
  - parameter를 받아서 Hello {name} 출력
- Let Test
- Let Service(on Browser)
- Aradon
  /test/net.ion.radon.TestFirst

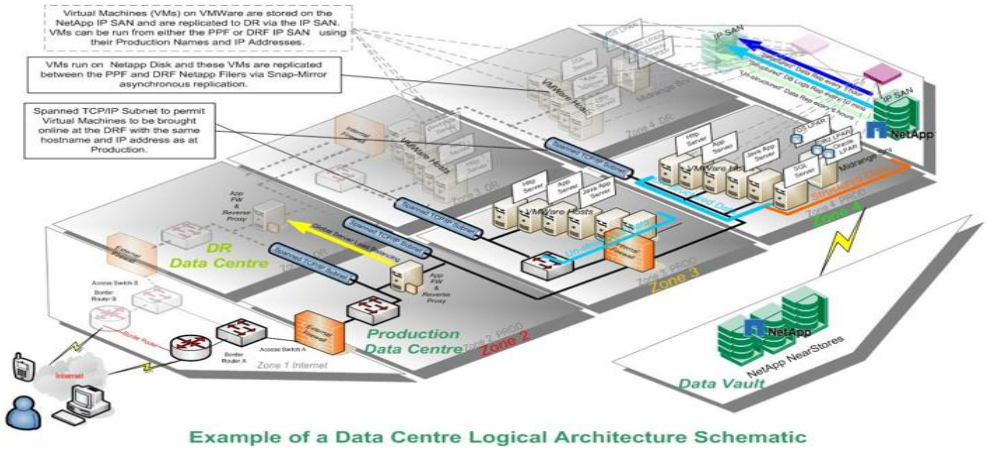
Chat 만들기(실습)

- WebSocket Handler 만들기
- WebSocket Client 로 접속하기
- StaticFile Handler 만들기
- Browser로 접속하기

  /WebSocketPlug
  /test/net.ion.chat.TestFirst

Repository 만들기(실습)

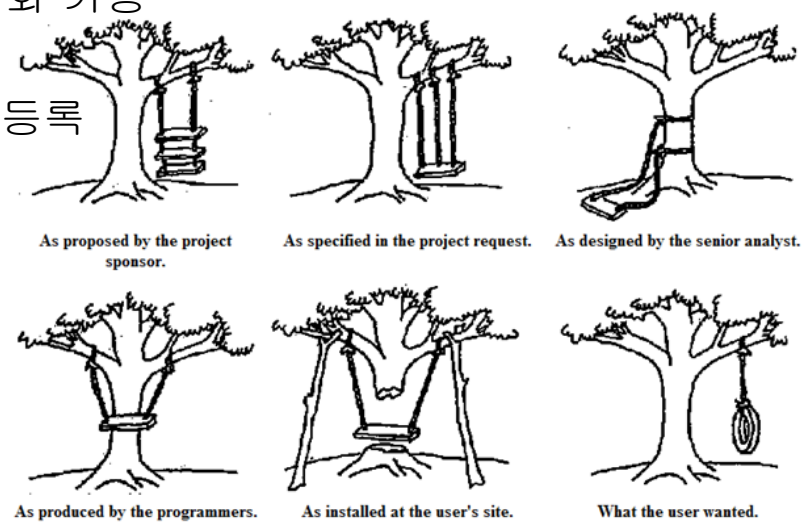
- RDB Service 등록
- Wrapper Client(MongoNode)
- /MongoSearch
  /test/net.ion.radon.repository.remote.rest.TestAmazonLet



Example of a Data Centre Logical Architecture Schematic

할 수 있는 것

- 서비스를 띄우지 않고 Test 가능
- 기존의 WAS에 Embed되어 Servlet REST Mode로 동작
- HTTP, WebDAV, WebSocket, EventSource 등의 다양한 Protocol 지원
- Streaming과 Request/Response Data 직렬화 가능
- War와 비슷하게 Zip 형태로 서비스 배포 및 등록
- 등록된 서비스의 통제 및 모니터링 등의 MetaService 가능
- 별도의 Java Developed Aradon Client



