Integrácia – SOA – REST

REST vs. SOAP (WS-*)

Peter Rybár
peter.rybar@centaur.sk

CENTAUR

Obsah

- **Integrácia**
- **SOA**
- **REST** vs. SOAP (WS-*)
  - SOAP (WS-*)
  - **REST**
    - REST Anti-patterns
    - REST Patterns
    - REST pre a proti
- Otázky

Integrácia

- Systémová **Integrácia**
  - Spájanie **komponentov** do jedného **systému** aby fungoval ako **celok**.
- **Integrácia v IT**
  - Spájanie **rôznych** softvérových aplikácií **fyzicky a funkčne**.
- Enterprise **Application Integration** (EAI)
  - Použitie **softvéru** a **architektonických princípov** na integráciu **sady aplikácií**

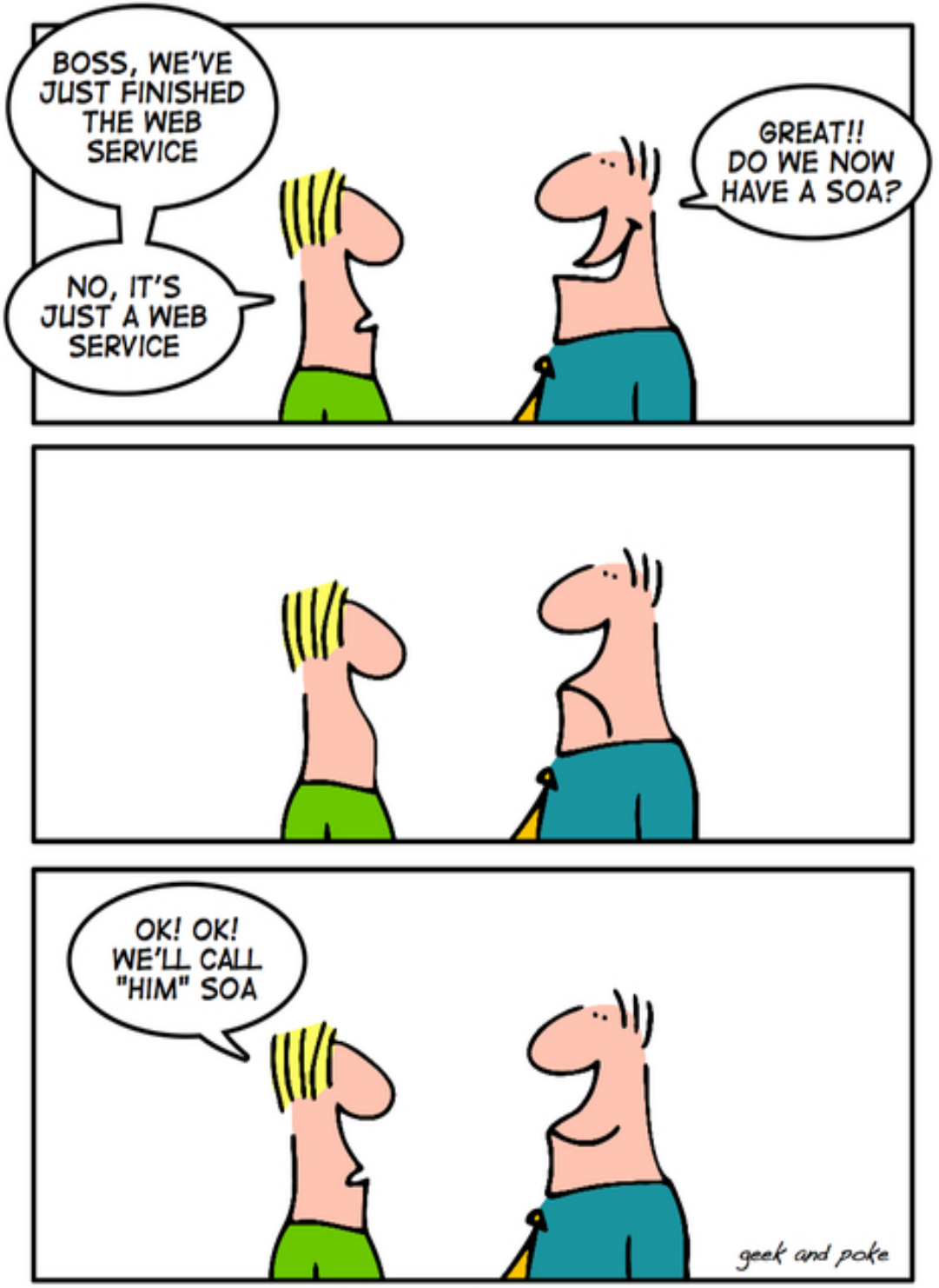
  ⇒ **Architektúra**

SOA

Servisne **O**rientovaná **A**rchitektúra

SOA

- **Čo je to SOA?**
  - Na túto otázku **je ťažké jednoznačne odpovedať**.
  - **Rôzni ľudia** chápu tento pojem **rôznym spôsobom**.



HOW TO GET A SOA

SOA

- **SOA** – **architektúra** orientovaná na **služby**
  - **Pracovná jednotka – Služba**
  - Výsledkom služby je **zmena stavu** poskytovateľa alebo konzumenta
  - Poskytovateľ aj konzument služby:
    - **Softvéroví agenti**

SOA a služby

- **Služba**
  - **Endpoint**
    - Adresa služby (**URI**).
  - **Kontrakt**
    - **Rozhranie**, kolekcia všetkých správ podporovaných službou.
  - **Správa**
    - Základná jednotka komunikácie
    - HTTP správa, SOAP správa, JMS, SMTP, ...

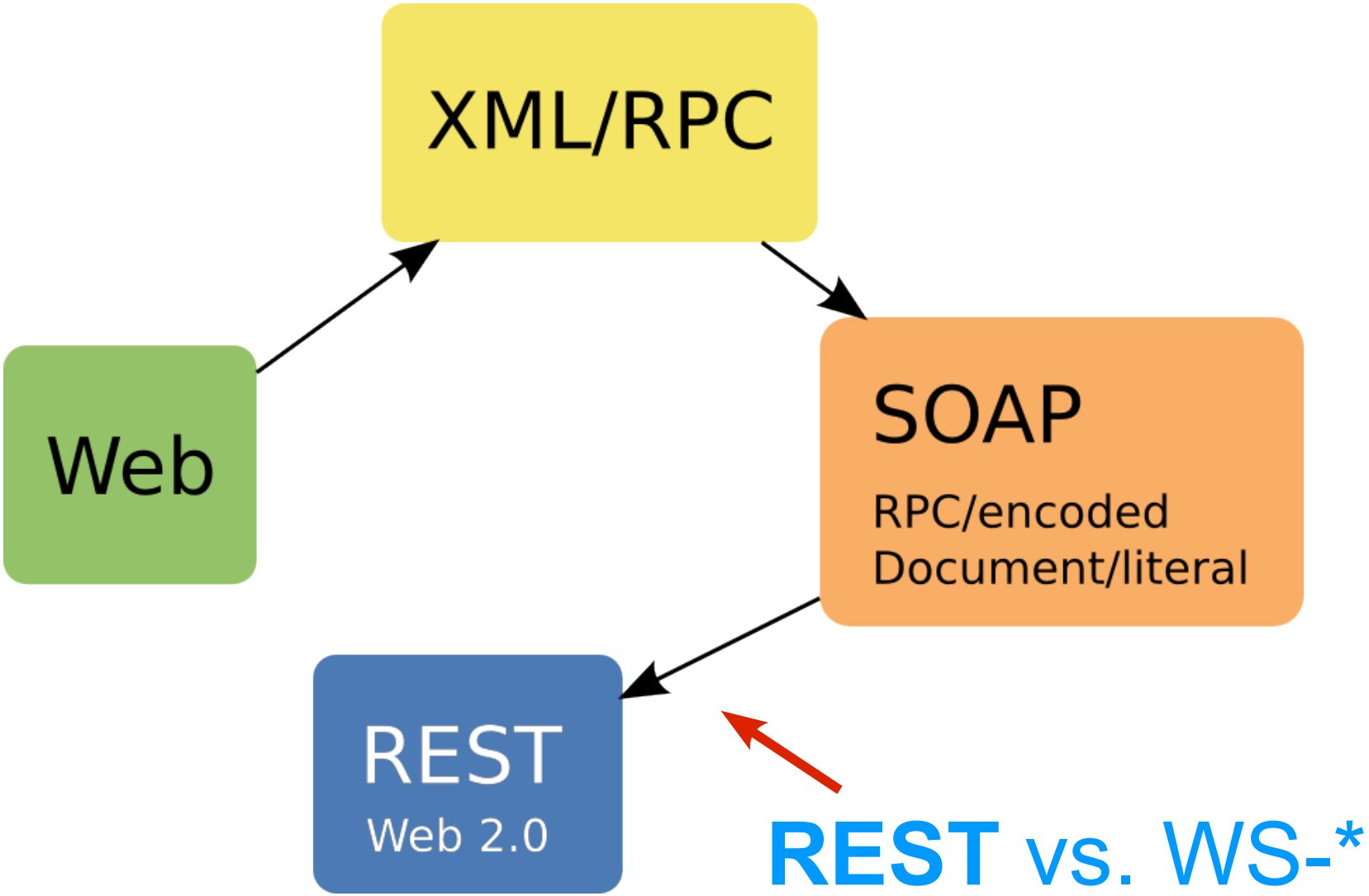
SOA - Pravidlá

- **SOA musí spĺňať** nasledujúce pravidlá:
  - **Správy sú popisné**
    - Hovoria čo sa má vykonať, nie ako sa to má vykonať.
  - **Otvorený formát správ**
    - Formát ktorý chápu všetky zúčastnené strany.
  - **Rozšíriteľnosť správ a rozhraní**
    - Inak by poskytovateľ a konzument služby boli viazaní na určitú verziu.
  - **Objaviteľnosť služieb**
    - Flexibilný mechanizmus, **nemusí** sa jednať o **centralizovaný register**.

