# 用Tornado开发RESTful API运用

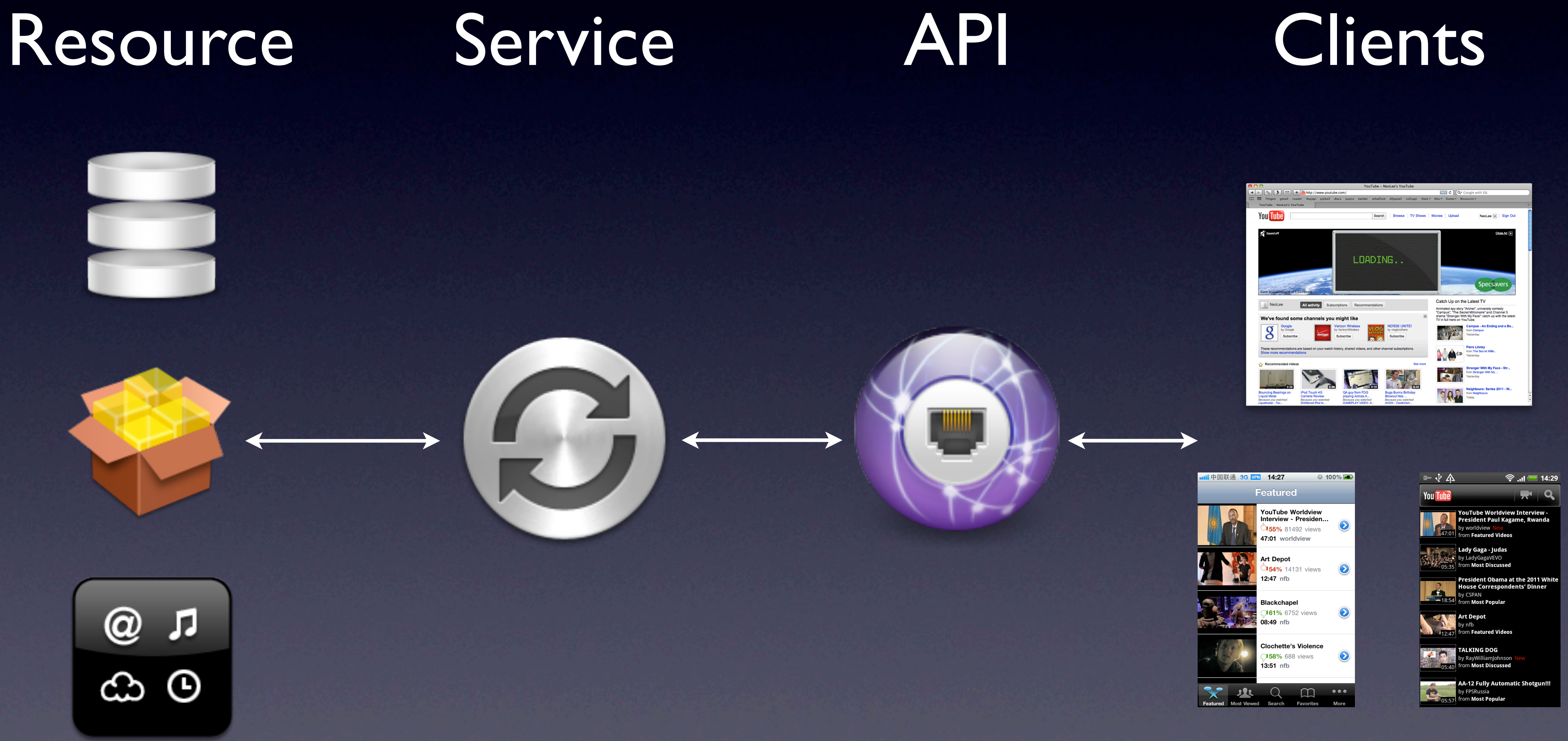
飞龙非龙 (http://feilong.me)
2012/10/20

# 议程

- RESTful API简介
- 用Tornado开发RESTful API应用
- D3status demo APP

The New York Times
Applistars
WINGA 10
my sms
skype
facebook
flickr
Cyber-shot
R.C. Central Park

