設計ガイド-v0.01

cfneoDocs-003
ver0.01

cfneo-Project

1

本文書の位置づけ

- cfneo-Projectからのご提案
  - cfneo-Projectの目的
    ＝ソフトウェア・テストを何とかする
    為の、cfneo-Projectからのご提案です
    - この文書では、
      テストというタスクを効果的に連携させる為の設計
      を目的とした設計について記述しています。
    - この文書は、
      内部設計・詳細設計と呼ばれる工程に
      に対応する事を想定しています。

cfneo-Project

2

用語

- 設計
  - 仕様の決定を指します。
- 仕様
  - ソースコード、各種ソフトウェアの設定が満たすべき機能を指します。
- テスト
  - 対象が、仕様を満たしているかどうかを評価するタスクを指します。
- タスク
  - 実際の作業を指します。

前提

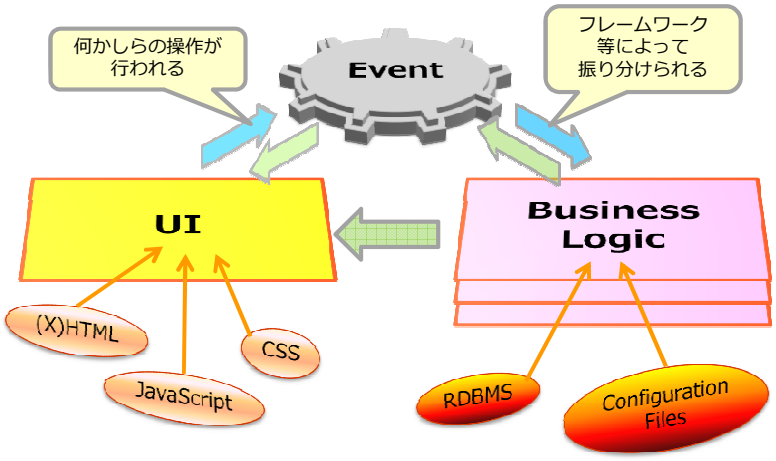
- 設計・開発対象は、以下を前提とします。
  - Webアプリケーション
  - cfneoの使用

設計(0:前置き-その1)

- UIと業務ロジックの関係
  - UIとはHTMLの生成によって作られるWebページです。
    このWebページは、(X)HTML、JavaScript、CSSなどのテクノロジによって構成されています。
  - このUIを操作される事で、イベントが発生します。
  - イベントは、発生後その結果がUIに反映される事で終了します。
  - ロジックとは、イベントから業務的な意味を持つ業務ロジックへの繋ぎを担う処理です。
  - 業務ロジックとは、イベントが発生してから終了するまでの間行われる処理です。

設計(0:前置き-その2)

- 前置きをモデル化。



補足-1

ColdFusionの実行

ColdFusion上で稼働するCFMLプログラムは、HTTP-REQUEST(get/post)のどちらかでのみ実行可能です。

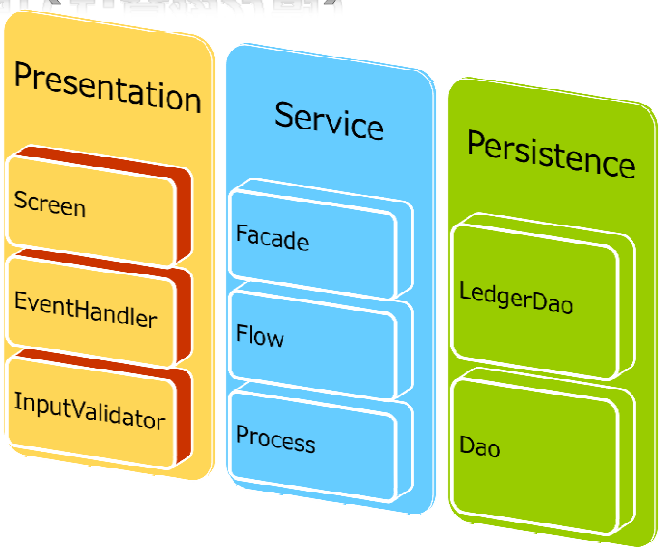
その為、Module(Function/Method)単位にテストを行うのに非常に不向きな実行環境と言えます。

それを解消する為に、cfneoにはCfneoUnitという仕組みがあります。

設計(1:責務分割)

- cfneoではFile／Moduleを責務によって分割します。
  - 責務＝役割です

設計(1:責務分割)



cfneo-Project

設計(1:責務分割)

- Presentation
  - Screen
    - UIとなるファイルです。
      - cfneo-rule:このファイルでSQLの実装は非推奨
  - EventHandler
    - UIからイベントを受け取るファイルです。
    - InputValidator、Facadeから投げられる例外は、基本的にここでキャッチします。
  - InputValidator
    - イベントとともに受け取ったFormのデータなどに対して、データベースアクセスを必要としない検査をします。

cfneo-Project

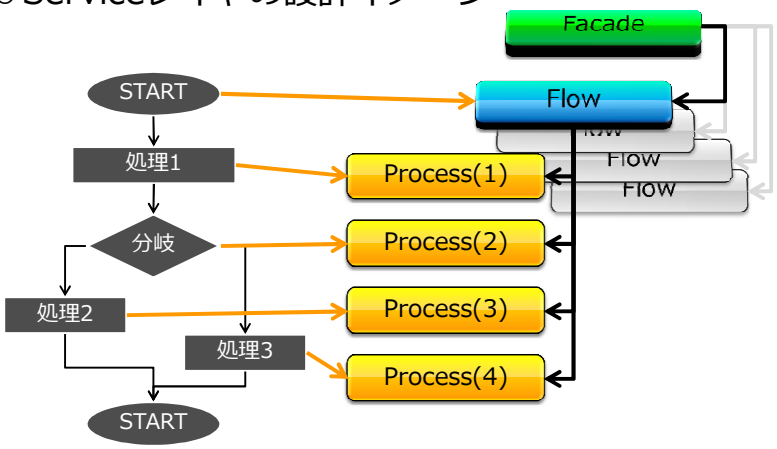
設計(1:責務分割)

