Programowanie w Ruby

Wykład 9

Marcin Młotkowski

5 stycznia 2017

Plan wykładu

Środowisko okienkowe Tcl/Tk

- Tcl: Tool Command Language
- Tk: Tk Toolkit, przenośna biblioteka graficzna

Przykład

Kalkulator graficzny

- Okienko do wprowadzania danych
- Przycisk <Oblicz>
- Miejsce na wynik
- Przycisk <Koniec>

Implementacja

```
require 'tk'

class Okienko
    def run
        @win = TkRoot.new { title 'Kalkulator' }
```

Implementacja

```
require 'tk'

class Okienko
    def run
        @win = TkRoot.new { title 'Kalkulator' }
```

```
        @entry = TkEntry.new(@win) { pack }
        @butt = TkButton.new(@win) { text 'Oblicz'; pack }
        @butt.command { self.oblicz }
        @label = TkLabel.new(@win) { text '0'; pack }
        TkButton.new(@win) { text 'KONIEC';
                             command { exit }; pack }
```

Implementacja

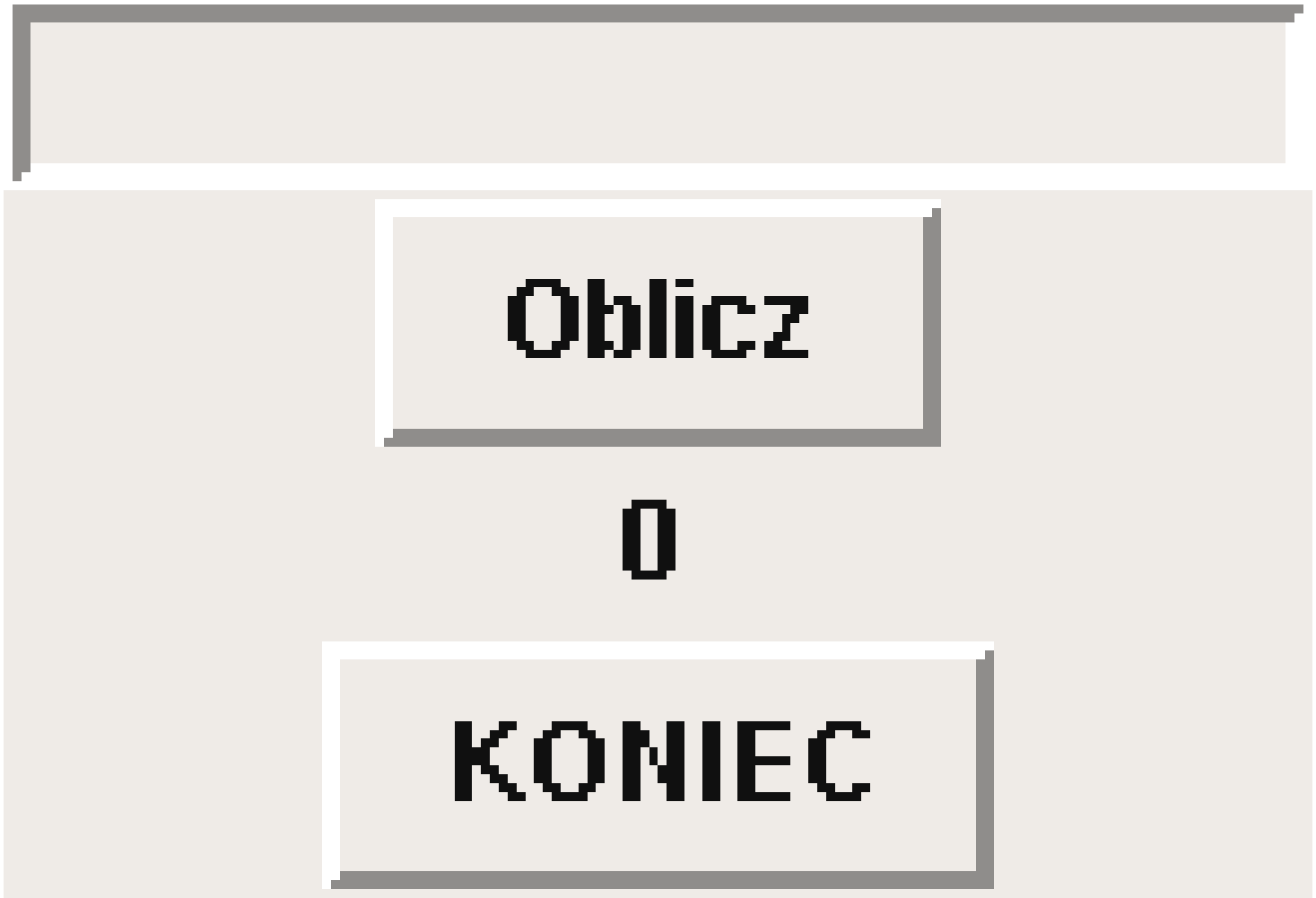
```
require 'tk'

class Okienko
    def run
        @win = TkRoot.new { title 'Kalkulator' }

        @entry = TkEntry.new(@win) { pack }
        @butt = TkButton.new(@win) { text 'Oblicz'; pack }
        @butt.command { self.oblicz }
        @label = TkLabel.new(@win) { text '0'; pack }
        TkButton.new(@win) { text 'KONIEC';
                             command { exit }; pack }

        Tk.mainloop

    end
```

