UNIVERSITAS SCIENTIARUM SZEGEDIENSIS
UNIVERSITY OF SZEGED
Department of Software Engineering

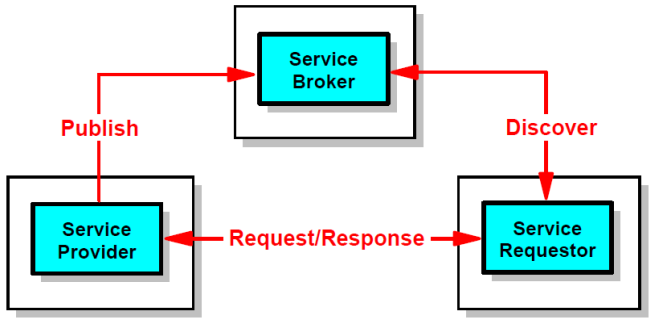
4. Web Szolgáltatások

Dr. Bilicki Vilmos
Szegedi Tudományegyetem
Informatikai Tanszékcsoport
Szoftverfejlesztés Tanszék

A mai előadás tartalma

- Bevezető
- REST
- JSON
- YAML
- Web Szolgáltatás szabványok
- SOAP
- WSDL
- JAX-RPC
- JEE – WS
- UDDI
- WS profilok
  - WS-Security
  - WS-Interoperability
- Web Szolgáltatás architektúrák

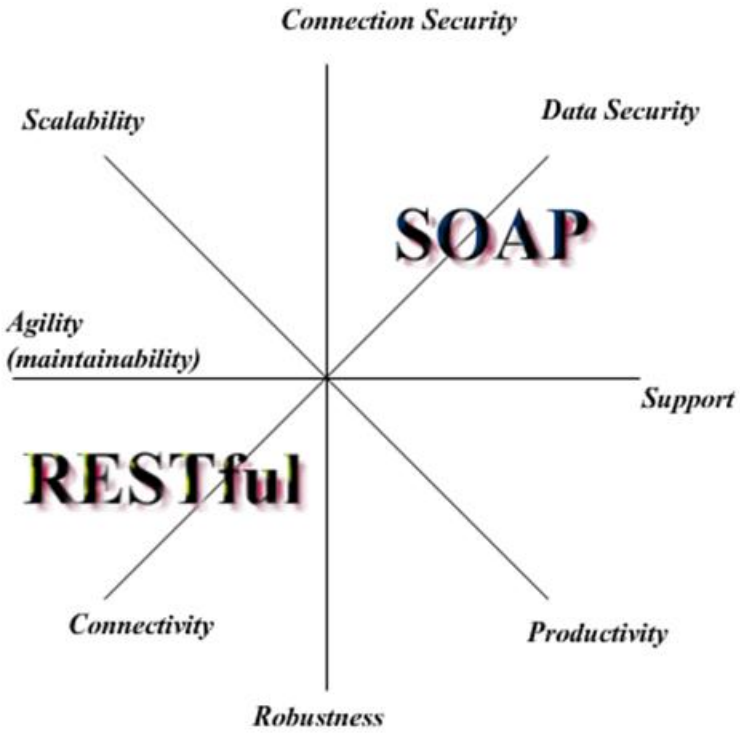
Bevezető



- Trendek
  - Integráció
  - Üzleti folyamatok teljes automatizálása (EDI)
- Szolgáltatás Orientált Architektúra
  - Szolgáltatás gyártó
  - Szolgáltatás közvetítő
  - Szolgáltatás fogyasztó
- Jellemzői
  - A kliens nem a szerverhez, hanem a szolgáltatáshoz kötődik
  - Az új és a régi komponensek blokkokba vannak csomagolva ezek web szolgáltatáson csatlakoznak
  - A komplex alkalmazásokon belül az üzleti logika el van különítve
  - Szolgáltatásokat futásidőben lehet cserélgetni
  - A csatolások konfigurációs fájlokban vannak definiálva

Megközelítésmódok

- Web Szolgáltatás
- REST
- RMI, ….



HTTP

- HyperText Transfer Protocol
- Web kliensek és szerverek közötti kommunikáció
- A leggyakrabban használt protokoll
- Utasításai: GET, PUT, POST, DELETE
- Lekérdezés paraméter kerterendszer

HTTP működése

- A kliens egy kérést küld
  - Metódus
  - URL
  - Fejléc
  - (esetenként) paraméterek
  - (esetenként) tartalom
- A szerver választ küld
  - Tartalom
  - Státusz
  - Fejlécek

Mit jelent a státusz

HTTP válaszok

- 50x: szerver oldali hiba
- 40x: kliens oldali hiba
- 30x: átirányítás
- 20x: rendben

Fejlécek, paraméterek

- Fejlécek
  - Metainformációk a kérésről
  - Pl.: adott formátumban kérjük a képet
- Paraméterek
  - Szűri vagy leírja az erőforrást
  - Megcímzi az erőforrást

Kérés/Válasz fejlécek

- Kérés(kliens)
  - Accept: Kérem az adott típusú választ Itt van az amit elfogadok.
    Accept: text/html,application/xhtml+xml,application/xml
- Válasz(szerver)
  - Content-Type: Ezt küldöm
    Content-Type: text/html; charset=UTF-8

Paraméterek

- Az URL része
- Minden a kérdőjel után
- http://www.example.com/search_people?this=that&foo=bar

Példa kérés

- Chrome böngésző kérése a Google-hoz
  - Method: GET
  - URL: http://www.google.com
  - Headers:
    - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
    - Accept-Language: en-US,en;q=0.8
    - Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
    - Connection: keep-alive
    - User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_3) AppleWebKit/535.19 (KHTML, like Gecko) Chrome/18.0.1025.168 Safari/535.19
    - Accept-Encoding: gzip,deflate,sdch
    - Cookie: NID=59=EudJ2a15ql8832PCysQA0qchtuvGWMoA7rkp79VpIYAQ8-j42lO17LFudCYNMXm9l6SHcu3YgrGRCdrRCyM468xPZaOek4Pi-AXQ8eARqU1SGYx6y7_9LW-c3HHb-vs2; PREF=ID=994f8de0e8b39a5b:U=237805f1f710dc73:FF=0:TM=1336752507:LM=1336752509:S=W0Hha7x4czdXp51U
    - Host: www.google.com

Példa válasz

- Google válasza
  - Headers:
    - Content-Length: 24716
    - Content-Encoding: gzip
    - Set-Cookie: NID=59=F48kbwfwOi-qCHJyrnMSUlDBVxK-ZVKZpq5B5jttt_25IRN4lS-0rQcVttq-dnOIlQzafw1i4HPQAO0RpZ7NuC0WCKWta7SYoekx0--YGf2zIFZ9VXIKS-_UEaOH9iBe; expires=Sat, 10-Nov-2012 21:26:46 GMT; path=/; domain=.google.com; HttpOnly
    - Expires: -1
    - Server: gws
    - X-XSS-Protection: 1; mode=block
    - Cache-Control: private, max-age=0
    - X-Frame-Options: SAMEORIGIN
    - Content-Type: text/html; charset=UTF-8
    - Date: Fri, 11 May 2012 21:26:46 GMT
  - Content: A bunch of HTML
  - Status: 200

Bináris tartalmak kezelése

- Metadata és bináris együtt
  - Multipart

```
POST /avatars HTTP/1.1
Host: localhost:3000
Authentication: Bearer some-token
Content-Type: multipart/form-data; boundary=MultipartBoundry
Accept-Encoding: gzip, deflate

--MultipartBoundry
Content-Disposition: form-data; name="image";
filename="12348024_1150631324960893_344096225642532672_n.jpg"
Content-Type: image/jpeg

rawimagecontentwhichlooksfunnyandgoesonforever.d.sf.d.f.sd.fsdkfjksl
hfdkshfkjsdfdkfh
--MultipartBoundry
Content-Disposition: form-data; name="myJsonData"
Content-Type: application/json

{"category":123,"location":123,-50}
--MultipartBoundry--
```

  - Csak bináris

```
POST /avatars HTTP/1.1
Host: localhost:3000
Authentication: Bearer some-token
Content-Type: image/jpeg
Content-Length: 284

raw image content
```

REST alapok

- Az erőforrások URI-val vannak azonosítva
- Az erőforrásokat a reprezentációjuk segítségével manipuláljuk
- Az üzenetek állapotmenetsek és önleíróak
- Egy erőforrásnak több reprezentációja is lehet
- Az alkalmazás állapota az erőforrások manipulációjval változik

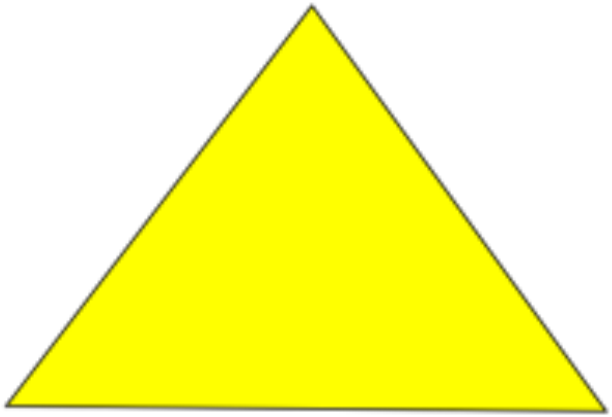
Erőforrások

- URI-val vannak azonosítva
  - Sohasem érhetőek el közvetlenül
  - A REST az erőforrás reprezentációkkal érintkezik
- Az erőforrás minden lehet
  - Ha valamit nem tudunk megnevezni akkor azzal foglalkozni sem tudunk
  - Erőforrás lehet pl.: a web dokumentum
  - A dokumentumok általában struktúrált információk

Állapot

- Az átviendő információ része
  - A szerver leállások nem okoznak problémát
  - Válthatunk szerverek között
  - A kliens tárolhatja a reprezentációt
- Az állapot átvitellel a rendszer skálázható lesz
  - Nem állapot függő
  - A meg van osztva a kliens és a szever között
- Laza csatolás (nem csak API vs dokumentum)
  - A kliens és a szerver elkülönül

REST háromszög



Nouns
(Unconstrained)
eg http://wikipedia.org/

Verbs
(Constrained)
eg GET

Content Types
(Constrained)
eg HTML

Főnevek

- Az erőforrások nevei
  - A legtöbb esetben URI-k
  - Az URI tervezés a REST rendszerek fontos aspektusa
- Az igék és a főnevek szeparációja:
  - az alkalmazások olyan erőforrásokkal is dolgozhatnak amiket nem ismernek
  - Új műveletek bevezeteése nem rombolj le a webet
  - Új tartalom típusok bevezetése nem rombolja le a webet

Igék

- Az erőforráson értelmezett műveletek
- A REST alap ötlete néhny unverzális ige használata
  - ezek minden főnévhez alklamazhatőak
- A legtöbb esteben a HTTP alap metódusok elegendőek
  - GET: Erőforrás lekérése
  - PUT: Erőforrás feltöltése
  - POST: Új erőforrás hozzáadása
  - DELETE: Erőforrás törlése

  CRUD: Create, Read, Update, Delete

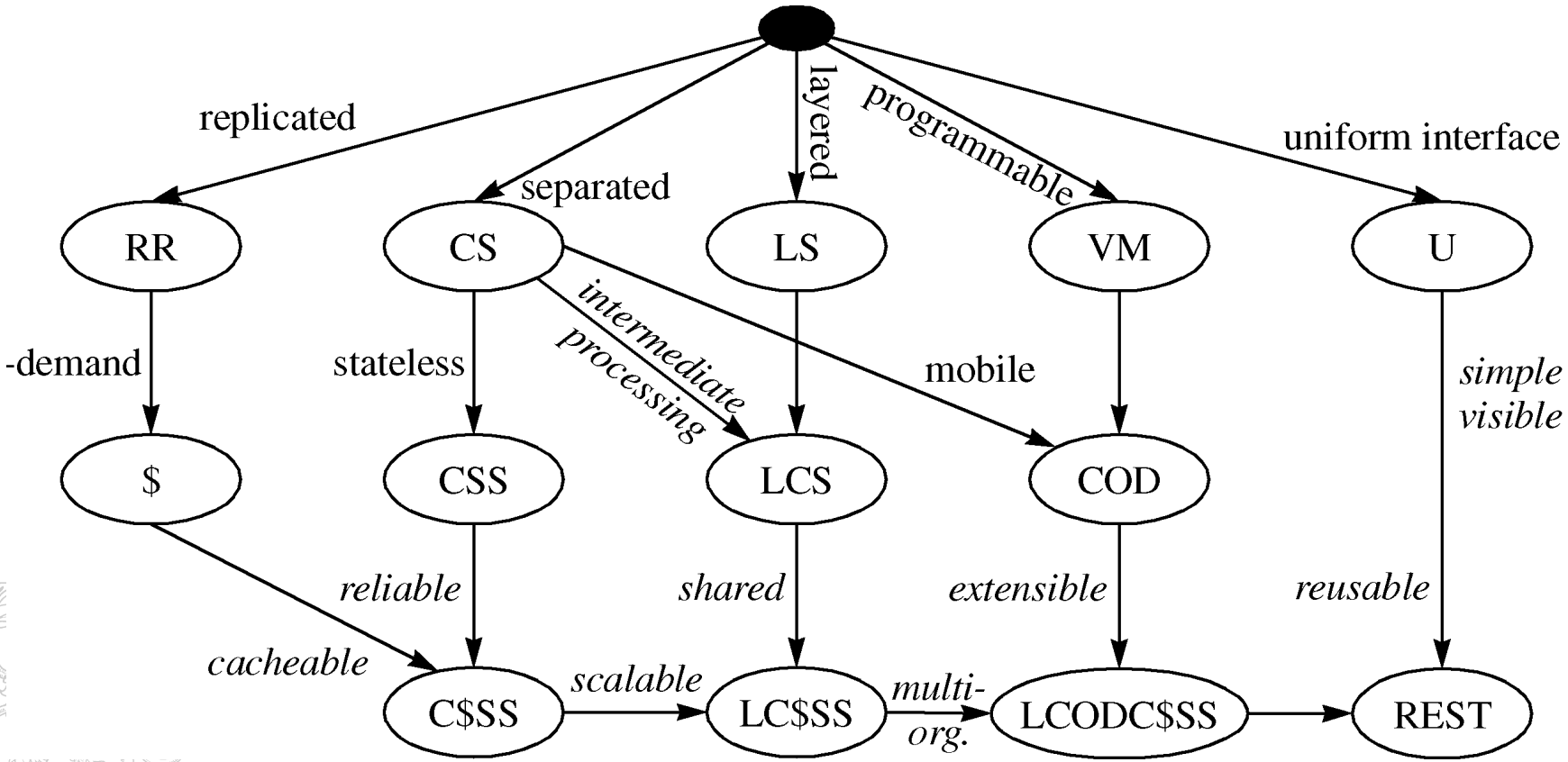
Tartalom típusok

- A reprezentációnak géppel is értelmezhetőnek kell lennie
- Az erőfforrás egy absztrakicója
  - több reprezentáció is lehet
  - a kliens választhat
- Tartalomtípusok hozzáadása, elvétele nem változtat a rendszer architektúrán
  - Különböző klienseknek különböző tartalomtípus
  - Content Negotiation lehetővé teszi ezt

REST

- Representative State Transfer – Reprezentatív Állapot Átvitel (HTTP Object Model)
- Célok:
  - Anarchikus skálálzhatóság
  - Biztonság
  - Hipermédia alapú
  - Független telepítés
- Egy szoftver architektúra stílus
  - Késleltetés minimalizálása
  - Függetlenség és skálázhatóság maximalizálása
- Szereplők:
  - Szolgáltató szerverek
  - Átjárók
  - Proxik
  - Kliensek
- Erőforrásokat visznek át (annak a reprezentációját), a felhasználó tényked
ése volt a tervezés fókuszában
- Egy kliens vagy alkalmazás álapotok közötti átmenetben van vagy pihen (rest)
- RESTful

Alapelvek


replicated
separated
layered
programmable
uniform interface
RR
CS
LS
VM
U
-demand
stateless
intermediate
processing
mobile
simple
visible
$
CSS
LCS
COD
reliable
shared
extensible
reusable
cacheable
scalable
multi-
org.
C$SS
LC$SS
LCODC$SS
REST

REST interfész célok / Adat elemek

- Elemek:
  - Adat
  - Konnektor
  - Feldolgozó
- Interfész célok:
  - Erőforrás azonosítás
  - Erőforrás manipulálás
  - Önleíró üzenetek
  - Hipermédia mint reprezentáció
- Adat elemek

Table I. REST Data Elements

Data Element	Modern Web Examples
resource	the intended conceptual target of a hypertext reference
resource identifier	URL, URN
representation	HTML document, JPEG image
representation metadata	media type, last-modified time
resource metadata	source link, alternates, vary
control data	if-modified-since, cache-control

