Agile Software Development

- 애자일 소프트웨어 디벨롭먼트
- 신속하고 변화에 유연하며 적응적인 소프트웨어 개발을 목표로 하는 다양한 경량 개발 방법론 전체를 일컫는 총칭
- 개발 대상을 다수의 작은 기능으로 분할하여 하나의 기능을 하나의 반복 주기 내에 개발하는 개발 방법
- 하나의 반복 기간을 프로젝트마다 다르지만 일반적으로 1주에서 4주 정도인 경우가 많다. 이 반복 주기를 계속해 나가면 하나씩 기능을 추가 개발하는 것이다.
- ref 애자일 개발방법론

Boilerplate code

- boilerplate code(보일러 플레이트) 또는 boilerplate(보일러플레이트)는 변경하는 것 없이 많은 곳에서 포함되어야만 하는 코드의 부분을 의미한다.

클라우드 네이티브 [Cloud Native]

- 단일 처리에 특화된 최소한의 서비스 단위로 개발을 하고, 이를 컨테이너 환경에서 실행하는 애플리케이션 개발 방식
- 수많은 컨테이너가 오케스트레이션을 통해 자동으로 관리, 운영하는 등 클라우드 네이티브는 클라우드 컴퓨팅 모델의 이점을 최대한 살린 개발방식

Dead Link

- **깨진 링크(broken link)** 또는 **죽은 링크(dead link)**는 영구적으로 이용할 수 없는 웹 페이지나 서버를 가리키고 있는 월드 와이드 웹 상의 링크를 말한다. 깨진 링크에 들어갈 때의 가장 일반적인 결과는 바로 웹 서버가 응답하지만 특정한 페이지를 찾을 수 없을 때에 나타나는 404오류이다. 브라우저는 웹 서버가 도메인 이름에서 찾을 수 없음을 가리키는 DNS오류를 반환하기도 한다ㅏ. 또, 콘텐츠 필터나 방화벽과 같은 차단 방식의 까닭으로 링크가 깨졌다고 보고하기도 한다
- 출처 깨진 링크

Design Pattern, 디자인 패턴

- 설계 문제의 해답을 문서화하기 위해 고안된 형식 방법이다. 과거의 소프트웨어 개발 과정에서 발견된 설계 노하우를 축적하고, 이름을 붙여 재사용하기 좋은 형태의 특정 규약으로 묶어 정리한 것이다. 코드로 바로 변환되는 형태는 아니지만, 특정 상황에서 구조적인 문제를 해결하는 방식을 설명해준다.(출처: 위키백과)

Docker

- Container-based Virtualization System [컨테이너 베이스드 버츄얼리제이션 시스템]
- 컨테이너 기반의 가상화 시스템

FinOps [핀옵스]

- Financial Operation 파이낸셜 오퍼레이션 - 비용 최적화
- 분산된 여러 팀이 속도, 비용, 품질 간에 비즈니스 균형을 이룰 수 있도록 가변적 지출 클라우드 모델에 재정적 책임을 부여하는 방식. 즉, 소프트웨어 엔지니어가 솔루션 비용을 책임지는 운영 모델이다.
- 기업이 클라우드 소비를 위한 지출 요건을 보다 잘 계획, 편성, 예측하는데 도움을 준다.
- 클라우드 비용 최적화 프랙티스...'핀옵스'에 쏠리는 관심

Framework

- “ 소프트웨어 개발 시 공통적으로 필요한 기능들을 재사용 관점에서 모아 구조화한 것 ” 으로 일관된 애플리케이션의 뼈대를 제공하며 필요에 따라 확장 가능한 기반 코드를 제공한다.

Functional Programming

- 펑서널 프로그래밍, 함수형 프로그래밍
- 자료 처리를 수학적 함수의 계산으로 취급하고, 상태와 가변 데이터를 멀리하는 프로그래밍 패러다임의 하나이다. 명령형 함수는 프로그램의 상태값을 바꾸는 부작용이 있을 수 있다. 때문에 참고 투명성이 없고 같은 코드라도 실행되는 프로그램의 상태에 따라 결과값이 다를 수 있다. 반면, 함수형 코드는 함수의 출력값은 그 함수에 입력된 인수에만 의존하므로 x에 값을 넣고 함수 f를 호출하면 항상 f(x)라는 결과가 나온다.(출처: 위키백과)

GraphQL

- Graph Query Language
- Server API를 통해 정보를 주고받기 위해 사용하는 Query Language
- GraphQL API는 주로 하나의 Endpoint를 사용한다.
- GraphQL API는 요청할 때 사용한 Query문에 따라 응답의 구조가 달라진다.

