Cross Platform Desktop Applications With Python

Using Browser Front Ends

Python UI Toolkits

- Cross Platform Toolkits
  - GTK+
  - wxWidgets
  - Qt

* Can’t forget tk!

What's Wrong With Them?

- Don't look good/native on all platforms
  - Particularly MacOS X
- Some overlap w/Standard Library
- Often unpythonic
  - Bad doc strings
  - Useless Introspection

* Sometimes incompatibility with standard library and types
* Can't introspect opaque objects
* You can still use them to talk to this server if that's really what you want

Why Browsers?

- Capabilities & Standards compliance have improved
  - CSS 2/3
  - AJAX
- Available on almost every platform
- Native look and feel

* Browsers available on every platform python supports

Why Browsers? (cont'd)

- AJAX allows us to build more responsive apps
  - Thanks Microsoft!
- Good embedded browser choices
  - XULRunner (mozilla)
  - WebKit (safari)

* Microsoft introduced the first AJAX functionality in Internet Explorer
** Probably to cause compatibility problems, but this backfires on Windows as a platform

What yadaf isn't

- A UI Toolkit
  - So why'd we just hear about toolkits?
- A pure Python environment
  - You're still stuck writing code in the browser
- Ultimately, in Javascript
  - Or Silverlight/Flash

* Yet Another Desktop Application Framework

JavaScript? Ewww....

- Lucky you, a new brand of tools has emerged
  - GWT
    - Compiles a Java-like language to Javascript & HTML that use JSON to talk to non-Java servers
  - OpenLaszlo
    - Compiles custom XML to JS & HTML
  - Ext-JS

* OpenLaszlo is possibly the original non-Javascript AJAX tool
* Ext-JS is pure javascript, but usable
* All the toolkits abstract away browser differences

So What is yadaf?

- Half of the desktop application / browser equation
  - Server backend
- Glue between the server and your application
  - Automatic translation of data between server and client
  - Abstract away JSON/SOAP/XML-RPC particulars

How yadaf Works

- Application Server
- Data Translation Layer
  - JSON Adapter Available, others easy to write
  - Your application core
- Browser as UI
  - Enforces a logical break between UI and Backend

* Separation of business logic and front-end generally a good idea
* Makes it really easy to script your application using the web service APIs

Limiting the Magic

- I personally despise too much “magic”
- Provide up-to-date architecture diagrams
- Make it easy to understand how parts work
  - Application Server
  - Translation Adapter
  - Instance Serialization (future?)

Translation Adapters

- Implement 3 methods on Application Instances
  - _read
    - Translate POST data into application format
  - _transform
    - Convert URLs
  - _startSession

* _read takes wire data and creates a dictionary or objects to pass to application
* _transform is what turns dashes in URL names into underscores in method names, for example
* _startSession is invoked on after your instance is first created
** usually just sends the session ID to the client

Applications

- Must implement getSessionID()
  - Unique ID to identify an instance
  - Used to pass requests to the right instance
  - All methods named ws_ are exported as web service APIs using the chosen translation adapter
  - http://localhost:xxxx/yy/do-something becomes Application.ws_do_something

* In theory for a desktop application there's only one instance
* All applications are hosted on a custom port and path root

Demo!

Issues

- Instances never close
  - Not really a problem for Desktop Applications
- Multipart form parsing is blocking
- Session management could be more automatic
  - Force applications to subclass a basic application

* Really big file movements using multipart/form can block the backend and cause the UI to become unresponsive
* Basic application would ensure that all translation adapter methods exist (even if empty) and session management occurs using uuid module

Future

- Browser Plugin
  - Cover Javascript DOM functionality in plugin that exposes Python API
- Bundled UI
  - Custom XULRunner/WebKit instance
  - Javascript extensions for control over menus, popup windows, etc.

Future (cont'd)

- Compiler for Python to JS/HTML
  - Less hard if only one renderer is supported
- Non-desktop apps
  - BaseHTTPServer may not cut it
  - Serialize instances to DB
- Integrate with Google Gears?

* pyjamas - GWT-like tool using Python, maintenance status unknown

Questions/Discussion