Munkamegosztás stílusok/ Erőforrások

- Munkamegosztás stílus
  - Szerver oldal (Csak az eredmény)
  - Kliens oldal + algoritmus (Mobil objektum)
  - Kliens oldal (Nyers adat)
- Erőforrás azonosítás:
  - Egy R függvény mely az időben változó módon rendeli az adott erőforrásokat az adott azonosítókhoz
- Erőforrás reprezentáció:
  - Adat + Metaadat
  - Vezérlő adat

Konnektorok\Komponensek

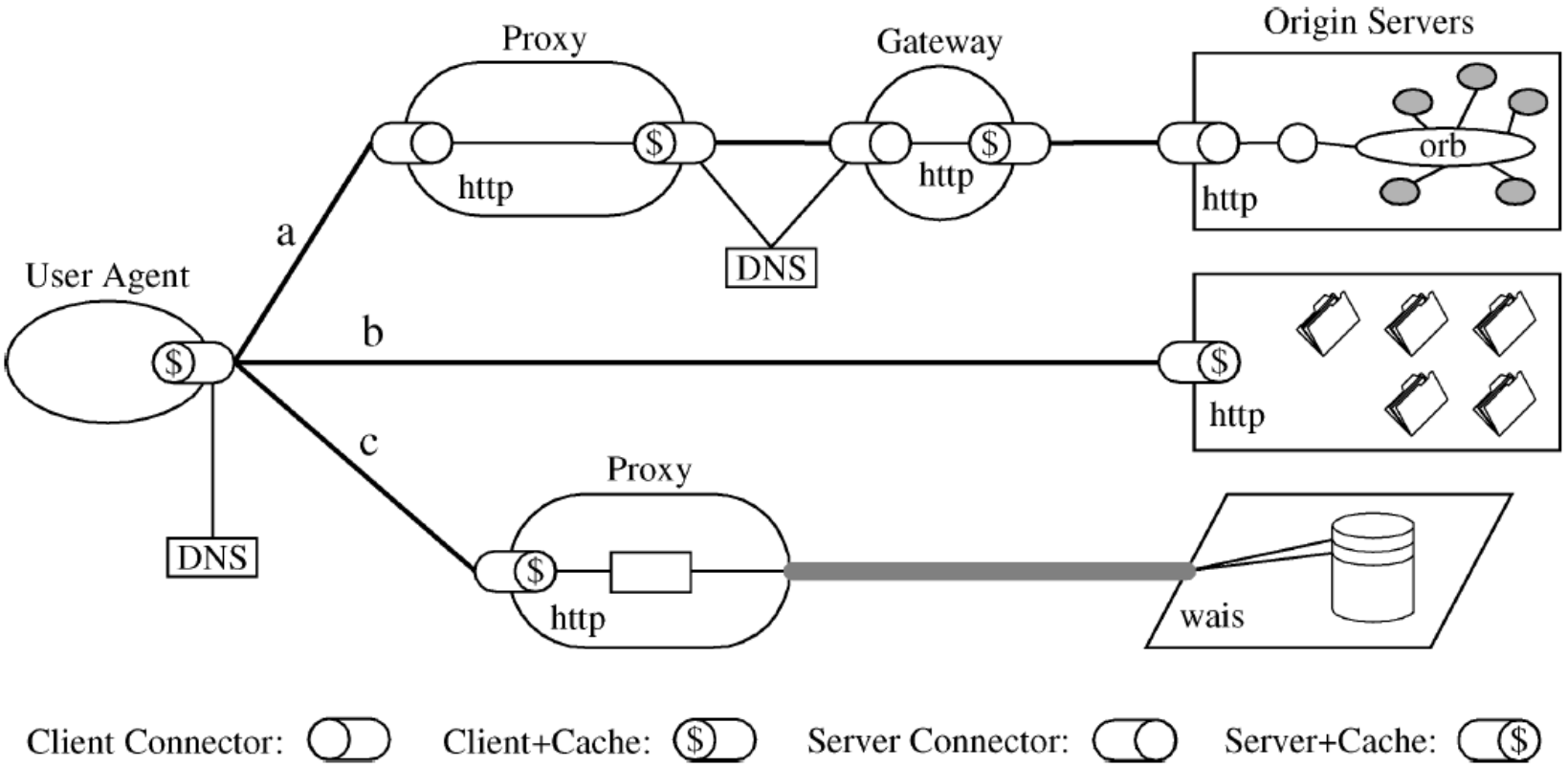
Table II. REST Connector Types

Connector	Modern Web Examples
client	libwww, libwww-perl
server	libwww, Apache API, NSAPI
cache	browser cache, Akamai cache network
resolver	bind (DNS lookup library)
tunnel	SOCKS, SSL after HTTP CONNECT

Table III. REST Component Types

Component	Modern Web Examples
origin server	Apache httpd, Microsoft IIS
gateway	Squid, CGI, Reverse Proxy
proxy	CERN Proxy, Netscape Proxy, Gauntlet
user agent	Netscape Navigator, Lynx, MOMspider

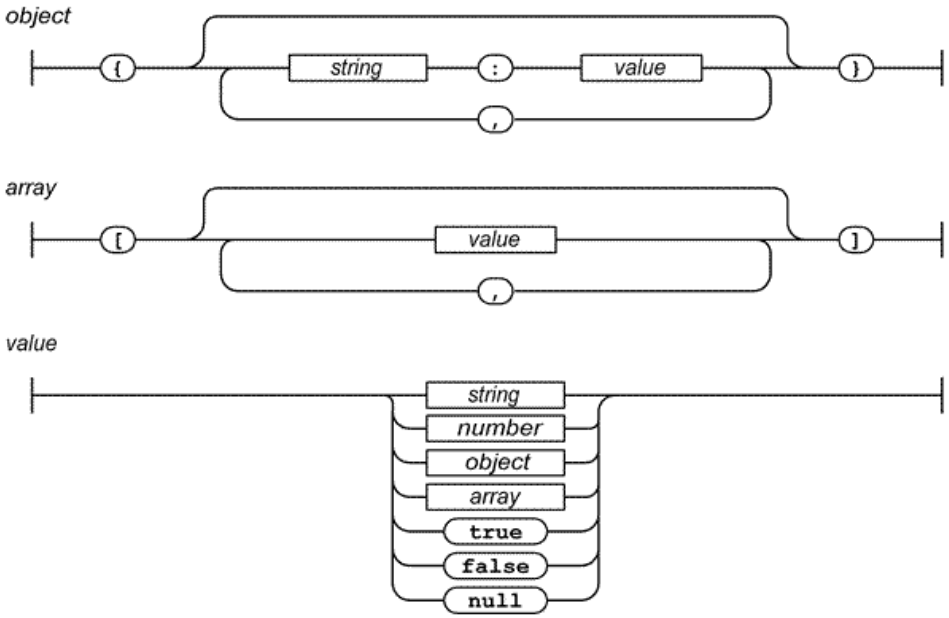
REST architektúra



JSON

- Egyszerű adatcsere formátum, RFC 4627
- Emberi szem számára

```
[
    {
      "precision": "zip",
      "Latitude":  37.7668,
      "Longitude": -122.3959,
      "Address":   "",
      "City":      "SAN FRANCISCO",
      "State":     "CA",
      "Zip":       "94107",
      "Country":   "US"
    },
    {
      "precision": "zip",
      "Latitude":  37.371991,
      "Longitude": -122.026020,
      "Address":   "",
      "City":      "SUNNYVALE",
      "State":     "CA",
      "Zip":       "94085",
      "Country":   "US"
    }
]
```



Example 2.27. Invoice

```
--- !<tag:clarkevans.com,2002:invoice>
invoice: 34843
date   : 2001-01-23
bill-to: &id001
    given  : Chris
    family : Dumars
    address:
        lines: |
            458 Walkman Dr.
            Suite #292
        city    : Royal Oak
        state   : MI
        postal  : 48046
ship-to: *id001
product:
    - sku         : BL394D
      quantity    : 4
      description : Basketball
      price       : 450.00
    - sku         : BL4438H
      quantity    : 1
      description : Super Hoop
      price       : 2392.00
tax  : 251.42
total: 4443.52
comments:
    Late afternoon is best.
    Backup contact is Nancy
    Billsmer @ 338-4338.
```

Example 2.28. Log File

```
---
Time: 2001-11-23 15:01:42 -5
User: ed
Warning:
  This is an error message
  for the log file
---
Time: 2001-11-23 15:02:31 -5
User: ed
Warning:
  A slightly different error
  message.
---
Date: 2001-11-23 15:03:17 -5
User: ed
Fatal:
  Unknown variable "bar"
Stack:
  - file: TopClass.py
    line: 23
    code: |
      x = MoreObject("345\n")
  - file: MoreClass.py
    line: 58
    code: |-
      foo = bar
```

REST HTTP alapon

- Az erőforrásokat URL-lel azonosítja
- Create, Read, Update, Delete
  - POST, GET, PUT, DELETE
- Hibak kódok
  - HTTP státusz kódok
- Kérés paraméterek
  - Kérés paraméterek
- Válasz és konfiguráció
  - Fejlécek

Példa REST kérés

- Blog Info from Tumblr
- GET (olvas)
  http://api.tumblr.com/v2/blog/synedra.tumbler.com/info
- api_key kulcs
- http://api.tumblr.com/v2/blog/synedra.tumblr.com/info?api_key=my_api_key

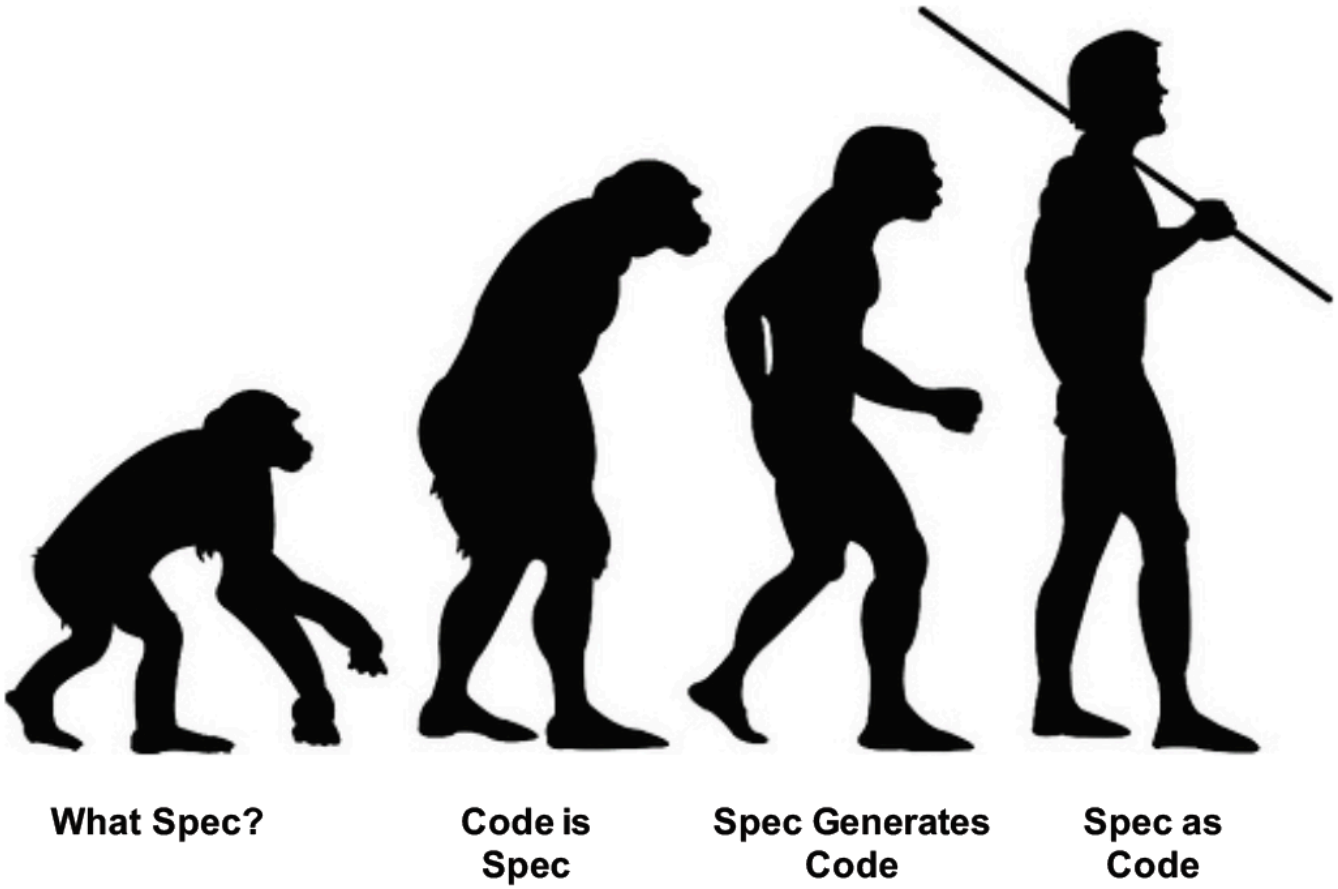
REST válasz

Status: 200

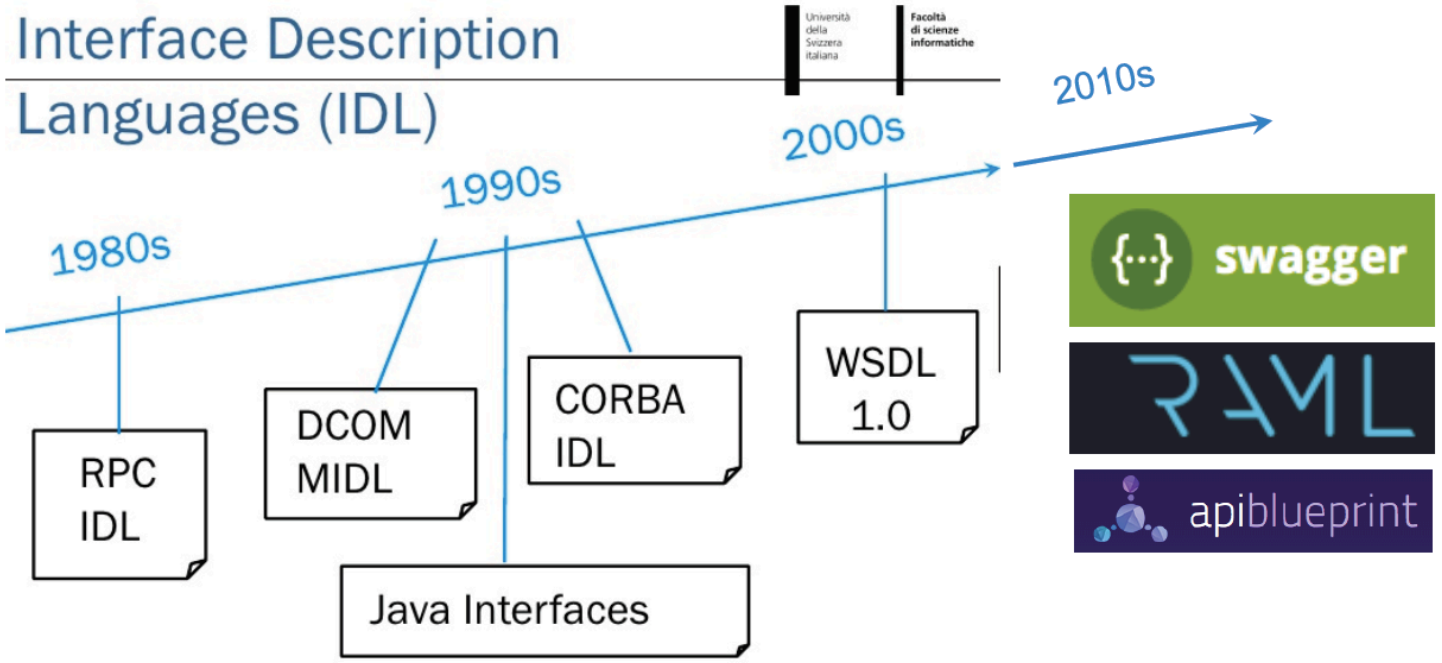
Content:

{"meta":
{"status":200, "msg":"OK" },
"response":{
"blog":{"title":"Untitled","posts":0,
"name":"synedra",
"url":"http:\/\/synedra.tumblr.com\/",
"updated":0,
"description":"","ask":false,"likes":0}}}

