제 1절 분산 서비스 플랫폼(ARadon)

1. 배경

시장의 Fact :

* 구글과 MS 그리고 IBM, 아마존등에서 클라우드 서비스를 한다.
* 스마트폰 그리고 아이패드 등의 새로운 플랫폼이 뜨고 있다.
* 구글은 크롬이라는 브라우저를 개발하면서 이는 앞으로 제공되는 서비스의 클라이언트 플랫폼이 되기를 기대하고 있다.
* 저가 PC와 UMPC 노트북 업체들이 다시 인기를 끈다.
* 서비스 패키지 형태가 아닌 앱스토어(애플)와 웹스토어(구글) 형식의 서비스 프로바이더 형태가 보인다.

혹자는 소프트웨어 개발에 있어서 비용,품질,일정을 의자뺐기 게임에 비유하곤 합니다.

보통의 경우에는 의자가 한개, 운이 좋다면 의자 2개를 두고서 빙글 빙글 돌다가 먼저 앉는 게임말이죠.

서비스 개발에 있어서 비용과 품질,일정은 일반적으로 프로젝트의 트리플 제약 조건이 되며

종종 불가능해 보이는 세가지 목표를 모두 충족시키기 위해 CMMI 등의 조직의 프로세스 개선을 시도하려는 시도는 꾸준이 있어왔지만 많은 경우에 과거를 다시 한번 반복했을 뿐이었습니다.

1. 목적

* HTTP Protocol을 이용해 추상화된 형태의 서비스 브로커
* Process와 Repository의 분산
* 서비스 생태계(EcoSystem) 구축

1. 중요성

서비스 플랫폼의 초기모델을 만들고 주위 개발자들과 기획자들과 얘기했을 때 가장 많이 듣는 질문은 이 기능이 되는가 혹은 저 기능이 되는가 였는데 어느 순간 질문 자체가 ARadon에게는 맞지 않다는 걸 깨달았습니다.

대답은 항상 2가지로 대답할 수 있기 때문입니다. 1) 네 됩니다. 만들었으니까. 2) 네 됩니다. 만들면요

서비스 플랫폼에 있어서 어떤 기능이 되느냐는 관심사가 아닙니다. 중요한 것은 특정 기능을 왜 이 플랫폼을 통해서 만들어야 하는가? 입니다. 즉 서비스 플랫폼인 Aradon에게 있어서 요점은 이 플랫폼을 통해서 만들면 어떤 이점이 있는가가 중요한 것입니다.

그럼 Service 플랫폼을 통해서 만들면 무슨 장점이 있는가?에 대한 답을 하기 위해서 왜 우리는 어셈블러를 사용하지 않고 자바를 사용하는가와 비슷한 답변을 할 수 있습니다.

일반적으로 문제에 합당한 최고 고차원 언어를 사용하면 생산성과 품질이 높아집니다. 고차원이란 알다시피 명품 등의 Quality가 높다라는 의미가 아니라 추상성이 높은 언어를 고차원 혹은 고급언어라고 부릅니다. 전통적으로 어셈블러는 거의 기계어와 동일하므로 추상성이 낮기 때문에 저차원 언어라 부르고 기계어를 1GL, 어셈블리어를 2GL로 부릅니다. 많이 알려진 PL/1, C, Java등은 고급언어 혹은 3GL 언어라 불리며 4GL은 좀더 자연어에 가까우면서도 데이터베이스 접근을 위한 언어 PL/SQL 같은 언어를 4GL이라고 표현합니다.

고차원 언어에서 저차원 언어로 내려가는 이유는 무엇일까요?

기능이 첫째 이유입니다. 흔히 고차원 언어는 특정 문제 영역에만 초점을 맞추므로, 잠정적으로 새로운 문제에는 유용성이 떨어질 가능성이 있습니다.

