



REST

Architektúra pre lepšie webové aplikácie a webové služby

Peter Rybár



REST

Architektúra pre lepšie
webové aplikácie a služby

Peter Rybár

Obsah

■ Web

- protokol, použitie, komponenty, služby

■ REST

- motivácia, poslanie, architektúra, štandardy

■ Web 2.0

- RIA, sémantika, mikroformáty, princípy, techniky

WEB

- **World Wide Web** (skrátene **Web**) je systém vzájomne nalinkovaných hypertextových dokumentov prístupných prostredníctvom **HTTP** v sieti Internet
- **World Wide Web** vytvorili Sir **Tim Berners-Lee** and **Robert Cailliau** v roku **1989** v **CERNe** v Ženeve, Švajčiarsko
- Princíp – používateľ Webu (HTTP klient) prezerá web stránky obsahujúce **text, obrázky, videá**, a iné **multimédiá** a naviguje sa medzi nimi pomocou **hyperliniek** v **HTML**

Web: HTTP

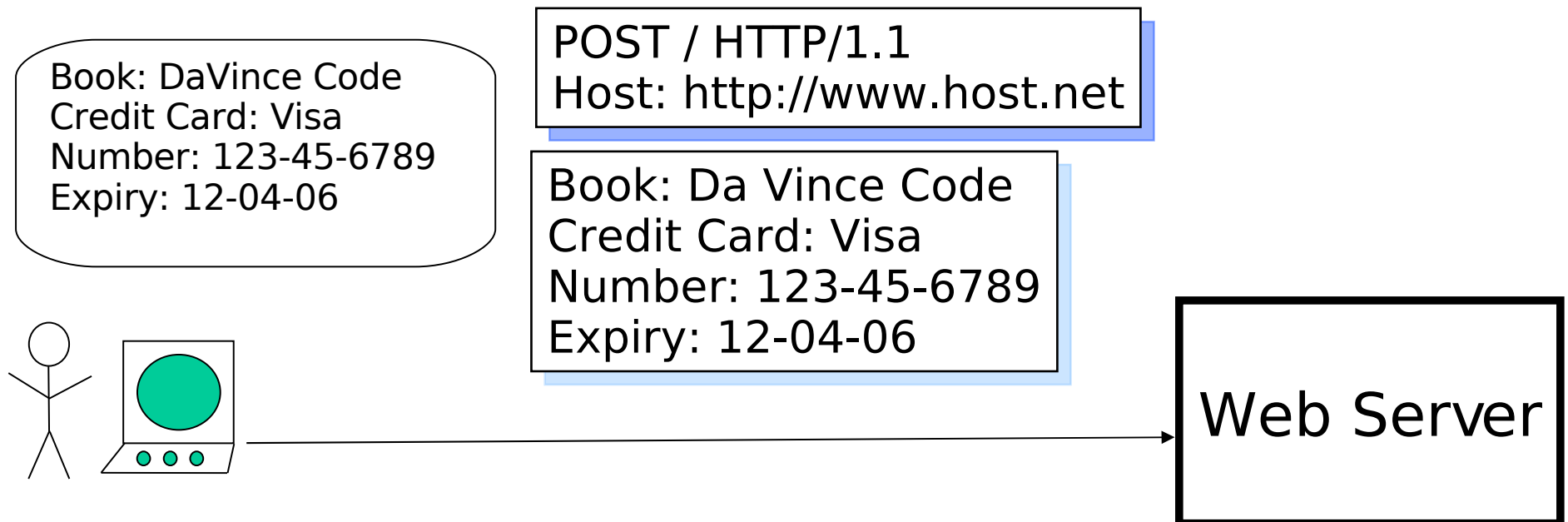
- **HTTP** – Hypertext Transfer Protocol
- Transportný protokol pre prenos informácií, hypertextových stránok na internete
- Všetok Web prenos je realizovaný prostredníctvom jednoduchého **HTTP API**:
 - **GET** = "daj mi nejakú informáciu" (Retrieve)
 - **POST** = "tu máš nejakú novú informáciu" (Create)
 - **PUT** = "tu je nejaká update infomácia" (Update)
 - **DELETE** = "vymaž nejakú informáciu" (Delete)
- **HTTP API** je **CRUD** (Create, Retrieve, Update, a Delete)

