

# Python으로 쿠키런 운영하기

흔한 1년차 개발자의 Python 사용기



Devsisters Corp. 이상곤

# 흔한 1년차 개발자의 Python 사용기

## 질문

난생 처음 들어간 회사

Production에서 쓰이는 기능을 개발할 수 있을까?

문법만 알던 Python

얼마나 빠르게, 얼마나 많은 성과를 낼 수 있을까?

# 강연자 소개

## 이상곤



2009.02~현재

KAIST 전산학과 학사과정



2013.07~2013.08

네이버 비즈니스 플랫폼 인턴



2013.09~현재

데브시스터즈 서버개발팀

## 쿠키런 소개



- 5,500만 누적 다운로드
- 최대 1,000만 DAU
- 한국, 일본, 태국 등 10여개국 다운로드 1위
- Top10 다운로드 국가 수 38개국



# 목차

---

01 산타맛 쿠키 보유 유저를 찾아라!

---

02 Celery를 이용하여 10분만에 전체 사용자 푸시보내기

---

03 게임 데이터 관리, 단번에 해결하기

---

01

**산타맛 쿠키 보유 유저를 찾아라!**



## 산타맛 쿠키?

- 2013년 크리스마스 이벤트 보상으로 지급된 쿠키
- 황금종  을 300개 모아야 얻을 수 있는 쿠키

# 이미 산타맛 쿠키를 가진 유저가?!

- 업데이트 준비가 거의 완료된 상황
- 뒤늦게 생각난 어뷰저의 존재
  - ✓ 어뷰징으로 이미 산타맛 쿠키를 가진 유저가 있다!
  - ✓ 이전에 로그 분석을 통해 산타맛 쿠키를 가진 어뷰저가 있다는 사실을 알았음
- 어뷰저들을 영구 정지 or 산타맛 쿠키만 삭제?
  - ✓ 어뷰저들이 가진 산타맛 쿠키를 삭제하는 방법을 선택



# 전략

# 1

- 일단 산타맛 쿠키 보유 유저의 목록을 뽑아내자!
  - ✓ Couchbase View를 사용하여 산타맛 쿠키 보유 유저의 목록을 추출

# 2

- 약 300명이 산타맛 쿠키를 보유함을 확인
  - ✓ Couchbase 웹 콘솔로 한 명씩 처리하긴 너무 많고..



# 전략

3

- Python에 Couchbase 라이브러리가 있음
  - ✓ Couchbase View의 쿼리 결과를 Python에서 iterator로 가져올 수 있음
  - ✓ 쿼리 결과를 코드로 직접 가져오니 실수할 위험도 적음

4

- 쿼리된 회원번호로 사용자의 인벤토리를 가져오자
  - ✓ 사용자 정보는 Couchbase에 JSON 형태로 저장되어 있음
  - ✓ Python에서 dict type으로 가져오므로 사용자 정보를 바로 수정 가능

5

- 인벤토리를 검색하여 직접 산타맛 쿠키를 제거

```
if 사용자정보[인벤토리][아이템 번호] == 산타맛 쿠키:  
    산타맛 쿠키 제거
```

# 속전속결!

- 1시간 안에 모든 산타맛 쿠키 제거 성공

- ✓ 약 300명의 어뷰저로부터 산타맛 쿠키 제거

- 간단한 일이지만 1개씩 지우자니 애매하고..

- ✓ 300개의 산타맛 쿠키 1개씩 지우기 vs 30줄의 스크립트 작성하기

- 자동화시킬 수 있는 스크립트를 만들자. 그런데 어떤 언어로?

