Robert Haken [MVP ASP.NET/IIS, MCT]
software architect, HAVIT, s.r.o.
haken@havit.cz, @RobertHaken

Optimalizace výkonu webových aplikací

Přemýšlejte v souvislostech…

- Scénáře a kontext použití aplikace
  - business-aplikace vs. prezentace
  - private (intranet) vs. public (internet)
  - počet uživatelů, resp. očekávaná zátěž, současné přístupy, automated-requests (např. RSS čtečky)
  - rozložení cílové skupiny (devices, konektivita, …)
  - zdroje k dispozici (servery, připojení, vývojáři, čas)
- Pro každou situaci jsou vhodné jiné techniky!!!

Zacílení optimalizace

- klient
- přenos
- server

Optimalizace na straně klienta

- progresivní rendering HTML
  - co je možné, přesunout na konec HTML kódu, zejména velké JScript knihovny
  - progressive-layout (např. IE: table-layout:fixed; width)
- JavaScript parsing je drahý, nejenom jeho běh
- raději optimalizované knihovny (jQuery, Prototype, ...) než vlastní invence
- respektujte reálné možnosti DHTML

Optimalizace přenosu

- eliminace přenosu
  - client-side (browser) cache
  - Altair Cacheování v .NET a HTTP, aneb jak neuhnat sebe a svůj server
    - http://www.aspnet.cz/articles/389-deep-dive-cacheovani-v-net-a-http-aneb-jak-neuhnat-sebe-a-svuj-server-zaznam-priklady
  - použití CDN serverů
- minimalizace objemu přenesených dat
  - HTTP komprese
  - ViewState
  - HTML/JS/CSS minifiers
  - ID-čka pro ASP.NET server-controls
- minimalizace počtu requestů
  - Script + CSS bundling
  - CSS Sprites

Diagnostické nástroje klient+přenos

- Fiddler Web Debugger
- IE Developer Tools <F12> popř. FireBug
- YSlow

Fiddler basics
Microsoft Ajax Minifier
CSS Sprites
YSlow

DEMO

ASP.NET Bundling & Minification

- Microsoft.Web.Optimization.dll
- NuGet Package: ASP.NET Optimization - Bundling
- BundleTable.Bundles.EnableDefaultBundles()
  - /css +/js
- Custom bundles

```
// Creates a new bundle located at the URL "[host]/customscript".
// The bundle uses the build-in class JsMinify to do post processing
var b = new Bundle("~/CustomScript", typeof(JsMinify));
// Adds all .js files in the Scripts folder. Does not include sub-folders
b.AddDirectory("~/scripts", "*.js", false);
// Add any text based file relative to the project root
b.AddFile("~/scripts/script.js");
// Registers the bundle on the global BundleTable
BundleTable.Bundles.Add(b);
```

```
<script src="/CustomScript"
```

ASP.NET Optimization – Bundles & Minification

DEMO

Serverová optimalizace výkonu

- nastavení ASP.NET
- cachování
- ViewState
- optimalizace přístupů do DB
- asymptotická složitost algoritmů
- asynchronní zpracování, webová farma, ...

Nastavení ASP.NET

- `<compile ~~debug= "true"~~ />` + Release build
  - WebResources.ashx zapnutí cachování client-side
  - menší JS skripty (ASP.NET AJAX, Controls,...)
  - optimalizace
- `<trace enabled= "false" />`
  - šetří menší desítky ms per request
- `Session` – minimalizace použití nebo vypnutí
- pre-compile aplikace

Diagnostické nástroje web server

- ASP.NET <trace />
  - čas strávený v jednotlivých fázích životního cyklu
  - velikost ViewState
- Fiddler Web Debugger
  - WCAT – Web Capacity Analysis Tool
  - Fiddler Extension for WCAT
- Visual Studio Debugger, IntelliTrace, Performace Analysis
- RedGate ANTS Profiler, SQL Profiler
- PerfView, Performance Counters

Fiddler + WCAT Stress Test
Trace.Write(), Trace.Warn(), Trace.axd
trace enabled=true/false
compile debug=true/false

DEMO

ASP.NET Caching

- Output caching
- Data caching - application-scope
  - Application-scope (HttpRuntime.Cache)
  - „User“-scope (Session – obvykle nevhodné!)
  - Request-scope (HttpContext.Current.Items)
- pozor na vzájemné křížení cachovacích technik!