효율성이 그 다음 이유입니다. 일반적으로 언어 수준이 높아질수록 컴퓨터 자원 효율은 떨어지며, 최악의 경우에는 감내하기 어려운 상태에 이르기도 합니다. 제안된 해결책은(이식성, 안정성, 인간 공학, 테스트 편의성, 이해 용이성, 수정 용이성 등) 반드시 품질 속성으로 평가되어야 합니다. 즉 품질 낮은 해결책으로 낭비하는 시간 비용이 값싼 프로그래밍 해결책으로 절약하는 비용을 넘어서선 안됩니다.

그럼 반대로 저차원 언어에서 고차원 언어로 올라가는 이유는 무엇일까요?

고차원 언어가 제공하는 장점은 많습니다.

1. 생산성이 높아진다. 가끔 놀랄 만큼 높아지기도 한다.
2. 고차원 언어는 오류가 발생할 기회를 제거하므로 전반적인 구현 수준을 높인다. 예를 들어, 어셈블러 프로그래머는 하드웨어 레지스터를 조작해야 하므로 레지스터 할당 오류를 저지를 위험이 있다. 고차원 언어 프로그래머는 레지스터 오류를 저지를 가능성이 없다. 고차원 언어에서는 레지스터를 참조하지도 못하고 참조할 필요도 없기 때문이다.
3. 고차원 해법은 코드 행수가 적다, 따라서 오류가 발생할 기회도 더 적다.
4. 고차원 언어로는 구조적 코드를 작성하기 쉽다. 흥미롭게도, 최고 고차원 해법은 코드가 구조적일 필요도 없다.
5. 고차원 언어로 작성한 코드는 이식성이 더 놓다.
6. 고차원 언어로 작성한 코드는 유지 보수성이 더 높다.
7. 고차원 언어로 작성한 코드는 테스트가 더 용이하다.

결론적으로 언어차원이 높아질수록 대다수 품질 속성이 좋아진다는 뜻 입니다. 즉 일반적으로 저차원 언어가 가지는 장점은 고차원 언어가 가지는 단점이 되고 그 역도 마찬가지입니다. 극히 효율적인 해법이 필요한 문제라면, 품질과 효율성 사이에 적절한 타협점을 찾을 때까지 언어수준을 낮추어야 합니다. 생산성과 품질은 좀처럼 함께 가지 못하지만 고차원 언어를 사용한다면 두 토끼를 모두 잡을 가능성이 커집니다.

물론 고차원 언어를 새로 정의하는 것이 만병 통치약이 되지는 못합니다. 첫째로 고차원 언어 자체가 가지는 오류나 코드 구조등에 문제가 있다면 오히려 생산성이 저하될수도 있습니다. 두번째로 새로운 고차원 언어를 배워야 하는 Learning Curve 시간입니다.

어쨌거나 이와 같은 관점에서 보면 ARadon은 Let 형태의 개별 서비스를 호출하는 사실 거의 최고 수준의 고차원 레벨의 언어를 사용합니다. 게다가 그 언어는 자체적으로 정한게 아니라 이미 정해져 있고(HTTP Protocol) 대부분의 사용자는 그 언어에 이미 익숙합니다. 언어의 메소드는 PUT, GET, POST, DELETE 이게 답니다. 그리고 대부분의 사용자가 GET, POST를 브라우저를 통해 매번 사용하고 있습니다. ARadon은 특정 URI(Address Key)를 PUT, GET, POST, DELETE 메소드로 호출함으로써 기능(서비스)을 사용할 수 하겠다는 뜻입니다.

ARadon은 고차원 언어가 가지는 저효율성의 문제를 분산이라는 다른 방식으로 해결합니다. ARadon을 사용할 때 PUT, GET, POST, DELETE 메소드와 URI가 전부라면 URI에 매핑된 서비스(혹은 기능)은 상세 구현에 대해 호출자에게 알려줄 사항이 거의 없기 때문에 분산으로 구현이 되어 저효율성의 문제를 상쇄시킬 수 있습니다.