Interfész specifikáció

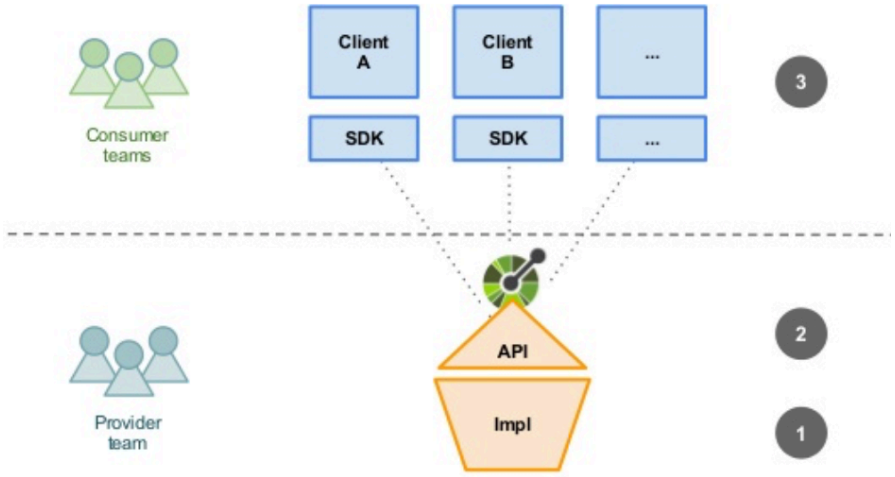


Interfész specifikáció



Használati esetek

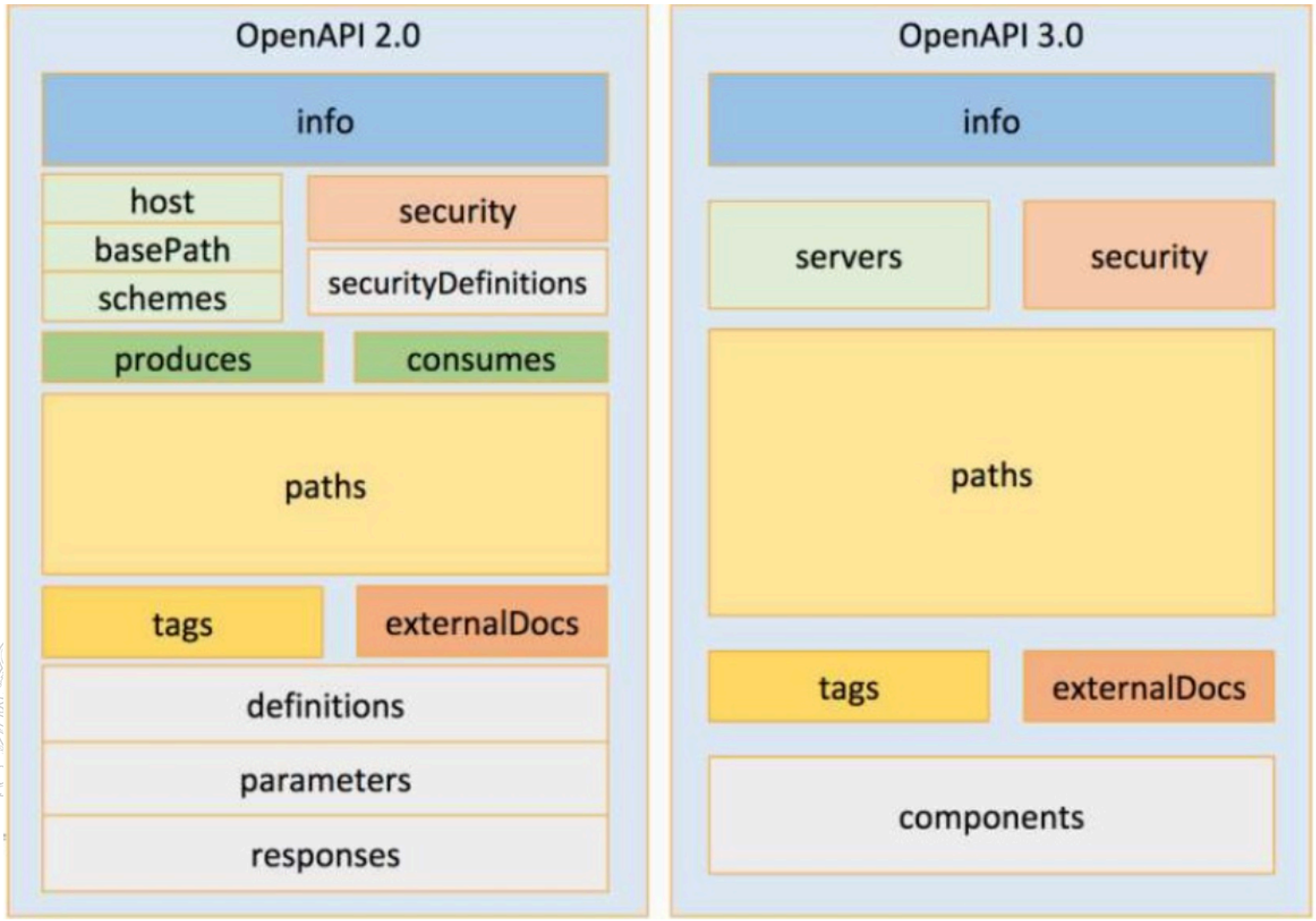
- Régi API
- Interfész először
- Szolgáltatás orientál



OpenAPI

- Szerkesztő
- API Felderítő
- Validáló
- Nyílt forrású generátorok
  - proxik (front-end)
  - vázak (back-end)

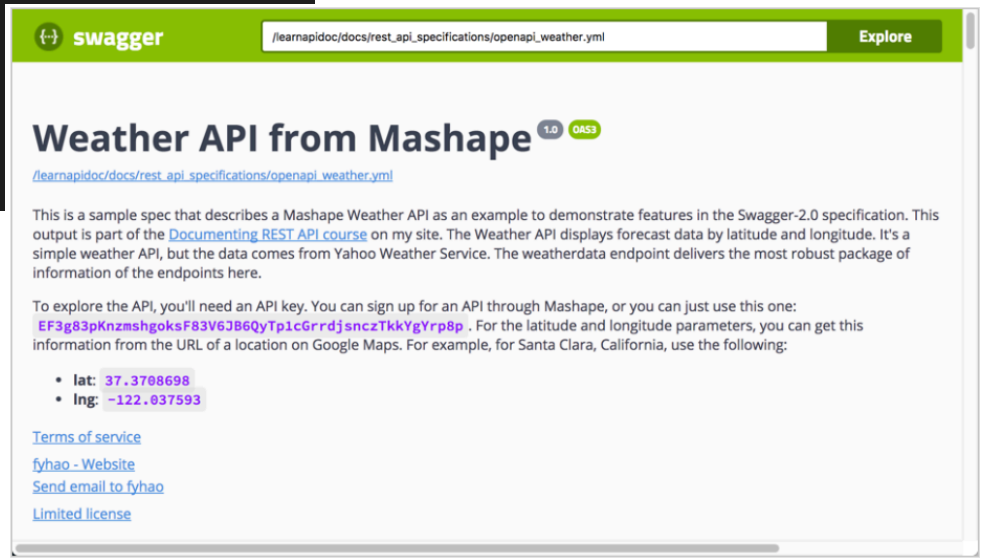
OpenAPI


OpenAPI 2.0
info
host
basePath
schemes
security
securityDefinitions
produces
consumes
paths
tags
externalDocs
definitions
parameters
responses
OpenAPI 3.0
info
servers
security
paths
tags
externalDocs
components

Info

- Alapvető információkat tartalmaz az API-ról
  - description: CommonMark Markdown

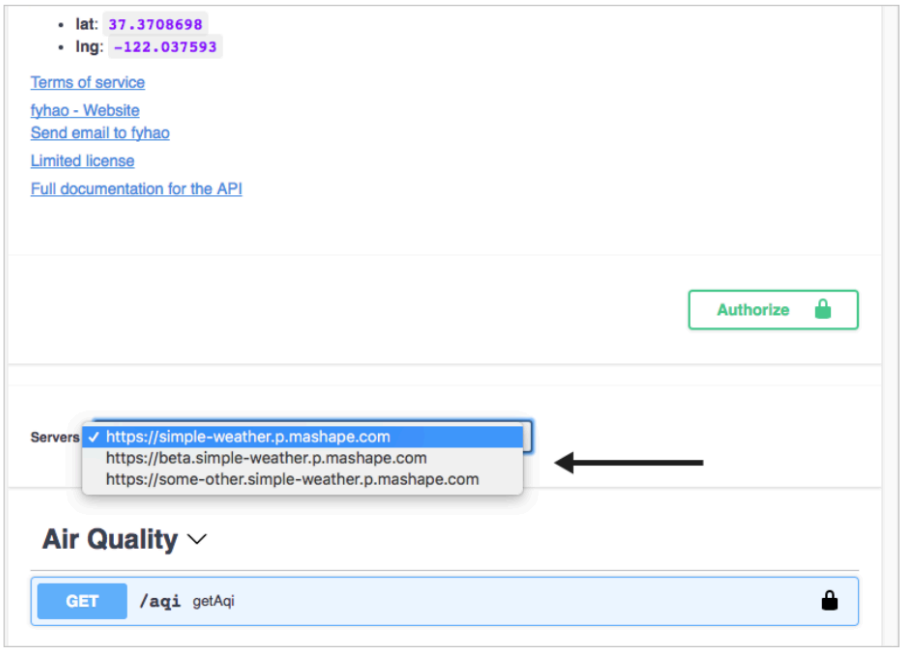
```yaml
info:
  title: Weather API from Mashape
  description: "This is a sample spec that describes a Mashape Weather API as an example to
  version: "1.0"
  termsOfService: https://konghq.com/terms/
  contact:
    name: fyhao
    url: https://market.mashape.com/fyhao
    email: some_email@gmail.com
  license:
    name: Limited license
    url: https://konghq.com/terms/
```



Servers

- A szerver URL-eket adjuk meg

```yaml
servers:
  - url: https://simple-weather.p.mashape.com
    description: Production server
  - url: https://beta.simple-weather.p.mashape.com
    description: Beta server
  - url: https://some-other.simple-weather.p.mashape.com
    description: Some other server
```



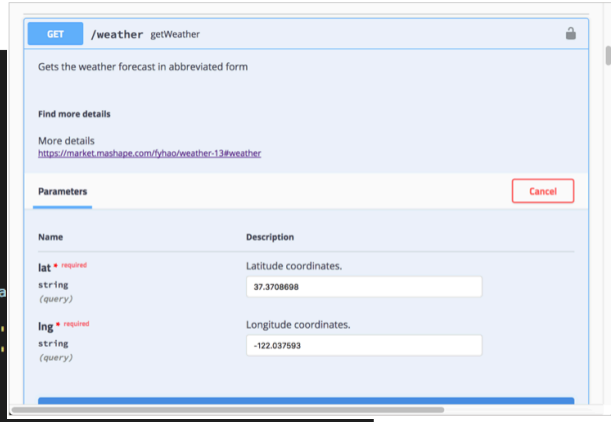
Paths

```
paths:
  /aqi:
    get:

  /weather:
    get:

  /weatherdata:
    get:
```

- A kezelt objektumok és azok hozzáférése
  - Művelet objektum: GET, POST, ...
  - tags: a swagger UI e szerint csoportosítja
  - summary: összefoglaló 5-10 szóban (CommonMark Markdown)
  - description: teljes leírás (CommonMark Markdown)
  - externalDocs: az útvonalra vonatkozó további dokumentumok
  - operationID: egyedi azonosító az útvonalhoz
- paremeters: nem tartalmazza a body paramétereket
- requestBody: hivatkozhat referenceObject-re amely hivatkozást tartalmazhat component-re
- responses: hivatkozhat referenceObject-re amely hivatkozást tartalmazhat component-re
- callbacks: hivatkozhat referenceObject-re amely hivatkozást tartalmazhat component-re
- security: azonosítási módszer, component objektum secruitySchemes adata
- server: más lehet mint a globális szerver objetum

```yaml
paths:
  /aqi:
    get:
      tags:
      - Air Quality
      summary: getAqi
      description: Gets the air quality index
      operationId: GetAqi
      externalDocs:
        description: More details
        url: "https://market.mashape.com/fyhao/weather-13#aqi"
      parameters:

      - name: lat
        in: query
        description: "Latitude coordinates."
        required: true
        style: form
        explode: false
        schema:
          type: string
        example: "37.3708698"

      - name: lng
        in: query
        description: "Longitude coordinates."
        required: true
        style: form
        explode: false
        schema:
          type: string
        example: "-122.037593"

      responses:
        200:
          description: AQI response
          content:
            text/plain:
              schema:
                type: string
                description: AQI response
                example: 52
  /weather:
    get:
      servers:
      - url: https://simple-weather.p.mashape.com
      tags:
      - Weather Forecast
      summary: getWeather
      description: Gets the weather forecast in abbreviated form
      operationId: GetWeather
      externalDocs:
        description: More details
        url: "https://market.mashape.com/fyhao/weather-13#weather"
```

```yaml
paths:
  /aqi:
    get:
      tags:
      - Air Quality
      summary: getAqi
      description: Gets the air quality index
      operationId: GetAqi
      externalDocs:
        description: More details
        url: "https://market.mashape.com/fyhao/wea
      parameters:
        - $ref: '#/components/parameters/latParam'
        - $ref: '#/components/parameters/lngParam'
      responses:
        200:
          description: AQI response
          content:
            text/plain:
              schema:
                type: string
                description: AQI response
                example: 52
  /weather:
    get:
      servers:
      - url: https://simple-weather.p.mashape.com
      tags:
      - Weather Forecast
      summary: getWeather
      description: Gets the weather forecast in abbreviated form
      operationId: GetWeather
      externalDocs:
        description: More details
        url: "https://market.mashape.com/fyhao/weather-13#weather"
      parameters:
        - $ref: '#/components/parameters/latParam'
        - $ref: '#/components/parameters/lngParam'
      responses:
        200:
          description:
          content:
            text/plain
              schema:
```




```yaml
components:
  parameters:
  - name: lat
    in: query
    description: "Latitude coordinates."
    required: true
    style: form
    explode: false
    schema:
      type: string
    example: "37.3708698"

  - name: lng
    in: query
    description: "Longitude coordinates."
    required: true
    style: form
    explode: false
    schema:
      type: string
    example: "-122.037593"
```

Components

- Adott elemeket szeretnénk újrahasznosítani
- &ref segítségével hivatkozhatunk rá
- megengedett elemek
  - schemas
  - responses
  - parameters
  - examples
  - requestBody
  - headers
  - securitySchemas
  - links
  - callbacks

```yaml
paths:
  /aqi:
    get:
      ...
      parameters:
            - $ref: '#/components/parameters/latParam'
            - $ref: '#/components/parameters/lngParam'
      ...
```

```yaml
components:
  parameters:
    latParam:
      name: lat
      in: query
      description: "Latitude coordinates."
      required: true
      style: form
      explode: false
      schema:
        type: string
      example: "37.3708698"
    lngParam:
      name: lng
      in: query
      description: "Longitude coordinates."
      required: true
      style: form
      explode: false
      schema:
        type: string
      example: "-122.037593"
```

Security

- Azonosítási protokoll sepcifikálása
- Swaager által támopgatott:
  - API key
  - HTTP
  - Oauth 2.0
  - OpenID Connect

API key alapú azonosítás

- HTTP mezőben átvitt érték
- Globális mező, de az átírható

(`X-Mashape-Key: 123456789`).

