

Firebird and PHPServer

November 2006

Topics

- What is LAMP?
- A short history of LAMP and AJAX
- PHP and AJAX in action
- LAMP and Firebird: what can you do with it?
- What's next?

What is LAMP?

- **Definition 1: a stack of four software components**

- **L** inux
- **A** pache
- **M** ySQL
- **P** HP

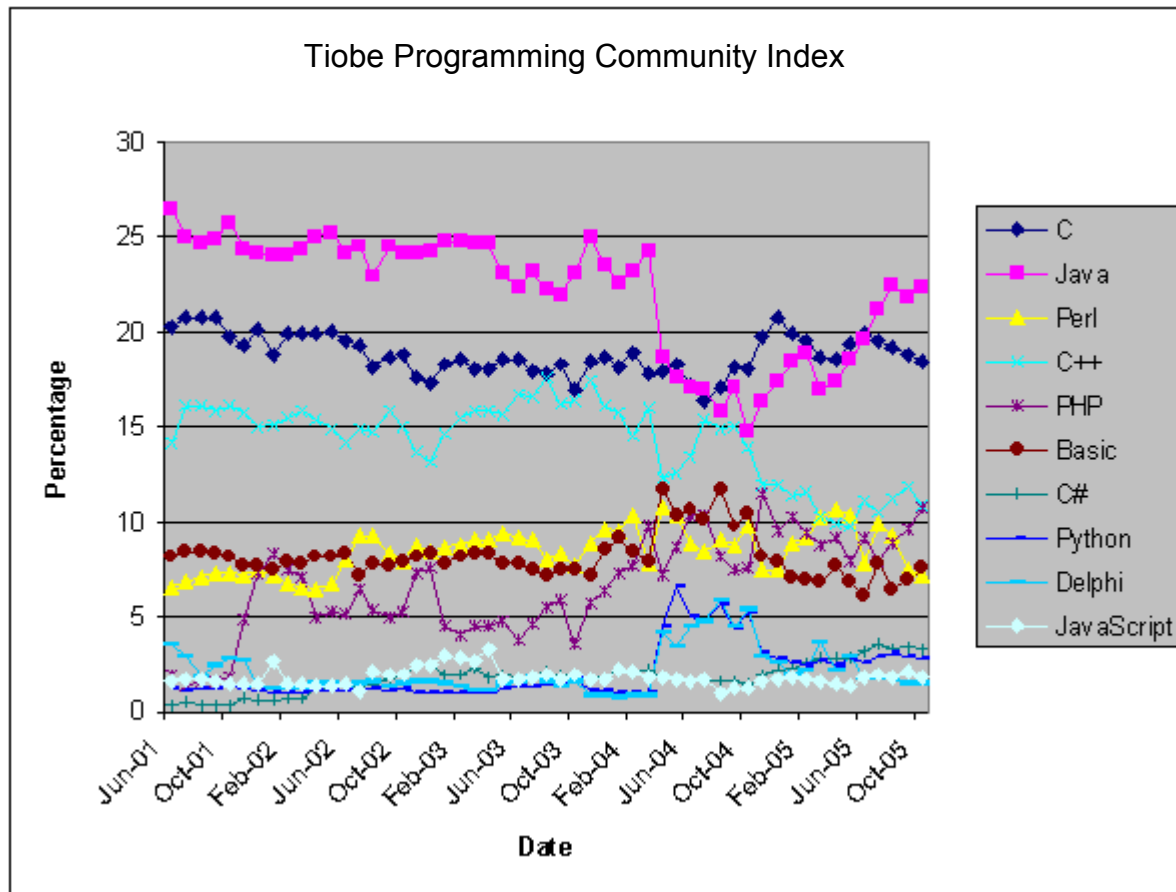
- **Definition 2: an application server based on**

