A Practical Comparison of Agile Web Frameworks

David Díaz Clavijo
david@diazclavijo.com

University of Las Palmas de Gran Canaria
School of Computer Science

Tutors:
Cayetano Guerra Artal
Alexis Quesada Arencibia
Lydia Esther Bolaños Medina

1 February, 2014
Table of contents

1. State of the art
2. Objectives and methodology
3. Web frameworks selection
4. Test design
5. Comparison of programming languages
6. Comparison of web frameworks
7. Dissemination results
8. Conclusions and future works
Outline

1. State of the art
   - A Comparison Model of Agile Web Frameworks
   - Raible’s matrix
   - Wähner categorization of web frameworks

2. Objectives and methodology

3. Web frameworks selection

4. Test design

5. Comparison of programming languages

6. Comparison of web frameworks

7. Dissemination results

8. Conclusions and future works
State of the art

- A Comparison Model of Agile Web Frameworks by The Technical University of Madrid

- Matt Raible’s speeches and his matrix

- Wähner comparison: categorizes web frameworks by classifying them into different types of web applications
## State of the art

A Comparison Model of Agile Web Frameworks

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rails</th>
<th>Grails</th>
<th>Trails</th>
<th>Roma</th>
</tr>
</thead>
<tbody>
<tr>
<td>D.1 – Are data migrations built-in in the development process? (20 %)</td>
<td>100 %</td>
<td>0 %</td>
<td>0 %</td>
<td>0 %</td>
</tr>
<tr>
<td>D.2 – Is schema database automatically inferred from domain definition? (40 %)</td>
<td>0 %</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
<tr>
<td>P.1 – Is there a generator of static presentation code for CRUD operations on models? (20 %)</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
<td>100 %</td>
</tr>
</tbody>
</table>
State of the art
A Comparison Model of Agile Web Frameworks
## State of the art

### Raible's matrix

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Struts 2</th>
<th>Spring MVC</th>
<th>Wicket</th>
<th>JSF 2</th>
<th>Tapestry</th>
<th>Stripes</th>
<th>GWT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer Productivity</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>Developer Perception</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Learning Curve</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Project Health</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>Developer Availability</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>Job Trends</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Templating</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
</tr>
<tr>
<td>Components</td>
<td>0.00</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Ajax</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>Plugins or Add-Ons</td>
<td>0.50</td>
<td>0.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Scalability</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Testing</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
</tr>
<tr>
<td>i18n and i10n</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Validation</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Multi-language Support (Groovy / Scala)</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Quality of Documentation/Tutorials</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Books Published</td>
<td>1.00</td>
<td>1.00</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>0.50</td>
<td>1.00</td>
</tr>
<tr>
<td>REST Support (client and server)</td>
<td>0.50</td>
<td>1.00</td>
<td>0.50</td>
<td>0.00</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>Mobile / iPhone Support</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Degree of Risk</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>14.5</strong></td>
<td><strong>17</strong></td>
<td><strong>15</strong></td>
<td><strong>13.5</strong></td>
<td><strong>15</strong></td>
<td><strong>14</strong></td>
<td><strong>17</strong></td>
</tr>
</tbody>
</table>
State of the art
Wähner categorization of web frameworks
Outline

1. State of the art

2. Objectives and methodology
   - Objectives
   - Methodology for dissemination of the project
   - Methodology for the comparison
   - Project plan

3. Web frameworks selection

4. Test design

5. Comparison of programming languages

6. Comparison of web frameworks

7. Dissemination results

8. Conclusions and future works

david@diazclavijo.com
Objectives

- To analyze the state of web frameworks technologies, focusing on agile web frameworks

- To compare a set of agile web frameworks through a practical test

- Dissemination of the project
Objectives and methodology

Methodology for dissemination of the project
Web frameworks are chosen based on the following criteria:

- Results in previous comparisons
- Results in benchmarks
- Statistics of usage
- Popularity in the community

A test is defined to be executed with each of the web frameworks. This test consists in the following steps:

- Reading the programming language documentation (8 hours)
- Making the exercises to demonstrate language knowledge (12 hours)
- Reading the web framework documentation (30 hours)
- Developing a simple website: A multiblog system with an author per blog, MyBlog (45 hours)
Objectives and methodology

