

06α Αρχιτεκτονικά πρότυπα στον Παγκόσμιο Ιστό

Τεχνολογία Λογισμικού

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Χειμερινό εξάμηνο 2017-18

Δρ. Κώστας Σαΐδης (saiko@di.uoa.gr)

Περιεχόμενα

- Ο Παγκόσμιος Ιστός
- Διεπαφές χρήστη στον Παγκόσμιο Ιστό
 - Η τεχνική AJAX
 - Τα πρότυπα MVVM και Observable
 - Ο μορφότυπος JSON
- Υπηρεσιοστραφής αρχιτεκτονική (Service-oriented architecture)
- Το αρχιτεκτονικό στυλ REST
- Μικρουπηρεσίες (Microservices)

Ο Παγκόσμιος Ιστός

Βασικά συστατικά

- Client/Server
- HTTP (Hyper Text Transfer Protocol)
- HTML (Hyper Text Markup Language)
- Unified Resource Identifiers, Locators and Names (URIs, URLs, URNs)

Μορφή ενός URL

```
scheme://[user:password@]host[:port]/path[?query][#fragment]
```

HTTP

- Μέθοδοι (HEAD, GET, PUT, POST, DELETE, κ.ά)
- Headers (Host, Accept, Cookie, κ.ά)
- Status codes (200, 404, 500, κ.ά)
- Εκδόσεις: 1.0, 1.1, 2.0

Παράδειγμα

Αίτηση

```
GET /index.html HTTP/1.1  
<line feed>
```

Απάντηση

```
HTTP/1.1 200 OK  
Date: Wed, 29 Mar 2017 14:38:00 GMT  
Server: Apache/1.3.27 (Unix) ...  
Last-Modified: Wed, 29 Mar 2017 01:16:05 GMT  
Accept-Ranges: bytes  
Content-Length: 6188  
Content-Type: text/html  
<html>  
<head>...</head><body>...</body>  
</html>
```

Μετασχηματισμός

- Από στατικές σελίδες & δυναμικά web sites
- Σε διαδραστικά, προσαρμόσιμα και ταχέως αποκρίσιμα Web applications που παρέχουν ή χρησιμοποιούν Web APIs
- Ο Παγκόσμιος Ιστός μετασχηματίζεται σε μια προγραμματιστική πλατφόρμα υπηρεσιών με τη Javascript να είναι η defacto γλώσσα

Διεπαφές χρήστη στον Παγκόσμιο Ιστό

- Η τεχνική AJAX
- Το πρότυπο Model-View-ViewModel
- Το πρότυπο Observable
- Τεχνικές ασύγχρονου προγραμματισμού (Promises) *
- Reactive Programming*
- Ο μορφότυπος JSON

*Σε επόμενη διάλεξη

AJAX

- Asynchronous Javascript and XML (JSON)
- Αποστολή του HTTP αιτήματος ασύγχρονα (σε ξεχωριστό thread) και παροχή callback hook για το αποτέλεσμα

Προ-AJAX

- Αργά και δύσχρηστα User Interfaces
- Δεν ήταν εφικτό να υλοποιηθεί αποτελεσματικό User Experience!
 - Π.χ. auto-complete as you type του google search.
 - Γιατί;

Παράδειγμα

<https://jsfiddle.net/2tqja09e/2577/>

Το Observable πρότυπο σχεδίασης

- Push paradigm
- Ο χρήστης/client του Observable (ο observer) ενημερώνεται για την αλλαγή της τιμής του observable (το observable κάνει push)
- Παρόμοια λογική με events & event handling

java.util.Observable

```
class Observable {  
    void addObserver(Observer o);  
    void deleteObserver(Observer o);  
    boolean hasChanged();  
    void notifyObservers();  
    void notifyObservers(Object arg);  
}
```

