Seagull Application Framework

PhpLondon Talk, November 2004
Demian Turner <demian@phpkitchen.com>

Introduction

- What is Seagull?
  - 1 year at Sourceforge, monthly releases
  - 100 developers on mailing list
  - BSD licensed
  - Translated into 16 languages
  - 134, 931 LOC
  - 794 kb download with PEAR installer
- Project goals
- Target audience
- Background

Why use a framework?

- The result of using a framework is that 10–20% of the total code for a project is application specific, giving a huge reduction in code size, and improving readabilty and maintainability considerably.

- Because it's useful to get standard foundations, the same way to build an app. It may be harder at the beginning but then everyone speaks the same language. It's a reason why Java is popular.

What does Seagull do?

- Permissions handling and authentication
- Facilitates team development environment
- Internationalisation/localisation
- Application config
- Personalisation: preferences and themes
- Cacheing and performance
- Simplified error handling

What does Seagull do? /cont.

- Workflow

- Db abstraction

- Class library integration

- Form management

- Data validation

- Component reuse

Figure 1: Typical screen
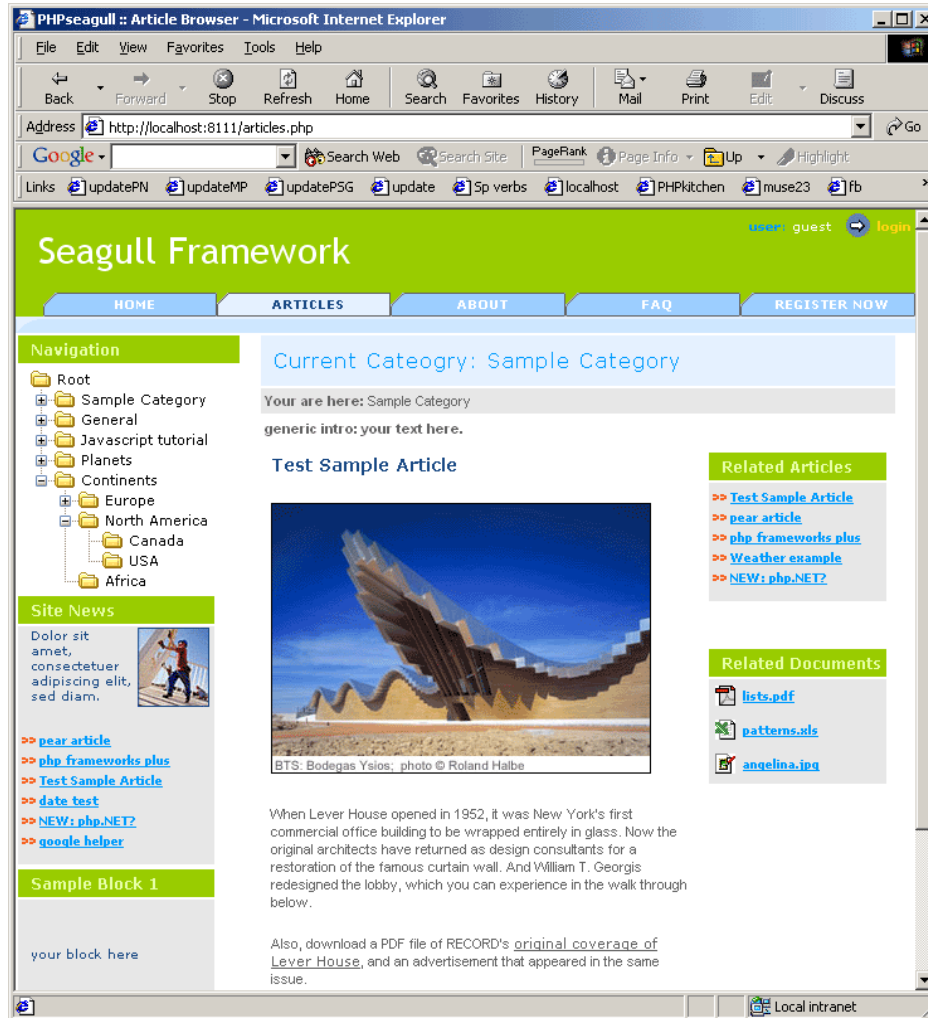
Refresher on OOP, design patterns

- Why use OOP approach?
- Benefits of MVC workflow
- Frequently used patterns:
  - Factory
  - Singleton
  - Template
  - Delegate

PEAR

Pros

- High quality code base
- Active peer review
- Package manager
- Responsive maintainers

Cons

- Inconsistency of quality
- Occasional package obesity

Packages Used

- Benchmark
- Cache_Lite
- Date
- DB
- DB_Pager
- DB_DataObject
- DB_NestedSet
- HTML_Javascript
- HTML_TreeMenu
- HTML_Template_Flexy
- Log
- Mail
- Net_UserAgent
- PHPdocumentor
- System
- Text_Password
- Text_Statistics
- Validate

