

# Lessons Learned: *Experience from the Front Line*

PHP London Conference 2008  
*Friday 29th February*

*Scott MacVicar  
Mike Sullivan*

# Who are we?

- Senior vBulletin Developers
- What is vBulletin?
- Involved with vBulletin since 2001

# Overview

- Specifications
- Revision Control
- Testing
- Internationalisation
- Optimisations

# Specifications

- Document to describe project
- Functional - *What*
- Technical - *How*

# Specifications: Functional

- Project Goals
- Features
- Usage Scenarios
- Get Specific

# Specifications: Technical

- Implementation Goals
- Implementation Details

# Specifications

- More detail is better
- Formality
- Keep updating
- Review

# Specifications: Reasons

- Puts everyone on the same page
- Easier to change now
- Lower barrier to involvement
- Basis for documentation
- Training material

# Revision Control

- Tracks changes to code
- Similar to traditional blueprint revisions

# Revision Control: Reasons

- Distribution of changes
- Comments on changes
- Rollback
- Branching

# Revision Control: Integrated Tools

- ViewVC
- Bug Tracker
- Trac
- IDE

# Revision Control: Examples

- CVS
- Subversion
- bzip / git

# Testing

- Unit / Integration
- System
- Performance
- Usability

# Unit Testing

- Test individual components
- Now for an example...

# Unit Testing: Example

```
function my_add($a, $b) {  
    return $a + $b;  
}
```

```
assert(my_add(2, 2) === 4);  
assert(my_add(2, 3) === 5);
```

# Unit Testing: Reasons

- Regression detection
- Encourages cleaner code
- Basis for API documentation

# Unit Testing: When To Write

- Before development
- When developing
- When fixing bugs

# Code Coverage

- Shows code executed by tests
- Helps find missing tests
- Provided by Xdebug extension

# Testing Tips

- Keep Tests Updated
- Automation
- Multiple Platforms
- Obscure Configurations
- Multiple Browsers
- *“Eat your own dog food”*

# Internationalisation

- Prepare application for worldwide use
- Time zones
- Date / Number formatting
- Text

# Internationalisation: Time zones

- 40 Time zones
- UTC -12 to UTC +14
- Daylight Savings

# Internationalisation: DateTime Object

- Based on the Olson Database
- 64-bit data type
- Handles DST automatically

# Internationalisation: Date Formatting

- Differs from country to country
- mm/dd/yyyy, dd/mm/yyyy, yyyy/mm/dd
- Day names differ per language

```
setlocale(LC_TIME, 'DE_de');  
echo strftime('%A');  
// prints Freitag
```

# Internationalisation: Number Formatting

- Thousand separators
- Decimal point
- 1,234.56; 1.234,56; 1 234,56
- `number_format`
- ICU

# Internationalisation: Text

- Separating interface text
- Break text into phrases
- Keep variables in phrases

# Internationalisation: Methods

- Gettext extension
- Include phrases based on script or groups
- Frameworks provide their own methods

# Internationalisation: Unicode

- Lots of different characters
- Unicode can represent all
- One set of rules
- 1 byte  $\neq$  1 character
- Use UTF-8

# Internationalisation: Unicode Support

- Browsers / Operating Systems
- JavaScript
- PHP 6
- Regular Expressions (PCRE)
- Most DBMS's

# Optimisation

- I think my application is too slow
- I think my application will be too slow
- “Premature Optimization is the root of all evil” (Donald Knuth)

# Optimisation

- "We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil."  
(Donald Knuth)
- Keep performance in mind during design

# Optimisation: Profiling

- Finds slow sections of code
- Apache bench
- XDebug

# Micro Optimisations

- Small changes to code
- Small benefit per change
- Hardware could be cheaper

# Micro Optimisation Examples

1 - `require_once('./includes.php');`

2 - `require_once('/var/www/includes.php');`

---

3 - `"This is some text."`

4 - `'This is some text.'`

# Big Optimisations

- Involves rewriting sections of code
- Bigger gains
- Bigger risks

# Big Optimisation Example

- Summing the first n integers

```
for ($i = 1; $i <= $n; $i++)  
    $total += $i;
```

---

```
$total = (($n * $n) + $n) / 2;
```

# Optimisation: Best vs Worse Case

- Test changes with multiple scenarios
- Potentially implement two methods

n	loop	equation
1	0.00011	0.00011
1,000	0.00083	0.00012
100,000	0.06068	0.00013

*Code was executed 5 times*

# Optimisation: Time vs Memory

- Execution time
- Memory usage
- Faster execution => data cached => more memory usage => less concurrency
- `xdebug_memory_usage()`

Questions?

# Thanks

- Resources
  - <http://talks.macvicar.net>
  - <http://talks.regexia.com>