Embedded Python 시작하기

소개

파이썬은 참 매력적인 언어입니다. 리스트, 튜플, 딕셔너리 같은 자료형들이 기본적으로 제공되고, 풍부한 라이브러리는 물론 쓰기도 편리합니다. 그리고 다른 언어와도 잘 어울립니다. 흔히 이를 일컬어 '풀 언어 (glue language)'라고 하죠.

파이썬은 C/C++에 비해 수행 속도가 느립니다. 대신 매우 생산성이 높습니다. 그러니 둘 사이를 잘 절충하여 좋은 구현을 만들어 보자는 것이 핵심이겠죠. 이는 두 가지로 방법으로 이해해볼 수 있습니다.

- 파이썬 기반 프로그램을에서, 빠른 처리가 필요한 부분을 C/C++로 만든다. **파이썬에서 C/C++ 코드를** 쓴다.
- C/C++ 기반 프로그램에서, 유연한 처리가 필요한 부분을 파이썬으로 만든다. **C/C++에서 파이썬 코드를** 쓴다.

Extending vs. Embedding

문서나 웹을 검색해보면 'extending/extended'와 'embedding/embedded'라는 표현이 상당히 자주 보입니다. 우선 이 extending, embedding이란 단어가 무얼 뜻하는지 짚고 넘어가겠습니다.

Extending

파이썬에서 extending, 즉 '확장'했다는 말은 파이썬 측에서 다른 언어를 바인딩하여 다른 언어가 제공하는 기능을 활용하는 것입니다. **파이썬에서 C/C++ 코드를** 쓰는 것은 이 예입니다. 본 문서는 이 부분을 다루지는 않습니다만 'extended python'이란 용어 자체는 '확장된 파이썬', python extension은 '파이썬 확장'이라고 해석하도록 하겠습니다.

Embedding

Extending의 반대입니다. 파이썬이 다른 코드에 '삽입'되었다는 뜻이며 **C/C++에서 파이썬 코드를** 쓴다는 것은 이쪽의 예입니다. 본 문서에서는 'embedded python'을 '삽입된 파이썬'으로, 'python embedding'은 '파이썬 삽입' 정도로 번역해 쓰겠습니다. Extending은 차후에 기회가 된다면 따로 다룰 생각이며, 본 문서에서는 삽입된 파이썬에 대해서만 다룹니다.

파이썬을 삽입하는 주체가 되는 언어는 C/C++로 한정하겠습니다. 사실 C 이외에 여러 다양한 언어가 파이썬과의 통합을 지원합니다. 심지어 같은 스크립트 언어인 Perl과도 가능합니다. 가능한 언어의 종류는 파이썬 위키를 참고하세요.

파이썬 삽입을 하게 되면 다음과 같은 장점을 가집니다.

- 파이썬 및 파이썬의 라이브러리들을 활용할 수 있습니다. 유연하고 자유분방한 파이썬을 이용해 빠르게 모듈을 생성합니다. '배터리 포함'된 파이썬 라이브러리를 사용하는 것도 꽤 멋집니다.
- 핵심 모듈을 매번 컴파일할 필요가 없습니다. 컴파일할 필요도 없을 뿐더러 모듈만 따로 파이썬 인터프리터를 통해 간편하게 테스트할 수 있습니다. 모듈 변경에 있어 상당히 유연해집니다.

그러나 다음과 같은 사항은 꼭 고려해 보셔야 합니다.

- 암만 해도 파이썬이 C/C++ 속도를 능가할 수는 없습니다. 설마 속도가 중요한 부분을 파이썬으로 구현하려는 것은 아니겠지만요.
- C/C++ 쪽에서 아무런 준비 없이 파이썬을 호출할 수는 없습니다. 그러한 인터페이스를 만드는 것 자체에 약간의 오버헤드는 있을 수 있으니, 그 오버헤드를 고려하는 것이 좋습니다.
- 언어와 언어가 교착하는 접점에서의 디버깅은 각 언어 내부에서 하는 디버깅보다는 조금 까다롭습니다. 그러므로 모듈을 만들기 전 상호간의 프로토콜을 잘 맞추어 두는 것이 좋겠습니다.
- 만들어진 소프트웨어를 배포할 경우, 파이썬의 코드가 반드시 포함되어야 합니다. Python C API의 라이브러리 뿐 아니라 의존성 있는 파이썬 스크립트까지 모두 포함해서 말입니다. 파이썬이 기본적으로 포함된 리눅스나 맥 기반이면 큰 부담은 없지만, 윈도우라면 정말 최악의 경우 사용자에게 파이썬 기본 설치를 요구할 수도 있습니다.

준비하기

파이썬 다운로드 및 설치

파이썬 언어가 시스템에 설치되어 있어야겠죠. 각 플랫폼별로 나누어 간략하게 설명하겠습니다.

Windows

윈도우는 모든 언어의 인터프리터, 컴파일러를 별도로 설치해야 합니다. 이 점은 좀 번거롭죠 
아무튼 2014년 3월 현재, Python 2.X의 최신 버전은 2.7.6이므로 문서는 이 버전의 파이썬을 기준으로 기록합니다. http://www.python.org/downloads/ 에서 윈도우용을 다운로드 받습니다. 단 64비트 시스템에서는 주의할 점이 한 가지 있습니다. 64비트 C 컴파일러를 사용한다면 파이썬도 64비트 버전을, 32비트 C 컴파일러를 사용한다면 32비트 버전의 파이썬을 설치하세요. 이걸 지키지 않으면 빌드할 때 에러가 발생합니다. 물론 64비트 시스템에 32비트 버전 파이썬을 설치해도 파이썬 인터프리터 실행이나 파이썬 스크립트 실행 만이라면 문제가 되지 않습니다.