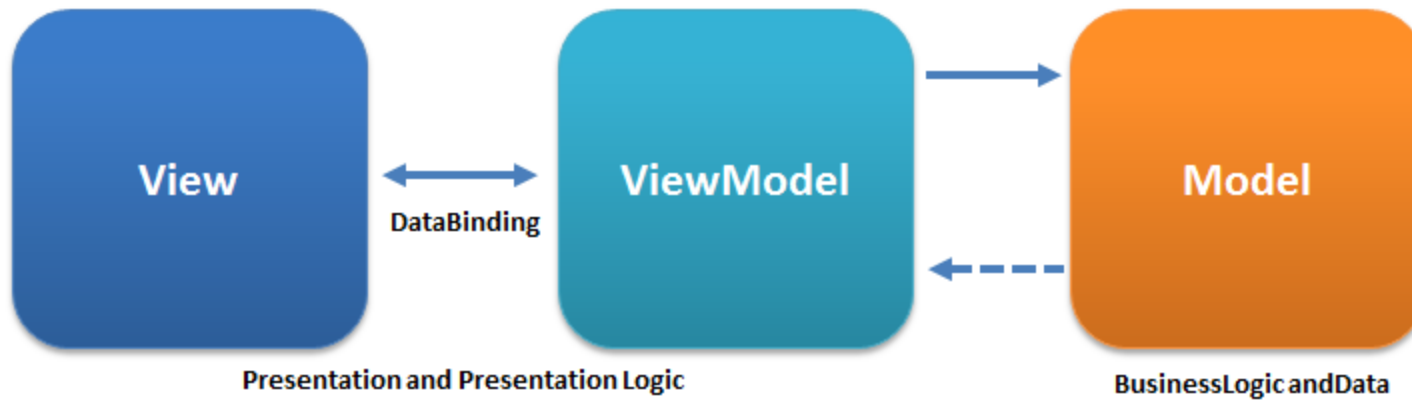
Observer

java.util.Observable

```
interface Observer {  
    void update(Observable o, Object arg);  
}
```

Το MVVM αρχιτεκτονικό πρότυπο

Παραλλαγή του Model-View-Controller

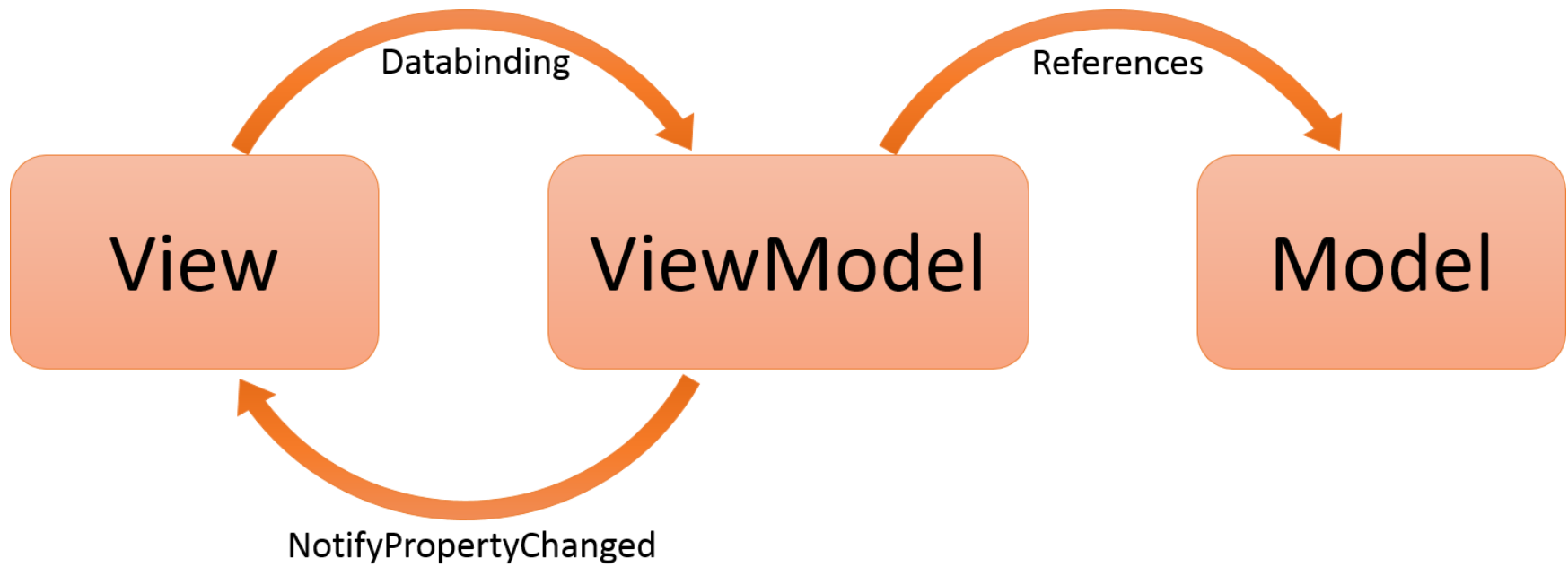


Συστατικά

- View: τα στοιχεία του UI
- Model: τα δεδομένα
- ViewModel (Presenter ή ViewController):
 - Data bindings (model elements <-> UI elements)
 - Change notifications