- An HTTP server
- Server-side embedded scripting
- A relational database

- **And for LAMP/AJAX: an application server based on**

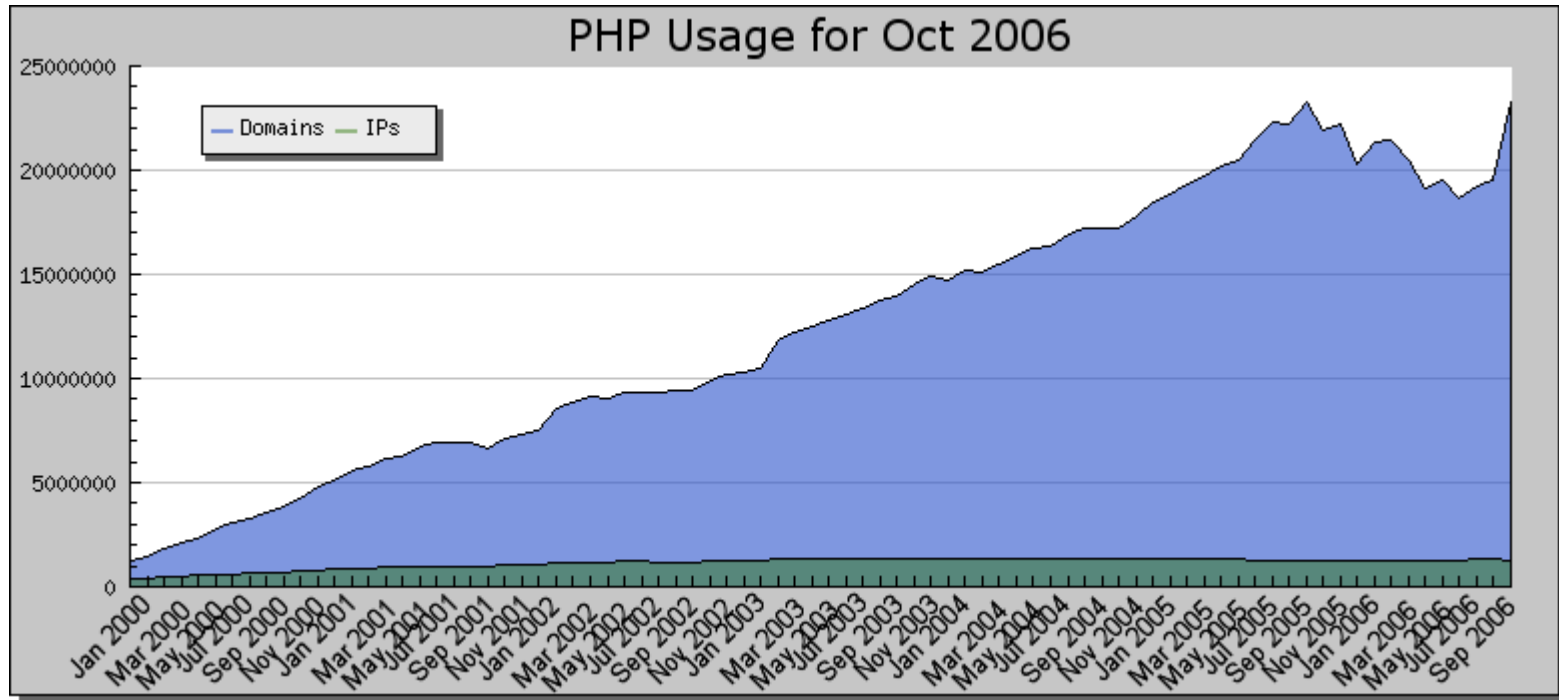
- A browser with client-side scripting
- An HTTP server
- Server-side (embedded) scripting
- A relational database

Why is PHPServer important for Firebird?



- Estimated developer community sizes:
 - C/C++: 5 mln
 - Java: 5 mln
 - Delphi/VB: 4 mln
 - PHP: 3 mln
 - C#/.Net 1 mln
- According to Netcraft, PHP is being used at:
 - 22.4 mln domains originating from
 - 1.3 mln IP addresses
- IT Industry recognizes PHP as an important technology
 - IBM
 - Oracle
 - Yahoo
 - Etc.

Why is PHPServer important for Firebird?

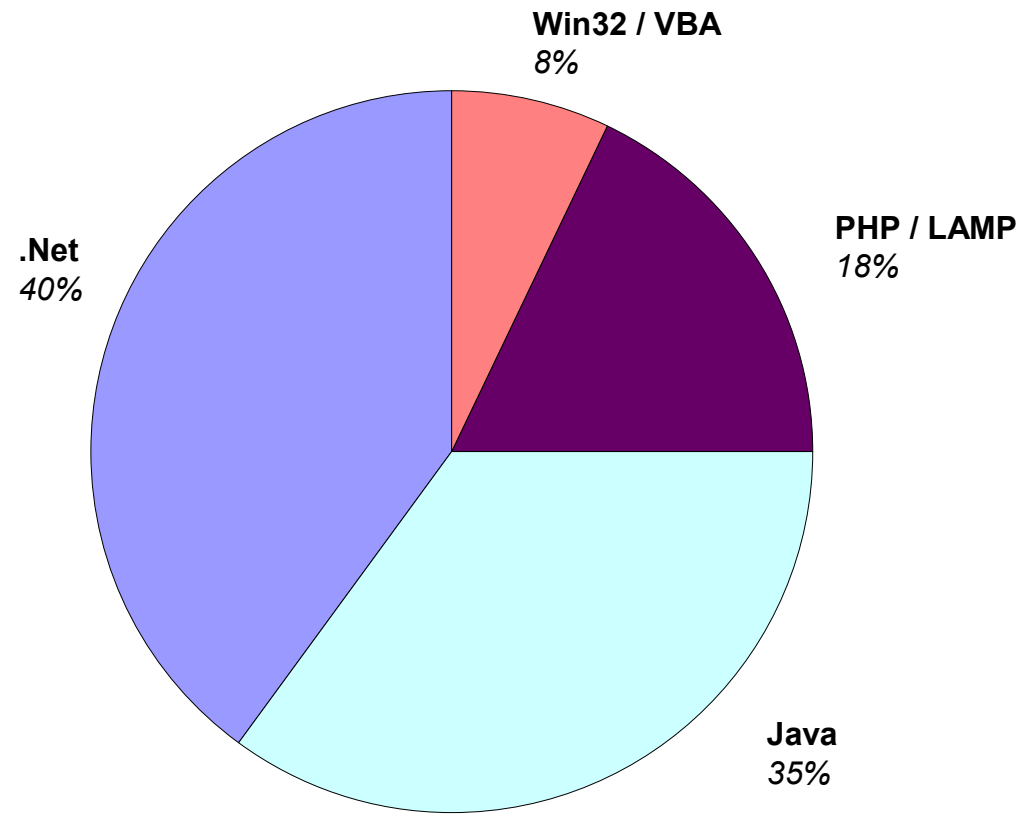


For comparison (Summer '04):

- PHP had 1.2 mln IP's
- ColdFusion had 80K IP's
- ASP.NET had 40K IP's

Why is PHPServer important for Firebird?