Web: Použitie HTTP GET



- Klient zadá do svojho prehliadača: **http://www.host.net**
- Prehliadač vytvorí **HTTP hlavičku**
 - HTTP hlavička identifikuje:
 - Požadovanú **akciu**: GET ("get me resource")
 - Cieľový **stroj** (www.host.net)

Web: Použitie HTTP POST



- Klient vyplní **formulár** na webovskej stránke
- Prehliadač vytvorí **HTTP hlavičku** a dáta z formulára umiestni do **HTTP body**
- **HTTP hlavička** identifikuje:
 - Požadovanú **akciu**: POST
 - Cieľový **server** (host.net)
- **HTTP body** obsahuje:
 - **Data** POSTnute (encoded form data)

Webové komponenty

■ Firewally:

- Rozhodujú ktoré HTTP správy (messages) môžu von a ktoré môžu dnu
- Uplatňujú **Webovú bezpečnosť**

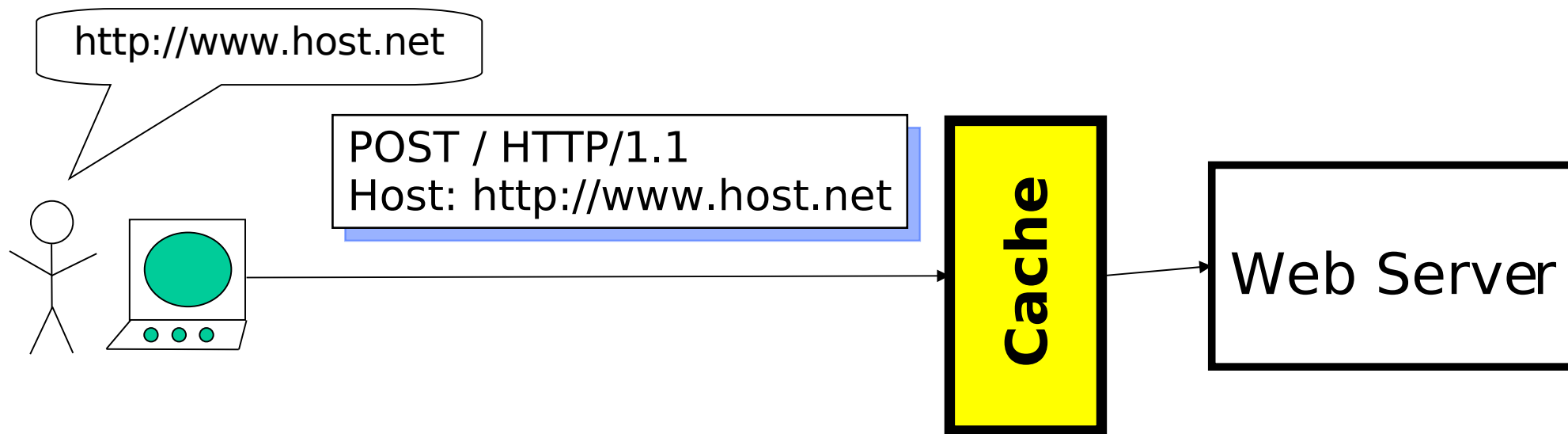
■ Routs:

- Rozhodujú kde poslať HTTP správy (messages)
- Riadia **Webovú škálovateľnosť**

■ Caches:

- Rozhodujú či uložená kópia HTTP správy (message) môže byť použitá
- Zvyšujú **Webovú rýchlosť**

Webové komponenty: Cache



- Cache sa rozhoduje, či by mala použiť ešte platnú kópiu HTTP dokumentu, alebo si vyžiadať aktuálnejšiu správu od web servera
- Všetky rozhodnutia sú na základe **HTTP hlavičky**
- **Cache nikdy nepozerá do prenášaných dát HTTP správ** (payload)

Webové služby

- Softvérový systém navrhnutý pre komunikáciu Stroj-Stroj prostredníctvom siete **internet**.
- Sada nástrojov, ktoré môžu byť použité rôznym spôsobom na rôzne účely.
- Tri najčastejšie spôsoby použitia sú:
 - **RPC**
 - **SOA**
 - **REST**