# Service



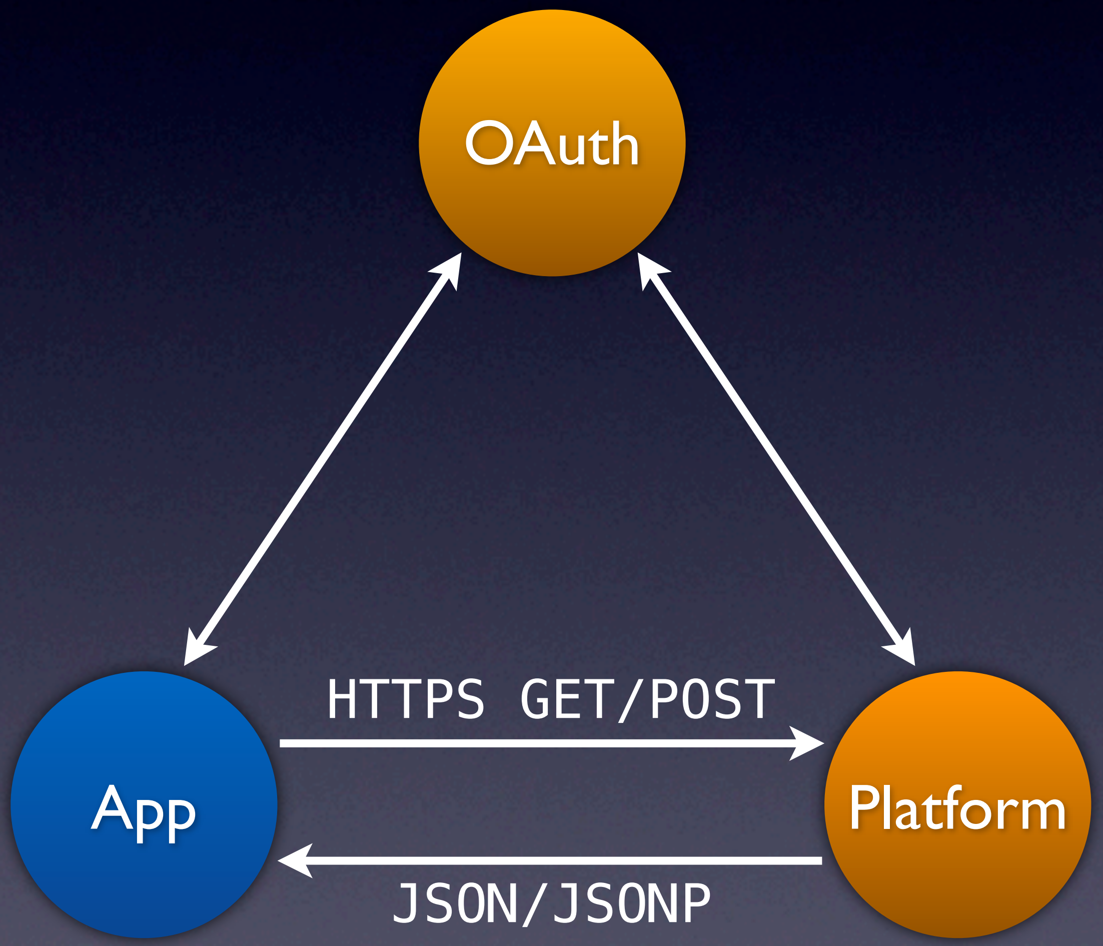
Resource

Service

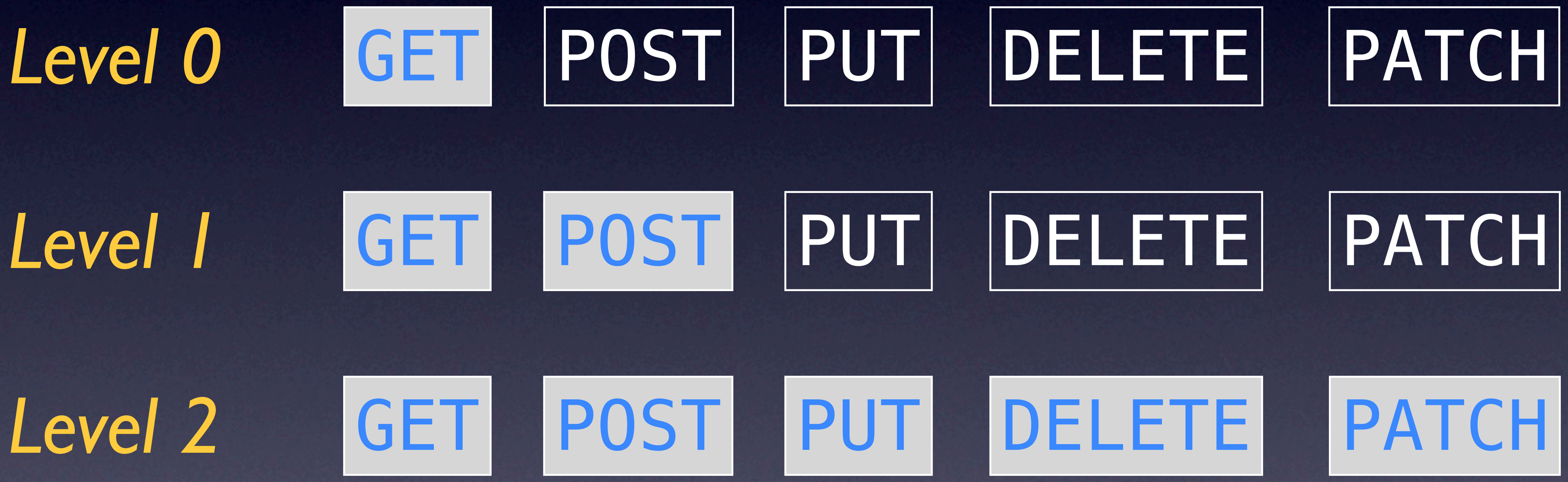
API

Clients

# RESTful API



# RESTful and HTTP Verbs



| | | | | | |
|---|---|---|---|---|---|
| *Level 0* | GET | POST | PUT | DELETE | PATCH |
| *Level 1* | GET | POST | PUT | DELETE | PATCH |
| *Level 2* | GET | POST | PUT | DELETE | PATCH |

# RESTful in Tornado

```python
class RequestHandler(object):
    """Subclass this class and define get() or post() to make a handler.

    If you want to support more methods than the standard GET/HEAD/POST, you
    should override the class variable SUPPORTED_METHODS in your
    RequestHandler class.
    """
    SUPPORTED_METHODS = ("GET", "HEAD", "POST", "DELETE", "PATCH", "PUT",
                         "OPTIONS")

    def head(self, *args, **kwargs):
        raise HTTPError(405)

    def get(self, *args, **kwargs):
        raise HTTPError(405)

    def post(self, *args, **kwargs):
        raise HTTPError(405)

    def delete(self, *args, **kwargs):
        raise HTTPError(405)

    def patch(self, *args, **kwargs):
        raise HTTPError(405)

    def put(self, *args, **kwargs):
        raise HTTPError(405)

    def options(self, *args, **kwargs):
        raise HTTPError(405)
```

# JSON & JSONP

```
class APIHandler(BaseHandler):

    def finish(self, chunk=None, notification=None):
        if chunk is None:
            chunk = {}

        if isinstance(chunk, dict):
            chunk = {"meta": {"code": 200}, "response": chunk}

            if notification:
                chunk["notification"] = {"message": notification}

        callback = escape.utf8(self.get_argument("callback", None))
        if callback:
            self.set_header("Content-Type", "application/x-javascript")

            if isinstance(chunk, dict):
                chunk = escape.json_encode(chunk)

            self._write_buffer = [callback, "(", chunk, ")"] if chunk else []
            super(APIHandler, self).finish()
        else:
            self.set_header("Content-Type", "application/json; charset=UTF-8")
            super(APIHandler, self).finish(chunk)
```

