МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Кафедра програмних засобів

Звіт

з лабораторної роботи №5

з дисципліни: «Крос-платформне програмування»

за темою: «Використання RubyGems, Fxruby, Ruby on Rails»

Виконав:

ст. гр. КНТ-117 М. І. Басанець

Прийняв:

старший викладач Є. М. Федорченко

2020

1 МЕТА РОБОТИ

Вивчити основні прийоми використання RubyGems. Розробити програми використовуючи FXRuby та Rails.

2 зАВДАННЯ ДО РОБОТИ

1. За допомогою FXRuby створити вікно на якому буде знаходитися кнопка , на якій буде зображені дійсні число, місяць і рік.
2. За допомогою FXRuby створити вікно на якому буде знаходитися кнопка , по натисканні на якій буде з’являтись вікно у якому можна буде задавати різний шрифт, розмір та стиль тексту цієї кнопки.
3. За допомогою FXRuby створити вікно на якому буде знаходитися кнопка по натисканні на котру буде з’являтися діалогове вікно на якому будуть знаходитися два випадаючі списка з підменю та клавіша по натисканні на котру можно буде повернутися до головного вікна.
4. За допомогою FXRuby створити програму у якій можна буде малювати зображення з кривих ліній та стирати вже намальовані зображення.
5. Завантажити та встановити гем Rails.За допомогою цього гему вивести на веб сторінку надпис, у різному розмірі “Hello Rails!!!”
6. За допомогою Ruby on Rails у браузері створити веб додаток обліку ЖКХ(з використанням Actrive Record) у якому можливість додавати записи про данні даних ЖКХ , редактувати вже створені записи , переглядати створені записи, та видаляти записи.

3 Результати виконання завдання

**Завдання 1.**

Код програми:

require 'fox16'

include Fox

time = Time.new()

current\_date = [time.day, time.month, time.year].join(".")

app = FXApp.new("Lab5\_1", "Fox")

mainWindow = FXMainWindow.new(app, "Basanets\_lab5\_task1", nil, nil, DECOR\_ALL)

FXButton.new(mainWindow, current\_date, nil, app, FXApp::ID\_QUIT)

app.create()

mainWindow.show(PLACEMENT\_SCREEN)

app.run()

Результати роботи програми зображено на рисунку 3.1

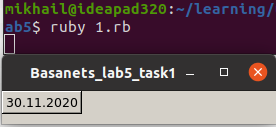


Рисунок 3.1 – Результат виконання завдання 1

**Завдання 2.**

Код програми:

require 'fox16'

include Fox

class FontButtonWindow < FXMainWindow

def initialize(app)

super(app, "Basanets\_lab5\_task2" , :width => 250, :height => 80)

@curr\_app = app

@fontButton = FXButton.new(self, "Select font", :opts => FRAME\_RAISED|FRAME\_THICK|LAYOUT\_CENTER\_X|LAYOUT\_CENTER\_Y)

@fontEditor = FXFontDialog.new(self, "Font...")

@fontButton.connect(SEL\_COMMAND, method(:onFontButtonClicked))

end

def create()

super

show(PLACEMENT\_SCREEN)

end

def onFontButtonClicked(sender, sel, ptr)

if @fontEditor.execute(PLACEMENT\_OWNER) == 1

selectedFont = @fontEditor.fontSelection

newFont = FXFont.new(@curr\_app, selectedFont)

newFont.create()

@fontButton.font = newFont

end

end

end

if \_\_FILE\_\_ == $0

FXApp.new do |app|

FontButtonWindow.new(app)

app.create

app.run

end

end

Результати роботи програми зображено на рисунку 3.2

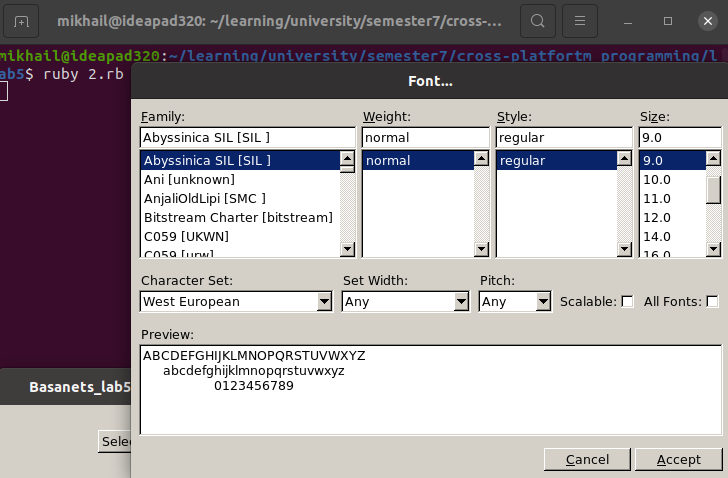


Рисунок 3.2 – Результат виконання завдання 2

**Завдання 3.**

Код програми:

