

Flash and Python

- Dynamic
- Object oriented
- Rapid development

What is Flash?

- Byte code is interpreted by VM in Flash Player
- Actionscript code is compiled to byte code
- AS2 - Flash Player 7+, Flash Player Lite
- AS3 - Flash Player 9+
- <http://www.adobe.com/products/flashplayer/>

What is Flex?

- Flex is a framework for developing RIA apps
- AS3 / Flash Player 9+
- Collection of useful classes
- Pre-compiler with support for mxml
- Open source
- <http://flex.org/>
- <http://www.adobe.com/products/flex/>
- Flex API

What is Air?

- Flash player for the desktop
- File system access
- Native window decoration
- Built in WebKit HTML renderer
- Extends Flex
- <http://www.adobe.com/products/air/>

Talking to the server

- All Flash Players support URLRequest class
 - Async req/resp with plain text, XML, JSON...
- AS2 Players support NetConnection class
 - AMF remoting packets
- AS3 Players support RemoteObject class
 - Additional remoting features

What is AMF?

- Adobe Messaging Format
- Binary object serialization format
- AMF0 works with AS2 players
 - [AMF0 spec](#)
- AMF3 works with AS3 players
 - [AMF3 spec](#)
 - More compact

AMF Features

- Compact format
- Serialized/Deserialized by native code in the player
- Built in types
- Typed and un-typed objects
- Object references
 - AMF3 also has string and class references

AMF Example

- Python:
 - {'spam': 'eggs'}
- AMF3
 - \x0A\x0B\x01 - \x09spam - \x06\x09eggs - \x01

AMF Packets

- Packets are encoded in AMF0
- Headers and messages may be encoded in AMF0 or AMF3

```
<packet>  
  <target>string</target>  
  <response>string</response>  
  
  <headers />  
  
  <messages>  
    <remoteObject />  
  </messages>  
</packet>
```

RemoteObject Features

- Automates RPC calls
- Can switch 'channels'
 - Transparently switch between AMF, SOAP, etc.

RemoteObject Example

```
remote_object_example.as x
1 package models
2 {
3     [RemoteClass(alias="models.User")]
4     public class User
5     {
6     //.....
7
8     public function getService(url:String):RemoteObject
9     {
10        // Create the AMF Channel
11        var channel:AMFChannel = new AMFChannel("amf-channel", url);
12
13        // Create a channel set and add your channel(s) to it
14        var channels:ChannelSet = new ChannelSet();
15        channels.addChannel(channel);
16
17        // Create a new remote object and set channels
18        var remoteObject:RemoteObject = new RemoteObject("UserService");
19        remoteObject.channelSet = channels;
20        return remoteObject;
21    }
22
23    public function getUser(id:int):void
24    {
25        var service:RemoteObject = getService(myUrl);
26        var operation:AbstractOperation = remoteObj.getOperation('getUser');
27        operation.addEventListener(ResultEvent.RESULT, getUser_resultHandler);
28        operation.send();
29    }
30
31    protected function getUser_resultHandler(event:ResultEvent):void
32    {
33        var localUser:User = event.result;
34    }
```

AMF packages for Python

- PyAmf
- AmFast - my package!!
- Django AMF

- Most mature / biggest community
- Pure Python
- Supports custom class mapping
- Supports NetConnection and RemoteObject
- Built in WSGI gateway
- Supports: Django, Google App Engine, SQLAlchemy
- **Examples for major frameworks**

PyAmf cont.

- Slow for large object graphs
- Not very customizable

PyAmf Example

```
echo.py x
1 from pyamf.remoting.gateway.wsgi import WSGIGateway
2 from wsgiref import simple_server
3
4 import pyamf
5
6 class Echo(object):
7     def __init__(self):
8         self.response = 'echo'
9
10    def echo(self):
11        return self
12 pyamf.register_class(EchoClass, 'echo.EchoClass')
13
14 if __name__ == '__main__':
15     echoer = Echo()
16     services = {
17         'echo': echoer.echo
18     }
19
20     gw = WSGIGateway(services)
21     httpd = simple_server.WSGIServer(
22         ('localhost', 8000), simple_server.WSGIRequestHandler)
23     httpd.set_app(gw)
24
25     try:
26         httpd.serve_forever()
27     except KeyboardInterrupt:
28         pass
```

AmFast

- Extension written with Python C API
- ~18x faster than PyAmf
- Alpha / no community yet
- Supports custom class mapping
- Supports NetConnection and RemoteObject
- Supports SQLAlchemy
- [Example project](#)

AmFast cont.

- Calls methods on user supplied Python objects to allow for complete customization
- Not as much functionality built-in as PyAmf, but it's easier to add additional features
- Exposes packet, header, and messages objects to callable targets

AmFast Example

```
1 import os
2 import optparse
3 import cherrypy
4 import amfast.remoting as remoting
5 import amfast.class_def as class_def
6
7 def amfhook():
8     """Checks for POST, and stops cherrypy from processing the body."""
9     cherrypy.request.process_request_body = False
10    cherrypy.request.show_tracebacks = False
11
12    if cherrypy.request.method != 'POST':
13        raise cherrypy.HTTPError(400, "AMF request must use 'POST' method.");
14 cherrypy.tools.amfhook = cherrypy.Tool('before_request_body', amfhook, priority=0)
15
16 class Echo(object):
17     def __init__(self):
18         self.response = 'echo'
19
20     def echo(self):
21         return self
22
23 class App(object):
24     @cherrypy.expose
25     @cherrypy.tools.amfhook()
26     def amfGateway(self):
27         try:
28             c_len = int(cherrypy.request.headers['Content-Length'])
29             raw_request = cherrypy.request.rfile.read(c_len)
30             return self.gateway.process_packet(raw_request)
31         except Exception, e:
32             raise cherrypy.HTTPError(500, "Internal Server Error")
33
34 if __name__ == '__main__':
35     gateway = remoting.Gateway()
36     gateway.class_def_mapper.mapClass(class_def.ClassDef(Echo, 'echo.Echo',
37                                                         ('response',)))
38
39     echoer = Echo()
40     service = remoting.Service('EchoService')
41     service.setTarget(remoting.CallableTarget(echoer.echo, 'echo'))
42     gateway.service_mapper.mapService(service)
43
44     # Start server
45     app = App()
46     app.gateway = gateway
47     cherrypy.quickstart(app, '/')
```

What is RTMP?

- Real Time Messaging Protocol
- Alternative to HTTP
- Used by expensive Adobe Live Cycle Data Services server
- Supports push to client
- Java's BlazeDS does not support RTMP, but it fakes push by supporting a HTTP polling interface

RTMP in Python

- RTMPy Twisted protocol
- Still in development, but looks promising
- Developed by the 2 main developers of PyAmf
- None of the Python AMF packages currently support psuedo-push through polling, but should be easy to add to AmFast.

Blog

- Visit my blog: limscoder.blogspot.com for copies of these slides and updates about Flash and Python