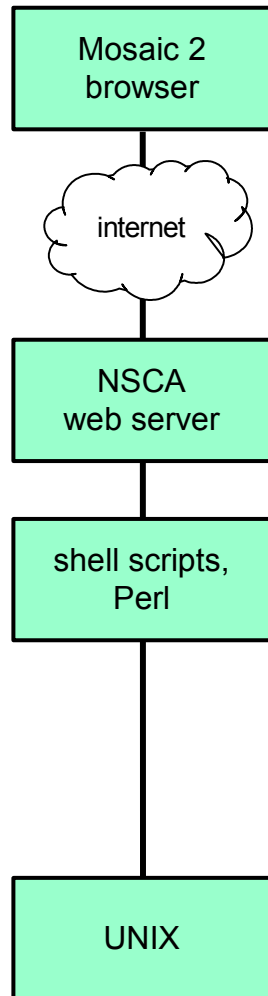
Technologies used to access Salesforce.com
(% of API access to the system)



Topics

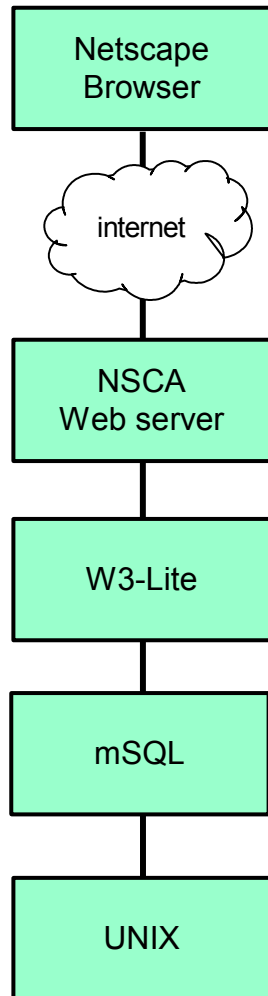
- What is LAMP?
- A short history of LAMP and AJAX
- PHP and AJAX in action
- LAMP and Firebird: what can you do with it?
- What's next?

1992..4: The basics evolve



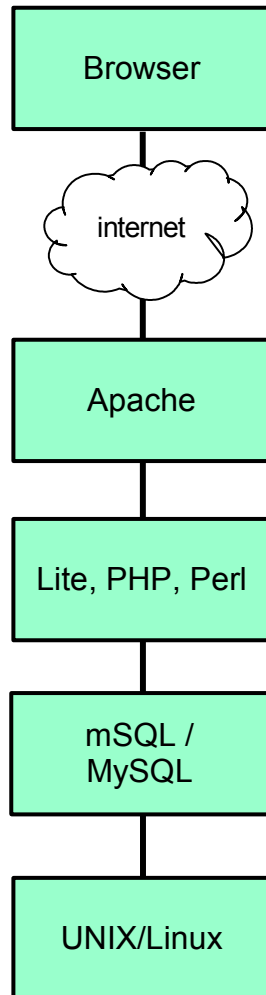
- The HTTP protocol is extended with the POST method
- The HTML language is extended with the <FORM ..> tag
- The web server is extended with the CGI api specification
- Typical CGI programs are Unix shell scripts or Perl scripts
- No database typically used!
- First e-Commerce application appears in the first months of 1995

1995..96: David Hughes invents LAMP



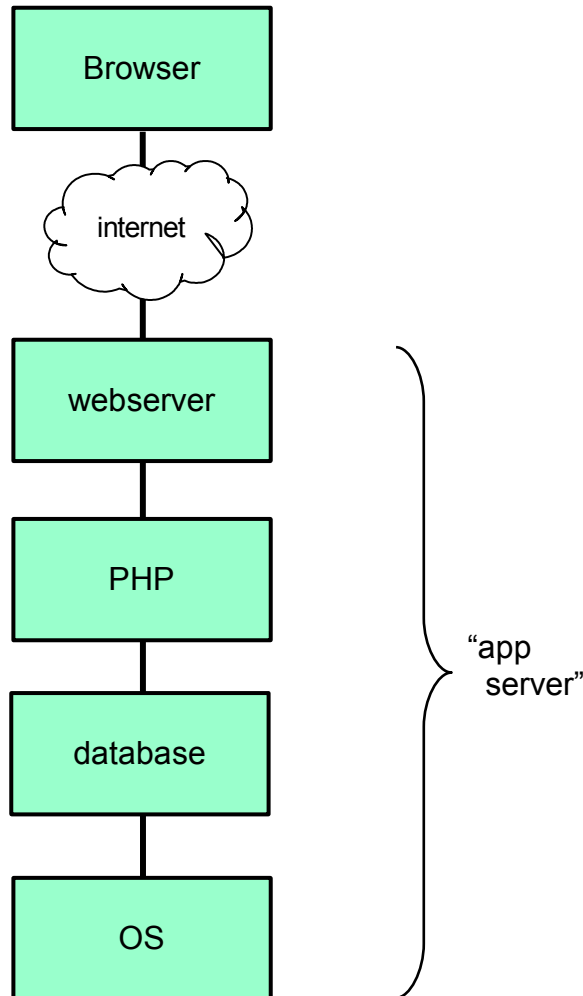
- Davis has an itch to scratch: a universal client for the Athena network monitoring software
- First uses Postgres95 with a SQL-to-QUEL preprocessor; this solution is too 'heavy'
- Then he writes a simple back-end to the preprocessor, thus creating mSQL, a mini SQL server
- Invents the embedded 'Lite' language to easily generate html in response to requests
- Releases the code under a restrictive open source license in April 1996
- mSQL quickly gains traction as a lightweight database in the emerging Linux world

1997..8: MySQL, PHP/Perl and “LAMP”