Hang

- hang 또는 freeze라고 하는 System hang이란, 아무런 반응을 하지 않는 상태로서 시스템 운영이 불가능한 상태를 의미한다.
- 정지(중단)된 상태를 말하는데, 커널 자신이 스스로 시스템의 문제를 인식하지 못하여, 결국은 시스템 마비를 일으킨 상태이다.
- System Hang과 Panic의 차이

IEEE 754

- IEEE에서 개발한 컴퓨터에서 부동소수점을 표현하는 가장 널리 쓰이는 표준이다. $+0$, -0 등의 수와 무한(Infinity), NaN (Not a Number)등의 기호를 표시하는 법과 이러하나 수에 대한 연산을 정의하고 있다.

In-Memory Computing

- 전체 데이터를 메모리에 적재하여 사용
- 컴퓨팅 시스템의 공식 기록 시스템(system of record)으로 하드 디스크를 사용하는 것이 아니라 메모리를 사용하는 것
- 데이터를 메모리에 보관,처리하고 하드디스크를 데이터의 안전하게 보관(stash)하기 위해 사용하는 구조
- stash[스태쉬- : (안전한 곳에) 넣어 두다[숨기다]
- 인메모리 컴퓨팅이란 무엇인가

log [로그, 운용 기록]

1. 기계 작동에 관한 기록, 기계 상황의 변경, 스위치 선택, 입출력 장치에 대한 사항, 제어 탁자에서 입력한 자료, 기계 정지 시의 상태나 원인 등에 관한 사항의 기록 등 컴퓨터 운용에 관계되는 모든 기록, 컴퓨터에 있는 로그 파일에 자동적으로 기록되게 할 수 있다.
2. 어떤 트랜잭션이 데이터베이스의 어떤 값을 갱신할 때 갱신 전의 값과 후의 값을 로그에 먼저 기록하는 것. 트랜잭션의 시작과 완료, 철회 등의 정보도 로그에 기록된다.

logging [로깅]

컴퓨터 동적 상태에 따른 변화를 시간에 따라 기록하는 것

Interpreter [인터프리터]

파이썬 코드를 입력한 후 엔터 키를 누르면 결과가 바로 나온다. 이처럼 코드를 한 줄 한 줄 실행하여 결과를 얻는 방식을 인터프리터(interpreter)방식이라고 부른다.

meme

- 인터넷 밈 (Internet Meme) 또는 줄여서 밈 (Meme)
- 인터넷에서 유행하는 특정 콘텐츠와 문화 요소를 이르는 망

Proxy Server [프록시 서버]

- 시스템에 방화벽을 가지고 있는 경우 외부와의 통신을 위해 만들어 놓은 서버
- 프록시 (Proxy)란 원래 '대리'라는 의미를 가지고 있음
- 클라이언트가 프록시 서버를 통해서 다른 네트워크 서비스에 간접적으로 접속할 수 있게 해주는 컴퓨터
- 시스템에 방화벽을 가지고 있는 경우 외부와의 통신을위해 만들어 놓은 서버
- 프록시 서버는 프록시 서버에 요청된 내용들을 캐시를 이용하여 저장
- 프록시 서버 (Proxy Server)

pseudo-code[수도코드]

- 의사(疑 비교할 의 / 似 비슷할 사 | Pseudo : 가짜의)
- 컴퓨터 프로그램이나 알고리즘이 수행해야 할 내용을 우리가 사용하는 언어(한국어 또는 영어 등)로 간략히 서술해 놓은 것을 말한다.
- 특정 프로그래밍 언어로 구현하기 이전에 일반 언어로 작성한 간단한 버전의 프로그래밍 코드

QD, Quantum Dot [퀀텀 닷]

- 양자점[量子點]은 wlfmadl 2~10nm(나노미터: 10억분의 1m의 길이]에 불과한 반도체 입자로 특이한 전기적, 광학적 성질을 지닌 입자
- 퀀텀닷 기본원리

Refactoring [리팩터링]

- 결과 변경 없이 코드 구조를 재조정하는 방법.
- 버그를 없애거나 새로운 기능을 추가하는 것이 아니라 코드 가독성과 유지보수성을 높이는 행위를 말한다.