Project Plan

State of the art

Objectives

Dissemination

Comparison

Project plan

Web frameworks selection

Test design

Comparison of programming languages

Comparison of web frameworks

Dissemination results

Conclusions and future works

david@diazclavijo.com

Comparison of Web Frameworks

Project plan. Hours in percentage. Total hours: 850

- State of the art
- Diffusion: blog development
- Test definition
- Exercises design
- Test execution
- Conclusions

0 10 20 30 40 50 60 70
Outline

1. State of the art
2. Objectives and methodology
3. Web frameworks selection
   - Chosen web frameworks
4. Test design
5. Comparison of programming languages
6. Comparison of web frameworks
7. Dissemination results
8. Conclusions and future works
A cross-language set of web frameworks was decided:
A cross-language set of web frameworks was decided:

- Innovator
- Good results in comparison
A cross-language set of web frameworks was decided:

- Innovator
- Good results in comparison

- Java is highly used
- It is Rails in JVM
- Presence in other comparisons
A cross-language set of web frameworks was decided:

- **Innovator**
  - Good results in comparison

- **Java is highly used**
  - It is Rails in JVM
  - Presence in other comparisons

- **Python usage**
  - Python’s framework
  - Usage of Django
A cross-language set of web frameworks was decided:

- Innovator
- Good results in comparison

- Java is highly used
- It is Rails in JVM
- Presence in other comparisons

- Python usage
- Python’s framework
- Usage of Django

- PHP is the most used server-side language
- Popularity
Outline

1. State of the art
2. Objectives and methodology
3. Web frameworks selection
4. Test design
   - Exercises for learning each programming language
   - Development steps of Myblog
5. Comparison of programming languages
6. Comparison of web frameworks
7. Dissemination results
8. Conclusions and future works
Test design

Exercises for learning each programming language

- Operations over files, strings and use of regular expressions
- Creation of a class to support operations from a number set
- The composite pattern: to practice inheritance
Test design
Development steps of Myblog

- User system with authentication
- CRUD operations with Blogs, Posts, Comments
- Adapting views to appear similar to what was designed: only in content. Good visual design is not an objective
- User-friendly URLs
Programming languages comparison

Hours required in the exercises development

<table>
<thead>
<tr>
<th>Exercises</th>
<th>Ruby</th>
<th>Groovy</th>
<th>Python</th>
<th>PHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Strings, files and regular expressions</td>
<td>4,5</td>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Class of number set</td>
<td>4</td>
<td>2,5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Composite Pattern</td>
<td>1,5</td>
<td>1</td>
<td>1</td>
<td>0,5</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>10</strong></td>
<td><strong>6,5</strong></td>
<td><strong>4</strong></td>
<td><strong>4,5</strong></td>
</tr>
</tbody>
</table>
## Comparison of programming languages

### Readability of programming languages

<table>
<thead>
<tr>
<th>Code examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Is the list of numbers empty?</strong></td>
</tr>
<tr>
<td><strong>Ruby</strong></td>
</tr>
<tr>
<td><code>@theNumberSet.empty?</code></td>
</tr>
<tr>
<td><strong>Python</strong></td>
</tr>
<tr>
<td><code>len(self.the_set) == 0</code></td>
</tr>
<tr>
<td><strong>Collect the names of each element and write them in a paper called result</strong></td>
</tr>
<tr>
<td><strong>Ruby</strong></td>
</tr>
<tr>
<td>`@composite.each{</td>
</tr>
<tr>
<td><code>result+=item.toString</code></td>
</tr>
<tr>
<td><code>}</code></td>
</tr>
<tr>
<td><strong>Python</strong></td>
</tr>
<tr>
<td><code>for element in self.listComponents:</code></td>
</tr>
<tr>
<td><code>string += element.toString()</code></td>
</tr>
</tbody>
</table>

---

david@diazclavijo.com
Groovy is 5 times faster than PHP, Ruby and Python. It is similar to Ruby.

Python is the language that has more useful libraries in this comparison. It is the language which allowed to finish development earlier.

Ruby is the most readable language. It is the language which needed more time to finish development too and the first one to develop with.

