Allgemeines

- Name:
  - Ursprünglich: „**P**ersonal **H**ome **P**age Tools“
  - Heute: **H**ypertext **P**reprocessor
- Scriptsprache zur Erstellung dynamischer Webseiten
- Syntax an C bzw C++ angelehnt
- PHP wurde 1995 von Rasmus Lerdorf entwickelt
- Ursprünglich Sammlung von Perl-Skripten
- Aktuelle Version: 5.2.6 (bzw. 4.4.8)

Vorteile

- Frei Verfügbar
- Plattformunabhängig / Browserunabhängig
- XAMPP/LAMPP: vorkonfigurierte Installationen
- Umfangreiche Datenbankunterstützung (MySQL, MSSQL, PostgreSQL, ...)
- Großer Bibliotheksumfang (Bilder, PDF, Flash, ...)
- Leicht zu erlernen
- Ausführliche Dokumentation

Vorteile PHP gegenüber ASP.NET

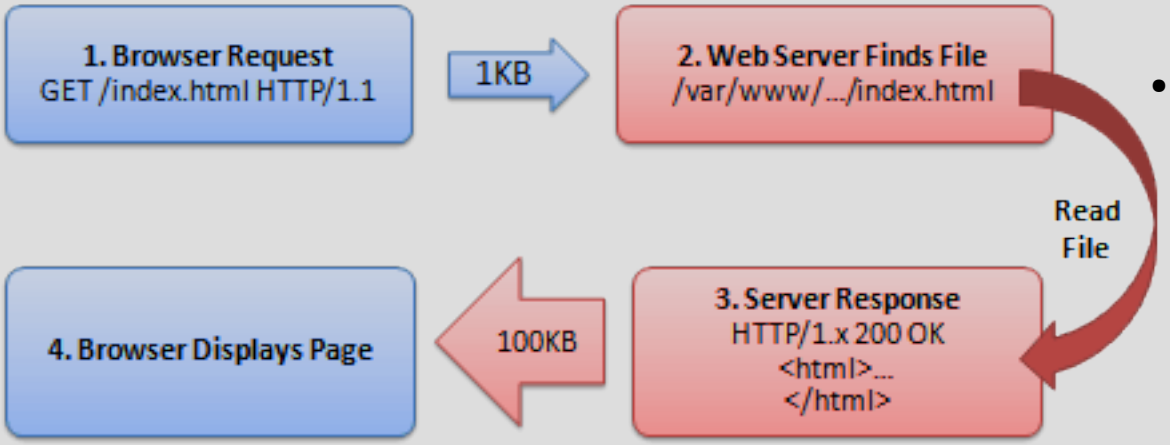
- **Geschwindigkeit:**
  - ASP: COM-basiert → overhead; mehr Speicherbedarf
  - PHP: Alles im PHP-Speicherbereich
- **Preis**
  - ASP: Windows für IIS benötigt, häufig MSSQL
  - PHP: läuft unter Linux, Apache, MySQL
- **Cross Platform Kompatibilität**
  - ASP: auf Windows (IIS) beschränkt
  - PHP: läuft unter Windows, Linux, Unix, Solaris
- **Keine direkte Möglichkeit in ASP Dateien hochzuladen, Mails zu versenden, ...**
- **Häufige Aufgaben wie FTP, MD5, eMail, ... direkt in PHP enthalten**

Ablauf

```
<html>
 <head>
 <title>Hallo-Welt-Beispiel</title>
 </head>
 <body>
 <?php echo "Hallo Welt!"; ?>
 </body>
</html>
```