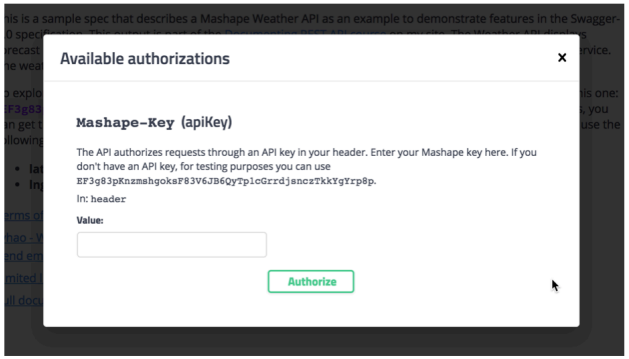
```
security:
- Mashape-Key: []
```

```
components:
  ...

  securitySchemes:
    Mashape-Key:
      type: apiKey
      description: "The API authorizes requests through an API key in your header. Enter your
      name: X-Mashape-Key
      in: header
```

```
/weatherdata:
  get:
    ...
    security:
    - Some-Other-Key: []
```

securitySchemas

- securitySchemasObject
  - type: apiKey, http, oauth2, openIdConnect
  - description
  - name: (csak apiKey-re)
  - in: query, header, cookie (csak apiKey-re)
  - scheme: http -hez
  - bearerFormat: http -hez
  - flows: oauth2-hez
  - openIdConnectionUrl: openIdConnect-hez



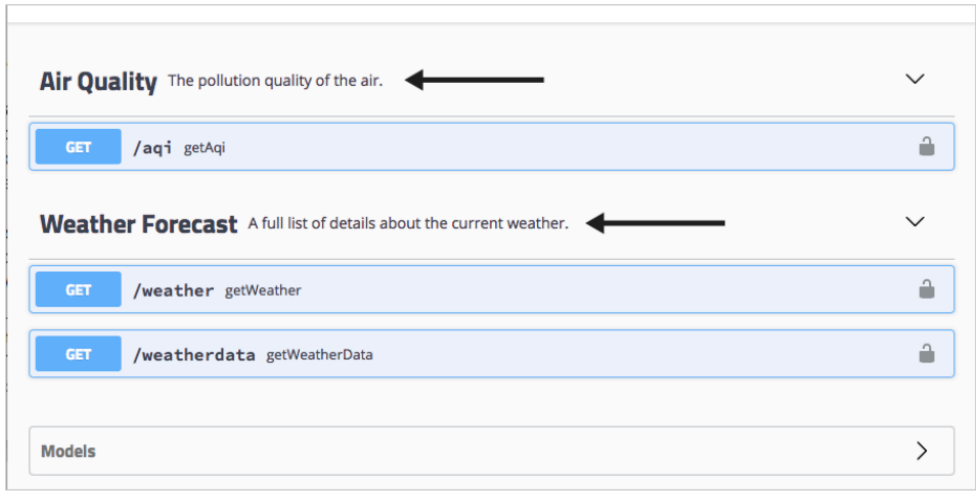

```
H "accept: text/plain" -H "X-Mashape-Key: EF3g83pKnzmshgoksF83V6JB6QyTp1cGrrdjsnczTkkYgYrp8p"
```

Tags

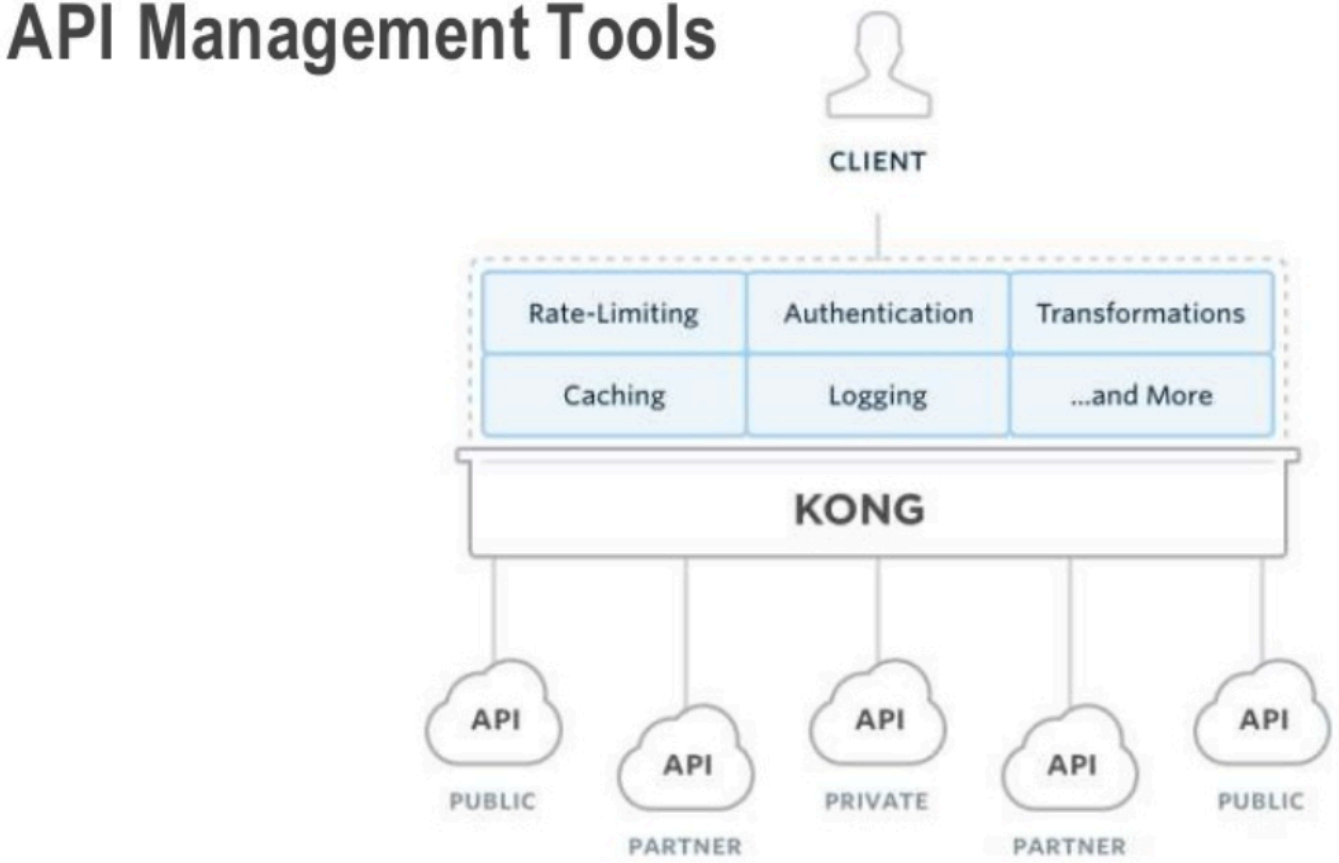
- segítségével csoprtosíthatjuk a végpontokat

```yaml
tags:
  - name: Air Quality
    description: The pollution quality of the air.
  - name: Weather Forecast
    description: A full list of details about the current weather.
```

```yaml
paths:
  ...
  /weather:
    get:
      servers:
      - url: https://simple-weather.p.mashape.com
      tags:
      - Weather Forecast
      ...
```



API Menedzsment



API verziózás

- URL

```
GET /v1/restaurants?location=SVQ
```

```
GET /v2/restaurants?location=SVQ&limit=30
```

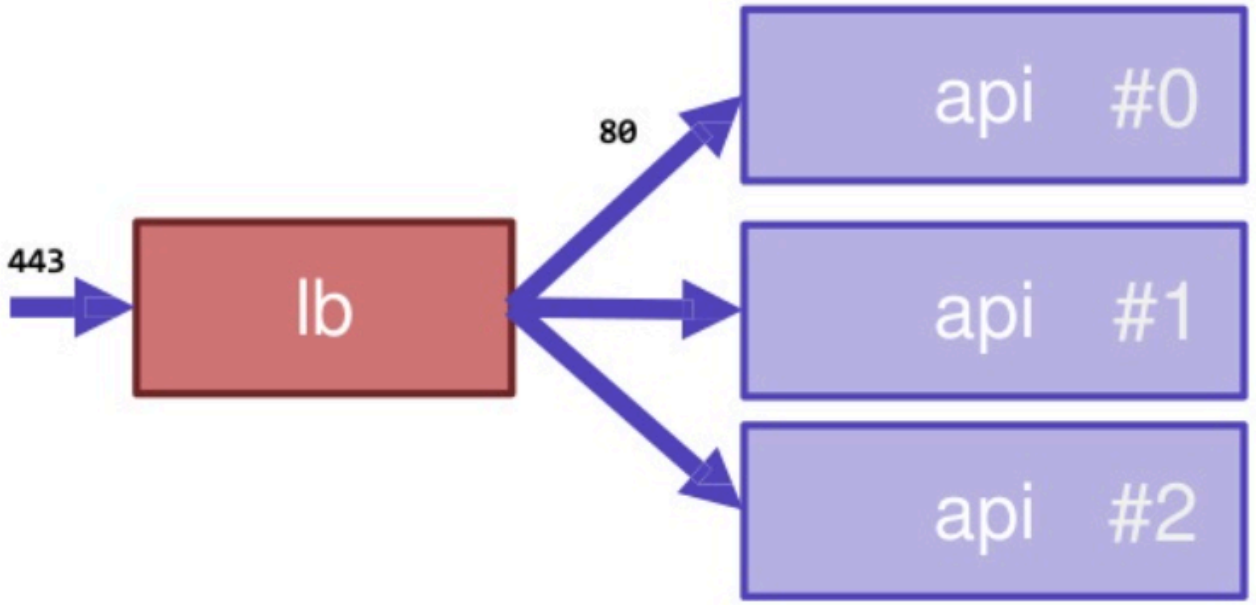
- Paraméter

```
GET /restaurants?location=SVQ&limit=30&v=2
```

- Fejléc

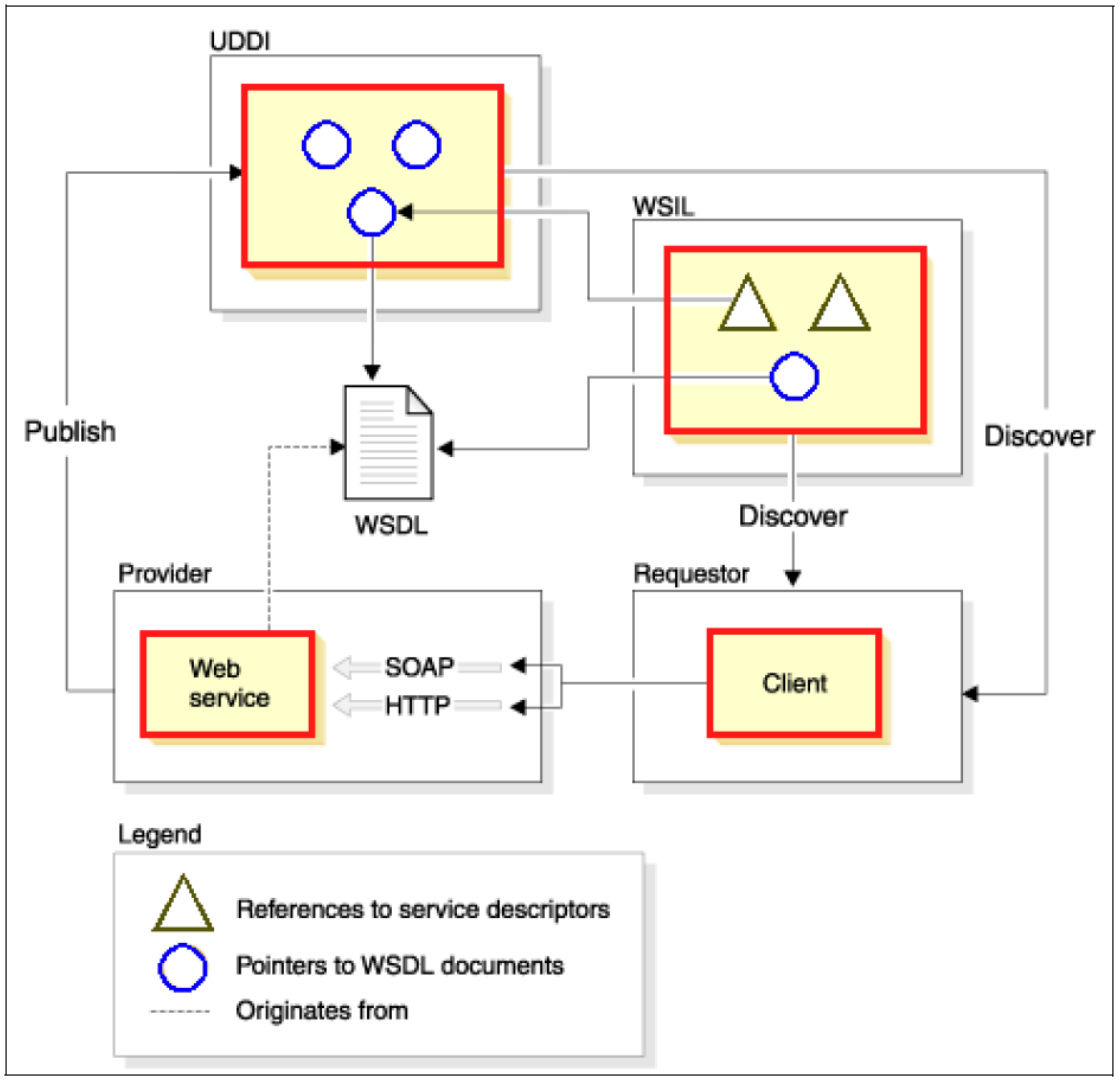
```
GET /restaurants?location=SVQ&limit=30
Version: 2
```

Skálázás



A SOA fő elemei

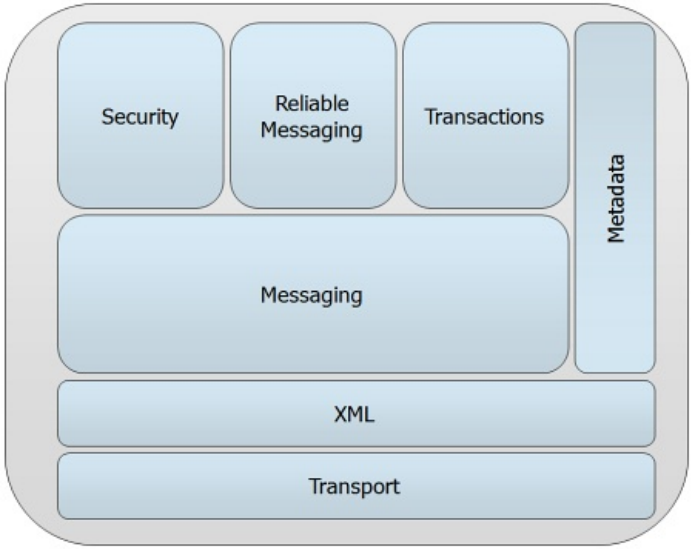
- XML
- SOAP
- WSDL
- WSIL
- UDDI



A web szolgáltatások jellemzői

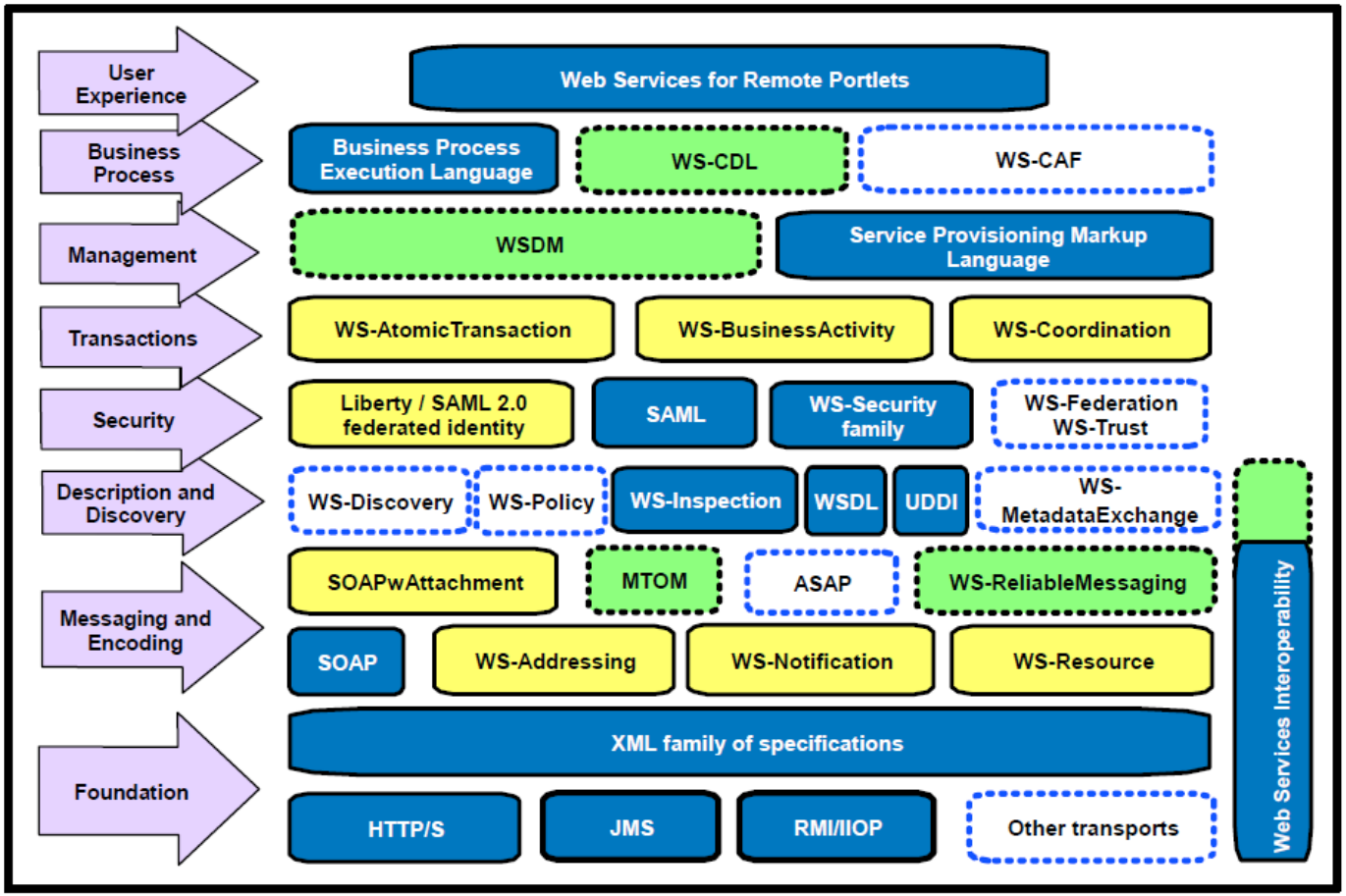
- Önhordó
- Önleíró
- A weben keresztül van publikálva, fellelve és használva
- Moduláris
- Nyelv független
- Nyílt szabvány
- Lazán csatoltak
- Dinamikusak
- Programozható hozzáférést biztosítanak

