

전자정부표준프레임워크 기반의 Open PaaS 개발

개발 가이드

python애플리케이션

|  |  |  |
| --- | --- | --- |
| 작성자 | 김도준(인) | 2015.12.15 |
| 검토자 | (인) |  |
| 승인자 | (인) |  |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

개 정 이 력

| 버전 | 작성일 | 변경내용[[1]](#footnote-2) | 작성자 | 승인자 |
| --- | --- | --- | --- | --- |
| 1.0 | 2015.12.15 | 최초작성 | 김도준 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

목차

[1. 개요 5](#_Toc441153027)

[1.1. 문서 개요 5](#_Toc441153028)

[1.1.1. 목적 5](#_Toc441153029)

[1.1.2. 범위 5](#_Toc441153030)

[1.1.3. 참고 자료 5](#_Toc441153031)

[2. python 애플리케이션개발가이드 6](#_Toc441153032)

[2.1. 개요 6](#_Toc441153033)

[2.2. 개발환경 구성 6](#_Toc441153034)

[2.2.1. python 설치 6](#_Toc441153035)

[2.2.2. python 가상환경 구성 8](#_Toc441153036)

[2.2.3. Django 설치 9](#_Toc441153037)

[2.3. 개발 10](#_Toc441153038)

[2.3.1. django 애플리케이션 생성 10](#_Toc441153039)

[2.3.2. 애플리케이션 환경설정. 12](#_Toc441153040)

[2.3.3. VCAP\_SERVICES 환경설정 정보 14](#_Toc441153041)

[2.3.4. Mysql 연동 15](#_Toc441153042)

[2.3.5. Cubrid 연동 16](#_Toc441153043)

[2.3.6. MongoDB 연동 17](#_Toc441153044)

[2.3.7. Redis 연동 17](#_Toc441153045)

[2.3.8. RabbitMQ연동 18](#_Toc441153046)

[2.3.9. GlusterFS 연동 19](#_Toc441153047)

[2.4. 배포 20](#_Toc441153048)

[2.4.1. 개방형 클라우드 플랫폼 로그인 20](#_Toc441153049)

[2.4.2. 서비스 생성 21](#_Toc441153050)

[2.4.3. 애플리케이션 배포 21](#_Toc441153051)

[2.4.4. 서비스 바인드 23](#_Toc441153052)

[2.4.5. 애플리케이션 실행 23](#_Toc441153053)

[2.5. 테스트 23](#_Toc441153054)

[3. 부록 24](#_Toc441153055)

[3.1. 이클립스 개발환경 설정 24](#_Toc441153056)

[3.2. 이클립스 python 설정 25](#_Toc441153057)

[3.2.1. PyDev 설치 25](#_Toc441153058)

[3.2.2. python 설정 28](#_Toc441153059)

[3.2.3. django 프로젝트 생성 31](#_Toc441153060)

[3.2.4. django 애플리케이션 생성 34](#_Toc441153061)

1. 개요
   1. 문서 개요
      1. 목적

본 문서(python 애플리케이션 개발 가이드)는 Open PaaS프로젝트의 서비스팩(Mysql, Cubrid, MongoDB, RabbitMQ, Radis, GlusterFS)을 python 애플리케이션과 연동하여서비스를 사용하고 애플리케이션을 배포하는 방법에 대해 제시하는 문서이다.

* + 1. 범위

본 문서의 범위는 Open PaaS 프로젝트의 python 애플리케이션 개발과 서비스팩 연동에 대한 내용으로 한정한다.

* + 1. 참고 자료

<http://docs.run.pivotal.io/buildpacks/python/index.html>

<http://www.cubrid.com/manual/93/api/python.html>

<https://docs.djangoproject.com/en/1.9/intro/tutorial01>

<http://pythontips.com/>

1. python 애플리케이션개발가이드
   1. 개요

Open PaaS에 등록된 다양한 서비스팩을 python 언어로 작성된 애플리케이션과 바인딩하고 해당 애플리케이션에 바인딩 된 환경정보(VCAP\_SERVICES)에서 각 서비스별 접속정보를 획득하여 애플리케이션에 적용하여 이용 할 수 있도록 Windows 환경에서 python 애플리케이션을 작성할 수 있도록 한다.

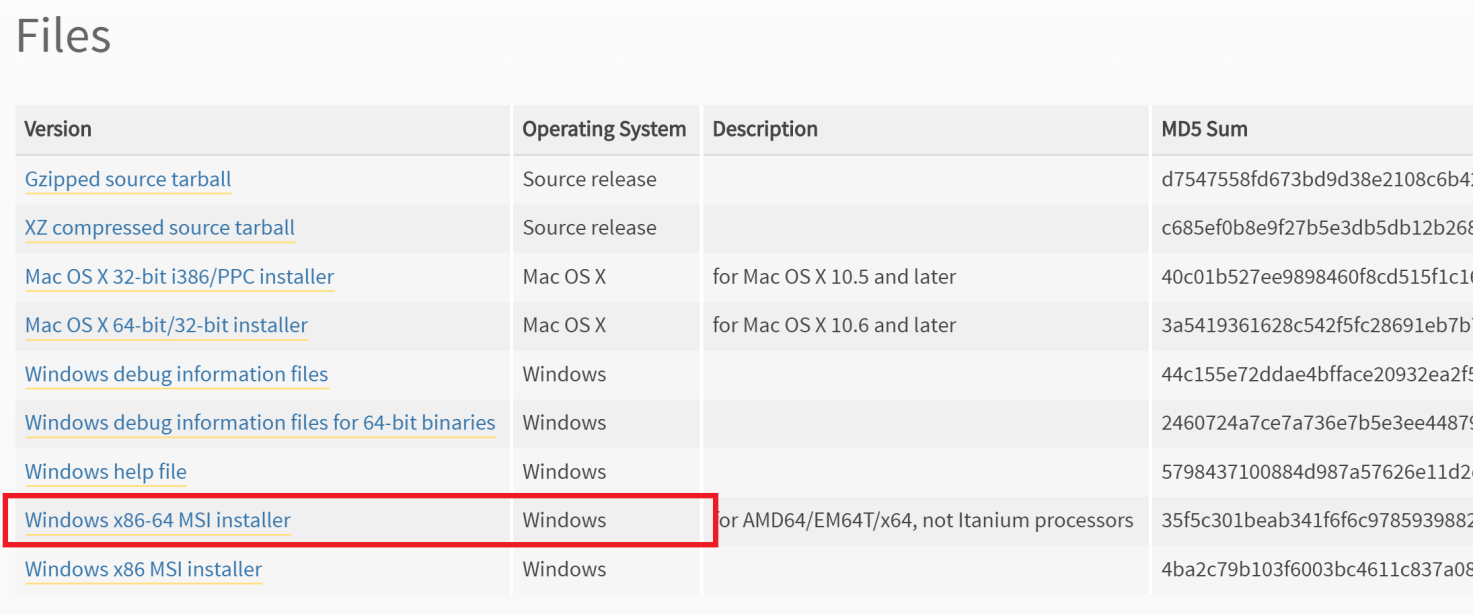
* 1. 개발환경 구성

python 샘플 애플리케이션 개발이 이루어진 환경은 다음과 같다..

* OS : Windows 8 64bit
* python : 2.7.10
* Framwork : Django 1.8.6
  + 1. python 설치

1. **python 2.7.10 다운로드**

|  |
| --- |
| https://www.python.org/downloads/release/python-2710/ |



* 다운로드

Windows x86-64 MSI installer

* + 사용자 각각의 환경에 따라 설치 파일은 달라질 수 있다.

1. **python 설치**

* 다운로드 받은 python-2.7.10.msi 더블 클릭하여 설치를 실행한다.
  + python 설치 시, 환경변수를 자동으로 추가 할 수 있도록 옵션을 선택할 수 있다.
* 환경변수 설정

python 설치 시, 환경변수 추가 옵션을 선택하지 않았다면 시스템 변수 Path에 다음과 같이 직접 추가한다.

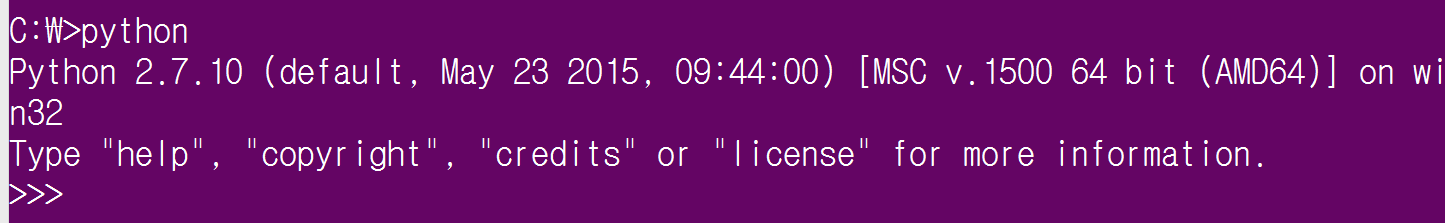
|  |
| --- |
| ;C:\Python27;C:\Python27\Scripts |

1. **python 설치 확인**

* 명령 프롬프트에서 python 실행

명령 프롬프트에서 'python' 을 입력하여 python의 실행을 확인한다.

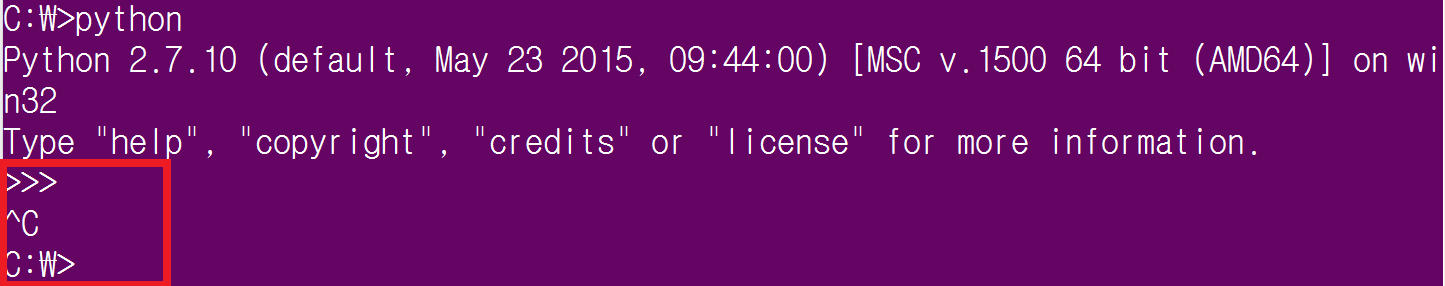
|  |
| --- |
| Python |



* python 실행 종료

python이 정상적으로 실행되었음을 확인하였으므로 'ctrl'+'c'를 입력하여 python을 종료한다.

>>> 'ctrl' + c

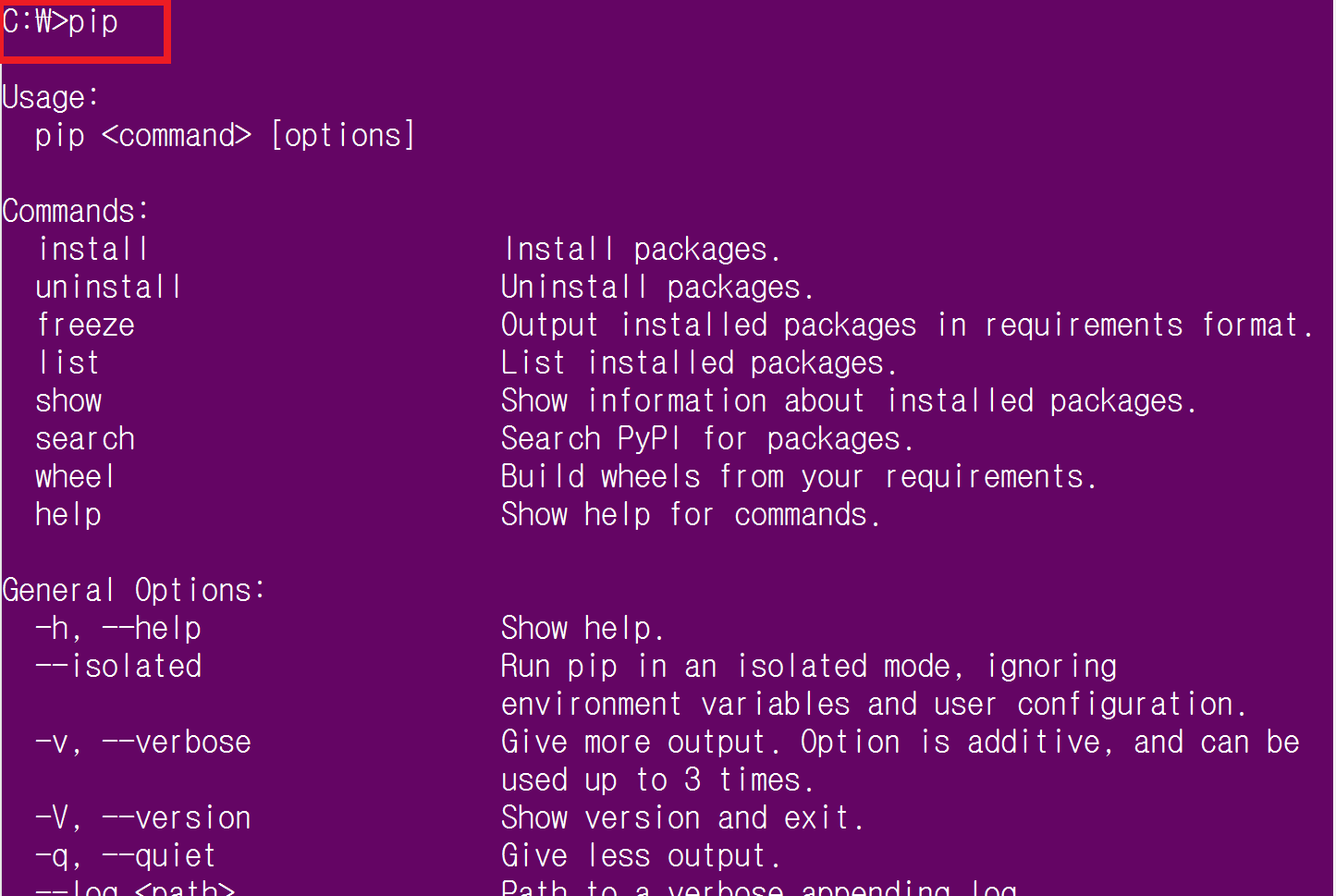


1. **pip 설치 확인**

* 명령 프롬프트에서 pip 입력

pip가 정상적으로 설치 되었을 경우, 명령어를 입력하면 pip 명령어에 대한 설명을 확인 할 수 있다.

|  |
| --- |
| pip |



* pip는 python 관련 패키지 설치를 지원하는 도구이다. 일반적으로 python을 설치하면 함께 설치되지만 경우에 따라 pip 설치가 되지 않을 수 있다. 이 때는 python 설치에 기본으로 제공되는 또 다른 설치도구인 easy\_install을 사용하여 pip를 설치한다.
* easy\_install 이용한 pip 설치

명령 프롬프트에서 python 설치 경로의 Scripts 디렉토리로 이동한다.

|  |
| --- |
| cd C:\Python27\Scripts |

pip 설치 명령어를 입력한다.

|  |
| --- |
| easy\_install pip |

* + 1. python 가상환경 구성

독립된 python 개발환경을 구성하기 위하여 python 가상환경 생성도구인 virtualenv를 설치하여 이용한다. 가상환경에서 패키지 설치를 진행한다. 가상환경 구성은 사용자의 필요에 따라 생략될 수 있다.