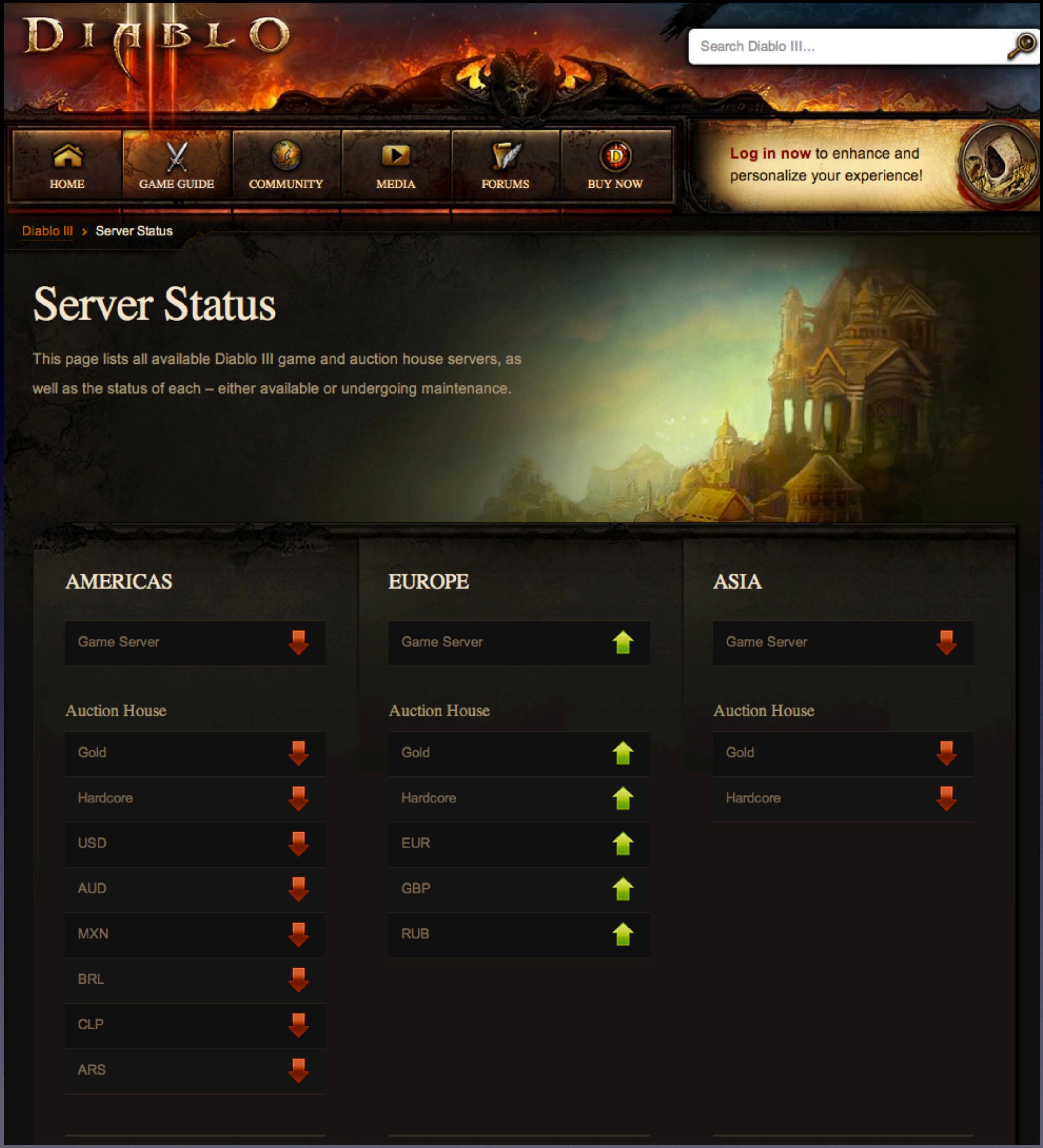
# Exception

```python
def write_error(self, status_code, **kwargs):
    """Override to implement custom error pages."""
    debug = self.settings.get("debug", False)
    try:
        exc_info = kwargs.pop('exc_info')
        e = exc_info[1]

        if isinstance(e, exceptions.HTTPAPIError):
            pass
        elif isinstance(e, HTTPError):
            e = exceptions.HTTPAPIError(e.status_code)
        else:
            e = exceptions.HTTPAPIError(500)

        exception = "".join([ln for ln in traceback.format_exception(*exc_info)])

        if status_code == 500 and not debug:
            self._send_error_email(exception)

        if debug:
            e.response["exception"] = exception

        self.clear()
        self.set_status(200)  # always return 200 OK for API errors
        self.set_header("Content-Type", "application/json; charset=UTF-8")
        self.finish(str(e))
    except Exception:
        logging.error(traceback.format_exc())
        return super(APIHandler, self).write_error(status_code, **kwargs)
```