Webové služby: RPC

- RPC Webové služby reprezentujú **interface pre volanie vzdialenej funkcie** (metódy)
- Sú **kritizované pre silnú väzbu**, pretože boli implementované *mapovaním služieb priamo do špecifických funkcií a metód daného jazyka*
- Keď bola do XML-RPC zavedená nová funkcionálna, bol uvedený nový štandard známy dnes ako SOAP
- Niektorí vývojári stále preferujú XML-RPC pred SOAP pre jeho **minimalizmus a jednoduchosť použitia**

Webové služby: RPC

Príklad XML-RPC

POST /RPC HTTP/1.0
Host: example.org
Content-Type: text/xml
Content-length: nnn

```
<?xml version="1.0"?>  
<methodCall>  
  <methodName>examples.getStateName</methodName>  
  <params>  
    <param>  
      <value><i4>41</i4></value>  
    </param>  
  </params>  
</methodCall>
```

Webové služby: SOA

- Základnou jednotkou komunikácie je **správa (message)**, skôr než procedúra
- Táto architektúra zvykne byť označovaná ako **“message oriented”**
- Implementácia konceptu **servisne orientovanej architektúry (SOA)**
- Najčastejšie používaný protokol pre SOA je **SOAP Document Literal**
- **Simple Object Access Protocol** neskôr ako **Service Oriented Architecture Protocol**

Webové služby: SOA

Príklad SOAP Document Literal

POST /InStock HTTP/1.1

Host: www.example.org

Content-Type: application/soap+xml; charset=utf-8

Content-Length: nnn

```
<?xml version="1.0"?>
```

```
<soap:Envelope
```

```
  xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
```

```
  soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
```

```
  <soap:Body xmlns:m="http://www.example.org/stock">
```

```
    <m:GetStockPrice>
```

```
      <m:StockName>IBM</m:StockName>
```

```
    </m:GetStockPrice>
```

```
  </soap:Body>
```

```
</soap:Envelope>
```

Webové služby: REST

- **REST** - Representational state transfer
- **Dôraz** sa kladie **na interakciu so stavovými zdrojmi** (resources), než na **správy** (messages) alebo **operácie** (procedures)
- RESTovské Webové služby sa pokúšajú emulovať HTTP a podobné protokoly obmedzením rozhrania na sadu štandardných operácií (ako, GET, PUT, POST, DELETE).

Webové služby: REST

Príklad REST

POST /parts/12345 HTTP/1.0

Host: example.org

Content-Type: text/xml

Content-length: nnn

```
<?xml version="1.0"?>
```

```
<p:Part xmlns:p="http://www.parts-depot.com">
```

```
  <ID>00345</ID>
```

```
  <UnitCost currency="USD">0.10</UnitCost>
```

```
  <Quantity>10</Quantity>
```

```
</p:Part>
```


Webové služby: SOAP vs REST

SOAP

The following is a sample SOAP request and response. The [placeholders](#) shown need to be replaced with actual values.

```
POST /DemoWebServices2.8/service.asmx HTTP/1.1
Host: api.efxnow.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "https://api.efxnow.com/webservices2.3/GetTime"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetTime xmlns="https://api.efxnow.com/webservices2.3" />
  </soap:Body>
</soap:Envelope>

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <GetTimeResponse xmlns="https://api.efxnow.com/webservices2.3">
      <GetTimeResult>string</GetTimeResult>
    </GetTimeResponse>
  </soap:Body>
</soap:Envelope>
```

Webové služby: SOAP vs REST

HTTP GET

The following is a sample HTTP GET request and response. The [placeholders](#) shown need to be replaced with actual values.

```
GET /DemoWebServices2.8/service.asmx/GetTime? HTTP/1.1
Host: api.efxnow.com
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="https://api.efxnow.com/webservices2.3">string</string>
```

HTTP POST

The following is a sample HTTP POST request and response. The [placeholders](#) shown need to be replaced with actual values.

```
POST /DemoWebServices2.8/service.asmx/GetTime HTTP/1.1
Host: api.efxnow.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
```