- ✓ 다른 언어를 사용했더라면 이렇게 쉽고 빠르게 해결할 수 있었을까?  
✓ \$ pip install couchbase

02

**Celery를 이용하여 10분만에  
전체 사용자 푸시보내기**



## 쿠키런 전체 사용자에게 푸시를 보내자!

- 유효한 푸시 토큰의 갯수 약 1,500만 개  
(쿠키런 for KAKAO 기준)
- 짧은 시간 안에 푸시가 모든 사용자들에게 전송되어야 함

# 요구 사항

- 빠른 시간 안에
- 전체 사용자에게 푸시를 전송
- 서버 개발자의 개입없이 누구나 사용할 수 있도록

# 계획

1

- Celery를 사용
  - ✓ 여러 개의 worker로 동시에 작업을 처리하도록

2

- 웹으로 만들어서 접근이 편리하도록
  - ✓ Django로 구현된 기존의 운영툴에서 사용할 수 있도록

3

- 정해진 시간에 푸시 전송을 예약할 수 있도록
  - ✓ Celery task가 실행될 시간을 게임 운영자가 설정할 수 있도록

# Celery로 간단하게!

- Celery 인스턴스를 정의하고

```
from celery import Celery

from django.conf import settings

app = Celery('macaron')

app.config_from_object('django.conf:settings')
app.autodiscover_tasks(lambda: settings.INSTALLED_APPS)
```

- 원하는 기능에 task() decorator를 붙여 task 구현

```
@app.task
def increase_aws_instances():
    """Double AWS instances"""
    cookierun_aws_client = CookierunAWSClient()
    aws_instance_num = cookierun_aws_client.get_instance_num()
    cookierun_aws_client.scale_up(aws_instance_num)
    cookierun_aws_client.close()
```

- 대기 중인 task들을 저장할 Queue만 지정해주면 끝!

# 그리고 만들었다

- 웹에서 메세지 내용과 전송시간을 입력하면 task가 예약됨

- ✓ AWS EC2 인스턴스를 2배로 늘려주는 task 1개
- ✓ 푸시를 보내는 task를 생성해주는 task 1개



Redis

Task: Double EC2 instances

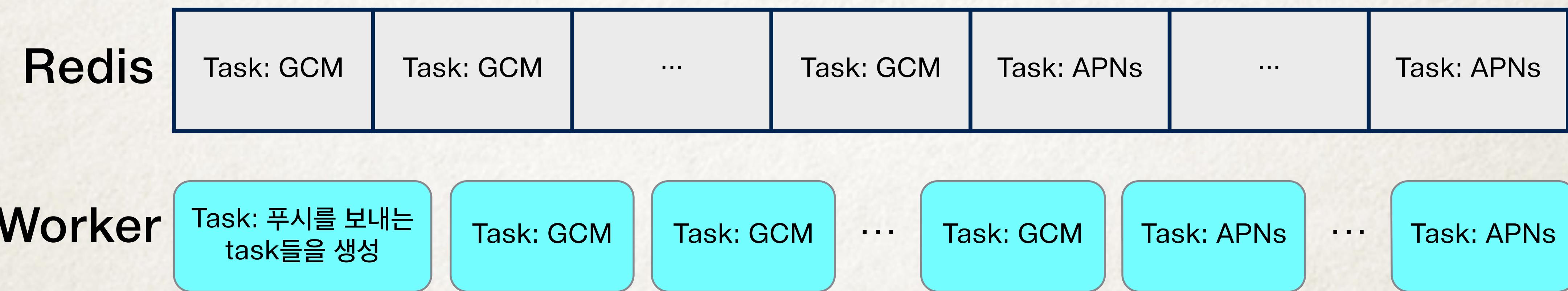
Task: 푸시를 보내는 task들을 생성

# AWS EC2 Instance 수를 2배로

- 사용자 급증에 대비하기 위해 게임 서버의 수를 늘림
  - ✓ “지금 접속하시면 크리스탈 100개를 드려요”라는 푸시가 전송된다면?
- 푸시 전송이 시작되기 전에 미리 늘려두어야 함
  - ✓ 푸시가 전송되기 10분 전에 task를 실행
- boto를 사용
  - ✓ 현재 구동되고 있는 서버의 수를 파악하여, 동수를 더 실행시킴

# 푸시를 보내는 task를 등록

- 푸시 토큰을 1000개씩 잘라서, 실제로 푸시를 보내는 task를 등록하는 역할
- OS별로 푸시 토큰을 구분하여 GCM, APNs 따로 보내도록 함
- 푸시를 보내는 task는 worker들이 바로 가져가서 처리하는 방식