Implementacja obsługi zdarzeń

butt.command { oblicz }

```
def oblicz
    @label.text = eval(@entry.value)
    @entry.value = ''
end
```

Wynik



Opis obrazka

Układanie kontrolek

Model grid

- Okno jest dzielone na komórki ułożone w wiersze i kolumny
- Kontrolki są wstawiane w komórki
- kontrolki mogą zajmowć więcej niż jedną komórkę

Układanie kontrolek w komórkach siatki

```
@entry = TkEntry.new(@win)
          { grid(:row => 1, :column => 1) }
@butt = TkButton.new(@root)
          { grid(:row => 1, :column => 2) }
@label = TkLabel.new(@root)
          { grid('row' => 1, 'column' => 3) }
TkButton.new(@root) { grid(:row => 2, :column => 2) }
```

Rezultat

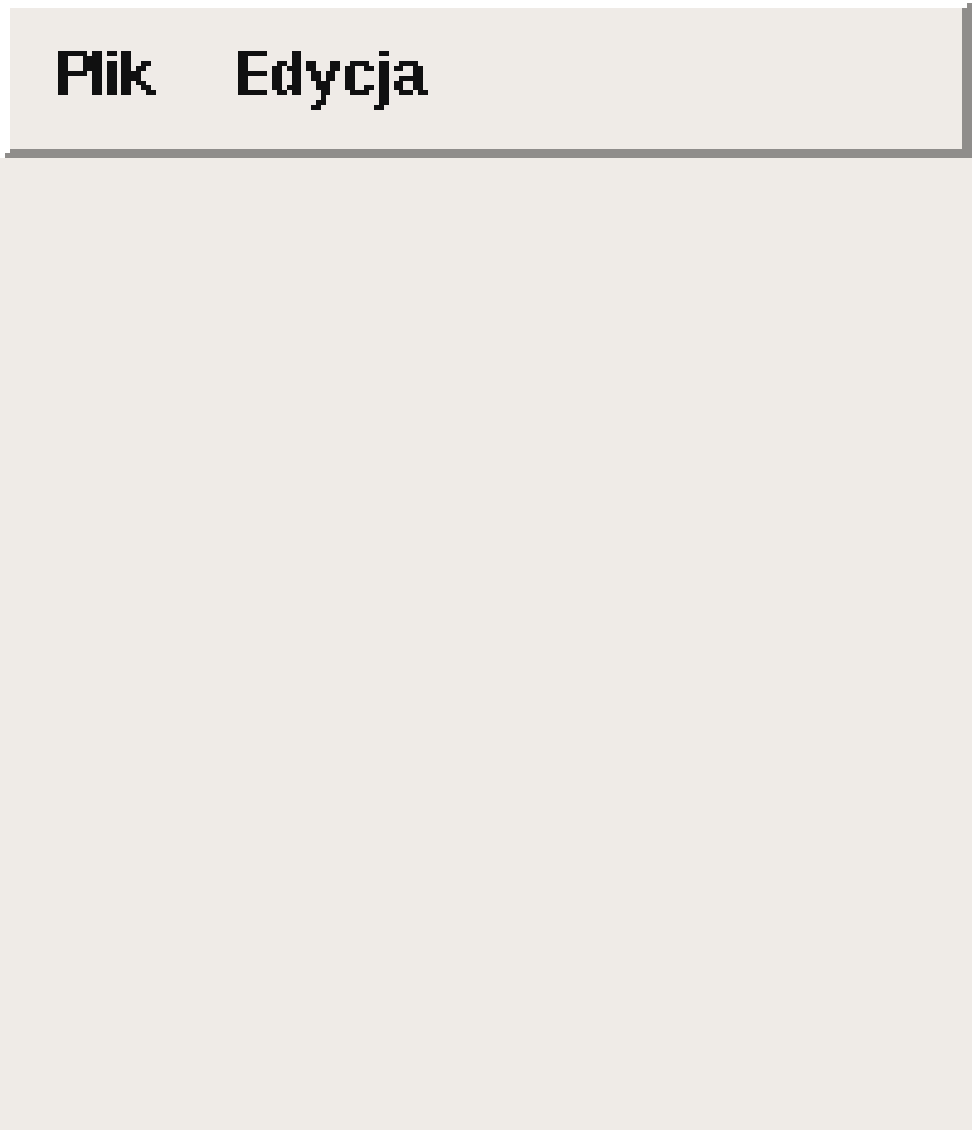
Pierwsze menu

```
menu = TkMenu.new()
one = TkMenu.new(menu)
menu.add('cascade', :menu => one, :label => 'Plik')
one.add('command', :label => 'Zapisz',
            :command => proc{ puts 'Zapisane'})
    one.add('command', :label => 'KONIEC',
            :command => proc{ root.destroy })
```

