

= 概説 Ruby Refactoring Browser

== リファクタリングについて
参考資料は

マーチン・ファウラー著「リファクタリング」
ピアソン・エデュケーション。
リファクタリングについての良い本です。ぜひ読みましょう。

* 名詞としての「リファクタリング」

外部から見たときのふるまいを
保ちつつ、理解や修正が簡単に
なるように、ソフトウェアの
内部構造を変化させること。

* 動詞としての「リファクタリングする」

一連のリファクタリングを
行なって、外部からの見た目の
変更なしに、ソフトウェアを
再構築すること。

適切なリファクタリングによって以下のような利益が得られる

- * 設計の向上
- * 理解を容易に
- * バグを発見しやすく

小さな定型的リファクタリングを繰り返すことで、大きな変更をする。
これによって動作の変更が起きないようにできる。

例えば、

- * 変数名/メソッド名の変更
- * メソッドの切り出し
- * メソッドの移動

などが挙げられる。それ以外にもファウラーの著書には
様々なリファクタリングのリストが載っている。

動作の変更が無いこと、デグレートしていないことを保証するため
にユニットテストが重要になる。

== RubyRefactoringBrowserの概要
変数名/メソッド名の変更のような小さな定型的リファクタリング
を自動化するツール。

Rubyのソースコードのパーズおよび解析を行なって指定された変更を
施すツール。

実際にやっていることは賢い(インテリジェントな)置換、コピー&ペースト。

以下のリファクタリングが利用できる。

- * ローカル変数名の変更
- * グローバル変数名の変更
- * インスタンス変数名の変更
- * クラス変数名の変更
- * メソッド名の変更
- * コード片をメソッドとして
切り出し
- * メソッドの親子間での移動
- * 共通スーパークラスの作成

そして以下のエディタ/IDEから利用可能

- * Emacs

一番積極的に開発保守されている。

全ての機能が利用可能

* Vim

RRBのアーカイブに含まれている。vimのrubyインターフェースを利用。
変数名の変更/メソッドの抽出のみ利用可能。

* コマンドライン

全ての機能が利用可能。Vimのインターフェースからできない
リファクタリングはこっちからする。

* xzyzy

((<URL:http://mzp.breeze.jp/>)) で配布。

* freeride

freeride本体に付属しているはず。
詳しくは((<URL:http://freeride.rubyforge.org/>))を参照。

また、以下のような点に注意。

* Emacsでは、そのプロセスに読みこまれている Ruby スクリプト全てを
リファクタリングの対象とする。

* リファクタリングの対象となるスクリプトは、その内容を
「定義部」と「実行部」に分け、「実行部」を
if __FILE__ == \$0 ... end で囲っておく必要がある。

* Vimでは、カレントバッファの Ruby スクリプトと同じディレクトリにある
*.rb なファイルをリファクタリングの対象とする。

* attr_{reader,writer,accessor} は無視される。

* その他 eval など特殊な機能を使っていると正しくない変更がされる可能性が
ある。

((<るびまの記事|URL:http://jp.rubyist.net/magazine/?0010-RubyRefactoringBrowser>))
にもEmacsでのチュートリアルが書かれているのでこちらも見てください。

== Ruby Refactoring Browserの組みこみ

((<るびまの記事|URL:http://jp.rubyist.net/magazine/?0011-RubyRefactoringBrowser>))
で詳細に解説してあります。

Ruby Refactoring Browserは大部分がRubyで書かれているため、
なんらかの方法でRubyのインタプリタを動かさなければならない。
そこで、例えばパイプのようなプロセス間通信を使うなどする必要がある。

また、ふつうのエディタにはIDEでいう「プロジェクト」の概念がないため、
どのスクリプトからどのスクリプトまでがリファクタリングの対象になるか考
えなければならない。例えばEmacsであれば全部読みこんでいるだろうとか、
vimであればそんなことはしないだろうとかである。また「プロジェクト」
のような概念を導入してユーザに指定させるのも一つの方法だろう。

== 今後の課題

ドキュメントの拡充が一番の問題である。まあぼちぼちやっていくつもりである。
協力者募集中。

あとはインターフェースの数を増やし、洗練させること。

== URLS

* ((<URL:http://www.kmc.gr.jp/proj/rrb/>))

RubyRefactoringBrowserのページ。
RubyRefactoringBrowserの配布をしている他、
ドキュメント、メーリングリスト、チュートリアルなどを置いている。

* ((<URL:http://rubyforge.org/projects/rrb/>))

RubyForgeのプロジェクトページ。CVSリポジトリ、英語メーリングリスト
BTSなど

* ((<URL:http://www.kmc.gr.jp/~ohai/rrb-kansai.html>))

今日の発表に関するページ。発表資料等はここに置きます。

* ((<URL:http://jp.rubyist.net/magazine/?0010-RubyRefactoringBrowser>))

* ((<URL:http://jp.rubyist.net/magazine/?0011-RubyRefactoringBrowser>))

RubyRefactoringBrowserに関するるびまの記事。
前者がEmacsでのチュートリアルで、後者が組み込みに関するもの。

* ((<URL:http://mzp.breeze.jp/>))

xyzyy用のインターフェース