SOA – implementácie

- **WEB** (1990)
- CORBA (1991)
- XML-RPC (1998)
- **WS-*** (1998)
  - SOAP – RPC/literal
  - SOAP – Document/literal (2001)
- **REST** (2000)

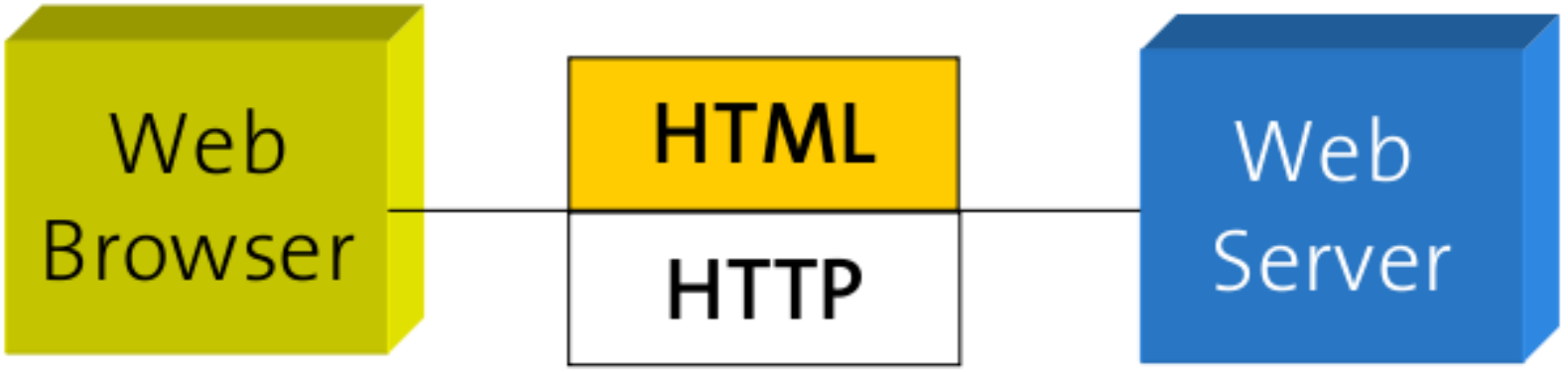
SOA – Web implementácie



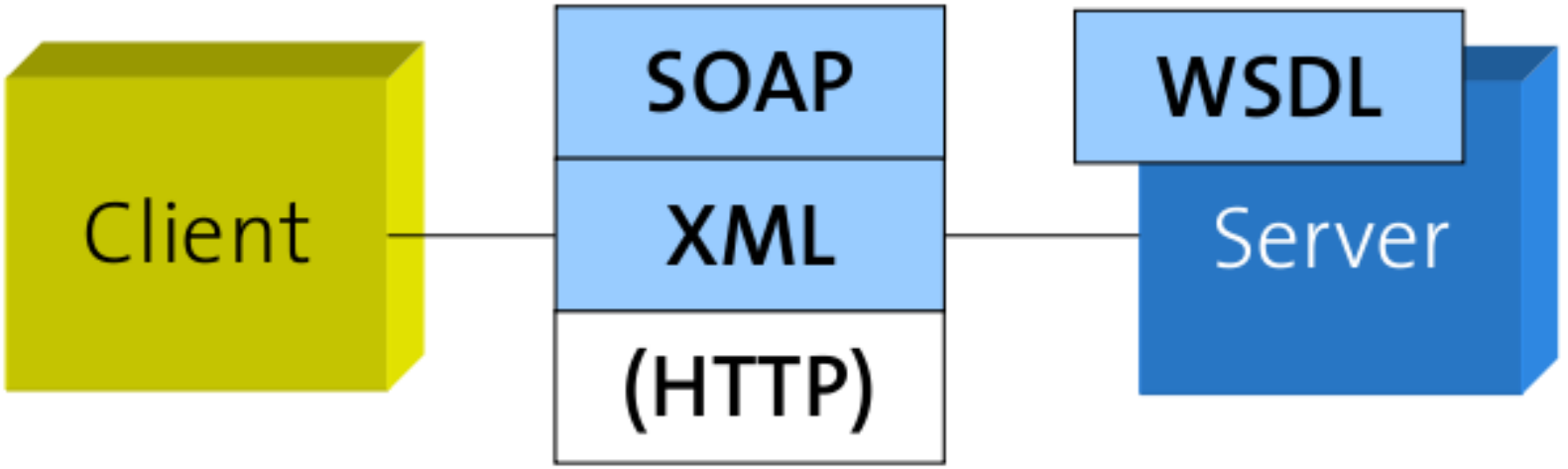
REST vs. SOAP (WS-*)

Ako priviesť Web naspäť do Web Služieb

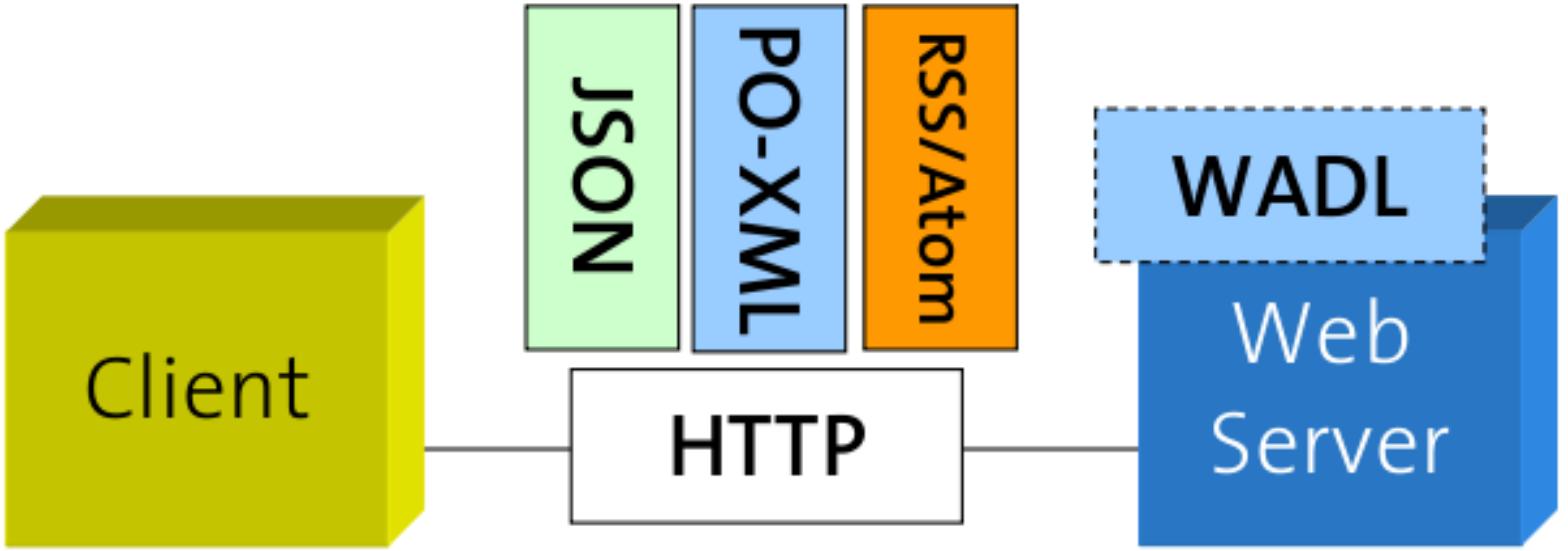
Web Sites (1992)



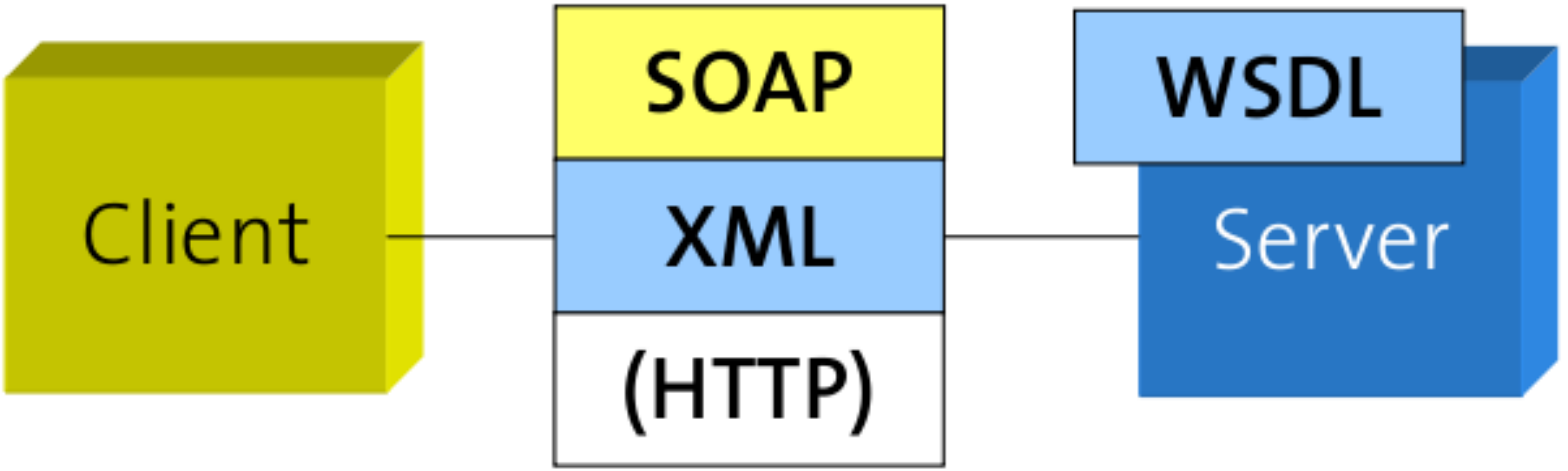
WS-* Web Services (2000)