Ablauf – GZIP mit PHP

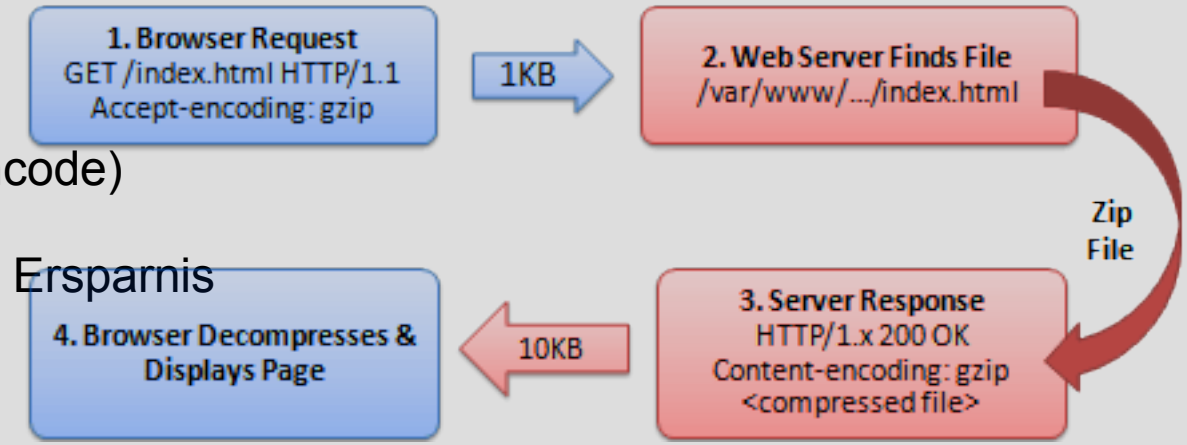
HTTP Request and Response



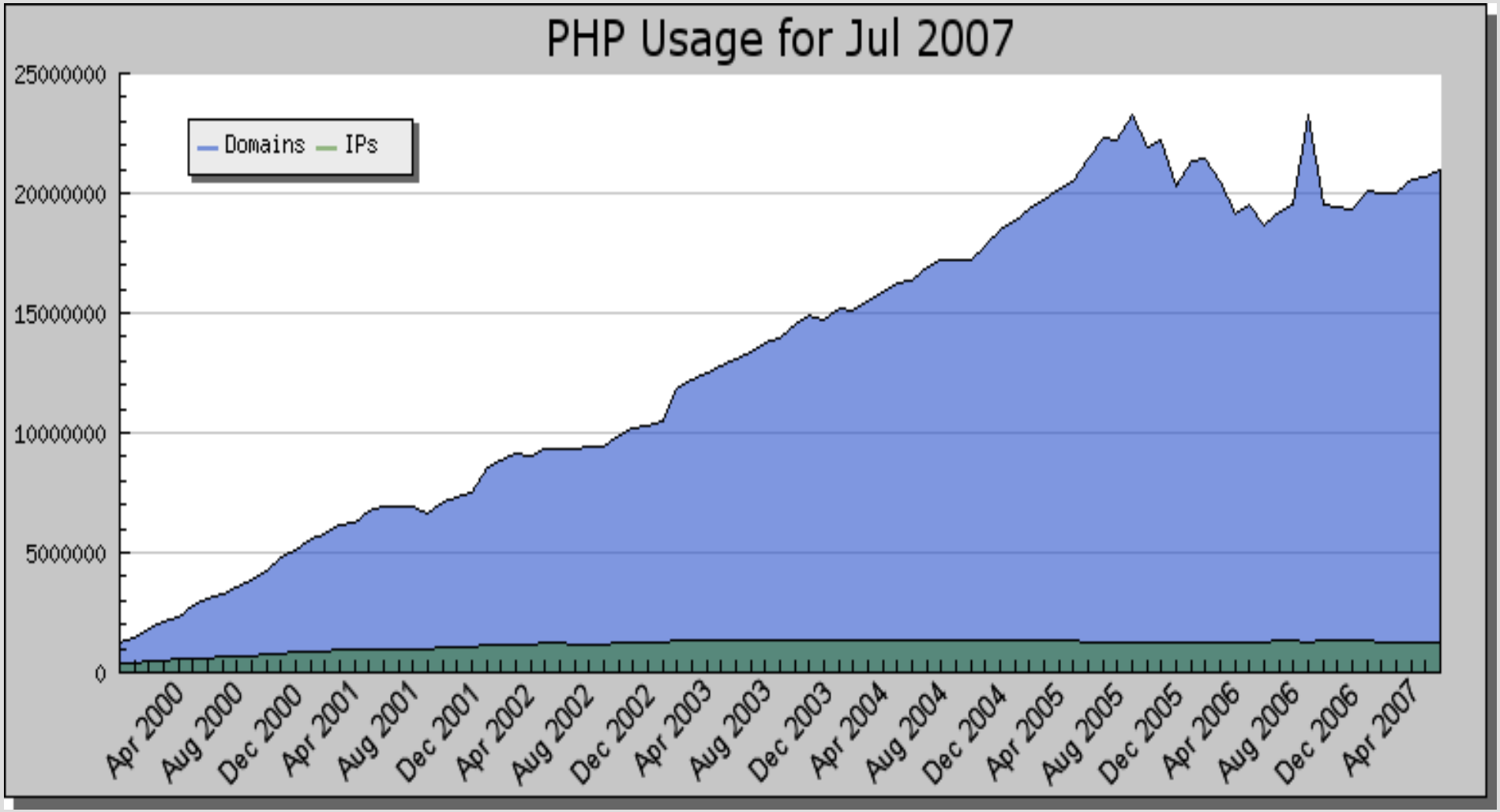
- reiner Plain-Text wird erzeugt
- unkomprimierter Versand