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length
```

```
<?xml version="1.0" encoding="utf-8"?>
<string xmlns="https://api.efxnow.com/webservices2.3">string</string>
```

REST

- Akronym pre **Representational State Transfer**
- **Štýl softvérovej architektúry** určený pre **distribuované hypermediálne systémy** ako je **World Wide Web**
- Zaviedol ho **Roy Fielding** v jeho Ph.D. práci pre **popis architektonického štýlu sieťou prepojených systémov**
- REST striktne definuje kolekciu princípov sieťovej architektúry, ktorá **vysvetľuje ako sú zdroje (resources) definované a adresované**



REST: Motivácia vzniku

- Zozbierať charakteristiky webu, ktoré ho spravili úspešným
- Následne použiť tieto charakteristiky ako návod pre evolúciu webu

REST: Poslanie

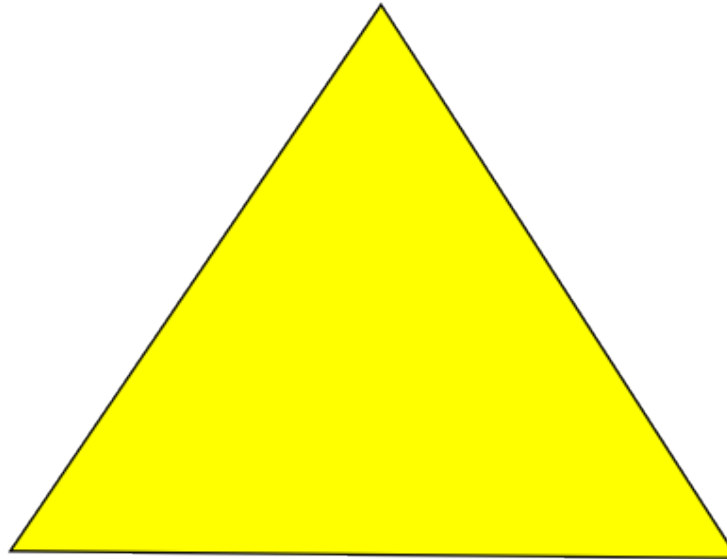
- REST je učený aby uviedol spôsob, **ako sa má správať dobre navrhnutá webová aplikácia**
 - Webová aplikácia je sieť web **stránok, virtuálnych stavových strojov**
 - Užívateľ prechádza cez aplikáciu prostredníctvom **liniek, prenosov stavu**
 - Výsledkom prenosu stavu je nasledujúca **stránka, reprezentujúca nasledujúci stav aplikácie**, prenesená k užívateľovi

REST: Restovský trojúhelník

Nouns

(Unconstrained)

eg <http://wikipedia.org/>



Verbs

(Constrained)

eg GET

Content Types

(Constrained)