Drugie menu

```
two = TkMenu.new(menu)
menu.add('cascade', :menu => two, :label => 'Edycja')
two.add('command', :label => "Skopiuj", :command => proc { })
root.menu(menu)
```

Wynik



Kontrolka TkCanvas

```
cv = TkCanvas.new(root)
```

Kontrolka TkCanvas

```
cv = TkCanvas.new(root)
```

Ustalenie rozmiarów

```
cv.place(:height => 200, :width => 200)
```

Rysowanie na płótnie

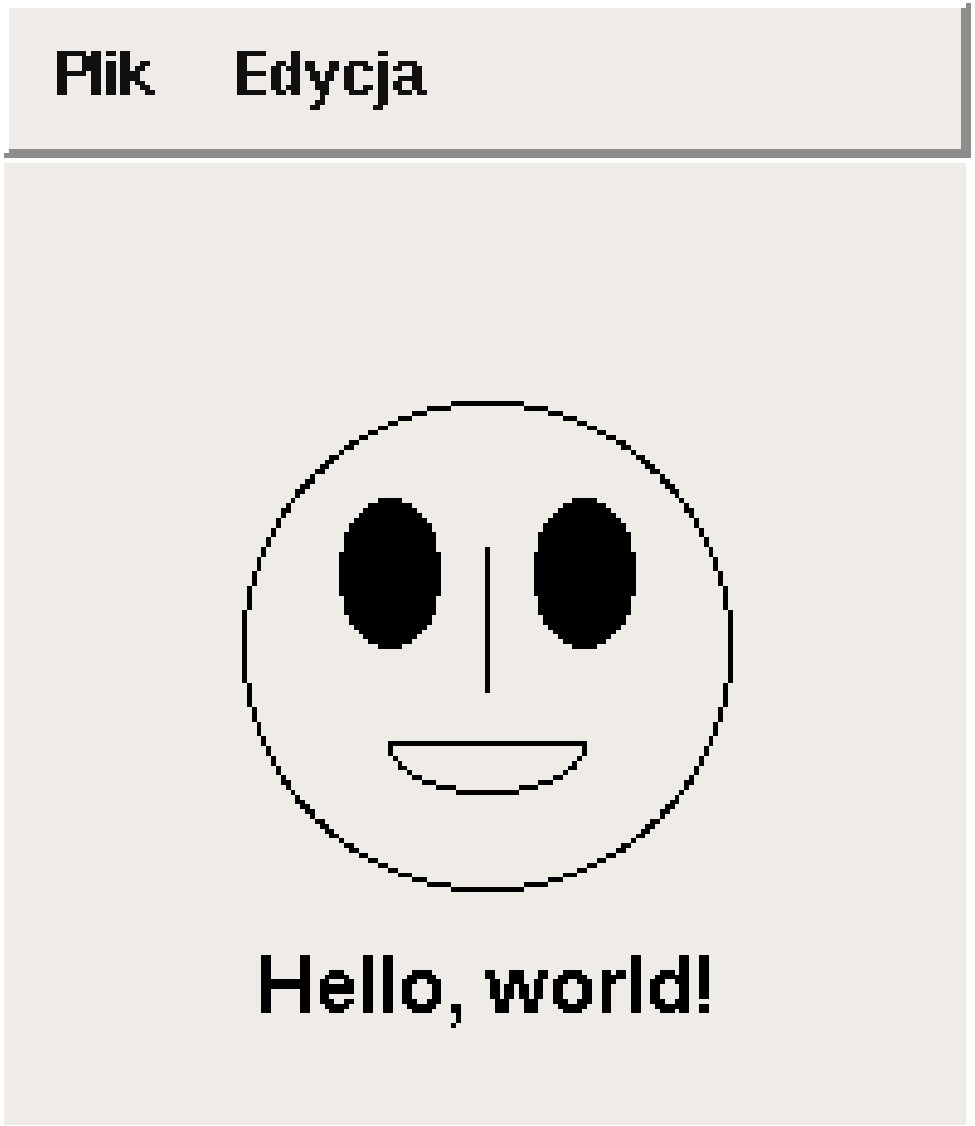
Na płótnie umieszcza się elementy Tkc*, na przykład TkcArc, TkcLine czy TkcBitmap.

Przykład

```
cv.create(TkcOval, 50, 50, 150, 150, 'width' => 1)
cv.create(TkcLine, 100, 80, 100, 110)
cv.create(TkcArc, 80, 110, 120, 130,
          :start => 180, :extent => 180)
cv.create(TkcOval, 70, 70, 90, 100, :fill => 'black')
cv.create(TkcOval, 110, 70, 130, 100, :fill => 'black')
TkcText.new(cv, 100, 170, :font => 'Arial 12 bold',
          :text => 'Hello, world!', :anchor => 'center')
```

Wynik

Wynik



Pliki graficzne

Kontrolki wyświetlające plik

```
img = TkPhotoImage.new(:file => 'plik.gif')
```

Uwagi

- Obrazków nie można umieszczać jako samodzielnych kontrolek
- Obrazy mogą być elementem przycisków lub etykiet

Uwagi