# 성과

- 10분만에 약 1,500만 개의 푸시 전송 가능
  - ✓ Celery worker 수를 늘리고, 좋은 EC2 인스턴스를 사용하면 성능 향상 가능
- 서버 장애를 일으키지 않는 안전한 푸시 전송
  - ✓ “지금 접속하시면 크리스탈 100개를 드려요”라는 푸시가 10분만에 전체 사용자에게 전송된다면?
  - ✓ 무엇보다 게임 서버의 안정성이 우선!
- 서버 개발자 없이도 푸시 전송 가능
  - ✓ 메세지와 전송 시간만 입력하면 누구든 전송 가능



03

게임 데이터 관리, 단번에 해결하기



## 당시 상황

- 엑셀 파일로 관리되는 게임 데이터
- 게임 데이터의 수정이 필요할 경우
  1. 엑셀 파일에서 값 수정
  2. DB 접속툴로 MySQL에 접속
  3. 해당하는 테이블에 엑셀 시트를 반영
  4. 이 과정을 모든 엑셀 시트에 대해 반복!

# 이거 계속 해야 하나요..?

- 데이터시트 반영은 단순 반복 작업

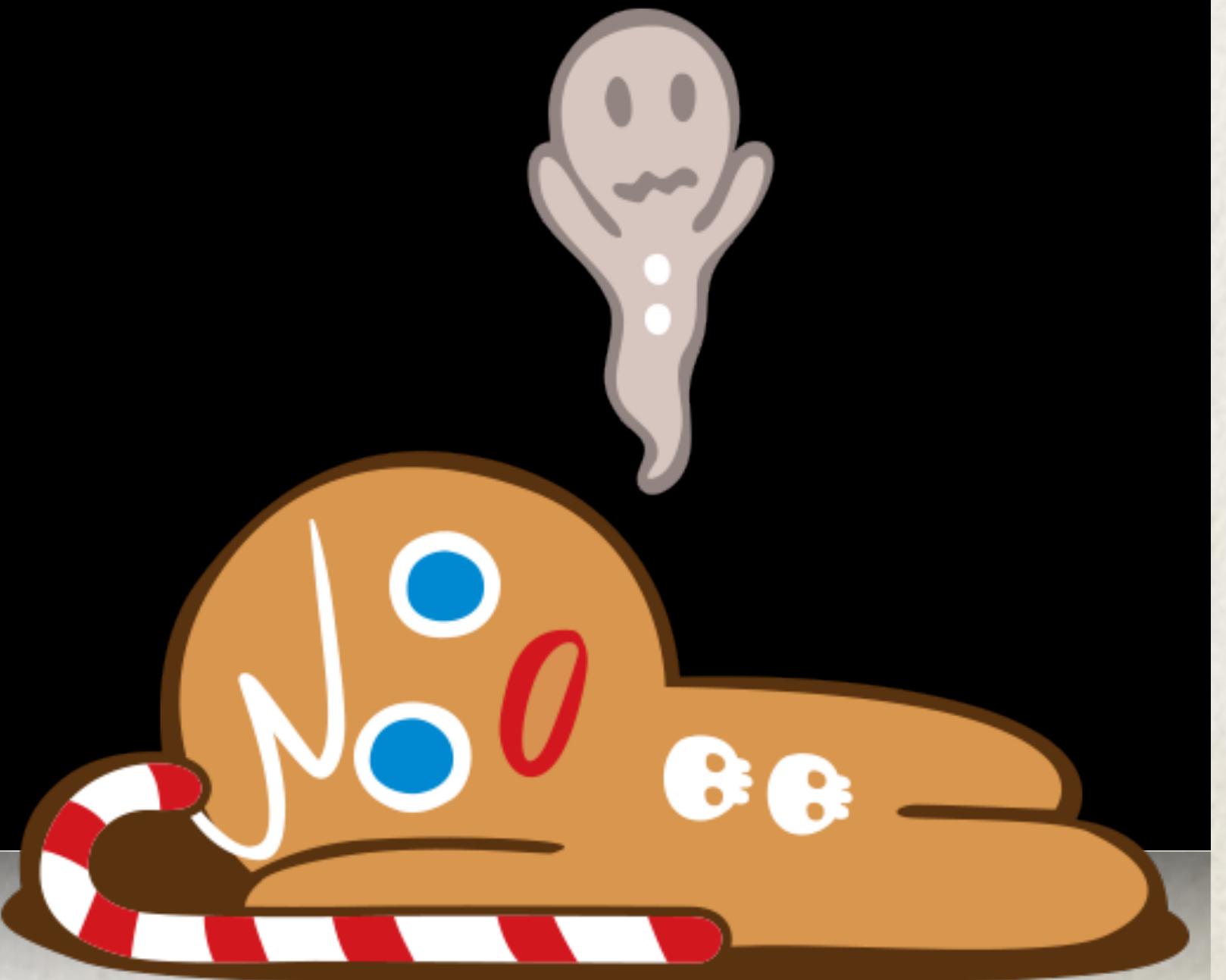
- ✓ 반영해야할 시트가 늘어날수록 작업량 증가

- 데이터를 잘못 넣었네..