Figure 3: MVC
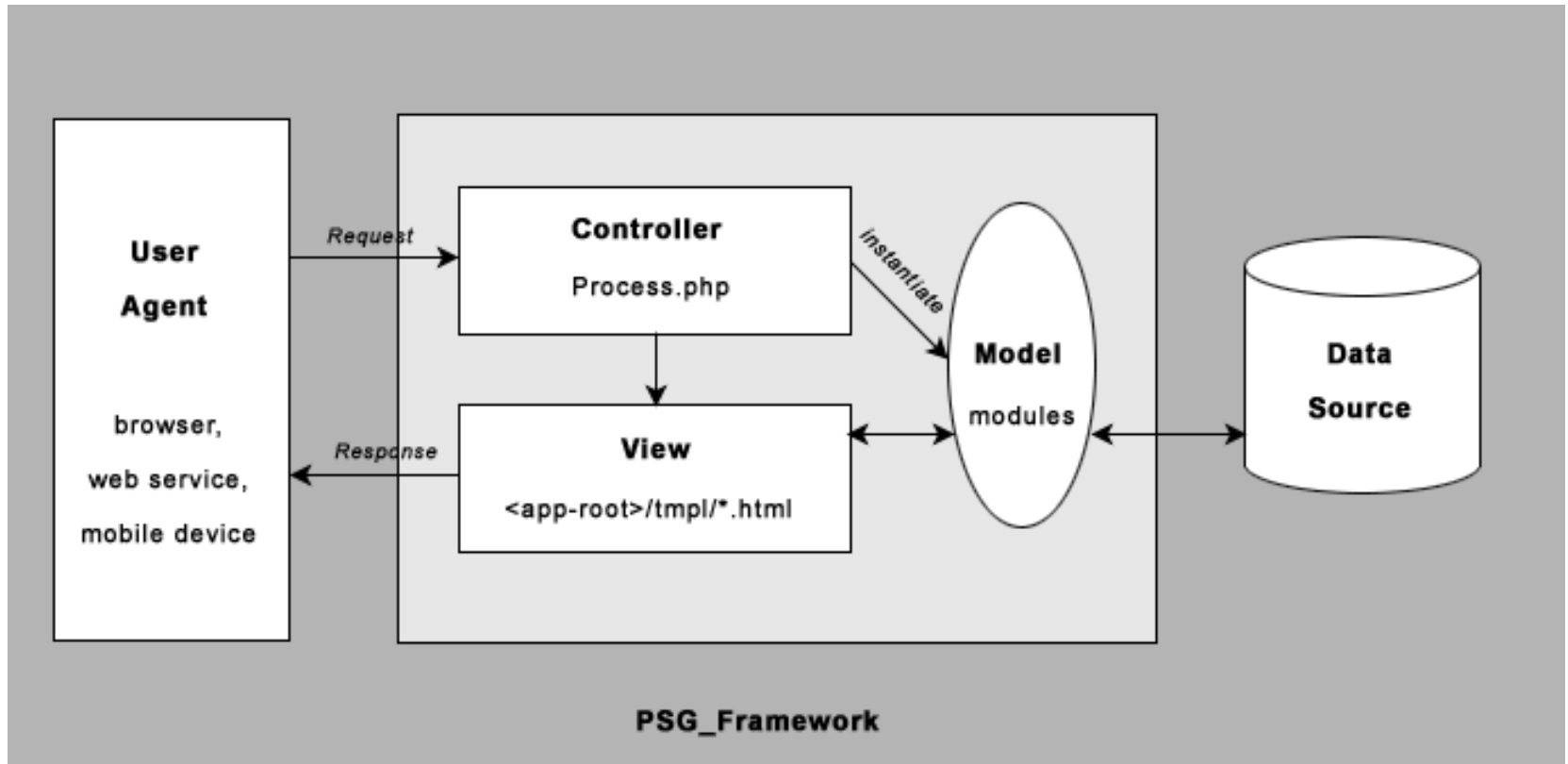
Seagull concepts

- MVC, OOP, modular design
- libs, modules, managers and controller
- workflow:
  - Validate
  - Process
  - Display