Compressed HTTP Response

- reiner Plain-Text wird erzeugt
- in PHP einfach zu realisieren (gzip_encode)
- komprimierter Versand mit bis zu 90% Ersparnis



Verbreitung



Beispiel

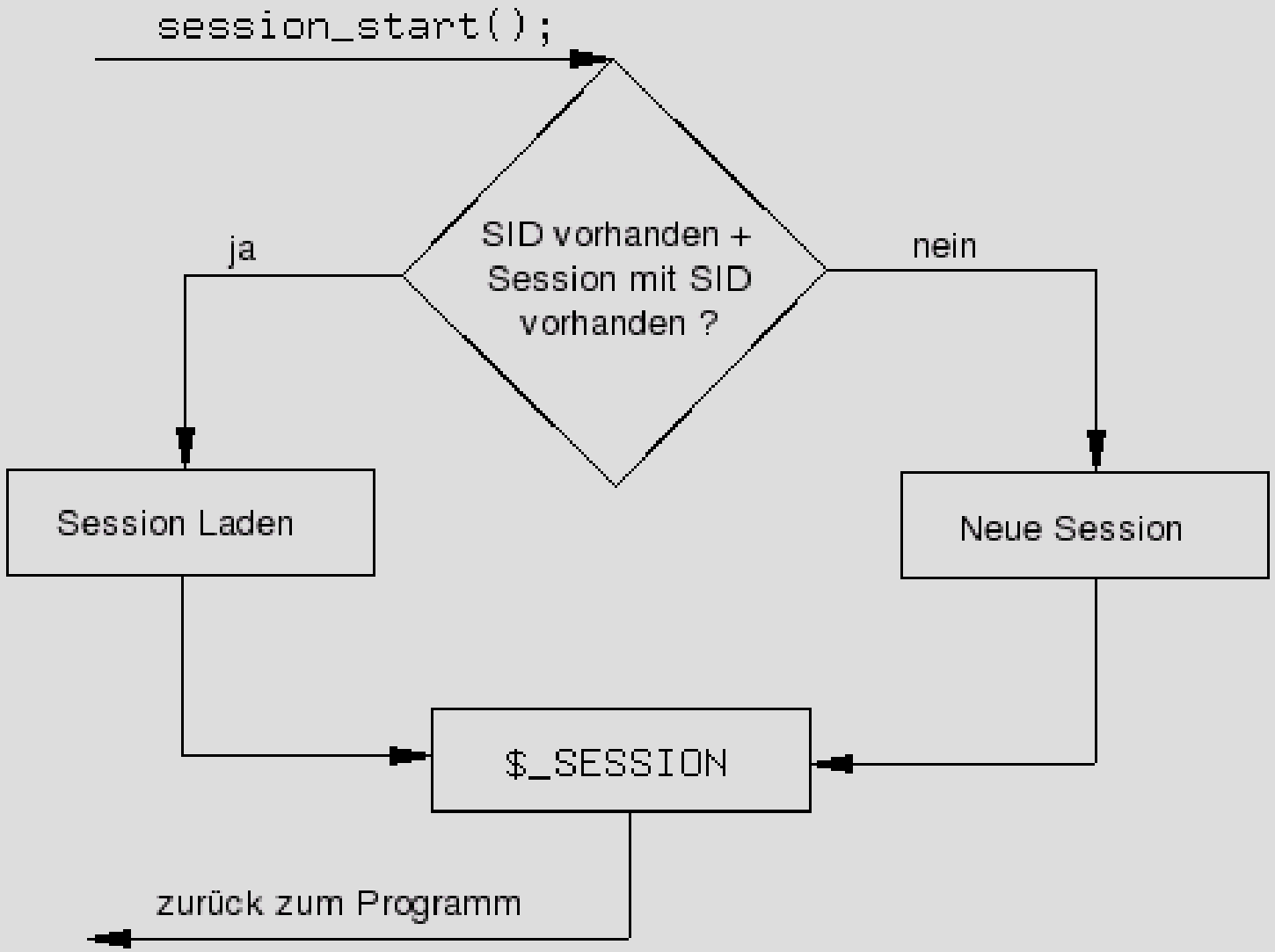
```
<?php
   if(isset($_POST['submit'])) {

      echo "Die Eingabe war: ".$_POST['eingabe'];

   } else {

      echo "<form method=\"POST\" action=\"myscript.php\">";
      echo "<input type=\"text\" name=\"eingabe\">";
      echo "<input type=\"submit\" name=\"b_submit\">";
   }
?>
```