Εφαρμογή του Observable στο MVVM

- ViewModel: Ενθυλακώνει τα δεδομένα (model elements) σε observables
- Τα UI elements γίνονται bind ως observers στα model elements



Παράδειγμα

<http://knockoutjs.com/examples/>

Συζήτηση

- Δηλωτικός κώδικας
- Υψηλό επίπεδο αφαίρεσης
- Αυτόματη -για τον προγραμματιστή- ενημέρωση του UI με τις αλλαγές στα δεδομένα
- Σύνθετο για απλές διεπαφές χρήσης
- Θέματα απόδοσης σε πολύ μεγάλες εφαρμογές ή σύνολα δεδομένων

Ο μορφότυπος JSON

JavaScript Object Notation (JSON)

- Text-based πρότυπο για την κωδικοποίηση δεδομένων
 - ECMA-262 3η έκδοση, rfc 4627, ECMA-404
 - Κωδικοποίηση UNICODE (UTF-8, UTF-16, UTF-32)
- Βασίζεται σε υποσύνολο της JavaScript, αλλά είναι πλήρως ανεξάρτητο της γλώσσας προγραμματισμού
- Το πλέον διαδεδομένο πρότυπο για:
 - την ασύγχρονη επικοινωνία μεταξύ Web Browser και Web Server
 - επικοινωνία μεταξύ Web APIs (services)

Οφέλη

- Απλότητα
 - Αναγνώσιμο από τον άνθρωπο
 - Επεξεργάσιμο από οποιαδήποτε γλώσσα προγραμματισμού
- Σταθερότητα
 - Δεν υπάρχουν διαφορετικές εκδόσεις
 - Δεν αναμένονται αλλαγές στο συνακτικό
- Ταχύτερο κι απλούστερο από την XML
 - WSDL, XML Web Services

Τύποι δεδομένων

- Πρωτογενείς τύποι
 - String
 - Number
 - Boolean
- null
- Σύνθετοι τύποι
 - Array
 - Object

JSON String

- Ακολουθία 0 ή περισσότερων UNICODE χαρακτήρων
- Περιέχονται σε διπλά εισαγωγικά "..."
- Backslash escapes
- Παράδειγματα
 - ""
 - "data"
 - "data \"with quotes\""

JSON Number

- Χωρίς εισαγωγικά
- Προσημασμένος δεκαδικός αριθμός (διπλής ακρίβειας, κινητής υποδιαστολής)
- Παραδείγματα:
 - 10 (ακέραιος)
 - -10.45 (πραγματικός)
 - 2.5e-5 (scientific notation)

JSON Boolean

- Χωρίς εισαγωγικά
- true ή false

JSON null

- Χωρίς εισαγωγικά
- null

JSON Array

- Διατεταγμένο σύνολο τιμών
- Οι τιμές εμπεριέχονται μεταξύ `[]` και χωρίζονται με `,`
- Παραδείγματα:
 - Πίνακας με Strings: `["red", "green", "blue"]`
 - Πίνακας με Numbers: `[10, -20, 0.30]`
 - Πίνακας με διάφορους τύπους: `[10, "red", false, null]`

JSON Object

- Μη διατεταγμένο σύνολο από ζεύγη ονόματος/τιμής (unordered name/value pairs)
- Ένα JSON Object έχει 0 ή περισσότερα ζευγάρια που εμπεριέχονται σε `{}`
- Κάθε ζευγάρι έχει ένα όνομα και μία τιμή
- Το όνομα είναι τύπου String
- Η τιμή μπορεί να είναι οποιουδήποτε τύπου (String, Number, Boolean, null, Object, Array)
- Το όνομα διαχωρίζεται από την τιμή με `:`
- Τα ζευγάρια διαχωρίζονται μεταξύ τους με `,`

