

Dynamic Languages in the Enterprise

Ivo Jansch - ivo@ibuildings.com

ibuildings
THE PHP PROFESSIONALS

What's an Enterprise?

"Any of several ships by that name
in the Star Trek universe" – Wikipedia

"A business, company or comparable organization"

Top 10 Polish Websites

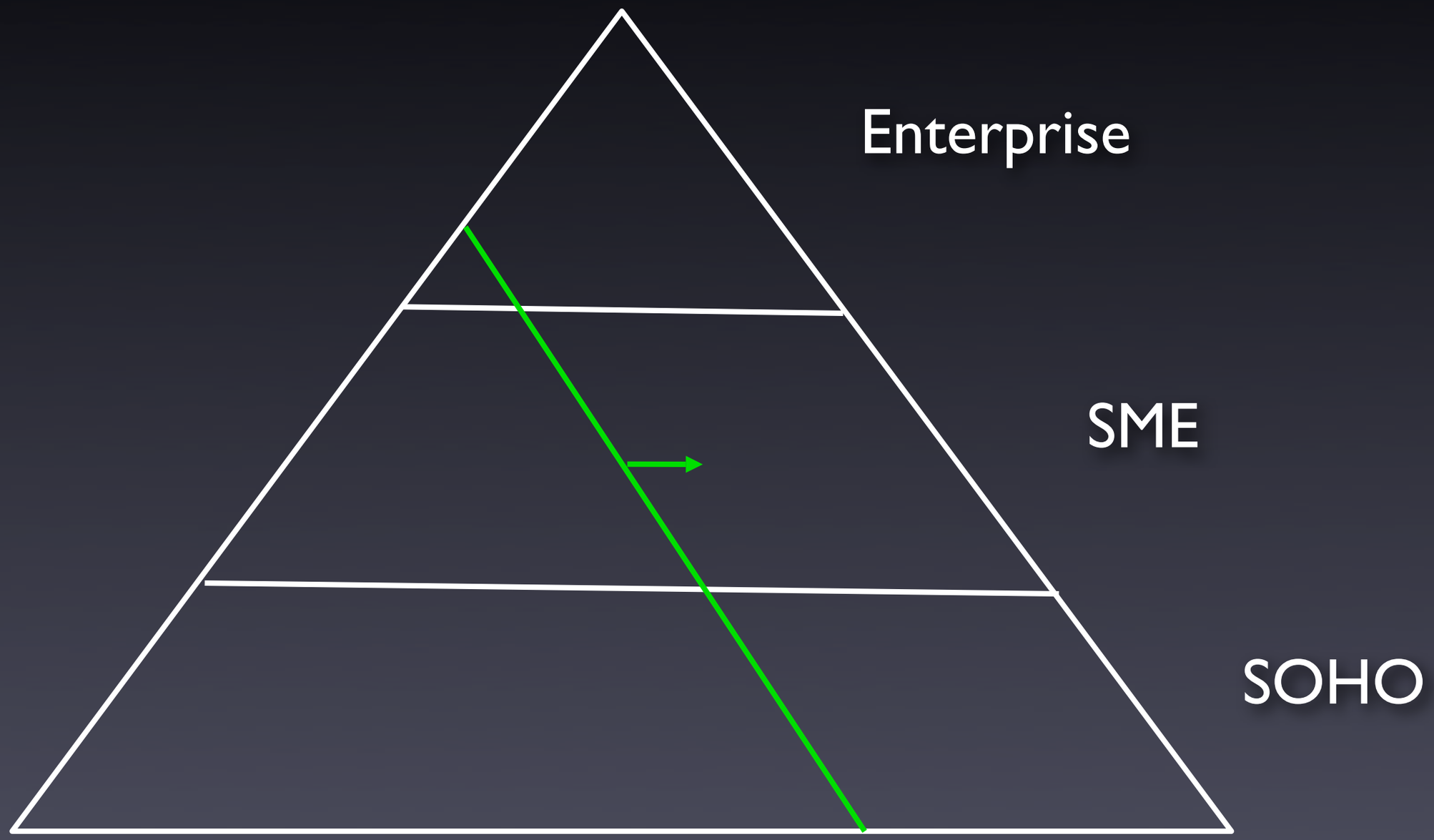
1. Google
2. Nasza-klasa.pl
3. Onet
4. Allegro
5. Wirtualna Polska
6. YouTube
7. Interia.pl
8. Wikipedia
9. Fotka.pl
10. Gazeta

Source: Alexa.com, March 2009

Top 10 Polish Websites

1.	Google	C, Python
2.	Nasza-klasa.pl	Java
3.	Onet	Tcl
4.	Allegro	PHP, Perl
5.	Wirtualna Polska	Java, PHP
6.	YouTube	Python
7.	Interia.pl	PHP
8.	Wikipedia	PHP
9.	Fotka.pl	PHP
10.	Gazeta	Java

The Quest So Far



Are we ‘enterprise ready’?

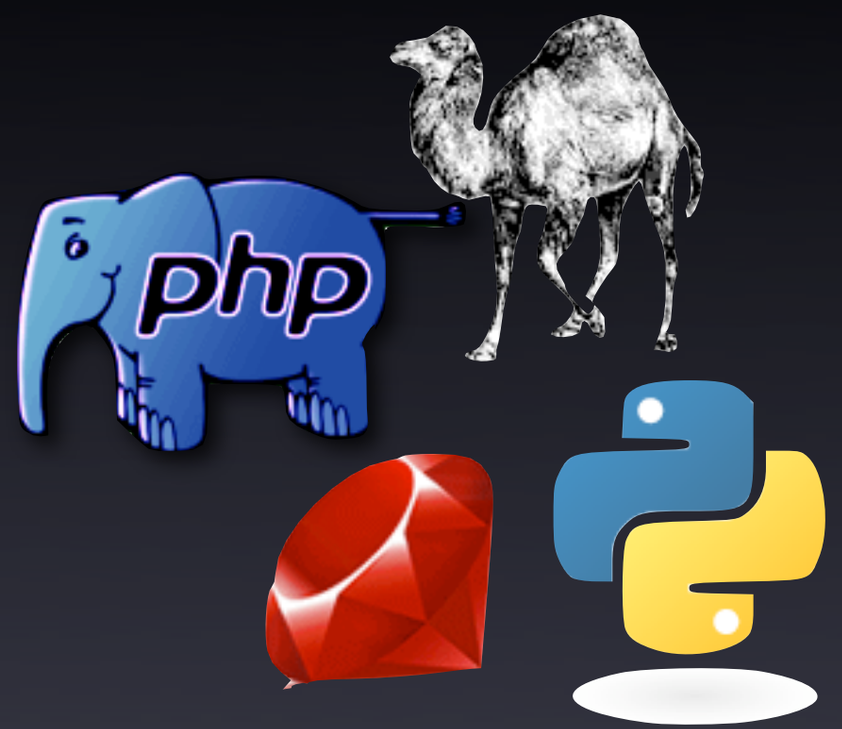
IBM

Microsoft

ORACLE

ibuildings [i]
THE PHP PROFESSIONALS

Are we‘enterprise ready’?