Története



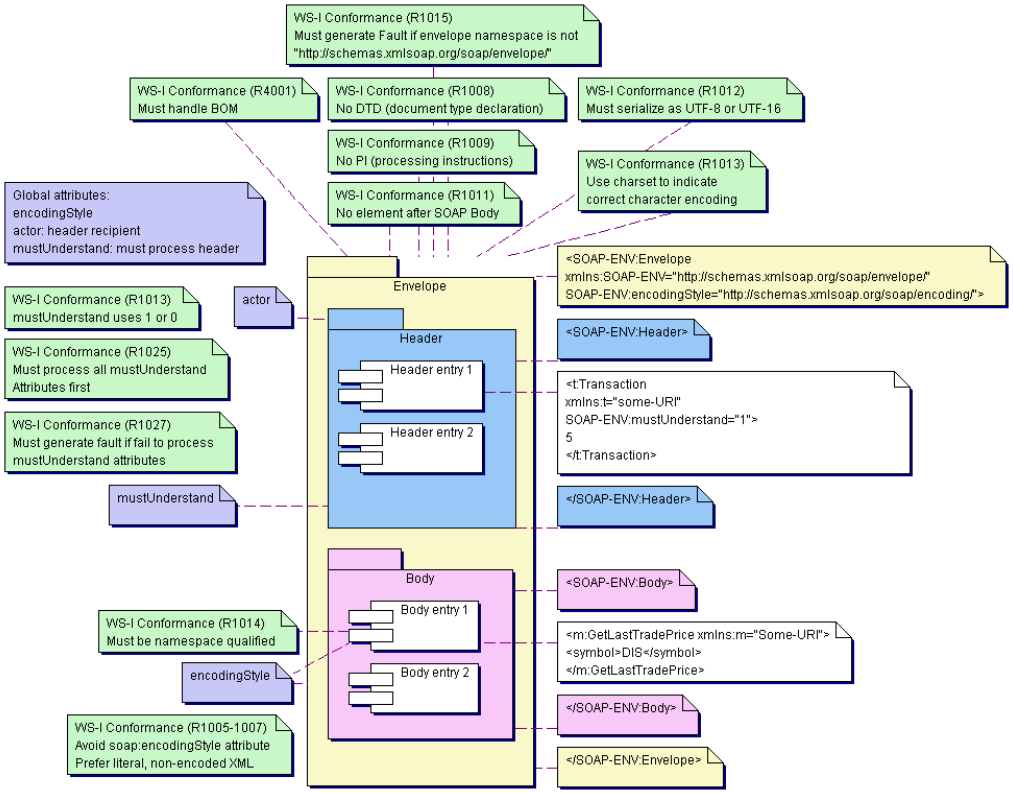
- Web sikersztori
- H2A működik
- A2A nem igazán
- 1999: Microsoft XML alapú protokol: SOAP
- IBM, Microsoft, Ariba: WSDL
- Ma több mint 40 ajánlás/specifikáció

Web szolgáltatás szabványok



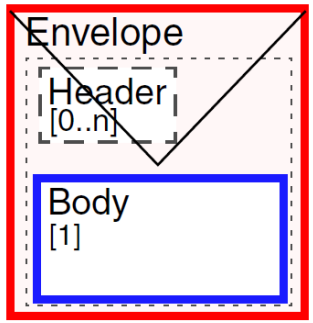
Alapvető szabványok

- SOAP: Simpe Object Access Protocol
  - Struktúrált és típusos XML dokumentumok cseréjét írja le elosztott környezetben
  - Önhordó, önleíró
  - Alapesetben állapotmentes, egyirányú kommunikáció
- WSDL: Web Service Description Language
  - A web szolgáltatást mind absztrakt végpontot definiálja
  - A műveletek és az üzenetek is megfelelő absztrakcióval vannak leírva
  - Az aktuális üzentekre építő protokoll pedig konkrét szolgáltatásokat specifikál
- UDDI: Universal Description, Discovery, and Integration
  - Web szolgáltatások felderítése és publikálása



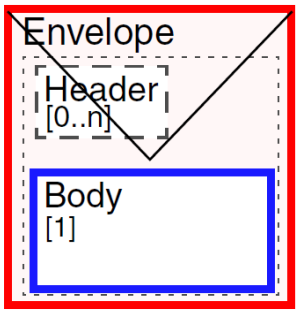
SOAP

- XML alapú protokol
  - Envelope
  - Header
  - Body
- Független az átviteli protokolltól (HTTP, JMS, FTP, …)
- Jelenleg HTTP (WS-I Basic Profile 1.0)
- Üzenetváltás minta (Message Exchange Pattern - MEP)
  - Egyirányú/Kétirányú




```
<Envelope>
  <Header>
    <actor>http://...org/soap/actor/next</actor>
    <qos mustUnderstand="1">log</qos>
  </Header>
  <Body>
    <getMessage ns1="urn:NextMessage" ...>
    <UserID type="string">JDoe</UserID>
    <Password type="string">OJDOEO</Password>
    </getMessage>
  </Body>
</Envelope>
```

SOAP elemei




```
<Envelope>
  <Header>
    <actor>http://...org/soap/actor/next</actor>
    <qos mustUnderstand="1">log</qos>
  </Header>
  <Body>
    <getMessage ns1="urn:NextMessage" ...>
    <UserID type="string">JDoe</UserID>
    <Password type="string">0JDOE0</Password>
    </getMessage>
  </Body>
</Envelope>
```

- Boríték (Envelope)
  - Ez tárolja a többit
  - Vezérlő információk
  - Cím, …
- Egy vagy több fejléc (Header)
  - Vezérlő információk (QoS)
  - **Ki és hogyan kezelje az üzenetet?**
- Egy törzs (Body)
  - Üzenet azonosítás
  - Paraméterek
  - **Mit csináljunk?**
- Kódolási szabályok
  - Megadja, hogy az adatot hogyan sorosítsuk
  - Programozási nyelv független adat séma (XSD)

Fejlécek

- Általános és flexibilis mechanizmus a SOAP üzenetek kiterjesztésére
- Nem szükséges a felek között előzetes egyeztetés
- Előre definiált fejléc attribútum:

```
<thens:qos xmlns:thens="someURI" SOAP-ENV:mustUnderstand="1">
   3
</thens:qos>
```

- SOAP köztes entitás
  - A fejlécek egy része ezekhez az entitásokhoz szól
    - SOAP-ENV:actor
- A hibák kezelése a MEP-től függ (mustUnderstand fault WS-I BP 1.0)
- A fejlécek viszik át a biztonság, tranzakció, titkosítás, .. infókat is
- Hordozhatnak kliens vagy projekt specifikus információkat is

WS-I konformancia fejléc

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" >
   <soap:Header>
      <!-- other headers -->
      <wsi:Claim conformsTo="http://ws-i.org/profiles/basic/1.0"
            xmlns:wsi="http://ws-i.org/schemas/conformanceClaim/" />
      <!-- other headers -->
   </soap:Header>
   <soap:Body>
      <!-- body content -->
   </soap:Body>
</soap:Envelope>
```

Törzs (Body)

- A végső címzettnek szóló információcserére szolgáll
- A Body elemen belül található XML elemek a test bejegyzések (body entries)
- A bejegyzések egymástól függetlenül vannak kódolva
- A legtöbb esetben a body tartalma:
  - Üzenet neve
  - Egy referencia a szolgáltatás példányra
  - Egy vagy több paraméter

Hibakezelés

- A SOAP definiál egy body elemet erre a célra
  - Fault element (nulla vagy egy lehet belőle)
    - faultcode
      - soapenv:Client
      - soapenv:Server
      - sopaenv:VersionMismatch
      - soapenv:MustUnderstand
    - faultstring
      - Ember által értelmezhető szöveges leírás
    - faultactor
      - Opcionális, a hiba forrását adja meg (URI)
      - A köztes elemeknek ezt kötelező kitöltenie
    - detail
      - Alkalmazás specifikus mező, opcionális

Adatmodell

- Nyelvfüggetlen absztrakció
- Egyszerű XSD típusok
- Összetett típusok
  - Struktúrák
  - Tömbök (benne lehet struktúra vagy tömb, …)
- A SOAP-ENC névtérben specifikálják az elemeket
- A SOAP csak azt mondja meg, hogy hogyan lehet az adattípusokat megadni, azt nem hogy ezek milyenek

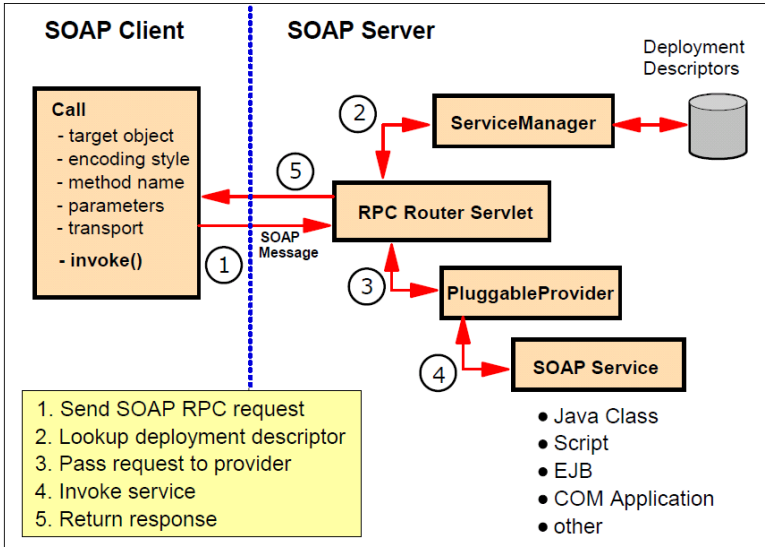
```
<ns1:getMessage xmlns:ns1="urn:NextMessage"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <UserID xsi:type="xsd:string">JDoe</UserID>
    <Password xsi:type="xsd:string">0JDOE0</Password>
</ns1:getMessage>
```

Tömbök

```
<ns1:getSubscribers xmlns:ns1="urn:SubscriberList"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENC:Array SOAP-ENC:arrayType="xxx:Subscribers[2]">
      <Subscribers>
         <UserID xsi:type="xsd:string">JDoe</UserID>
         <Password xsi:type="xsd:string">OJDOEO</Password>
      </Subscribers>
      <Subscribers>
         <UserID xsi:type="xsd:string">MDoe</UserID>
         <Password xsi:type="xsd:string">OJMDOEO</Password>
      </Subscribers>
   </SOAP-ENC:Array>
</ns1:getSubscribers>
```

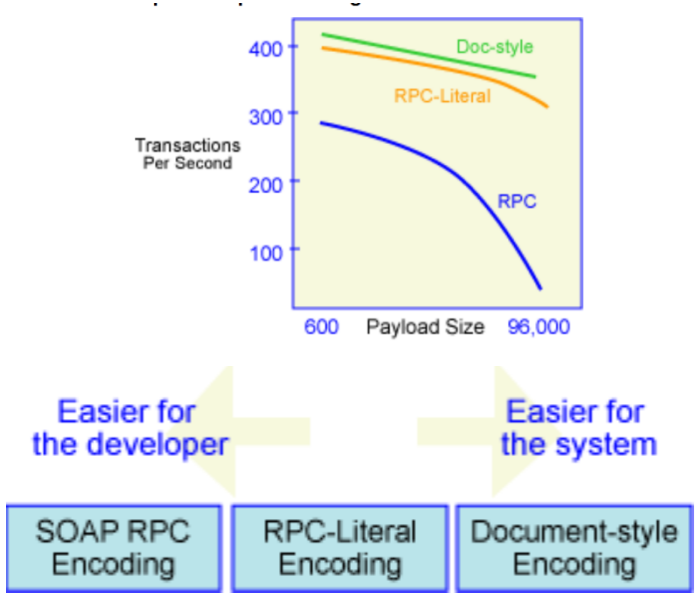
Kommunikációs stílusok

- Dokumentum
  - Üzenet orientált stílus
  - Alacsonyabb absztrakciós megoldás
  - Az in paraméter egy XML dokumentum
  - A válasz bármi (vagy semmi)
- RPC
  - Szinkron kommunikáció
  - Részei
    - A távoli objektum címe (URI)
    - A metódus neve
    - Paraméterek
    - Opcionális fejléc adatok

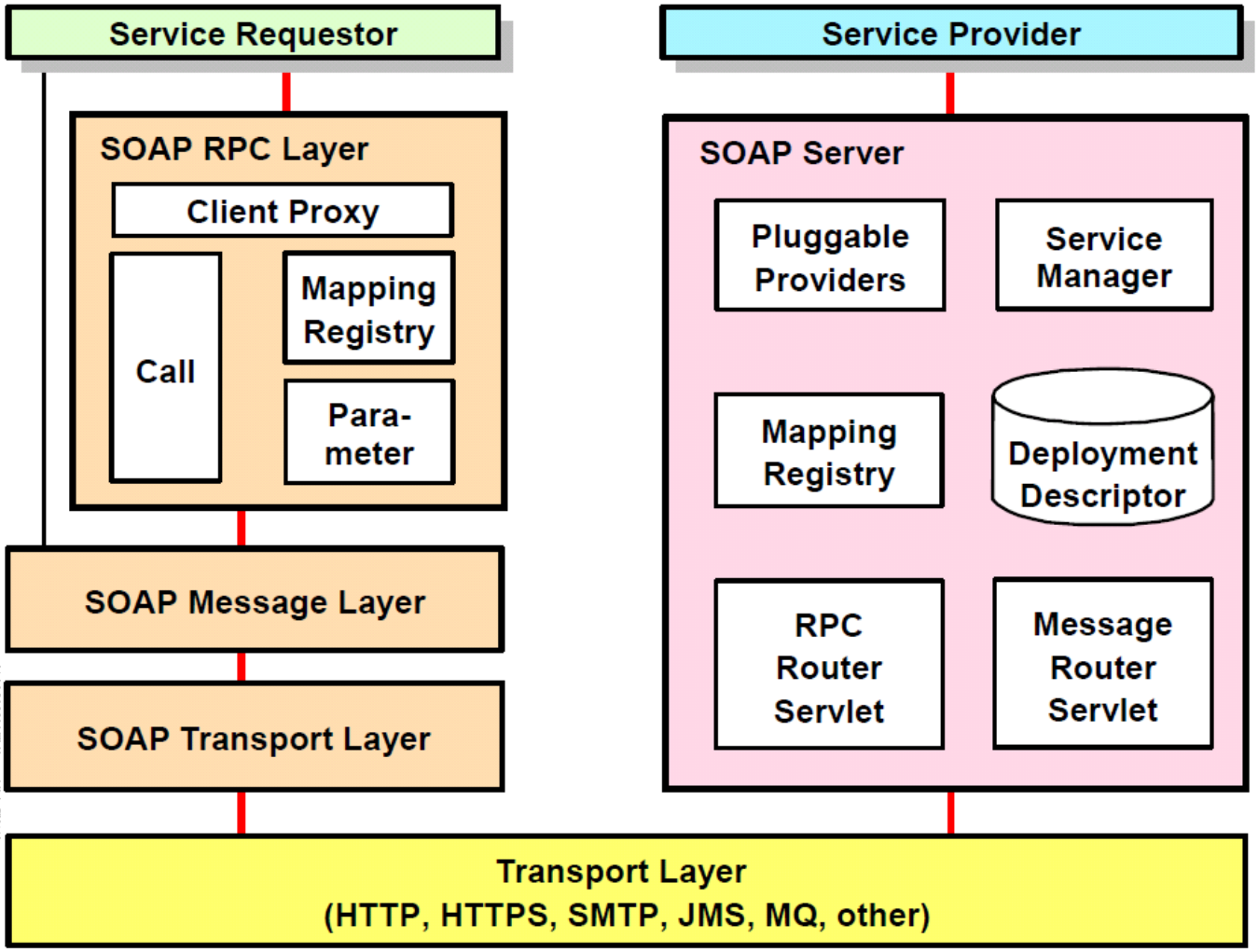


Kódolás/Üzenetváltás módok

- A sorosítás, visszaállítás módját adja meg
- Programozás nyelv független!
- Típusai:
  - SOAP encoding (SOAP adat modell elemek)
  - Literal (XSD) – ezt támogatja a WS-I basic profile
  - Literal XML (nem használják)
- Üzenetváltás módok
  - Document/Literal – a legjobb megoldás Java és nem Java alkalmazások együttműködésére
  - RPC/Literal – Java – Java
  - RPC/Encoded – régi java implementációk
  - Document/Encoded – Nem használt

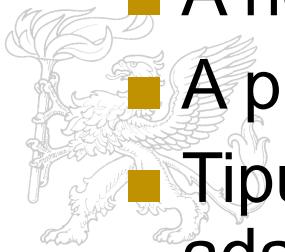


SOAP megvalósítások



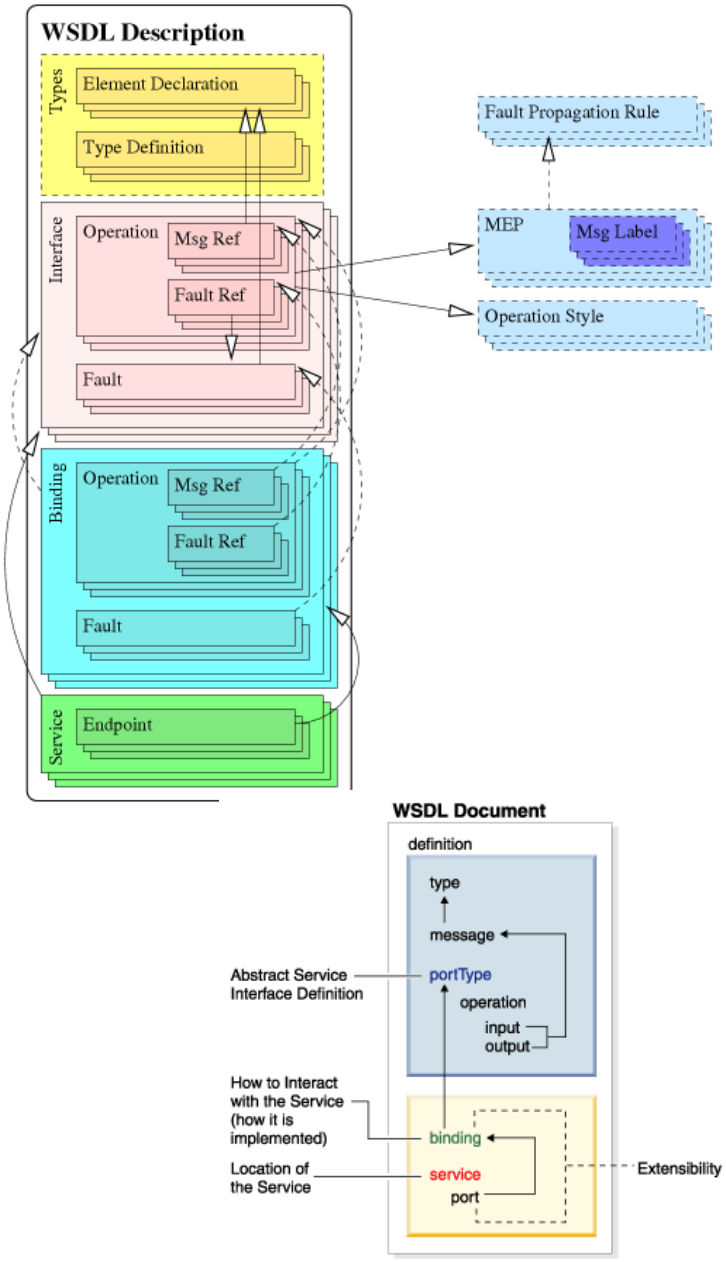
WSDL

- XML alapú
- Megadja, hogy
  - Mit csinál a web szolgáltatás
  - Hol tudjuk elérni
  - Hogyan lehet meghívni
- A web szolgáltatás biztosítója megadhatja:
  - A nevét
  - A protokollt és a kódolást
  - Tipus információkat (műveletek, paraméterek, adattípusok)



A WSDL szerkezete

- Types – adattípus definiciók tárolója. Pl.: XSD
- Message – Az átküldött adat absztrakt típusos megadása
- Port type – egy vagy több prot által támogatott absztrakt műveletek megadása
  - Operation – a szolgáltatás által támogatott akció leírása (kimenő/bejövő üzenet esetleg hiba)
- Binding – Konkrét protokol és adatformátum egy adott prot típushoz. (protokol név, meghívási mód, szolgáltatás id, kódolás)
- Service – összetratozó portok listája
  - Port – egy végpont kötés – hálózati cím összekapcsolása



types