require 'fox16'

include Fox

class Dialog < FXDialogBox

def initialize(owner)

super(owner, "Dialog Box", DECOR\_TITLE|DECOR\_BORDER)

buttonsFrame = FXHorizontalFrame.new(self,

LAYOUT\_SIDE\_BOTTOM|FRAME\_NONE|LAYOUT\_FILL\_X|PACK\_UNIFORM\_WIDTH,

:padLeft => 40, :padRight => 40, :padTop => 20, :padBottom => 20)

# Contents

contents = FXHorizontalFrame.new(self,

LAYOUT\_SIDE\_TOP|FRAME\_NONE|LAYOUT\_FILL\_X|LAYOUT\_FILL\_Y|PACK\_UNIFORM\_WIDTH)

submenu = FXMenuPane.new(self)

FXMenuCommand.new(submenu, "First")

FXMenuCommand.new(submenu, "Second")

FXMenuCommand.new(submenu, "Third")

# Menu

menu = FXMenuPane.new(self)

FXMenuCommand.new(menu, "&Accept", nil, self, ID\_ACCEPT)

FXMenuCommand.new(menu, "&Cancel", nil, self, ID\_CANCEL)

FXMenuCascade.new(menu, "Submenu", nil, submenu)

FXMenuCommand.new(menu, "&Quit\tCtl-Q", nil, getApp(), FXApp::ID\_QUIT)

# Popup menu

pane = FXPopup.new(self)

%w{January February March April May June July August September October November December}.each do |s|

FXOption.new(pane, s, :opts => JUSTIFY\_HZ\_APART|ICON\_AFTER\_TEXT)

end

# Option menu

FXOptionMenu.new(contents, pane, (FRAME\_RAISED|FRAME\_THICK|

JUSTIFY\_HZ\_APART|ICON\_AFTER\_TEXT|LAYOUT\_CENTER\_X|LAYOUT\_CENTER\_Y))

# Button to pop menu

FXMenuButton.new(contents, "&Menu", nil, menu, (MENUBUTTON\_DOWN|

JUSTIFY\_LEFT|LAYOUT\_TOP|FRAME\_RAISED|FRAME\_THICK|ICON\_AFTER\_TEXT|

LAYOUT\_CENTER\_X|LAYOUT\_CENTER\_Y))

# Accept

accept = FXButton.new(buttonsFrame, "&Accept", nil, self, ID\_ACCEPT,

FRAME\_RAISED|FRAME\_THICK|LAYOUT\_RIGHT|LAYOUT\_CENTER\_Y)

# Cancel

FXButton.new(buttonsFrame, "&Cancel", nil, self, ID\_CANCEL,

FRAME\_RAISED|FRAME\_THICK|LAYOUT\_RIGHT|LAYOUT\_CENTER\_Y)

accept.setDefault()

accept.setFocus()

end

end

# Subclassed main window

class DialogTester < FXMainWindow

def initialize(app)

super(app, "Basanets\_lab5\_task3", :opts => DECOR\_ALL, :width => 200, :height => 100)

contents = FXHorizontalFrame.new(self,

LAYOUT\_SIDE\_TOP|FRAME\_NONE|LAYOUT\_FILL\_X|LAYOUT\_FILL\_Y|PACK\_UNIFORM\_WIDTH)

modalButton = FXButton.new(contents,

"&Open Dialog",

:opts => FRAME\_RAISED|FRAME\_THICK|LAYOUT\_CENTER\_X|LAYOUT\_CENTER\_Y)

modalButton.connect(SEL\_COMMAND, method(:onCmdShowDialogModal))

end

def onCmdShowDialogModal(sender, sel, ptr)