Παραδείγματα

```
{"img":"web.jpg", "width":800, "height":600}
```

```
{  
  "user": "saiko",  
  "enabled": true,  
  "favorites": [10, 11, 20],  
  "roles": [  
    {"id":"editor", "text":"Content Editor"},  
    {"id":"manager", "text":"System Manager"}  
  ]  
}
```

Μορφότυπος JSON

- Εύκολα αναγνώσιμο/διαχειρίσιμο και από ανθρώπους και από προγράμματα
- Αναπαριστά εύκολα τις ευρέως χρησιμοποιούμενες δομές δεδομένων (records, lists, trees)
- Οι αλγόριθμοι ανάλυσης (parsing) είναι απλοί, γρήγοροι και συνεπείς λόγω του απλού συντακτικού
- Υποστήριξη UNICODE
- MimeType: application/json

JSON with Padding/Prefix (JSONP)

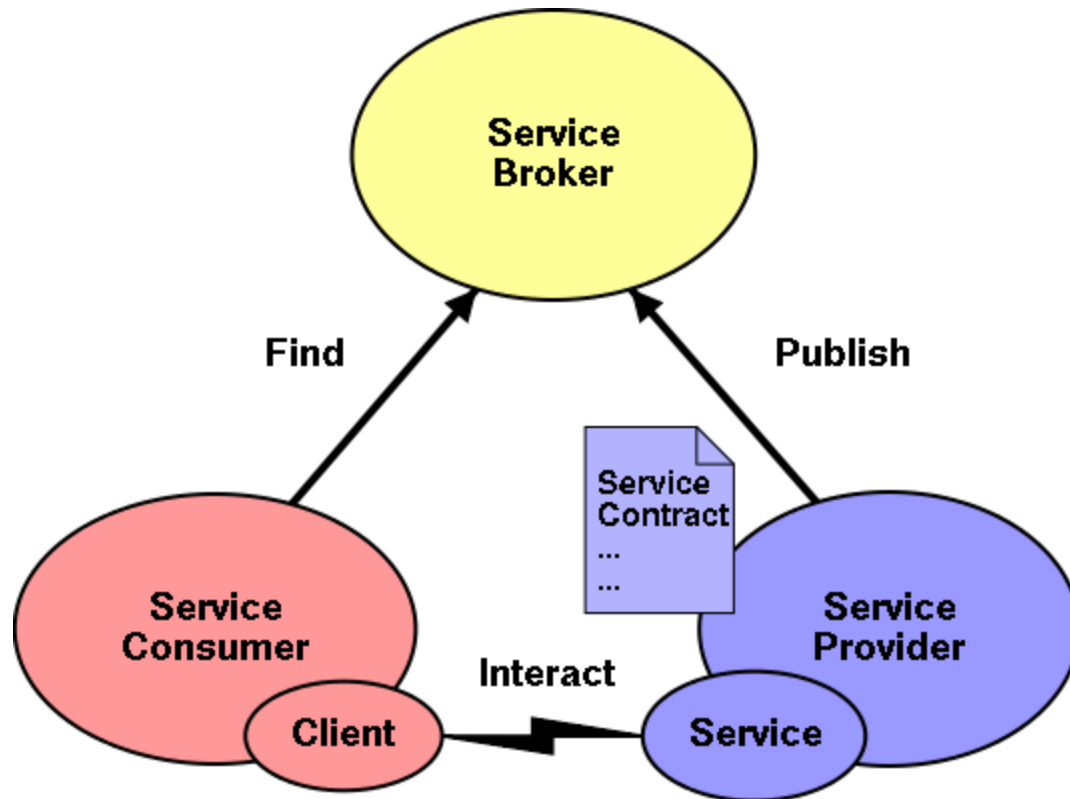
- Τα AJAX calls περιορίζονται από το same-origin policy (το ποιό;)
- Μηχανισμός για να υλοποιηθούν cross-origin λειτουργίες, συνήθως με τη χρήση ενός callback function (δηλαδή;)
- MimeType: application/javascript
- Παράδειγμα;

Service-oriented architecture

Η κεντρική ιδέα

- Υπηρεσίες που επικοινωνούν μέσω ενός πρωτοκόλλου επικοινωνίας και είναι:
 - κατανεμημένες,
 - αυτοτελείς (separately maintained & deployed)
 - χαλάρα συνδεδεμένες (loosely-coupled)
 - ανεξάρτητες της τεχνολογίας υλοποίησης (technology-neutral)
 - ανεξάρτητες του κατασκευαστή (no vendor lock-in)
- Σύνθεση της εφαρμογής μέσω της ολοκλήρωσης (integration) των υπηρεσιών.

