

Twisted

アリエルネットワーク(株)
おおたに

スレッドは体によくない

今日の指令

- Deferred(コールバック) を理解する
- Web へ
- Twitter(<http://twitter.com/>) で遊ぶ (宿題)

Deferred を理解する

- Twisted はイベント駆動型プログラミングモデル
- つまりは全てはコールバックで処理
- コールバックの制御は deferred オブジェクトが担当
- コールバックはチェーンになる
- エラーが起きたらエラーバック
- エラーバックもチェーンになる

Deferred の例

```
from twisted.internet import reactor
from twisted.web.client import getPage
```

```
def mycallback(result): (2) コールバック関数
    print "you got data"
    return result
```

```
deferred = getPage("http://www.liris.org/blog/")
deferred.addCallback(mycallback) (3) コールバック関数登録
```

```
reactor.run() (1) イベントループ
```

Deferred オブジェクト

- `addCallback(func, *args, **kwds)`
 - コールバック関数を追加
- `addErrback(func, *args, **kwds)`
 - エラーバック関数 (エラー処理) を追加
- `func` : コールバックされる関数
- `*args, **kwds` はコールバック関数に渡される引数

Deferred を作ってみよう

```
from twisted.internet.defer import Deferred
deferred = Deferred()
deferred.callback("my callback result")
```

→ `callback(result)`

- `deferred` に登録されたコールバック関数を実行

→ `errback(failure)`

- `deferred` に登録された `errback` 関数を実行

コールバック関数はチェーンになる

- ひとつの deferred オブジェクトに複数のコールバック関数を登録できる

```
deferred.addCallback(  
    myfunc1).addCallback(  
    myfunc2).addCallback(  
    myfunc3)
```

- addCallback の返り値は deferred 自体
- まるで Unix Pipe のよう

課題： deferred を体感しよう

- deferred を作ってコールバック関数を登録してください
- deferred の callback メソッドをコールして凝るバック関数が実行されることを確認してください
- deferred に複数のコールバック関数を登録して実行されるのを確認してください。(関数が返す result を少しずつ変化させると面白いことが起こります)

課題のスケルトン

```
from twisted.internet import reactor  
from twisted.internet.defer import Deferred
```

```
# ここにプログラムを記述
```

```
reactor.run()
```

Web サーバを作ってみる

- twisted.web (HTTP/1.0)
 - 安定している
 - パフォーマンスに難
- twisted.web2 (HTTP/1.1)
 - API などが変わるかも
 - 難しい

ファイルを **feed** する Web サーバ

```
# myweb.py
# ブラウザで http://localhost:1853/ にアクセス
from twisted.internet import reactor
from twisted.web import static, server
root = static.File(".")

site = server.Site(root)
reactor.listenTCP(1853, site)

reactor.run()
```

Web へアクセス

- `twisted.internet.web.client`
- `twisted.internet.web2` もありますがまだ危険
 - API が変わるかも
 - 使いかたが難しい
- 単純なことは `getPage`
- 複雑なことは `Protocol` などの仕組みがわからないと難しい
 - `Proxy` を通すとか

getPage

- `def getPage(url, contextFactory=None, *args, **kwargs)`
- `*args, **kwargs` は
 - `method='GET'`
 - `postdata=None`
 - `headers=None agent="Twisted PageGetter"`
 - `timeout=0`
 - `cookies=None`
 - `followRedirect=1`

課題： Web ページを取得しよう

- 先ほど作った Web サーバに対して、適当なページを getPage を使って取得して標準出力に出力してみてください
- errback も登録してエラーも処理してみてください
- トップページにある Web サーバのコンテンツを全部取得してみてください。
- 全部取得するときに、同時アクセスする数を制限するには？

Twitterへの準備が完了！

- API(<http://groups.google.com/group/twitter-development-talk/web/api-documentation>)
- Twitter の API は REST 形式
- ユーザ認証は Basic 認証
- 結果は XML,JSON,RSS/ATOM で取得可能
 - 取得する情報によって選択できるフォーマットが限定される
- 友達のタイムライン (friends_timeline)
 - http://twitter.com/statuses/friends_timeline.xml
- 更新 (update)
 - <http://twitter.com/statuses/update.xml>

宿題：Twitterで遊ぼう

- 友達の更新情報を取得しよう
- 更新できるように頑張ってみよう