1. **가상환경 도구 설치**

virtualenv 설치

|  |
| --- |
| pip install virtualenv |

* 가상환경을 생성할 디렉토리로 이동하여 가상환경 생성

|  |
| --- |
| cd c:\  virtualenv my\_virtual\_env |

* 다양한 버전의 python이 설치되어 있다면, 가상환경 생성시에 python의 경로를 '-p' 옵션으로 지정하여, 가상환경 구성에 사용할 python을 선택할 수 있다. 다음은 예시이다.

|  |
| --- |
| virtualenv -p C:\Python34\python.exe my\_virtual\_env\_34 |
| virtualenv -p C:\Python27\python.exe my\_virtual\_env\_27 |

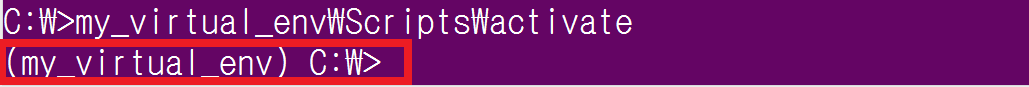
1. **가상환경 실행**

* 가상환경 실행

명령 프롬프트에서 다음의 명령어를 이용하여 가상환경을 실행한다.

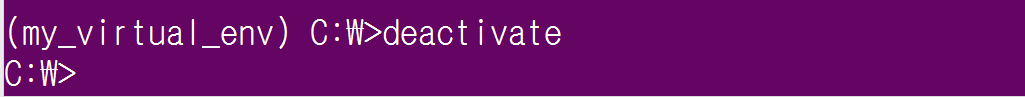
|  |
| --- |
| my\_virtual\_env\Scripts\activate |

가상환경이 정상적으로 실행되면 명령 프롬프트의 다음 화면과 같이 명령어 입력 줄에 가상환경 명칭이 붙는다.



* 가상환경을 실행을 종료시키기 위해서는 deactivate 명령어를 이용한다. 가상환경이 종료되면 명령어 입력 줄의 가상환경 명칭이 제거된다

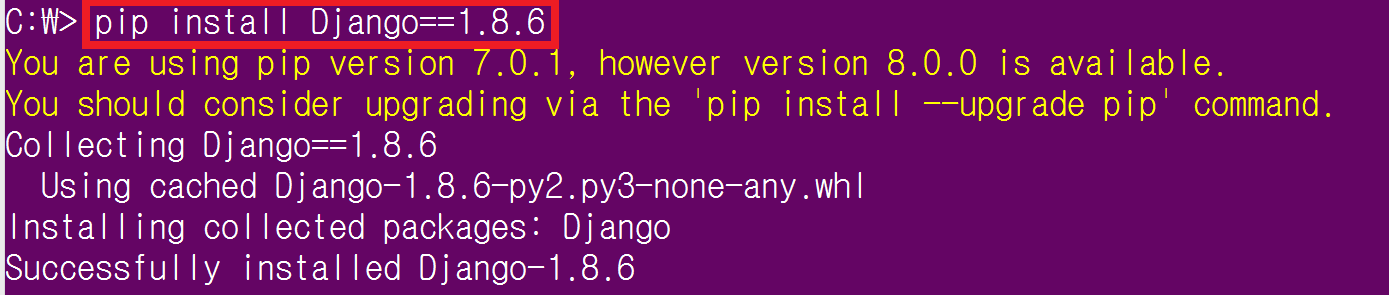
|  |
| --- |
| deactivate |



* + 1. Django 설치

샘플 어플리케이션은 Django framework를 적용하여 개발되었기 때문에, 어플리케이션 생성을 위해 Django를 설치 한다. 설치하는 django의 버전은 1.8.6 버전이다. pip를 사용하여 django를 설치한다.

|  |
| --- |
| pip install Django==1.8.6 |



* 가상환경에 장고를 설치하고자 하는 사용자는 가상환경을 실행시킨 상태에서 명령어를 입력한다.
  1. 개발

샘플 애플리케이션의 데이터 관리는 MySQL, CubridDB, MongoDB 중 하나를 이용하기 때문에 API 요청시 요청 본문의 DBType 값을 가지고 결정한다.

* + 1. django 애플리케이션 생성

1. **django 프로젝트 생성**

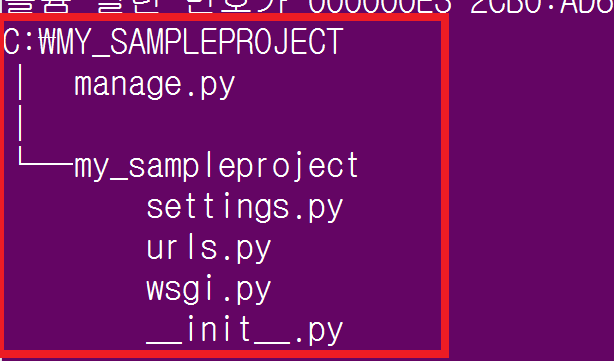
django 프로젝트를 생성할 디렉토리로 이동하여 하단의 명령어를 입력하여 샘플 애플리케이션 프로젝트를 생성한다.

* + 가상환경을 구성하여 가상환경에 Django를 설치한 사용자는 모든 명령어를 가상환경을 실행한 상태에서 입력한다. 가상환경 구성 및 실행은 본 문서의 [2.2.2. python 가상환경 구성]을 참고한다.

|  |
| --- |
| django-admin startproject my\_sampleproject |

* + django 프로젝트 생성시 프로젝트명에 '-'(hyphen)을 포함할 수 없기 때문에 '\_'(underscore)를 사용했다.

다음과 같은 구조로 파일이 생성된다.



|  |  |
| --- | --- |
| manage.py | 사이트 관리를 지원하는 파일. 서버 기동, 앱 생성 등의 기능을 지원한다. |
| setting.py | 사이트에 대한 설정을 관리. 본 문서의 python 샘플 앱에서는 모든 서비스에 대한 연동 설정을 포함한다. |
| urls.py | URL패턴 목록을 정의하는 모듈. |
| wsgi.py | WSGI application 어플리케이션을 정의하는 모듈. |
| \_\_init\_\_.py | 해당 패키지가 임포트될 때 수행되는 스크립트를 정의. \_\_init\_\_.py 파일을 삭제할 경우, 해당 디렉토리에 있는 서브모듈을 찾을 수 없게 되어 임포트에 실패한다. |

1. **django 애플리케이션 생성**

django 프로젝트를 생성했다면, 생성한 프로젝트 디렉토리로 이동하여 애플리케이션을 생성한다. 애플리케이션 생성 명령어는 다음과 같다.

프로젝트 디렉토리로 이동

|  |
| --- |
| cd my\_sampleproject |

애플리케이션 생성

|  |
| --- |
| python manage.py startapp my\_sampleapp |

* + django 애플리케이션 생성시 애플리케이션명에 '-'(hyphen)을 포함할 수 없기 때문에 '\_'(underscore)를 사용했다.

프로젝트의 파일 구조를 보면 붉은색 박스 부분이 추가 된 것을 확인할 수 있다



|  |  |
| --- | --- |
| admin.py | 장고에서 제공하는 데이터베이스 관리자 대시보드를 사용하기 위해 필요한 모듈이다. 본 샘플앱에서는 장고의 데이터베이스 관리 기능을 사용하지 않기 때문에 삭제해도 무방하다. |
| model.py | 장고에서 객체관계매핑(ORM)을 위해 데이터를 정의하는 모듈. 본 샘플앱에서는 사용하지 않기 때문에 삭제해도 무방하다. |
| tests.py | 장고 test 모듈. 본 샘플앱에서는 사용하지 않기 때문에 삭제해도 무방하다. |
| views.py | MVC 패턴의 controller 역할을 수행하는 모듈. 장고에서는 view가 controller로 기능한다. |
| \_\_init\_\_.py | 해당 패키지가 임포트될 때 수행되는 스크립트를 정의. \_\_init\_\_.py 파일을 삭제할 경우, 해당 디렉토리에 있는 서브모듈을 찾을 수 없게 되어 임포트에 실패한다. |

* + 1. 애플리케이션 환경설정.

django 애플리케이션에서 환경설정은 settings 모듈에 정의하도록 되어 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성을 통해 자동 생성된 my\_sampleproject 디렉토리의 settings.py 파일을 의미한다. 샘플어플리케이션에서 사용하는 패키지를 django 애플리케이션에서 사용하기 위해서는 이 settings 모듈에 설정을 추가하거나 수정하여야 한다. 하단에 settings 모듈에서 추가 또는 수정하여야 하는 부분을 설명과 함께 기술한다.

※ 녹색음영으로 표시된 부분이 추가되는 코드, 붉은색 음영으로 표시된 부분이 삭제되는 코드이다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| **INSTALLED\_APPS = (**  **'django.contrib.admin',**  **'django.contrib.auth',**  **'django.contrib.contenttypes',**  **'django.contrib.sessions',**  **'django.contrib.messages',**  **'django.contrib.staticfiles',**  **'rest\_framework',**  **'my\_sampleapp',**  **)**   * django rest framework 패키지를 사용할 수 있도록 INSTALLED APPS에 추가한다. * [2.3.1.2. django 애플리케이션 생성]에서 생성한 애플리케이션을 INSTALLED APPS에 추가한다.   **TEMPLATES = [ 생략 ... ]** #다음과 같이 수정  **TEMPLATES = [**  **{**  **'BACKEND': 'django.template.backends.django.DjangoTemplates',**  **'DIRS': [os.path.join(BASE\_DIR, 'static/templates'), ],**  **},**  **{**  **'BACKEND': 'django.template.backends.jinja2.Jinja2',**  **'DIRS': [os.path.join(BASE\_DIR, 'static/jinja2'), ],**  **},**  **]**   * 기존의 TEMPLATES를 연두색 음영으로 표시된 부분으로 수정한다. * ..\my\_sampleproject\static\template 디렉토리에 위치한 템플릿은 django의 기본 템플릿 엔진으로 읽고, ..\my\_sampleproject\static\jinja2 디렉토리에 위치한 템플릿은 jinja2 엔진으로 읽기 위해서 위와 같이 수정한다. jinja2 엔진은 템플릿 main.html과 manage.html에 쓰인 일부 신택스를 django의 기본 템플릿 엔진이 정상적으로 읽어낼 수 없기 때문에 사용한다.   **TIME\_ZONE = 'UTC'** #다음과 같이 수정  **TIME\_ZONE = ' Asia/Seoul'**   * 애플리케이션의 시간 설정을 'Asia/Seoul'로 변경한다   **STATIC\_BASE\_DIR = os.path.dirname(os.path.abspath(\_\_file\_\_))**  **STATICFILES\_DIRS = (**  **os.path.join(STATIC\_BASE\_DIR, '../static/resources'),**  **)**  **STATIC\_ROOT = 'staticfiles'**   * STATIC\_BASE\_DIR는 애플리케이션의 경로를 의미한다. 본 문서의 안내대로 진행했다면 애플리케이션의 경로는 '..\my\_sampleproject\my\_sampleapp'가 된다. 개방형 클라우드 플랫폼에 애플리케이션을 배포할 때, STATIC\_BASE\_DIR의 상위 디렉토리의 static/resources 경로에 위치한 파일들이 STATIC\_ROOT로 정의된 'staticfiles' 디렉토리에 모이게 된다.   **STATIC\_URL = '/static/'** #다음과 같이 수정  **STATIC\_URL = '/resources/'**   * 샘플의 템플릿에서 사용하는 리소스에 접근할 수 있는 URL   ※ settings 모듈에 DATABASES 설정과 cache 설정 등, 각각의 서비스와 연동할 수 있는 VCAP\_SERVICES 정보를 획득하는 코드가 추가되어야 하지만 하단의 [2.3.4.] ~ [2.3.9]에서 자세히 소개하므로 이 장에서는 다루지 않는다. |

WhiteNoise를 사용할 수 있도록 wsgi 모듈을 수정한다.

..\my\_sampleproject\my\_sampleproject\wsgi.py

|  |
| --- |
| **application = get\_wsgi\_application()**  **from whitenoise.django import DjangoWhiteNoise**  **application = DjangoWhiteNoise(get\_wsgi\_application())** |

* + 1. VCAP\_SERVICES 환경설정 정보

개방형 플랫폼에 배포되는 애플리케이션이 바인딩 된 각각의 서비스의 접속 정보를 얻기 위해서는 각각의 애플리케이션에 등록되어있는 VCAP\_SERVICES 환경설정 정보를 읽어 들여 정보를 획득 해야 한다.

1. 개방형 플랫폼의 애플리케이션 환경정보

* 서비스를 바인딩하면 JSON 형태로 환경설정 정보가 애플리케이션 별로 등록된다.

|  |
| --- |
| {  "VCAP\_SERVICES": {  "CubridDB": [  {  "credentials": {  "host": "10.30.60.23::",  "hostname": "10.30.60.23",  "jdbcurl": "jdbc:cubrid:10.30.60.23::fccf1d7869ff72ce:d90e3b43e151bc27:a7a7 cdfec36147bc:",  "name": "fccf1d7869ff72ce",  "password": "a7a7cdfec36147bc",  "port": "",  "uri": "cubrid:10.30.60.23::fccf1d7869ff72ce:d90e3b43e151bc27:a7a7cdfec3614 7bc:",  "username": "d90e3b43e151bc27"  },  "label": "CubridDB",  "name": "sample-cubrid-instance",  "plan": "utf8",  "tags": [  "cubrid",  "document"  ]  }  ],  "Mongo-DB": [  {  "credentials": {  "hosts": [  "10.30.60.53:27017"  ],  "name": "e37e541c-75de-4f01-8196-63e2d902e768",  "password": "c5649c42-ca5e-42be-926a-c3231aa81dc1",  "uri": "mongodb://b5d67268-897  …..(이하 생략)….. |

* + 1. Mysql 연동

각 서비스에 대한 VCAP\_SERVICES 환경설정 정보는 settings 모듈에서 획득할 수 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성시 자동으로 생성된다. MySQL의 경우는 MySQL-python 드라이버가 django 연동을 지원하기 때문에 settings 모듈의 DATABASES 정보가 정의된 부분을 찾아 다음과 같이 수정함으로써 연동이 가능하다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| import json  if 'VCAP\_SERVICES' in os.environ:  vcap\_services = json.loads(os.environ['VCAP\_SERVICES'])  if '**Mysql-DB**' in vcap\_services:  mysql\_srv = vcap\_services['**Mysql-DB**'][0]  mysql\_cred = mysql\_srv['credentials']  DATABASES = {  'default': {  'ENGINE': 'django.db.backends.mysql',  'NAME': mysql\_cred['name'],  'USER': mysql\_cred['username'],  'PASSWORD': mysql\_cred['password'],  'HOST': mysql\_cred['hostname'],  'PORT': mysql\_cred['port'],  },  } |

* DATABASES에 'default' 데이터베이스로 MySQL을 정의하였다. django에서는 데이터베이스 명칭에서 'default'는 반드시 존재해야 하기 때문에 'mysql'이 아닌 'default'를 사용하였다. 다른 데이터베이스를 추가하고자 한다면 추가되는 데이터베이스에 대해서는 임의의 명칭을 사용할 수 있다.
* 파란색 글씨로 표시된 부분은 개방형 클라우드 플랫폼에서 서비스되는 서비스의 이름이다. 실제 서비스 되는 이름과 정확히 일치해야 한다. 개방형 클라우드 플랫폼에서 'cf marketplace' 명령어를 입력하여 서비스명을 확인할 수 있다.

mysql\_views.py 모듈에서 connections를 임포트하여 다음과 같은 형태로 cursor를 생성한다.

..\my\_sampleproject\my\_sampleapp\mysql\_views.py

|  |
| --- |
| from django.db import connections |

|  |
| --- |
| def make\_connection():  db\_type = 'default'  cursor = connections[db\_type].cursor()  return cursor |

* + 1. Cubrid 연동

각 서비스에 대한 VCAP\_SERVICES 환경설정 정보는 settings 모듈에서 획득할 수 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성시 자동으로 생성된다. CUBRID-Python 드라이버는 django 연동을 지원하지 않기 때문에 settings 모듈에서 VCAP\_SERVICES 환경설정 정보의 credentials 정보를 cubrid\_views 모듈에서 읽어와 connection을 생성한다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| import json  if 'VCAP\_SERVICES' in os.environ:  vcap\_services = json.loads(os.environ['VCAP\_SERVICES'])  if '**CubridDB**' in vcap\_services:  cubrid\_srv = vcap\_services['**CubridDB**'][0]  **CUBRID\_CRED** = cubrid\_srv['credentials'] |

* django의 settings.py 모듈에 정의된 값을 다른 모듈에서 불러오기 위해서는 settings.py 모듈에서 영문 대문자로 정의되어 있어야만 한다.
* 파란색 글씨로 표시된 부분은 개방형 클라우드 플랫폼에서 서비스되는 서비스의 이름이다. 실제 서비스 되는 이름과 정확히 일치해야 한다. 개방형 클라우드 플랫폼에서 'cf marketplace' 명령어를 입력하여 서비스명을 확인할 수 있다.

