Tvorba webových aplikácií

MICHAL BARLA

Pojmy

HTTP, HTML, Javascript, CSS, Server, Prehliadač, MVC, REST, AJAX, Webové služby, Rails, Php, Tomcat, Django, J2EE, Silverlight, ASP.NET, Apache, Nginx, jQuery, node.js, Firebug, Opera, ~~Internet Explorer,~~ Firefox, Chrome, XML, AWS, Flash, JavaFX, Prototype, YUI, GWT, Capistrano, Sinatra, Rack, Response, Request, Cucumber, URI

Prečo webové aplikácie?

- Dostupnosť
- Prirodzená multiplatformovosť
- Podpora spolupráce používateľov
- Interoperabilita
- Aktualizácie

Základné delenie prístupov

- Klasické
  - Klient – HTML renderer + javascript
  - Server – HTML/XML/JSON
- „desktopové“
  - Klient –Silverlight/Flash/Java Applet zásuvný modul (plugin) v prehliadači
  - Server – poskytuje služby, ktoré klientský program využíva

Porovnanie

Klasické	Tučný klient cez plugin

• Bez potreby zásuvného modulu na strane klienta
• Logika sa vykonáva na strane servera
  ○ Ten má dáta
• Bezstavové
  ○ REST (zachvíľu)
• Ajax pre Look&Feel desktopu
	• Potreba zásuvného modulu na strane klienta
	• Logika sa vykonáva na strane klienta
	  ○ Ten potrebuje mať dáta
	• Programuje sa ako klasická desktopová aplikácia so všetkými výhodami a nevýhodami

REST

- **Re**presentational **S**tate **T**ransfer
- Prístup klientov k zdrojom
  - Zdroj má jednoznačný identifikátor
  - Zdroje môžu byť v rôznych reprezentáciách (HTML,JSON,...)
- Stav je iba na klientovi, nie na serveri
  - Každý dopyt obsahuje všetko potrebné pre jeho vyriešenie
  - Operácia nad zdrojom
- HTTP GET,POST,PUT,DELETE,...
  - Aplikovanie slovies nad menami

Výhody RESTu

- „Donúti“ Vás urobiť aplikáciu „poriadne“
  - Bude sa vám ľahšie rozširovať
- Jeden kód pre veľa klientov
  - html pre ľudí (teda pre browsre)
  - xml pre RSS agregátory
  - json pre klientov idúcich cez vaše API
- Krajšie URLky ☺
  - add.py?id=123 – koho zaujíma, že viete python?
  - StudentManager.aspx?GetStudentWithId=123
- Jednoduchšie cachovanie

Príklady

- http://www2.fiit.sk/~bielik/courses/psi-slov/

  vs.

  http://www2.fiit.sk/~bielik/courses/GetCourse.aspx?course=psi-slov

- GET http://alef.fiit.stuba.sk/learning_objects/1072799192
- GET http://alef.fiit.stuba.sk/learning_objects/1072799192/comments/1
- GET http://alef.fiit.stuba.sk/learning_objects/
- POST http://alef.fiit.stuba.sk/learning_objects/
- PUT http://alef.fiit.stuba.sk/learning_objects/1072799192
- DELETE http://alef.fiit.stuba.sk/learning_objects/1072799192

Model-View-Controller

- Model
  - Dáta a biznis logika
  - Má stav, ktorý mení controller a dopytuje sa naň view
- View
  - Vie zobraziť model do podoby, s ktorou sa dá interagovať
- Controller
  - Odchytáva vstup používateľa a iniciuje odpoveď menením stavu modelu

Model-View-Controller

- Model
  - Údaje s ktorými webová aplikácia pracuje
    - Objednávka, Produkt, Choroba, Izba, Aukcia,...
  - Najčastejšie objektový model nad relačnou databázou (ideálne s O/R mapovačom)
- View
  - Šablóna, ktorá definuje spôsob, akým sa údaje zobrazia
- Controller
  - Vyberie niečo z modelu a posunie do šablóny na zobrazenie

Príklad – RESTful blog

- Čo chceme
  - http://nejaka-webka.sk/posts
  - http://nejaka-webka.sk/posts/1
  - http://nejaka-webka.sk/posts/1/comments
- Potrebujeme niečo, čo nám namapuje tieto URLky na jednotlivé metódy controllerov
  - O to sa postará framework, ale až nabudúce

Model-View-**Controller**

- PostsController
  - Zobrazenie všetkých príspevkov (GET)
  - Zobrazenie formulára na pridanie príspevku (GET)
  - Zobrazenie formulára na update príspevku (GET)
  - Pridanie/update príspevku (POST/PUT)
  - Vymazanie príspevku (DELETE)
- CommentsController podobne

