Κεφάλαιο 6

Πλοήγηση σε Συνδεδεμένα Δεδομένα

Στα προηγούμενα κεφάλαια γνωρίσαμε τα πρότυπα που εισήγαγε ο Σημασιολογικός Ιστός:

- Το πρότυπο RDF, για την έκφραση της σημασιολογικής πληροφορίας σε μορφή τριάδων.
- Το σχήμα της RDF (RDFS) και τη γλώσσα OWL για την περιγραφή λεξιλογίων και οντολογιών.
- Τη γλώσσα SPARQL για τη διατύπωση ερωτημάτων σε σημασιολογικές βάσεις δεδομένων.

Μάθαμε επίσης για τη σημασία των αναγνωριστικών URIs, τα οποία χρησιμοποιούνται ως αναφορές σε οντότητες οποιασδήποτε μορφής. Τέλος, παρακολουθήσαμε την εξέλιξη του Σημασιολογικού Ιστού στο κίνημα των Συνδεδεμένων Δεδομένων.

Είμαστε πλέον σε θέση να δούμε πώς λειτουργεί μια εφαρμογή που καταναλώνει σημασιολογικά δεδομένα. Στη συνέχεια, αφού θυμηθούμε τις βασικές ιδιότητες του πρωτοκόλλου επικοινωνίας στον παγκόσμιο ιστό (HTTP), θα εξερευνήσουμε τους διαθέσιμους τρόπους ανακάλυψης, λήψης και επεξεργασίας των σημασιολογικών δεδομένων.

6.1 Πώς λειτουργεί μια σημασιολογική εφαρμογή

Θα ονομάσουμε *σημασιολογική εφαρμογή* (semantic application) μία εφαρμογή που καταναλώνει σημασιολογικά δεδομένα, αναγνωρίζει τη σημασία τους και

τα χρησιμοποιεί προς όφελος του τελικού χρήστη. Μια τέτοια εφαρμογή δεν διαφέρει πρακτικά από οποιαδήποτε άλλη εφαρμογή που χειρίζεται δεδομένα στον παγκόσμιο ιστό· είναι ο τρόπος που χρησιμοποιεί τα δεδομένα και τα πρότυπα πάνω στα οποία βασίζει τη λειτουργία της που την ξεχωρίζουν από τις υπόλοιπες εφαρμογές.

Το κύριο γνώρισμα της σημασιολογικής εφαρμογής, η οποία χρησιμοποιεί Συνδεδεμένα Δεδομένα, είναι η δυναμική αξιοποίηση πληροφοριών από διάφορες πηγές στον παγκόσμιο ιστό. Η διαδικασία είναι πλήρως δυναμική διότι η εφαρμογή δεν γνωρίζει εκ των προτέρων πώς θα εκπληρώσει τον στόχο που έχει θέσει ο τελικός χρήστης. Αντιθέτως, η εφαρμογή θα πρέπει:

- να *συνδυάσει* διαφορετικές πηγές γνώσης (εδώ θα βοηθήσουν τα μονοσήμαντα URIs)
- να *συμπεράνει* πρόσθετη γνώση (μέσω λεξιλογίων και οντολογιών που έχουν εκφραστεί με τη βοήθεια των RDFS και OWL)
- και να *ανακαλύψει* νέες πηγές γνώσης (επισκεπτόμενη τα URIs που είναι και διευθύνσεις στον παγκόσμιο ιστό, σύμφωνα με το κίνημα των Συνδεδεμένων Δεδομένων).

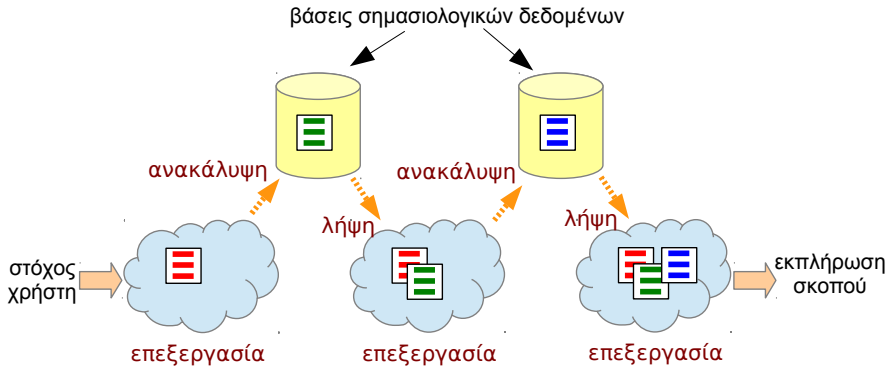
Η διαδικασία επαναλαμβάνεται ενσωματώνοντας τη νέα πληροφορία στην υπάρχουσα γνώση μέχρι την επίτευξη του στόχου.