..\my\_sampleproject\my\_sampleapp\cubrid\_views.py

|  |
| --- |
| from django.conf import settings  import CUBRIDdb |

|  |
| --- |
| def make\_connection():  if settings.**CUBRID\_CRED**:  credentials = settings.**CUBRID\_CRED**  connection = CUBRIDdb.connect(  'CUBRID:'+credentials['hostname']+':33000:'+credentials['name']+':::',  credentials['username'],  credentials['password']  )  return connection |

* + 1. MongoDB 연동

각 서비스에 대한 VCAP\_SERVICES 환경설정 정보는 settings 모듈에서 획득할 수 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성시 자동으로 생성된다. pymongo 드라이버는 django 연동을 지원하지 않기 때문에 settings 모듈에서 획득한 VCAP\_SERVICES 환경설정 정보의 credentials 정보를 mongo\_views 모듈에서 읽어와 db를 생성한다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| import json  if 'VCAP\_SERVICES' in os.environ:  vcap\_services = json.loads(os.environ['VCAP\_SERVICES'])  if '**Mongo-DB**' in vcap\_services:  mongo\_srv = vcap\_services['**Mongo-DB**'][0]  **MONGO\_CRED** = mongo\_srv['credentials'] |

* django의 settings 모듈에 정의된 값을 다른 모듈에서 불러오기 위해서는 settings 모듈에서 영문 대문자로 정의되어 있어야만 한다.
* 파란색 글씨로 표시된 부분은 개방형 클라우드 플랫폼에서 서비스되는 서비스의 이름이다. 실제 서비스 되는 이름과 정확히 일치해야 한다. 개방형 클라우드 플랫폼에서 'cf marketplace' 명령어를 입력하여 서비스명을 확인할 수 있다.

..\my\_sampleproject\my\_sampleapp\mongo\_views.py

|  |
| --- |
| from django.conf import settings  from pymongo import MongoClient |

|  |
| --- |
| def make\_connection():  if settings.**MONGO\_CRED**:  credentials = settings.**MONGO\_CRED**  mongo\_hosts = credentials['hosts'][0].split(':')  connection = MongoClient(mongo\_hosts[0], int(mongo\_hosts[1]))  db = connection[credentials['name']]  db.authenticate(credentials['username'], credentials['password'])  return db |

* + 1. Redis 연동

각 서비스에 대한 VCAP\_SERVICES 환경설정 정보는 settings 모듈에서 획득할 수 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성시 자동으로 생성된다. redis의 경우는 django-redis-cache 드라이버가 django 연동을 지원하기 때문에 settings 모듈의 CACHES 정보가 정의된 부분을 찾아 다음과 같이 수정함으로써 연동이 가능하다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| import json  if 'VCAP\_SERVICES' in os.environ:  vcap\_services = json.loads(os.environ['VCAP\_SERVICES'])  if **'redis-sb'** in vcap\_services:  redis\_srv = vcap\_services[**'redis-sb'**][0]  redis\_cred = redis\_srv['credentials']    CACHES = {  "default": {  "BACKEND": "redis\_cache.RedisCache",  "LOCATION": str(redis\_cred['host'])+":"+str(redis\_cred['port']),  "OPTIONS": {  "CLIENT\_CLASS": "django\_redis.client.DefaultClient",  'PASSWORD': redis\_cred['password'],  },  },  } |

* 파란색 글씨로 표시된 부분은 개방형 클라우드 플랫폼에서 서비스되는 서비스의 이름이다. 실제 서비스 되는 이름과 정확히 일치해야 한다. 개방형 클라우드 플랫폼에서 'cf marketplace' 명령어를 입력하여 서비스명을 확인할 수 있다.

..\my\_sampleproject\my\_sampleapp\redis\_views.py

|  |
| --- |
| from django.core.cache import cache |

* cache를 임포트하면 다음과 같은 형태로 바로 사용이 가능하다.

|  |
| --- |
| cache.get('key값')  cache.set('key값','value값')  cache.delete('key값') |

* + 1. RabbitMQ연동

각 서비스에 대한 VCAP\_SERVICES 환경설정 정보는 settings 모듈에서 획득할 수 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성시 자동으로 생성된다. settings 모듈에서 획득한 VCAP\_SERVICES 환경설정 정보의 credentials 정보를 rabbitmq\_views 모듈에서 읽어와 connection을 생성한다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| import json  if 'VCAP\_SERVICES' in os.environ:  vcap\_services = json.loads(os.environ['VCAP\_SERVICES'])  if '**p-rabbitmq**' in vcap\_services:  rabbitmq\_srv = vcap\_services['**p-rabbitmq**'][0]  **RABBITMQ\_CRED** = rabbitmq\_srv['credentials'] |

* django의 settings 모듈에 정의된 값을 다른 모듈에서 불러오기 위해서는 settings 모듈에서 영문 대문자로 정의되어 있어야만 한다.
* 파란색 글씨로 표시된 부분은 개방형 클라우드 플랫폼에서 서비스되는 서비스의 이름이다. 실제 서비스 되는 이름과 정확히 일치해야 한다. 개방형 클라우드 플랫폼에서 'cf marketplace' 명령어를 입력하여 서비스명을 확인할 수 있다.

..\my\_sampleproject\my\_sampleapp\rabbitmq\_views.py

|  |
| --- |
| from django.conf import settings  import pika |

|  |
| --- |
| def make\_connection():  if settings.**RABBITMQ\_CRED**:  credentials = settings.**RABBITMQ\_CRED**  rabbitmq\_protocols = credentials['protocols']  rabbitmq\_uri = rabbitmq\_protocols['amqp+ssl']['uri']  parameters = pika.URLParameters(rabbitmq\_uri)  connection = pika.BlockingConnection(parameters)  return connection |

* + 1. GlusterFS 연동

각 서비스에 대한 VCAP\_SERVICES 환경설정 정보는 settings 모듈에서 획득할 수 있다. settings 모듈은 [2.3.1.1. django 프로젝트 생성]에서 프로젝트 생성시 자동으로 생성된다. settings 모듈에서 획득한 VCAP\_SERVICES 환경설정 정보의 credentials 정보를 gluster\_views 모듈에서 읽어와 connection을 생성한다.

..\my\_sampleproject\my\_sampleproject\settings.py

|  |
| --- |
| import json  if 'VCAP\_SERVICES' in os.environ:  vcap\_services = json.loads(os.environ['VCAP\_SERVICES'])    if '**glusterfs**' in vcap\_services:  gluster\_srv = vcap\_services['**glusterfs**'][0]  **GLUSTERFS\_CRED** = gluster\_srv['credentials'] |

* django의 settings 모듈에 정의된 값을 다른 모듈에서 불러오기 위해서는 해당 값이 settings 모듈에서 영문 대문자로 정의되어 있어야만 한다.
* 파란색 글씨로 표시된 부분은 개방형 클라우드 플랫폼에서 서비스되는 서비스의 이름이다. 실제 서비스 되는 이름과 정확히 일치해야 한다. 개방형 클라우드 플랫폼에서 'cf marketplace' 명령어를 입력하여 서비스명을 확인할 수 있다.

..\my\_sampleproject\my\_sampleapp\gluster\_views.py

|  |
| --- |
| from django.conf import settings  import swiftclient |

|  |
| --- |
| def make\_connection():  if settings.**GLUSTERFS\_CRED**:  credentials = settings.**GLUSTERFS\_CRED**  auth\_url = credentials['auth\_url']  username = credentials['username']  password = credentials['password']  tenantname = credentials['tenantname']  connection = swiftclient.client.Connection(  auth\_version='2',  authurl=auth\_url,  user=username,  key=password,  tenant\_name=tenantname  )  return connection |

* 1. 배포

개발 완료된 애플리케이션을 개방형 클라우드 플랫폼에 배포하는 방법을 기술한다. 배포과정은 개방형 클라우드 플랫폼 로그인, 서비스 생성, 애플리케이션 배포, 서비스 바인드, 애플리케이션 실행의 과정으로 이루어져 있다.

* + 1. 개방형 클라우드 플랫폼 로그인

애플리케이션 배포과정을 진행하기 위해 개방형 클라우드 플랫폼의 사용자 계정으로 로그인한다.

로그인을 하기 이전에 먼저 API Endpoint를 지정한다. 명령어는 아래와 같다.

|  |
| --- |
| **# cf api [Endpoint URL]**  $ cf api --skip-ssl-validation https://api.cf.open-paas.com |

Endpoint 지정이 완료되었다면, 로그인 명령어를 통해 로그인한다.

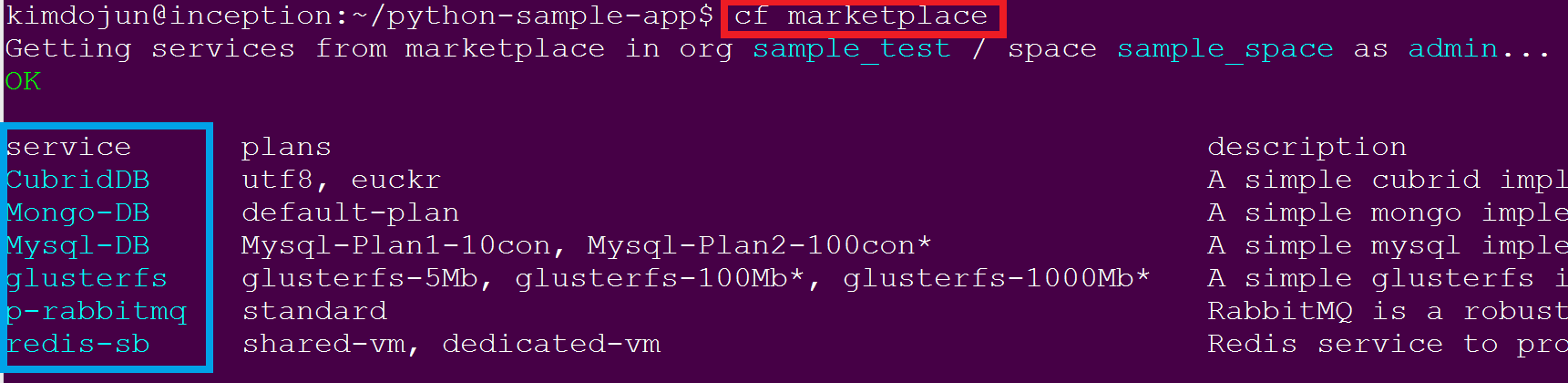
|  |
| --- |
| **# cf login –u [user name] –o [org name] –s [space name]**  $ cf login -u testUser -o sample\_test -s sample\_space |

* + 1. 서비스 생성

서비스 생성은 개방형 클라우드 플랫폼에서 제공하는 서비스에 대해서 사용자가 서비스 인스턴스를 생성하는 과정이다. 개방형 클라우드 플랫폼에서 제공하지 않는 서비스에 대해서는 서비스 생성이 불가능하며, 서비스 제공 여부는 플랫폼 관리자(운영자)가 결정한다.

먼저 아래의 명령어를 통해, 사용 가능한 서비스의 목록을 확인한다.

|  |
| --- |
| $ cf marketplace |



상단의 명령어를 통해 확인한 서비스 목록에서 사용하고자 하는 서비스를 생성한다. 샘플 애플리케이션에서는 MySQL, Cubrid, MongoDB, Resdis, RabbitMQ, GlusterFS 서비스를 사용하므로 6개의 서비스를 생성한다. 서비스 생성 명령어는 다음과 같다. 상단의 그림에서 파란색 박스로 표시된 부분이 서비스 명이다.

|  |
| --- |
| **# cf create-service SERVICE PLAN SERVICE\_INSTANCE [-c PARAMETERS\_AS\_JSON] [-t TAGS]**  예시)  $ cf create-service Mysql-DB Mysql-Plan1-10con python-mysql  $ cf create-service CubridDB utf8 python-cubrid  $ cf create-service Mongo-DB default-plan python-mongodb  $ cf create-service redis-sb shared-vm python-redis  $ cf create-service glusterfs glusterfs-5Mb python-glusterfs  $ cf create-service p-rabbitmq standard python-rabbitmq |

* cf create-service 명령어는 서비스명, 플랜, 서비스 인스턴스명을 순서대로 입력하게 되어 있다. 서비스명과 플랜은 cf marketplace 명령어를 통해 확인하고, 서비스 인스턴스명은 임의의 명칭을 사용한다.
  + 1. 애플리케이션 배포

1. **requirements.txt 생성**

requirements.txt 파일에 python 샘플 애플리케이션 구동에 필요한 패키지들이 정의된다. 개방형 클라우드 플랫폼에서는 애플리케이션이 배포될 때, requirements.txt 파일에 정의된 패키지들을 설치한다. 따라서 requirements.txt 파일이 존재하지 않거나 내용이 잘못 되어 있을 경우, 애플리케이션 실행에 문제가 발생한다.

..\my\_sampleproject\requirements.txt

|  |
| --- |
| Django==1.8.6  djangorestframework==3.3.1  gunicorn==19.1.1  jinja2==2.8  WhiteNoise==2.0.4  MySQL-python==1.2.3  CUBRID-Python==9.3.0.0001  pymongo==2.8  pika==0.10.0  django-redis-cache==1.6.4  python-keystoneclient==2.0.0  python-swiftclient==2.6.0 |

각각의 패키지에 대한 설명은 하단 표를 참고한다.

|  |  |  |
| --- | --- | --- |
| **패키지명** | **버전** | **설명** |
| Django | 1.8.6 | python 웹 프레임워크 |
| djangorestframework | 3.3.1 | django 기반의 웹 api 구현 프레임워크 |
| Gunicorn | 19.1.1 | 유닉스 환경에서 애플리케이션 실행을 위한 WSGI server. 개방형 클라우드 플랫폼에 배포하기 위해서 필요. |
| jinja2 | 2.8 | django template 엔진. 샘플 애플리케이션 template의 일부 syntax를 django의 기본 template 엔진이 정상적으로 읽을 수 없기 때문에 사용. |
| WhiteNoise | 2.0.4 | Python 웹 애플리케이션에서 정적파일(static files)을 제공하기 위해 사용 |
| MySQL-python | 1.2.3 | Django 애플리케이션과 mysql을 연동하는 드라이버 |
| CUBRID-Python | 9.3.0.0001 | python과 Cubrid를 연동하는 드라이버 |
| pymongo | 2.8 | python과 Mongodb를 연동하는 드라이버 |
| pika | 0.10.0 | python과 RabbitMQ를 연동하는 드라이버 |
| django-redis-cache | 1.6.4 | django 애플리케이션과 Redis를 연동하는 드라이버 |
| python-keystoneclient | 2.0.0. | Openstack의 인증 API를 사용하기 위한 클라이언트. GlusterFS 연동에 필요 |
| python-swiftclient | 2.6.0 | Openstack의 swift 클라이언트. GlusterFS 연동에 필요 |

1. **manifest.yml 생성**

애플리케이션 배포 시, manifest.yml 파일을 참조하여 배포가 이루어진다.

..\my\_sampleproject\manifest.yml

|  |
| --- |
| ---  applications:  - name: python-sample-app **# 애플리케이션 이름**  buildpack: python\_buildpack **# 빌드팩 이름**  memory: 512M **# 애플리케이션 메모리 사이즈**  instances: 1 **# 애플리케이션 인스턴스 개수** |

* + manifest.yml 파일은 더 많은 설정을 포함할 수 있지만, 필요한 부분만 기술하였다.