- Big
- Enterprise friendly
- enterpriCe
- CS required

- Small
- Lightweight
- Inexpensive
- Easy to learn

(ok, except perl ;-))

ibuildings
THE PHP PROFESSIONALS

A word from my girlfriend

It's not the **SIZE** of the tool...

It's how you USE it.

Traditional metaphor

LEGO bricks

Improved metaphor

The web is no longer a toy

Let's use the metaphor of actual bricks

About Bricks

- Extreme Simplicity
- Easy to learn
- Versatile
- Inexpensive

Enterprise Development

In 10 steps

Step 1 – The Team

"I have read about your plans to build a new skyscraper and I am applying for a job.

I have a lot of experience with Bricks. I taught myself how to use them and have been maintaining our family shed for a few years now."

Step 1 – The Team

- Be a Software Engineer
- Train your skills
- Study OO principles
- Consider Certification

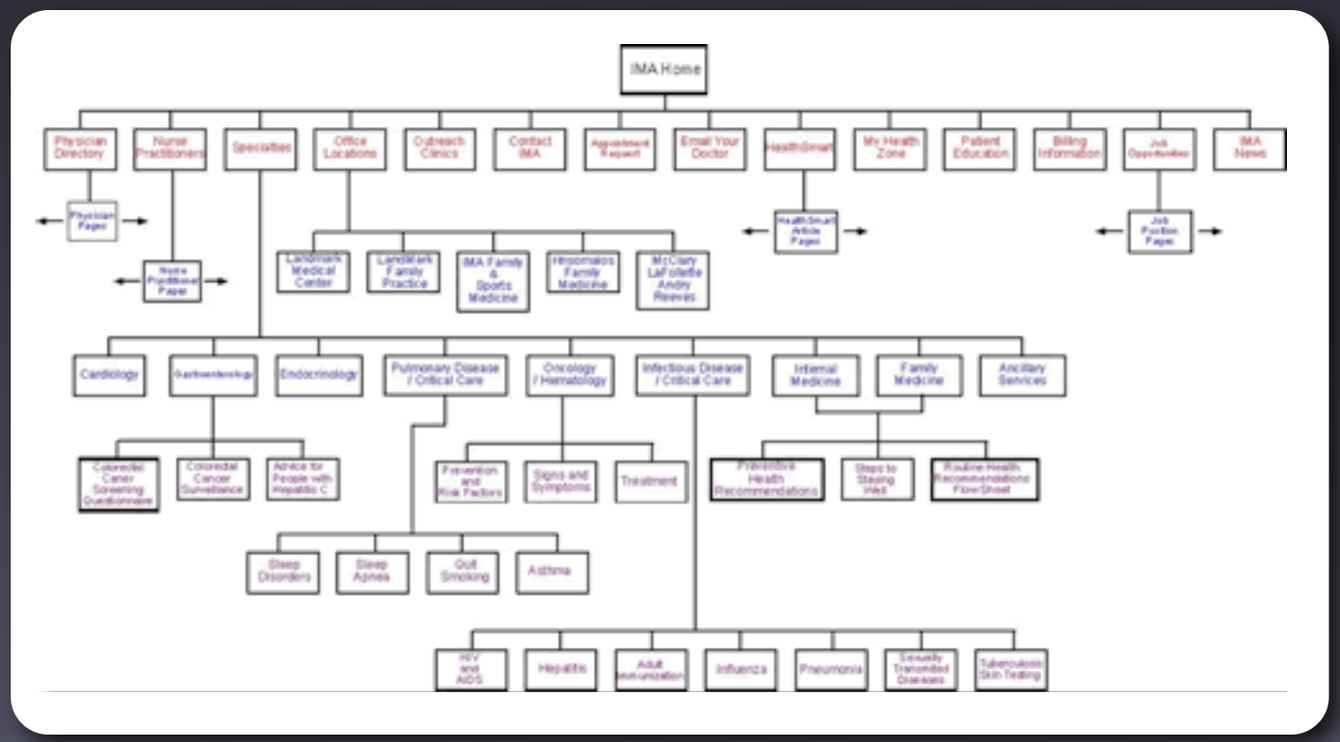
Step 2 - Requirements

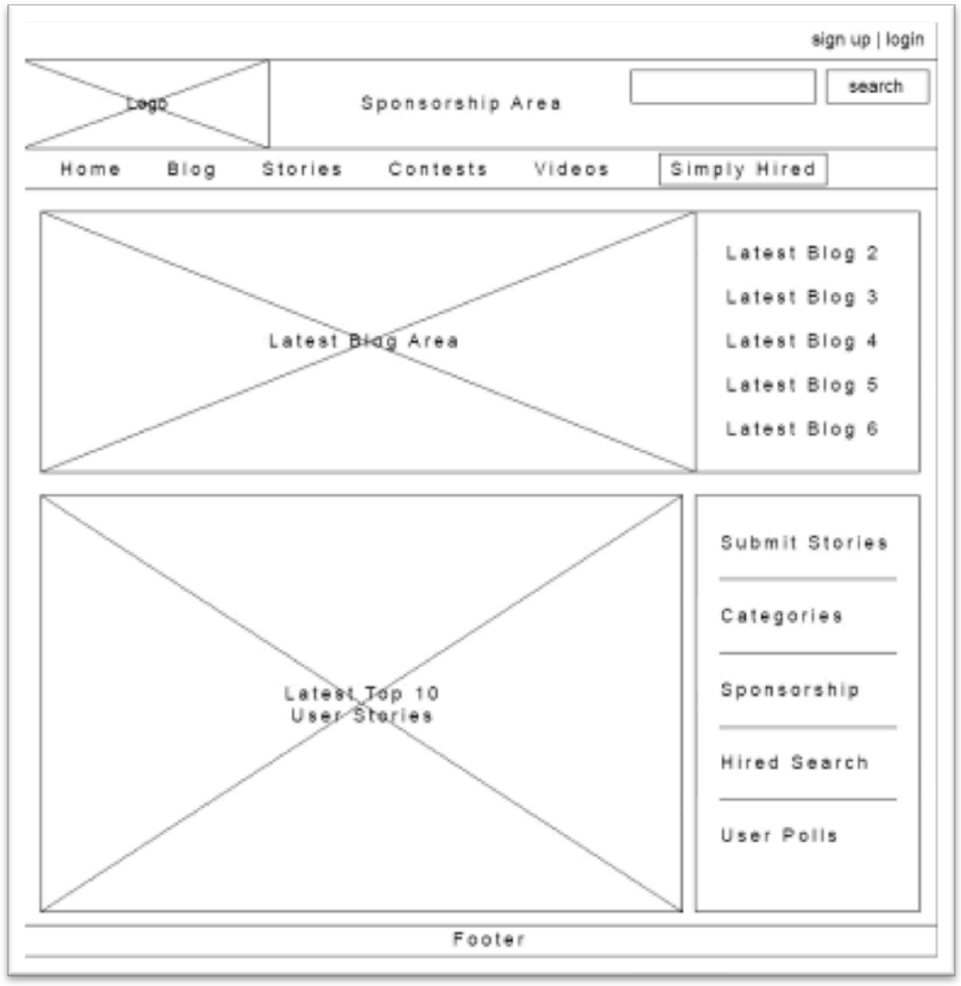
- What does the customer want?
- What will visitors want?
- What does the customer really need?