# Exception

```
class HTTPAPIError(HTTPError):
    """API error handling exception

    API server always returns formatted JSON to client even there is
    an internal server error.
    """
    def __init__(self, status_code=400, error_detail="", error_type="",
                 notification="", response="", log_message=None, *args):

        super(HTTPAPIError, self).__init__(int(status_code), log_message, *args)

        self.error_type = error_type if error_type else \
            _error_types.get(self.status_code, "unknow_error")
        self.error_detail = error_detail
        self.notification = {"message": notification} if notification else {}
        self.response = response if response else {}

    def __str__(self):
        err = {"meta": {"code": self.status_code, "errorType": self.error_type}}
        self._set_err(err, ["notification", "response"])

        if self.error_detail:
            err["meta"]["errorDetail"] = self.error_detail

        return escape.json_encode(err)
```

DIABLO



**Log in now** to enhance and personalize your experience!

# Server Status

This page lists all available Diablo III game and auction house servers, as well as the status of each – either available or undergoing maintenance.

## AMERICAS

| | |
|---|---|
| Game Server | ↓ |

### Auction House

| | |
|---|---|
| Gold | ↓ |
| Hardcore | ↓ |
| USD | ↓ |
| AUD | ↓ |
| MXN | ↓ |
| BRL | ↓ |
| CLP | ↓ |
| ARS | ↓ |

## EUROPE

| | |
|---|---|
| Game Server | ↑ |

### Auction House

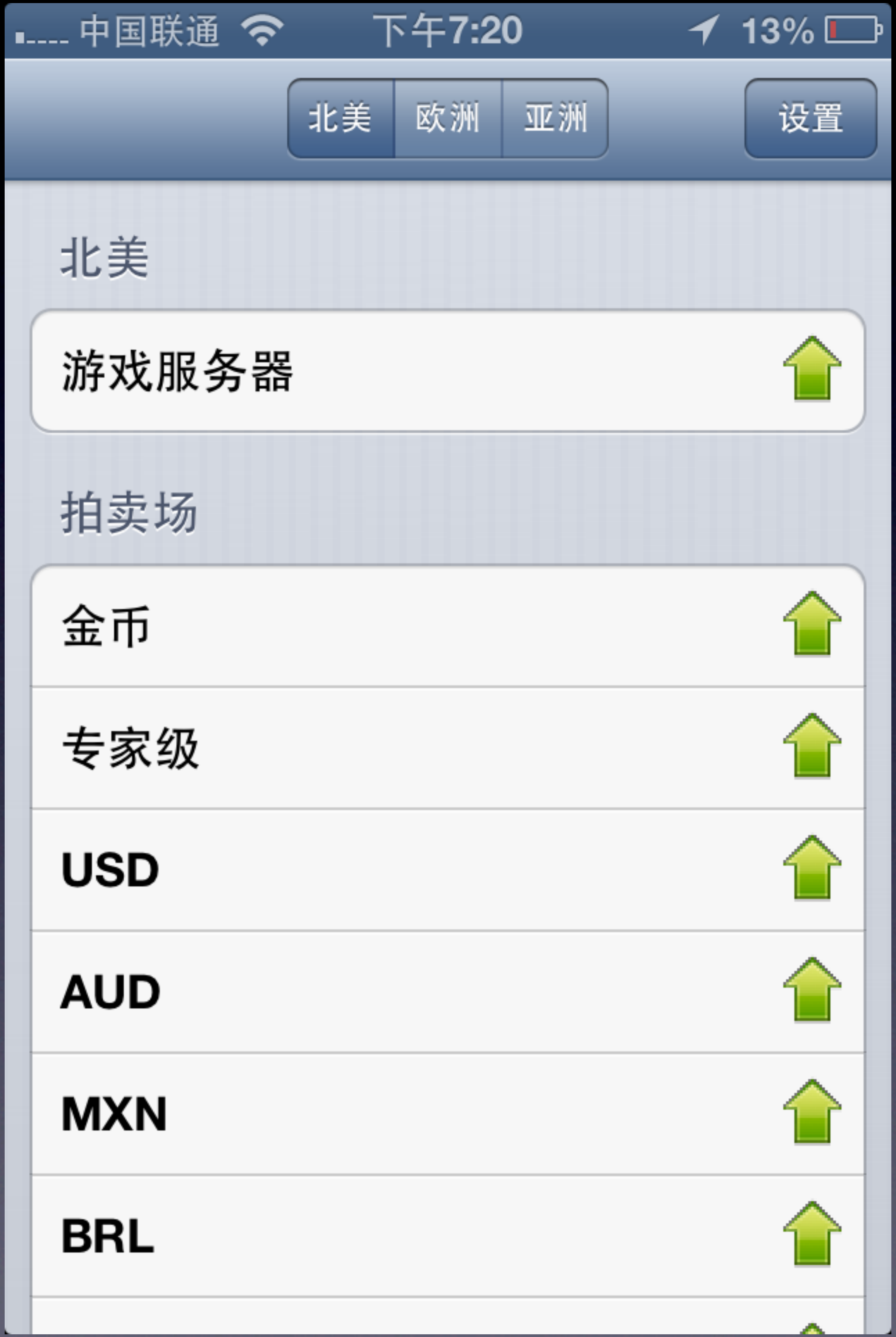
| | |
|---|---|
| Gold | ↑ |
| Hardcore | ↑ |
| EUR | ↑ |
| GBP | ↑ |
| RUB | ↑ |