- ✓ 수동으로 넣다보니 실수할 가능성 증가

- 그렇다면 자동화할 수 있는 툴을 만들자

- ✓ 시간도 절약하고
  - ✓ 실수도 줄이고



# 사전 조사

## • 요구 사항을 파악

- ✓ 엑셀 파일의 값을 MySQL에 적용시킬 수 있어야 함
- ✓ MySQL에 저장된 값을 엑셀 파일로 뽑아낼 수 있어야 함

## • 최대한 사용하기 쉽게 만들자

- ✓ 서버 개발자의 도움없이 자유롭게 데이터를 관리할 수 있도록
- ✓ 많은 시트를 한꺼번에 넣을 수 있도록

## • 엑셀 파일을 다루는 Python 라이브러리가 있을까?

- ✓ OpenPyXL



# 그리고 만들었다

- Django로 구현된 기존 운영툴에 붙임
  - ✓ 웹으로 운영되므로 누구나 쉽게 이용 가능

### Datasheet Manager - Import

Import Export

Sheet Name	Uploaded Date	Delete
		<span>Delete</span>

**Errors**

No datasheet file!

Import

**Difference**

Same! :p

### Datasheet Manager - Export

Import Export

<input checked="" type="checkbox"/>	Name	MAINLOCAL Rows	SUBLOCAL Rows
<input type="checkbox"/>	Table1	5	5
<input type="checkbox"/>	Table2	5	5
<input type="checkbox"/>	Table3	7	7
<input type="checkbox"/>	Table4	100	100
<input type="checkbox"/>	Table5	74	77
<input type="checkbox"/>	Table6	8	0
<input type="checkbox"/>	Table7	17	17
<input type="checkbox"/>	Table8	2	2

MAINLOCAL SUBLOCAL Export

# Import

- 엑셀 시트의 내용을 MySQL에 반영

- Diff를 출력

- ✓ 엑셀 파일을 브라우저에 끌어다 놓으면 DB와 시트의 차이점을 보여준다.
- ✓ Git의 diff 명령에서 착안

## Difference

TABLE1 +2 -0

	col1	col2	col3	col4
+	1	110	TEST1	3
+	2	120	TEST2	4

TABLE2 +2 -2

	col1	col2	col3	col4	col5
-	1	AA	0	100	200
-	4	B	4	130	None
+	4	B	5	130	None
+	12	QT	1	300	200

TABLE3 +0 -2

# 운영툴이 시트 이름을 구분할 수 있다?

- 대부분의 시트 이름은 Django model명과 동일
- 일부 시트의 경우 model별로 시트 이름에 대한 정의가 필요

- ✓ 하나의 model에 엑셀 시트가 여러 개 들어가는 경우
- ✓ 부득이하게 시트명을 바꿀 수 없는 경우

```
# model 이름과 sheet 이름이 다를 경우에 대한 mapping을 저장.  
# 혹은 Stuff 테이블처럼 실제 시트를 조개해서 관리하는 경우 이를 처리해주는 데이터구조  
MODEL_SHEET_MAP = {  
    cookierun.models.Stuff: {  
        u'EquipmentItemSeoul': lambda x: x.exclude(stuff_type__in=['T', 'M', 'AT']),  
        u'TreasureItemData': lambda x: x.filter(stuff_type='T'),  
        u'Material': lambda x: x.filter(stuff_type='M'),  
        u'Artifact': lambda x: x.filter(stuff_type='AT'),  
    },  
    cookierun.models.LevelGuide: {  
        u'LevelContents': lambda x: x,  
    },  
    cookierun.models.GashaponStep1: {  
        u'Gashapone_Step1_Rate': lambda x: x,  
    },  
    cookierun.models.GashaponStep2: {  
        u'Gashapone_Step2_Rate': lambda x: x,  
    },  
    cookierun.models.WeeklyRankingReward: {  
        u'WeeklyRankReward': lambda x: x,  
    },  
}
```