- Obrazków nie można umieszczać jako samodzielnych kontrolek
- Obrazy mogą być elementem przycisków lub etykiet

Przykłady

```
TkLabel.new(win) { image img; pack }

TkButton.new(win) { image img; pack }

TkcImage.new(cv, 100, 100, 'image' => img)
```

Plan wykładu

Wątki

- Zadania permanentnie wykonywane podczas działania programu
- Zadania wykonywane na żądanie w niezależnym wątku

Przykład

Zegar

```
@clock = TkLabel.new(@win) { text '' ''; pack }
@timer = TkAfter.new(1, -1,
        proc{ @clock.text = Time.now.to_s })
@timer.start
```

Zadania w wątkach

```
menu.add('command',
        :label => 'Wykonaj',
        :command => proc{ Thread.new { self.run } })
```

Plan wykładu

Ruby on Rails (RoR)

Co to jest

Środowisko do szybkiego tworzenia aplikacji webowych

Ruby on Rails – nieco historii

- Stworzony przez Davida Heinemeiera Hanssona

Ruby on Rails – nieco historii

- Stworzony przez Davida Heinemeiera Hanssona
- DRY – Don't Repeat Yourself

Ruby on Rails – nieco historii

- Stworzony przez Davida Heinemeiera Hanssona
- DRY – Don't Repeat Yourself
- Mnóstwo gotowców (Convention Over Configuration) i skryptów generujących szablony