1. **애플리케이션 배포**

애플리케이션 배포 명령어는 다음과 같다. 별도의 옵션을 넣지 않으면 manifest.yml에 정의된 설정을 사용한다. 서비스가 애플리케이션에 바인드되지 않은 상태로 애플리케이션이 배포되기 때문에 --no-start 옵션을 넣어 애플리케이션이 실행되지 않도록 한다.

|  |
| --- |
| $ cf push --no-start |

* + 1. 서비스 바인드

[2.4.2. 서비스 생성]에서 생성한 서비스와 [2.4.3. 애플리케이션 배포]에서 배포한 애플리케이션을 연결하는 것을 서비스 바인드(bind)라고 한다. 서비스 바인드를 통해서 애플리케이션은 서비스에 접근할 수 있는 VCAP\_SERVICES 환경설정 정보를 얻을 수 있게 되고 이는 [2.3.3. VCAP\_SERVICES 환경설정 정보]에서 확인할 수 있다.

|  |
| --- |
| **# cf bind-service APP\_NAME SERVICE\_INSTANCE [-c PARAMETERS\_AS\_JSON]**  $ cf bind-service python-sample-app python-mysql  $ cf bind-service python-sample-app python-cubrid  $ cf bind-service python-sample-app python-mongodb  $ cf bind-service python-sample-app python-redis  $ cf bind-service python-sample-app python-glusterfs  $ cf bind-service python-sample-app python-rabbitmq |

* + cf bind-service 명령어는 바인드할 애플리케이션명과 서비스 인스턴스명을 순서대로 입력하여 사용한다. 이때 '-c' 옵션을 이용해 애플리케이션의 VCAP\_SERVICES 환경설정정보에 필요한 값을 추가할 수 있다.
    1. 애플리케이션 실행

서비스 바인드가 완료되었다면 애플리케이션을 정상적으로 실행시킬 수 있다. 애플리케이션 실행 명령어는 다음과 같다.

|  |
| --- |
| $ cf start python-sample-app |

* + MySQL과 Cubrid 서비스는 사용자가 직접 DB에 접속하여 테이블을 먼저 생성해야 사용이 가능하다.
  1. 테스트

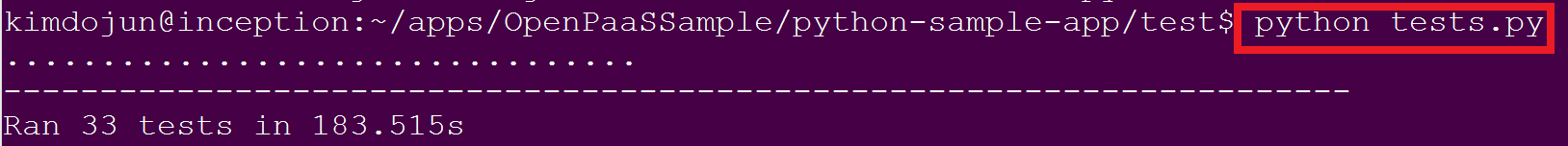
python 샘플 애플리케이션은 REST 서비스로 구현되어 있으며, REST 테스트를 위해 python에 기본적으로 내장된 모듈인 unittest 모듈을 사용한다. django의 테스트 기능이 있지만, 샘플 애플리케이션에서 사용하기에는 적절하지 않아 tests 모듈을 생성하여 REST 테스트를 진행하였다. 추가적으로 테스트 결과를 html파일로 저장하기 위해 nose-html-report를 설치하였다.

1. **테스트 실행**

테스트 모듈이 위치한 디렉토리에서 다음 명령어를 사용한다.

|  |
| --- |
| python tests.py |

테스트를 실행하면 다음과 같은 화면을 확인할 수 있다.



1. **테스트 결과를 html 파일로 저장할 경우**

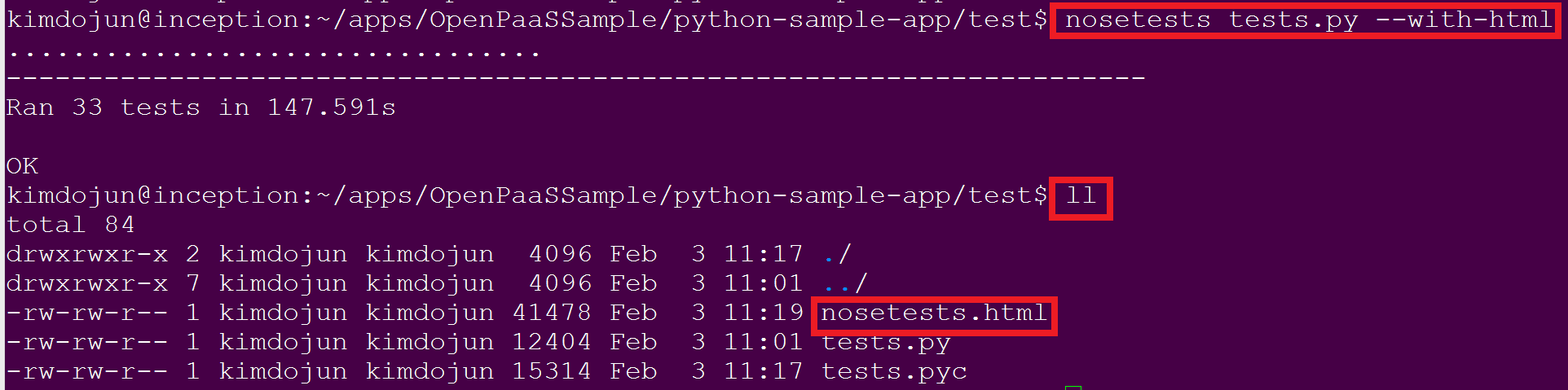
테스트 결과를 html 파일로 저장하기 위해서 nose-html-report를 설치한다. 설치는 pip를 이용한다.

|  |
| --- |
| pip install nose-html-reporting |

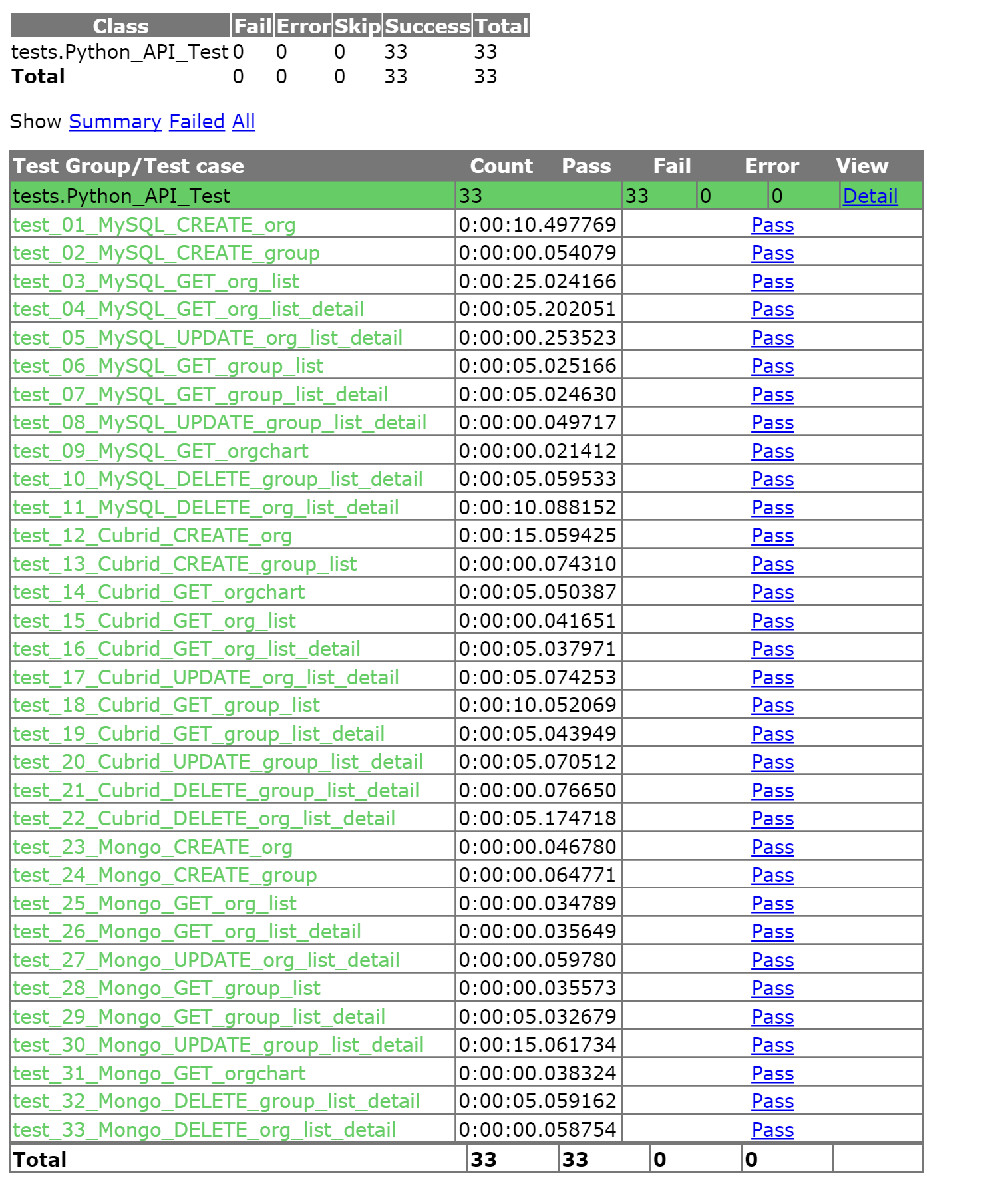
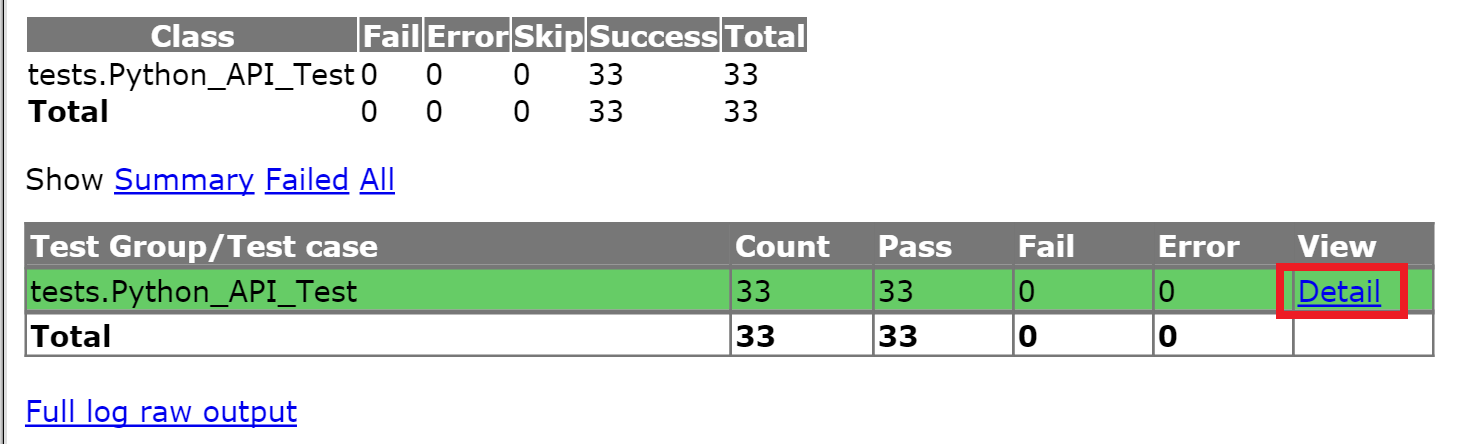
테스트 실행 명령어는 다음과 같다.

|  |
| --- |
| nosetests tests.py --with-html |

테스트가 완료되면 nosetests.html 파일이 생성되는 것을 확인할 수 있다.



생성된 nosetests.html 파일은 다음과 같은 화면을 출력한다. Detail 버튼을 눌러 세부 사항을 확인할 수 있다.



1. 부록
   1. 이클립스 개발환경 설정

본 장은 본 문서의 [2.3. 개발] 과정에서 이클립스(eclipse)를 개발 도구로 사용할 수 있도록 안내한다. 본 장은 사용자가 본 문서의 [2.2.3. Django 설치]까지 완료하였음을 전제로 기술된다.

이클립스 개발환경 설정은 아래와 같은 환경에서 이루어졌다.

* OS : Windows 8 64bit
* python : 2.7.10
* Framwork : Django 1.8.6
* Eclipse: version Mars.1 Release (4.5.1)

이클립스 다운로드는 하기 링크를 이용한다. 사용자 각각의 환경에 맞는 이클립스를 다운로드한다.

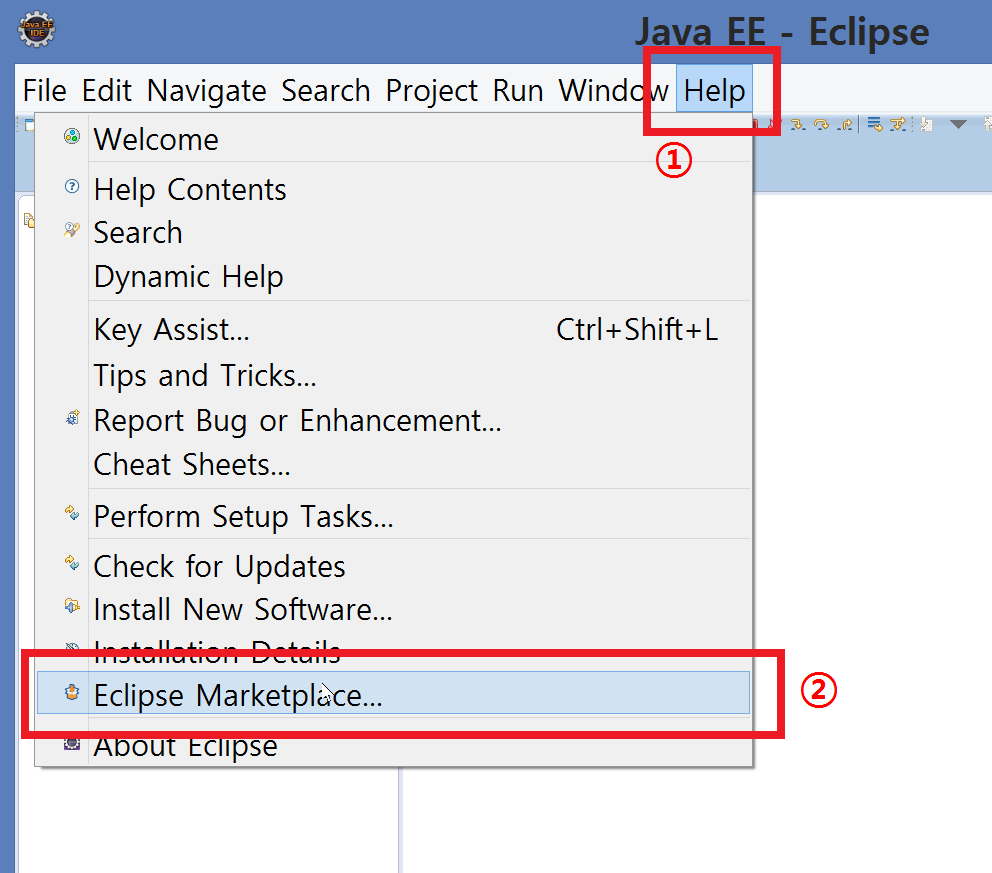
<http://www.eclipse.org/downloads/>

* 작성자의 경우, 다운로드 받은 파일의 파일명은 'eclipse-jee-mars-1-win32-x86\_64.zip'이며 다운로드 링크는 아래와 같다. <http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/mars/1/eclipse-jee-mars-1-win32-x86_64.zip>