# Diff를 구하는 알고리즘

- 시트의 모든 내용을 반영하도록 구현
- MySQL REPLACE 구문으로 넣는 것처럼 구현
- dict를 활용하여 존재 여부를 간단히 확인
- Git의 diff 명령처럼 추가될 row와 삭제될 row를 구함

```
def compare_objs(db_objs, sheet_objs):  
    diff = []  
    db_seq_map = dict((x.pk, x) for x in db_objs)  
    sheet_seq_map = dict((x.pk, x) for x in sheet_objs)  
  
    # Items to delete  
    for db_seq, db_obj in db_seq_map.items():  
        if db_seq not in sheet_seq_map:  
            diff.append(('DELETE', db_obj))  
  
    # Items to update  
    for db_seq, db_obj in db_seq_map.items():  
        if db_seq not in sheet_seq_map:  
            continue  
        sheet_obj = sheet_seq_map[db_seq]  
        if db_obj != sheet_obj:  
            diff.append(('DELETE', db_obj))  
            diff.append(('INSERT', sheet_obj))  
  
    # Items to insert  
    for sheet_seq, sheet_obj in sheet_seq_map.items():  
        if sheet_seq in db_seq_map:  
            continue  
        diff.append(('INSERT', sheet_obj))  
  
    return DatasheetDiff(diff)
```

# 데이터시트 반영

- 원하는 테이블을 선택하고 Import 버튼만 누르면 끝!