Ruby on Rails – nieco historii

- Stworzony przez Davida Heinemeiera Hanssona
- DRY – Don't Repeat Yourself
- Mnóstwo gotowców (Convention Over Configuration) i skryptów generujących szablony
- MVC – Model-View-Controller

Podstawowe moduły RoR

ActionPack

odpowiada za przesyłanie żądań (tj. URLi) do odpowiednich modułów oraz odbiera i wyświetla wyniki działań

Podstawowe moduły RoR

ActionPack

odpowiada za przesyłanie żądań (tj. URLi) do odpowiednich modułów oraz odbiera i wyświetla wyniki działań

ActiveRecord

definiuje ORM (object-relational mapping)

Podstawowe moduły RoR

ActionPack

odpowiada za przesyłanie żądań (tj. URLi) do odpowiednich modułów oraz odbiera i wyświetla wyniki działań

ActiveRecord

definiuje ORM (object-relational mapping)

ActiveSupport

różne dodatki i rozszerzenia

Podstawowe moduły RoR

ActionPack

odpowiada za przesyłanie żądań (tj. URLi) do odpowiednich modułów oraz odbiera i wyświetla wyniki działań

ActiveRecord

definiuje ORM (object-relational mapping)

ActiveSupport

różne dodatki i rozszerzenia

ActiveMailer

wysyłanie maili

Scenariusz

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#prezenty

Scenariusz

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#prezenty

http://mikolaj.pl/store/do_koszyka/konsola
StoreController#do_koszyka(konsola)

Scenariusz

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#prezenty

http://mikolaj.pl/store/do_koszyka/konsola
StoreController#do_koszyka(konsola)

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#przezenty

Scenariusz

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#prezenty

http://mikolaj.pl/store/do_koszyka/konsola
StoreController#do_koszyka(konsola)

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#przezenty

http://mikolaj.pl/store/do_koszyka/dvd
StoreController#do_koszyka(dvd)

Scenariusz

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#prezenty

http://mikolaj.pl/store/do_koszyka/konsola
StoreController#do_koszyka(konsola)

http://mikolaj.pl/przegladanie/prezenty
PrzegladanieController#przezenty

http://mikolaj.pl/store/do_koszyka/dvd
StoreController#do_koszyka(dvd)

http://mikolaj.pl/platnosci/przelew
PlatnosciController#przelew

Konwencje

http://mikolaj.pl/store/do_koszyka/dvd:

- w pliku store_controller.rb

Konwencje

http://mikolaj.pl/store/do_koszyka/dvd:

- w pliku store_controller.rb
- jest klasa StoreController

Konwencje

http://mikolaj.pl/store/do_koszyka/dvd:

- w pliku store_controller.rb
- jest klasa StoreController
- z metodą do_koszyka

Konwencje

http://mikolaj.pl/store/do_koszyka/dvd:

- w pliku store_controller.rb
- jest klasa StoreController
- z metodą do_koszyka
- Jeszcze jest widok do_koszyka.rhtml