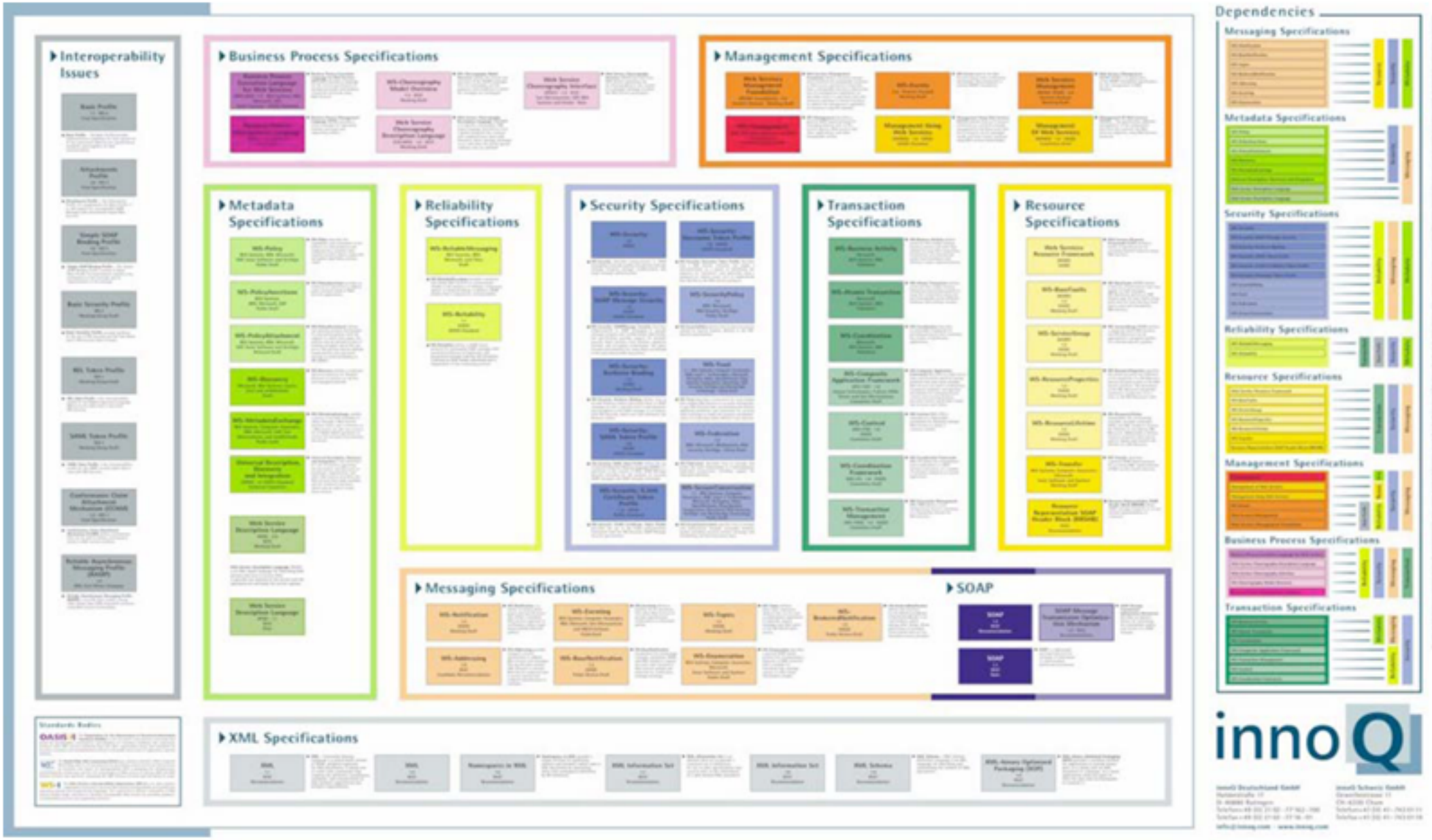
RESTful Web Services (2007)



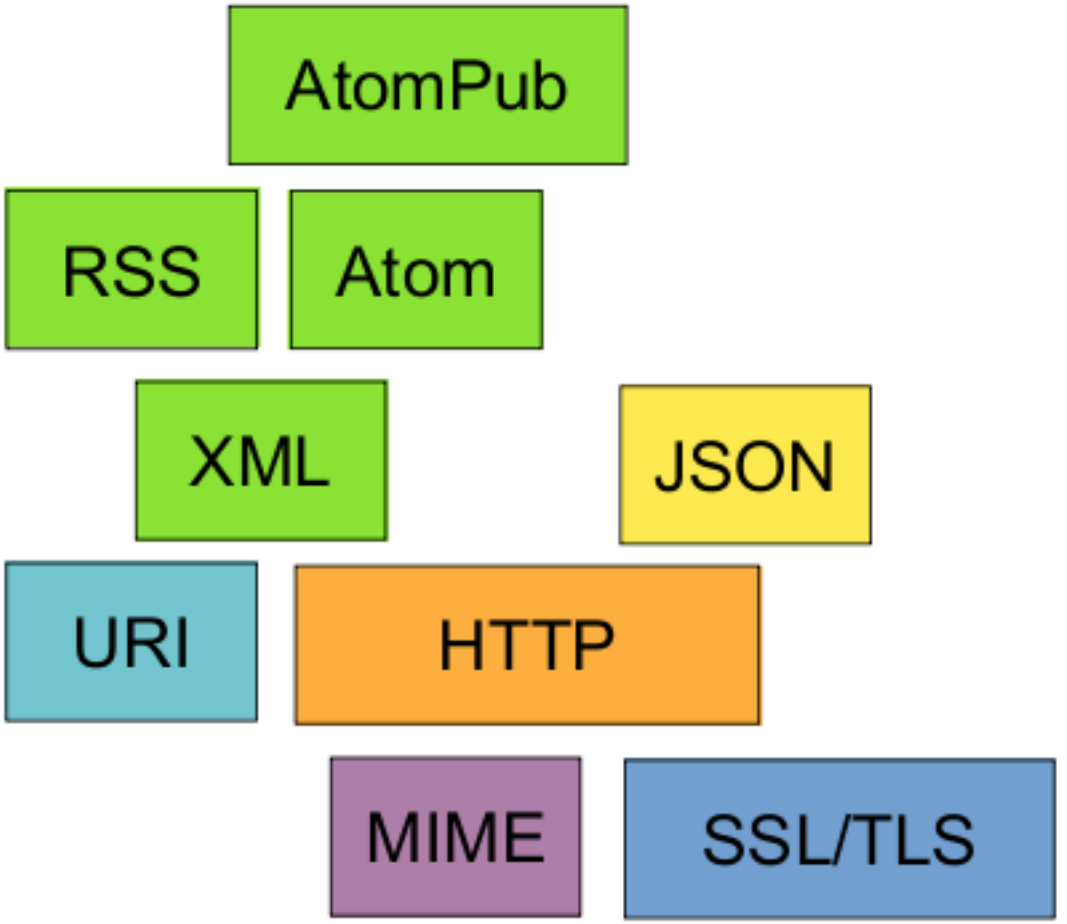
WS-* Web Services (2000)