- Service
  - Facade
    - EventHandlerからFlowに制御を移譲する為の文字通りの窓口です。
      - cfneo-rule:Flowに対してTransactionをかけたい場合は、ここで実装します。
  - Flow
    - Processの実行順序を定義します。
    - 全ての処理はProcessで実装し、ここには実装しません。
      - cfneo-rule:分岐の為の条件は、Processの戻り値で行います。
  - Process
    - 具体的な処理を実装します。

cfneo-Project

設計(1:責務分割)

- Serviceレイヤの設計イメージ



cfneo-Project

設計(1:責務分割)

- Persistence
  - LedgerDao
    - 画面表示用のJoinされたクエリを実装します。
      - cfneo-rule:画面表示用のクエリ以外以外は実装しません。
  - Dao
    - CRUDパターンのSQLを実装します。
      - cfneo-Rule:Transactionは原則としてここでは実装しません。

設計
(2:モジュールのインターフェイス)

- Screen/EventHandler
  - ScreenからEventHandlerを使用するには、インターフェイスは、HTTP-METHODのgetまたはpostを使用します。
  - EventHandlerからScreenを出力をする場合は、cfincludeまたはcflocationを使用します。
    - cfneo-rule:cfincludeを使用すると、URLが一定しなくなる為、非推奨です。
      - cfneoでは、一時的にSessionを利用する事でデータの受け渡しをサポートするsetValue、getValue、isExistValueが提供されています。

設計
(2:モジュールのインターフェイス)

- EventHandler/InputValidator
  - EventHandlerからInputValidatorはCFMLまたはActionScriptによるInvokeを行います。
    - InputValidatorのpublicメソッドは、validateの1種類のみ、引数は検査したいものを指定します。
      - 審査したいものは、URLScope、FormScope、Sessionにストレージされたデータとなります。
      - データベース・アクセスを要するチェックをここに実装するのは推奨できません。
  - InputValidatorはvoidで実装されます。
    - 入力審査結果でメッセージを出し分けたい場合は、例外オブジェクトを用います。

設計
(2:モジュールのインターフェイス)

- EventHandler/Facade
  - EventHandlerからFacadeは、 CFMLまたはActionScriptによるInvokeを行います。
    - Facadeから例外が返される場合は、InputValidatorと同様にEventHandlerでcatchを行います。
  - Facadeの提供するメソッドは、Flowのインターフェイスと基本的に同じものにします。
    - いくつかのFlowを束ねる場合、Flowのコンポーネント名をベースにするなどで対応します。

設計
(2:モジュールのインターフェイス)

- Facade/Flow
  - FacadeからFlowは、CFMLまたはActionScriptによるInvokeを行います。
    - その際、Flowから先でTransactionを貼らずに済むよう、FlowのInvokeをcftransactionで囲みます。
  - 原則として、Facadeの1メソッドでFlowの1つのみを使うようにします。
  - Flowからの戻り値の取扱い
    - cfneo-rule:Rollbackが発生する際に画面の遷移などを制御したい場合は、cftransactionでRollbackを実行後、cfthrowまたはcfrethrowを用いて、Eventhandlerに例外を戻すようにします。

cfneo-Project

設計
(2:モジュールのインターフェイス)

- Flow/Process
  - FlowからProcessは、CFMLまたはActionScriptによるInvokeを行います。
  - Flowに業務ロジックは実装しません。
    - cfneo-rule:基本的に命令をすべてProcessに委譲し、分岐はProcessから返される戻り値で判断するようにすると、効果的なUnitTestを利用可能になります。

cfneo-Project

設計
(2:モジュールのインターフェイス)

- Process/other
  - Processから、必要であればDaoや共通ユーティリティを使用します。
    - 当然Processも使用可能です。ただし、FacadeやFlowを使用するのは非推奨です。
  - Processでのみロジックが実装されます。

設計
(2:モジュールのインターフェイス)

- Dao
  - INSERT/SELECT/UPDATE/DELETE等各種SQLを実装します。
    - cfneo-rule:SELECT系のメソッドの戻り値は、QueryObjectとします。

設計(3:モジュール連携)

- 各モジュールの連携は、#getComponentManager()で返されるComponentManagerを使用することで、アプリケーション名を自動解決されるようにできます。

cfmからの利用

```
getComponentManager().getComponent("****")
```

cfcからの利用

```
THIS.comMgr.getComponent("****")
```


設計(4:アプリケーション配置)

- cfneoで作られたアプリケーションは、必ず2階層目をアプリケーション・ノードとします。

一般的なURL

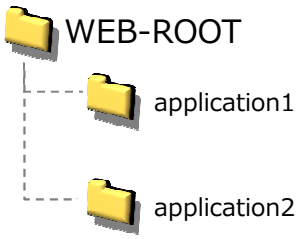
http://hoge.ne.jp/index.cfm

cfneoの推奨URL

http://hoge.ne.jp/**appname**/index.cfm

設計(4:アプリケーション配置)補足

- アプリケーションをWebサーバのトップノードに配置しないのは？
  - 開発時にいくつかのバージョンを同時に配置出来るようにする為です。



サポート

- 不明な点や過不足は以下まで
  お問い合わせください。
  - Blogに突撃
    - 思い悩むBlog(http://d.hatena.ne.jp/itengineer/)
  - メールで突撃
    - 今井(itengineer1978@gmail.com)