Figure 3: Worflow
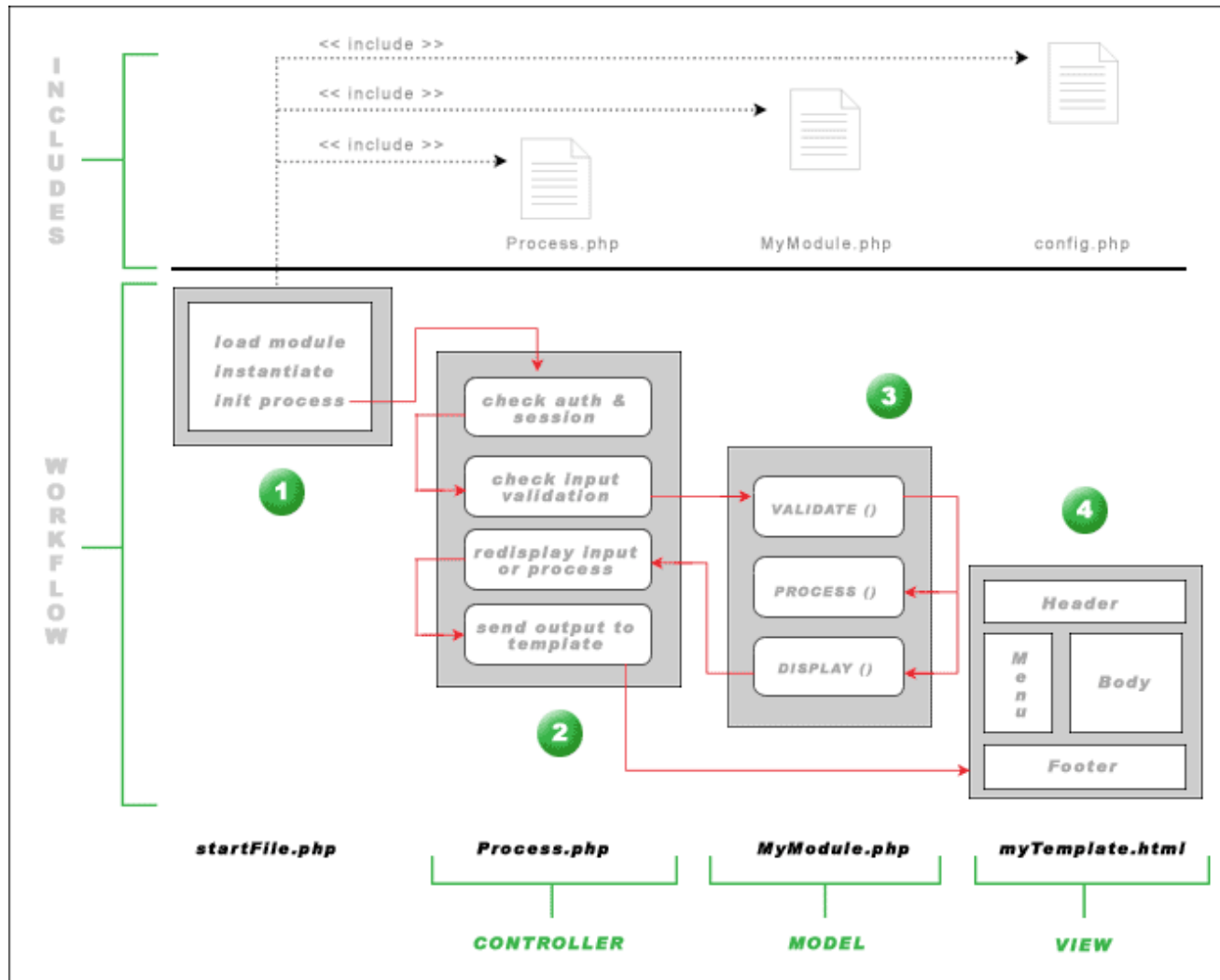
System objects

- Session
- Config
- DB
- Cache
- Preferences
- User

Performance

- Without PHP cacheing or bytecode cache, around 12 reqs/sec
- Typical configuration
  - cpu: amd 1.4 Ghz
  - ram: 512MB
  - Apache/linux

63 reqs/sec

Community

- Active Sourceforge project
- 15 Core developers
- Mailing list
- Wiki documentation project
- Range of skills

Best practices

- PEAR coding standards
- library usage
- unit testing: SimpleTest
- template/business logic separation
- security conscious
- performance
- self-generating documentation: phpDoc

Practical Example A

- FAQ manager example (see http://muse23.com/cgi-bin/cvsweb.cgi/seagull/modules/faq/classes/FaqMgr.php)

Practical Example B

- User manager example (see http://muse23.com/cgi-bin/cvsweb.cgi/seagull/modules/user/classes/UserMgr.php)

Projects using Seagull/1

Projects using Seagull/2

Projects using Seagull/3

Projects using Seagull/4

Projects using Seagull/5

Road Map

- xml-rpc wizard for upgrading/installing modules like PEAR/webmin
- separate core framework
- write tests for all modules
- 3rd party application bridge
- content versioning
- increase db vendor support

Resources

- http://seagull.phpkitchen.com
- http://seagull.phpkitchen.com/docs
- https://sourceforge.net/projects/seagull
- http://marc.theaimsgroup.com/?l=seagull-general
- http://www.phpkitchen.com/index.php?topic=phpFrameworks
- demian@phpkitchen.com

Thank you