Open Source Frameworks for Rapid Application Development

Marek Krętowski
Krzysztof Bandurski, Tomasz Łukaszuk, Tomasz Rybak

Software Departament
Faculty of Computer Science
Bialystok University of Technology

m.kretowski@pb.edu.pl
k.bandurski@pb.edu.pl, t.lukaszuk@pb.edu.pl, t.rybak@pb.edu.pl

Lecture topic

Introduction

Introduction: Table of content

Sprawy organizacyjne

Wykłady:

- Ruby on Rails, Django (teoria i praktyka)
- slajdy po angielsku (i polsku)
- udostępniane w cez.wipb.pl
- zaliczenie na 15 wykładzie

Pracownia specjalistyczna:

- wprawki (6 zajęć) + projekt (9 zajęć)
- projekt w grupach 2-4 osobowych
- projekt w wybranym frameworku
- 3 raporty z postępów prac nad projektem
- ocena na podstawie projektu

Lectures

1. Introduction
2. Ruby language introduction
3. Python language introduction
4. Configuration
5. Models, Object-Relational Mapping
6. URL mapping and management
7. Controllers
8. Views
9. Forms
10. Administration
11. Authorisation, sessions
12. Additional modules and plugins
13. Performance, caching, deploying
14. I10n, i18n
15. Testing applications

Current state of network-applications

- Currently many applications are using network
  - Check for updates
  - Load additional data, like cover art
  - Send usage details, bug reports
  - ...
- Most of those are using HTTP
  - Simple protocol
  - Many libraries help using it
  - Many frameworks allow easy creation of server-side application
  - Port 80 is usually not blocked on firewalls

Resource identification

- All resources on the web are referenced using URI
- Transmission of data is done using HTTP
- Those technologies are standardised
- They are result of years of development
- They are described in “Request For Comment” (RFC) documents

URI

- Uniform Resource Identifier (URI), RFC 1630 RFC 2396
- Uniform Resource Locator (URL)
- protocol :// server : port /resource ? data
- variable = value & variable = value
- Valid characters: A-Z, a-z, 0-9, dash, dot, exclamation point, tilde, underscore, parentheses
- Other characters are encoded: %XX, where XX is hexadecimal code of encoded character