```
<wsdl:types>
  <schema targetNamespace="http://address.jaxrpc.samples"
            xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="AddressBean">
    <sequence>
     <element name="street" nillable="true" type="xsd:string"/>
     <element name="zipcode" type="xsd:int"/>
    </sequence>
   </complexType>
   <element name="AddressBean" nillable="true" type="impl:AddressBean"/>
  </schema>
  <import namespace="http://www.w3.org/2001/XMLSchema"/>
 </wsdl:types>
```

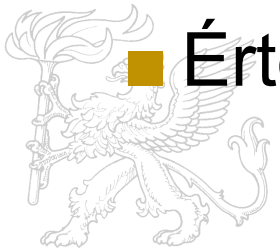
message

- Egy vagy több logikai részt tartalmaz
- Egy interakciót ír le

```
<wsdl:message name="updateAddressRequest">
   <wsdl:part name="in0" type="intf:AddressBean"/>
   <wsdl:part name="in1" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="updateAddressResponse">
   <wsdl:part name="return" type="xsd:string"/>
</wsdl:message>
<wsdl:message name="updateAddressFaultInfo">
   <wsdl:part name="fault" type="xsd:string"/>
</wsdl:message>
```

Port type

- Absztrakt műveletek és a felhasznált absztrakt üzenetek halmaza
- Műveletek
  - Egyirányú
  - Kérés-Válasz
  - Megszólítás-Válasz
  - Értesítés



```
<wsdl:portType name="AddressService">
   <wsdl:operation name="updateAddress" parameterOrder="in0 in1">
      <wsdl:input message="intf:updateAddressRequest"
                  name="updateAddressRequest"/>
      <wsdl:output message="intf:updateAddressResponse"
                  name="updateAddressResponse"/>
      <wsdl:fault message="intf:updateAddressFaultInfo"
                  name="updateAddressFaultInfo"/>
   </wsdl:operation>
</wsdl:portType>
```

Bindings

- Protokol specifikus általános csatoló adatok (pl.: SOAP kommunikációs stílus)

```
<wsdl:binding name="AddressSoapBinding" type="intf:AddressService">
   <wsdlsoap:binding style="rpc"
                     transport="http://schemas.xmlsoap.org/soap/http"/>
   <wsdl:operation name="updateAddress">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="updateAddressRequest">
         <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://address.jaxrpc.samples" use="encoded"/>
      </wsdl:input>
      <wsdl:output name="updateAddressResponse">
         <wsdlsoap:body
                encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                namespace="http://address.jaxrpc.samples" use="encoded"/>
      </wsdl:output>
      <wsdl:fault name="updateAddressFaultInfo">
         <wsdlsoap:body
             encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
             namespace="http://address.jaxrpc.samples" use="literal"/>
      </wsdl:fault>
   </wsdl:operation>
</wsdl:binding>
```

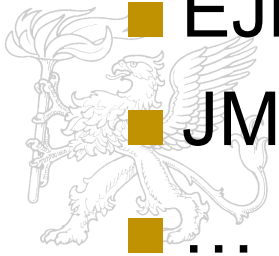
Service definition/port definition

- Szolgáltatás
  - Összefog több portot egy név alatt
- Port
  - Egy konkrét végpont egy konkrét címmel

```
<wsdl:service name="AddressServiceService">
   <wsdl:port binding="intf:AddressSoapBinding" name="Address">
      <wsdlsoap:address
            location="http://localhost:9080/Router/services/Address"/>
   </wsdl:port>
</wsdl:service>
```

WSDL csatolás típusok

- Kiegészítő fejlécek
  - SOAP – binding, operation, body, fault, address, header, headerfault
  - HTTP – get/post (address, binding)
  - MIME – több részből állhat, … (content, multipartRelated, body, mimeXml)
  - EJB
  - JMS
  - …



Kötés

```
public void myMethod(int x, float y);
```

Listing 3. RPC/encoded SOAP message for myMethod

```
<soap:envelope>
    <soap:body>
        <myMethod>
            <x xsi:type="xsd:int">5</x>
            <y xsi:type="xsd:float">5.0</y>
        </myMethod>
    </soap:body>
</soap:envelope>
```

Listing 5. RPC/literal SOAP message for myMethod

```
<soap:envelope>
    <soap:body>
        <myMethod>
            <x>5</x>
            <y>5.0</y>
        </myMethod>
    </soap:body>
</soap:envelope>
```

Listing 7. Document/literal SOAP message for myMethod

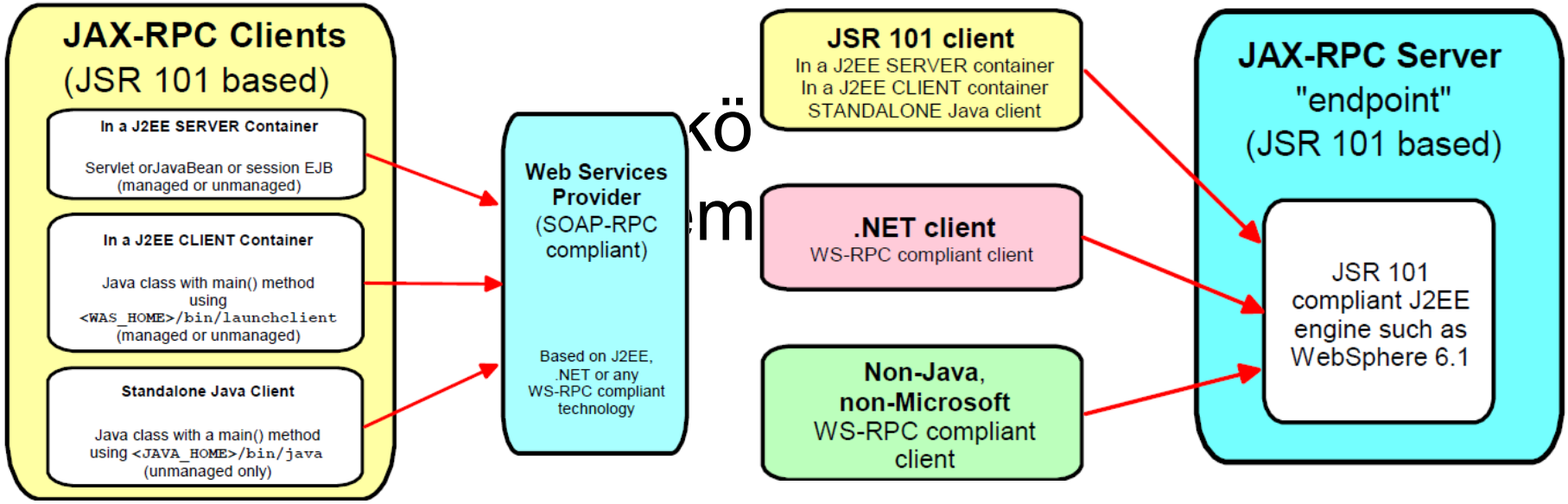
```
<soap:envelope>
    <soap:body>
        <xElement>5</xElement>
        <yElement>5.0</yElement>
    </soap:body>
</soap:envelope>
```

Listing 9. Document/literal wrapped SOAP message for myMethod

```
<soap:envelope>
    <soap:body>
        <myMethod>
            <x>5</x>
            <y>5.0</y>
        </myMethod>
    </soap:body>
</soap:envelope>
```

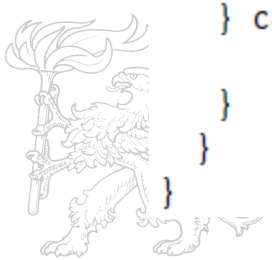
JAX-RPC

- Java API for XML based RPC
- Programozás model a SOAP alapú alkalmazásokhoz
- Leképezést biztosít a Java és a WSDL

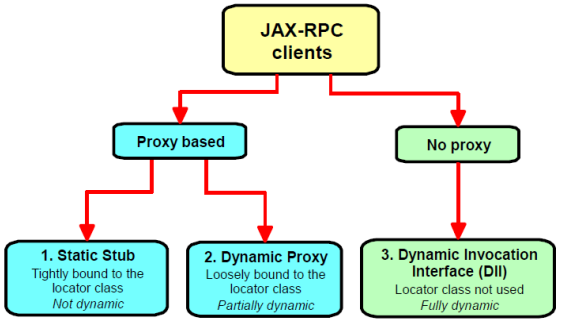


JAX-RPC

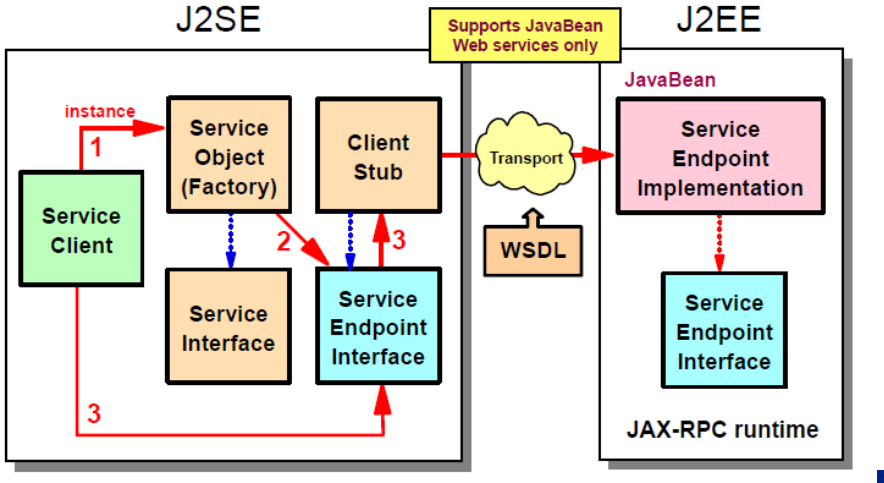
```
package itso.test;

import java.io.*;
import java.util.*;
import itso.test.*;

public class WeatherForecastClient {
  public static void main(String [] args) {
   try{
      WeatherForecastServiceLocator wsl = new WeatherForecastServiceLocator();
      WeatherForecastService ws = (WeatherForecastService) wsl.getWeather();
      String temperature = ws.getTemperature();
      System.out.println(temperature);
      System.out.println("WeatherForecastClient completed");
   } catch (Exception e) {
      e.printStackTrace();
   }
  }
}
```



WS kliensek



- Statikus csonk
  - WSDL-ből generált csonkokat használ
    - Szolgáltatás végpont interfész (SEI)
    - Szolgáltatás interfész (hogyan kapjuk meg a SEI-t)
    - Szolgáltatás kereső osztály (hozzáférés a SEI-hez)
    - Kapcsolódó csonk (az aktuális hívásokat kezeli)



WS kliensek

- Dinamikus proxy
  - A web szolgáltatás cím változhat

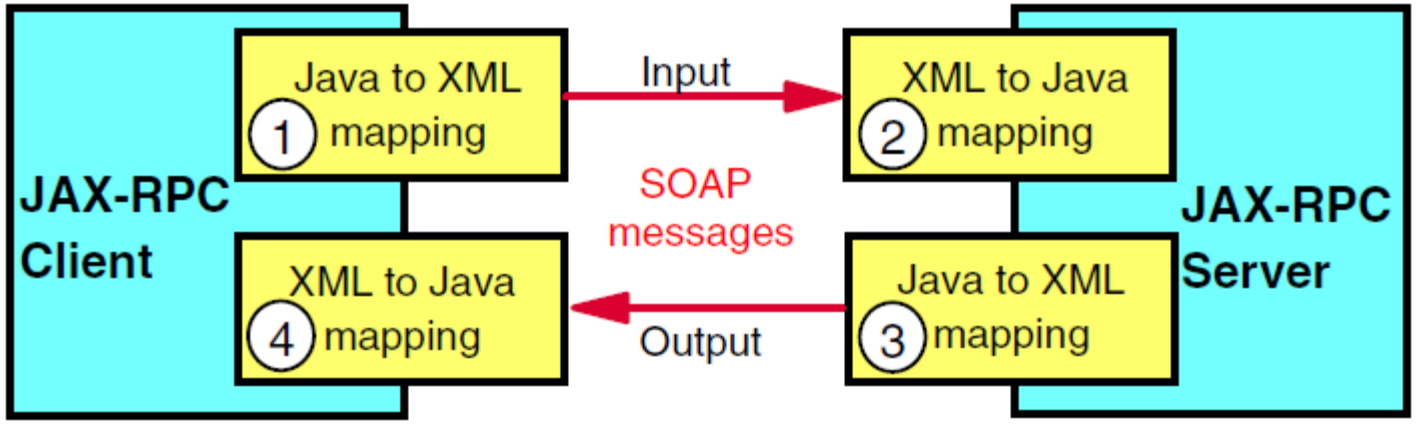
```
import javax.xml.namespace.QName;
import java.io.*;
import java.util.*;