Ο δυναμικός τρόπος επίλυσης που μόλις περιγράψαμε δεν είναι καινοτόμος. Στην πραγματικότητα αντιγράφει πιστά *τον συνήθη τρόπο αναζήτησης πληροφοριών από τον άνθρωπο*! Ο τρόπος αυτός αναζήτησης περιγράφεται ως «συλλογή άγριων καρπών» (berry-picking) στο κλασικό πλέον άρθρο του [1]: ο συλλέκτης πληροφορίας κινείται από πηγή σε πηγή πληροφορίας, επεξεργάζεται διαρκώς τη γνώση που αποκτά και αναπροσαρμόζει δυναμικά την πορεία του στις επόμενες πηγές γνώσης. Αν θεωρήσουμε τη λειτουργία μίας σημασιολογικής εφαρμογής ως παράδειγμα berry-picking, η διαδικασία απεικονίζεται στο σχήμα 6.1.

Στο προηγούμενο σχήμα, το κλειδί της επιτυχίας είναι η άντληση σημασιολογικής πληροφορίας και η ανακάλυψη νέων πηγών δεδομένων. Με τα θέματα αυτά θα ασχοληθούμε διεξοδικά στις επόμενες ενότητες, αφού πρώτα θυμηθούμε τις βασικές ιδιότητες του πρωτοκόλλου HTTP (Hypertext Transfer Protocol). Το πρωτόκολλο αυτό αποτελεί τη βάση κάθε εφαρμογής που δουλεύει με δεδομένα στον παγκόσμιο ιστό, συνεπώς και των εφαρμογών που χειρίζονται Συνδεδεμένα Δεδομένα.

6.2 HTTP: Βασικές γνώσεις

Το πρωτόκολλο HTTP (Hypertext Transfer Protocol), στην έκδοση 1.1, αποτελεί τον ακρογωνιαίο λίθο του παγκόσμιου ιστού. Στο αρχικό κείμενο του προτύπου



Σχήμα 6.1: Η σημασιολογική εφαρμογή ως διαδικασία «berry-picking»

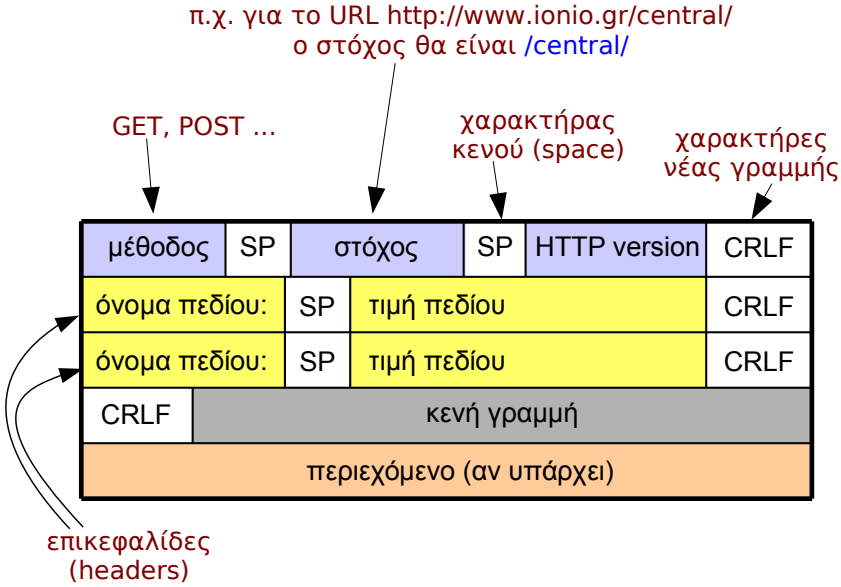
(RFC2616, 1999)[1] περιγράφεται η μορφή και αλληλουχία ανταλλαγής μηνυμάτων για τη λήψη και την ενημέρωση πληροφορίας στον παγκόσμιο ιστό. Το κείμενο είναι εκτενές· εμείς θα δούμε μόνο τα βασικά στοιχεία που χρειαζόμαστε για την υλοποίηση των σημασιολογικών εφαρμογών.