Plan wykładu

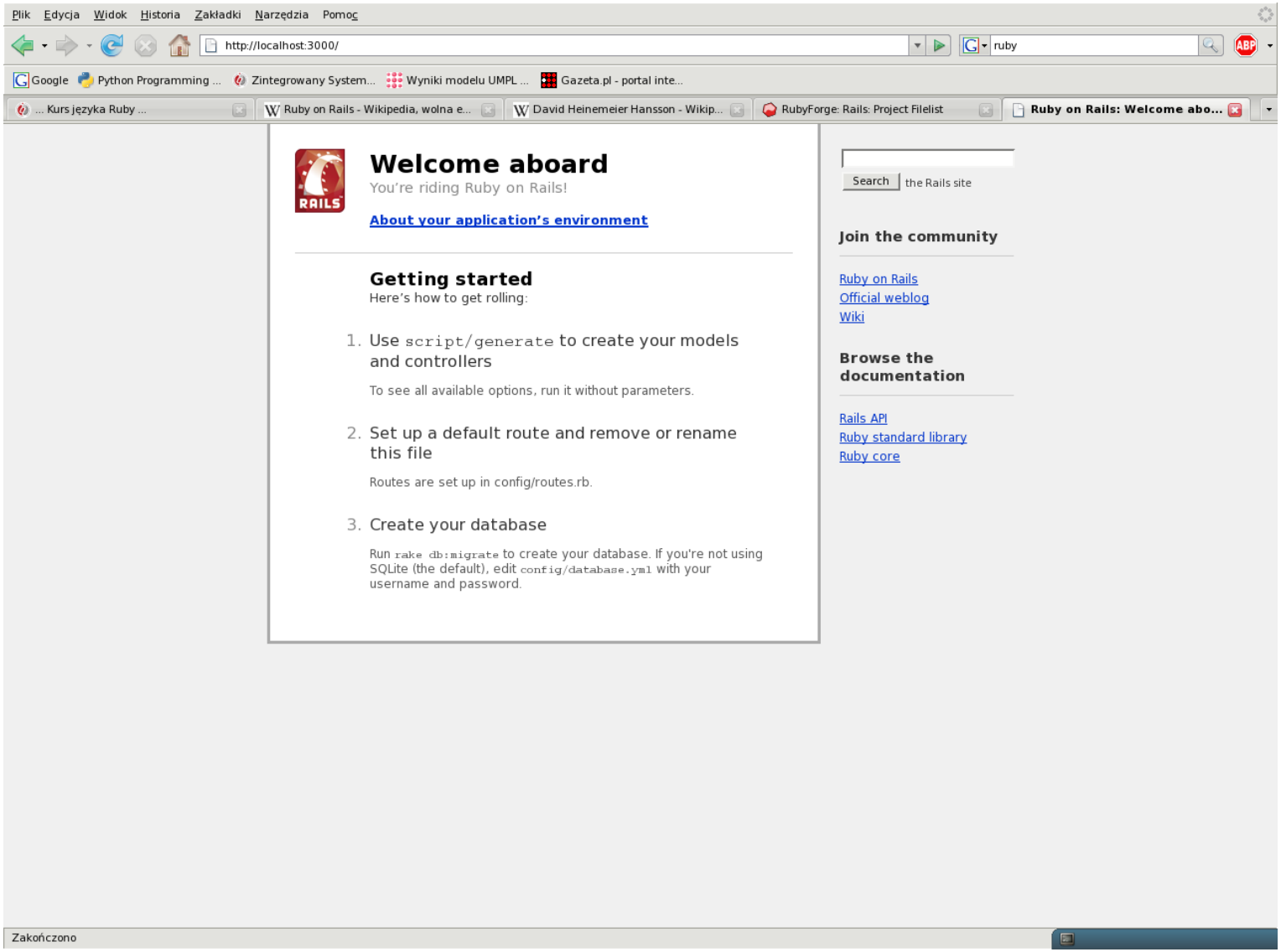
Pierwsza aplikacja

$ rails demo

```
create
create app/controllers
create app/helpers
create app/models
create app/views/layouts
create config/environments
...
```

Uruchomienie aplikacji

```
$ ruby demo/script/server
=> Booting WEBrick...
=> Rails 2.2.2 application started on http://0.0.0.0:3000
=> Ctrl-C to shutdown server; call with --help for options
```


Plik Edycja Widok Historia Zakładki Narzędzia Pomoc
http://localhost:3000/
ruby
Google
Python Programming ...
Zintegrowany System...
Wyniki modelu UMPL ...
Gazeta.pl - portal inte...
... Kurs języka Ruby ...
Ruby on Rails - Wikipedia, wolna e...
David Heinemeier Hansson - Wikip...
RubyForge: Rails: Project Filelist
Ruby on Rails: Welcome abo...
Welcome aboard
You're riding Ruby on Rails!
About your application's environment
Getting started
Here's how to get rolling:
1. Use script/generate to create your models and controllers
To see all available options, run it without parameters.
2. Set up a default route and remove or rename this file
Routes are set up in config/routes.rb.
3. Create your database
Run rake db:migrate to create your database. If you're not using SQLite (the default), edit config/database.yml with your username and password.
Search
the Rails site
Join the community
Ruby on Rails
Official weblog
Wiki
Browse the documentation
Rails API
Ruby standard library
Ruby core
Zakończono

Zbudowanie kontrolera

$ ruby script/generate controller Hello

```
create app/views/hello
create app/controllers/hello_controller.rb
create test/functional/hello_controller_test.rb
create app/helpers/hello_helper.rb
```

Zbudowanie kontrolera

$ ruby script/generate controller Hello

```
create app/views/hello
create app/controllers/hello_controller.rb
create test/functional/hello_controller_test.rb
create app/helpers/hello_helper.rb
```

Konwencja

Kontrolery są w katalogu app/controllers.