## ASIA

| | |
|---|---|
| Game Server | ↓ |

### Auction House

| | |
|---|---|
| Gold | ↓ |
| Hardcore | ↓ |



北美

游戏服务器 ⬆

拍卖场

金币 ⬆

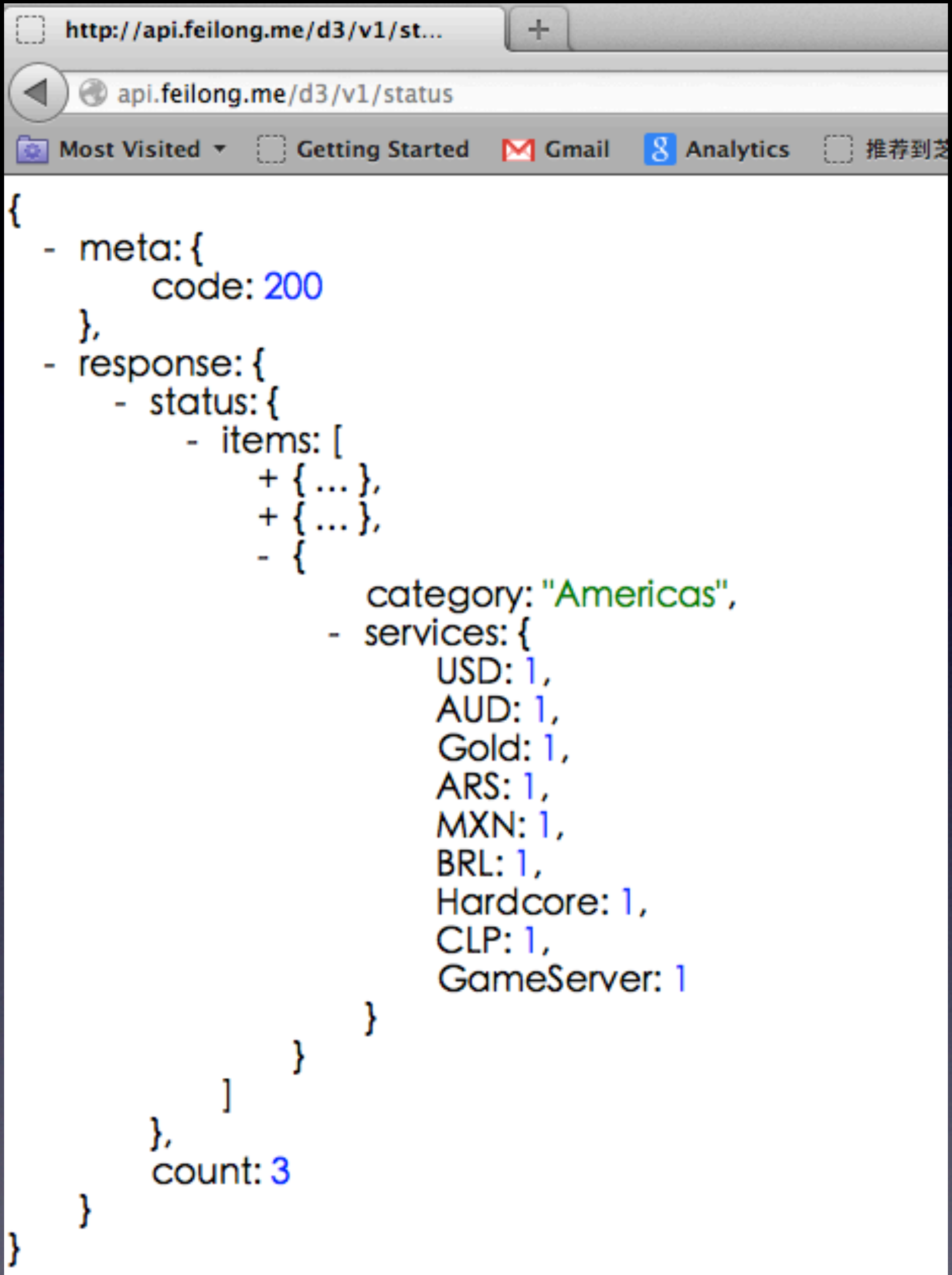
专家级 ⬆

USD ⬆

AUD ⬆

MXN ⬆

BRL ⬆


http://api.feilong.me/d3/v1/st...
+
api.feilong.me/d3/v1/status
Most Visited
Getting Started
Gmail
Analytics
推荐到

```
{
  - meta: {
        code: 200
    },
  - response: {
      - status: {
          - items: [
              + { ... },
              + { ... },
              - {
                    category: "Americas",
                  - services: {
                        USD: 1,
                        AUD: 1,
                        Gold: 1,
                        ARS: 1,
                        MXN: 1,
                        BRL: 1,
                        Hardcore: 1,
                        CLP: 1,
                        GameServer: 1
                    }
                }
            ]
        },
        count: 3
    }
}
```

# 网页抓取

```
def update_server_status():
    url = options.d3_server_status_url
    req = HTTPRequest(url=url)

    client = HTTPClient()
    response = client.fetch(req)
    if response.code == 200:
        status = _parse_server_status(response.body)
```

# 网页解析

```
def _parse_server_status(body):
    status = {}

    q = pq(etree.fromstring(body))
    boxes = q(".box")  # category box
    for box in boxes:
        box_q = pq(etree.fromstring(etree.tostring(box)))
        category = box_q(".category")[0].text.strip()
        status[category] = {}
        servers = box_q(".server")
        for server in servers:
            server_q = pq(etree.fromstring(etree.tostring(server)))
            server_name = server_q(".server-name")[0].text.strip().replace(" ", "")
            if server_name:
                status_icon = server_q(".status-icon")[0]