Σε μια τυπική «συνομιλία» για τη λήψη δεδομένων, η *εφαρμογή-πελάτης* (web client) στέλνει μία αίτηση στην *εφαρμογή-εξυπηρετητή* (web server). Η αίτηση αυτή έχει τη μορφή του σχήματος 6.2.

- Προηγείται μια γραμμή σε μορφή απλού κειμένου με τα βασικά στοιχεία της αίτησης: *ποιον στόχο* επισκεπτόμαστε (URL, χωρίς όμως το όνομα DNS) και *τι θέλουμε να κάνουμε* (μέθοδος). Η τυπική μέθοδος λήψης δεδομένων είναι η `GET`, ενώ το πρωτόκολλο προσδιορίζει και άλλες μεθόδους για ενημέρωση (εγγραφή και διαγραφή) δεδομένων.
- Ακολουθούν οι *επικεφαλίδες* (headers), και αυτές σε μορφή απλού κειμένου, μία ανά γραμμή. Θεωρήστε ότι είναι οι «παράμετροι» της αίτησης (μερικές υποχρεωτικές και άλλες προαιρετικές).
- Στη συνέχεια υπάρχει μία κενή γραμμή και *τα δεδομένα* (περιεχόμενο) που συνοδεύουν την αίτηση, αν αυτά υπάρχουν (π.χ. η αίτηση της μεθόδου GET δεν ακολουθείται από δεδομένα).

Στο παράδειγμα που ακολουθεί φαίνεται η τυπική αίτηση ενός φυλλομετρητή (web browser) στη διεύθυνση `http://sws.geonames.org/734890/`. Εκτός από την πρώτη γραμμή, οι υπόλοιπες είναι επικεφαλίδες (η αίτηση GET δεν έχει περιεχόμενο). Παρατηρήστε ότι το όνομα του web server (`sws.geonames.org`) τοποθετείται στις επικεφαλίδες κι όχι στην πρώτη γραμμή της αίτησης:

[1] Τυπικά, τον Ιούνιο του 2014 το RFC2616 αντικαταστάθηκε από τους διαδόχους του (RFC7230, RFC7231, RFC7232, RFC7233, RFC7234 και RFC7235). Στην πραγματικότητα όμως, πρόκειται για τον «εκμοντερνισμό» του αρχικού προτύπου, χωρίς ουσιαστική μεταβολή του.



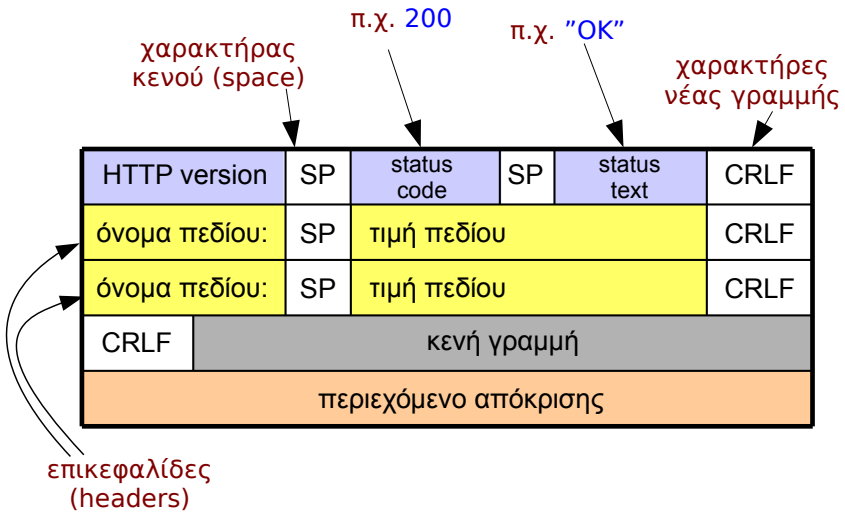
Σχήμα 6.2: Η αίτηση HTTP

```
GET /734890/ HTTP/1.1
Host: sws.geonames.org
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/41.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://dbpedia.org/page/Mount_Olympus
Cookie: JSESSIONID=D51DF676D492CA36382522248F8EDC93
Connection: keep-alive
```