Step 2 - Requirements

Functional Design

- Requirements Definition
- Interaction Design / Wireframes
- Flow Diagrams
- Tip: Axure





Step 3 - Architecture

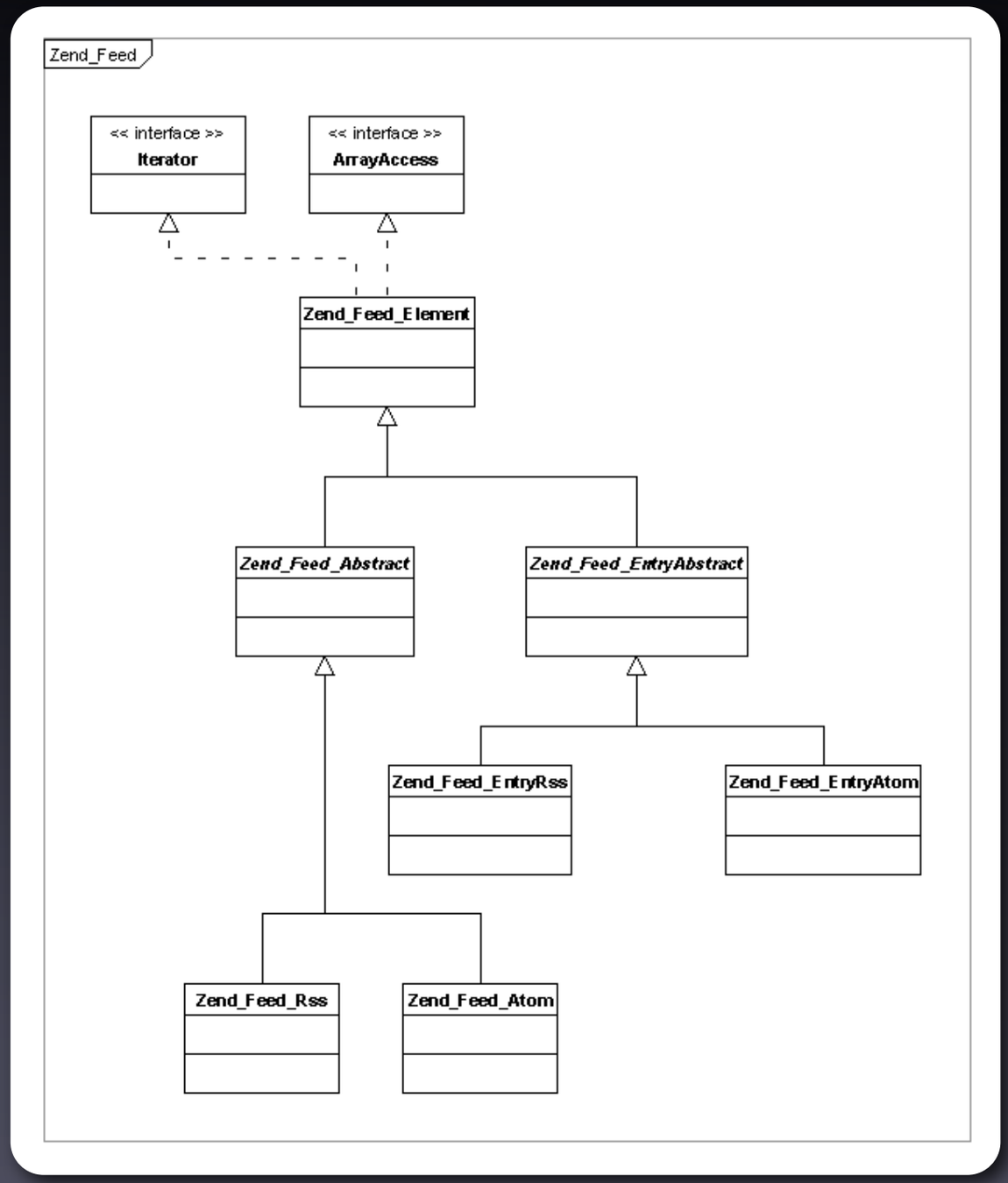
Don’t just start stacking bricks

Create an architecture first

Step 3 - Architecture

Technical Design

- Modeling
  - Class Diagrams
  - ER Diagrams (data model)
  - Collaboration Diagrams
  - Use Cases etc.
- Tools:
  - UML
  - Whiteboards



Step 3 - Architecture

High Level Architectures:

- MVC (Model, View, Controller)
- SOA (Service Oriented Architecture)
- Multi-tier development (Frontend, Application, BL, Data)
- CBD (Component Based Development)