ARadon에 올라가는 서비스는 자바로 구현되지만 그 추상성으로 인해 Product Independent, Language Independent, Platform Independent라는 세가지 독립성을 유지 할 수 있습니다. 다시 말해서 ARadon에 올라가는 서비스를 활용하는데 있어서 특정 제품이나, 언어, 무선이나 유선등의 플랫폼에 구애 받지 않음으로서 서비스의 재사용의 범위를 늘릴 수 있습니다.

1. 내용 및 범위

서비스 플랫폼(ARadon)은 기본적으로 세가지 기본 서비스를 제공합니다.

1. (분산) Repository
2. Search
3. Server Push

3가지 기본서비스는 다른 서비스를 만들기 위한 기초 서비스 역할을 합니다.

ARadon은 분산서비스를 위해 StandAlone 서비스 형태를 가집니다. 즉 톰캣이나 웹로직등에서 실행되는 특정 서비스가 아니라 톰캣이나 웹로직 그 자체에 비견될 수 있습니다. 우리가 WAS에 자바로 war 형태의 서비스를 만들어서 제공하는 것처럼 ARadon에 Let 형태의 서비스를 만들어서 서비스를 제공하는 형태를 가집니다.

ARadon은 세가지 기본서비스와 별개로 약간의 Example Let과 Help document를 포함하는데 재미있는건 Example의 구동과 Help document 자체도 Let 형태의 개별 서비스 형태로 구현될 예정입니다.

추가적으로 1-2개의 Killer Service 역할을 할수 있는 서비스가 추가될 예정입니다. 그 첫번째로 아마도 xpressengine 형태의 사이트 관리자 형태가 되리라 생각합니다.

1. 기대효과

혹자는 소프트웨어 개발에 있어서 비용,품질,일정을 의자뺐기 게임에 비유하곤 합니다.

보통의 경우에는 의자가 한개, 운이 좋다면 의자 2개를 두고서 빙글 빙글 돌다가 먼저 앉는 게임말이죠.

서비스 개발에 있어서 비용과 품질,일정은 일반적으로 프로젝트의 트리플 제약 조건이 되며 종종 불가능해 보이는 세가지 목표를 모두 충족시키기 위해 CMMI 등의 조직의 프로세스 개선을 시도하려는 시도는 꾸준이 있어왔지만 많은 경우에 과거를 다시 한번 반복했을 뿐이었습니다.

다른 방향으로 각기 도망가는 세마리의 토끼를 잡기 위해서 저는 휠러의 "전산학의 모든 문제는 또 다른 수준의 간접 층으로 해결할 수 있다." 라는 유명한 말을 토대로 좀 다른 시도를 해보기로 했습니다. (휠러는 추가적으로 "그러나 그러면 또 다른 문제가 생기는 것이 일반적이다" 라는 말도 덧붙였습니다. )

다른 시도라고는 하지만 코드의 중복을 해결하기 위해 라이브러리가 나왔고 구조의 중복을 해결하기 위해 프레임 워크가 나왔다면 서비스의 중복을 해결하기 위해 서비스 플랫폼이 필요하지 않을까? 라는 상식적인 생각에 기인합니다.

서비스 플랫폼이란 결국 각각의 서비스를 돌릴 수 있는 플랫폼 서비스(PasS)의 역할을 하는 서비스를 말하며 사실 그 자체로 몇 가지의 embed 서비스를 제공하기 때문에 요즘 Cloud 환경에서 말하는 PasS와 SaaS 중간 정도가 맞겠군요.

충분한 킬러 서비스를 올릴수 있다면 자신의 컴퓨터에서 직접 서비스를 제공하지 않고 단지 이용함으로서 자신의 비즈니스를 실행할 수 있게 됩니다. 즉 ARadon이 기대하는 최종 목적은 서비스를 만드는 게 아니라 이용하겠다는 Sass의 기본 철학과 상통합니다.