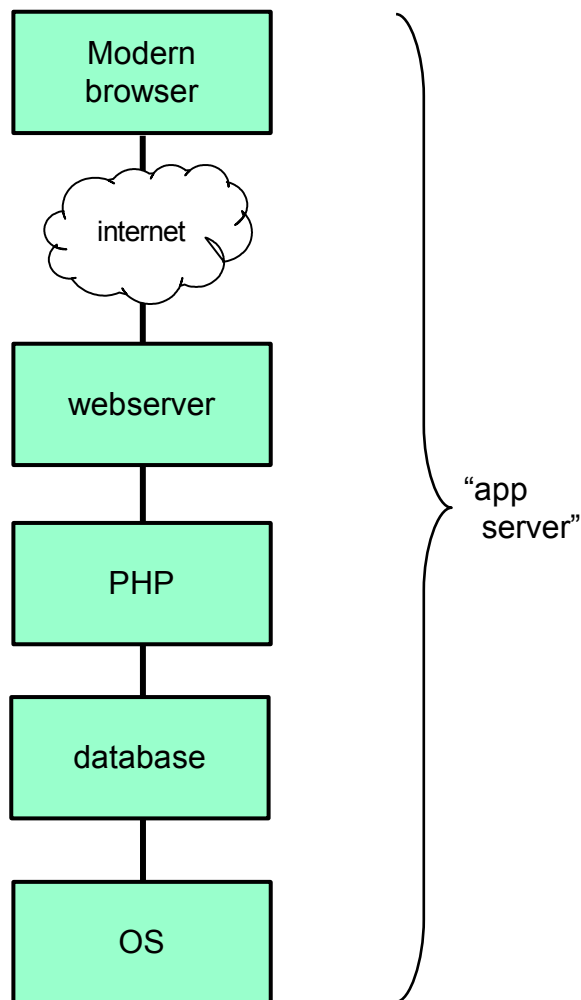
- Monty Widenius is a mSQL user but finds performance lacking (mSQL did not support indexes)
- MySQL is created: an API and SQL compatible clone of mSQL, with support for indexes, otherwise just as basic; license LGPL
- PHP starts as a set of Perl scripts in 1995; it remains a one man project for several years. PHP version 3 combines the best elements of Perl and Lite.
- In the summer of 1998, c't journalist Michael Kunze is the first to use the acronym “LAMP”
- LAMP becomes a popular technology in Europe
- Linux becomes a popular platform for running web servers, mainly Apache

1999..2004: LAMP becomes mainstream



- Tech publisher O'Reilly notices the popularity of LAMP in Europe and starts to promote the concept in the US; ONLamp website started
- PHP gets rewritten twice (PHP4, PHP5) and becomes the most popular scripting choice
- LAMP programming is much easier/faster than e.g. J2EE or ASP.Net; Popularity skyrockets
- MySQL changes its license policy. The default database of PHP becomes SQLite
- Oracle and IBM start supporting PHP as part of their database offerings
- Virtually every OS supports LAMP: *nix, windows, OS/400, etc.
- A large pool of open source PHP applications is developed

2005: client side processing, AJAX

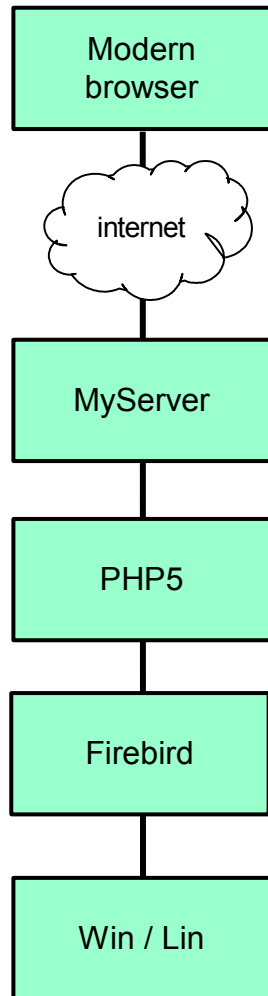


- After many years of work, browsers are becoming more standards compliant and powerful
 - Fast JavaScript
 - Cascading Style Sheets
 - Document Object Model
 - Asynchronous HTTP requests
- Google proves that “good enough” user interfaces can be made using modern browsers (Google maps, Gmail)
- “AJAX” becomes next buzzword
- But...
 - with increasing power, some of the simplicity that made LAMP attractive is unfortunately lost
 - IE6 still has relatively slow JavaScript processing

Topics

- What is LAMP?
- A short history of LAMP and AJAX
- PHP and AJAX in action
- LAMP and Firebird: what can you do with it?
- What's next?

PHPServer: a Firebird PHP application server



- PHPServer is a PHP-based application server for Firebird
- Entirely open source
- Capable, compact, easy, free; Works 'out-of-the-box'
- 4-click GUI install on both Linux and Windows; identical GUI web server management on both Linux and Windows
- Advanced webserver-PHP interaction:
 - FastCGI call interface
 - Webserver manages PHP server
 - Persistent database connections
 - Persistent sessions

Serving a basic page

File: "step1.html"

```
<html>
<head>
<title>Step 1</title>
</head>
<body>
  <table align="center">
    <tr bgcolor="#0000ff">
      <td width="800">Header</td></tr>
    <tr bgcolor="#cccc33"><td>Row 1</td></tr>
    <tr bgcolor="#cccc33"><td>Row 2</td></tr>
    <tr bgcolor="#cccc33"><td>Row 3</td></tr>
  </table>
</body>
</html>
```

Output

Header

Row 1

Row 2

Row 3

Adding some PHP scripting

File: "step2.php"

```
<html>
<head>
<title>Step 2</title>
</head>
<body>
  <table align="center">
    <tr bgcolor="#0000ff">
      <td width="800">Header</td></tr>
    <? for( $i=1; $i<4; $i++ ) {
      echo '<tr bgcolor="#cccc33">';
      echo '<td>Row '.$i.'</td></tr>';
    }
    ?>
  </table>
</body>
</html>
```