설치한 후 환경변수 설정을 해 주어야 쓰기 편하죠. 기본적으로 `C:\Python27`에 파이썬이 설치되니까 PATH에 `C:\Python27`을 추가합니다. 아래와 같이 명령 프롬프트를 아무 경로에서 실행해 보아 파이썬 동작을 확인합니다.

```
> python --version
Python 2.7.6
```

별도로 파이썬 패키지 설치를 한다면 setuptools를 설치합니다. 환경변수를 설정한 후,
`setuptools-x.x.tar.gz`를 다운로드 받은 다음 압축을 풀고, 압축을 풀어낸 디렉토리로 이동한 후

```
> python setup.py install
```

하면 설치가 됩니다. 이렇게 하면 `C:\Python27\Scripts`에 `easy_install.exe`가 생성됩니다.

C:\Python27을 환경변수에 등록한 방법과 동일하게 C:\Python27\Scripts 또한 환경변수에 등록합니다. 마찬가지로 임의의 경로에서 easy_install을 실행하여 환경변수 등록을 확인합니다.

```
>easy_install --version
setuptools 2.2
```

'pip'도 패키지 관리에 편합니다. easy_install이 설치되었다면 간단하게 다음 명령으로 설치 가능합니다.

```
>easy_install pip
....
```

pip가 패키지 삭제 기능도 있어 좀 더 편리합니다.

본 문서에서는 직접적으로 easy_install이나 pip를 직접적으로 필요로 하지 않습니다. 설치 시 참고만 하시면 됩니다.

Linux

배포판별로 차이가 조금씩 있겠지만, 리눅스는 거의 기본적으로 파이썬이 포함되어 있습니다. 문서의 설명을 따라해 보기 위해서는 파이썬 2버전에 맞춰 주기만 하면 됩니다.

MacOS

맥도 기본적으로 파이썬이 탑재되어 있어 별도로 설정할 필요는 없습니다.

빌드 설정

Windows

비주얼 스튜디오에서 빌드하려면 따로 include path, library path를 프로젝트 설정에서 지정해 주어야 합니다. Property sheet를 사용하는 것이 편리하겠죠.

불행히도, 또는 일반적으로도 그렇듯이 유닉스 계열에 비해 윈도우 쪽이 빌드할 때 훨씬 불편합니다. 일단 윈도우에서는 이후 설명할 python-config-X.Y를 사용할 수가 없습니다. 파이썬 자체에서 제공하는 sysconfig 모듈을 이용하는 방법도 있으나 이마저도 빌드에 필요한 모든 정보가 제공되지는 않습니다. 대체로 .lib 파일은 C:\Python27\libs에 설치됩니다.

또한 디버그 버전 빌드 때에는 링킹 단계에서 python27_d.lib 파일을 찾을 수 없다는 메시지까지 나옵니다. 파이썬 공식 홈페이지에서 받은 인스톨러 패키지에는 디버그 버전의 라이브러리가 빠져 있기 때문입니다. python27_d.lib 파일을 온전히 생성하려면 파이썬 소스를 받아다 직접 컴파일해야 합니다.

참고로 파이썬은 bzip, Berkely DB, OpenSSL, SQLite, Tcl/Tk에 의존성이 있고, 또 의존성 패키지들은 Perl과

NASM 등을 필요로 합니다. 윈도우에 패키지 관리자가 있었으면 참 좋겠는데 하는 생각이 들죠. 아무튼 비주얼 스튜디오를 이용하는 경우 직접 파이썬을 빌드하지 않는 한 좀 아쉽더라도 릴리즈 모드를 이용해야 할 겁니다.

MinGW를 이용한다면 `-I` 옵션과 `-L` 옵션, 그리고 `-lpython27` 옵션까지 다 집어 넣으면 되겠습니다. 인스톨러에 `libpython27.a`가 같이 제공됩니다.

```
>gcc -o <output> <input> -I "C:\Python27\include" -L "C:\Python27\libs"
-lpython27
```

Linux

리눅스에서는 `python-config`을 통해 좀 더 편하게 설정을 알아낼 수 있습니다. 시스템에 따라 'python2.7-config'이나 'python-config-2.7' 처럼 약간 차이는 있을 수 있습니다. 이 명령어로 파이썬 자체가 빌드될 때 컴파일 옵션, 링커 플래그 등을 알아낼 수 있습니다.

`--cflags`, 혹은 `--ldflags`의 값들은 파이썬 자신이 빌드될 때의 설정인지라 반드시 준수해야 할 것은 아닙니다. 경우에 따라서 헤더 파일의 경로, 라이브러리 경로와 라이브러리 이름만 지정해도 충분합니다.

그래도 적절히 이식성 좋은 컴파일 명령어를 위해서는 다음 정도가 적절합니다.

```
$ gcc -o <output> <input> `python-config --includes` `python-config --libs`
```

MacOS

리눅스의 방법과 동일하게 진행하면 됩니다. 파이썬의 경로가 조금 다르다거나 python-config의 이름이 약간 다르다는 것을 제외하고는 거의 다른 점이 없습니다.