

# A brief history of PHP at Deakin University: A case study



Kathy Reid

**php**women

Information Technology Services Division  
Deakin University  
[kathy.reid@deakin.edu.au](mailto:kathy.reid@deakin.edu.au)

Asia-Pacific Rep  
PHPWomen.org  
[kathy@phpwomen.org](mailto:kathy@phpwomen.org)  
KathyReid on #phpwomen on freenode

# A brief history of PHP at Deakin

- First used in 1999, circa PHP 3
- Used as a basic **templating** tool
- No database connectivity
- Mmmm anyone like pasta: SPAGHETTI CODE

*Things got better...or did they??  
[dum dum dum-dum...]*

# A brief history of PHP at Deakin

- Database connectivity introduced circa 2001
- A step forward, **dynamic data** could be displayed
- Code designed for ease of development, not ease of maintenance
- **Integrating** with existing corporate applications was difficult
- DB connections were not well managed
- No central code library
- More **SPAGHETTI (bleh! bleh!)**

*We begin to see the light*

# A brief history of PHP at Deakin

- By 2004 there were many database-driven small web applications (50-ish)
- Integration with **corporate applications** (Student system, authentication system) began to occur in an **organic** fashion
- **Large maintenance overhead**, more time maintaining than building new functionality.
- **Some code re-use possible**, but not to a large extent
- **Coding standards** not well defined
- Slightly less spaghetti
- Developers **external to central IT** begin using PHP (skill sets widely vary)

*The next evolution...*

# A brief history of PHP at Deakin

- PHP 4 **end of life** was announced
- Need recognised not just to migrate to PHP 5, but to improve **structure, framework and coding practices** (and make the move to PHP 6 easier)
- **Object orient** our applications (now at 80 web apps and growing, with significant integration with corporate apps)
- Separate '**core**' and '**non-core**' (reports, authentication) functionality in applications
- Time to write new / better apps, **less time spent on maintenance**
- Dealing with **external developers** still an issue “my code is broke, will you fix it for me” at one end of the scale to “wow! You've just written something amazing that we could deploy centrally” at the other

# What we're doing:

- **Shared code library** proposed and designed for re-usability
- Comprehensive **coding standards** (enforced with Snoopy)
- Investigation of enterprise-ready **coding frameworks** using **MVC architecture** (CodeIgniter looking good)
- Common **autodocumentation** (PHPDoc)
- Redesigning applications to be **object-oriented**, and reusable
- Utilising **PHPUnit** to ensure the code that goes into the code library is well tested
- Using ITIL Change Management principles to manage updates to the shared code library
- Providing some public objects to **external developers** without exposing the details underneath

# The future

- PHP 5 in a clustered web environment
- The need to manage integration between applications
- An applications 'portfolio' to understand the lifecycle of applications
- Better able to integrate existing apps with FOSS offerings (WordPress, \*wiki, etc)
- Better, low-cost prototyping
- The need to optimise PHP for the enterprise (ie understanding bottlenecks)
- Planning for PHP 6
- Determining whether PHP is the way to go (or is C# / ASP.NET a better offering?)

# Questions, heckling, etc