@OutputCache - ASP.NET Output Caching

- Page-level caching, Fragment caching (User-controls)
- VaryBy... (VaryByParam, VaryByHeader, ...)
  - VaryByCustom , global.asax:

```
public override string GetVaryByCustom(HttpContext context, string arg)
```

- Duration [sec], Location
- <asp:Substitution MethodName="...
- system.web/caching/outputCache/enableOutputCache + enableFragmentCache – hromadné vypnutí pro debugging
- pozor na vzájemné křížení!

ASP.NET Output Caching

DEMO

Data Caching

- `HttpRuntime.Cache`
  - popř. `MemoryCache.Default` (.NET 4.0+)
- pozor na rozsáhlé objektové grafy (root-referenced)
- datové ostrůvky s markerem
- opatrně s dependency
- použitelné i mimo webové aplikace

Application Scope Data Caching
HttpRuntime.Cache

DEMO

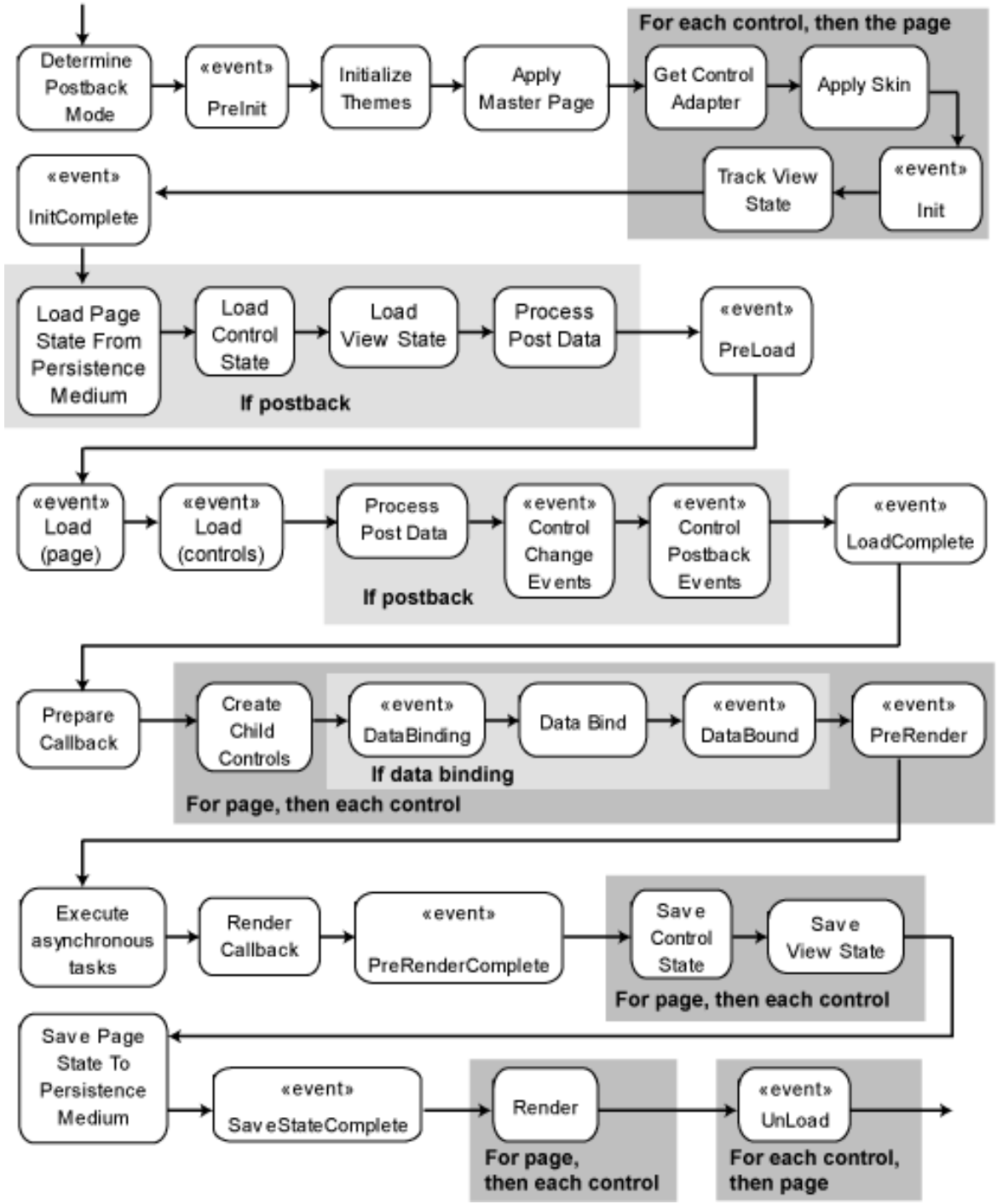
Request-scope data caching

- opakované použití jednou získaných dat
- `Page/Control/UserControl`
  - izolované instance třídy
  - => lokální fieldy
- `HttpContext.Current.Items[key] = value`