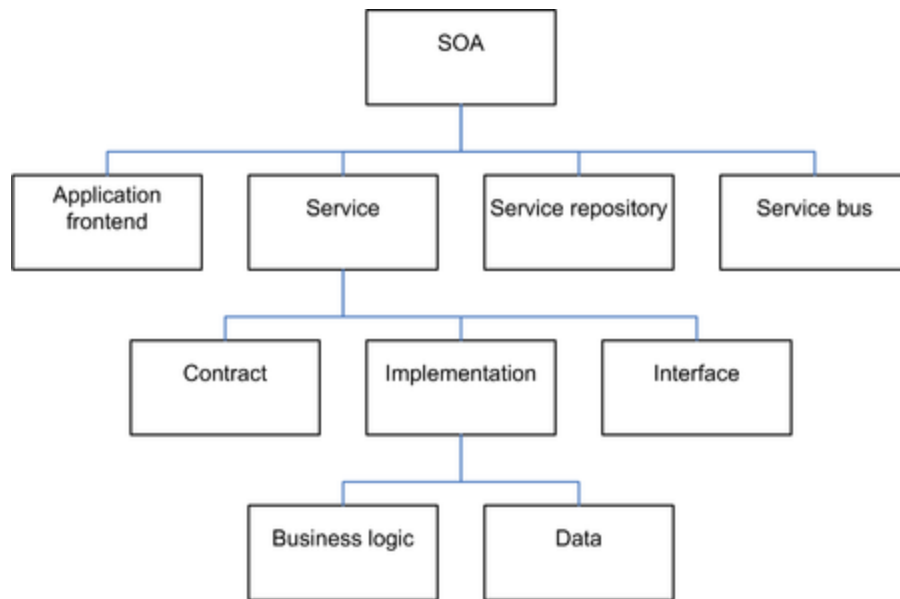
Ρόλοι



- Service consumer
- Service producer
- Service broker

SOA Αρχές

- Service contract
- Metadata
- Composition
- Autonomy
- Discovery
- Reusability



Υλοποίηση

- Web Services (SOAP, WSDL, UDDI)
- Remote-procedure call (RPC)
- Message-driven middleware
- RESTful APIs
- κ.ά

Ζητήματα

- Stateful vs Stateless
- Απόδοση
- Πολυπλοκότητα
- Έλεγχος και επαλήθευση (testing)

Η Αρχιτεκτονική Representational State Transfer (REST)

RESTful APIs

- Υπάρχουν παντού (Web, Microservices, IoT)
- Σχεδόν πάντα με υποστήριξη για JSON

Τι είναι το REST

- Ένα αρχιτεκτονικό στυλ για τη λειτουργία του Παγκόσμιου Ιστού (ή, γενικά, κατανεμημένων συστημάτων)
- Οριοθετεί αρχές, περιορισμούς και βασικές λειτουργίες
- Ανεξάρτητα της γλώσσας προγραμματισμού, του πρωτοκόλλου επικοινωνίας ή του είδους των δεδομένων

ΣΥΝΟΠΤΙΚΑ

Τέσσερις βασικές έννοιες:

- Resources, Representations, Requests, Responses

Έξι βασικές αρχές:

- Client-server, Stateless, Cacheable, Layered System, Uniform Interface, Code on demand (προαιρετικά)

Οφέλη

- Η σωστή χρήση των RESTful αρχών βελτιώνει όλα τα σημαντικά χαρακτηριστικά μιας αρχιτεκτονικής:
 - Απόδοση
 - Κλιμάκωση
 - Απλότητα
 - Επεκτασιμότητα
 - Αξιοπιστία

1η RESTful Αρχή

Client – Server

- Πλήρης διαχωρισμός ενδιαφερόντων (separation of concerns) μεταξύ client & server
- Χρήση ομοιόμορφης διεπαφής (uniform interface) για την επικοινωνία τους