Sessions

- Fähigkeit, Daten über mehrere Aufrufe hinweg festzuhalten
- Besucher wird beim ersten Aufruf eine eindeutige ID zugewiesen
- ID wird bei jedem Aufruf mitgesendet
- Beliebige Anzahl von Variablen registrierbar
- Objekte können abgelegt werden (Serialisierung)

```
<?php
session_start();
$_SESSION["username"] = "User Name";
$_SESSION["lottozahlen"] = array(9,13,20,30,41,45);

echo "<a href=\"myscript.php?".SID."\">Link</a>";
?>
```

Sessions

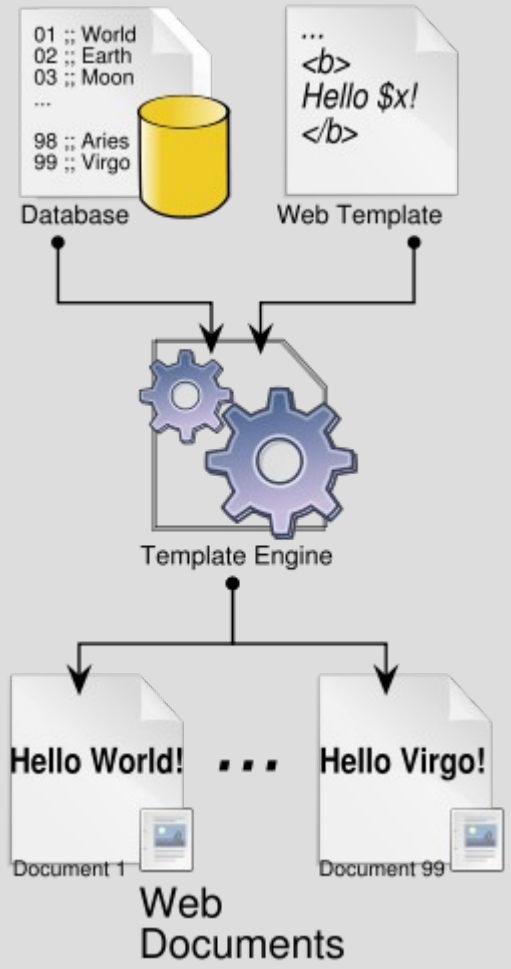

session_start();
SID vorhanden + Session mit SID vorhanden ?
ja
nein
Session Laden
Neue Session
$_SESSION
zurück zum Programm

Templates

- Trennung von Programmcode und Design
- Platzhalter im HTML-Code werden im PHP-Script ersetzt

```
<HTML>
<BODY>
Herzlich Willkommen, ##VORNAME##
</BODY>
</HTML>
```

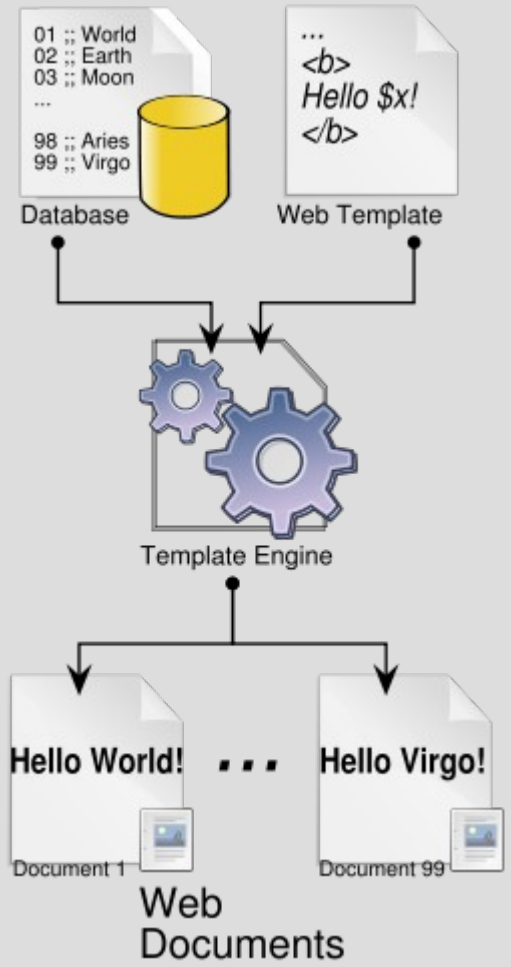
```
<?php
$template = file("mytemplate.tpl");
$template = implode("",$template);
$template =  str_replace(
 "##VORNAME##", "Hans", $template);
echo $template;
?>
```