PHP is the second language which allowed to develop in shorter time. It is closer to C++, less readable than others in this comparison and it is designed for web development.
Outline

1. State of the art
2. Objectives and methodology
3. Web frameworks selection
4. Test design
5. Comparison of programming languages
6. Comparison of web frameworks
   - Hours in development
   - Documentation comparison
   - Authentication implementation
   - CRUD functions
   - Lines of code
   - Conclusions
7. Dissemination results

david@diazclavijo.com
Comparison of web frameworks

Hours in development

State of the art
Objectives and methodology
Web frameworks selection
Test design
Comparison of programming languages
Comparison of web frameworks
Timings
Documentation
Authentication
CRUD functions
Lines of code
Conclusions
Dissemination results

david@diazclavijo.com

Comparison of Web Frameworks 24 / 43
## Comparison of web frameworks

**Documentation comparison**

<table>
<thead>
<tr>
<th>Framework</th>
<th>Complete and easy to understand</th>
<th>Full of examples</th>
<th>Differentiates between beginner and advanced</th>
<th>Great support from the community</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rails</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Grails</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Django</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>CodeIgniter</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

**Rails**
- Complete and easy to understand
- Full of examples
- Differentiates between beginner and advanced
- Great support from the community

**Grails**
- Long documentation and difficult to understand
- Lack of examples
- It is not clearly differentiated between beginner and advanced
- Poor support from the community

**Django**
- Complete and easy to understand
- Full of examples
- Differentiates between beginner and advanced
- Great support from the community

**CodeIgniter**
- Concise and easy to understand
- There are no many examples but they are not needed
- Enough support from the community
<table>
<thead>
<tr>
<th><strong>Rails</strong></th>
<th><strong>Grails</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A complete how-to to implement an own authentication system</td>
<td>Several plugins to choose from</td>
</tr>
<tr>
<td>It required the least amount of time to finish: 13.5 hours</td>
<td>Plugins documentation is difficult to understand</td>
</tr>
<tr>
<td></td>
<td>It was tried to build an own system. Finally, a plugin was used</td>
</tr>
<tr>
<td></td>
<td>It required the biggest amount of time: 31.5 hours</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Django</strong></th>
<th><strong>CodeIgniter</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>It has an authentication system included in the core</td>
<td>It has a library which is well accepted</td>
</tr>
<tr>
<td>It needed 20 hours to be adapted to the requirements of the project</td>
<td>The library was copied into the project and adapted</td>
</tr>
<tr>
<td></td>
<td>It required 15 hours</td>
</tr>
</tbody>
</table>
### Comparison of web frameworks

**CRUD functions**

<table>
<thead>
<tr>
<th>Framework</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rails</td>
<td>Using a command, which defines the table and its fields, the code for the CRUD operations is created</td>
</tr>
<tr>
<td>Grails</td>
<td>The domain and its relations are defined and, afterwards, the code can be generated with a command</td>
</tr>
<tr>
<td>Django</td>
<td>It uses the class-based views</td>
</tr>
<tr>
<td>CodeIgniter</td>
<td>Most of the code has to be written</td>
</tr>
</tbody>
</table>
### Comparison of web frameworks

#### CRUD functions

<table>
<thead>
<tr>
<th>Framework</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rails</strong></td>
<td>Using a command, which defines the table and its fields, the code for the CRUD operations is created</td>
</tr>
<tr>
<td><strong>Grails</strong></td>
<td>The domain and its relations are defined and, afterwards, the code can be generated with a command</td>
</tr>
<tr>
<td><strong>Django</strong></td>
<td>It uses the class-based views</td>
</tr>
<tr>
<td><strong>CodeIgniter</strong></td>
<td>Most of the code has to be written</td>
</tr>
</tbody>
</table>

A small example will be exposed.

The controllers in different frameworks for creating a post:
Comparison of web frameworks
Code generated for creating a post in Rails

The next command is executed to generate the code:

```
$ rails generate scaffold Post title:string body:texts
```

Code generated:

```ruby
def create
  @post = Post.new(params[:post])

  respond_to do |format|
    if @post.save
      format.html { redirect_to @post, :notice => 'Post was successfully created.' }
      format.json { render :json => @post, :status => :created, :location => @post }
    else
      format.html { render :action => "new" }
      format.json { render :json => @post.errors, :status => :unprocessable_entity }
    end
  end
end
```
The domain is defined:

```java
class Post {
    String title
    String body
}
```