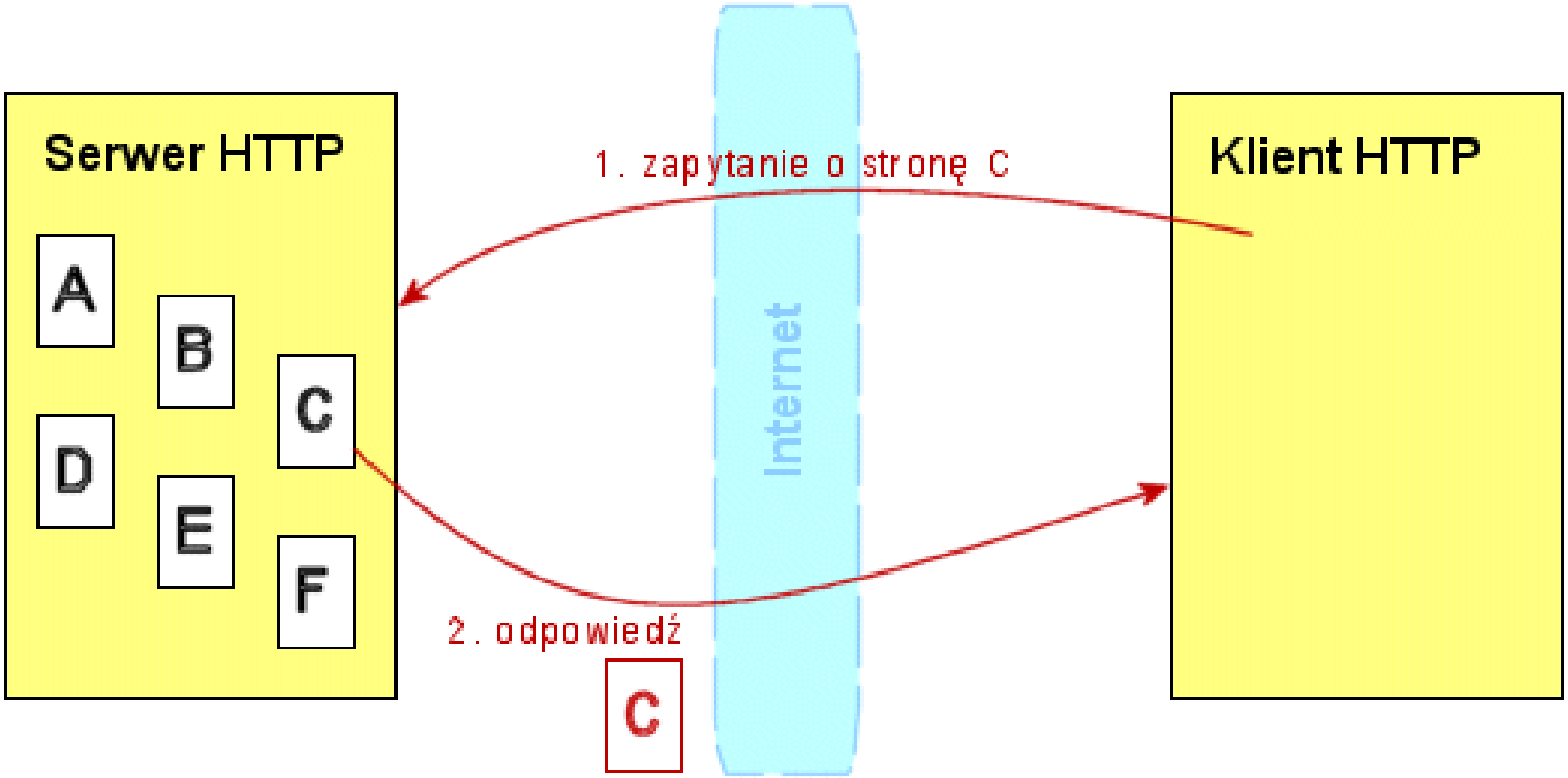
HTTP

- “Hypertext Transfer Protocol — HTTP/1.1”, RFC 2616
- Protocol of application layer from OSI
- Stateless protocol
- Resources are pointer by using URL
- Protocol can be extended and used to transfer other data than HTML files
- Communication request — response
- Server is passive, do not perform any activity except for waiting for client’s requests
- Since protocol version 1.1 connection need not to be closed after transmission but can be reused
- This improves performance

Machines performing HTTP transmission

- Server
- Client (called agent in documentation)
- Proxy servers

Http protocol



Shape of request and response

- Header
- Empty line
- Body

Body need not to always be present. It is not send after response for HEAD request, when response status is 1xx, 204, 304; in those cases response ends with empty line.

Header

- Line of request or status
- Variables are send as: “name: value”
- Each variable is in separate line
- Request: Method URI HTTP/X.x
- Status: HTTP/X.x Code Description

HTTP Methods

GET	receives data pointed by URI
HEAD	just like GET, but does not gets data but only response header
POST	server processes data sent in request's body
PUT	stores data placed in request's body where URI points to
DELETE	removes resource pointed by URI
TRACE	sends back request in response's body with status 200 and Content-Type "message/http"
OPTIONS	returns server's configuration, description, etc.
CONNECT	creates tunnel to the proxy server

Other methods are HTTP extensions

Response codes

1xx request received, processing

2xx success

3xx redirection

4xx failure — agent's fault

5xx failure — server's fault

Details of all codes and their descriptions can be found in the RFC 2616.

HTTP transmission, GET method

```
GET /search?q=php&start=0&ie=utf-8&oe=utf-8&client=firefox
&rls=org.mozilla:en-US:unofficial HTTP/1.0

HTTP/1.0 302 Found
Location: http://www.google.pl/search?q=php&start=0&
ie=utf-8&oe=utf-8&client=firefox&rls=org.mozilla:en-US:unof
Set-Cookie: PREF=ID=abe75ab050fbc4c0:TM=1138805197:LM=11388
:S=e9IJV3s9BKacuAs3;expires=Sun, 17-Jan-2038 19:14:07 GMT;
domain=.google.com
Content-Type: text/html
Server: GWS/2.1
Content-Length: 327
Date: Wed, 01 Feb 2006 14:46:37 GMT
Connection: Keep-Alive
```

HTTP transmission, POST method

```
POST /search HTTP/1.0
Content-Type: application/x-www-form-urlencoded
Content-Length: 80

q=php&start=0&ie=utf-8&oe=utf-8&client=firefox&
rls=org.mozilla:en-US:unofficial

HTTP/1.0 302 Found
```

Growth of rich content on the network

- For many years there was development of technologies used to present pages to the user
- Everything started with the static pages
- But almost immediately people wanted ability to change information presented to the user
- Users also wanted to interact with the web pages
- Many different technologies solve this in different ways

Web framework

- Web framework is responsible for managing web pages
- It helps with managing data storage
- It easies dealing with repetitive tasks
- This allows for programmer to focus on clou of the created project
- It allows for reusing code and web pages
- It allows separating tasks like writing code, pages, content, style
- This easies creating content by non-programmers
- Provides separation of concerns

Static pages

- They are created by human
- Stored as files on the server
- Agent requests page
- Server loads file from disk and sends it
- It is very fast
- Allows easy caching on the server or by proxy servers
- Any changes must be done by human by changing HTML files

SSI

- Server Side Includes
- Small fragments of “code” embedded in HTML
- HTML file must be executable
- Ability to call external program
- Rather limited set of functionality
- Security risks

CGI

- Common Gateway Interface
- Protocol for communicating between HTTP server and external program
- Server sends request from agent to program
- Program sends back response, including headers
- One request — one program is running
- Problematic serving of many pages by one program
- Limited role of server — it just resends response from CGI program

Other technologies

- mod_* in Apache
- PHP
- Java Server Pages (JSP)
- ASP and ASP.NET
- ...