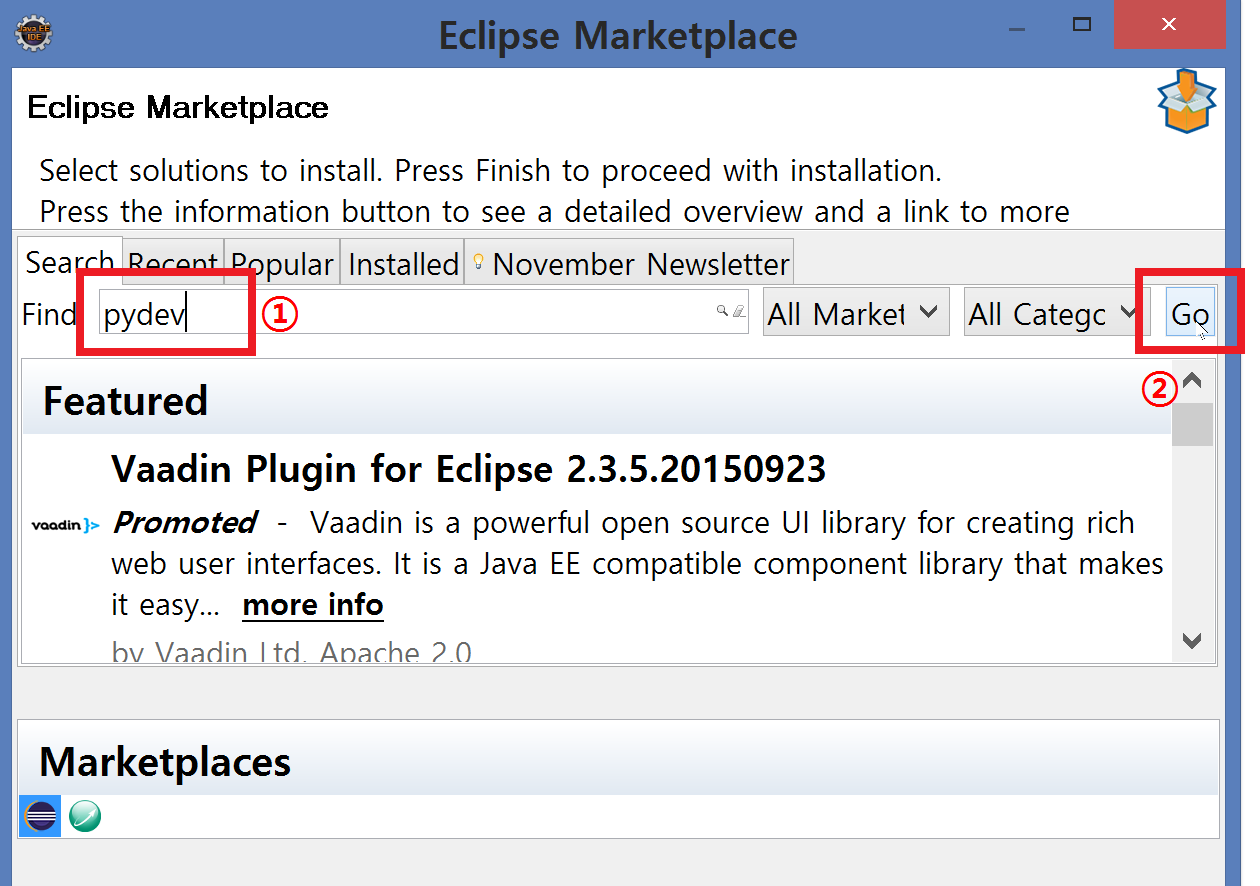
다운로드가 완료되면 압축을 풀고 이클립스를 실행한다.

* 1. 이클립스 python 설정
     1. PyDev 설치

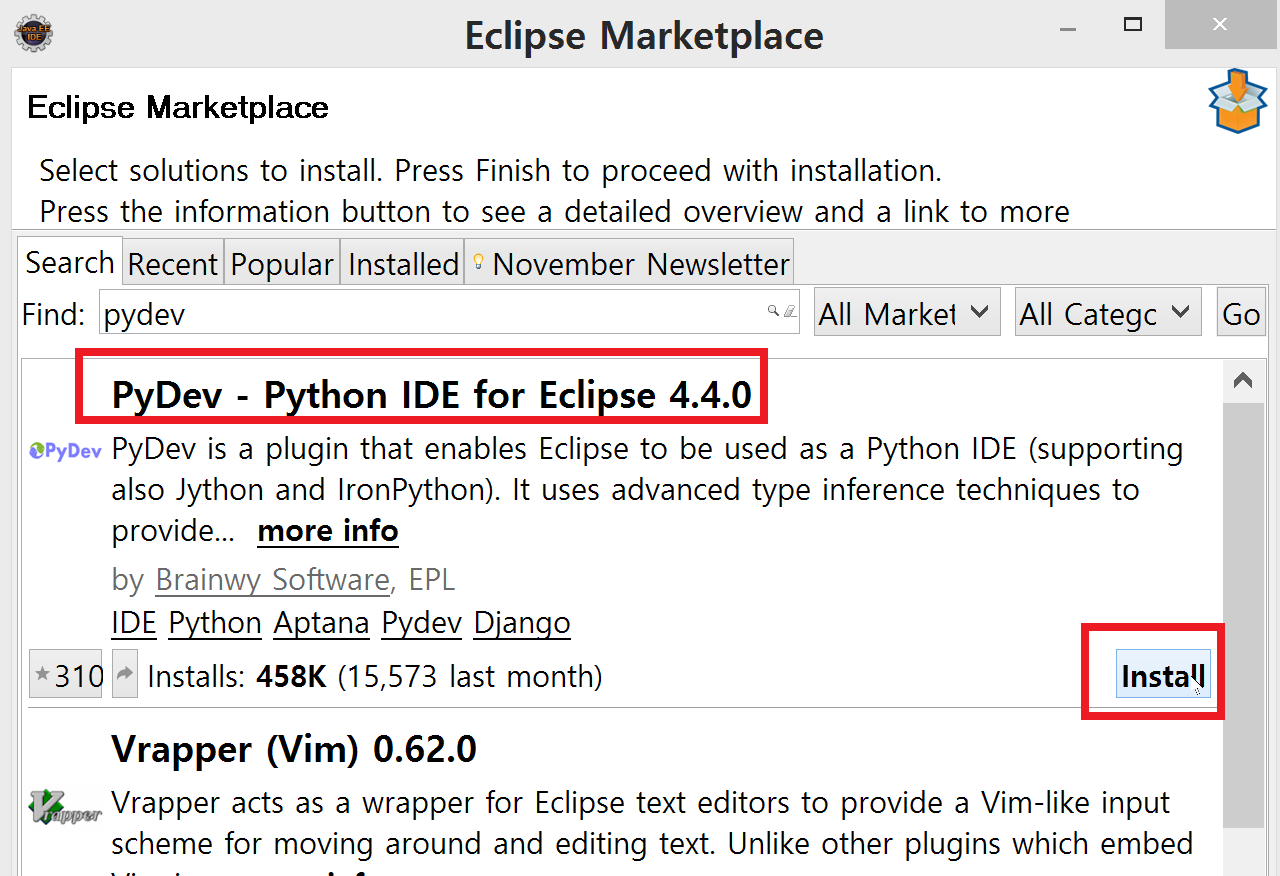
PyDev는 이클립스에서 python 개발을 가능하게 하는 이클립스 플러그인이다.



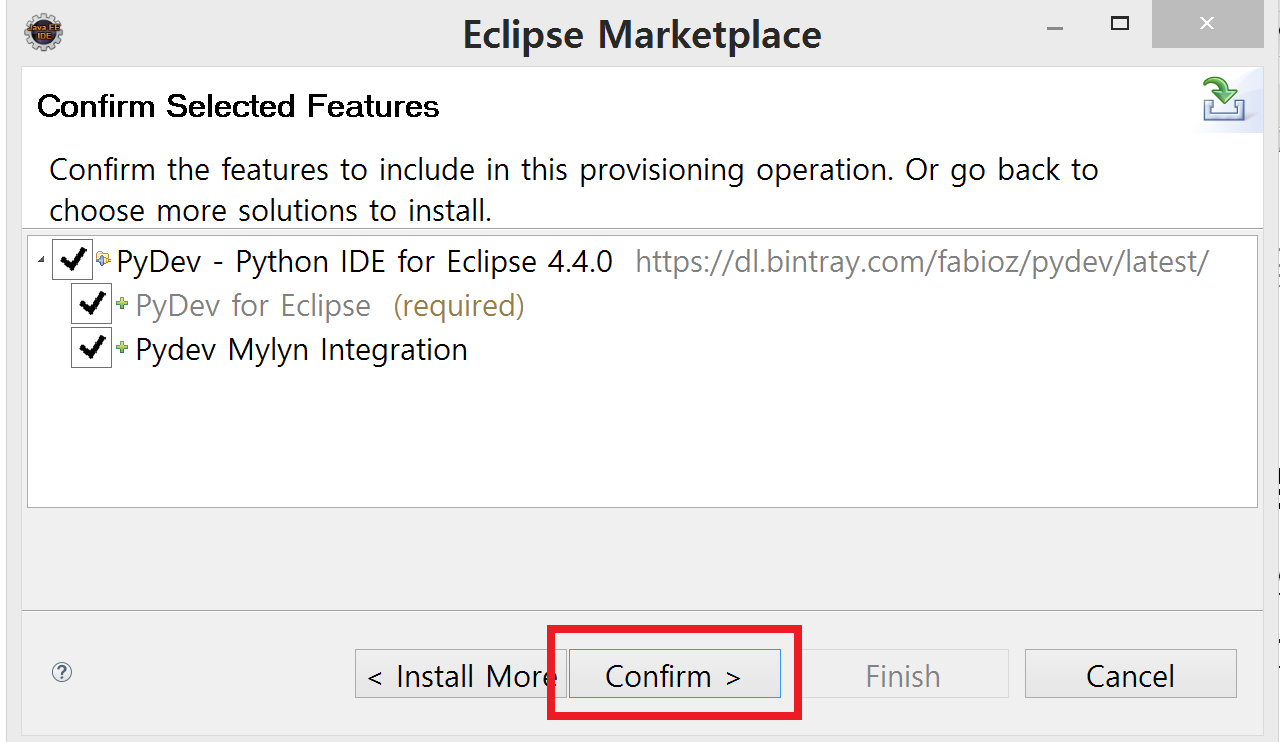
이클립스가 실행되면 상단 메뉴바의 help - Eclipse Marketplace를 클릭한다.



마켓 플레이스에서 'pydev'로 검색한다.

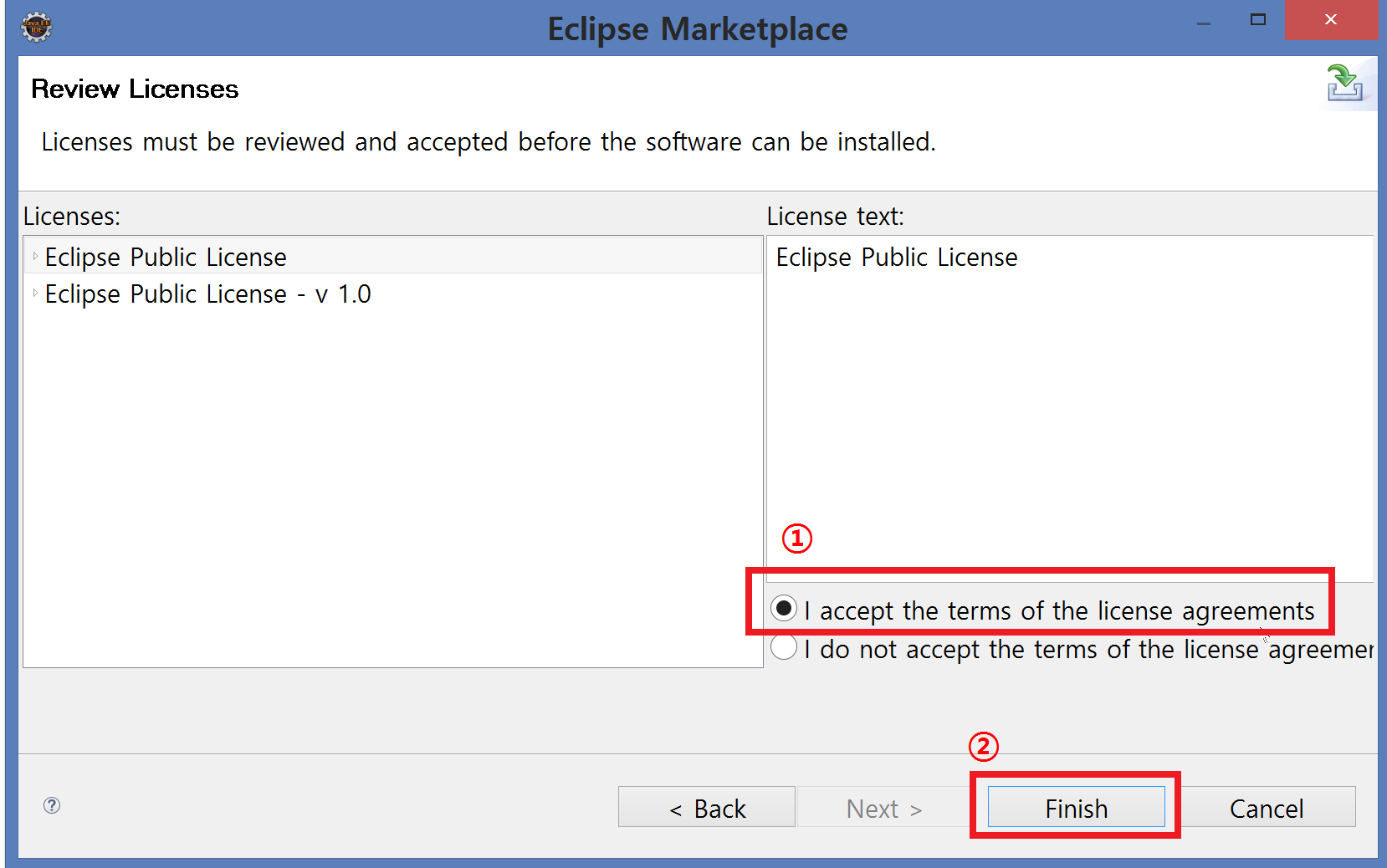


검색 결과에서 'PyDev'를 찾아 'install' 버튼을 클릭한다.

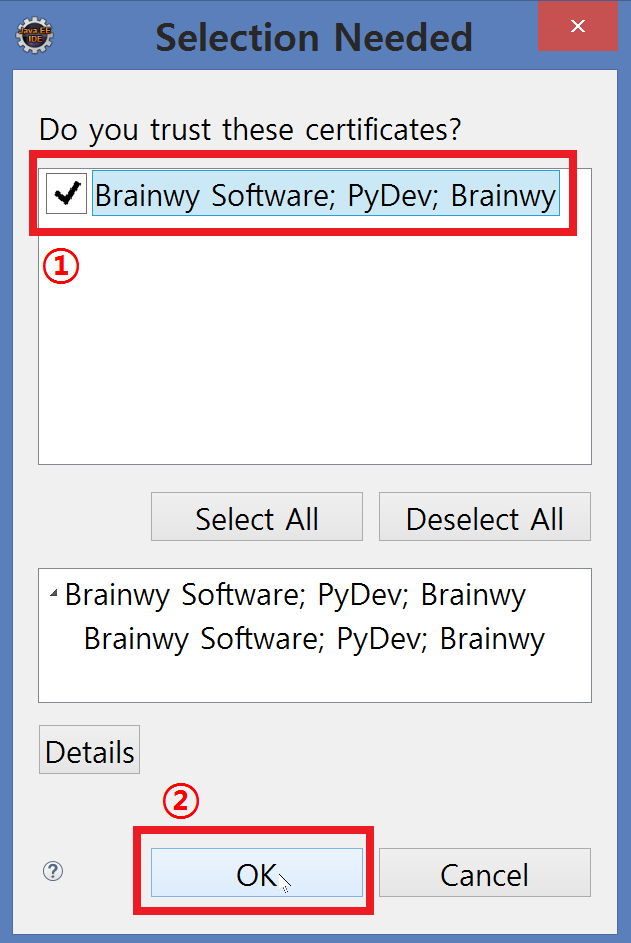


설치가 시작되면 몇 가지 사항에 대해 사용자에게 확인 및 동의를 요구한다.

