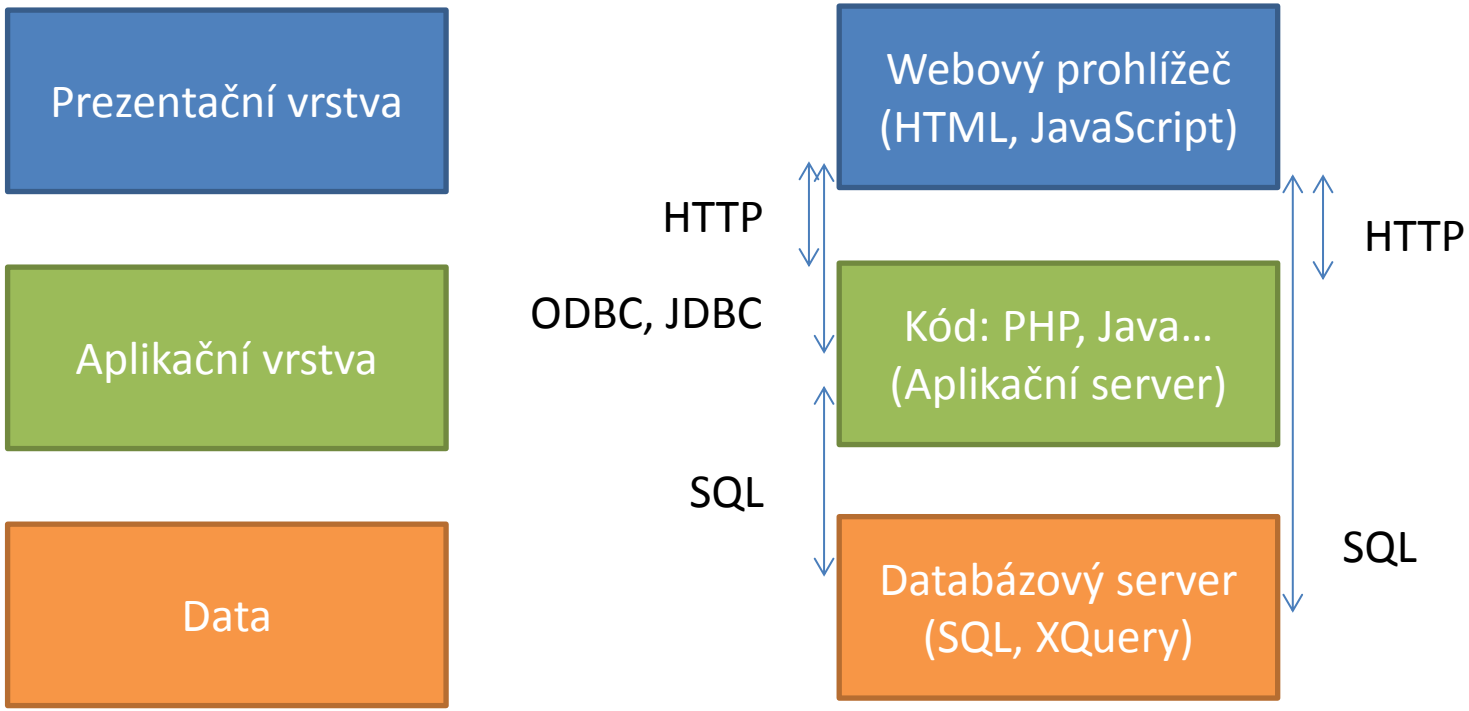
Databáze ve webových aplikacích

Lenka Kosková Třísková
NTI TUL

3 vrstvá architektura

Webová databázová aplikace



Databázový server

- Oracle, MS SQL, MySQL, PostgreSQL, Sybase, DB/2
- Typicky komunikuje po síti prostřednictvím TCP/IP
- Sledujeme:
  - Odolnost vůči zátěži
  - Schopnost spravovat velká úložiště
  - Schopnost být v clusteru
  - Implementaci SQL, další funkce
  - Zabezpečení
- *Pozor na tzv. databáze, co ukládají jen do souboru (MS Access)*

Standardizovaná konektivita: JDBC, ODBC

- ODBC = Open database connectivity (od MS)
  - Knihovna v PHP
  - Není objektové, původně C+ API (v .NET se nepoužívá)
- JDBC = Java database connectivity (pro J2EE)
- Databáze nabízí konkrétní driver, s nímž v aplikaci pracujeme
- Při změně databáze měníme jen ovladač

Připojení databáze k aplikaci

- Vytvoření spojení s databází
- Předání požadavku od klienta k databázi
- Zpracování výsledků
- Prezentace výsledků
- Odpojení od databáze

Spojení s databází

- HTTP bezzstavový:
  - Problém s více uživateli
  - Optimistický scénář:
    - Před zahájením práce lokálně uložíme původní data
    - Po ukončení práce kontrolujeme změnu
    - Případně řešíme konflikt
  - Pesimistický scénář:
    - Zamykání záznamu v databázi
    - Například nastavením zámku a přes „test set and lock“
    - Nebo uložením session do databáze, vhodná tabulka!
    - Pozor na časové platnosti, spojení je nestabilní a padá – nuté časové značky spolu se session a odpojit!

Jednoduchý příklad

```
<?php
 try { // připojení k databázi
 $db = new PDO("mysql:host=localhost;dbname=test", "jméno", "heslo");
    // zaslání dotazu a čtení výsledku
    foreach ($db->query("SELECT * FROM Zamestnanci ORDER BY Jmeno") as $radka) {
                // zpracování jednotlivých řádek výsledku
                echo $radka["OsobniCislo"], " ", $radka["Jmeno"], "<br>\n";
   } }
   catch (PDOException $e) {
                // obsluha případné chyby při práci s databází
   echo "Při práci s databází došlo k chybě: " . $e->getMessage();
}
?>
```

Object relational mapping (ORM)

- Mezivrstva, která překládá „objekty do SQL“
- Objekty pro uživatele
- SQL pro databázi
- Při použití mapujeme objekty na prvky databáze, dále probíhá automaticky
- PHP:
  - Dibi, malé, jednoduché
  - NotORM, jednoduché intuitivní API
  - Doctrine project – komplexnější knihovna pro ORM

Jiná datová úložiště

- XML databáze
  - Data jsou uložena v XML, ne v tabulkách
  - Dotazování prostřednictvím jazyků XQuery nebo XPath
  - Vhodné pro „svět dokumentů“, nevhodné pro relační data

Problémy s připojením

- Connection pool
  - „Bazének“ připravených spojení užívaných na vyžádání
  - Odstraní neustálé navazování a ukončování spojení
  - Dáno aplikačním serverem
  - Např. v Oracle i podpora pro PHP

Zvyšování výkonu

- „Nečistoty“ oproti databázové technologii
  - Cache předgenerovaných stránek (soubory nebo sdílená paměť)
  - Cache může mít několik úrovní
  - Odpovědi na typické dotazy jsou statické stránky
  - Složitá struktura tabulek odpovídající typickým požadavkům

Aplikační server

- Balíček všeho užitečného a navzájem provázaného
  - Webový server
  - Databázový server
  - Podpora ORM
  - Zabezpečení, správa uživatelů
  - Spojení s databází, persistentní spojení, pooling
  - Možnost ladit na výkon a paměťovou zátěž, podpora pro clustering
  - Správa verzí
  - Instalační balíčky, dobrá rozšiřitelnost a další vývoj konkrétní aplikace
- Další vrstva abstrakce při vývoji aplikace
- Typicky pro J2EE (IBM, Oracle)
- ZendServer pro PHP
- .NET Framework