eg HTML

REST: Prenos stavu

- Klient referencuje **zdroj** pomocou **URL**
- Klientovi je vrátená **reprezentácia** zdroja
- Reprezentácia uvedie klienta do **nového stavu**
- Klient zvolí hyperlinku ďalšieho **zdroja**
- Klientová aplikácia teda robí transfér stavu s každou reprezentáciou zdroja
- **Nová reprezentácia – nový stav!**

REST: Arch. štýl, nie štandard

- **REST nie je štandard!**
- Neuvidíte W3C špecifikáciu RESTu
- Nemali by ste vidieť predávať developerské toolkity od IBM, Microsoft alebo Sunu
- **Prečo?**
- Lebo **REST je “iba” architektonický štýl** (analógia s klient-server štýlom)
- Môžete RESTu iba rozumieť a dizajnovat' aplikácie podľa neho

REST: Štandardy

- Hoci **REST** nie je štandard, používa štandardy:
 - **HTTP** (Transport protocol)
 - **URL** (Resource identifier)
 - **XML/HTML/GIF/JPEG/...** (Resource Representations)
 - **text/xml, text/html, image/gif, image/jpeg, ...** (MIME Types)

REST: Web je REST systém

- Webové služby, ktoré používame už roky:
 - book-ordering services
 - search services
 - online dictionary services
- Všetko sú to RESTovské Webové služby
- Používali sme REST, stavali sme RESTovské webové služby a ani sme o tom “nevedeli” !
- **Google – koniec SOAP – už iba REST !**

REST: Pre a proti


■ Výhody

- Linkovanie a bookmarkovanie – “google friendly URLs”