Based on the domain, the code generated:

```java
def save() {
    def postInstance = new Post(params)
    if (!postInstance.save(flush: true)) {
        render(view: "create", model: [postInstance: postInstance])
        return
    }

    flash.message = message(code: 'default.created.message', args: [message(code: 'post.label', default: 'Post'), postInstance.id])
    redirect(action: "show", id: postInstance.id)
}
```
In Django, it is inherited from the class CreateView:

```python
class PostCreate(CreateView):
    model = Post

    @method_decorator(login_required)
    def dispatch(self, *args, **kwargs):
        return super(PostCreate, self).dispatch(*args, **kwargs)

    def form_valid(self, form):
        form.instance.blog = self.request.user.blog
        return super(PostCreate, self).form_valid(form)
```

Create a post in Django using class-based views, inheritance.
In CodeIgniter, the following method would be written:

```php
function create()
{
    $this->tank_auth->logged_in_or_redirect();
    $this->form_validation->set_rules($this->mpost->rules);
    // set rules
    if ($this->form_validation->run())
    {
        $user_id = $this->tank_auth->get_user_id(); // always to the current
        if ($this->mpost->create($post))
        {
            $this->session->set_flashdata('message', 'Post successfully created');
            redirect('/'.$this->tank_auth->get_username());
            ...
        }
    }
    $this->load->view('templates/header', $data);
    $this->load->view('post/create', $data);
}
```

## Comparison of web frameworks

### CRUD functions

<table>
<thead>
<tr>
<th>Framework</th>
<th>Feature</th>
<th>Feature</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rails</strong></td>
<td>Code generated by using a command defining the table</td>
<td>Concise code and readable</td>
<td>Great support for JSON and Javascript</td>
</tr>
<tr>
<td><strong>Grails</strong></td>
<td>The domain and its relations are defined. Based on this definition, the code is generated</td>
<td>More extensive code and less readable</td>
<td>It supports internationalization</td>
</tr>
<tr>
<td><strong>Django</strong></td>
<td>It uses the class-based views</td>
<td>This produces the code which best follows the DRY principle</td>
<td>The most concise code</td>
</tr>
<tr>
<td><strong>CodeIgniter</strong></td>
<td>Most of the code has to be written</td>
<td>It is not necessary to learn too much to start. Hence, it is faster to begin to develop</td>
<td>The code is larger and more repetitive</td>
</tr>
</tbody>
</table>
## Comparison of web frameworks

### Lines of code

<table>
<thead>
<tr>
<th></th>
<th>Rails</th>
<th>Grails</th>
<th>Django</th>
<th>CodeIgniter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Views</strong></td>
<td>375</td>
<td>777</td>
<td>314</td>
<td>507</td>
</tr>
<tr>
<td><strong>Models</strong></td>
<td>59</td>
<td>76</td>
<td>56</td>
<td>429</td>
</tr>
<tr>
<td><strong>Controllers</strong></td>
<td>271</td>
<td>436</td>
<td>296</td>
<td>516</td>
</tr>
<tr>
<td><strong>Routing files</strong></td>
<td>20</td>
<td>44</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>725</td>
<td>1333</td>
<td>692</td>
<td>1470</td>
</tr>
</tbody>
</table>
Comparison of web frameworks
Myblog in Django and CodeIgniter

State of the art
Objectives and methodology
Web frameworks selection
Test design
Comparison of programming languages
Comparison of web frameworks
Timings
Documentation
Authentication
CRUD functions
Lines of code
Conclusions
Dissemination results

Myblog

Search

- Login
- Register

tres first post

Created by tres on 2013-07-26 15:30:11

post tres post in the tres world dude!

number of comments 0

uno second post, later

Created by uno on 2013-07-26 13:35:11

second uno post

number of comments 2

1 2 >
<table>
<thead>
<tr>
<th>Web Frameworks</th>
<th>Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Rails</strong></td>
<td></td>
</tr>
<tr>
<td>- Excellent documentation</td>
<td></td>
</tr>
<tr>
<td>- It requires too much time to learn it</td>
<td></td>
</tr>
<tr>
<td>- It has tool for database management: migrations</td>
<td></td>
</tr>
<tr>
<td>- Great support for Javascript</td>
<td></td>
</tr>
<tr>
<td><strong>Grails</strong></td>
<td></td>
</tr>
<tr>
<td>- It presents the best performance</td>
<td></td>
</tr>
<tr>
<td>- It requires too much time to learn it</td>
<td></td>
</tr>
<tr>
<td>- Lack of documentation and low quality</td>
<td></td>
</tr>
<tr>
<td><strong>Django</strong></td>
<td></td>
</tr>
<tr>
<td>- Excellent documentation</td>
<td></td>
</tr>
<tr>
<td>- It requires too much time to learn it</td>
<td></td>
</tr>
<tr>
<td>- The most concise code</td>
<td></td>
</tr>
<tr>
<td>- Class-based views</td>
<td></td>
</tr>
<tr>
<td><strong>CodeIgniter</strong></td>
<td></td>
</tr>
<tr>
<td>- It is the framework which allows to develop faster for an inexperienced programmer</td>
<td></td>
</tr>
<tr>
<td>- It produces a repetitive code and is less readable than other frameworks</td>
<td></td>
</tr>
<tr>
<td>- Its documentation is short, clear and easy to understand</td>
<td></td>
</tr>
</tbody>
</table>
Outline