Step 4 - Tools

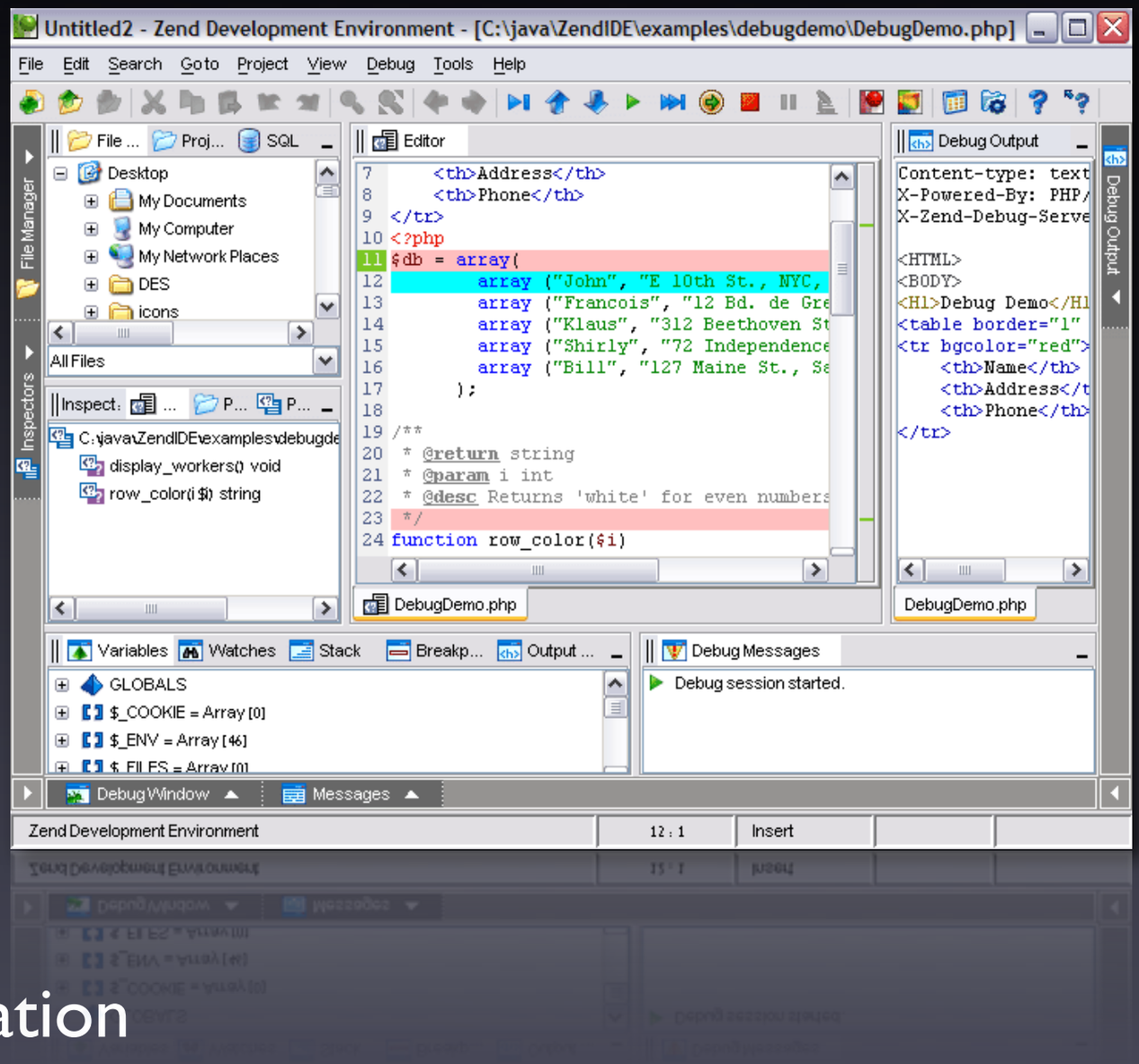
You don't need tools…

But they make you productive

Step 4 - Tools

Development:

- Eclipse
- Zend Studio
- Vim
- Komodo



IDE's versus Editors

- Debugging & Profiling
- Syntax Check
- Cross-Referencing / Navigation

Step 4 - Tools

Source Control

- CVS / SVN / MS VSS
- Bitkeeper / GIT

Documentation

- PHP - phpDocumentor http://phpdoc.org
- Python - Epydoc http://epydoc.sourceforge.net/
- Perl - OODoc http://perl.overmeer.net/oodoc/
- Ruby - RDoc http://rdoc.sourceforge.net/
- Wiki http://en.wikipedia.org/wiki/List_of_wiki_software

Step 5 - Foundation

Start stacking bricks?

- How many bricks does it take?
- What about stability?

Start with a foundation

Step 5 - Foundation

Use a framework

- Provides guidelines (frame)
- Off the shelf components

Examples

- PHP - Zend Framework http://framework.zend.com
- Ruby - Rails http://rubyonrails.org/
- Python - Django http://www.djangoproject.com
- Perl - Catalyst http://www.catalystframework.org

Step 5 - Foundation

The "Not Invented Here" Syndrome:

- "The existing frameworks are no good.
  I can do this better."
- "This framework is no good.
  It can do A through Y but I need Z."
- "This framework is too big.
  It provides A-Z but we only need A and B."
- "I know there's a good framework.
  But I would like to learn."

Step 6 - Design Patterns

Requirement 1056.4:

We need to be able to look outside,
but we can't make holes in the wall
(rain should be kept outside).

When it's sunny, a hole is ok.

Step 6 - Design Patterns

- A 'window' is a concept
  - Best practice way of solving a particular problem
- In IT, we call this a 'design pattern'
- Popular patterns on the web:
  - MVC, Factory, Singleton, Registry, Decorator
- Good read:
  - php|architect's Guide to PHP Design Patterns - Jason E. Sweat

Step 7 - Testing

Is your software tested **after** it has gone live?

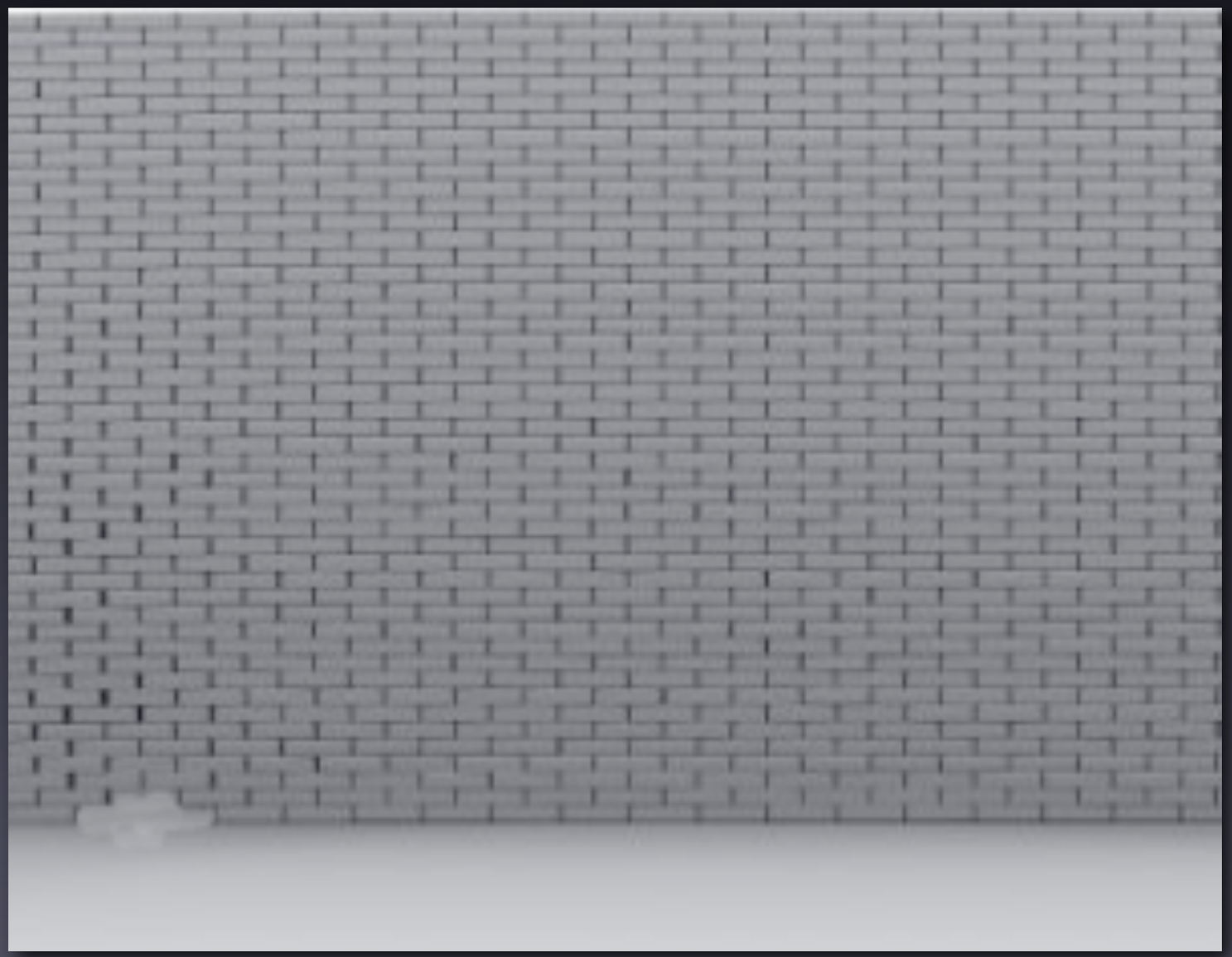
Step 7 - Testing

Various types of testing:

- Developer testing
- Functional testing
- Environment testing
- Performance testing
- Usability testing

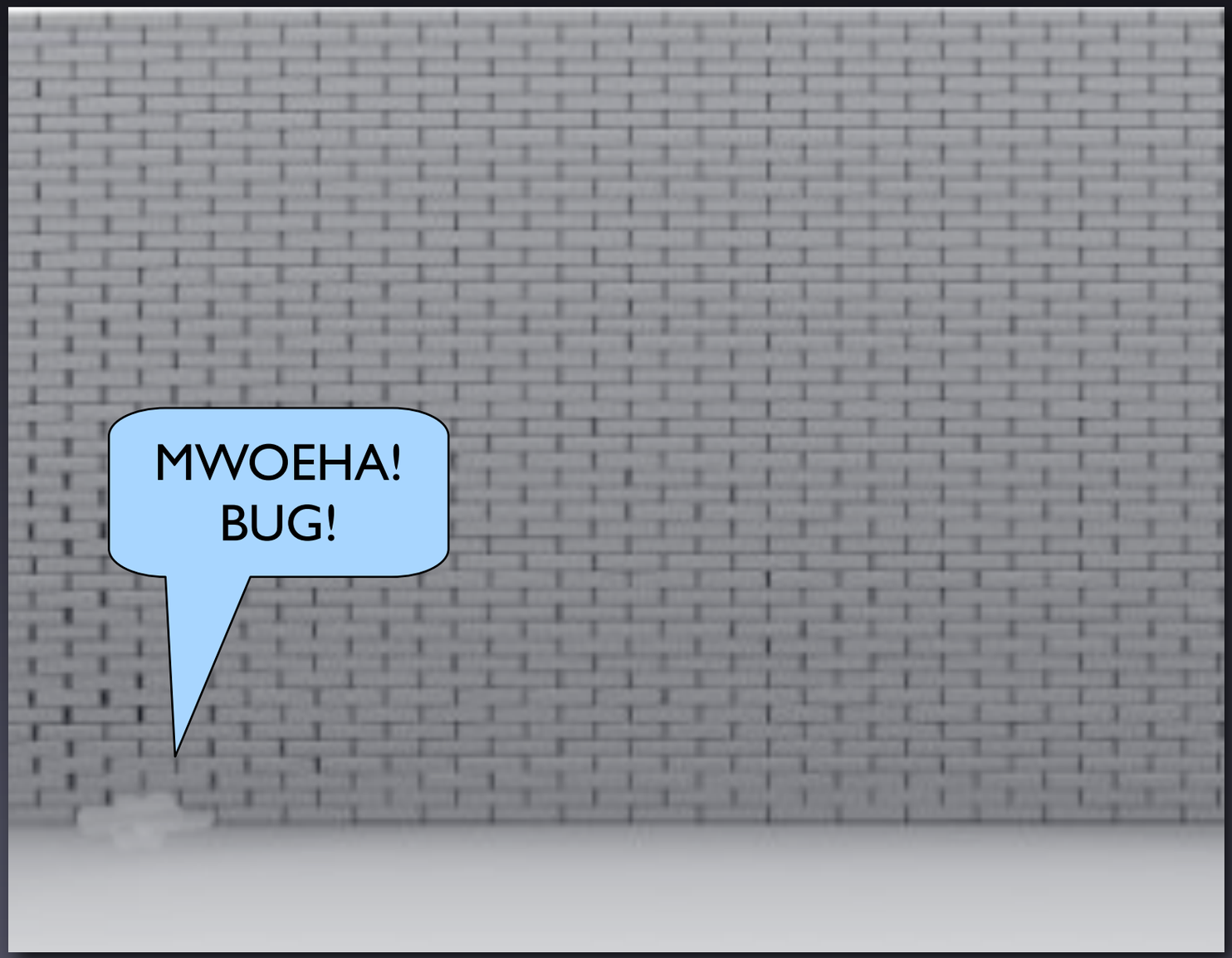
Step 7 - Testing

A common scenario…



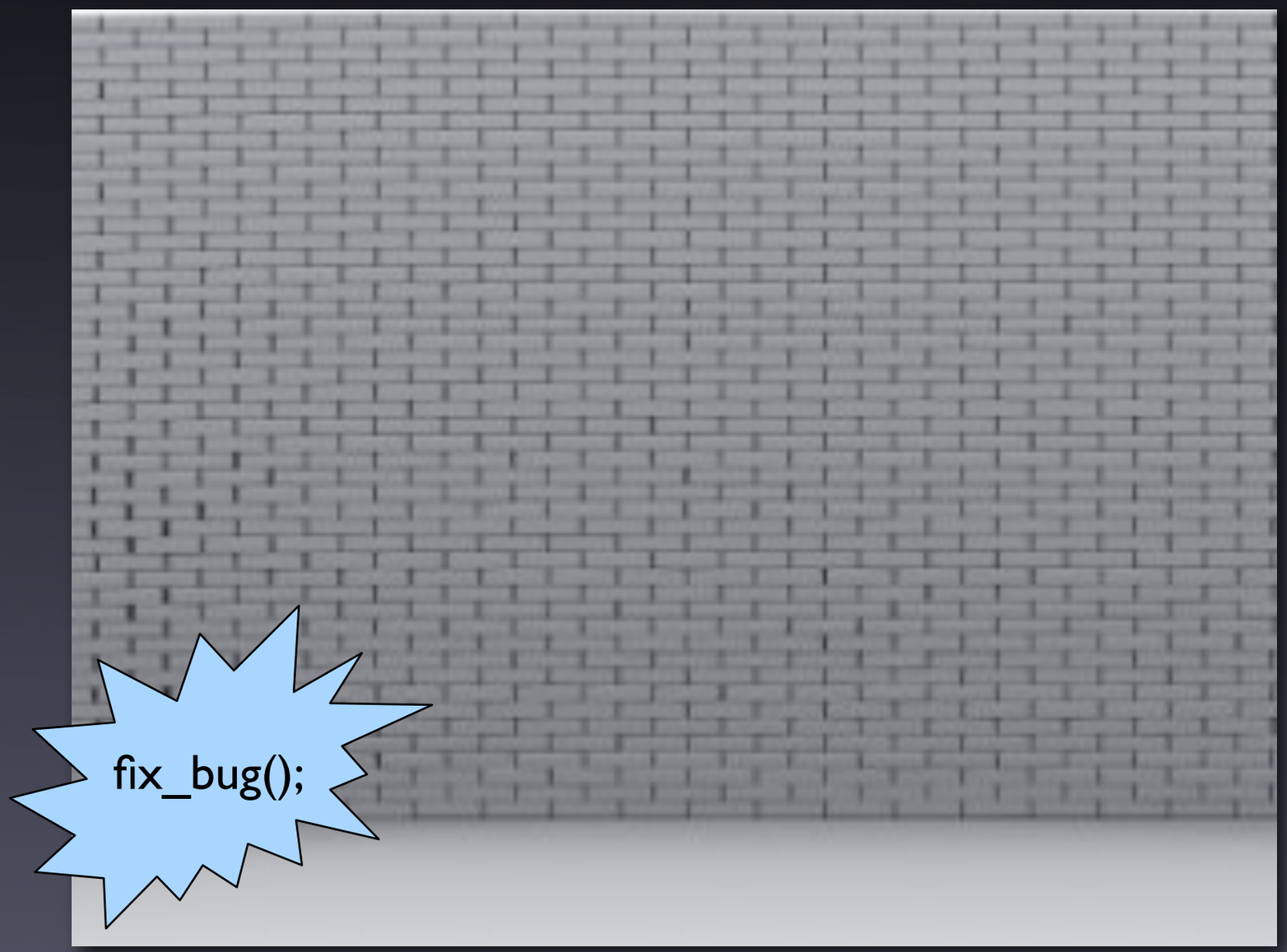
Step 7 - Testing

The user complains…



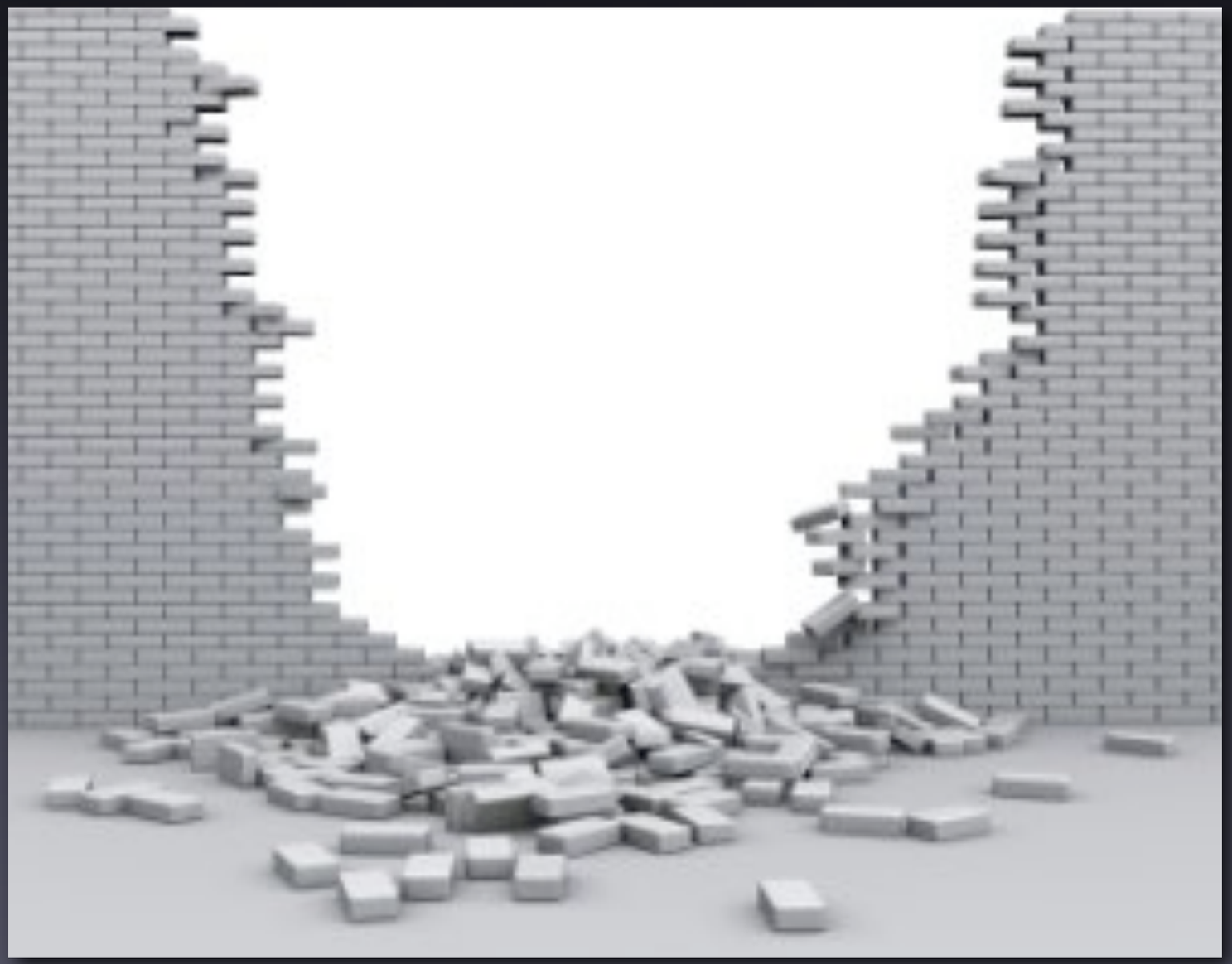
Step 7 - Testing

Developer attacks the problem…



Step 7 - Testing

Problem Solved!



Step 7 - Testing

- Solution: Unit Tests
  - Automated testing after each change
  - Prevents regressions
- Testing for web applications:
  - PHP - PHPUnit http://www.phpunit.de/
  - Ruby - Test::Unit http://tinyurl.com/testunit
  - Python - Unittest http://docs.python.org/library/unittest.html
  - Perl - PerlUnit http://perlunit.sourceforge.net/
- Continuous Integration:
  - CruiseControl http://cruisecontrol.sourceforge.net/

Step 7 - Testing

- Test Driven Development