하고 싶은 것

- Aradon Agent
  Service Dynamic Deploy & ReDeploy
  Java Agent를 활용한 Debugging Mode Service 개발

- Repository Service(SaaS)
  Data, Chunk, Streaming

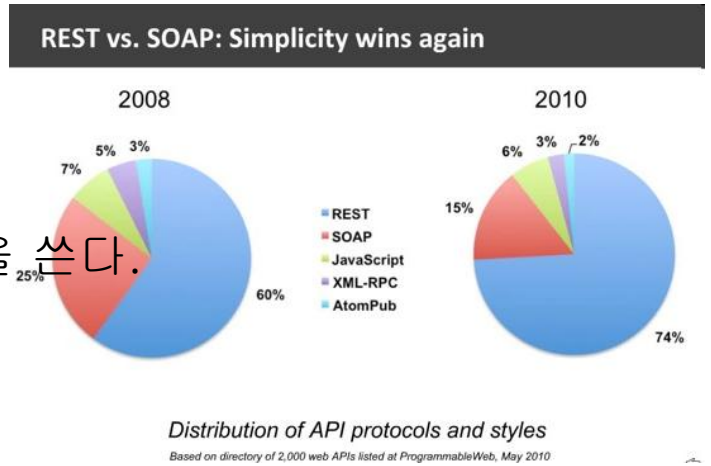
못 하는 것

- IaaS
  - AWS IaaS 사용

- 컴퓨팅 가상화 플랫폼 서비스
  - AWS나 KT의 가상화 플랫폼을 사용
  - 컴퓨팅 가상화와 관련된 OpenSource 활용

안 하는 것

- Servlet Container Service
  - 간단한 Servlet Container : http://winstone.sourceforge.net/
- View Template Service
  - 누구에게나 맞는 모자는 없다.
  - 그냥 Struts, Spring의 JSTL을 쓰거나 Velocity등 Opensource 이용
- Cluster Mode Server
- WSDL WebService
  아주 간단한걸 하기 위해 너무 복잡한 기술을 쓴다.
  Learning Curve가 높다.



왔다간거 다 알고있습니다.

댓글이라도 달아 주시지 그러세요~?

Reference

- GitHub

  ServerHome : https://github.com/bleujin/aradon

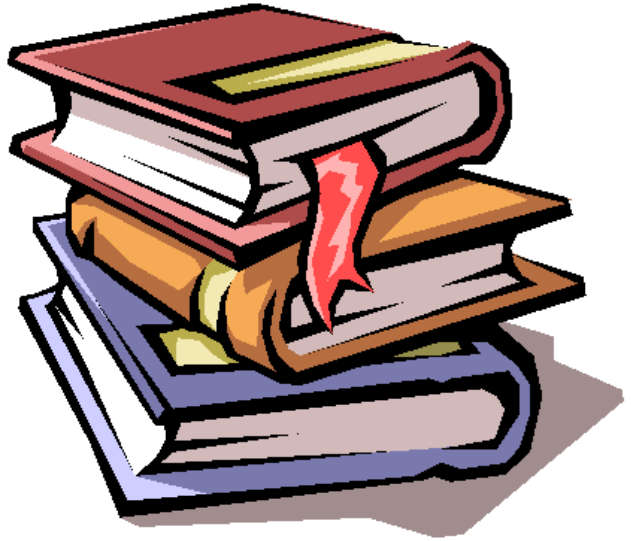
  ClientHome : https://github.com/bleujin/aradonClient

  ExtendHome : https://github.com/bleujin/aradonExtend

  doc : http://bleujin.springnote.com/pages/10761118

- AWS

  https://console.aws.amazon.com/ec2/home



Thank You