

# ADELのためのテスト機構

## 簡易仕様書 Ver 0.1

知能メディア工学研究室  
eラーニンググループ

平成 20 年 2 月 26 日

### 目 次

1	はじめに	2
2	システムの概要	3
2.1	動作に必要な物	3
2.2	動作環境の構築方法	3
3	外部仕様	4
4	内部仕様	6
4.1	インタフェースと主処理	6
4.1.1	主要変数	6
4.1.2	処理の大まかな流れ	6
4.2	出題モジュール	8
4.2.1	各メソッドの外部仕様	8
4.3	評価モジュール	10
4.3.1	各メソッドの外部仕様	11
5	これから改善すべき点	13
6	更新履歴	14

## 1 はじめに

このマニュアルは開発段階で書いた簡易的なもので所謂叩き台である。バージョン管理を行った上での改良は自由だが、変更を加えた場合はマニュアルに適宜反映させる、バージョン管理でログを残すなど適当な処置を施す事。

また、仕様書の内容には手違いにより誤りが含まれている可能性があるため、プログラムのソースコードと照らし合わせて読むことを推奨する。万が一誤りがあった場合は、早急に修正する事。

## 2 システムの概要

### 2.1 動作に必要な物

### 2.2 動作環境の構築方法

### 3 外部仕様

テスト機構への要求は、すべてインタフェースである `adel_exam.cgi` に対して行われる。テスト機構が提供する機能は以下の通り。

1. テスト全体の出題
2. プレ評価（選択肢を選んだ時点での事前評価）
3. 本評価（解答ボタンを押した時点での最終的な評価）
4. 正規化したテストの成績（テスト全体の評価）
5. 今現在出題しているテストの固有識別子

これらの機能の呼び出しは、`adel_exam.cgi` の引数 `mode` に特定の値を指定する事で行う。

例) `./adel_exam.cgi?mode=set&user_id=manabu`

以下に上記の機能呼び出し際の具体的な入出力について述べる。

1. テスト全体の出題

入力

`mode` set  
`src` 問題呼び出し記述 (examination 要素以下)  
`user_id` ユーザ ID

出力

xhtml(body 要素の子要素。body 要素は含まない)

2. プレ評価

入力

`mode` pre\_evaluate  
`ques_pkey` 問題の固有識別子 (xhtml 中の input 要素の属性値から取得)  
`value` 選んだ選択肢の識別子 (xhtml 中の input 要素の属性値から取得)  
`type` 問題の種類 (現状では radio のみサポート)  
`selected` 選んだ選択肢 (xhtml 中の input 要素の属性値から取得)【今回の実装では未使用】

出力

`e_result` 要素。この要素は評価結果をテキスト要素として保持している。デバッグに用いていたので、実際には使用していない。

3. 本評価

入力

`mode` evaluate  
`ques_pkey` 問題の固有識別子 (出題したテストの input 要素の属性値から取得)

**name** 選んだ選択肢 ( xhtml 中の input 要素の属性値から取得 )【今回の実装では未使用】

出力

問題単位の評価結果。評価結果、問題文、選択肢、解説を含んだ xhtml。これを問題を提示している div 要素に innerHTML メソッドを用いて表示させる。