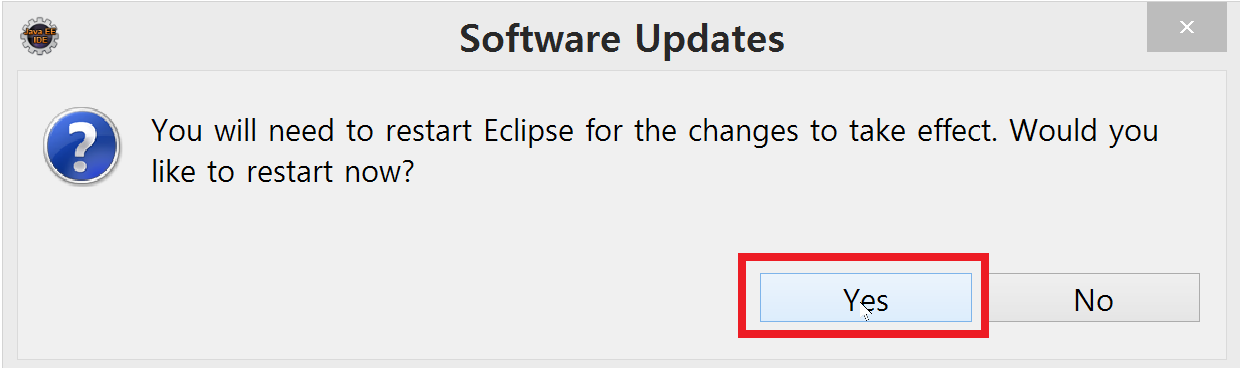
'Confirm' 버튼을 클릭하여 다음단계로 진행한다.



①번과 같이 저작권 조항에 동의를 표하고 'finish' 버튼을 클릭한다.

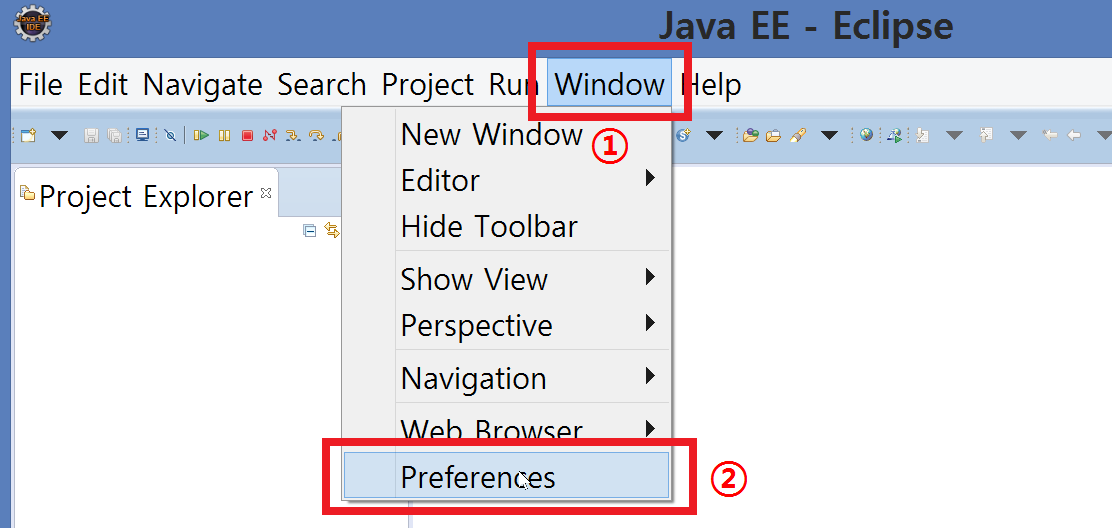


① 체크박스에 체크하고 'ok' 버튼을 클릭한다. 이 과정에서 체크박스에 체크하지 않으면 PyDev설치가 취소된다.

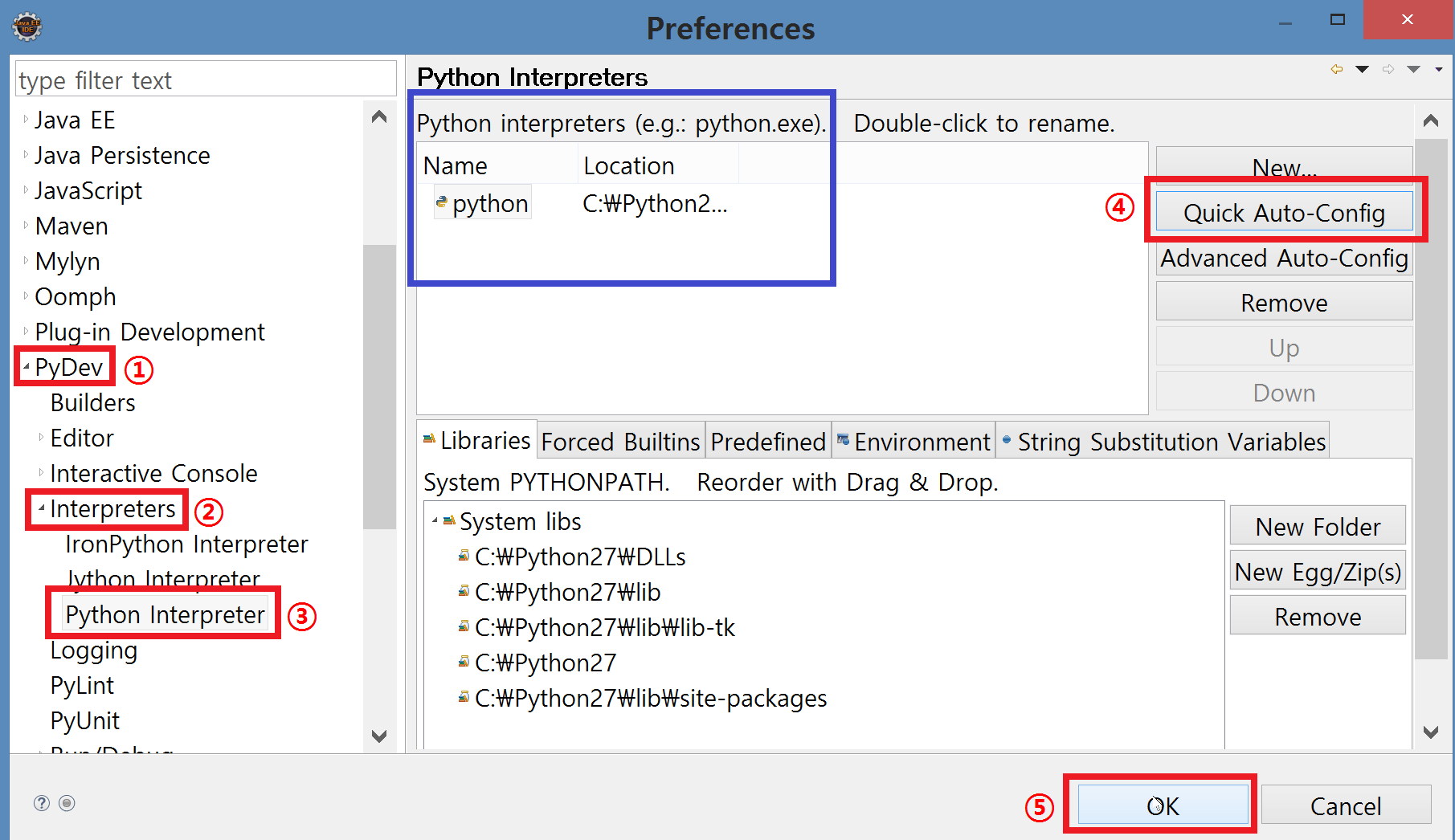


설치가 완료되면 이클립스를 재시작해야 한다. 'Yes' 버튼을 선택한다.

* + 1. python 설정

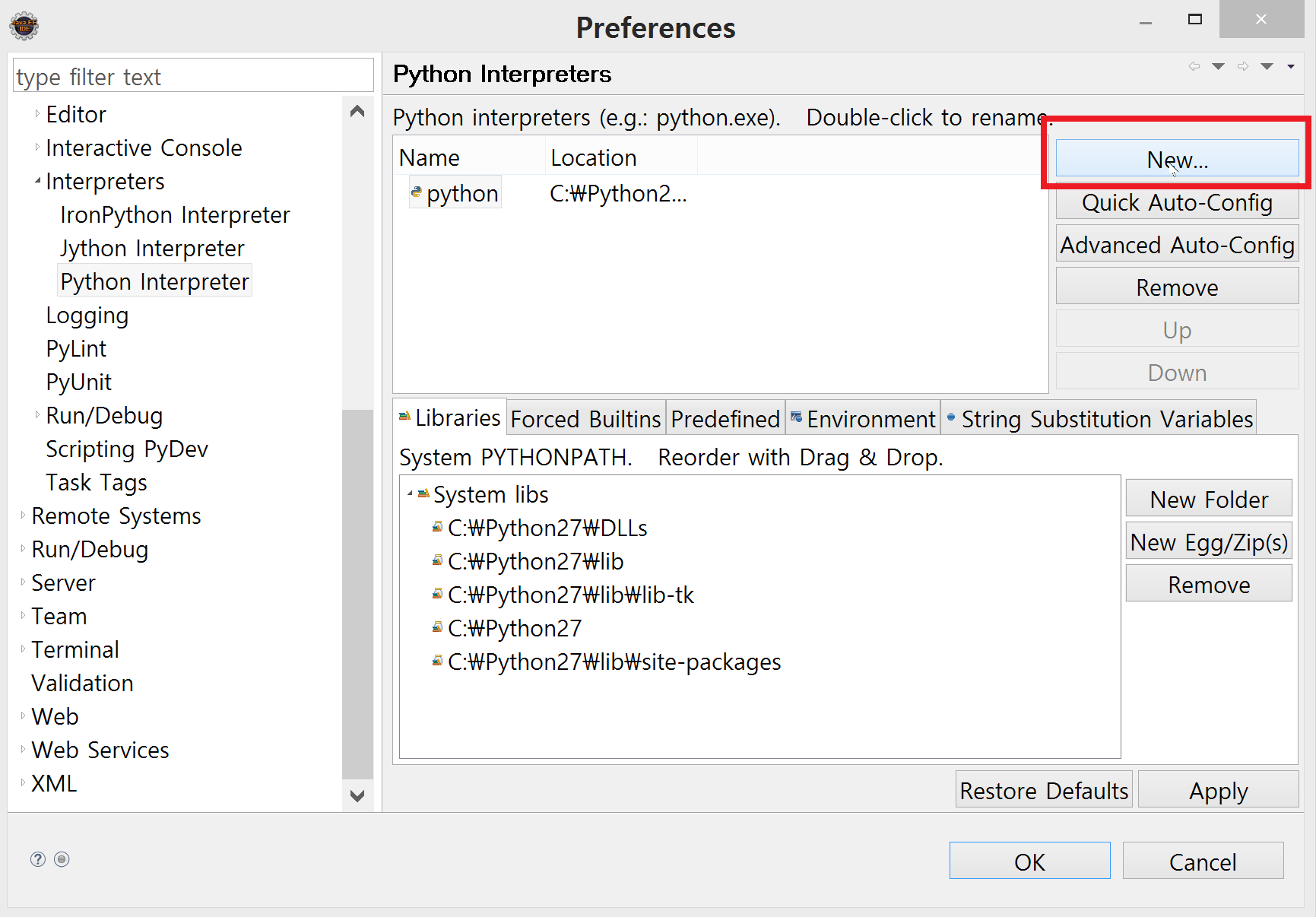


재실행된 이클립스에서 상단 Windows - Preferences를 클릭하여 설정화면을 연다.

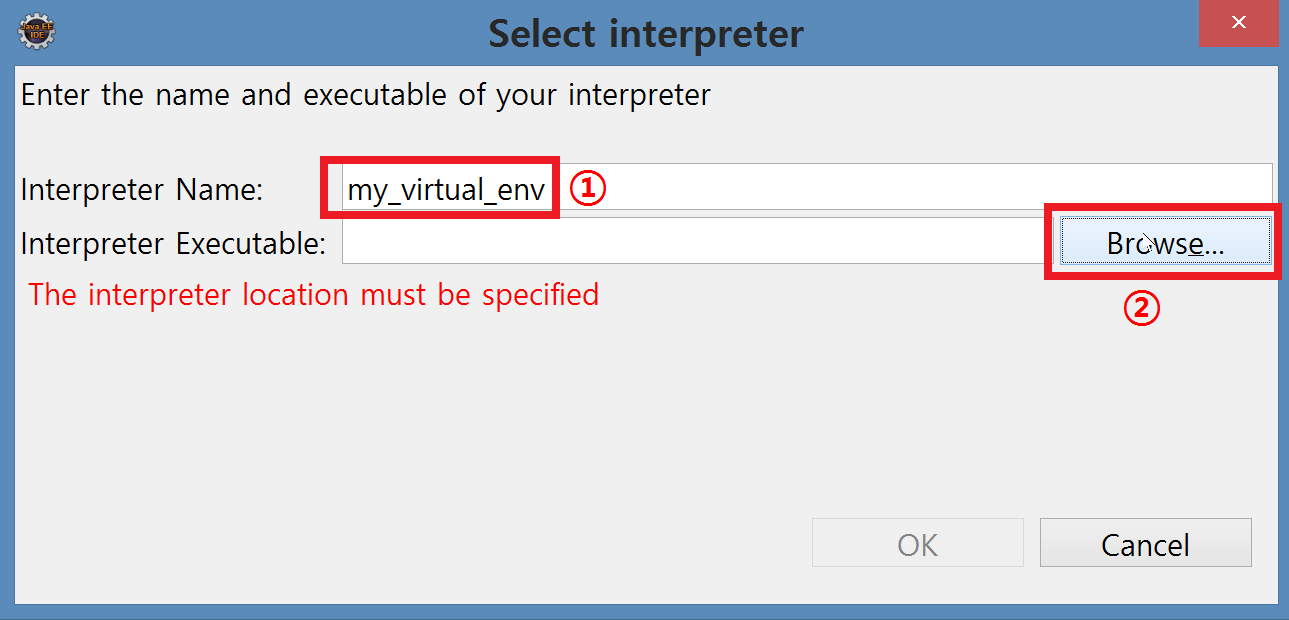


그림처럼 ①,②,③의 순서로 클릭하여 Python Interpreters 설정 화면을 연다. ④번의 Quick Auto-Config를 클릭하면 중앙의 파란색 박스처럼 설치된 python 버전을 확인할 수 있다. python 설치 시, 환경변수 설정을 하지 않은 경우는 python 정보가 제대로 나타나지 않을 수 있다.

