eZ systems

Enterprise PHP development

- What does “enterprise” mean?

Share your information

eZ systems

Share your information

Speaker

- Kore Nordmann
  - Studies computer science at the University Dortmund
  - Working as a Software Developer for eZ systems on eZ components and eZ publish
  - Maintainer and Developer of Image_3d

eZ systems

Agenda

- Classification of “enterprise”
- Classification of “enterprise PHP”
- Scalable web applications in practice

Share your information

eZ systems

What does “enterprise” mean?

- Extensibility
- Integratebility
- Flexibility
- Scalability

Share your information

eZ systems

Share your information

What does “enterprise PHP” mean? (1/3)

- Extensibility
  - Often neglected topic in PHP
    - Bad example: PhpBB
  - Multiple approaches for modules in applications
    - Patches
    - Classes implementing interfaces or extending one class
    - RPC calls on external applications
  - No application independent module interfaces
    - Impossible and application specific

eZ systems

What does “enterprise PHP” mean? (2/3)

- Integratebility & Flexibility
  - Accessing foreign applications
  - Use open interface specifications
    - XML-RPC
    - SOAP
    - REST
  - PEAR and ZF offer lots of different implementations for webservices
    - Zend_Service_[Amazon|Yahoo]_*
    - PEAR::Services_*
  - Custom protocols
    - Payment_*
  - Application integration in existing services
    - LDAP (eZ publish, Horde)
    - SAP (multiple appications and interfaces)

eZ systems

What does “enterprise PHP” mean? (3/3)

- Scalability
  - Caching strategies
    - Content caching
    - Static caching
    - Template caching
    - Opcode caching
  - Database abstraction vs. optimization
    - For example: Oracle sometimes behaves completely different then MySQL (LIKE)
    - Denormalize, DataMining, Aggregation
  - Session management
    - Standard file handler vs. database handler
  - Content storage
    - SAN vs. Database

Share your information

eZ systems

Share your information

Content

- Remember during development
- Content caching
- Normalization
- Session handling
- Binary files
- Server setup

eZ systems

Share your information

1) What is important during development?

- Find performance bottlenecks
- Remember content and static caching from the first second of development
- Abstract each access on external ressources
  - Adds processing overhead
  - Enables to change real data source later
- Test your application under high load
  - ab (Apache benchmark)

eZ systems

Share your information

2) Content caching

- No easy solution
  - Highly application dependant
- Determine possible caching times for each part of your content
- Under high load 30s are a very long timespan
- A lot customers don't know, that 5 minutes are “immediately” enough.
  - News
  - Statistics
- Use flexible caching classes
- Try to always export static content as static files

eZ systems

Share your information

3) Normalization

- Always use normalized databases
- Aggregate data in non normalized tables
  - Forum
    - Fame points
    - Post statistics
  - Most popular $xyz

eZ systems

Share your information

4) Session handling

- Store session data in database or filesystem
- File access is always faster than database access
  - Databases are a filesystem on top of a filesystem
    - But, Oracle uses its own partitions
  - Databases are accessed through sockets <> Filesystems are accessed by kernel calls
    - Both can be in memory
- Session in database always requires connect to database

eZ systems

Share your information

5) Binary files (1/2)

- Do NOT wrap fileaccess by scripting languages (like PHP)
  - Processing and memory overhead
  - Webservers are build to deliver large files
  - Accesscontrol needs to be delegated to webserver modules
    - Lighttpd: mod_secdownload, mod_trigger_b4_dl
    - HTTP-Auth against LDAP, ...
- For highly application authorization delegate file sendout to webserver
  - Frees the process / thread and memory
  - Lighttpd / Apache: mod_sendfile

eZ systems

Share your information

5) Binary files (2/2)

- SANs are expensive
  - Open standards
  - Expensive servers
- NFS is slow and full of locks
  - Hard to debug
- Databases are sometimes usefull to store binary files
  - Easy clustering
  - Optimized for networking
  - Overhead on webserver side

eZ systems

Share your information

6) Server setup

- Use a fast, tiny webserver
  - lighttpd
- Use fastcgi
  - lighttpd + PHP-fcgi is about two times faster than apache + mod_php
  - Easy user jailing
  - Clustering of fcgi processes
    - Connect to fcgi-Instances per socket
    - fcgi-Instances may be on external servers
    - Native clustering
- Use opcode caches
  - APC, xcache, eaccelerator
- Remove includes if possible

eZ systems

Share your information

Emergency

- Put a proxy in front of your webserver
  - Squid (Wikipedia, ...)
- Akamai
- `wget` your site and put static files on a static webserver

eZ systems

Share your information

Conclusion

- Design your application with extensibility in mind on each level of devopment
- Ressources
  - http://php.net/
  - http://lighttpd.net/
  - http://pecl.php.net/package/APC
  - http://www.squid-cache.org/
  - http://kore-nordmann.de/
  - http://talks.php.net/show/ezkey06
- Thanks for listening