  1. Define functionality
  2. Create testcase
  3. Run test => test fails
  4. Implement functionality
     - Test succeeds? Done
     - Test fails? Refactor

Repeat step 4 until finished

Step 8 - Optimization

Users complain:

"Every time I want coffee I have to go to the top floor to get some."

Step 8 - Optimization

- Solution:
  - Create small coffee corners on every floor so people don't have to go to the main restaurant every time.
- Developers call this 'caching'
  - Don't query the database everytime you need data
  - Use locally stored copy (file or memory)
- Caching solutions:
  - Check your framework for caching support
  - Memcached http://danga.com/memcached/
  - Varnish http://varnish.projects.linpro.no/

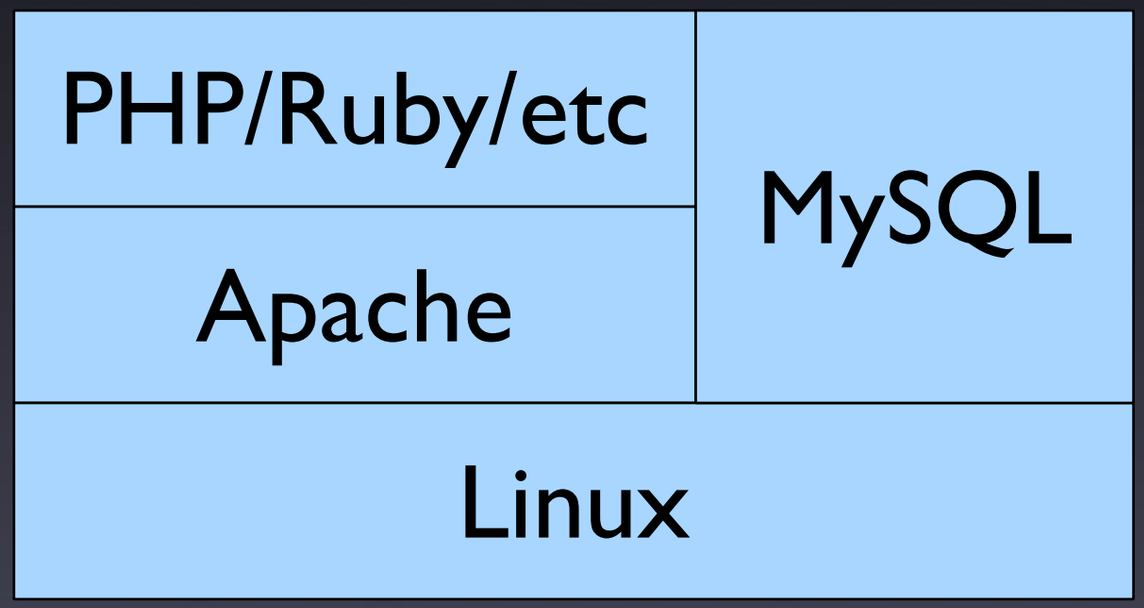
Step 9 - Deployment

- Lifecycle:
  - Develop
  - Test
  - Deploy to staging environment
  - Deploy to live