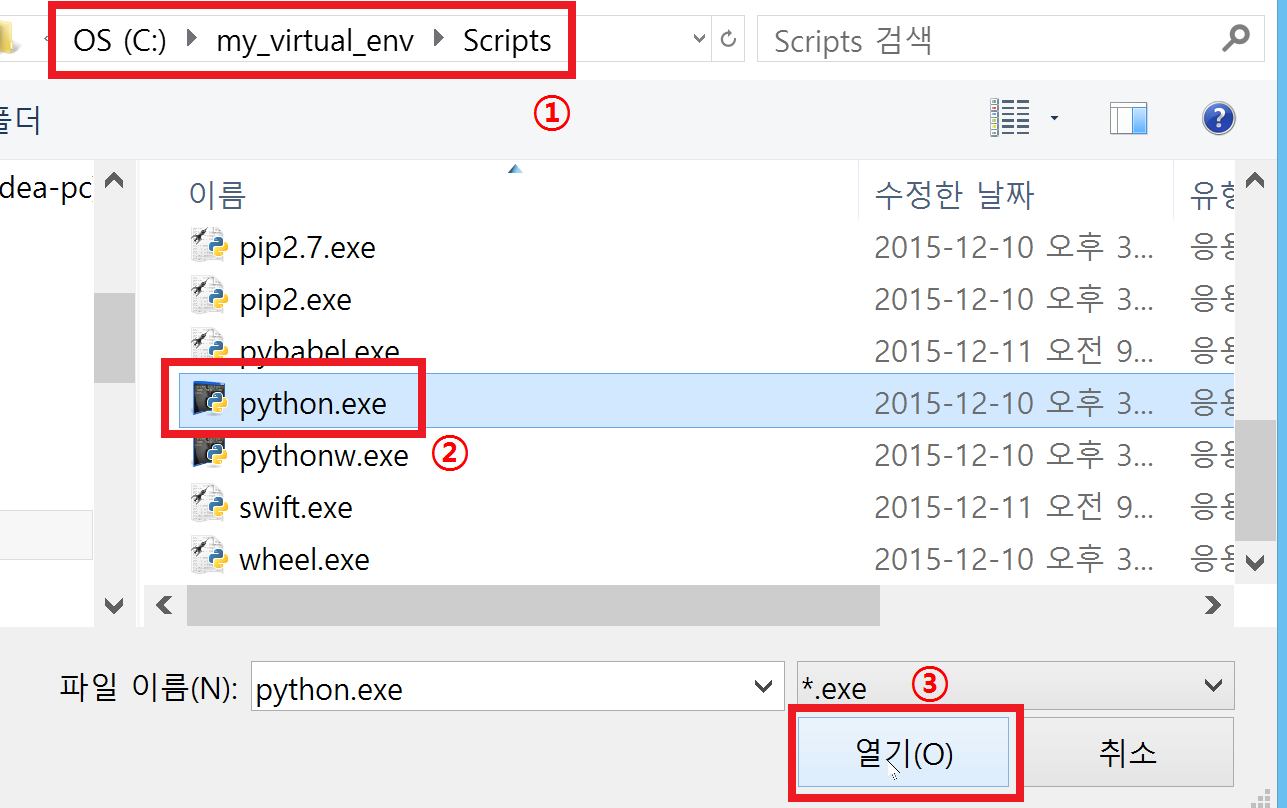
* + **가상환경을 구성하여 패키지를 설치한 사용자는 다음의 안내를 따른다.**



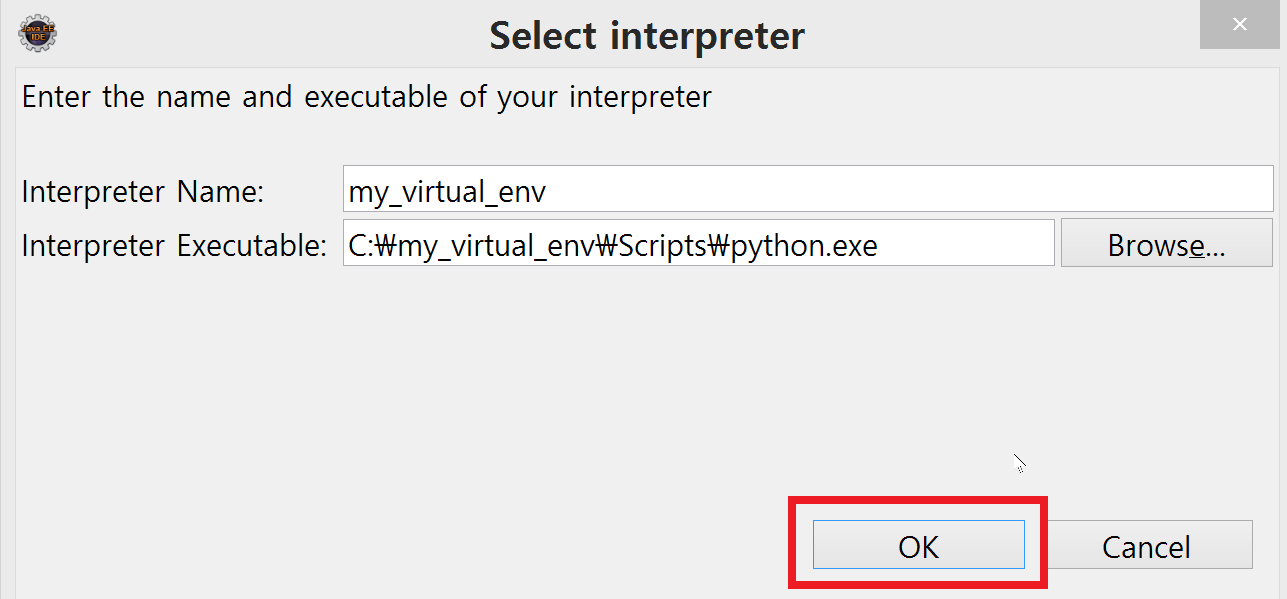
'New' 버튼을 클릭하여 가상환경을 추가한다.



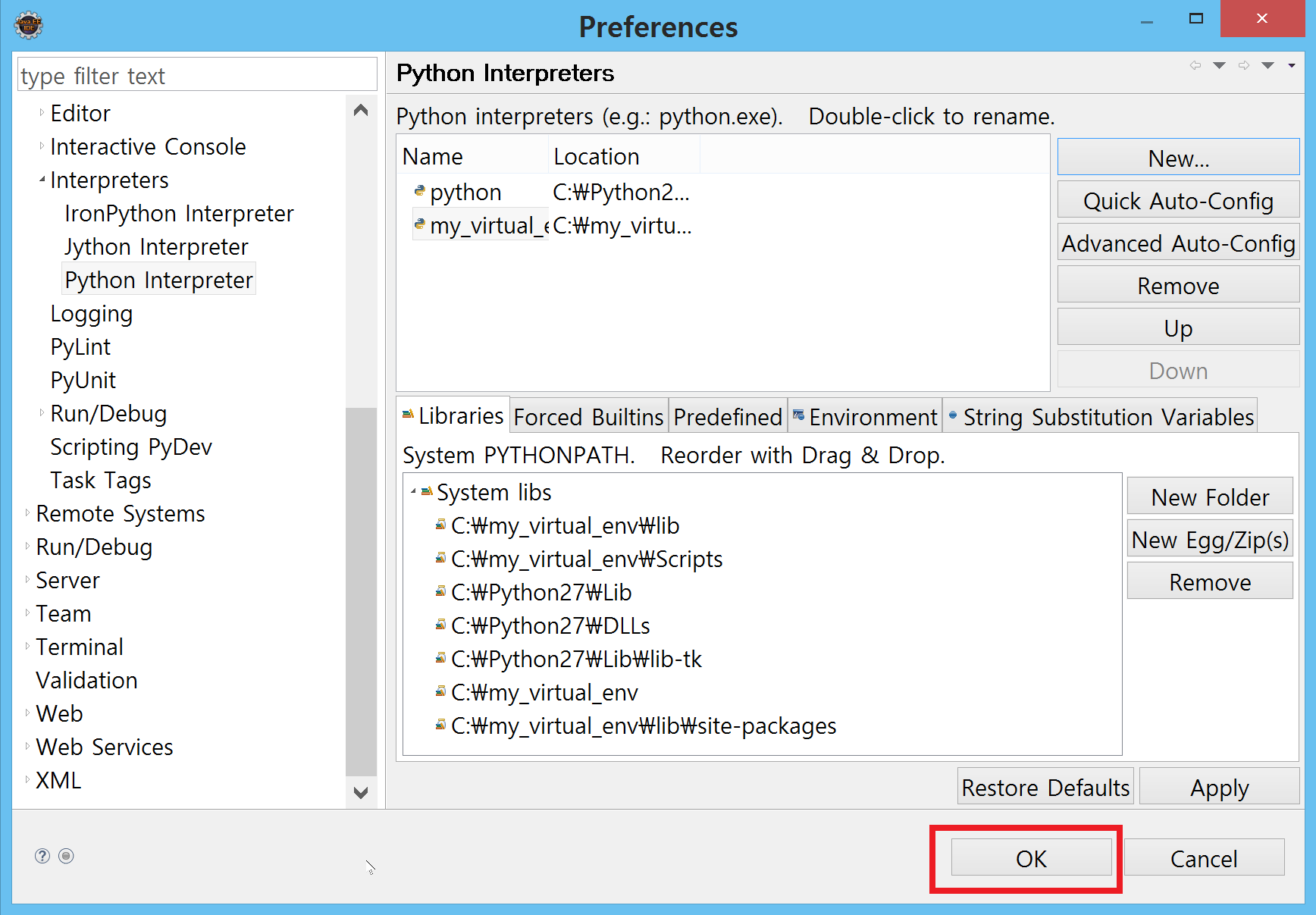
가상환경 명칭을 입력하고 Browse 버튼을 클릭한다.



가상환경의 python.exe 파일의 경로를 지정한다.

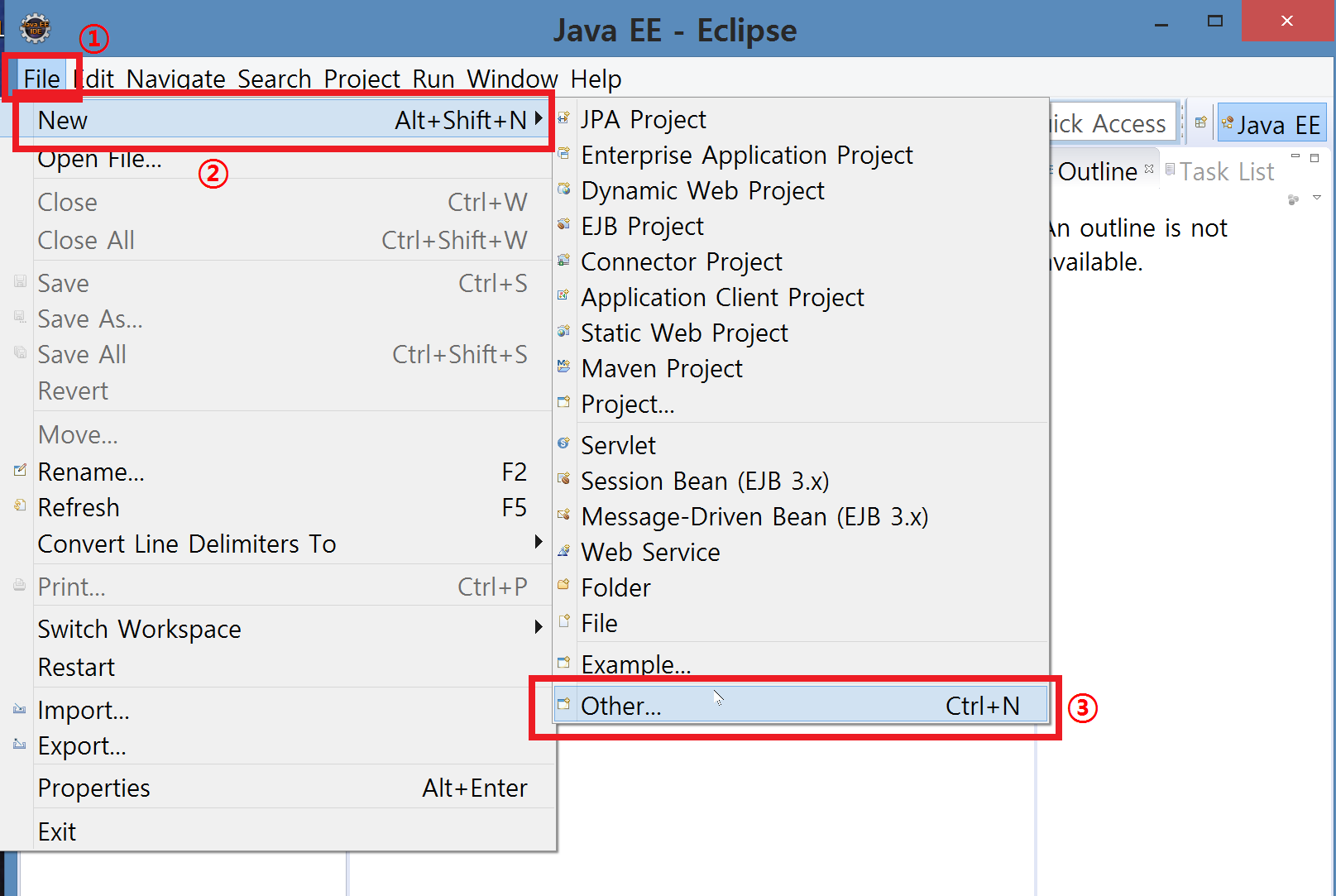


'ok' 버튼을 클릭해 인터프리터 추가를 완료한다.

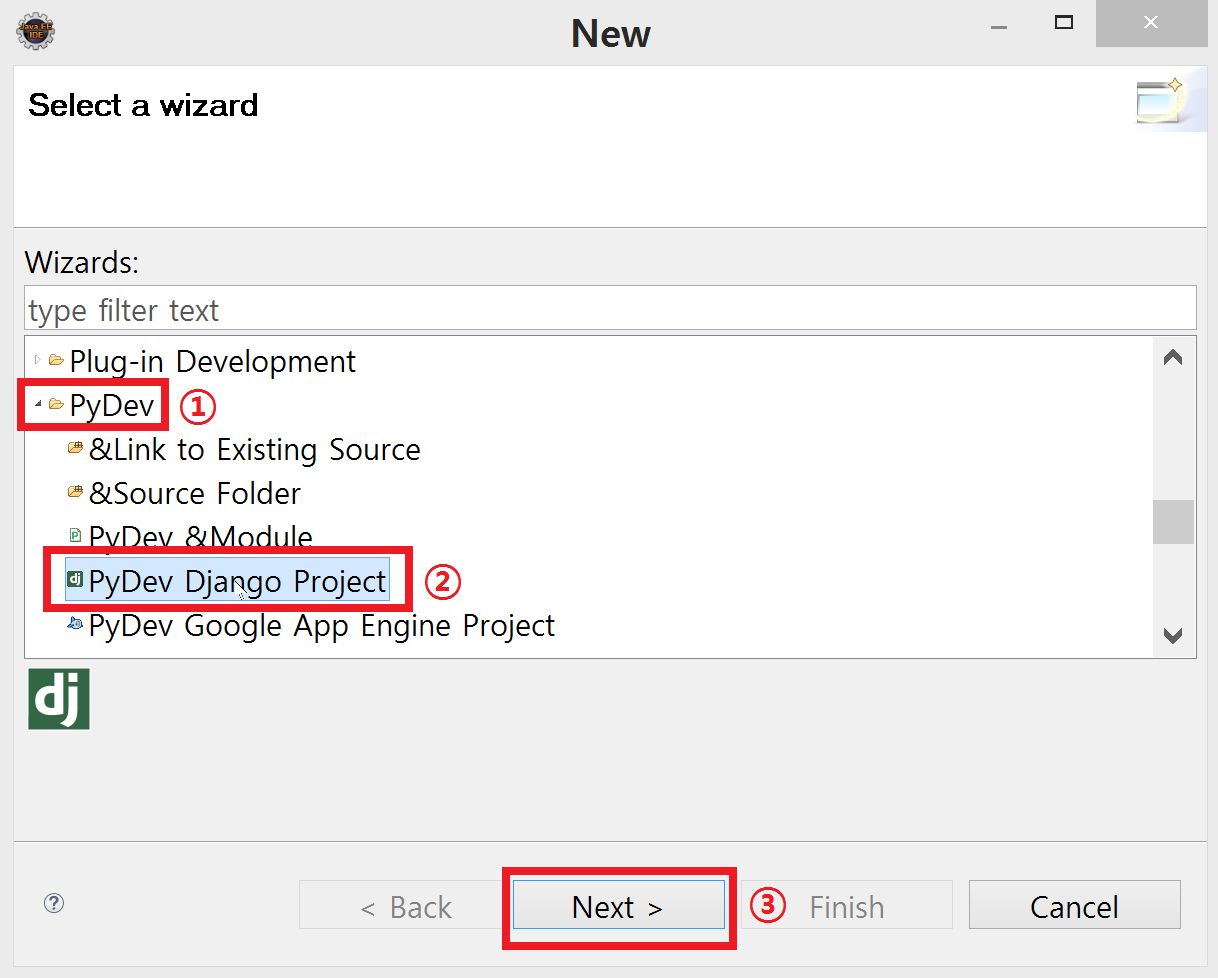


인터프리터를 추가했다면 'ok' 버튼을 눌러 설정을 완료한다.