**Datasheet Manager - Import**

Import Export

Sheet Name	Uploaded Date	Delete
Datasheet Sample.xlsx	2014-08-25 21:03:21	Delete

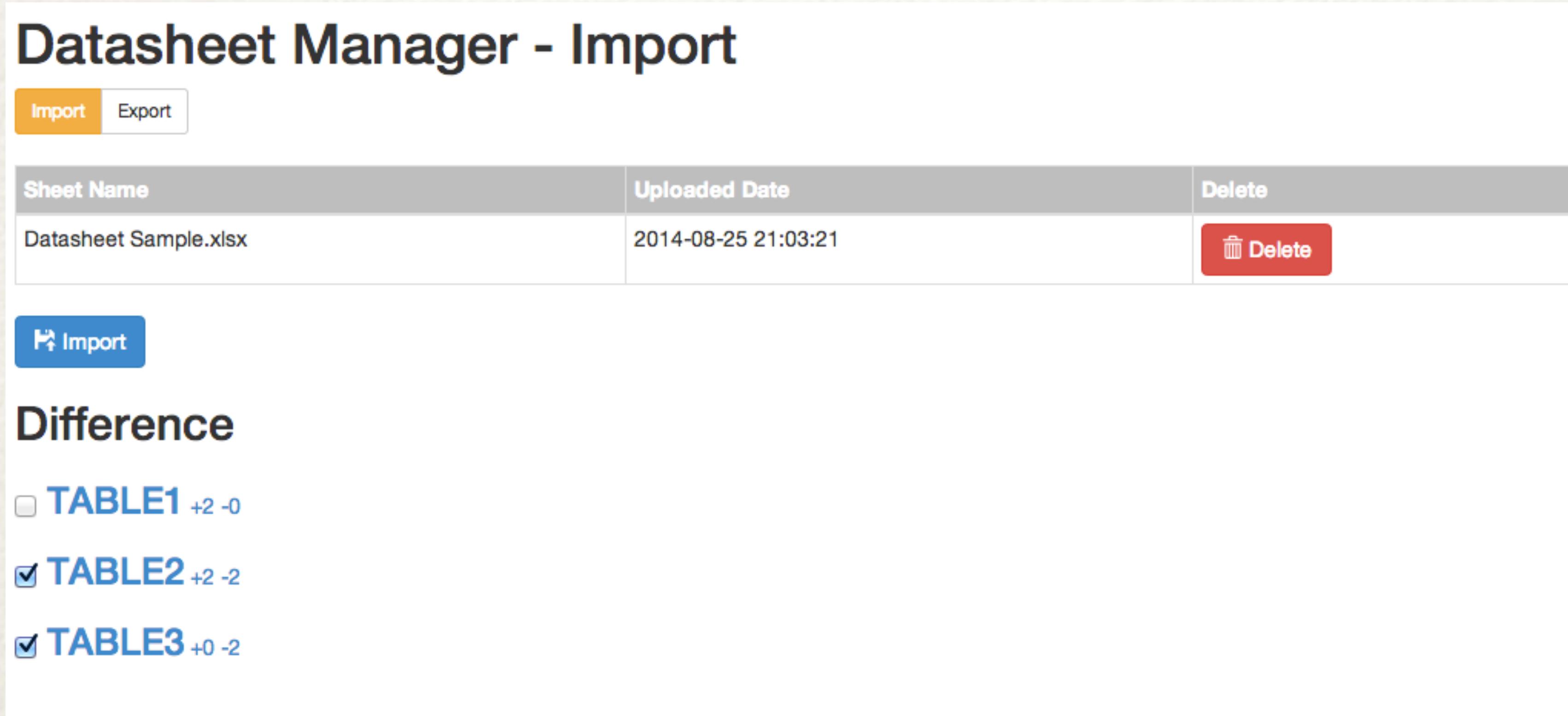
Import

**Difference**

TABLE1 +2 -0

TABLE2 +2 -2

TABLE3 +0 -2



# Export

- MySQL의 데이터를 엑셀 파일로 추출하여 볼 수 있는 기능
- 원하는 테이블들을 선택하고 Export 버튼을 누르면 엑셀 파일이 생성됨
- 여러개의 시트를 하나의 파일로 묶어서 관리 가능

Datasheet Manager - Export			
	Name	MAINLOCAL Rows	SUBLOCAL Rows
<input type="checkbox"/>	Table1	5	5
<input checked="" type="checkbox"/>	Table2	5	5
<input type="checkbox"/>	Table3	7	7
<input checked="" type="checkbox"/>	Table4	100	100
<input type="checkbox"/>	Table5	74	77
<input checked="" type="checkbox"/>	Table6	8	0
<input type="checkbox"/>	Table7	17	17
<input type="checkbox"/>	Table8	2	2

MAINLOCAL  SUBLOCAL

# DB 스키마가 변경되었어요! 어떡하죠?

- DB 스키마 변경에도 유연한 대처가 가능
- column이 추가되었을 때
  - ✓ model에 해당 column을 정의만 해주면 끝!
- table이 추가되었을 때
  - ✓ 새로운 model을 정의만 해주면 끝!
- inspectdb를 사용하면 DB로부터 Django model을 쉽게 만들 수 있음
  - ✓ \$ python manage.py inspectdb

# 성과

- 데이터시트를 한땀 한땀 올리는 업무 프로세스, 더 이상은..

- ✓ 몇번의 클릭만으로 가능해진 데이터시트 반영
- ✓ 한눈에 들어오는 diff

- 자유의 몸이 된 서버개발자

- ✓ 데이터 관리자가 직접 DB에 데이터를 반영할 수 있게 됨
- ✓ 불필요한 커뮤니케이션의 감소로 업무 효율 증가

- 내친김에 DB와 DB도 비교해서 올릴 수 있게 하자

- ✓ 테스트 환경에서 확인이 끝난 데이터를 production 환경에 바로 올릴 수 있도록
- ✓ Diff 구하는 알고리즘을 똑같이 적용, DB와 DB 비교에 활용



# Python이 아니었다면?

- 1년차 개발자도 Production 환경에서 개발할 수 있다!
- 간단한 일부터 까다로운 일까지
  - ✓ 급한 불을 끌 수 있는 일회성 스크립트
  - ✓ 전체 사용자를 주무르는 배치 서비스
  - ✓ 업무 프로세스를 180도 바꾸는 운영툴
  - ✓ 이 모든 일을 Python으로!
- 빠르게 많은 일들을 가능하게 해준 Python



# WE ARE HIRING!

[career@devsistors.com](mailto:career@devsistors.com)

- 최고의 인재들



# WE ARE HIRING!

[career@devsistors.com](mailto:career@devsistors.com)

- 흔한 게임 회사의 세끼 식사



# 감사합니다