- Podpora štandardnej sady operácií (CRUD)
- Škálovateľnosť
- Jednoduchá implementácia – HTTP libs
- Slabá väzba komponentov (loose coupling)
- Možná neskorá väzba – HTTP status 302
- Vyššia možnosť znovupoužitia kódu

REST: Pre a proti

■ Nevýhody

- Viazaný na HTTP
- Veľké množstvo objektov
- Správa URI menných priestorov (namespace) môže byť ťažkopádna – závisí na architektovi



Webová aplikácia ako klient ku RESTovským webovým službám, RIA, Web 2.0

Peter Rybár

Klasické Web aplikácie

■ Výhody

- *Jednoduchosť* – dokumentovo orientované
- *Dostupnosť* – URL reprezentuje dokument (REST)
- *Jednoduchá implementácia*

■ Nevýhody

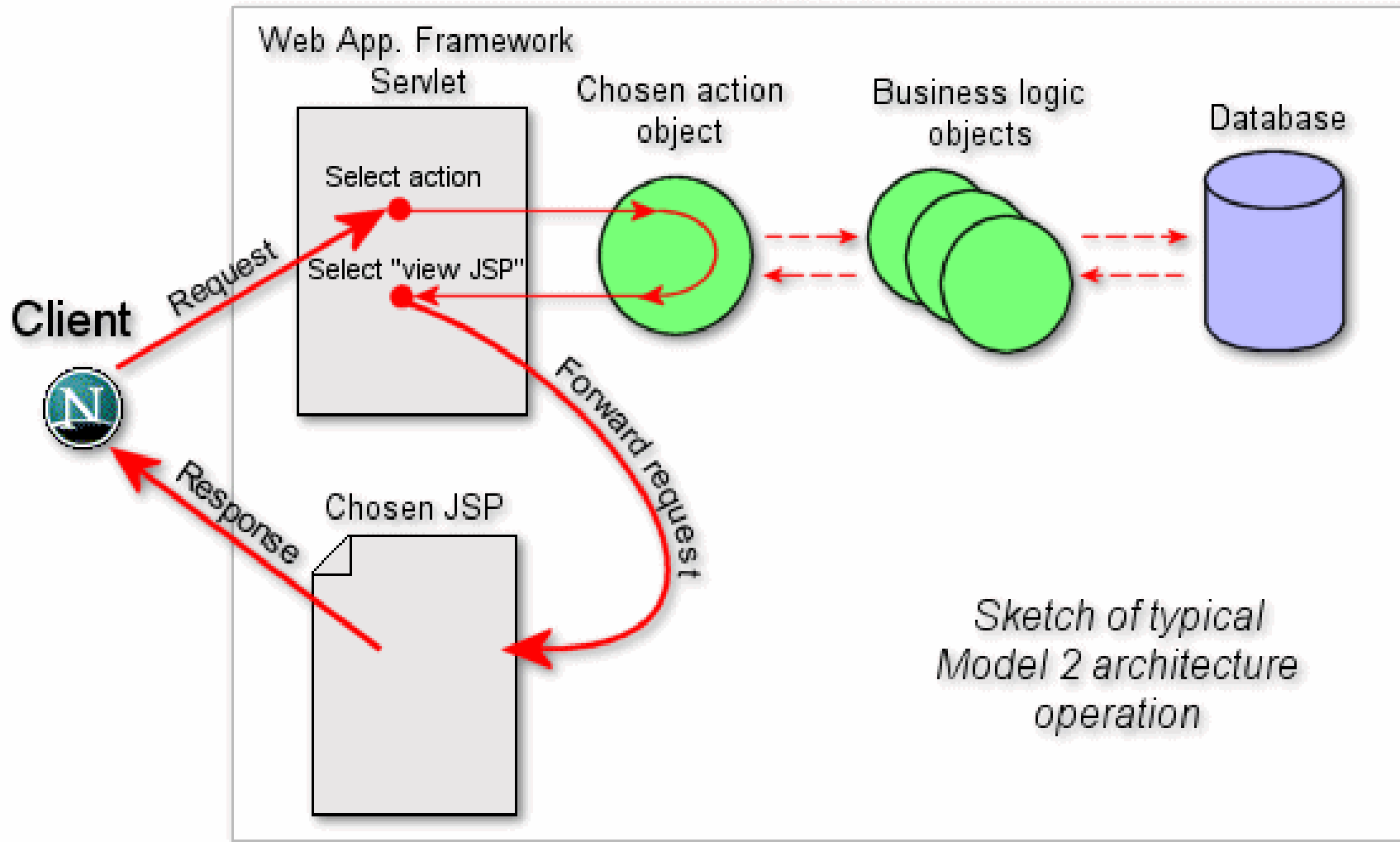
- Nutnosť generovať kvôli malej zmene celú stránku
- Žiadna dynamika komponentov

MVC – Design pattern pre Web

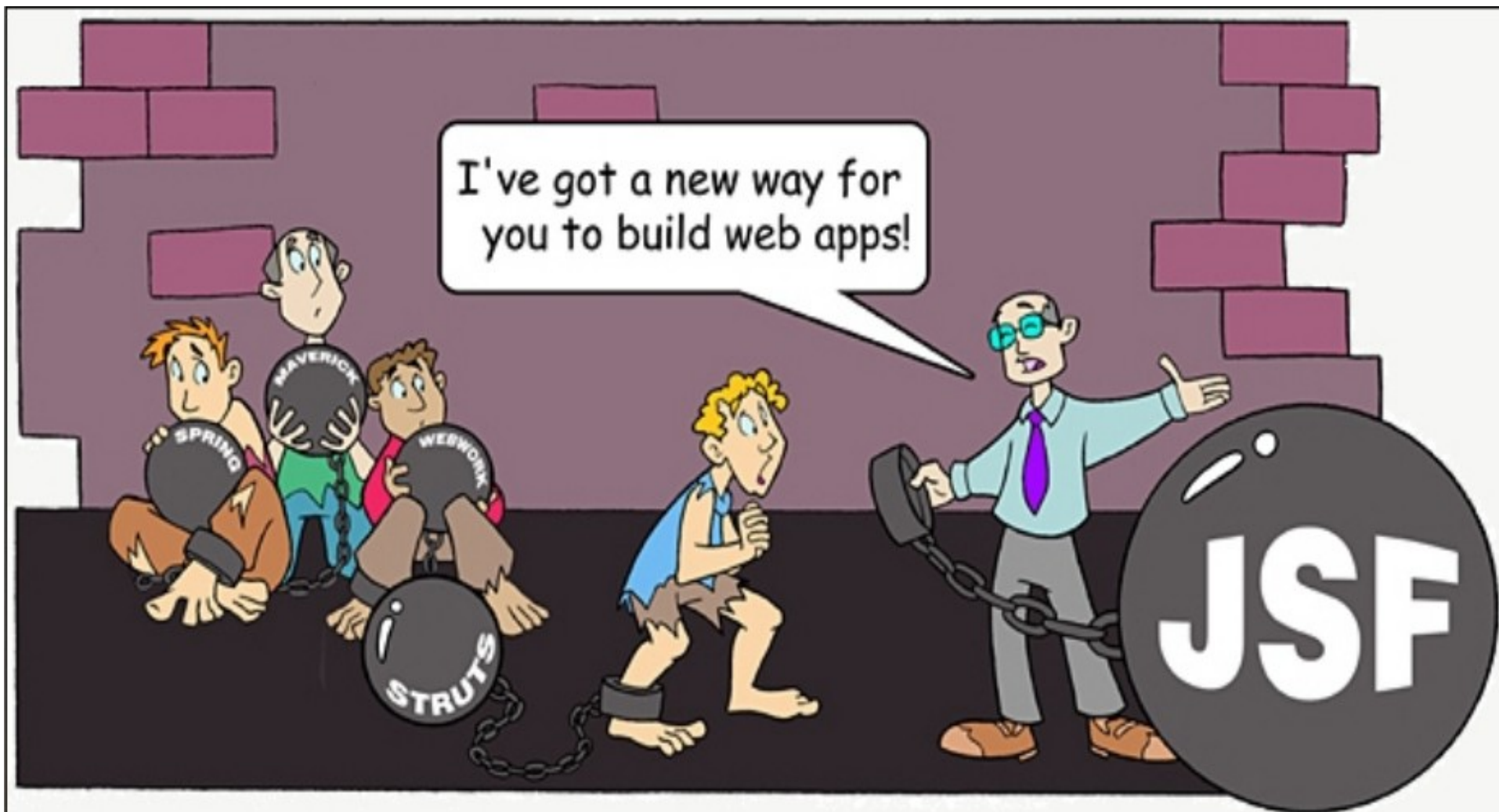
- Preferovaný pattern pre server side REST
- Rozdeľuje aplikáciu na 3 vrstvy
 - **model** – *biznis proces vrstva*
 - modeluje dáta a biznis procesy
 - **view** – *prezentačná vrstva*
 - zobrazuje výsledok biznis logiky (model)
 - **kontroler** – *riadiaca vrstva*
 - spojenie medzi užívateľskou interakciou a biznis procesmi v pozadí
 - zodpovedný za výber reprezentácie (view)



Server



REST – Súčasná situácia na poli Webových frameworkov



Novodobé Web 2.0 aplikácie