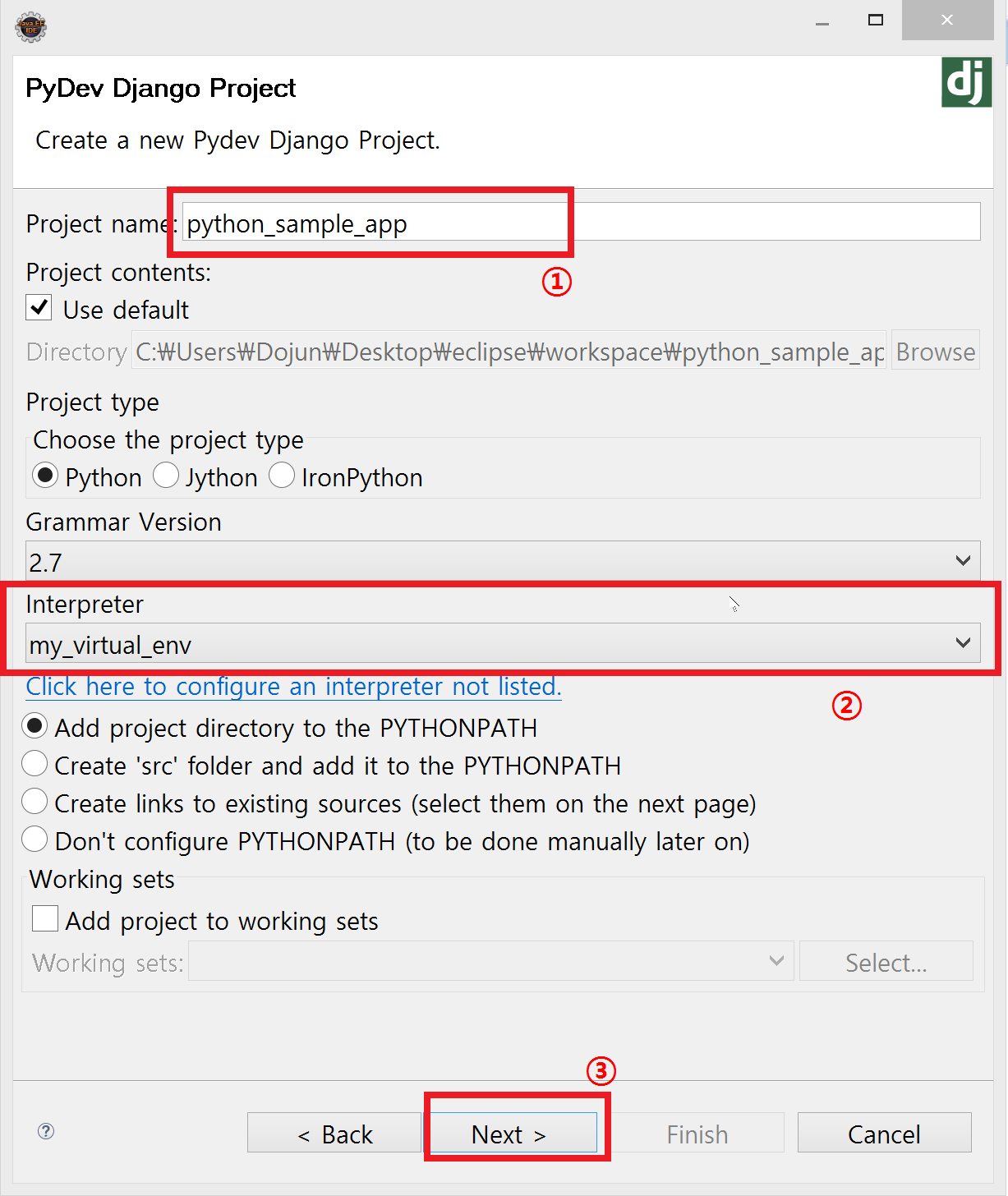
* + 1. django 프로젝트 생성



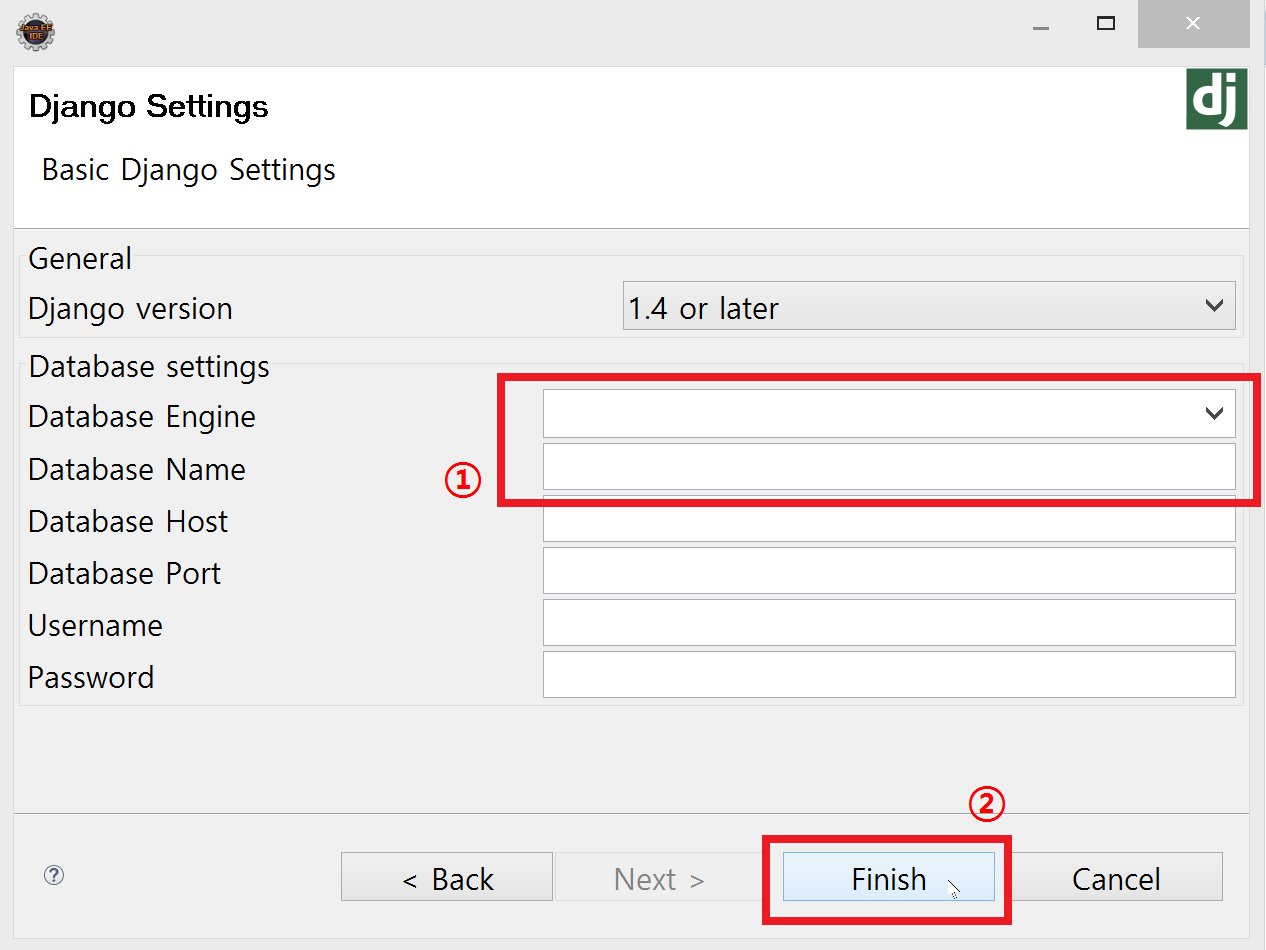
화면 상단의 File - New - Other를 클릭하여 프로젝트 생성 창을 실행한다.



①번과 같이 PyDev를 클릭해 하단 메뉴를 열고 메뉴 중 ② PyDev Django Project를 선택하고 'Next '버튼을 눌러 다음단계로 진행한다.

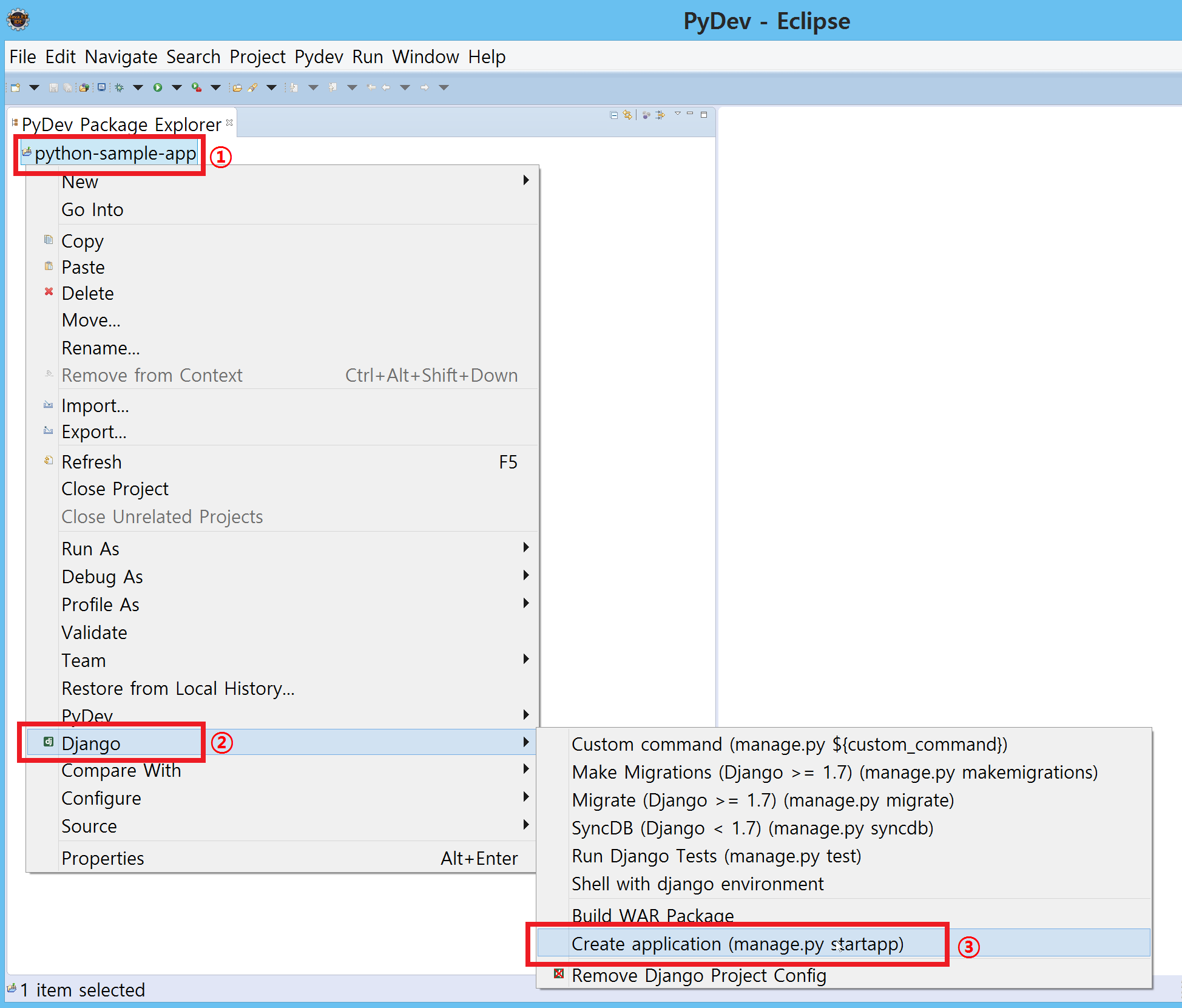


①번 위치에 프로젝트 이름을 입력한다. 주의할 점은 명령어를 사용할 때와 마찬가지로 django는 프로젝트 이름에 '-'(hyphen)을 포함할 수 없기 때문에 '\_'(underscore)로 대체한다. ②번 박스처럼 인터프리터를 선택한다. 가상환경을 구성한 사용자는 가상환경을 선택한다. 그리고 'Next' 버튼을 눌러 다음 단계로 진행한다.

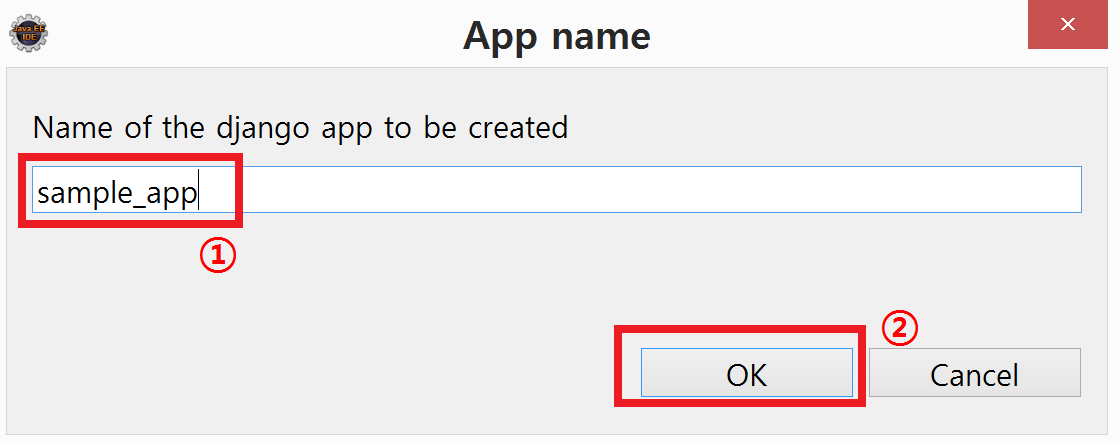


샘플 애플리케이션에서는 django의 ORM(객체 관계 매핑) 기능을 사용하지 않기 때문에 Database 설정을 하지 않는다. 그림처럼 기존의 값을 삭제하고 'Finish'버튼을 눌러 프로젝트 생성을 완료한다.

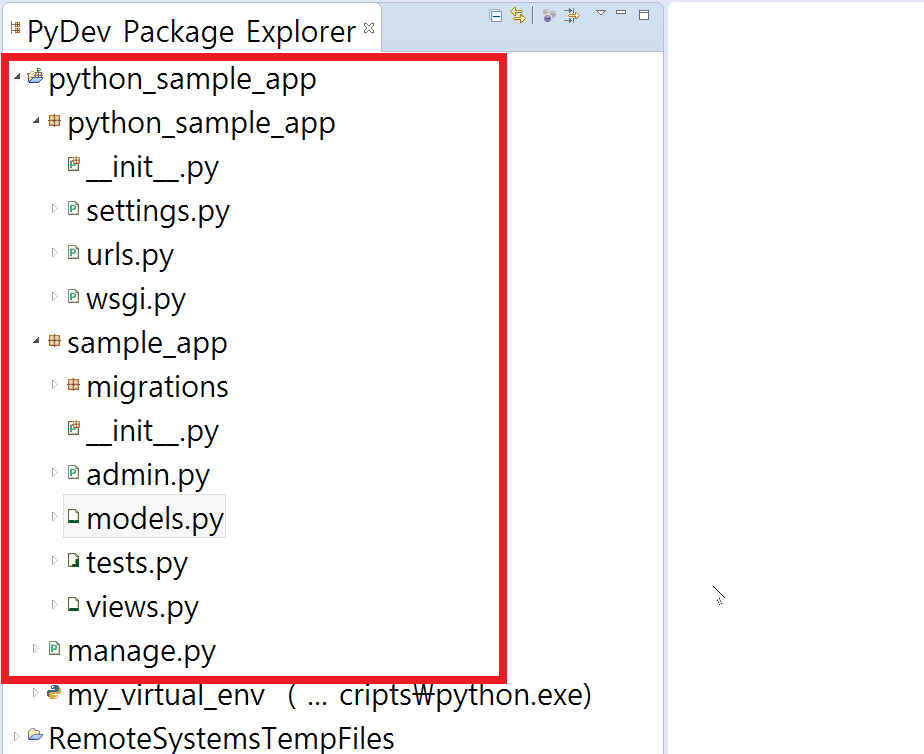
* + 1. django 애플리케이션 생성



이클립스 화면 좌측의 PyDev Package Explorer에서 생성한 프로젝트를 찾을 수 있다. 생성한 프로젝트에 마우스를 올리고 마우스 오른쪽 버튼을 클릭하여 메뉴를 연다. 메뉴에서 Django 메뉴를 찾아 'Create appliation (manage.py startapp)'을 선택한다.



애플리케이션 이름을 입력하는 창이 생성된다. 애플리케이션 이름 역시 프로젝트와 마찬가지로 '-'(hyphen)을 포함할 수 없기 때문에 '\_'(underscore)로 대체한다. 'ok' 버튼을 입력하면 django 애플리케이션 생성이 완료된다.



생성된 애플리케이션의 구조를 확인한다.

1. 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.) [↑](#footnote-ref-2)