Templates

Die bekanntesten Engines sind

- Smarty
- Heyes Template Class
- FastTemplate
- ShellPage
- STP Simple Template Parser
- OO Template Class
- Btemplate...



OOP mit PHP5

- Gültigkeitsbereiche definieren (private, public, protected)
- Statische Methoden
- Vererbung
- Abstrakte Klassen / Interfaces
- Fehlerbehandlung / Fehlerklassen
- Serialisierung von Objekten
- Iteratoren

Serialisierung von Objekten

- Serialisierung = Speicherung des Zustands
- Funktionen: serialize(), unserialize()
- Optional: __sleep-Methode, __wakeup-Methode

```
<?php
class Klasse {
 ...
}

$objekt = new Klasse;
$serialisiertesObjekt = serialize($objekt);
$objekt = unserialize($serialisiertesObjekt);
?>
```

PHP5: __autoload($className)

- Deklaration im globalen Sichtbarkeitsbereich
- Aufruf bei Objekterzeugung einer nichtdeklarierten Klasse

  → nur benötigte Klassen werden automatisch geladen

```
<?php
$GLOBALS['klassen'] = array(
'Projekt_Klasse' => 'Projekt/Klasse.php'
);

function __autoload($klasse) {
if (isset($GLOBALS['klassen'][$klasse])) {
require_once $GLOBALS['klassen'][$klasse];
}
}

$objekt = new Projekt_Klasse;
?>
```

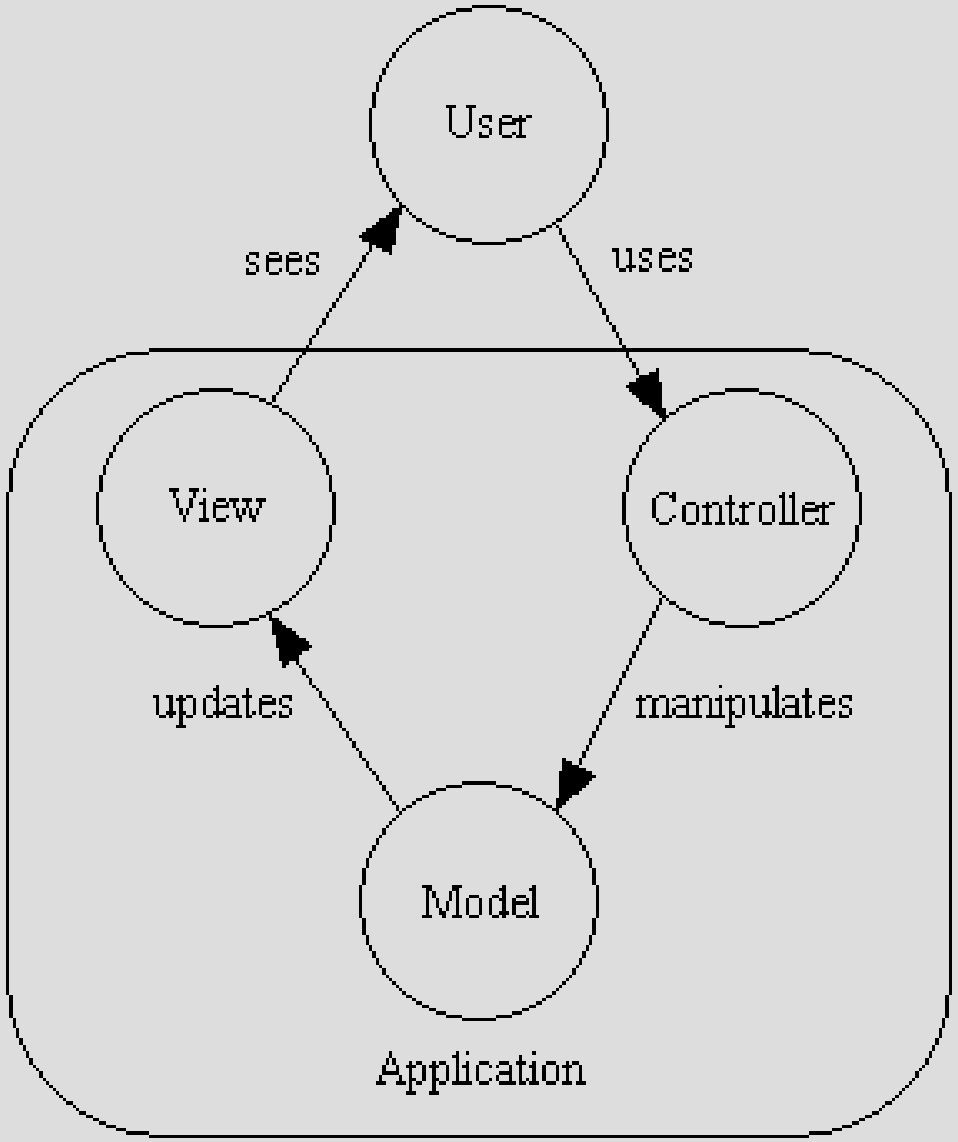
PHP5: __toString()