- **RIA** – Rich Internet Application – dôraz na tučnú funkcionálnosť
- Techniky sú zvyčajne **Ajax-based** – **Dátový zdroj pre AJAX sú RESTovské webové služby!**
- **Cascading Style Sheets** separuje prezentáciu od obsahu
- **Formáty, Mikroformáty** rozširujú web stránky o novú prídavnú sémantiku

Formáty: XML, XHTML

- XML – eXtensible Markup Language
 - <http://www.w3.org/XML/>
- Najznámejší jazyk na popis dát
- Ľahko párovateľný a rozšíriteľný
- Popis dát spolu s dátami
- **REX: REST-Enabled XHTML**
- **Mikroformáty** – xoxo, Atom, Geo, hCalendar, hCard, hReview, hResume, ...

Formáty: JSON

- **JSON** - JavaScript Object Notation
 - <http://json.org/>
- Ľahko čitateľný
- Vytvorený pre komunikáciu v jazyku JavaScript
- Podmnožina jazyka YAML
- Odporúča sa použiť pre AJAX – výkon a priame spracovanie do natívnych typov JavaScriptu

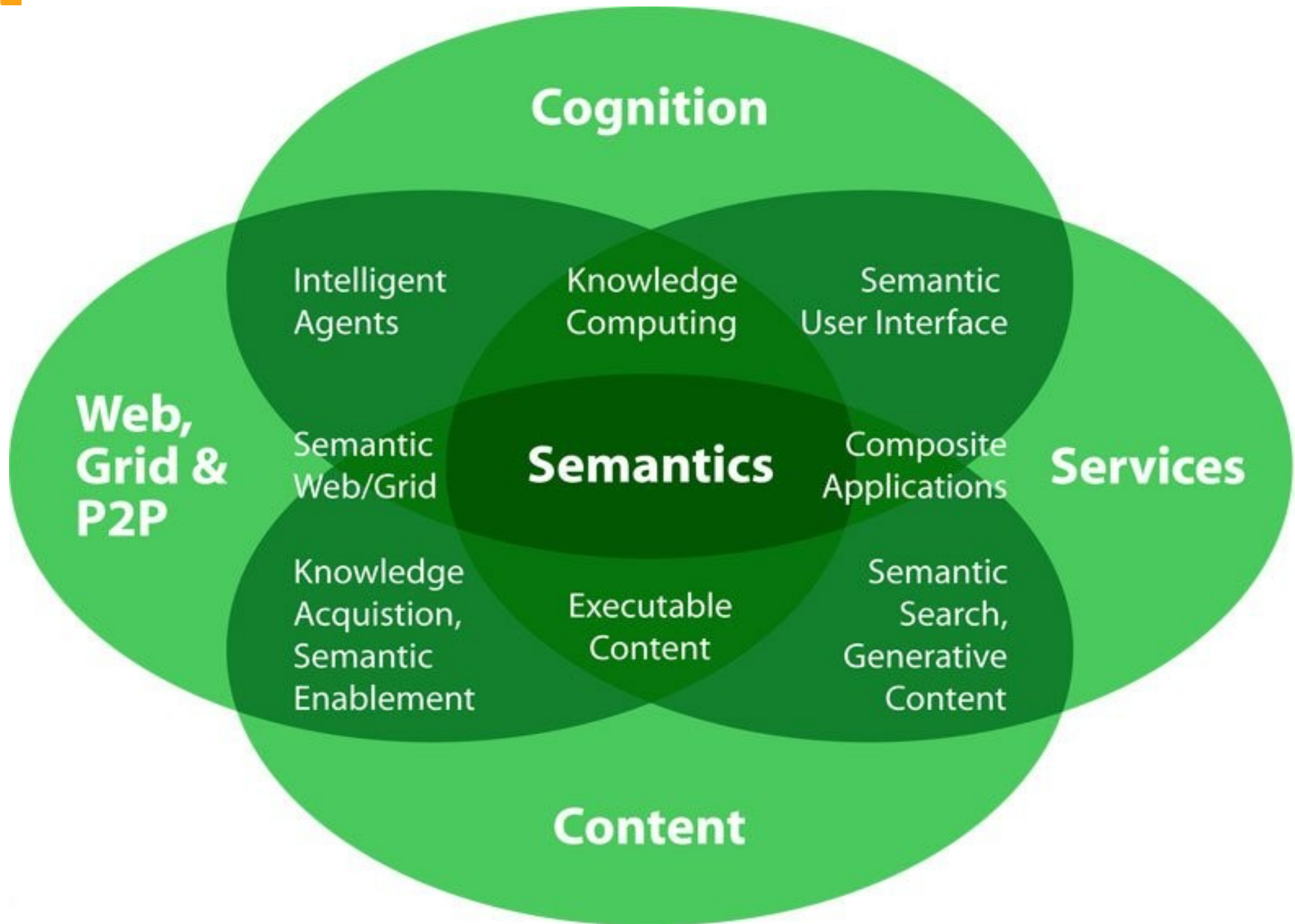
Formáty: **YAML**

- **YAML** – YAML Ain't Markup Language
 - <http://www.yaml.org/>
- Strojovo spracovateľný **formát na serializáciu dát**
- Ľahko **čitateľný** a pochopiteľný
- Vytvorený pre **skriptovacie jazyky** ako **Python, Ruby, Perl, ...**
- Podpora ukazovateľov, referencií, streamového spracovania dokumentov

Novodobé Web 2.0 aplikácie

- **Sémantika**, valídne XHTML
- **syndikácia, agregácia a notifikácia** pomocou RSS alebo Atom feedov