#### 4. 正規化したテストの成績

入力

**mode** result

**test\_key** テストの固有識別子 ( 下記のテストの固有識別子から取得 )

出力

テスト全体の評価結果。問題グループと点数をコロン、この組をカンマで区切る文字列を返す。

例 ) group1:10,group2:50

未解答の問題がある場合、その問題は誤答したものとして扱われる。

#### 5. テストの固有識別子

入力

**mode** get\_testkey

**user\_id** ユーザ ID

出力

user\_id で指定されたユーザに出題された最新のテストの固有識別子。テストの成績の問合せに用いる。

## 4 内部仕様

### 4.1 インタフェースと主処理

adel\_exam.cgi は、インタフェースとしての機能の他に、各機能別モジュールを操作し処理を行う C 言語で言う所の main 関数に近い働きを行う。

#### 4.1.1 主要変数

データベースに接続するための設定は、以下に挙げる adel\_exam.cgi の変数に格納する。

base\_eXist\_host eXist が起動しているホスト名

base\_eXist\_port eXist に接続するためのポート番号

base\_pgsql\_host PostgreSQL が起動しているホスト名

base\_pgsql\_port PostgreSQL に接続するためのポート番号

pgsql\_user\_name PostgreSQL に接続するためのユーザ名

pgsql\_user\_passwd PostgreSQL に接続するユーザのパスワード（平文である事に注意）

各データベースに対して問合せを行うための設定一覧を以下に挙げる。

base\_db\_uri テスト問題を格納している問題 DB

base\_xslt\_all\_uri テスト画面全体を作るための XSLT スタイルシート

base\_xslt\_eval\_uri 各問題の画面を作るための XSLT スタイルシート

テストの出題と評価に関する設定

base\_inputType\_uri 問題形式から xhtml の input 要素の type 属性値を決定するための変換テーブル

その他の設定

base\_err\_uri エラー時に表示する xhtml

#### 4.1.2 処理の大まかな流れ

このプログラムの処理は、実行時に渡された mode の引数によって 5 つの処理に大別される。mode の値は、params にハッシュの一部として格納される。この mode キーの値によって実行すべき処理を決定する。

例) puts params["mode"] # mode キーの値を表示

以下に、それぞれの処理の大まかな流れを述べる。

## テスト全体の出題

テストの出題は、問題呼び出し記述を受け取り、その内容を一度ハッシュに格納する。ハッシュは出題する問題ごとに作られ、最終的にはハッシュのリスト (setTable 変数) となる。ここではこれを出題テーブルと呼ぶ。出題テーブル作成の際に、問題グループからランダムや問題形式を指定と言った出題に対して、実際に出題される問題が決定される。また、テスト中に出題された各問題とテストその物に対してそれぞれ固有識別子が付与される。出題された各問題に付与された固有識別子は、プレ評価と本評価の際に用いられる。

次に、この出題テーブルを用いて出題履歴を記録され、さらに実際に提示するテスト画面となる xhtml の作成が行われる。xhtml の作成には、一度中間的な xml 文書が作成され、これを XSLT を用いて xhtml と変換を行う。

## プレ評価

プレ評価は、テストが記述された xhtml 中の input 要素が持つ type 属性値と value 属性値を、問題 DB 中の問題記述が持つ評価用の情報とを照らし合わせることで行う。どの問題を出題したかという問題 DB に対する問合せは、出題履歴を用いて行われる。評価は正解の重みと不正解の重みを用いた評価が行われ、評価結果が評価履歴に記録される。

## 本評価

本評価では、プレ評価で行った評価結果をもとに解答を確定する。この処理は出題した各問題ごとに行われる。まず、プレ評価結果が存在するかを評価履歴に対して問合せを行い、無かった場合は未解答として評価履歴に記録する。これは後ほど正規化の際に、誤答として扱われる。プレ評価結果があった場合は、一番最新のプレ評価結果に対して確定済みのマークを評価履歴に対して記録する。

次に、学習者に提示するための評価結果を記述した xhtml の生成を行う。この時、評価履歴中に記録された点数の閾値と実際に獲得した点数との比較を行い、閾値に達していた場合は正解、達していなかった場合は不正解といった提示を行う。

## 正規化したテストの成績

テストの成績の正規化では、確定された評価結果を用いて正規化を行う。確定された解答が無い問題に対しては、評価履歴に未解答として記録し、その問題を誤答として扱う。正規化の計算は評価用モジュールによって行われ、ハッシュとして得られた結果を書式化し、これを返り値として返す。

## テストの固有識別子

テストの固有識別子では、受け取ったユーザ ID に出題されたテストの一番新しい固有識別子を返す。

## 4.2 出題モジュール

出題に関する機能を持つモジュールは `set_question.rb` である。このモジュールはインタフェース兼主処理を行う `adel_exam.cgi` から呼び出され、使用される。以下に、出題モジュールが持つメソッドの外部仕様を示す。

### 4.2.1 各メソッドの外部仕様

- `initialize`

概要 `Set_question` クラスのインスタンス生成時に実行される。

入力

`user_id` ユーザ ID

出力

無し。最後に評価された変数の値が返る事があるが、内容については保証しない。

- `make_table`

概要 問題呼び出し記述を内部的な出題テーブルに変換。

入力

`input_xml` 呼び出し記述 (`REXML::Element` クラスのオブジェクト)

`base_eXist_host` eXist が動作しているホスト名

`base_eXist_port` eXist に接続するためのポート番号

`base_db_uri` eXist 中にある問題 DB の URI (`set_table` メソッド中で使用)