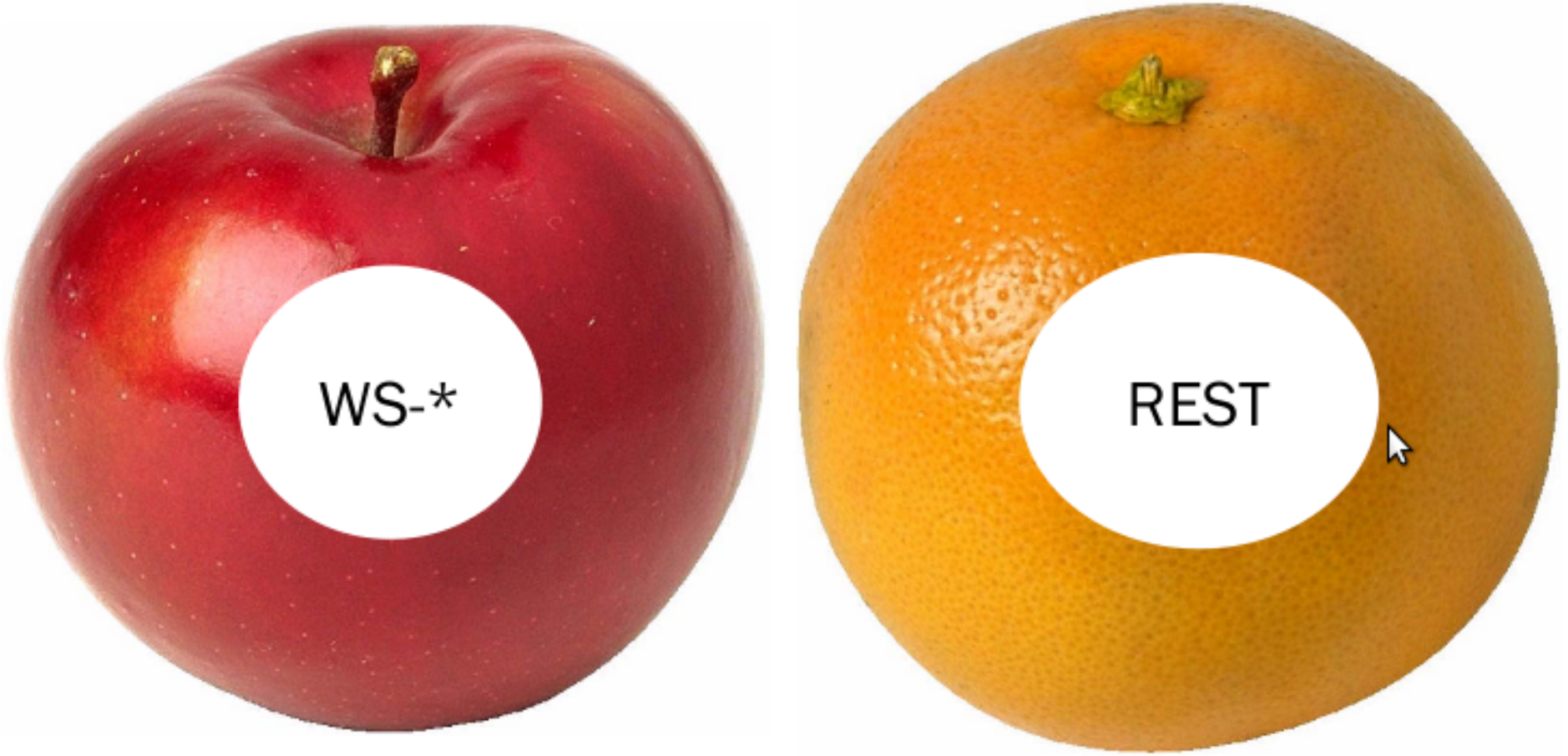
WS-* Standards stack



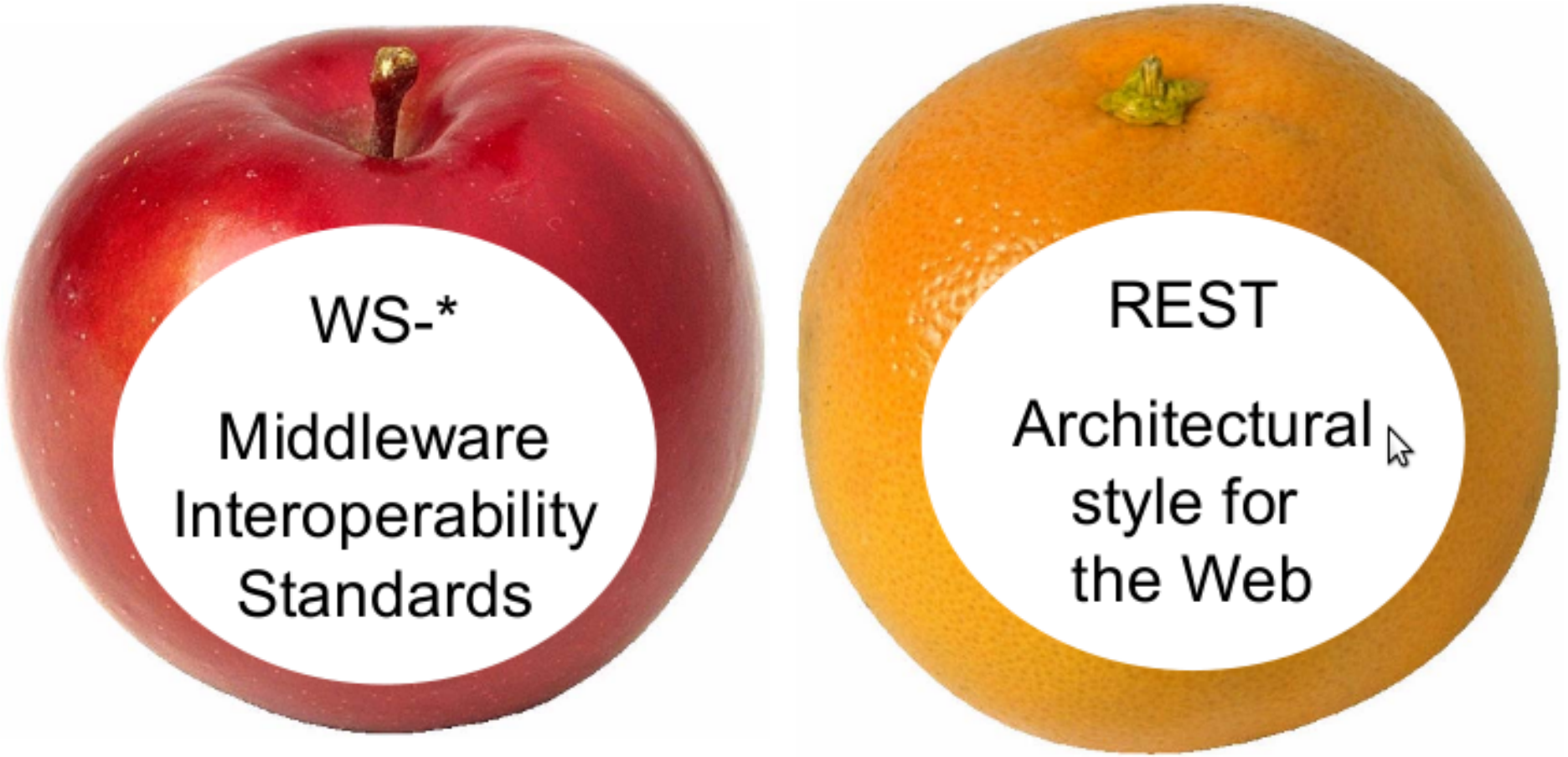
REST Standards stack



REST vs. WS-*



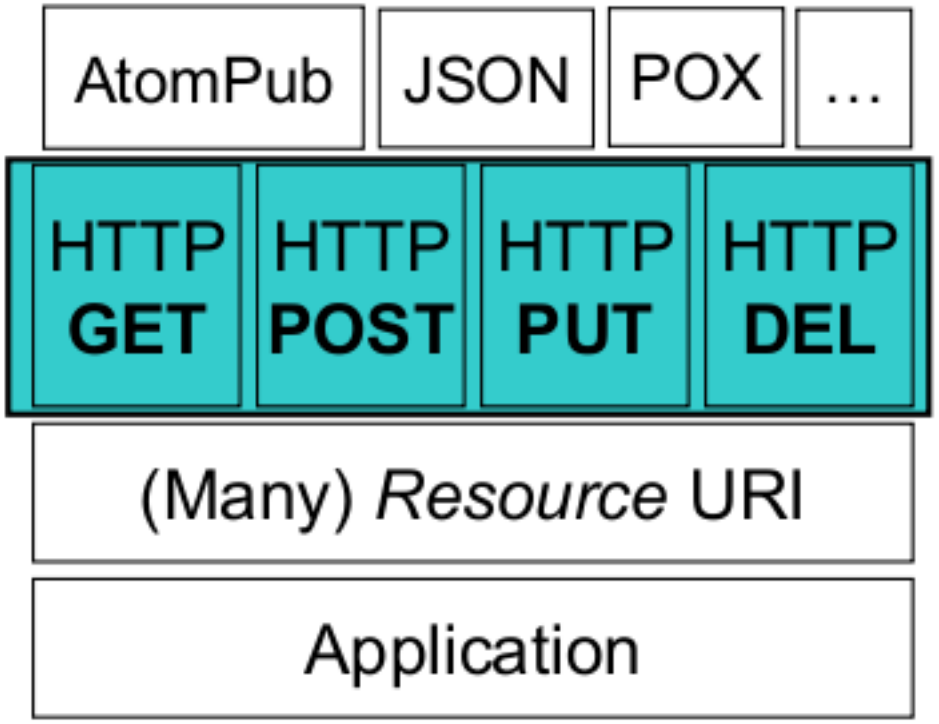
REST vs. WS-*

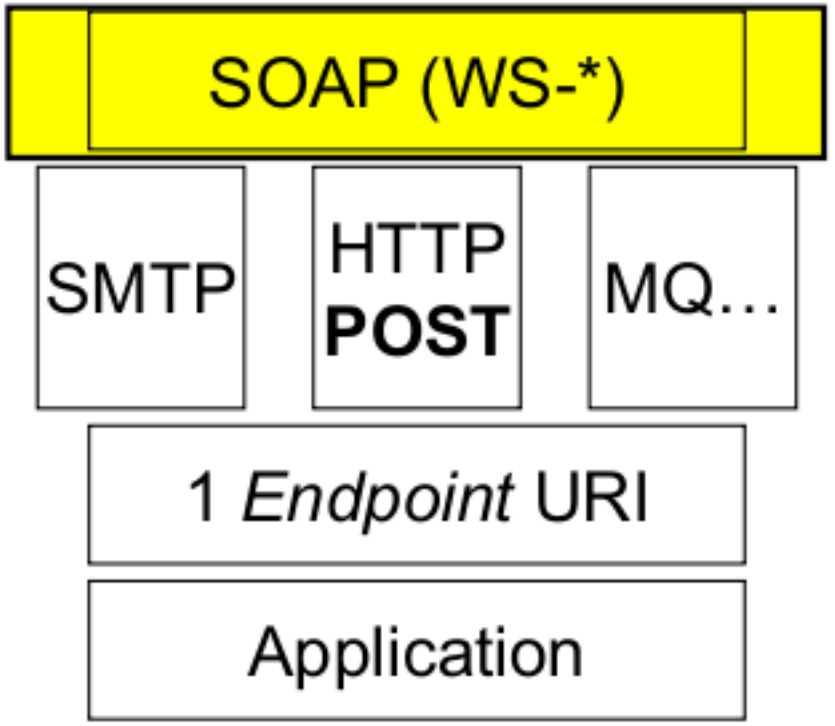

WS-*
Middleware
Interoperability
Standards
REST
Architectural
style for
the Web

REST vs. WS-*

- Web – svet globálne prístupných informácií
- Aplikácie publikujú dáta na Webe prostredníctvom URI