- Echo auf Objekt ergibt Ausgabe einer Identifikationsnummer
- Textuelle Repräsentation mittels __toString()

```
<?php
class BankAccount {
  private $balance = 3.5;
  public function __toString() {
    return sprintf(
      'Kontostand: %01.2f Euro.', $this->balance);
  }
}

$bankAccount = new BankAccount;
print $bankAccount;
?>
```

Ausgabe: Kontostand: 3.50 Euro

MVC - Pattern


User
sees
uses
View
Controller
updates
manipulates
Model
Application

Erzeugunsmuster: Singleton

- Anzahl der Objekte einer Klasse soll beschränkt werden
- Lösung: Konstruktor als private / protected; Objekterzeugung durch statische Methode getInstance()

```
<?php
class Singleton {
  private static $uniqueInstance = NULL;

  protected function __construct() { }

  public static function getInstance() {
    if (self::$uniqueInstance == NULL) {
      self::$uniqueInstance = new Singleton;
    }
    return self::$uniqueInstance;
  }
}
$a = Singleton::getInstance();
$b = Singleton::getInstance();
?>
```

Erzeugungsmuster: Factory

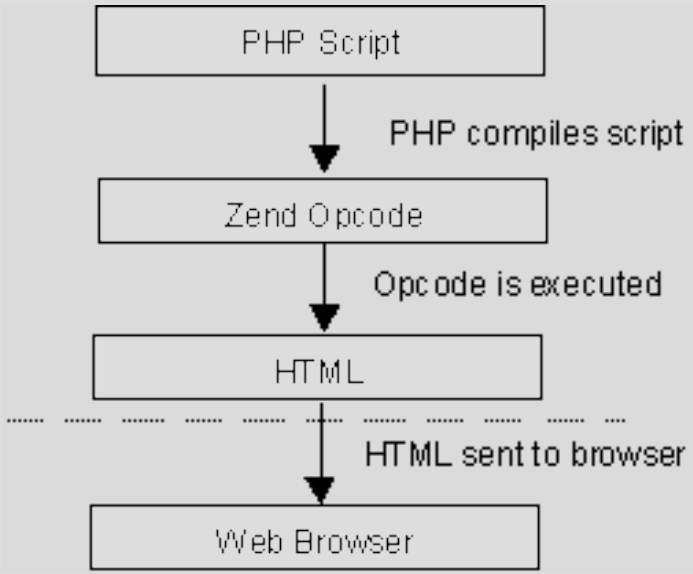
- Objekte verwandter Klassen erzeugen; verwendete Klasse erst zur Laufzeit festlegen
- Lösung:
  - gemeinsame Funktionalität in abstrakter Basisklasse
  - Basisklasse bietet statische Methode zur Objekterzeugung an

```
<?php
$interface = PartnerInterface::factory($type);
?>
```