- **mashup** – spájanie obsahu z rôznych zdrojov, klient- a server-side
- **web log** – komentáre, zápisky, udalosti
- **wiki** – podpora užívateľom generovaného obsahu
- **CMS** – správa obsahu

Aggregators Wikis Folksonomy User Centered Joy of Use
Blogs Participation Six Degrees Usability Widgets
Pagerank XFN Social Software FOAF Browser
Recommendation Sharing Collaboration Perpetual Beta Simplicity AJAX
Videocasting Podcasting Audio IM Video Design
Convergence Web 2.0 CSS Pay Per Click
UMTS Mobility Atom XHTML SVG Ruby on Rails VC Trust Affiliation
OpenAPIs RSS Semantic Web Standards SEO Economy
OpenID Remixability REST Standardization The Long Tail
DataDriven Accessibility XML
Modularity SOAP Microformats Syndication



Známe REST, RIA, Web 2.0 aplikácie

- Webové aplikácie ponúkané firmami:
 - **Google**
 - <http://code.google.com/webtoolkit/>
 - **Yahoo!**
 - <http://developer.yahoo.com/javascript/howto-ajax.html>
 - **Amazon**
 - <http://developer.amazonwebservices.com/>

Princípy tvorby Web aplikácií

- **Architektúra** je dôležitá, **nie technológia!**
- Na webe preferujeme **otvorené protokoly** a **otvorené formáty**
- **Sémantika** dáva webu zmysel
- **REST** – nezávislosť riešenia na platforme
- Dovoľme riešeniu rásť – **škálovateľnosť** a **modularita**
- Pri návrhu riešení dodržiame „**Design Principles**“



Ďakujem

Otázky