- Use SVN
- Code is just a part, don't forget the database
- Create a 'Deployment & Release Profile'

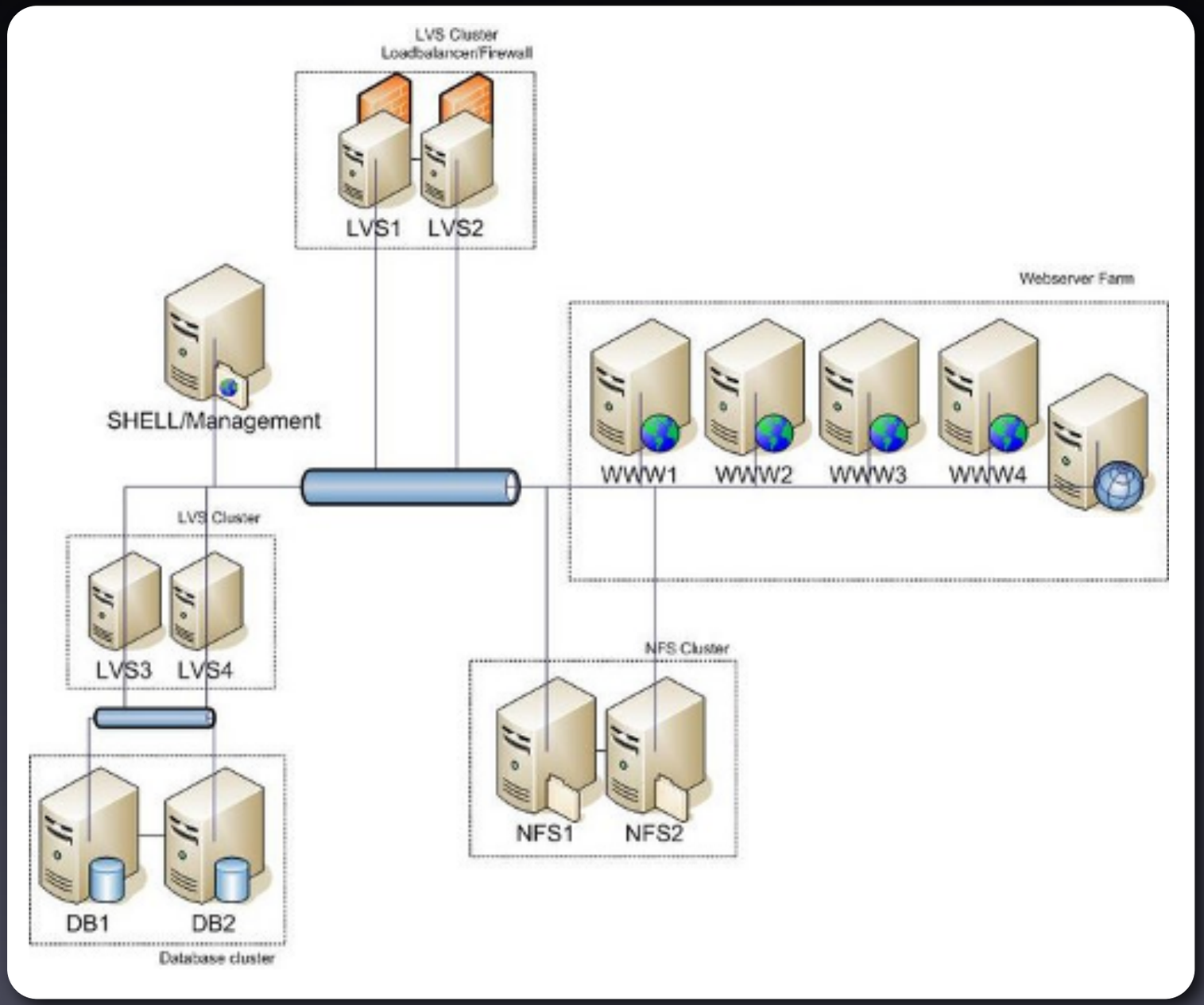
Step 9 - Deployment

System Architecture



From a simple single machine stack…

Step 9 - Deployment



… to a High Availability, Scalable Architecture

Step 10 - Operations

Monitoring:

- Logfiles
- Monitor infrastructure (tools such as Nagios)
- Monitor application (tools such as Zend Platform)
- Monitor business (is the money still pouring in?)

Step 10 - Operations

Debugging

- Ideal:

  “I had error x when I selected y after I clicked z”