REST, RESTful

- REST : REpresentational State Transfer

- 모든 Resource(자료, User,...)들을 하나의 Endpoint에 연결해놓고, 각 Endpoint는 그 Resource와 관련된 내용만 관리하게 하자는 방법론
- 이런 REST의 조건을 만족하는 API를 RESTful API라고 부르고, 이런 방식으로 API를 작성하는 것을 RESTful하다고 한다.
- GraphQL과 RESTful API

Sandbox [샌드박스]

- 보호된 영역 안에서 프로그램을 작동시키는 보안 소프트웨어
- ITWorld 용어풀이 | 샌드박스(SandBox)

Snippet

- 스니핏
- 웹 사이트 디자인에서 정형화된 코드가 자주 사용되는데 이를 별도로 저장해 두었다가 필요할 때마다 복사하여 사용하는 것을 말한다.
- 재사용 가능한 소스 코드, 기계어, 텍스트의 작은 부분을 일컫는 프로그래밍 용어이다. 사용자가 루틴 편집 조작 중 반복 타이핑을 회피할 수 있게 도와준다.(위키백과)

SOLID 원칙

- 로버트 마틴이 정립한 객체 지향 프로그래밍 및 설계의 다섯 가지 원칙을 마이클 페더스가 새로 정리한 원칙

1. S : SRP [Single Responsible Principle, 단일 책임 원칙]
    - 한 클래스는 하나의 책임만 가져야 한다.
2. O : OCP [Open/Closed Principle, 개방-폐쇄 원칙]
    - 소프트웨어 요소는 확장에는 열려 있으나 변경에는 닫혀 있어야 한다.
3. L : LSP [Liskov-substitution Principle, 리스코프 치환 원칙]
    - 객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스로 바꿀 수 있어야 한다.
4. I : ISP [Interface Segregation Principle, 인터페이스 분리 원칙]
    - 특정 클라이언트를 위한 인터페이스 여러 개가 범용 인터페이스 하나보다 낫다.
5. D : DIP [Dependency Inversion Principle, 의존관계 역전 원칙]
    - “ 추상화에 의존해야지 구체화에 의존하면 안 된다 ” 는 원칙을 따르는 방법의 하나

Statically typed Language (정적 타입 언어)

정적 타입 언어 (Statically typed Language)는 컴파일 시 변수의 타입이 결정되는 언어를 말합니다.
따라서, 프로그래머가 변수에 들어갈 값의 형태에 따라 직접 변수의 타입을 명시해줘야 합니다.
정적언어라는 것은 위에서 언급한 '타입' 즉 자료형을 컴파일 시에 결정하는 것 입니다.

컴파일 시에 자료형에 맞지 않는 값이 들어있으면 컴파일 에러가 발생합니다.
정적 타입 언어로는 Java, C, C++, C#, Scala, Fortran, Haskell, ML, Pascal 등이 있습니다.

장점:
타입 에러로 인한 문제점을 초기에 발견할 수 있어 타입의 안정성이 높음.
컴파일 시에 미리 타입을 결정하기 때문에 실행속도가 빠름.

단점:
매번 코드 작성 시 변수형을 결정해줘야 하는 번거로움이 있음.

Dynamically typed Language (동적 타입 언어)

동적 타입 언어 (Dynamically typed Language)는 컴파일 시 자료형을 정하는 것이 아니라 런타임 시 결정됩니다.

tokenizer [토크나이저]

- tokenization [토크너제이션] : 토큰화
- 문자열을 여러 개의 조각, 즉 여러 개의 Token(토큰)들로 쪼개는 것을 말한다.
- Token은 문자열의 한 조각으로 하나의 단어가 하나의 토큰이라고 할 수 있다.

Use Case

- 유즈 케이스
- 시스템의 동작을 사용자의 입장에서 표현한 시나리오이며, 시스템에 관련한 요구사항을 알아내는 과정이며, 소프트웨어 개발 프로세스 중 개발을 위한 소프트웨어의 기능을 개략적으로 설명이 가능한 방법이다.
- 유즈 케이스는 시스템을 분석하는 사람과 사용할 사람이 함께 시스템의 사용 방법을 결정하는데에 도움을 준다. 작성 시 주의할 점은, 시스템을 개발하는 사람보다는 사용자의 입장에서 이해하기 쉽고, 간단하게 작성되어야 한다.
- ref 유즈케이스(Use Case)에 대하여

Upper Case, Lower Case

- Uppercase [어퍼케이스] : 대문자 = capital letter [케피를 레러]
- Lowercase [로어케이스] : 소문자 = small letter [스몰 레러]
- 대문자를 upper case, 소문자를 lower case라 부르는 이유

Wireframe

- 와이어 프레임

- 레이아웃과 UI 요소들에 대한 뼈대를 제공하는 그림
- PENCIL PROJECT

오션, IT, 용어

From:
http://rwiki.repia.com/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다. - 2023.12**

Permanent link:
http://rwiki.repia.com/doku.php?id=wiki:user:emblim98:terms&rev=1621300200

Last update: **2022/03/10 19:52**