                class_ = status_icon.get("class")
                if class_:
                    st = 0
                    if "up" in class_:
                        st = 1
                    status[category][server_name] = st

    return status
```

# 任务队列

```
@task
def status_notification_task(changed_status):
    status_notifciation(changed_status)


def status_notifciation(changed_status):
    notifications = {}
    for category, services in changed_status.iteritems():
        for name, st in services.iteritems():
            # just push notification about game server now
            if name == "GameServer":
                notifications[category] = st

    for category, st in notifications.iteritems():
        status = "Available" if st else "Unavailable"

        offset = 0
        limit = 200
        while True:
            subscribers = load_model("subscribers").get_subscribers(limit, offset)
            if not subscribers:
                break

            for subscribe in subscribers:
                if category in subscribe.categorys:
                    alert = _trans_alert("Diablo3 %s server status has changed to %s",
                                         category, status, subscribe.locale)
                    apns_tasks.apns_push_task.delay(subscribe.token, {},
                                                    alert=alert, badge=1,
                                                    sound="default")
            offset += len(subscribers)
```

# 其它

- Apple push notification
- i18n
- crontab

# 相关资源

- https://github.com/felinx/d3status
- http://www.tornadoweb.org
- http://www.tornadoweb.cn
- http://tornado.poweredsites.org
- http://tornadogists.org
- http://en.wikipedia.org/wiki/Representational_state_transfer

# Q & A

@飞龙非龙

http://feilong.me/