public class WeatherForecastDynamicProxyClient {

  public static void main(String [] args){
   try{
      WeatherForecastServiceLocator wsl = new WeatherForecastServiceLocator();
      QName qn = new QName("http://www.somewhere.com", "WeatherForecast");
      WeatherForecast ws = (WeatherForecast)
                                      wsl.getPort(qn,WeatherForecast.class);
      String temperature = ws.getTemperature();
      System.out.println(temperature);
      System.out.println("DynamicProxyJavaClient completed");
   } catch (Exception e){
         e.printStackTrace();
   }
  }
}
```

WS kliensek

- Dinamikus hívó interfész
  - A WSDL változhat
  - Nem használ proxy fájlokat hanem a WSDL-t használja futás időben

Adat típus csatolás

- Java-XML, XML-Java
- Egyszerű típusok automatikusan
- Egyes adatstruktúrákra is adott



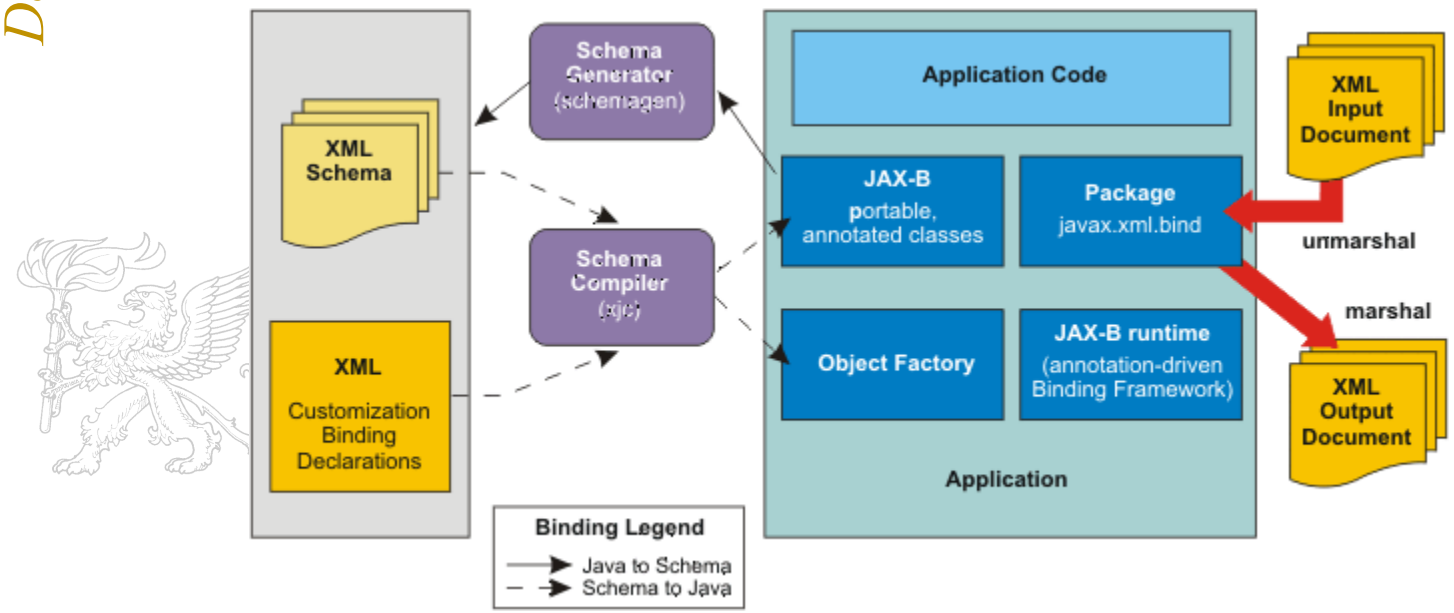
JAX-WS

JAX-RPC 1.1 Code

```
public interface StockQuote extends Remote {
  public float getQuote(String sym) throws
      RemoteException;
}
public class QuoteBean implements {
      public float getQuote(String sym) { ... }
}
```

JAX-WS 2.0 Code

```
@WebService public interface StockQuote {
   public float getQuote(String sym);
}