1. State of the art
2. Objectives and methodology
3. Web frameworks selection
4. Test design
5. Comparison of programming languages
6. Comparison of web frameworks
7. Dissemination results
   - The blog
   - Scope of the dissemination
   - Referrers
8. Conclusions and future works
Comparison of the ease of learning of Web Frameworks: Rails, Grails, Django and Codelgniter

This article compares the ease of learning of web frameworks: Rails, Grails, Django and Codelgniter based on the experience of learning how to use the framework and developing a toy application, MyBlog, in fixed time. More details about this test can be found here. Points of comparison will be timing, documentation, community support, steps to execute common functions, some features, lines of code and readability of solutions.[…]

Comparison of programming languages: Ruby, Groovy, Python and PHP

Recent Comments
- Wframeworks on Comparison of the ease of learning of Web Frameworks: Rails, Grails, Django and Codelgniter
- Wframeworks on Comparison of programming languages: Ruby, Groovy, Python and PHP
- Comparison of programming languages: Ruby, Groovy, Python and PHP - Blog Websites Frameworks on Programming exercises for language knowledge testing in a comparison of web frameworks
- Comparison of programming languages: Ruby, Groovy, Python and PHP - Blog Websites Frameworks on The practical test for web framework comparison: learning and development steps
- Olisi on Server-side programming language statistics
Dissemination results
Scope of the dissemination

- A post in Raible’s blog about the project
- Several comments which produced feedback
- Some emails from different people and universities asking for the project
# Referrers

<table>
<thead>
<tr>
<th>Referrers</th>
<th>Visits</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. t.co</td>
<td>578</td>
</tr>
<tr>
<td>2. raibledesigns.com</td>
<td>472</td>
</tr>
<tr>
<td>3. java.dzone.com</td>
<td>234</td>
</tr>
<tr>
<td>4. plus.url.google.com</td>
<td>171</td>
</tr>
<tr>
<td>5. websitesframeworks.com</td>
<td>131</td>
</tr>
<tr>
<td>6. facebook.com</td>
<td>115</td>
</tr>
<tr>
<td>7. stackoverflow.com</td>
<td>94</td>
</tr>
<tr>
<td>8. forum.root.cz</td>
<td>68</td>
</tr>
<tr>
<td>9. google.com</td>
<td>64</td>
</tr>
<tr>
<td>10. m.facebook.com</td>
<td>52</td>
</tr>
</tbody>
</table>
Outline

1. State of the art
2. Objectives and methodology
3. Web frameworks selection
4. Test design
5. Comparison of programming languages
6. Comparison of web frameworks
7. Dissemination results
8. Conclusions and future works
   - Conclusions
   - Future works
Conclusions and future works

Conclusions

- It has been a practical approach to the problem. Although the definition of the test is strict and precise, the comparison still lacks objectivity: programmer background and the order in development influenced in the results.

- 125 hours spent in each web framework does not allow to measure productivity.

- A longer time project is useless due to the appearance of new versions.

- A feature-rank comparison, as Villamor and Raible, would have been theoretical and made by someone who had not real experience.

- This approach has taken the advantage of an inexperienced programmer to measure the ease of learning of these frameworks, and an overview of the main features has been made.
Conclusions and future works

Future works

- To complete the model of Villamor for a set of web frameworks
- To compare less popular frameworks and more innovative ones
- To compare more distinct web frameworks. Request-based vs component-based
- To compare front-end web frameworks
- Far away from comparisons, but interesting in the web development knowledge: to choose a web framework and improve it
Thank you very much for your attention,

your questions or comments would be appreciated