Príklad: Zobrazenie príspevku

```
def show
    @post = Post.find(params[:id])
    respond_to do |format|
        format.html # show.html.erb
        format.xml  { render :xml => @post }
    end
end
```

Príklad: **Model**-View-Controller

```
Class Post < ActiveRecord::Base
    validates :title, :presence => true
    has_many :comments
End

Class Comment < ActiveRecord::Base
    belongs_to :post
end
```

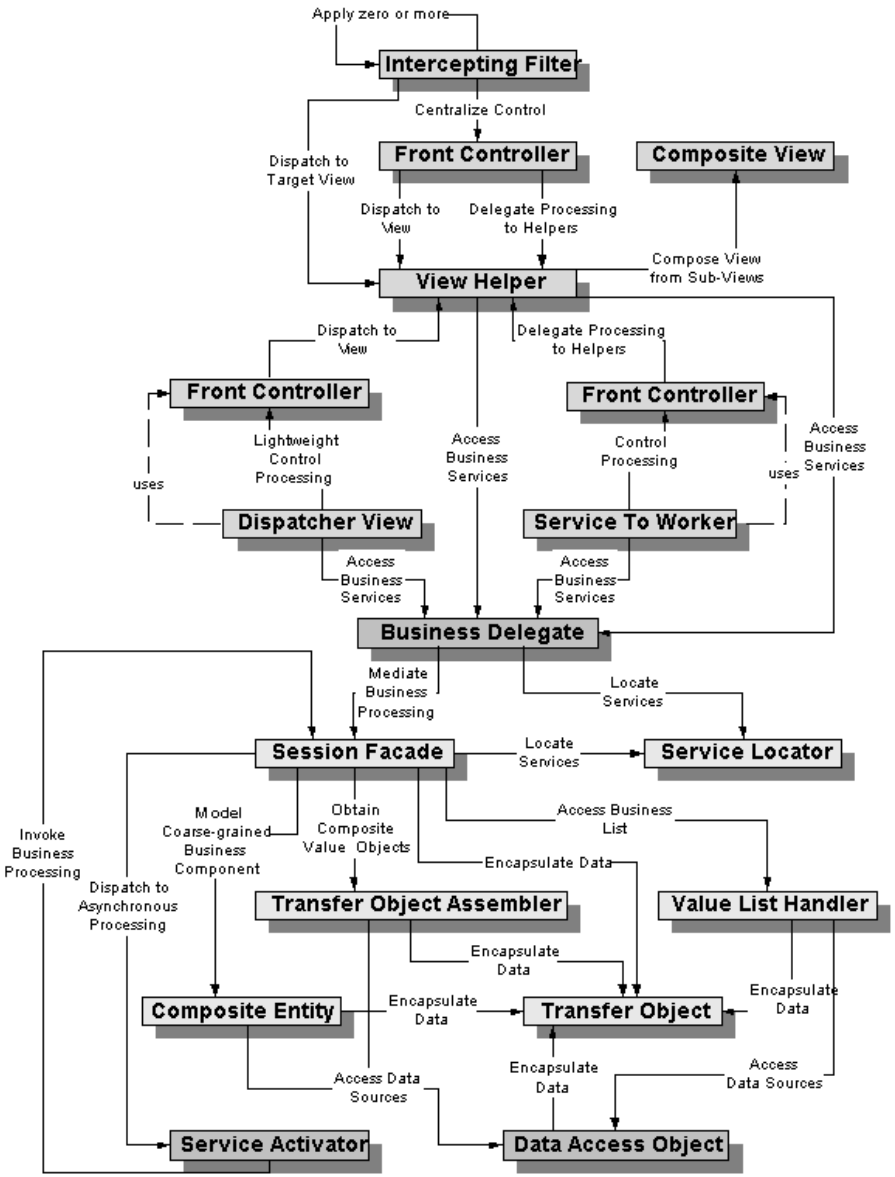
Príklad Model-**View**-Controller

```
<p>
  <b>Title:</b>
  <%= @post.title %>
</p>

<p>
  <b>Content:</b>
  <%= @post.content %>
</p>

<%= link_to 'Edit', edit_post_path(@post) %> |
<%= link_to 'Back', posts_path %>
```