- Web – univerzálne transportné médium pre informácie
- Aplikácie vzájomne interagujú - zostávajú mimo Web





SOAP (WS-*)

SOAP

- **Čo je SOAP?**

- **Započal v 1998 ako remote procedure call protocol**
  - Mal prejsť cez firewall (Simple Object Access Protocol)
  - Prvé verzie SOAP:
    - Definovali **extenzibilný message formát** založený na **XML**
    - Serializácia do **RPC** parametrov v XML
    - Transportný protokol HTTP prechádzal cez firewall

- **Posledných 7 rokov** vývoja SOAP:
  - Flexibilnejší prenos XML dát – document / literal
  - **Protocol agnostic** – nezávislosť na transportnom protokole

SOAP je iba SOAP

- **SOAP**

- Protokol “framework” na dosiahnutie potrebnej **interoperability** medzi **message-based middleware** nástrojmi

- Priemyselný štandard
  Keď dvaja robia to iste, nemusí to byt vždy to isté

Na čo sú SOAP WS dobré?

- **SOAP/WSDL** – **komunikačná technológia** poskytujúca interoperabilitu, spoločnú bázu pre messaging aplikácie pracujúce spoločne nad HTTP
- Legacy reuse a integrácia
  - Zastaralé systémy nie sú Web-friendly (nepoužívajú HTTP) – používajú multicast, RPC/RMI binárne protokoly, asynchrónny messaging, dávkové hromadné transféry
- Flexibilita a adaptácia
  - Ten istý interface je možné naviazať na rôzne protokoly podľa toho ako sa menia biznis a technologické požiadavky

SOAP Web Services slabosti

- **Priveľká komplexita**
- Problematický štandardizačný proces

- Nezačína sa už podobať na niečo? **CORBA?**
- Kedy už SOAP WS interoperability **začne konečne skutočne fungovať?**
- Naozaj potrebujeme platiť za použitie XML **nízkym výkonom**?

SOAP Web Services slabosti

- **Skryté problémy**
  - **Absencia architektonickej koherencie**
  - **Fragmentácia**
  - **Design by committee**
  - **Feature Bloat**
    - spájanie konkurujúcich si špecifikácií
  - **Absencia referenčnej implementácie**
  - **Štandardizácia nad štandardami** (WS-I)

SOAP slabosti ?

- Show me the interoperable, **full and free implementations of WS-*** in **Python, Perl, Ruby and PHP**.

  You won't see them, because there's **no intrinsic value in WS-*** unless you're trying to **suck money out of your customers**.

  Mark Nottingham, ex BEA, Principal Technical Yahoo!

  http://www.mnot.net/blog/2006/05/10/vendors

REST

Čo je **REST** ?

- **REST**
  - **Re**presentational **S**tate **T**ransfer
  - **Roy Fielding PhD, 2000**
  - **Architektonický štýl**

- REST stojí na princípoch, ktoré umožňujú HTTP byť tak dobre škálovateľný
- REST „je Web“ – nie je tunelovaný cez Web

REST – Princípy

- **Princípy**:

1. **URI** – identifikácia zdroj (všetko je zdroj)
2. **CRUD** – jednotné rozhranie pre všetky zdroje
3. **Reprezentácie** – rôzne podoby správy (MIME)
4. **Bezstavovsť** – umožňuje škálovateľnosť
5. **Hypermédiá** – prelinkovanie médií/reprezentácií

REST – Princípy

- **URI**:
  - Všetko sú **zdroje** ↔ **ROA**
  - Zdroje sú identifikované **URI**
  - Zdroje sú **podstatné mená**

http://example.net/**customer**

http://example.net/**car**

http://example.net/**shopping-cart**

REST – Princípy

- **CRUD**:
  - **jednotné rozhranie** pre prácu so zdrojmi
    - POST – **C**reate, vytvára nový zdroj
    - GET – **R**ead, bezpečná operácia
    - PUT – **U**pdate, idempotentná operácia
    - DELETE – **D**elete, idempotentná operácia

REST – Princípy

- **Reprezentácie:**
  - Ku zdroju pristupujeme cez reprezentácie
  - **Jeden zdroj – viacero reprezentácií**

REST – Princípy

- **Reprezentácie**:
  - Ku zdroju pristupujeme cez reprezentácie
  - **Jeden zdroj – viacero reprezentácií**
    - text/html, image/png, application/pdf
  - Typ reprezentácie je v HTTP hlavičke
    - Request – **Accept**
    - Response – **Content-Type**

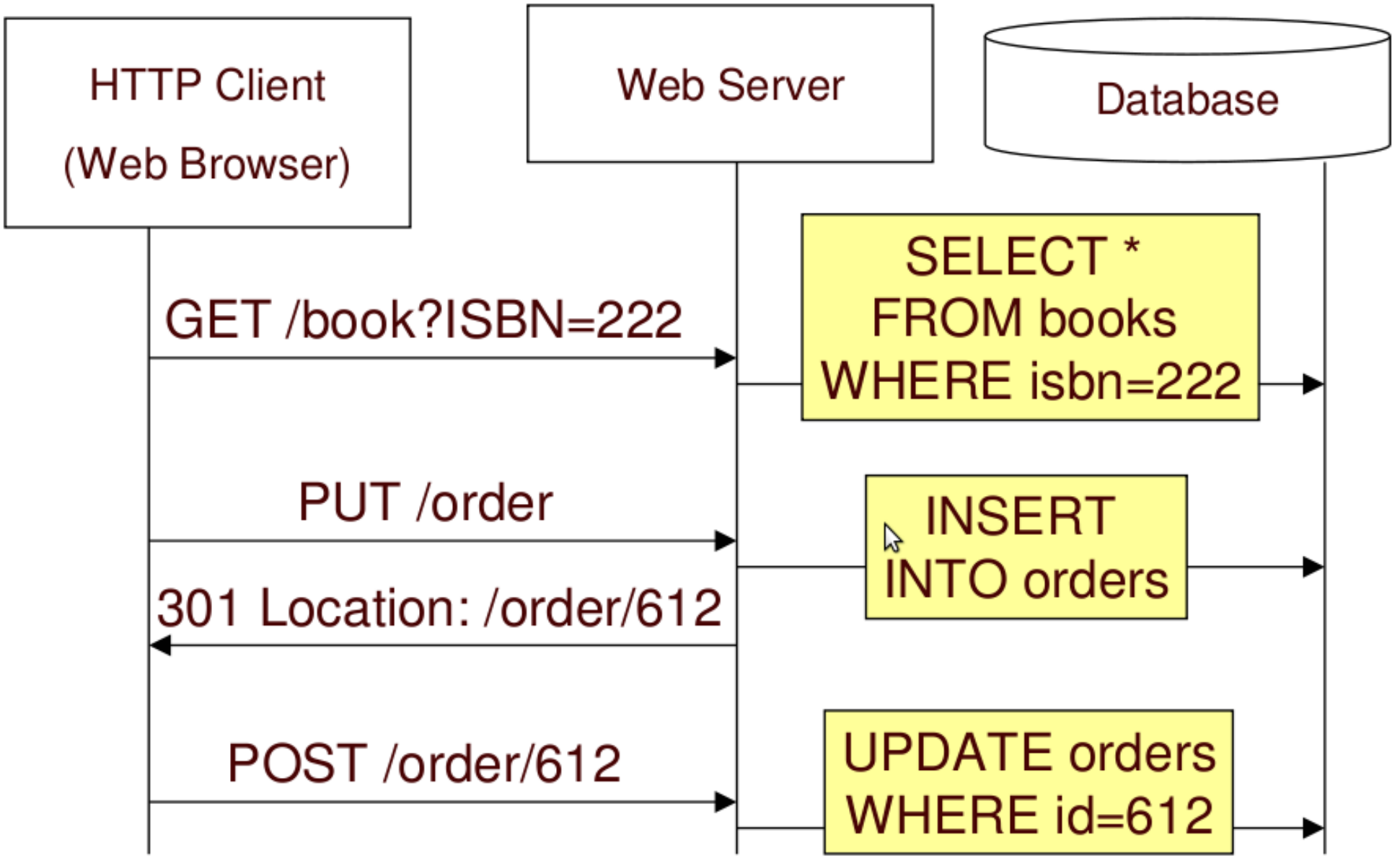
REST – Princípy

- **Bezstavovsť**:
  - **HTTP server nepozná stav**
    - **Neexistuje HTTP Session!**
  - Klient udržuje stav cez **linky**
    - Funguje **back button**
    - Funguje **bookmarkovanie**
  - Škálovateľný systém