@WebService public class QuoteBean implements
      StockQuote {
      public float getQuote(String sym) { ... }
}
```

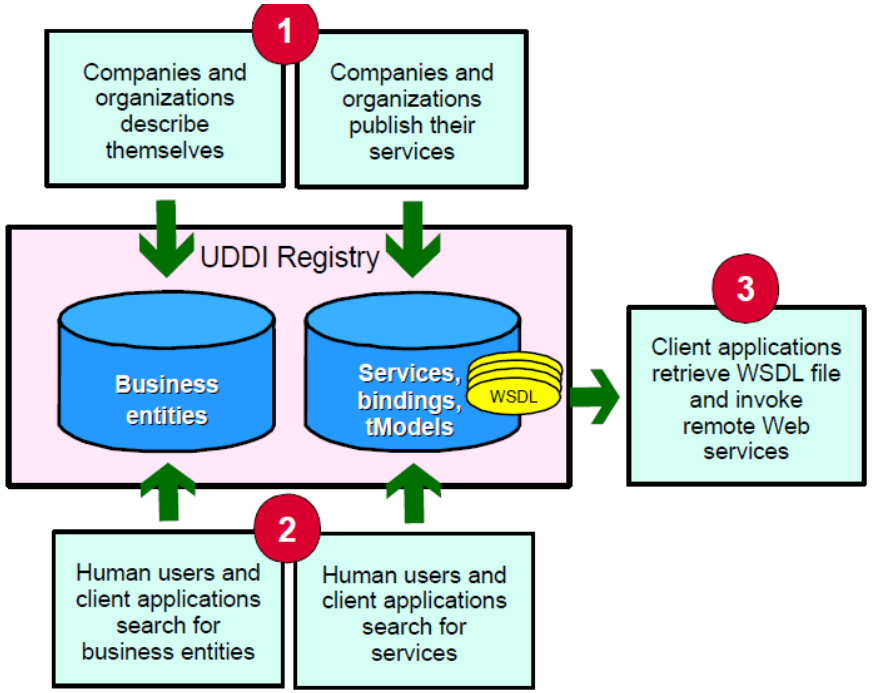


Web Szolgáltatások JEE környezetben

- WSEE
- Hogyan valósítsuk meg a web szolgáltatásokat J2EE környezetben?
- Kliens
- Szerver
  - Web konténer
  - EJB konténer
- Kezelők
  - Egy feldolgozási láncban kezelhetik a SOAP fejléceket
- Tranzakció (a helyi tranzakciókat felfüggesztik)/Biztonság nincs (HTTPS, …) (?)

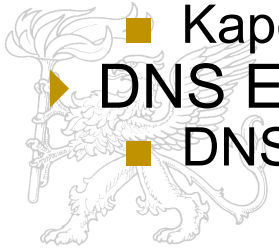
UDDI

- Univerzális Leírás, Felderítés és Integráció
- Segítségével egyszerűbbek a B2B tranzakciók
- UDDI felépítés
  - Üzleti entitás
  - Üzleti szolgáltatás
  - Kötő minta
  - tModel
  - Takszonómia
  - Publákációs megjeg

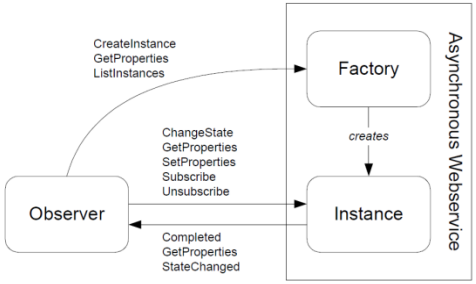


Leírás és felderítés

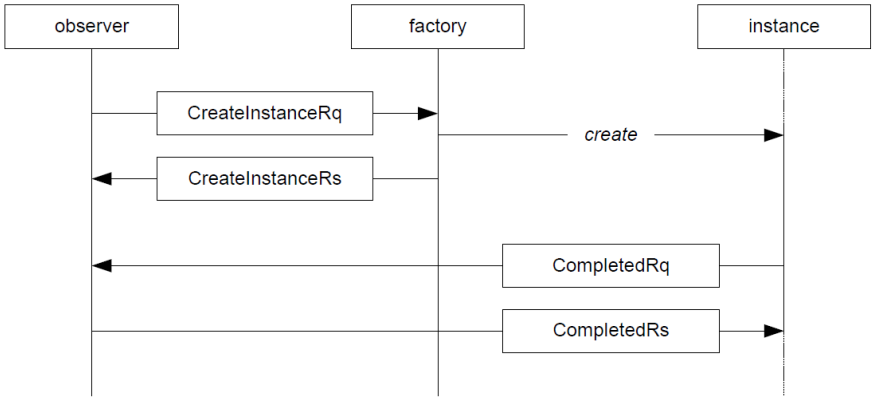
- WS-Inspection: Web Services Inspection Language (WSIL)
  - UDDI nélküli felderítés
- WS-Discovery
  - Többesküldés alapú web szolgáltatás felderítés
- WS-MetadataExchange
  - Üzenetváltás a kezdeti infócseréhez (XSD,WSDL, WS-Policy)
- WS-Policy
  - Szabályok leírása (azonosítás, QoS, …)
- WS-PolicyAssertions
  - Általános követelmény gyűjtemény (szöveg kódolás, …)
- WS-PolicyAttachment
  - Kapcsolatok leírása
- DNS Endpoint Discovery (DNS-EPD)
  - DNS alapú felderítés



Üzenetküldés

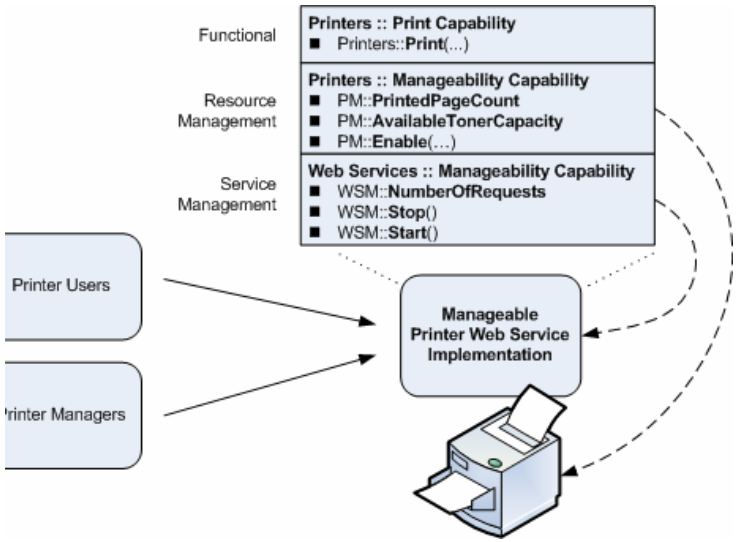


- ASAP: Asynchronous Services Access Protocol
  - Vezérlés, Monitorozés
- SOAP Messages with Attachments (SwA)
  - MIME kezelés
- SOAP Message Transmission Optimization Mechanism
  - Szelektív kódolás
- WS-Addressing
- WS-Notification
  - Publish/Subscirbe
- WS-Eventing
- WS-Enumeration
- WS-MessageDelivery
- WS-ReliableMessaging
- WS-Resources
- WS-Transfer



Menedzsment

- WSDM: Web Services Distributed Management
- WS-Manageability
- SPML: Service Provisioning Markup Language
- WS-Provisioning

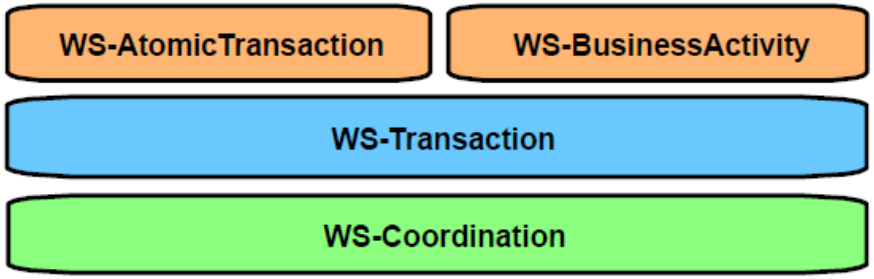


Üzleti folyamatok

- BPEL: Business Process Execution Language
- WS-CDL
- WS-CAF

Tranzakciók

- WS-Coordination (WS-COOR)
- WS-Transaction
- WS-AtomicTransaction (WS-AT)
- WS-BusinessActivity (WS-BA)

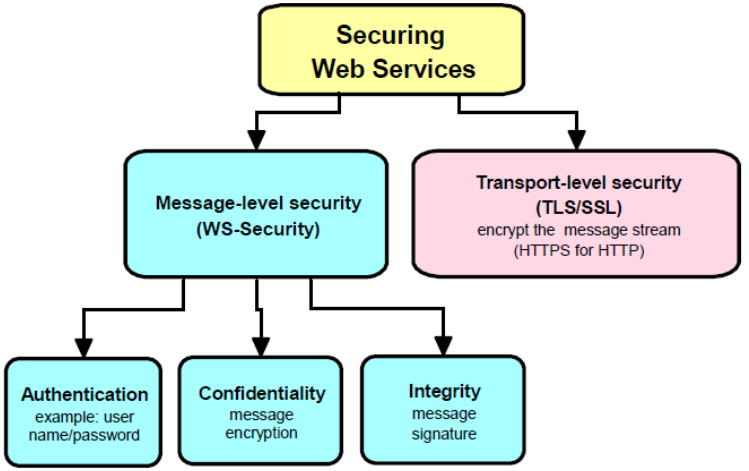


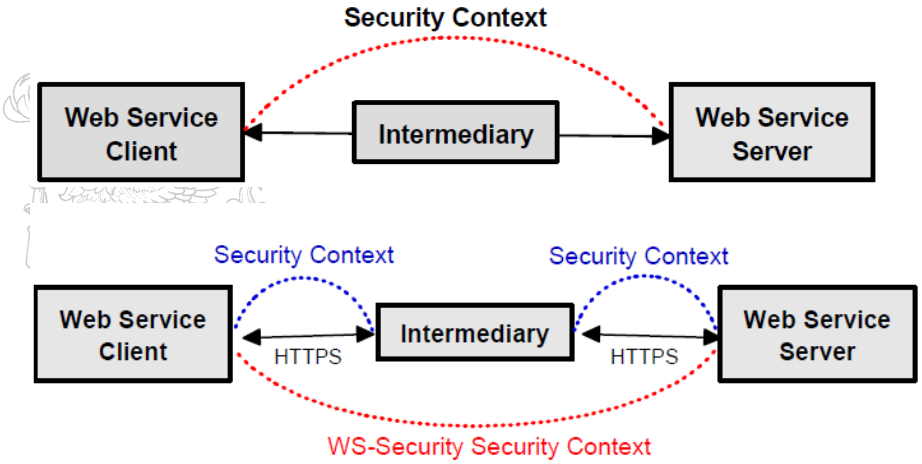
Biztonság

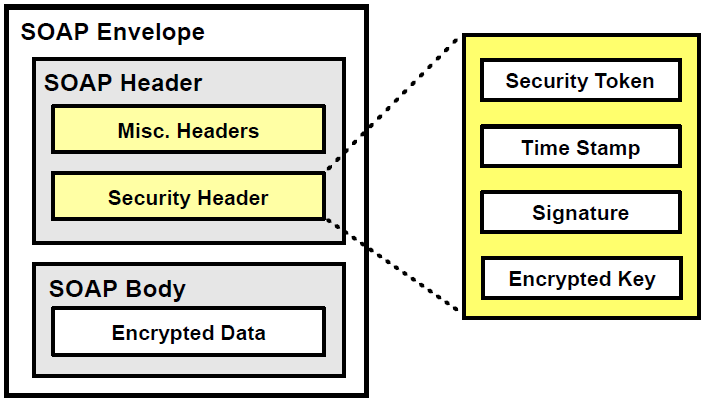
- XML-Encryption
- XML-Signature
- WS-Security
- WS-SecureConversation
- WS-SecurityPolicy
- WS-Trust
- WS-Federation
- SAML: Security Assertion Markup Language

Web Szolgáltatás biztonság

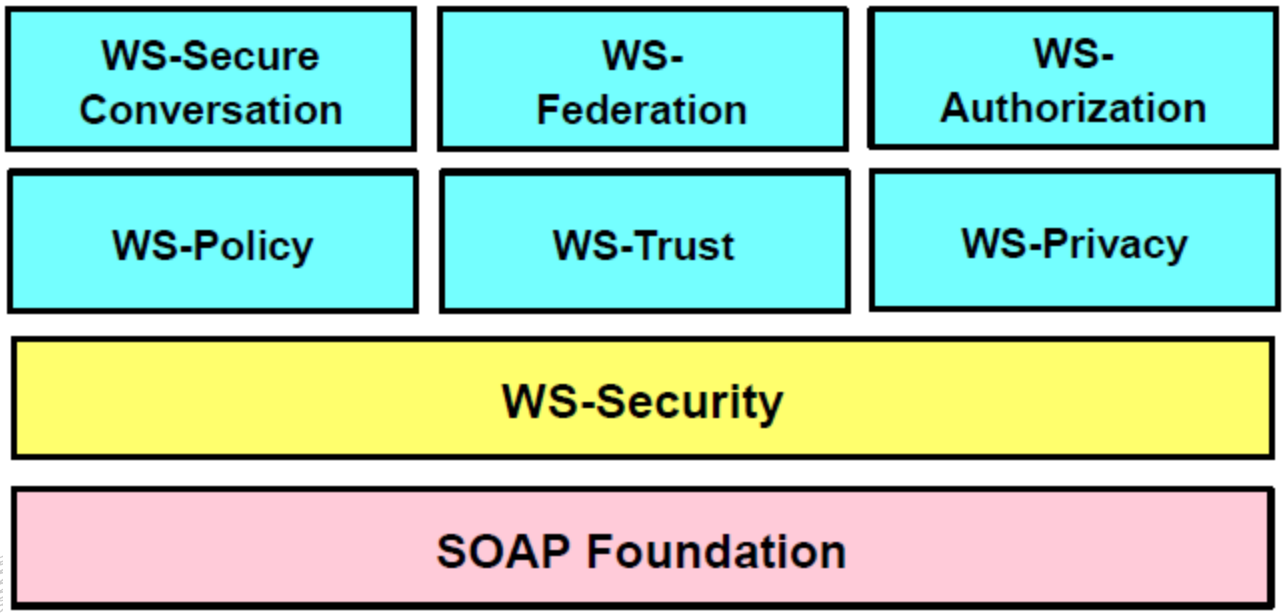
- Tipikus problémák
- Megoldások
  - TLS-SSL
  - WS-Security
    - Üzenet szintű biztonsági beállítások
    - Vég-Vég megoldás







WS-Security

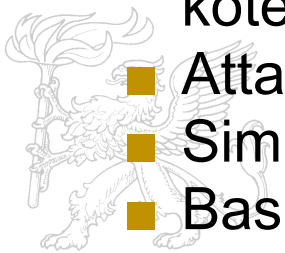

WS-Secure Conversation
WS-Federation
WS-Authorization
WS-Policy
WS-Trust
WS-Privacy
WS-Security
SOAP Foundation

Példa

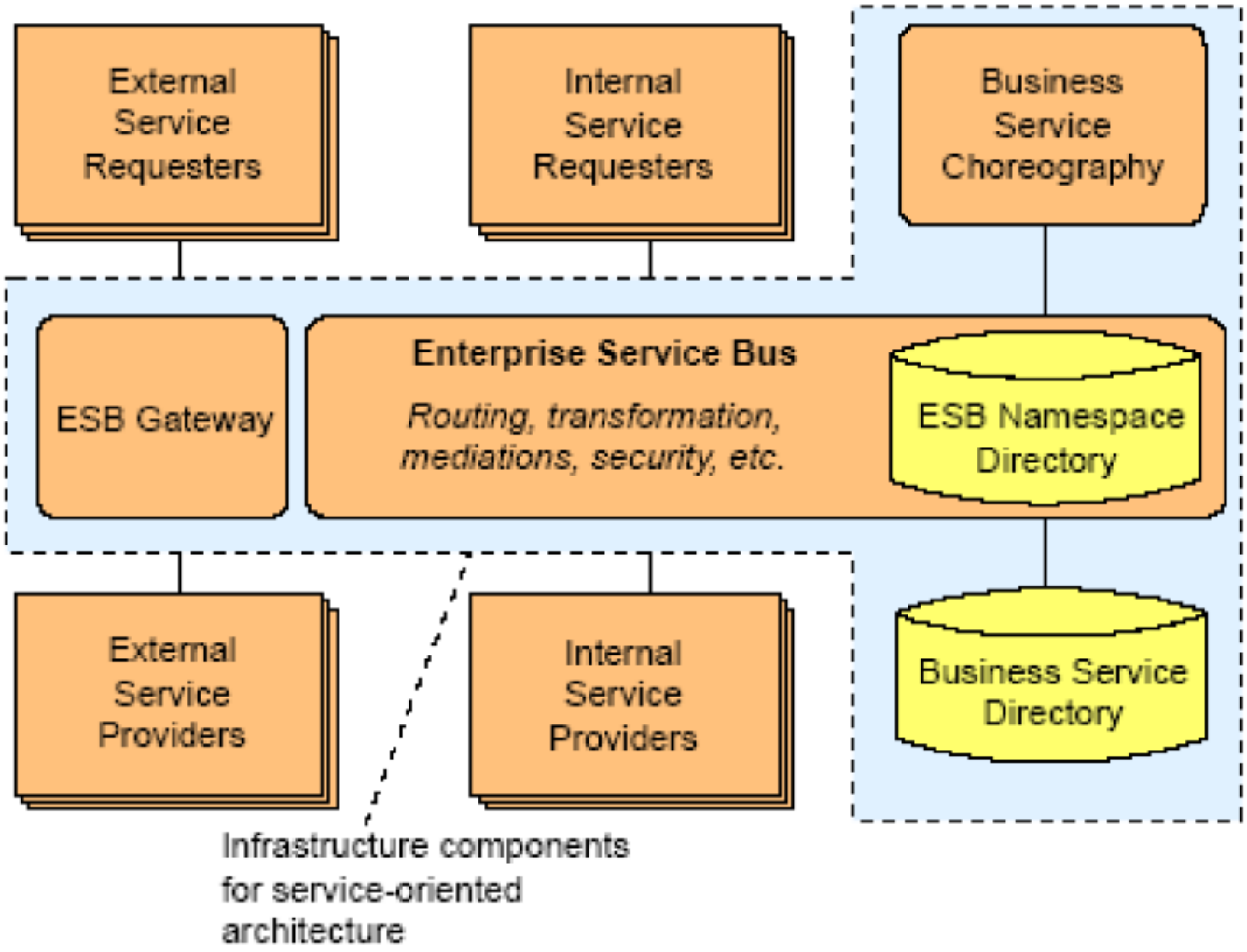
```
<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <soapenv:Header>
      <wsse:Security soapenv:mustUnderstand="1"
          xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-secext-1.0.xsd">
         <wsse:UsernameToken>
            <wsse:Username>David</wsse:Username>
            <wsse:Password
                Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
                username-token-profile-1.0#PasswordText">divaD</wsse:Password>
         </wsse:UsernameToken>
      </wsse:Security>
   </soapenv:Header>
   <soapenv:Body>
      <p821:getDayForecast xmlns:p821="http://bean.itso">
         <theDate>2004-11-25T15:00:00.000Z</theDate>
      </p821:getDayForecast>
   </soapenv:Body>
</soapenv:Envelope>
```

WS-I Web Szolgáltatások együttműködése

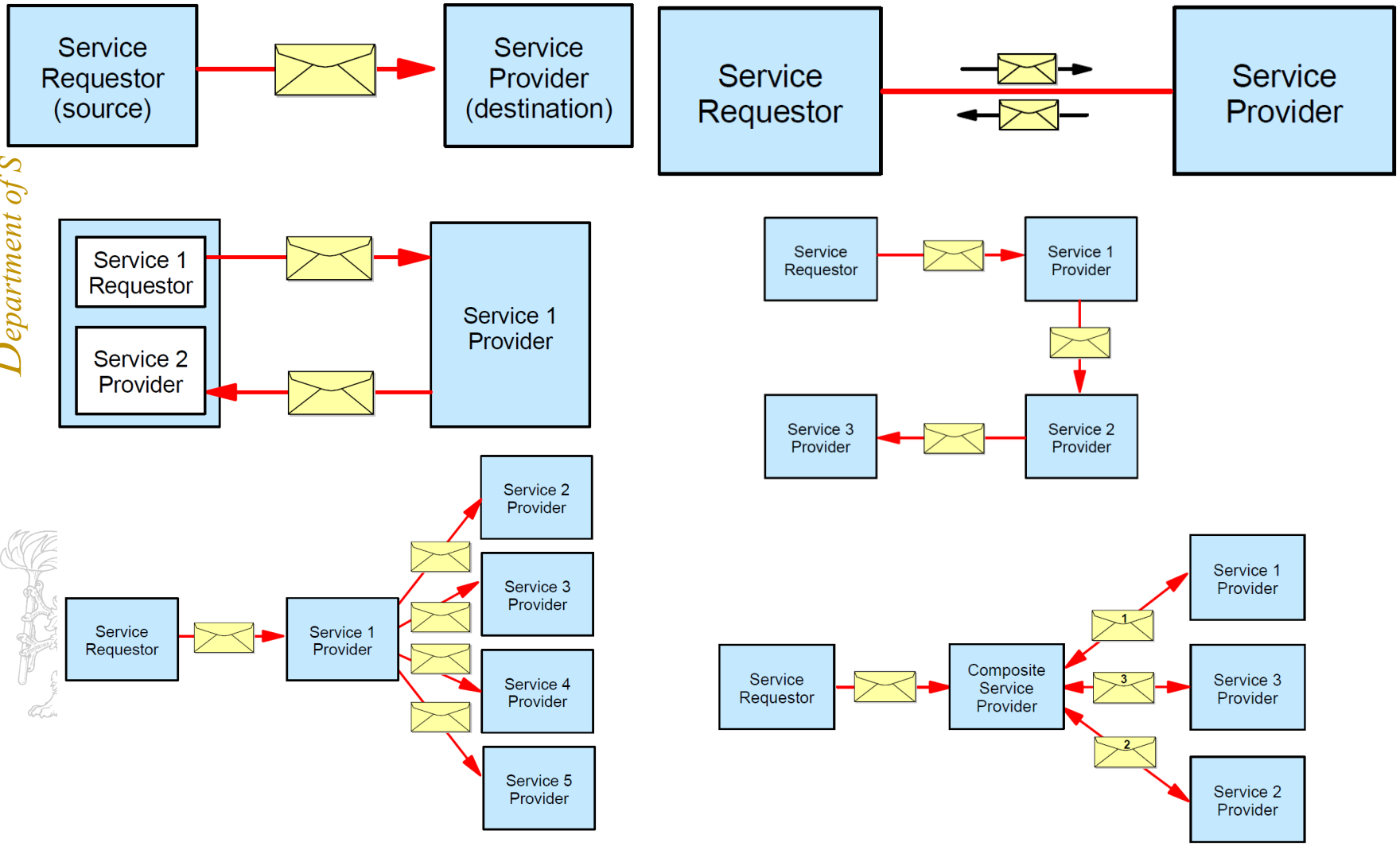
- A web szolgáltatás elvileg azért jó mert platform független , …
- Sok SOAP megvalósítása
- Sok szabványosítási testület (OASIS, IETF, W3C, …)
- WS-I együttműködési minimum specifikálása
- WS-I profilok
  - Implementációs javaslatok
  - Basic Profile v1.1 (pl.: document/literal vagy RPC/literal kötelező, SOAP/HTTP kötés, HTTP POST metódus, …)
  - Attachements Profile v1.0
  - Simple SOAP binding Profile v1.0
  - Basic Security Profile
- Minta alkalmazások
- Teszt eszközök



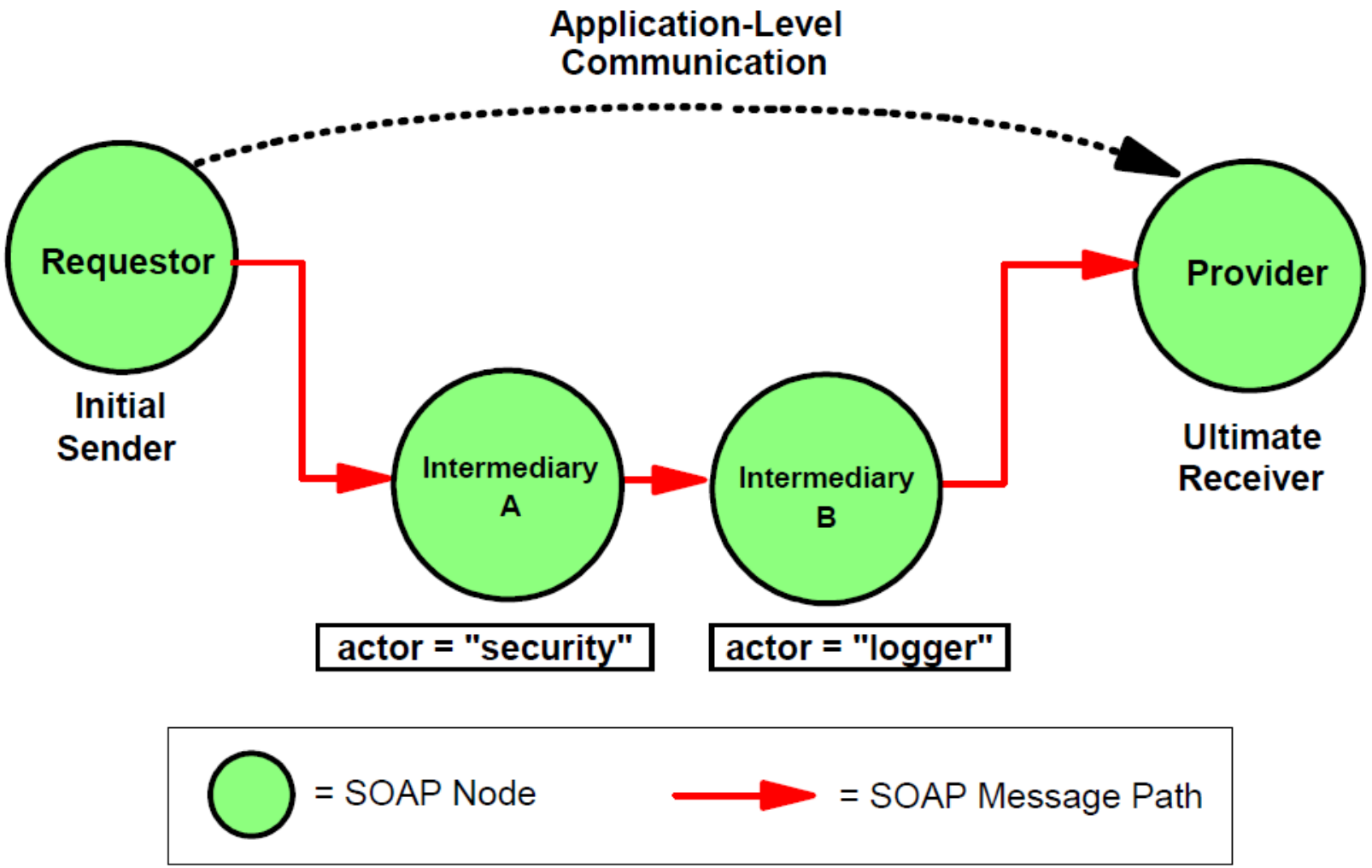
WS architektúra



MEP


Service Requestor (source)
Service Provider (destination)
Service Requestor
Service Provider
Service 1 Requestor
Service 2 Provider
Service 1 Provider
Service Requestor
Service 1 Provider
Service 3 Provider
Service 2 Provider
Service Requestor
Service 1 Provider
Service 2 Provider
Service 3 Provider
Service 4 Provider
Service 5 Provider
Service Requestor
Composite Service Provider
1
3
2
Service 1 Provider
Service 3 Provider
Service 2 Provider

SOAP modell


Application-Level Communication
Requestor
Initial Sender
Intermediary A
Intermediary B
Provider
Ultimate Receiver
actor = "security"
actor = "logger"
= SOAP Node
= SOAP Message Path

Összefoglaló

- Bevezető
- Web Szolgáltatás szabványok
- SOAP
- WSDL
- JAX-RPC
- JEE – WS
- UDDI
- WS profilok
  - WS-Security
  - WS-Interoperability
- Web Szolgáltatás architektúrák