Output

Header

Row 1

Row 2

Row 3

Accessing Firebird from PHP

- PHPServer has Firebird driver pre-configured: no need to install separately. PHPServer does not install Firebird itself.
- Simple API
 - `fbird_connect`: opens a connection to the database
 - `fbird_query`: executes a query, returning a result set object
 - `fbird_fetch_row`: fetch the current row, returning array of fields
 - `fbird_free_result`: clean up the result set object
 - `fbird_close`: close the connection
- Possibility to keep database connection open between successive page loads
 - Faster response
 - Reduce load on server
 - Use '`fbird_pconnect`' instead of '`fbird_connect`' to make connection persistent

Accessing Firebird from PHP

File: "step3.php"

```
<html>
<head>
<title>Step 3</title>
</head>
<body>
  <table align="center">
    <tr bgcolor="#0000ff">
      <td width="800">Header</td></tr>
      <? $db = fbird_connect("localhost:c:\phpdemo.fdb",
        "sysdba","masterkey");
        $rs = fbird_query($db, "select * from demo");
        while( $rw = fbird_fetch_row($rs) ) {
          echo '<tr bgcolor="#cccc33">';
          echo '<td>'.$rw[0].</td></tr>';
        }
        fbird_free_result($rs);
        fbird_close($db);
      ?>
    </table>
  </body>
</html>
```

Output

Header
Result row a
Result row b
Result row c

Adding some AJAX

- Modern browsers contain the XMLHttpRequest object, which makes it possible to request data from the server without reloading the main page

- Simple API
 - open: set the request URL and method (get, post)
 - onreadystatechange: set the callback function to process answer
 - send: send the request to the server
 - readyState: get the status of the request (pending, ready, error, etc.)
 - responseText: fetch the result

- Typically, the result is assigned to an object already contained on the main page
 - <div name=*id*> tag often appropriate
 - Use JavaScript “DOM” to get access to the tag object

Adding some AJAX

File: "step4.php"

```
<html>
<head>
<title>Step 4</title>
<script language="javascript" type="text/javascript"
src="./ajaxdemo.js"></script>
</head>
<body>
  <table align="center">
    <tr bgcolor="#0000ff">
      <td width="800">
        <form name="form_select">
          <select name="order_select"
            onChange="getItems();">
            <option>Ascending</option>
            <option>Descending</option>
          </select>
        </form>
      </td></tr>
      <tr><td>
        <div id="result_area">Select ordering...</div>
      </td></tr>
    </table>
  </body>
</html>
```

Output



Ascending

Row 1

Row 2

Row 3

Adding some AJAX – ajaxdemo.js

File: “ajaxdemo.js”

```
var http = createRequestObject();

/* Function called to get the product categories list */
function getItems() {
    http.open('get', 'worker.php?action=get_rows&id=' + document.form_select.order_select.selectedIndex);
    http.onreadystatechange = handler_1;
    http.send(null);
}

/* Function called to handle the list that was returned from the worker.php file.. */
function handler_1(){
    if(http.readyState == 4) { //Finished loading the response
        var response = http.responseText;
        document.getElementById('result_area').innerHTML = response;
    }
}
```

Adding some AJAX – worker.php

File: “worker.php”

```
<?
if( $_GET['action'] == 'get_rows' ) {
    switch( $_GET['id'] ) {
        case 0: // Ascending
            echo ' <table>
                <tr bgcolor="#cccc33"><td>Row 1</td></tr>
                <tr bgcolor="#cccc33"><td>Row 2</td></tr>
                <tr bgcolor="#cccc33"><td>Row 3</td></tr>
                </table>';
            break;
        case 1: // Decending
            echo ' <table>
                <tr bgcolor="#cccc33"><td>Row 3</td></tr>
                <tr bgcolor="#cccc33"><td>Row 2</td></tr>
                <tr bgcolor="#cccc33"><td>Row 1</td></tr>
                </table>';
            break;
        default:
            echo '<b>You didn\'t select an item from above!</b>';
            break;
    }
}
?>
```

Combining it all

- Use HTML/CSS to generate the basic page layouts
- Use embedded PHP to customize pages or to generate dynamic content
- Use JavaScript to enhance the user interface, making it responsive
 - Visual feedback, tool tips, etc.
 - Basic input validation
 - Partial screen updates
- Use PHP “worker pages” to respond to AJAX requests from the browser
 - Deep input validation
 - Data-driven screen updates
- Example 5 is the same as example 4, with only 'worker.php' changed