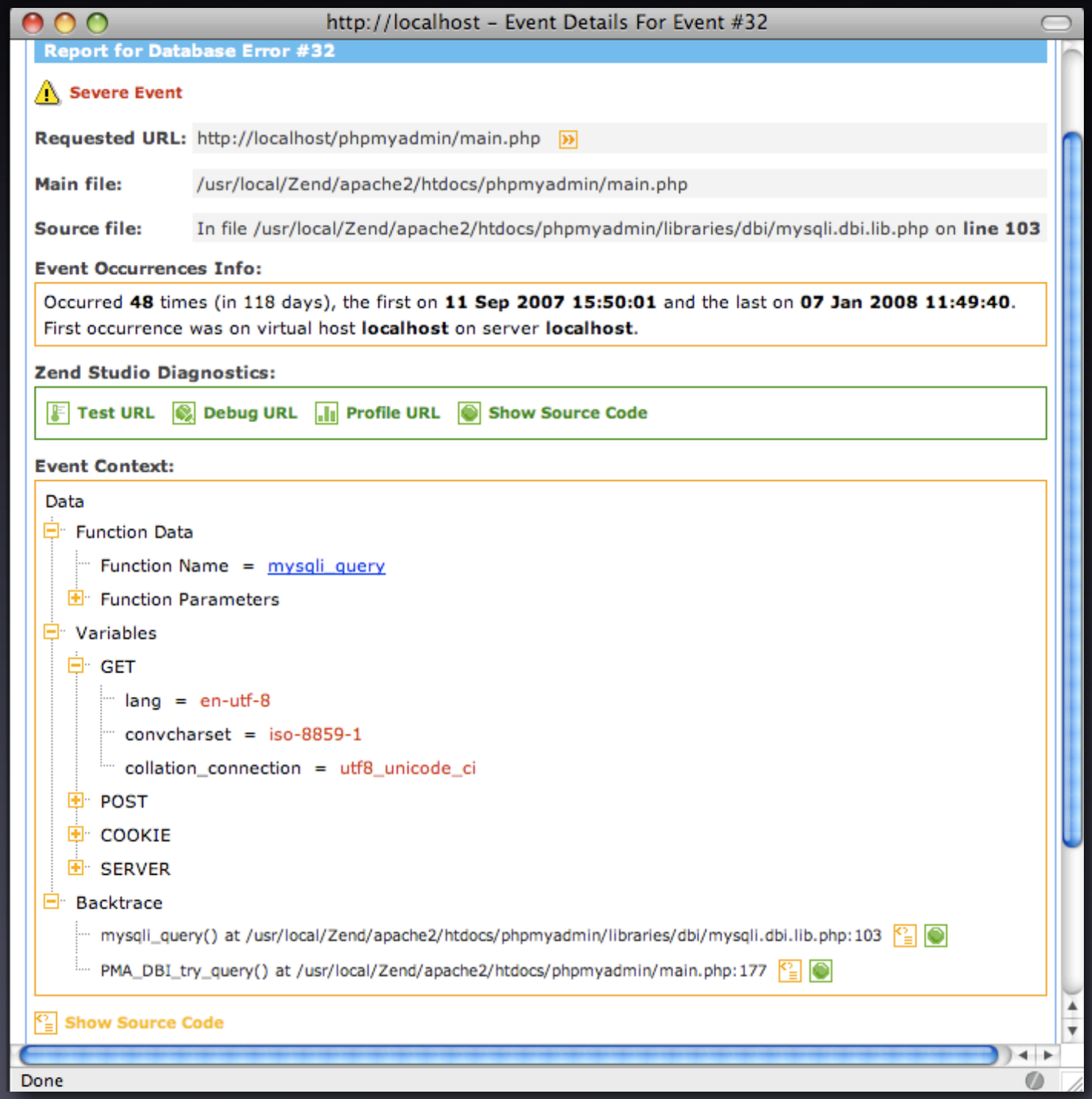
- Reality:

  “It doesn't work!”
  “What exactly doesn't work, and what did you do?”
  “It just ******** didn't work, FIX IT.”

Step 10 - Operations

Solutions:

- A 'root cause' analysis tool
- Logging
- Context Capture



Step 10 - Operations

Change management:

- Ticket system
- Stick to your deployment - use the DRP
- DON'T TOUCH THE LIVE ENVIRONMENT. EVAH!

The Big Picture

(We're almost done)

The Big Picture

- No ‘cowboy coding’, but structured steps
  - Higher quality
  - Software will be easier to maintain (and cheaper)
- Development methodology
  - Waterfall
  - Agile (DSDM, XP, Scrum)
- Software Development Life Cycle (SDLC)
  - Adapt from existing methods, learn from experts
  - Document it properly
  - Make it the cornerstone of your development efforts

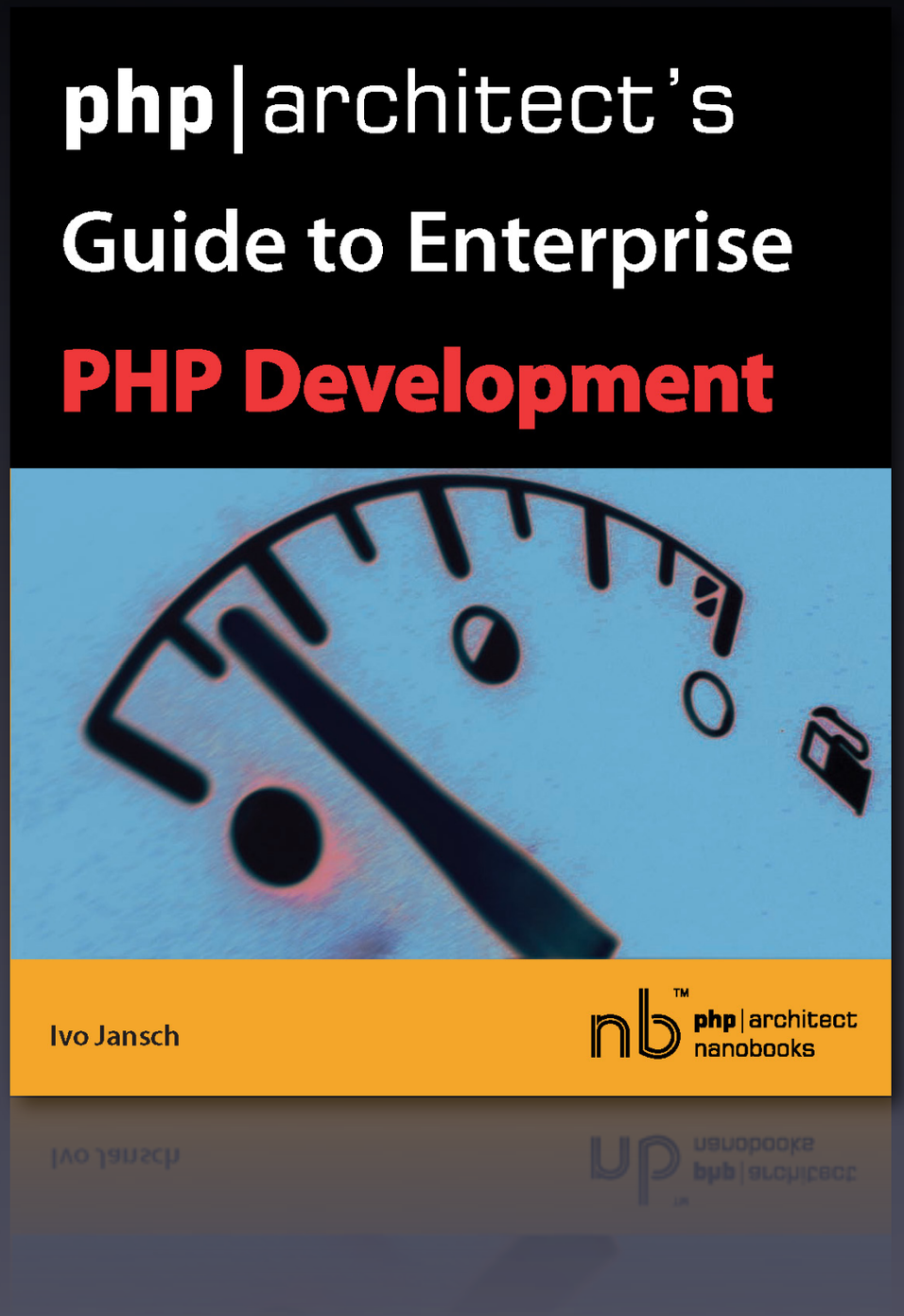
More things to think about

- Planning
- Coding
- Implementation
- Security

Shameless Plug

Book about this all:

- php|architect's Guide to Enterprise PHP Development
- ISBN: 978-0-9738621-8-8
- Order at phparch.com or amazon.com
- http://www.enterprisephp.nl



Questions?

ivo@ibuildings.com

http://www.ibuildings.com

http://www.jansch.nl

Twitter: @ijansch @ibuildings