Παράδειγμα: ο client δεν πρέπει να «ενδιαφέρεται» για το data storage, ο server δεν πρέπει να «ενδιαφέρεται» για το user interface

2η RESTful Αρχή

Stateless

- Ο server δεν αποθηκεύει καμιά πληροφορία για το state της εφαρμογής κατά την επικοινωνία του με τον client
- Το ακριβές application state τηρείται μόνο από τον client

Αλλά, προφανώς, ο server μπορεί να αποθηκεύει πληροφορίες του user session σε κάποια βάση δεδομένων

3η RESTful Αρχή

Cacheable

- Ο client πρέπει να μπορεί να αποθηκεύσει προσωρινά επιλεγμένα responses από τον server
- Ο server πρέπει να υποδεικνύει στον client ποια responses μπορούν να αποθηκευθούν προσωρινά και ποια όχι

Παράδειγμα: χρήση κατάλληλων cache headers του HTTP

4η RESTful Αρχή

Layered System

- Ο client πρέπει να μπορεί να συνδεθεί είτε με τον end server είτε με κάποιον ενδιάμεσο server, χωρίς να μπορεί να «διακρίνει» τη διαφορά
- Οι ενδιάμεσοι server πρέπει να μπορούν να προσθέτουν λειτουργικότητα: π.χ. caching, authorization, security, κτλ.

Παράδειγμα: Web proxies, N-tier architectures

5η RESTful Αρχή

Code on demand

- Ο client πρέπει να μπορεί να εκτελέσει κώδικα που προέρχεται από τον server δυναμικά (π.χ. applets, scripts, κτλ.)
- Προαιρετικά

6η RESTful Αρχή

Uniform Interface

- Η πλέον σημαντική αρχή!
- Αναγνωριστικά πόρων (Resource Identification)
- Διαχείριση πόρων μέσω αναπαραστάσεών τους (representations)
- Hypermedia as the engine of application state (HATEOAS)

Με απλά λόγια

- Ο client στέλνει requests στον server χρησιμοποιώντας URIs για τον προσδιορισμό των resources (και των representations).
- Τα resources διαχωρίζονται από τα representations: ο server μπορεί να στείλει τα στοιχεία ενός resource σε μορφή XML, JSON, κτλ.
- Ο client, κατέχοντας οποιοδήποτε representation ενός resource, μαζί με ενδεχόμενα metadata, μπορεί να τροποποιήσει το αντίστοιχο resource.

Με απλά λόγια (συνέχεια)

- Τα responses που στέλνει ο server έχουν όλη την απαιτούμενη πληροφορία που απαιτείται για το σωστό τους χειρισμό από τον client (μορφότυπο, caching, κτλ.)
- HATEOAS: ο client επικοινωνεί με τον server μόνο μέσω δυναμικά παρεχόμενων hypermedia (π.χ. hyperlinks).
- Δηλαδή, ο client δεν χρειάζεται να γνωρίζει από πριν κάποιο server ή κάποιο interface (όπως στις παραδοσιακές SOA ή στο RPC, κτλ.). Αρκεί μόνο να μπορεί να «καταλάβει» τη λειτουργία των hypermedia.

RESTful Web Service (API)

Αποτελείται από:

- Ένα HTTP base URL (REST endpoint)
- Ένα ή περισσότερα MIMEType για τα representations
- HTTP Methods: GET, PUT ή PATCH, POST, DELETE

Παράδειγμα

- Έστω μια συλλογή (collection) από στοιχεία (items).
- Υπάρχουν δύο βασικά endpoints:
 - `[baseUrl]/items` : Ενέργειες στη συλλογή (εν συνόλω).
 - `[baseUrl]/items/id` : Ενέργειες στο στοιχείο (ειδικά).
- REST Calls
 - `GET [baseUrl]/items`
 - `PUT [baseUrl]/items/32`

Συλλογές

| HTTP Method | Ενέργεια |
|--------------------|---|
| GET | Εμφάνιση της λίστας με τα elements του collection, παρέχοντας το αντίστοιχο URI για το καθένα και ενδεχόμενα πρόσθετα στοιχεία (π.χ. metadata) καθώς και σελιδοποίηση (αν υποστηρίζεται). |
| PUT | Αντικατάσταση του υπάρχοντος collection με νέο collection (αν υποστηρίζεται). |
| POST | Προσθήκη νέου element στο υπάρχον collection. |
| DELETE | Διαγραφή του υπάρχοντος collection (αν υποστηρίζεται). |