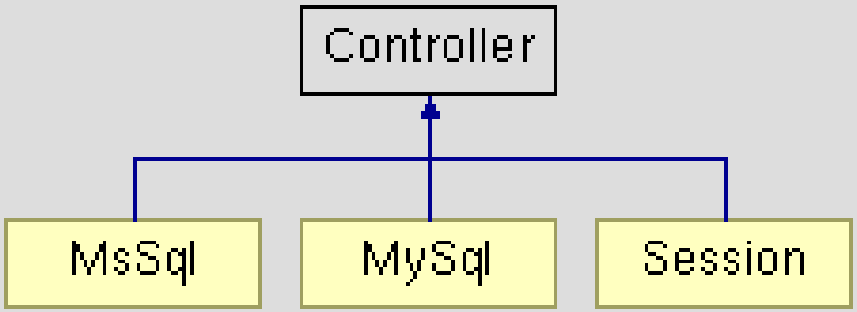
```
<?php
require_once 'PartnerInterface.php';

class PartnerInterface_HTTP extends PartnerInterface {
...
?>
```

Projekt-Eckdaten

- Server-OS: Linux x86_64 2.6.22.17 (OpenSuSE 10.3)
- Webserver: Apache 2.2.8
- PHP-Version: Version 5.2.5
  Zend Engine v2.2.0 with eAccelerator v0.9.5.2
- Datenbank: MySQL Version 5.0.51a



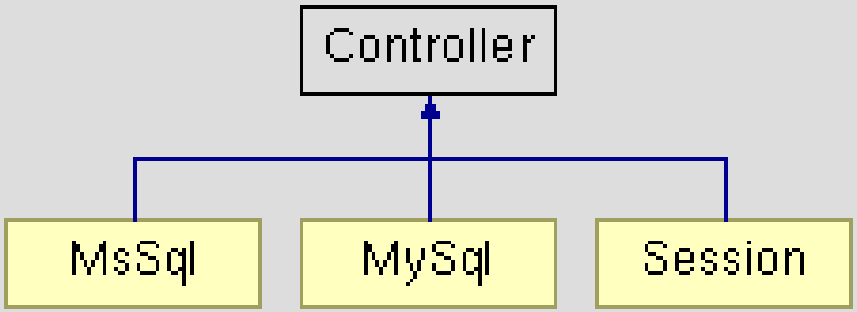
Die Realisierung



• Der Controller sorgt dafür, dass eine Instanz der Session Klasse erstellt wird und danach die der jeweiligen Datenbank Klasse

```
00112                   protected function register($classname, $parameters = array(), $alias = "")
00113                   {
00114                           if ($alias != "" && ! isset($_SESSION[$alias]))
00115                                   {
00116                                           $_SESSION[$alias] =& new $classname($parameters);
00117                                   }
00118                           else if (! isset($_SESSION[$classname]) && $alias == "")
00119                                   {
00120                                           $_SESSION[$classname] =& new $classname($parameters);
00121                                   }
00122                   }
00123
00134                   protected function unregister($classname, $alias = "")
00135                   {
00136                           if (isset($_SESSION[$classname]))
00137                                   {
00138                                           unset($_SESSION[$classname]);
00139                                   }
00140                           if ($alias != "" && isset($_SESSION[$alias]))
00141                                   {
00142                                           unset($_SESSION[$alias]);
00143                                   }
00144                   }
```

Die Realisierung



- Der Controller sorgt dafür, dass eine Instanz der Session Klasse erstellt wird und danach die der jeweiligen Datenbank Klasse