REST – Princípy

- **Hypermédiá:**
  - Reprezentácie zdrojov – **hypermédiá**
  - **Hypermédiá – obsahujú linky na iné médiá**
  - Zmena stavu klienta – cez linky v hypermédiách
  - Linky poskytuje server

RESTful Web Application Example



High REST vs. Low REST

- **Best practices** sa rôznia:
  - **Low REST**
    - HTTP **GET** pre idempotentné požiadavky, **POST** pre všetko ostatné
    - Odpovede v ľubovoľnom MIME Type (**XHTML**)
  - **High REST**
    - Doporučované používanie „pekných“ URI
    - Plné využitie **GET, POST, PUT, DELETE**
    - Využitie dátových MIME Type formátov **JSON, YAML, ATOM**

REST Anti-Patterns

REST Anti-Patterns

- **Čo REST nie je**:
  - **POX** (Plain Old XML) **bez SOAP obálky**
    - Zneužíva Web rovnako ako WS-*
  - Použitie **HTTP GET/POST** pre **RPC**
    - **HTTP je aplikačný protokol**
    - HTTP nie transportný protokol pre volanie vzdialených metód

REST Anti-Patterns

- **Tunelovanie cez HTTP GET**
  - http://example.net/api?method=find&id=37
  - http://example.net/api/find/37

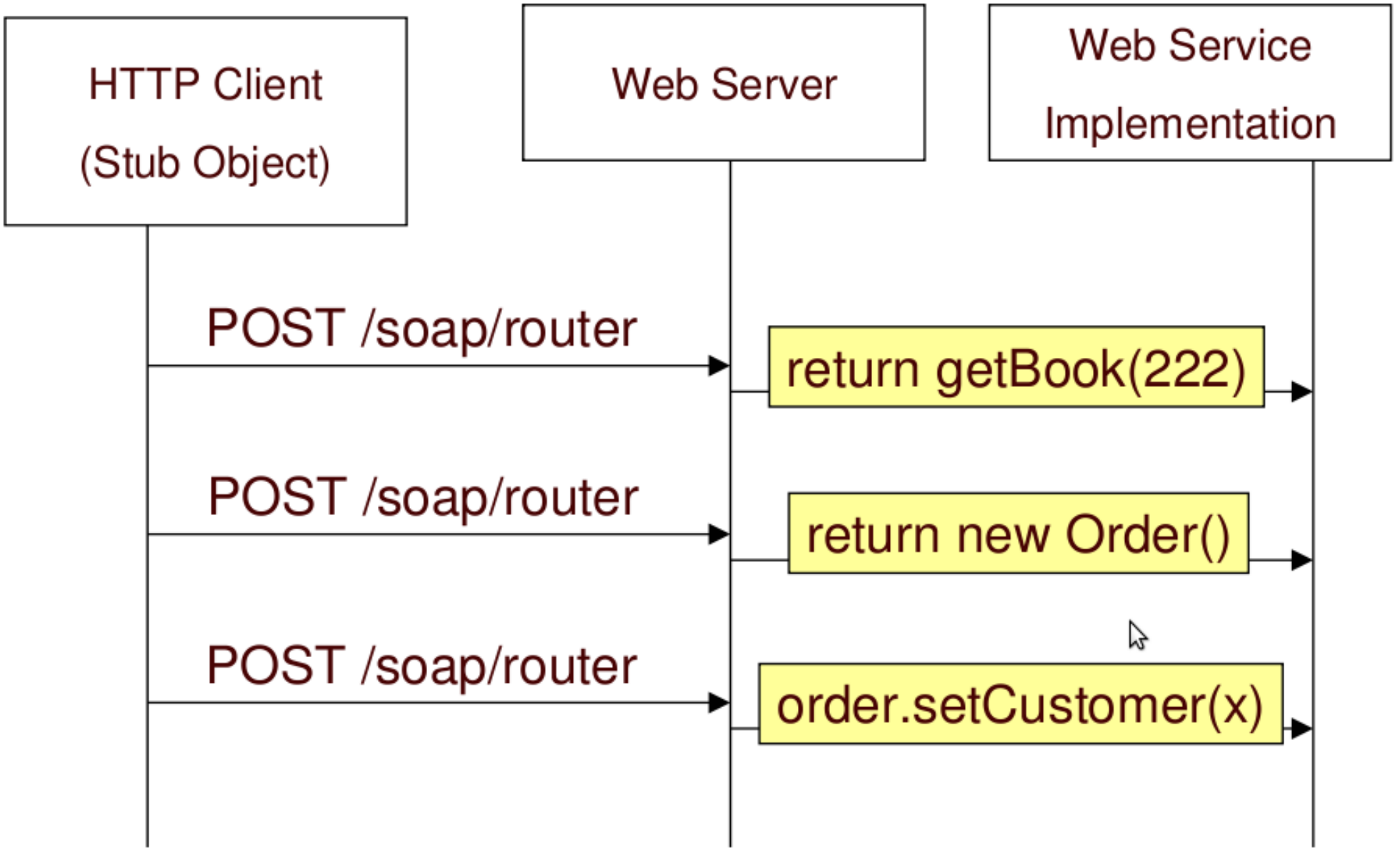
- **Tunelovanie cez HTTP POST**
  - POST http://example.net/api/

```
<method name="find">
    <id>37</id>
</method>
```

TUNNELING

In real life it can prove your courage.
On the internet it proves you're a
douche bag.

Web Service Example (from REST perspective)



REST Anti-Patterns

- Ignorovanie HTTP Cache
- Ignorovanie HTTP Status Codes
- Nesprávne použitie Cookies
- URL nereprezentuje aplikačný stav
- Opomínanie hypermédií, prelinkovania
- Ignorovanie MIME Types – application/xml
- Opomínanie samo-popisnosti

REST Patterns

REST Patterns

- **Coollection Resource** – súvisiace resources spájať do skupín
  - **Riešenie**
    - Zmeniť položky kolekcie na resources
    - Použiť linky ako odkazy na prvky kolekcie
  - **Príklad**
    - GET http://example.net/customers

```
<customers>
    <customer>
        <name>Company XY</name>
        <link type="text/html" href="/customers/731" />
    </customer>
</customers>
```

REST Patterns

- **Paging Coollection** – výsledky sú príliš veľké aby sa získali naraz
  - **Riešenie**
    - Zmeniť položky kolekcie na resources
    - Vracať subset kolekcie
    - Uviesť linky na nasledujúci a predošlý subset
  - **Príklad**
    - GET http://example.net/customers

```
<customers>
    <link rel="prev" href="/customers/?page=1" />
    <link rel="next" href="/customers/?page=3" />
    <customer>
        ...
    </customer>
</customers>
```

REST Patterns

- **Špecializované Pohľady** – potrebujeme rôzne pohľady na resorce
  - **Riešenie**
    - Vytvoriť ďalšie resources zodpovedajuce iným pohľadom
    - Uviesť linky na resorces ďalších úrovní
  - **Príklad**
    - http://example.net/customers
    - http://example.net/customers?region=3
    - http://example.net/customer-addresses
    - http://example.net/customers/new?limit=20
    - http://example.net/orders/2009/09/02/120-350

REST Patterns

- **Vytváranie Resources** – resorces sú vytvárané konkurenčne a vyžadujú unikátne URI
  - **Riešenie 1**
    - **POST** dáta na URI kolekcie, ktorá bude dáta obsahovať
    - Server vytvorí URI/UUID pre nový resource
    - **201 HTTP Response Code** + **Location HTTP Header** s vygenerovaným URI
  - **Riešenie 2**
    - Klient vytvorí UUID
    - **PUT** dáta na URI/UUID

REST Patterns

- **Notifikácie** – klient by mal vedieť o zmenách na resorces
  - **Riešenie**
    - Definonvať náhľad agregujúci zmeny
    - Vystaviť RSS alebo Atom feed
    - Overiť chache control headers
  - **Techniky**
    - Polling
    - Long polling

REST Patterns

- **Content Negotiation (conneg) extensions**
  – podpora linkovania špecifických formátov
  - **Riešenie**
    - Vytvoriť všeobecný resource s podporou volby formátu
    - Vytvoriť rôzne resources pre rôzne reprezentácie odlíšené príponou
  - **Príklad**
    - http://example.net/customer/**3715**
    - http://example.net/customer/**3715.xml**
    - http://example.net/customer/**3715.pdf**

REST Patterns

- **Canonical representation**

  – zabezpečiť zovšeobecnenú reprezentáciu vhodnú pre spracovanie a ľudskú interakciu

  - **Riešenie**
    - Poskytnúť štandardnú **HTML** reprezentáciu
    - Použiť **www-form-data** pre **jednoduché spracovanie**
    - Poskytnúť **HTML form** pre zadávanie požiadaviek

REST Patterns

- **External server Caching**
  - zjednodušiť implementáciu serverovej cache
  - **Riešenie**
    - Vyhodiť server chaching mechanizmy a implementácie
    - Produkovať cache-control HTTP headers
    - Predradiť serveru HTTP cache – **proxy**