Dialog.new(self).execute()

end

def create

super

show(PLACEMENT\_SCREEN)

end

end

if \_\_FILE\_\_ == $0

FXApp.new do |app|

DialogTester.new(app)

app.create()

app.run()

end

end

Результати роботи програми зображено на рисунку 3.3

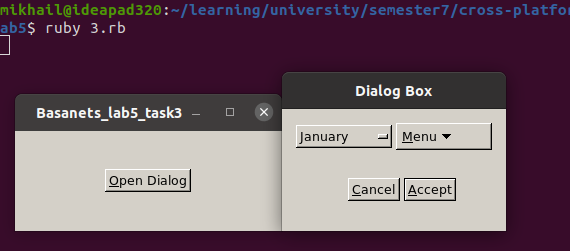


Рисунок 3.3 – Результат виконання завдання 3

**Завдання 4.**

Код програми:

require 'fox16'

include Fox

White = Fox.FXRGB(255, 255, 255)

Black = Fox.FXRGB(0, 0, 0)

class PaintWindow < FXMainWindow

def initialize(app)

super(app, "Basanets\_lab5\_task4" , :width => 500, :height => 500)

@curr\_app = app

self.enable()

vFrame1 = FXVerticalFrame.new(self, :opts => LAYOUT\_FILL)

@canvas = FXCanvas.new(vFrame1, :opts => LAYOUT\_FILL)

@canvas.connect(SEL\_PAINT, method(:onCanvasRepaint))

@canvas.connect(SEL\_LEFTBUTTONPRESS, method(:onCanvasPressed))

@canvas.connect(SEL\_MOTION, method(:onCanvasMove))

@canvas.connect(SEL\_LEFTBUTTONRELEASE, method(:onCanvasReleased))

@clearButton = FXButton.new(vFrame1, "Clear", :opts => FRAME\_RAISED|FRAME\_THICK|LAYOUT\_CENTER\_X)

@clearButton.connect(SEL\_COMMAND, method(:onClearButtonPressed))

@is\_clear\_pressed = false

@is\_pressed = false

@old\_x, @old\_y, @curr\_x, @curr\_y = 0, 0, 0, 0

end

def onCanvasPressed(sender, sel, event)

@old\_x, @old\_y = event.click\_x, event.click\_y

@is\_pressed = true

end

def onCanvasMove(sender, sel, event)

if @is\_pressed

@curr\_x, @curr\_y = event.win\_x, event.win\_y

@canvas.update()

end

end

def onCanvasReleased(sender, sel, event)

@is\_pressed = false

end

def onCanvasRepaint(sender, sel, event)

dcw = FXDCWindow.new(@canvas, event)

dcw.lineWidth = 5

dcw.lineCap = 2

if !event.synthetic? || @is\_clear\_pressed

dcw.foreground = White

dcw.fillRectangle(0, 0, @canvas.width, @canvas.height)

@is\_clear\_pressed = false

return

end

dcw.foreground = Black

dcw.drawLine(@old\_x, @old\_y, @curr\_x, @curr\_y)

dcw.end()

@old\_x, @old\_y = @curr\_x, @curr\_y

end

def onClearButtonPressed(sender, sel, event)

@is\_clear\_pressed = true

@canvas.update()

end

def create()

super()

@canvas.create()

show(PLACEMENT\_SCREEN)

end

end

if \_\_FILE\_\_ == $0

FXApp.new do |app|

wnd = PaintWindow.new(app)

app.create()

app.run()

end

end

Результати роботи програми зображено на рисунку 3.4

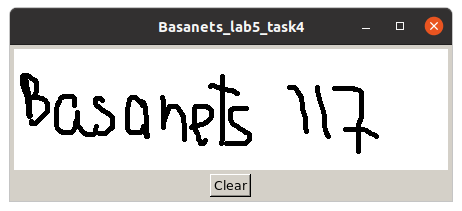


Рисунок 3.4 – Результат виконання завдання 4

**Завдання 5.**

Код програми:

Файл routes.rb:

Rails.application.routes.draw do

get 'hello/index'

root 'hello#index'

#resource :hello

# For details on the DSL available within this file, see https://guides.rubyonrails.org/routing.html

end

Файл hello\_controller.rb:

class HelloController < ApplicationController

def index

end

end

Файл index.html.erb:

<h1>Hello Rails</h1>

<h2>Hello Rails</h2>

<h3>Hello Rails</h3>

Результати роботи програми зображено на рисунку 3.5

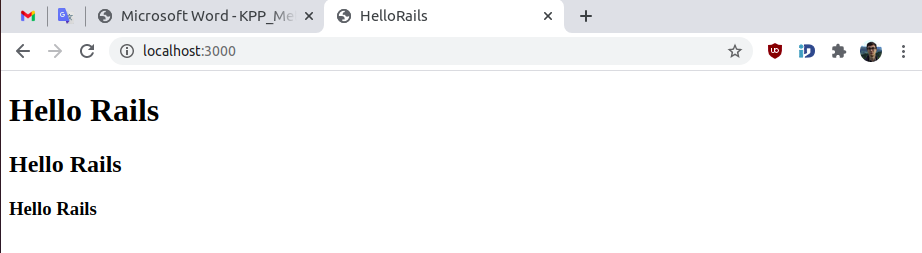


Рисунок 3.5 – Результат виконання завдання 5

**Завдання 6.**

Код програми:

Файл communal\_service\_records\_controller.rb:

class CommunalServiceRecordsController < ApplicationController

before\_action :set\_communal\_service\_record, only: [:show, :edit, :update, :destroy]

# GET /communal\_service\_records

# GET /communal\_service\_records.json

def index

@communal\_service\_records = CommunalServiceRecord.all

end

# GET /communal\_service\_records/1

# GET /communal\_service\_records/1.json

def show

end

# GET /communal\_service\_records/new

def new

@communal\_service\_record = CommunalServiceRecord.new

end

# GET /communal\_service\_records/1/edit

def edit

end

# POST /communal\_service\_records

# POST /communal\_service\_records.json

def create

@communal\_service\_record = CommunalServiceRecord.new(communal\_service\_record\_params)

respond\_to do |format|

if @communal\_service\_record.save

format.html { redirect\_to @communal\_service\_record, notice: 'Communal service record was successfully created.' }

format.json { render :show, status: :created, location: @communal\_service\_record }

else

format.html { render :new }

format.json { render json: @communal\_service\_record.errors, status: :unprocessable\_entity }

end

end

end

# PATCH/PUT /communal\_service\_records/1

# PATCH/PUT /communal\_service\_records/1.json

def update

respond\_to do |format|

if @communal\_service\_record.update(communal\_service\_record\_params)

format.html { redirect\_to @communal\_service\_record, notice: 'Communal service record was successfully updated.' }

format.json { render :show, status: :ok, location: @communal\_service\_record }

else

format.html { render :edit }

format.json { render json: @communal\_service\_record.errors, status: :unprocessable\_entity }

end

end

end

# DELETE /communal\_service\_records/1

# DELETE /communal\_service\_records/1.json

def destroy

@communal\_service\_record.destroy

respond\_to do |format|

format.html { redirect\_to communal\_service\_records\_url, notice: 'Communal service record was successfully destroyed.' }

format.json { head :no\_content }

end

end

private

# Use callbacks to share common setup or constraints between actions.

def set\_communal\_service\_record

@communal\_service\_record = CommunalServiceRecord.find(params[:id])

end

# Only allow a list of trusted parameters through.

def communal\_service\_record\_params

params.require(:communal\_service\_record).permit(:owner\_name, :owner\_last\_name, :service\_type, :city, :street, :house, :apartment)

end

end

Файл communal\_service\_record.rb:

class CommunalServiceRecord < ApplicationRecord

end

Файл index.html.erb (для communal\_service\_records)

<p id="notice"><%= notice %></p>

<h1>Communal Service Records</h1>

<table>

<thead>

<tr>

<th>Owner name</th>

<th>Owner last name</th>

<th>Service type</th>

<th>City</th>

<th>Street</th>

<th>House</th>

<th>Apartment</th>

<th colspan="3"></th>

</tr>

</thead>

<tbody>

<% @communal\_service\_records.each do |communal\_service\_record| %>

<tr>

<td><%= communal\_service\_record.owner\_name %></td>

<td><%= communal\_service\_record.owner\_last\_name %></td>

<td><%= communal\_service\_record.service\_type %></td>

<td><%= communal\_service\_record.city %></td>

<td><%= communal\_service\_record.street %></td>

<td><%= communal\_service\_record.house %></td>

<td><%= communal\_service\_record.apartment %></td>

<td><%= link\_to 'Show', communal\_service\_record %></td>

<td><%= link\_to 'Edit', edit\_communal\_service\_record\_path(communal\_service\_record) %></td>