Στοιχεία

| HTTP Method | Ενέργεια |
|--------------------|---|
| GET | Ανάκτηση μιας αναπαράστασης του συγκεκριμένου item. |
| PUT ή PATCH | Τροποποίηση του συγκεκριμένου item. |
| POST | Δε χρησιμοποιείται ευρέως για items. |
| DELETE | Διαγραφή του συγκεκριμένου item. |

SOA & REST

- Ένα σύνολο από RESTful HTTP end-points που ανταλλάσσουν δεδομένα σε μορφή JSON
- Ενδεχομένως ο απλούστερος τρόπος υλοποίησης μιας SOA

Microservices

Κωδικοποιημένα

Microservices = SOA + Unix Principles + Agile + DevOps

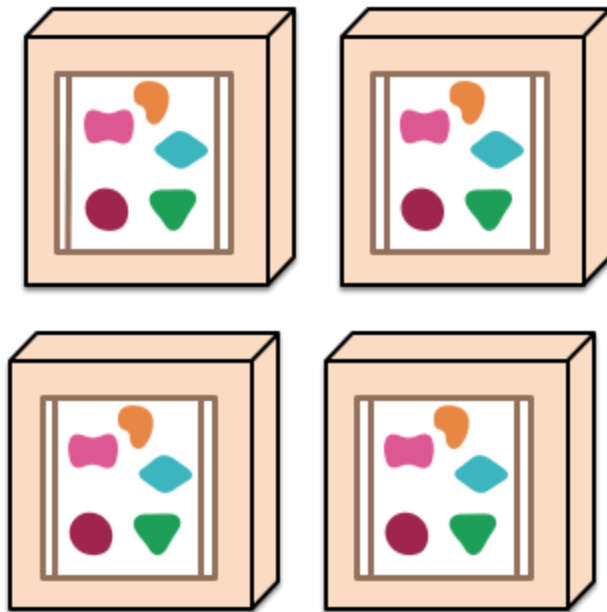
Monolithic Server-side Applications

- A single logical executable
- Cloud deployment issues:
 - Small changes -> rebuild and redeploy the whole app
 - Scalability -> scale it all or not at all

A monolithic application puts all its functionality into a single process...



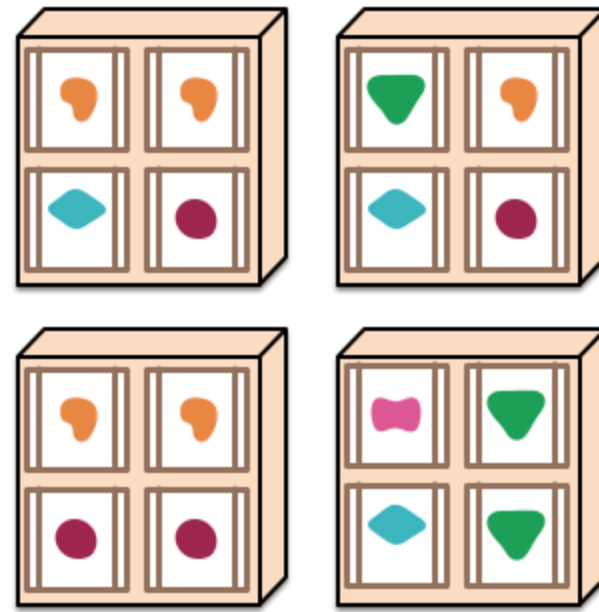
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...



... and scales by distributing these services across servers, replicating as needed.



Χαρακτηριστικά

- Componentization via services
 - As in SOA
- Organized around business capabilities
 - Cross-functional teams, as in Agile
- Products not projects
 - A team should own a product
 - You build it, you run it
- Smart end-points, dumb pipes
 - Simple communication (Unix-style)

- Decentralized governance
 - Each team governs the implementation details of its product
- Decentralized data management
 - Different datastore per service/app
- Infrastructure automation
 - Build & test automation
 - Continuous delivery & deployment
- Design for failure
 - Sophisticated monitoring & logging