Programowanie w języku Erlang - Rebar

Andrzej Romański
Katarzyna Halska

Plan prezentacji

- Podstawowe informacje
- Historia
- Zalety
- Sposób instalacji
- Możliwe zastosowania

Co to jest?

Rebar to narzędzie Erlanga ułatwiające kompilację i testowanie aplikacji Erlanga, sterowników oraz nowych wersji oprogramowania (release).

Dostęp

Główna strona projektu https://github.com/rebar/rebar

Opis możliwych funkcji https://github.com/rebar/rebar/wiki

Licencja

Erlang Public License (EPL) wersja 1.1

Historia

Pierwszy wersja została udostępniona w grudniu 2009 roku pod nazwą release 1. Obecne oficjalne wydanie jest z 25 listopada 2013 roku(2.1.0-pre2), jednakże aplikacja jest nieustannie usprawniana poprzez nowe aktualizacje.

Zalety rebara

- Gdzie tylko możliwe rebar używa standardowej konwencji Erlang/OTP co minimalizuje ilość potrzebnej pracy podczas konfiguracji.

- Zapewnia wszechstronne zarządzanie zależnościami umożliwiając twórcom aplikacji łatwe ponowne wykorzystanie wspólnych bibliotek z różnych źródeł (git, Hg, etc)

- prosty w obsłudze, automatycznie generuje potrzebne pliki

- Jako samodzielny skrypt Erlanga (escript) jest łatwy w rozpowszechnianiu oraz umożliwia umieszczenie go bezpośrednio w projekcie

Instalacja

Dla systemów Linux, wymagany zainstalowany git

```
$ git clone git://github.com/rebar/rebar.git
$ cd rebar
$ ./bootstrap
Recompile: src/getopt
...
Recompile: src/rebar_utils
==> rebar (compile)
Congratulations! You now have a self-contained script called "rebar" in
your current working directory. Place this script anywhere in your path
and you can use rebar to build OTP-compliant apps.
```

Instalacja

Dla systemów Windows podobnie, także wymagany jest zainstalowany git oraz w zmiennej PATH ustawiona ścieżka do działającej wersji Erlanga.

```
C:\> git clone git://github.com/rebar/rebar.git
C:\> cd rebar
C:\Rebar> bootstrap.bat
Recompile: src/rebar
==> Rebar (compile)
==> Rebar (escriptize)
Congratulations! You now have a self-contained script called "rebar" in
your current working directory. Place this script anywhere in your path
and you can use rebar to build OTP-compliant apps.
```

Tworzenie projektu

```
$ rebar create-app appid=myapp
⇒ directory (create-app)
Writing src/myapp.app.src
Wriring src/myapp_app.erl
Writing src/myapp_sup.erl
```

Używając systemowego wzorca rebara tworzymy szkielet aplikacji, rezultatem jest pojedynczy podkatalog src który zawiera trzy pliki:

- Myapp.app.src – aplikacja OTP
- Myapp_app.erl - implementacja wzorca OTP
- Myapp_sup.erl - implementacja Supervisor behaviour

Kompilacja

```
$ rebar compile
⇒ directory (compile)
Compiled src/myapp_app.erl
Comppiled src/myapp_sup.erl
```

Pojawia się nowy katalog ebin zawierający plik .beam związany z plikiem źródłowym Erlanga w katalogu src.

Jest tez plik ebin/myapp.app. Rebar dynamicznie tworzy odpowiednią aplikację OTP używając src/myapp.app.src jako wzorca, dodając informację o wszystkich kompilowanych modułach aplikacji w sekcji modules pliku myapp.app.

`Rebar clean` – czyści projekt po komilacji

Generowanie plików

Aby wygenerować release potrzebujemy 3 pliki:

1. reltool.config
2. rebar.config
3. “nazwa_aplikacji”.app.src

Na podstawie ich generowana jest wersja ze wszystkimi wyszczególnionymi zależnościami.

Wykrzystywanie

Główną funckjonalnością rebara jest możliwość zarządzania zależnościami aplikacji.
Na podstawie pliku reltool.config generuje on zależności i linkuje potrzebne moduły.

Wykorzystanie

- Uruchamianie unit testów poprzez wykorzystanie modułu Eunit

```
rebar eunit app=app_name
```

- Generowanie dokumencji przy pomocy Edoc

```
rebar doc app=app_name
```

Dziękujemy za uwagę