Adding some AJAX – a db driven worker

File: "worker2.php"

```
<?php
if($_GET['action'] == 'get_rows') {

    if($_GET['id']==0) {
        $qry = "select id from demo order by 1 ascending";
    } else {
        $qry = "select id from demo order by 1 descending";
    }

    $db = fbird_pconnect("localhost:c:\phpdemo.fdb", "sysdba","masterkey");
    $rs = fbird_query($db, $qry);

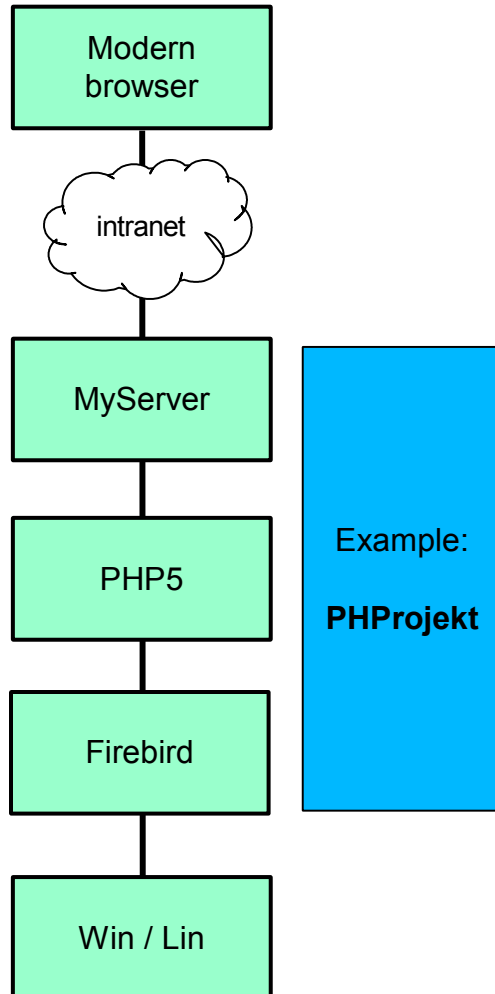
    echo '<table>';
    while( $rw = fbird_fetch_row($rs) ) {
        echo '<tr bgcolor="#cccc33">';
        echo '<td>'.$rw[0].'/td></tr>';
    }
    echo '</table>';

    fbird_free_result($rs);
    fbird_close($db);
}
?>
```


Topics

- What is LAMP?
- A short history of LAMP and AJAX
- PHP and AJAX in action
- LAMP and Firebird: what can you do with it?
- What's next?

Opportunity 1: use for a “web application”



- PHPServer can be used to run any PHP application. Many good quality applications are available as open source:
- Some examples are:
 - SugarCRM
 - PHPProjekt
 - phpBB
 - Hundreds more!
- Sometimes historical dependence on MySQL, but increasing shift to independence or support for multiple databases:
 - Firebird
 - Oracle -> Fyracle
- Or you can code your own. A few pages of PHP is often enough for a simple application

Example: PHProjekt

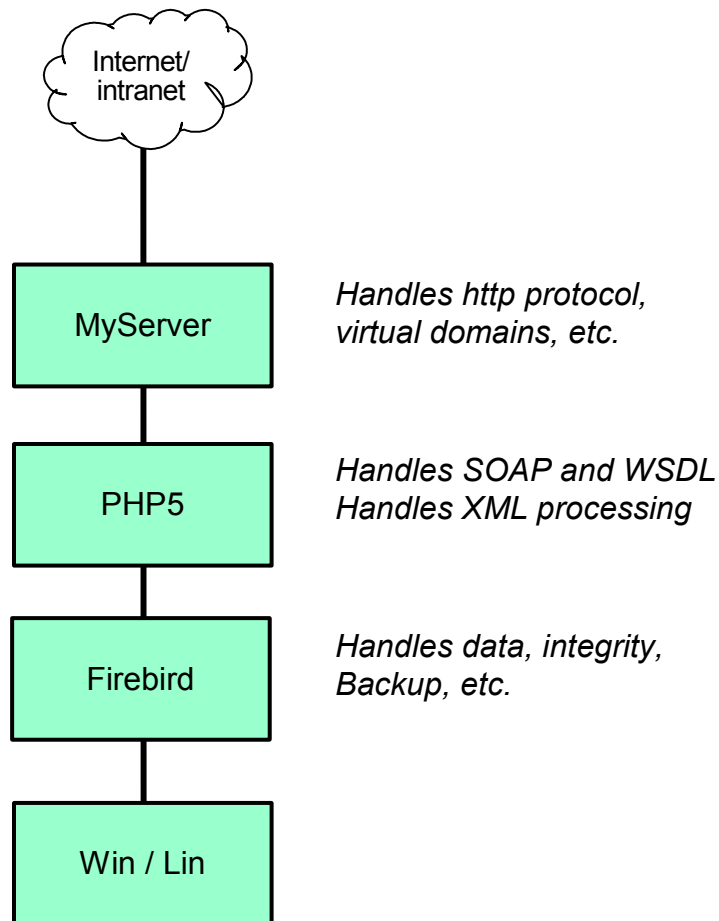
- Large corporations use elaborate groupware systems. The two leading choices are:
 - Microsoft (Exchange, Outlook): 150 mln 'seats'
 - IBM (Notes): 120 mln 'seats'Both are nice, but expensive

- PHProjekt is a good alternative for medium sized organisations
 - Already used in hundreds of organisations
 - Typical deployment 20..50 'seats', ranging up to hundreds
 - Used by the city of Munich

- PHProjekt supports Firebird 'out-of-the-box'