REST Patterns

- **Transakcie** – jedným requestom modifikovať viacero resources
  - **Riešenie**
    - Previesť transakcu na transakčný resource
    - Modifikovať transakčný resource – aj na viac krokov.
    - **Commit** relizovať HTTP **PUT** na transakčný resource

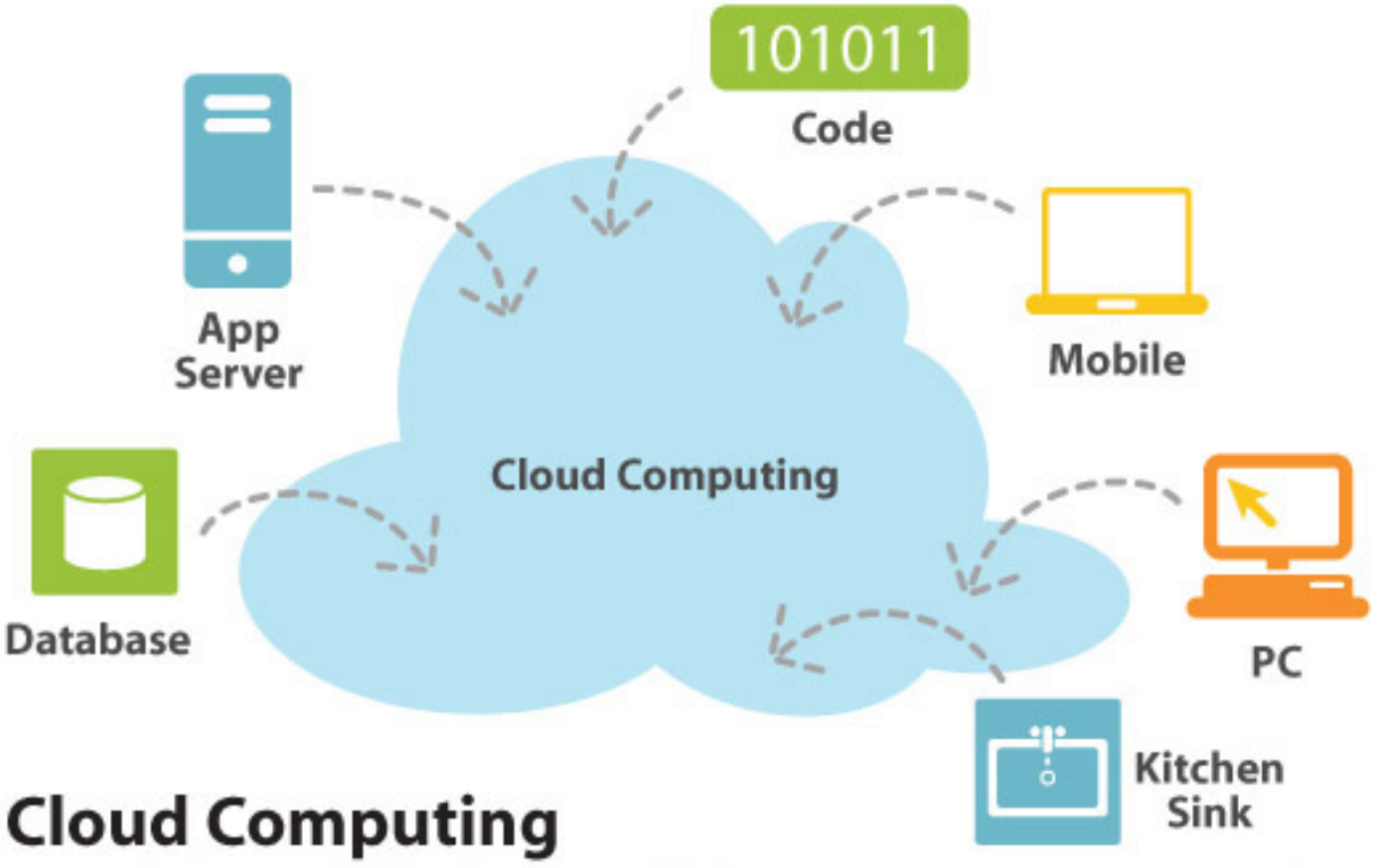
REST pre a proti

REST – Výhody

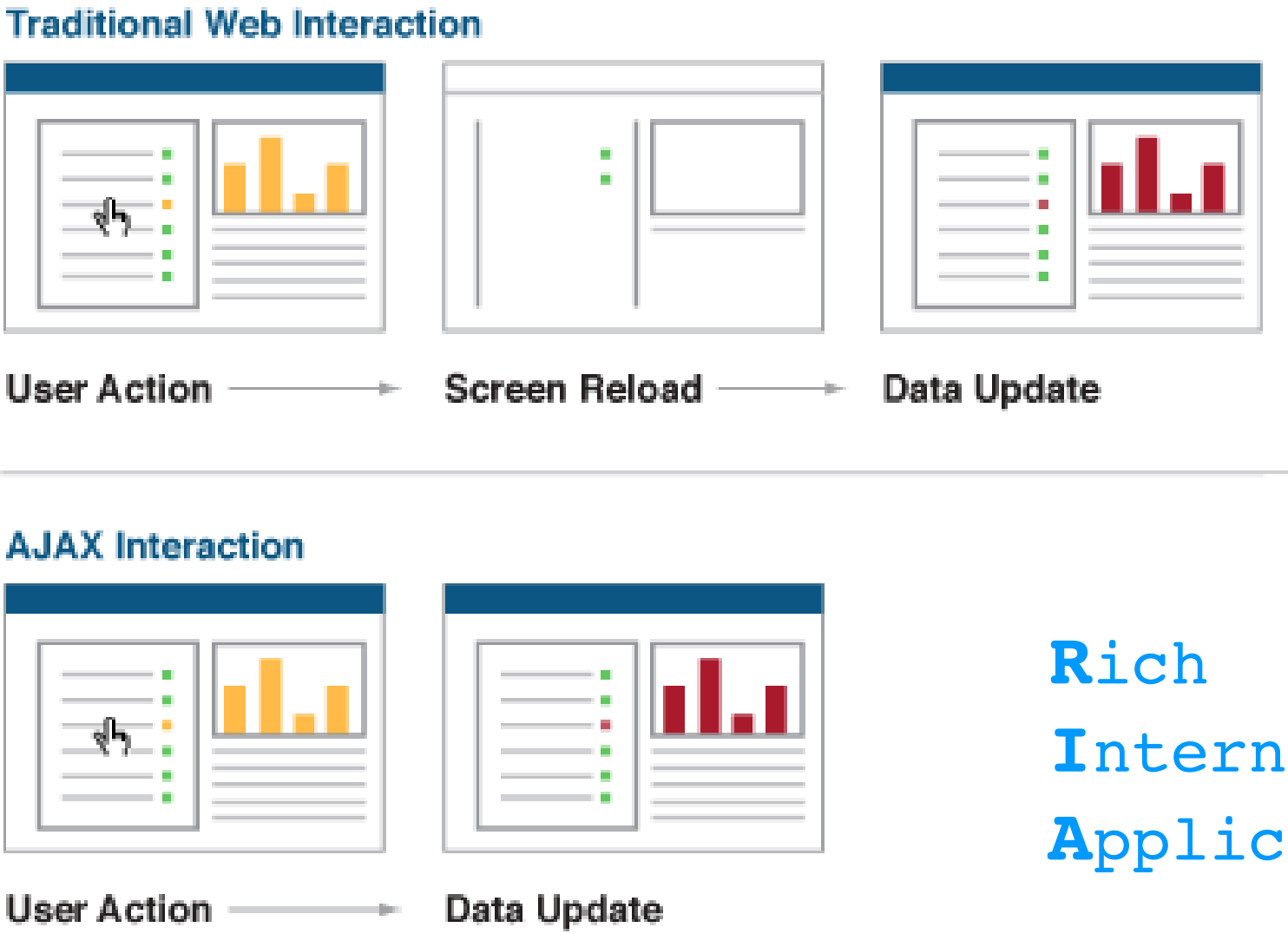
- **Jednoduchosť** => **Nízka cena**
  - Jednotné nemenné rozhranie **CRUD**
  - **HTTP** je **všadeprítomný** – povolený na FW
- **Bezstavová interakcia** => **Škálovateľnosť**
- **Tenká infraštruktúra** => **Ľahká adopcia**
  - Stačí Webový prehliadač
  - **Nie je potreba kupovať drahý WS-* middleware**

REST – Výhody

- **Fundamentálny prístup** => **SEO**
  - Silná podpora **Web 2.0 – Google, Amazon, ...**