Zope

- Application server written in Python
- Component based
- Programmer writes objects that communicate with other objects
- Cooperation is done by using interfaces
- Components and their connections described in configuration files
- Zope Configuration Merkup Languge (ZCML)
- Hierarchy of components: objects may be placed in other objects
- Different component types: content, adapter, view

Sophistication of web servers

- Big web sites can be sophisticated
  - User management
  - Many pages
  - Templates
  - Similar fragments of the web pages
  - Shared functionality
  - Deploying
  - Configuration management
- This leads to large configuaration files and enormous code bases

Approach used in open-source frameworks

- Cross-platform
- Based on high-level scripting languages
  - allowing for fast testing of new ideas
- Usage of existing standards and libraries
- Convention over configuration
- Nice-looking URLs
- Easy to begin with
- “Easy tasks should be easy and hard should be possible to do” — Perl motto

Features of open-source frameworks

- Open source
- Scripting language allows to see internals of the framework
- But this is not required to do most common tasks
- Possibility to extend application when it becomes popular
- Plug-ins, add-ons
- Ability to extend framework

High-level languages

- Python and Ruby are scripting languages
- They have many built-in types: integer, float, boolean, string
- Built-in more sophisticated types: sets, dictionaries, tuples
- Large libraries providing needed functionalities
- Advanced programming techniques:
  - Iterators
  - Anonymous methods
  - Possibility of dynamic changing of available methods

Hidden parallelism

- Server can serve many requests at the same time
- Programmer need not to worry about parallelism
- Threads/processes are managed by server
- Programmer should not use shared resources from different threads

Duck-typing

- “When it talks like a duck and walks like a duck — it is duck”
- Lack of strict hierarchies of classes
- Object must provide needed methods and attributes
- Type of object is not tested
  - Possible because scripting languages are dynamic
- Eases testing and mocking
- Allows easy “injecting” of objects with additional functionality

Model-View-Controller pattern

- Most current GUI libraries use Model-View-Controller pattern
- Model is responsible for holding data
- View displays data to the user
- Controller is responsible for managing what part of data is displayed and also allows for operating on the data (changing it)
- In Django it looks slightly differently
- Model-Template-View
- Template displays data
- View (methods stored in view.py) are responsible for deciding what data is displayed and how to operate on in

Convention vs configuration

- Modern frameworks are rather sophisticated
  - Database configuration
  - Directories
  - Content management
- This increases complication of configuration files and code that manages all those options
- This also means that all web sites are slightly different
- Increases debugging difficulties
- Django and Ruby on Rails use approach “convention over configuration”
- The most options are already chosen to sensible defaults
- If one wants to change them it is possible, but not recommended
- Convention means that most sites have similar options and they differ in what is important

Main design principles

- Code quality and readability
- Loose coupling
- Don’t Repeat Yourself (DRY) principle
- Failing on errors — never hide error

Django framework

- Django is written in Python
- At the beginning one needs to create system that will be responsible for responding to user's requests
- django-admin.py startproject name
- It will create directory called "name" and few files inside
  - manage.py
  - __init__.py
  - settings.py
  - urls.py

Django history

- Django was started in 2003 in Lawsence Kansas
- It was initially intended to be used to manage web page of the newspaper Lawrence Journal-World
- It was released as Open Source in summer 2005
- The origin from the newspaper makes Django well suited for “content” sites getting information from databases
- It comes from the real world usage

Project and application

- Application is standard Python package
- Projects consist of one or more applications (defined in settings.py)
- Instead of using __init__.py to recognize application, Django uses models.py
- Each applications can therefore has own models

Ruby on Rails history

- Created by David Heinemeier Hansson in 2004
- Was initially used for managing site 37signals
- Was included with Mac Os X Leopard 10.5
- Is using Ruby Gems to manage packages

Django sites

- Main site of the project
  - http://www.djangoproject.com/
- Book describing Django
  - http://www.djangobook.com/en/2.0/
  - Book is also available to buy in the print
- Two versions of the book
  - 1.0 describes Django 0.96
  - 2.0 is yet unfinished, describes Django 1.0
- Django Developer Kit Appliance http://www.ogmaciel.com/?p=828

Ruby on Rails project site

- Main site of the project
  - http://rubyonrails.com/
- Main Ruby site
  - http://www.ruby-lang.org/en/
- Aptana is IDE for Ruby
  - http://www.radrails.org/
- It is also available as Eclipse plugin
  - Link to URL for Eclipse update manager
  - http://download.aptana.com/tools/radrails/plugin/install/radrails-bundle

Python sites

- Python site
  - http://python.org
- PyDev plugin for Eclipse
  - http://pydev.org/ http://pydev.sourceforge.net/
- Update URL for Eclipse
- http://pydev.org/updates