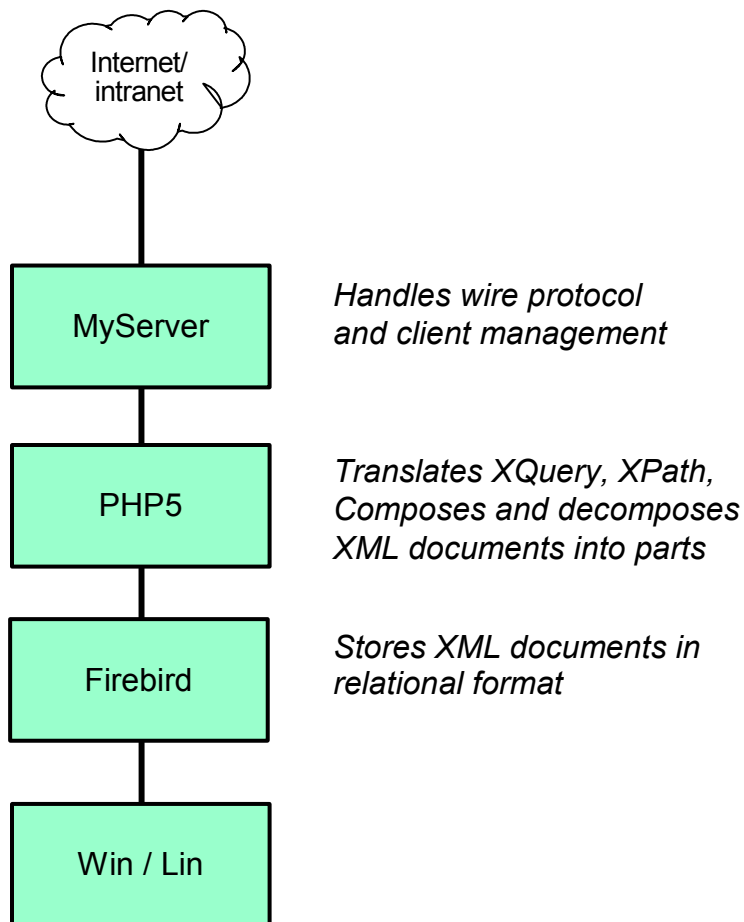


Opportunity 2: use as web service app server



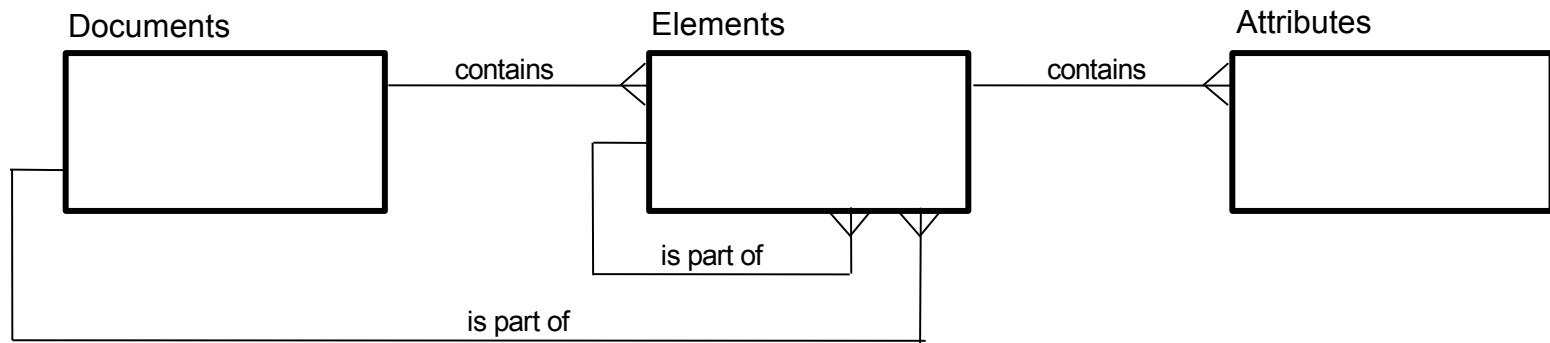
- Currently “web services” and “server oriented architectures (SOA)” are big buzzwords
- Breaking up software systems into modules with specific functionality, which can be locally or remotely accessed not a bad idea
- Currently promoted implementation style is SOAP and WDSL, but XML-RPC remains popular
 - SOAP & WDSL promoted by “big IT”
 - XML-RPC simpler, effective technology
- PHPServer is a good basis to build your web services and SOA's
 - PHP5 has native support for SOAP/WDSL
 - Safe, efficient & easy to manage Firebird data store

Opportunity 3: use as “XML database”



- PHPServer can be used to provide XML database type functionality to Firebird
- A small PHP program can be used as a pre-processor to handle XQuery and XPath requests.
 - This may sound strange, but the first implementations of dynamic SQL for Firebird worked just the same back in the late 80's
 - PHP has good XML processing libraries
- XML documents can be decomposed into documents, elements and attributes
 - Any XML document can be stored this way in a fixed small set of tables
 - Original document can be stored in separate blob for legal purposes (keep MD5 sum)
- Firebird Fyracle's hierarchical query capability handy for retrieving nested element sets in a single pass

XML-Relational mapping to 3 core tables

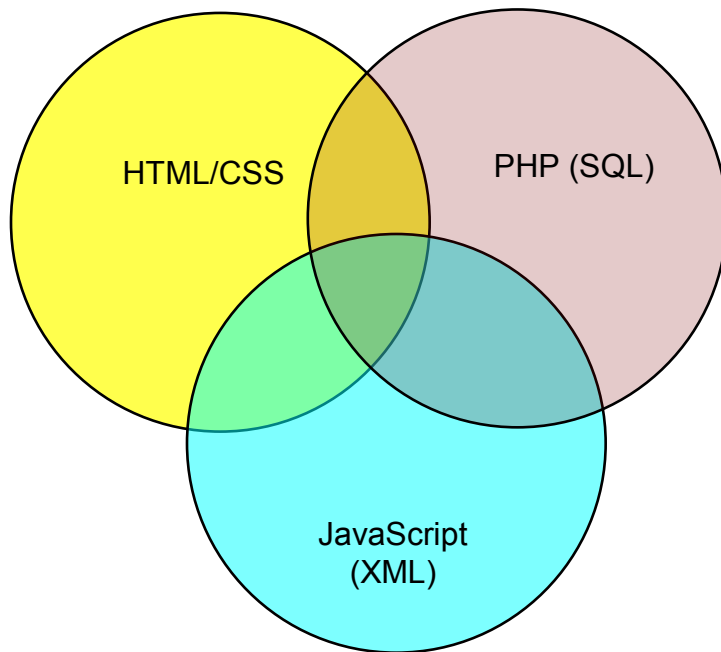


Use Fyracle or Firebird 3
hierarchical queries to
efficiently retrieve nested
element sets