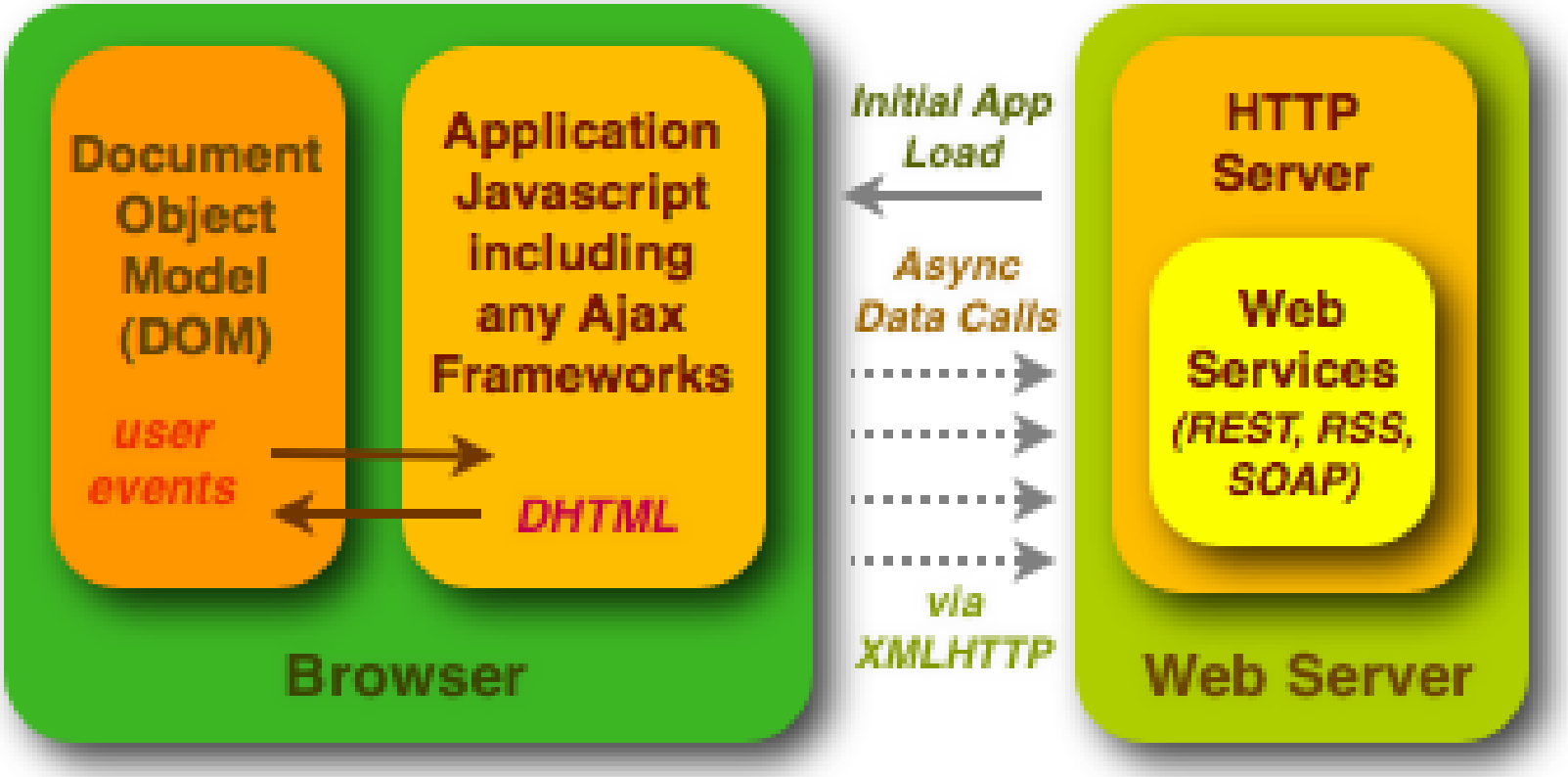


REST – Výhody


Traditional Web Interaction
User Action
Screen Reload
Data Update
AJAX Interaction
User Action
Data Update
Rich
Internet
Applications

Ajax

The organic and 100% open standards-based RIA model



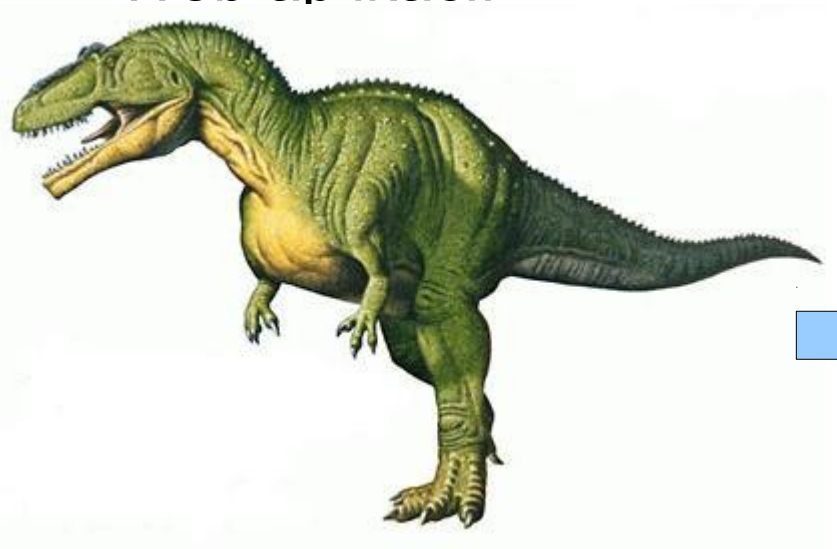
Source: http://hinchcliffe.org

REST slabé stránky

- **Nerozhodnosť** v použití High REST vs. Low REST
- Priamo neponúka riešenia nad rámec **HTTP/SSL**
- **Zdanlivá absencia štandardov**
  - Iných než URI, HTTP, XML, MIME, HTML, JSON, YAML
- **Sémantika/Syntax** je dosť neformálna
  - **user/human oriented**

REST – Nástroje

- **Zmena architektúry => Zmena nástrojov**
  - Dôraz na jednoduchosť, odľahčenie
  - Jednoduchý stack technológií
  - Jedny nástroje pre vývoj:
    - Web služieb
    - Web aplikácií



REST vs. SOAP (WS-*)

Porovnanie z hľadísk:

1. Metodológia návrhu
2. Stavovosť
3. Bezpečnosť

REST vs. SOAP - Metodológia návrhu

- REST
  - Identifikovať **zdroje**
  - Definovať “pekné” **URL**
  - Zabezpečiť **prelinkovanie zdrojov**
  - Implementovať a nasadiť na **Web server**
- SOAP
  - Identifikovať **operácie**
  - Definovať dátový **model**
  - Zvoliť **transportný protokol**
  - Implementovať a nasadiť na **Web service container**

REST vs. SOAP - Stavovosť

- REST
  - explicitné stavové prechody
  - Server je bezstavový
  - Klienti udržujú stav korektne využitím **liniek**
- Techniky zavedenia session:
  - Cookies (HTTP Headers)
  - URL Re-writing
  - Hidden Form Fields

- SOAP
  - implicitné stavové prechody
  - Stav naprieč správami
  - Klienti udržujú stav určovaný **stavovým strojom** služby
- Techniky zavedenia session:
  - Session Headers – neštandardný spôsob

REST vs. SOAP – Security

- REST
  - Security je o HTTPS
  - Proven track record
    - SSL1.0 od 1994
  - HTTP Basic Authentication
    - RFC 2617, 1999
    - RFC 1945, 1996
  - Secure – point to point
    - Autentifikácia
    - Integrita
    - Encryption
- SOAP
  - WS-Security extension (2004)
  - XML Encryption (2002)
  - XML Signature (2001)
  - Implementácie sa práve vynárajú
    - Problémy interoperability
    - Performance?
  - Secure – end-to-end
    - Self-protecting SOAP messages

REST vs. SOAP

If I were an **enterprise architect today**, and I were genuinely concerned about **development costs, agility, and extensibility**, I'd be looking to solve everything I possibly could with **dynamic languages and REST**, and **specifically the HTTP variety of REST**.

I'd **avoid ESB and the typical enterprise middleware frameworks** unless I had a problem that really required them. I'd also try to totally **avoid SOAP and WS-***.

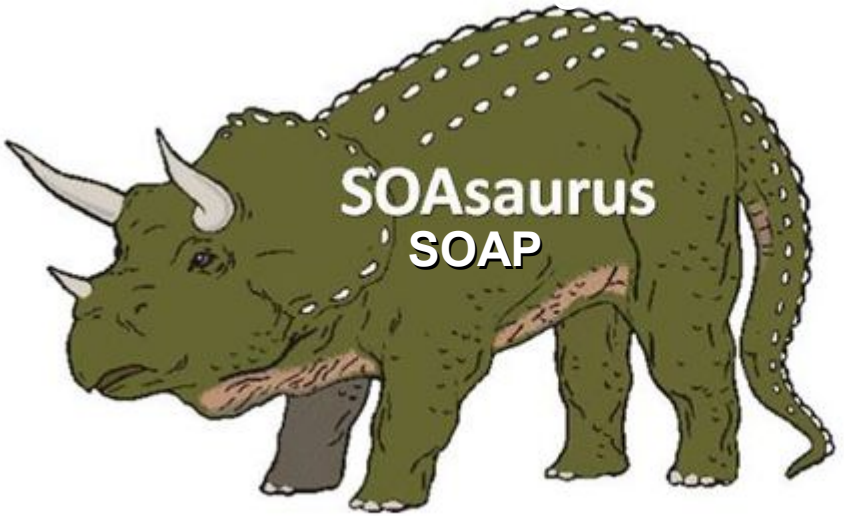
Steve Vinoski

IONA Technologies, Vice President Platform Technologies,
Chief Architect, ASP Technical Director, and IONA Fellow,
W3C Web Services Architecture Working Group.

http://steve.vinoski.net/blog/2007/10/04/the-esb-question/
http://www.almaden.ibm.com/institute/bio/biovinoski.html

Evolúcia?

REST



CENTAUR

Peter Rybár
peter.rybar@centaur.sk