Wygenerowane szablony

app/controllers/hello_controller.rb

```
class HelloController < ApplicationController
end
```

Wygenerowane szablony

app/controllers/hello_controller.rb

```
class HelloController < ApplicationController
end
```

Rozszerzenie implementacji

```
class HelloController < ApplicationController
    def world
        # tu możemy sobie coś policzyć
    end
end
```

Widoki

- Stronę html generują tzw. widoki
- Każdej metodzie kontrolera odpowiada plik z widokiem
- Widoki są w katalogu
  app/views/<nazwa klasy>/<metoda>

Rodzaje widoków

- Szablony rhtml: html z osadzonym Rubym
- Szablony rxml: xml z osadzonym Rubym
- Szablony rjs: dynamicznie generowany JavaScript

Budowa widoków

app/views/hello/world.rhtml

- Html
- + wstawki w Rubym
   <%= 2 + 2 %>

Wstawki rhtml

```
<%= 5.times do | licz | %>
<div>Hello world</div>
<%= end %>
```

Wstawki rhtml

```
<%= 5.times do | licz | %>
<div>Hello world</div>
<%= end %>
```

Dodatkowa konwencja

```
<%= 2 + 2 ->
```

Podsumowanie

- Mamy obsługę akcji (aktywowanych odwołaniami url)
- Mamy widoki

Jak połączyć obliczenia z akcji z widokami?

Rozszerzanie funkcjonalności kontrolera

```
class HelloController < ApplicationController
    def world
        @greeting = 'Hello world'
    end
end
```

Widok do kontrolera

hello/world.rhtml

```
<%= @greeting %>
```

lub

```
<%= h(@greeting) %>
```

Widok do kontrolera

hello/world.rhtml

```
<%= @greeting %>
```

lub

```
<%= h(@greeting) %>
```

Funkcja h(string)

Funkcja h przekształca argument na napis z zamienionymi znakami specjalnymi html'a na odpowiedniki, np.
h('<--') = '<--'

Referencje do innych stron

Podejście bezpośrednie:

```
<a href='hello/bye'>żegnaj</a>
```

Po Railsowemu

```
<%= link_to 'żegnaj', :action => 'bye' %>
```

Co trzeba uzupełnić

- Dopisać metodę bye w kontrolerze HelloController
- Dopisać widok bye.rhtml

Domyślna strona kontrolera

Referencji //localhost:3000/kontoler odpowiada

- Za tę stronę odpowiada metoda kontrolera KontrolerController#index
- Oraz widok app/view/kontroler/index.rhtml

Wspólny układ strony

- Można wszystkim widokom z kontrolera przypisać jeden wspólny układ stron
- Dla kontrolera hello trzeba utworzyć szablon
  app/views/layouts/hello.rhtml

Treść szablonu

```
<html><head>
<%= stylesheet_link_tag 'styl', :media => all %>
<title>Szablon</title>
</head><body>
<%= yield :layout %>
</body></html>
```

Treść szablonu

```
<html><head>
<%= stylesheet_link_tag 'styl', :media => all %>
<title>Szablon</title>
</head><body>
<%= yield :layout %>
</body></html>
```

Widoki kontrolera

Widoki kontrolera powinny teraz generować tylko fragmenty html'a.

Układ stron jeszcze raz

```
class HelloController < ApplicationController
    layout 'standard'
    ...
end
```

A gdzie jest strona główna

public/index.html

Domyślny moduł

- W pliku config/routes.rb jest wiersz:
  ```
  # map.root :controller => "welcome"
  ```
- Można wstawić swój kontroler z metodą i widokiem index
- Trzeba usunąć plik public/index.html