出力

作成された出題テーブル。データ構造はハッシュのリスト。

例)

```
{{"group_id" => グループ ID, "mark" => 配点, "item_id" => 問題 ID, "ques_pass" => 評価用閾値, "ques_type" => 問題形式, "selection_type" => 出題形式, "ques_correct" => 履歴を考慮した出題, "time" => 出題時間, "test_key" => テストの固有識別子}, {...}}
```

- `set_table`

概要 ランダム、問題形式を指定と言った出題に対して、実際に出題する問題を決定する。

`make_table` メソッドから呼び出される。

入力

`tbl` `make_table` メソッドで作成した出題テーブル (ハッシュのリスト)

`base_eXist_host` eXist が動作しているホスト名

`base_eXist_port` eXist に接続するためのポート番号

`base_db_uri` eXist 中にある問題 DB の URI

出力

出題する問題がすべて確定した出題テーブル。



- `make_xml`

概要 出題テーブルから XSLT で変換するための中間 XML 文書を作成。

入力

`tbl make_table` メソッドで作成した出題テーブル (ハッシュのリスト)  
`base_eXist_host` eXist が動作しているホスト名  
`base_eXist_port` eXist に接続するためのポート番号  
`base_db_uri` eXist 中にある問題 DB の URI  
`base_inputType_uri` 問題形式から xhtml の input 要素の type 属性値を決めるための変換テーブル

出力

生成した中間 XML 文書 (REXML::Element オブジェクト)

- `make_xhtml`

概要 作成した中間 XML 文書から xhtml を作成。

入力

`input_xml` `make_xml` メソッドで作成した中間 XML 文書 (REXML::Element オブジェクト)  
`base_eXist_host` eXist が動作しているホスト名  
`base_eXist_port` eXist に接続するためのポート番号  
`base_xslt_uri` eXist 中にある XSLT スタイルシートの URI

出力

作成した xhtml (REXML::Element オブジェクト)

- `get_item`

概要 問題グループの ID と問題候補の ID から該当する問題記述を問題 DB から取得。

入力

`group_id` 問題グループの ID  
`item_id` 問題グループ中の問題候補の ID  
`base_eXist_host` eXist が動作しているホスト名  
`base_eXist_port` eXist に接続するためのポート番号  
`base_db_uri` eXist 中にある問題 DB の URI

出力

該当する問題記述 (REXML::Element オブジェクト)

- `get_itemId`

概要 あるテストの出題時に、問題グループ中のまだ出題されていない問題候補の ID を返す (ランダムで 1 つ)。

入力

`group_id` 問題グループの ID  
`item_type` 問題形式  
`itemList` 既に出題されている問題候補の ID  
`mode` ランダム出題か問題形式の指定か  
`base_eXist_host` eXist が動作しているホスト名  
`base_eXist_port` eXist に接続するためのポート番号  
`base_db_uri` eXist 中にある問題 DB の URI

出力

出題されていない問題候補があれば、その ID。無ければ-1 が返る。

- `get_testId`

概要 現在出題中のテストの固有識別子を返す。

入力

無し

出力

テストの固有識別子

- `convInputType`

概要 問題記述中の問題形式から解答形式 (input 要素の type 属性値) を決定。make\_xml メソッドから呼ばれる。

入力

`type` 問題形式

`base_eXist_host` eXist が動作しているホスト名

`base_eXist_port` eXist に接続するためのポート番号

`base_inputType_uri` 問題形式から xhtml の input 要素の type 属性値を決めるための変換テーブル

出力

`base_inputType_uri` で指定された変換テーブルに記述された type 属性値。

- `randomize`

概要 出題テーブル中の要素をシャッフル

入力

`tbl` make\_table メソッドで作成した出題テーブル (ハッシュのリスト)

出力

シャッフルされた出題テーブル

## 4.3 評価モジュール

評価に関する機能を持つモジュールは `evaluate.rb` である。このモジュールはインタフェース兼主処理を行う `adel_exam.cgi` から呼び出され、使用される。以下に、評価モジュールが持つメソッドの外部仕様を示す。