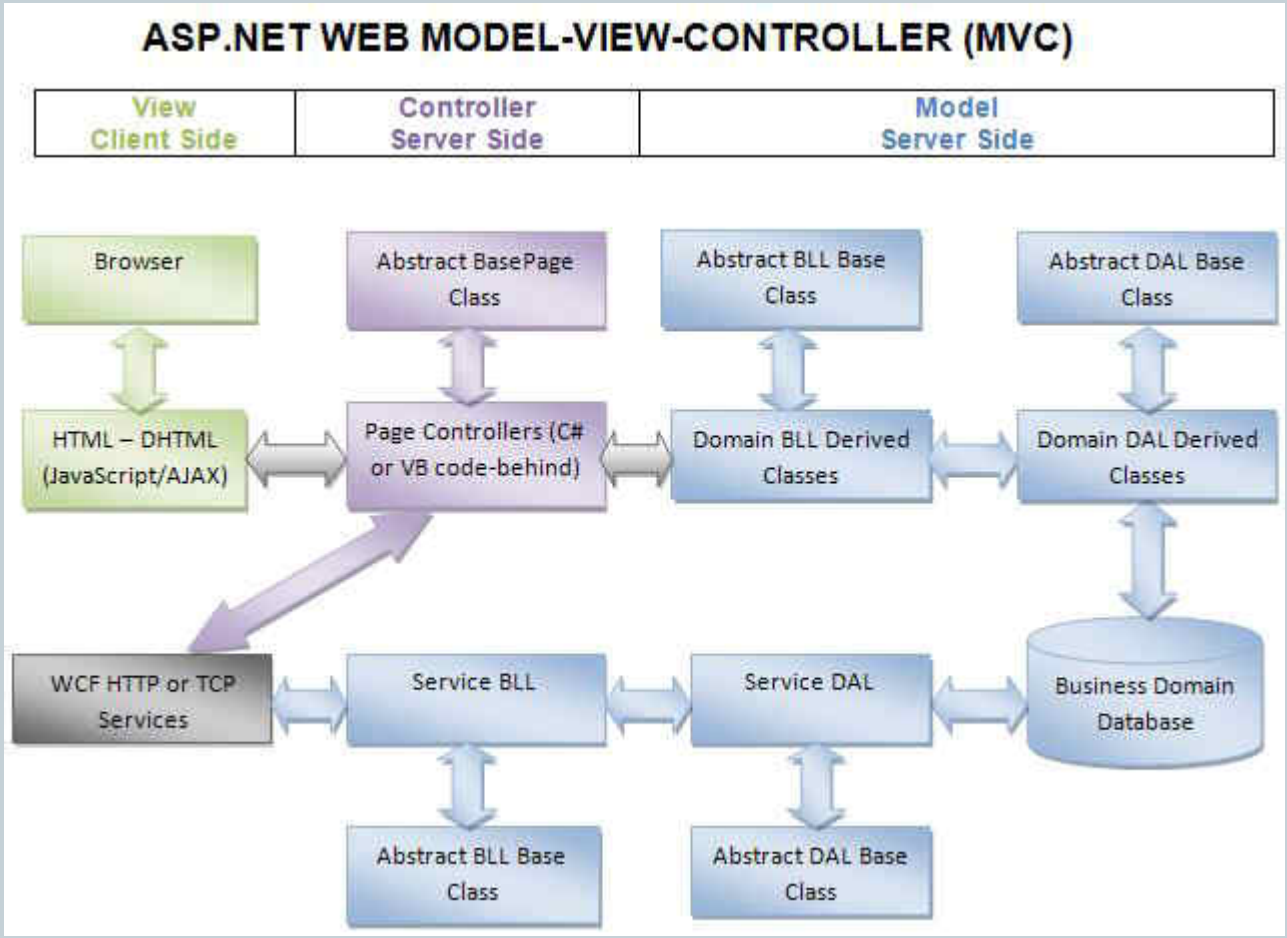
Tvorba webových aplikácií

- MVC je základ, ktorý je už teraz všade
  - php/ruby/python frameworky (vo štvrtok)
  - „Enterprise“ svet (u nás)
    - J2EE
    - .NET MVVM

Core J2EE patterns



.NET


ASP.NET WEB MODEL-VIEW-CONTROLLER (MVC)
View
Client Side
Controller
Server Side
Model
Server Side
Browser
Abstract BasePage Class
Abstract BLL Base Class
Abstract DAL Base Class
HTML – DHTML (JavaScript/AJAX)
Page Controllers (C# or VB code-behind)
Domain BLL Derived Classes
Domain DAL Derived Classes
WCF HTTP or TCP Services
Service BLL
Service DAL
Business Domain Database
Abstract BLL Base Class
Abstract DAL Base Class