ViewState

- minimalizace ViewState
  - EnableViewState= “false“
  - ViewStateMode= “Enabled|Disabled|Inherit“
- přesměrování ViewState na jiné uložiště (↓přenos)
  - SessionPageStatePersister
  - disk serveru
- komprimace ViewState

Page/Controls Life-Cycle



ViewState

DEMO

Webová Farma

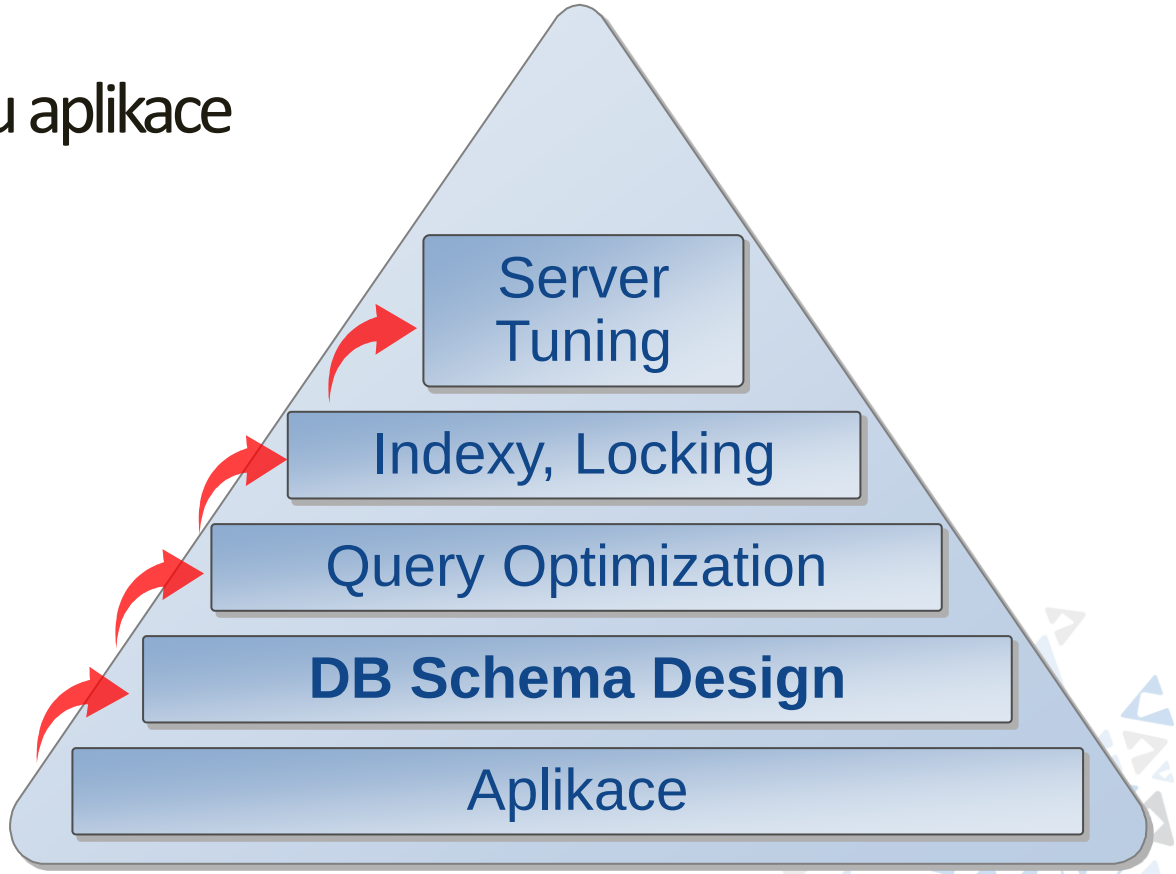
- out-proc Session nebo vůbec
- sdílení MAC kódu
- kvalitní load-balancing
- web-garden obvykle moc nepřináší

Optimalizace přístupu do DB

- získávání dat „s rozumem“ (např. stránkování)
- cachování dat získaných z DB
- „přirozená“ normalizace **schématu DB**
- connection-pooling
- indexy, údržba statistik
- execution plans
- multiple-resultsets

Proč je schéma DB důležité?