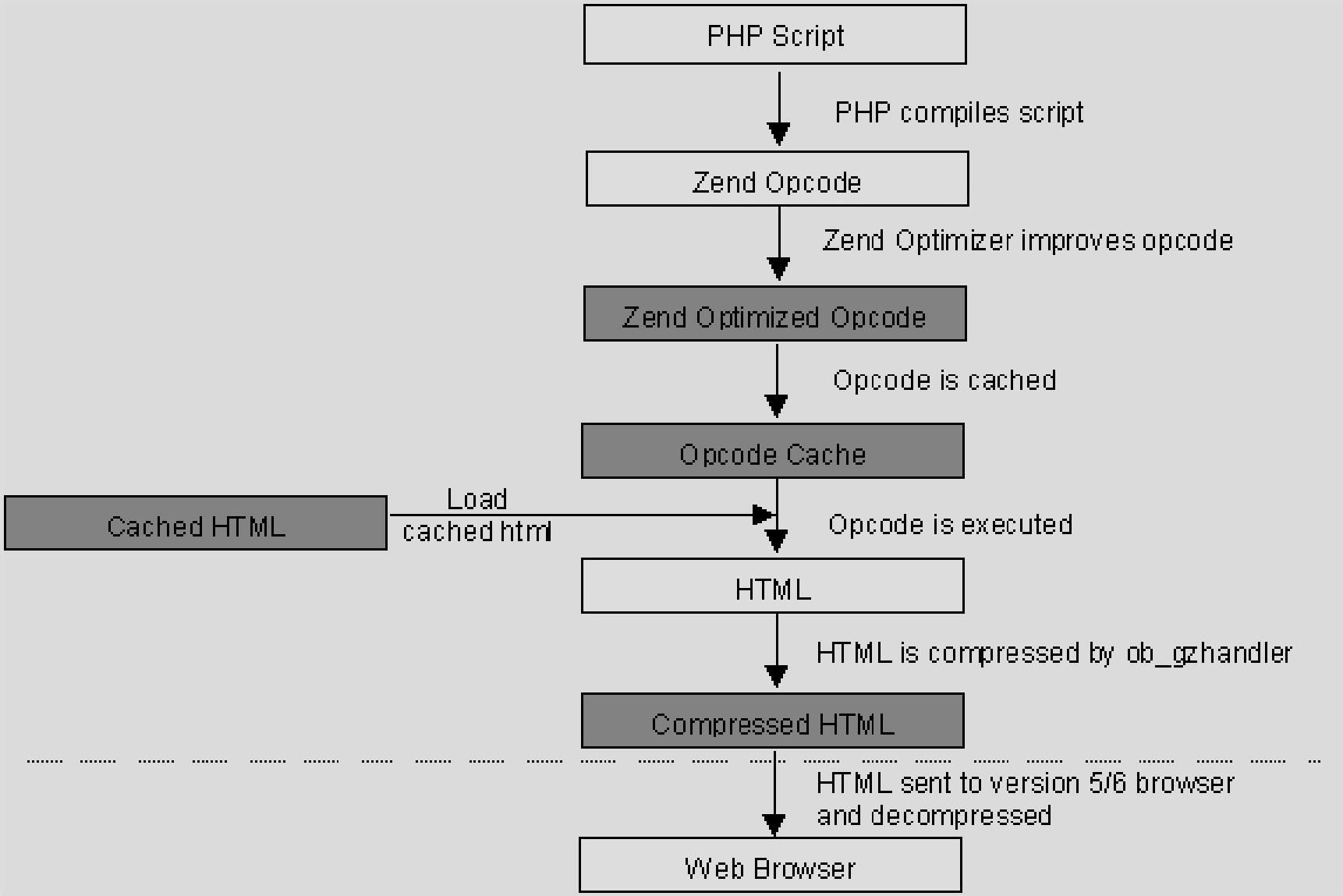
- Durch diesen „Trick“ bleiben die Objekte am „Leben“ auch wenn die Seite neu geladen wird

```
00069                   // try to register needed objects for current session or die
00070                   try
00071                           {
00072                                   $this->register("Memory",array(),"MEMORY");
00073
00074                                   $this->register("Timer",array("Controller",true),"TIMER.P
00075                                   $_SESSION["TIMER.PHP"]->reset();
00076                                   $_SESSION["TIMER.PHP"]->start();
00077
00078                                   $this->register("MySql", $_SETTINGS["MySql"], "MYSQL");
00079                                   $_SESSION["TIMER.MYSQL"]->reset();
00080
00081                                   $this->register("Template", $_SETTINGS["Template"], "HTML
00082                                   $this->register("Session",  array(), "CLIENT");
00083                           }
00084                   catch (Exception $e)
00085                           {
00086                                   die($e->getMessage());
00087                           }
```

„MVC“ Pattern

```
00001 <?php
00002 /* vim: set expandtab sw=4 ts=4 sts=4: */
00019 // warn, if php version older than 5
00020 if (0 > version_compare(PHP_VERSION, '5'))
00021         {
00022                 die('Diese Programm verwendet mindestens PHP5!');
00023         }
00024
00025 require_once "./functions.php";
00026
00027 try
00028         {
00029                 $Controller =& new Controller;
00030                 $Controller->prepare_actions();
00031                 $Controller->prepare_templates();
00032                 $Controller->create_menu();
00033                 $Controller->create_page();
00034         }
00035 catch (Exception $e)
00036         {
00037                 die($e->getMessage());
00038         }
00039
00040 ?>
```

Zend Encoder (eAccelerator)



ONLINE DEMO

@

http://www.omega2k.de/TimeRecording/