Topics

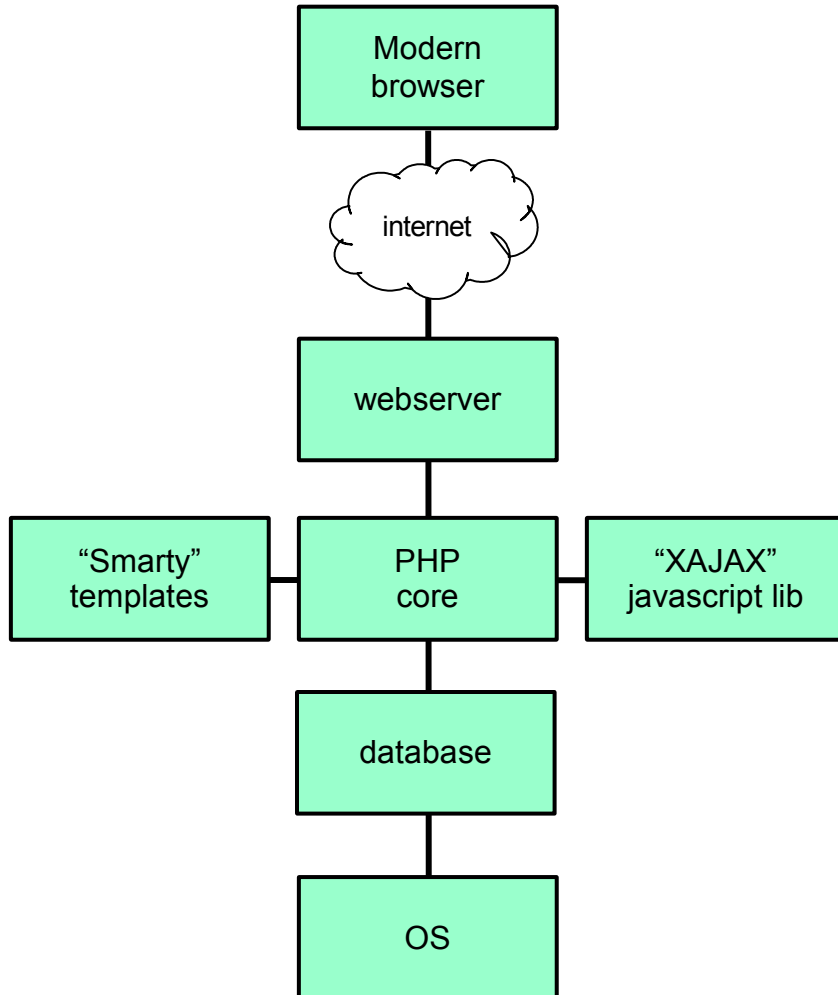
- What is LAMP?
- A short history of LAMP and AJAX
- PHP and AJAX in action
- LAMP and Firebird: what can you do with it?
- What's next?

What is the problem of LAMP/AJAX?



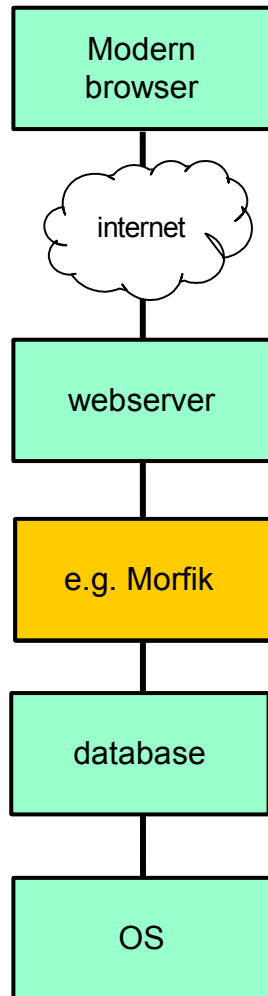
- The LAMP model is not as easy as it once was and combines several languages/technologies:
 - HTML/CSS
 - PHP (often containing SQL)
 - JavaScript (often handling XML data)
- Lots of scope for unmaintainable code:
 - Mixed code snippets
 - Unrelated snippets in a single file
 - One bit of functionality spread out over several files
- Requires experienced developers to get right

Solution 1: clean up the mess



- In the last year a few libraries have become popular that separate the various elements into maintainable units
- The HTML and PHP code can be separated more by using “templating engines”. A popular choice is “Smarty”.
- The JavaScript code, especially the AJAX part, can be moved into a framework library. A popular choice is XAJAX.
- Raw PHP is used for worker pages. The object orientation of PHP5 enables proper layering of this code.
- Using this route makes the code more maintainable, but...
- ...the demands on the experience of the developer are even bigger, with two more frameworks to learn

Solution 2: swap out PHP



- In the last 10 years nearly every component in David Hughes' original LAMP-stack got changed
 - From Unix to Linux to any OS
 - From mSQL to MySQL to any database
 - From Mosaic to Apache to any webserver
 - From Mosaic to Netscape to any browser
 - From Lite to Perl to PHP to ...?
 - Morfik is an IDE and a compiler for the LAMP application server:
 - GUI design of user interfaces
 - One development language for all code (pick from Delphi, VB, C# and Java)
 - Compiler generates
 - HTML/CSS
 - JavaScript
 - “worker pages”
- from this input, handling all complexity