- klíčový prvek designu aplikace
- zásadní vliv na výkon
  - rychlost
  - objem dat
- konzistence dat



Časová a paměťová složitost

- Relativně vůči velikosti vstupu … N
- Časová složitost
  - Počet „elementárních operací“ vzhledem k velikosti dat
    - Element. operace trvá stejně dlouho bez ohledu na velikost vstupu
- Paměťová složitost
  - Spotřebovaná paměť vzhledem k velikosti dat

Třídy složitosti

- $O(1)$ … konstantní
- $O(\log N)$ … logaritmická
- $O(N)$ … lineární
- $O(N \log N)$ … lineárně logaritmická
- $O(N^2)$ … kvadratická
- $O(N^X)$ … polynomiální (kubická, ...)

- $O(a^N)$ … exponenciální

Exponenciální složitost

- např. nejkratší průjezd všemi body grafu, atp.

- N = 1 … t = 0,002s
- N = 2 … t = 0,004s
- N = 3 … t = 0,008s
- N = 4 … t = 0,015s
- N = 5 … t = 0,031s
- N = 10 … t = 1s
- N = 11 … t = 2s

Exponenciální složitost

- Průšvih

- N = 12 … t = 4s
- N = 14 … t = 16s
- N = 20 … t = 17 min
- N = 30 … t = 12 dní 3 hodiny
- N = 40 … t = 34 let
- N = 50 … t = 34 841 let
- N = 100 … t = $4 . 10^{19}$ let

Známé algoritmy

- O(1) - konstantní
  - Hashovací tabulka
- O(log N) - logaritmická
  - Hledání v seřazeném poli (půlení intervalů)
  - Hledání ve vyhledávacím stromě
- O(N) - lineární
  - Hledání minima, maxima v poli
  - Hledání prvku v neseřazeném poli
  - Slití dvou seřazených polí
  - Hledání v textu
- O(N log N) – lineárně logaritmická
  - Seřazení pole
    - QuickSort, HeapSort, MergeSort

Složitost operací

	Přidání na konec	Odebrání z konce	Vložení doprostřed	Odebrání z prostředka	Přístup podle indexu	Sekvenční přístup	Vyhledávání elementu	Poznámky
Array	O(n)	O(n)	O(n)	O(n)	O(1)	O(1)	O(n)	Nejefektivnější využití paměti; vhodné v případě neměnného počtu položek.
List<T>	většinou O(1); nejhůře O(n)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)	Vnitřně implementováno pomocí pole, při zaplnění se 2x zvětší. Přidávání nebo odebírání z prostředka či začátku jsou pomalé.
LinkedList<T>	O(1)	O(1)	O(1)*	O(1)*	O(n)	O(1)	O(n)	Obousměrný spojový seznam. Pomalé přístupy doprostřed, rychlé přidávání na začátek a na konec. *) pokud už mám referenci na příslušnou položku
Stack<T>	většinou O(1); nejhůře O(n)	O(1)	N/A	N/A	N/A	N/A	N/A	Zásobník, vhodné pro implementaci různých algoritmů. Last in, first out.
Queue<T>	většinou O(1); nejhůře O(n)	O(1)	N/A	N/A	N/A	N/A	N/A	Fronta, vhodná pro implementaci různých algoritmů. First in, first out.
Dictionary<K,T> HashTable	většinou O(1); nejhůře O(n)	O(1)	většinou O(1); nejhůře O(n)	O(1)	O(1)*	O(1)*	O(1)	Vnitřně implementováno pomocí hashovací tabulky, vhodné pro rychlé vyhledávání podle klíče. *) postavení pole Keys a Values má složitost O(N), ale stačí ho postavit jen jednou

Performance killers z praxe

- drahé datové operace
  - nesprávné načítání dat z DB
  - nepoužívání data-cache, nesprávné použití
  - opakovaný data-binding (bez kontroly IsPostback při zapnutém ViewState)
  - extensivní logování, nekorektní počítadla návštěv, atp.
- chybný resource-management
  - používejte using() pro IDisposable
- nesprávné použití AJAX, zejm. UpdatePanelů
  - mají obalovat blok, který se má aktualizovat,
  - ne controly, jejichž události se zachytávají!!!
- nadužívání Session

Q & A