<td><%= link\_to 'Destroy', communal\_service\_record, method: :delete, data: { confirm: 'Are you sure?' } %></td>

</tr>

<% end %>

</tbody>

</table>

<br>

<%= link\_to 'New Communal Service Record', new\_communal\_service\_record\_path %>

Результати роботи програми (перегляд записів, додавання, видалення, редагування, перегляд конкретного запису) зображено на рисунках 3.6-3.10

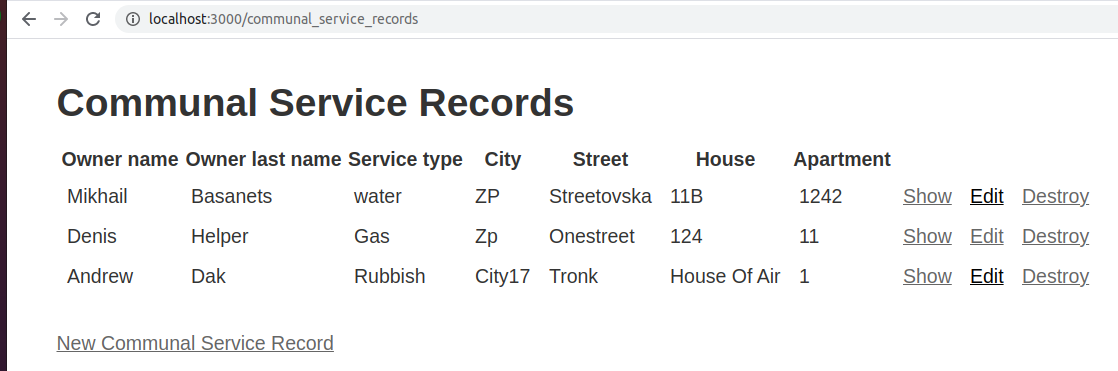


Рисунок 3.6 – Перегляд усіх записів

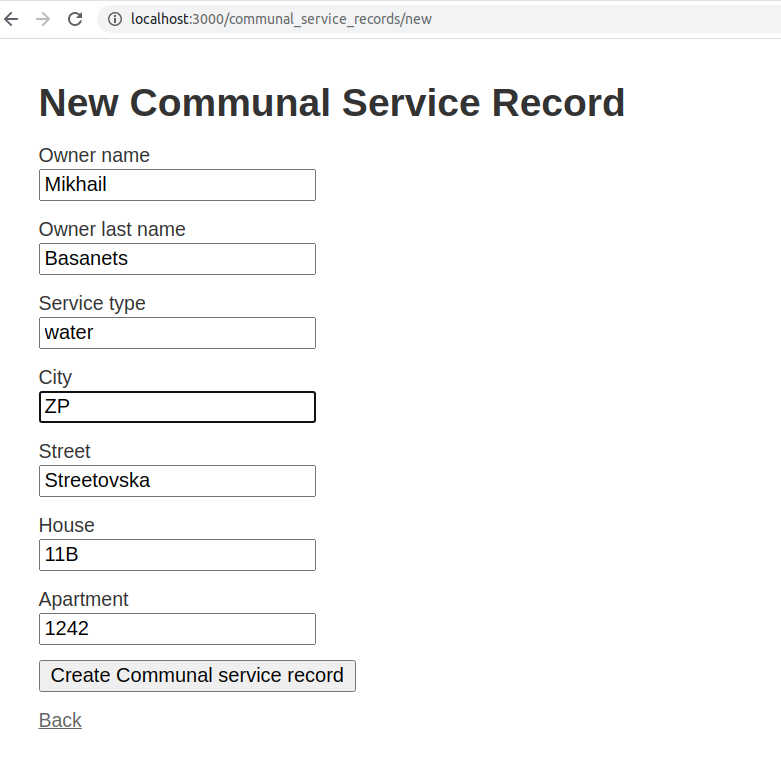


Рисунок 3.7 – Додавання запису

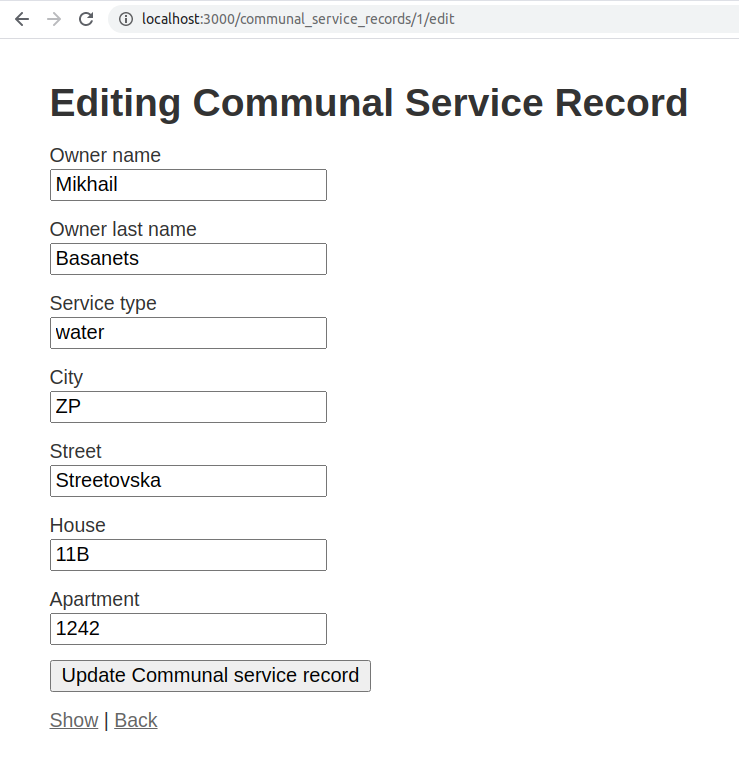


Рисунок 3.8 – Редагування запису



Рисунок 3.9 – Перегляд конкретного запису

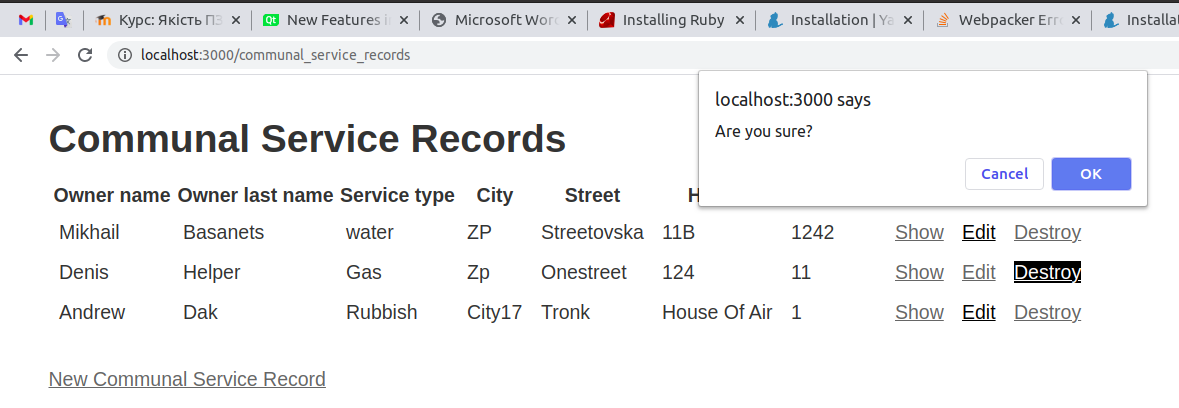


Рисунок 3.10 – Видалення запису

4 висновки

При виконанні даної лабораторної роботи на практиці засвоїв основні принципи та засади роботи з rubygems, fxruby, ruby on rails.

Відповіді на контрольні запитання:

1. Що таке RubyGems?

RubyGems - система керування пакетами (gems) у Ruby.

2. Якою командою можна дивитися версії гемів?

gem list

3. Якою командою можна установити геми?

gem install <required\_gem>

4. Якою командою можна прочитати документацію до гемів?

ri <gem>

5. Як створити кнопку у FXRuby?

@button = FXButton.new(frame, "Clear", :opts => FRAME\_RAISED|FRAME\_THICK|LAYOUT\_CENTER\_X)

Створення кнопки, що знаходиться на формі frame, з назвою Clear та з параметрами FRAME\_RAISED|FRAME\_THICK|LAYOUT\_CENTER\_X.