#### 4.3.1 各メソッドの外部仕様

- initialize

概要 Evaluate クラスのインスタンス生成時に実行される。

入力

無し

出力

無し

- preEvaluate

概要 選んだ選択肢のプレ評価（事前評価）を行う。実際の評価は問題形式ごとに用意した評価メソッドを用いる。

入力

type 問題形式

ques\_pkey 出題された問題に付与された固有識別子

value xhtml 中の input 要素（選んだ選択肢）が持つ value 属性値

setHisHash 固有識別子に ques\_pkey を持つ問題の出題履歴

base\_eXist\_host eXist が動作しているホスト名

base\_eXist\_port eXist に接続するためのポート番号

base\_db\_uri 問題 DB の URI

出力

評価結果を返す。サポートしていない形式だった場合は-1 を返す。

- evalRadioType

概要 単一選択問題の評価を行う。preEvaluate メソッドから呼ばれる。

入力

ques\_pkey 出題された問題に付与された固有識別子

value xhtml 中の input 要素（選んだ選択肢）が持つ value 属性値

setHisHash 固有識別子に ques\_pkey を持つ問題の出題履歴

base\_eXist\_host eXist が動作しているホスト名

base\_eXist\_port eXist に接続するためのポート番号

base\_db\_uri 問題 DB の URI

出力

評価結果を格納したハッシュ。

例)

```
{ "chk_selection" => 選んだ選択肢, "eval_result" => 取得したポイント "crct_weight" => 正解の重みの合計値, "incrct_weight" => 不正解の重みの合計値, "total_weight" => 重みの合計値, "eval_pkey" => 問題の固有識別子, "time" => 評価した時間, "total_point" => 正解時に与えられるポイント }
```

- evaluate

概要 評価結果の正規化

入力

tbl あるテストの評価結果の履歴（ハッシュのリスト）

出力

正規化された評価結果。問題グループと点数をコロン、この組をカンマで区切る文字列を返す。

例) group1:10,group2:50

## 5 これから改善すべき点

初期バージョンを作った時点で発覚した問題点に対する改善点など。優先順位が高いと思われる順に、現在見つかった問題点と簡単な解決方法の道筋について述べる。

### 形跡的評価への対応

現在のバージョンでは、ADEL からのテストの成績の問合せに対して結果を返すと言う形を取っている。この方法では、テストの問題 1 問に解答する度に ECA ルールが発火し、テストの途中で復習に入ると言った学習の制御が難しい。そこで、問題に解答する度にテスト機構が ADEL の学習履歴情報に対して直接更新を行う。これにより、問題に解答する度に ECA ルールが発火し、形式的評価への対応が可能となる。

### 処理順の制御

問題ごとの評価の流れは、選択肢を選んだ時にプレ評価を行い、次に解答ボタンによる本評価を行うと言った 2 段階の評価を行う。このプレ評価が終了する前（DB に結果が格納される前）に本評価を行うと正しい評価を行う事が出来ない。そこで、プレ評価中は解答ボタンを無効化する、評価処理のジョブキューを作成するなどの処置が必要となる。

### 履歴 DB のスキーマの改善

評価、出題テーブルの各主キーと、両テーブルを関連付けている外部キーに MD5 によるハッシュ値を用いている。これらを、テーブルに記録した順に値を割り振る「serial」型にするのが分かりやすい。

### 扱う問題形式の拡張

今現在出題できる問題形式が単一選択のみであるので、最低でも複数選択、出来れば記述式など出題の幅を広げる。

### 実装形態の変更

現時点で、テスト機構は Ruby による CGI を用いて実装されている。一方 ADEL は Ruby on Rails を用いて実装されており、これら 2 つのシステム間での連携に問題がある。例として、動作環境の構築に fast\_cgi などの導入と言った手間がかかり、実行速度にも影響がある。そこで、Ruby の require 文で利用するモジュール化や Ruby on Rails を用いた実装を行うなど、実装形態の変更を検討する。

## 6 更新履歴

システムの改良や実装などの簡単な更新履歴。

### 2007-12-11

初期バージョンには無かった、出題・評価結果の履歴を記録する履歴 DB をシステムに追加。  
履歴 DB の追加に伴い、複数のユーザとテストに対応。

### 2007-11-27

ADEL の外部公開に合わせて新しくテスト機構を実装。仕様は DBWS2006 ( 2006, 西 ) にほぼ準拠。

### 2005-12-x

卒業論文 ( 2005, 西 ) の一部としてプロトタイプを実装するも、仕様変更からお蔵入り。