Για τα Συνδεδεμένα Δεδομένα, ιδιαίτερη σημασία έχει η επικεφαλίδα `Accept`. Όπως θα δούμε αναλυτικά σε επόμενη ενότητα, η επικεφαλίδα αυτή μας επιτρέπει να διαπραγματευτούμε τη μορφή (αναπαράσταση) της πληροφορίας που ζητάμε. Η δυνατότητα αυτή είναι ιδιαίτερα σημαντική, καθώς επιτρέπει τη λήψη των ίδιων δεδομένων σε διαφορετική μορφή, ανάλογα με το ποιος κάνει την αίτηση (άνθρωπος ή σημασιολογική εφαρμογή).

Η γενική μορφή της απόκρισης HTTP φαίνεται στο σχήμα 6.3.

- Στην πρώτη γραμμή της απόκρισης επιστρέφεται ο *κωδικός αριθμός και το κείμενο της κατάστασης* (status code και status text): από την πληροφορία αυτή ενημερώνεται η εφαρμογή αν η αίτηση που έκανε ήταν επιτυχής ή όχι.



Σχήμα 6.3: Η απόκριση HTTP

- Ακολουθούν, όπως και στην περίπτωση της αίτησης, οι *επικεφαλίδες* της απόκρισης.
- Τέλος, ύστερα από μία κενή γραμμή, ακολουθεί το περιεχόμενο (ιστοσελίδα, εικόνα ή οποιοδήποτε άλλο ψηφιακό έγγραφο) της απόκρισης.

Οι σπουδαιότεροι κωδικοί κατάστασης παρατίθενται στον πίνακα 6.1.

Πίνακας 6.1: Οι πιο σημαντικοί κωδικοί κατάστασης

Status code	Status text
200	OK
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
400	Bad Request
403	Forbidden
404	Not Found
415	Unsupported Media Type
500	Internal Server Error
501	Not Implemented

Ως χρήστες είμαστε όλοι εξοικειωμένοι με τον κωδικό 404 (Not Found) και

ενδεχομένως και με τον 500 (Internal Server Error) που υποδηλώνουν ιστοσελίδες που δεν υπάρχουν ή εσφαλμένο προγραμματισμό στην πλευρά του εξυπηρετητή. Οι κωδικοί 200, 301, 302 και 304 είναι αυτοί που συνοδεύουν συνήθως τις αποκρίσεις στις αιτήσεις μας, αλλά, επειδή σηματοδοτούν επιτυχία, δεν τους βλέπουμε καν! Για τον Σημασιολογικό Ιστό, όπως θα δούμε αργότερα, ιδιαίτερη σημασία έχει ένας διαφορετικός κωδικός, ο οποίος χρησιμοποιείται σπάνια για ιστοσελίδες ή άλλα έγγραφα:

```
303 See Other
```

Ως παράδειγμα απόκρισης, παρατηρήστε την απάντηση στην αίτηση για το `http://www.bbc.co.uk/programmes/b00nj6cl.rdf`. Η αίτηση ήταν επιτυχής (`200 OK`). Προσέξτε την επικεφαλίδα `Content-Type: application/rdf+xml` που αναφέρει το είδος του επιστρεφόμενου περιεχομένου (δεδομένα RDF). Μετά τις επικεφαλίδες ακολουθούν τα δεδομένα (δεν φαίνονται στο παράδειγμα).

```
HTTP/1.1 200 OK
Server: Apache
Access-Control-Allow-Origin: *
Content-Encoding: gzip
Content-Type: application/rdf+xml
Cache-Control: private, max-age=0, no-store
Content-Length: 2998
Date: Sun, 04 Oct 2015 10:30:31 GMT
Connection: keep-alive
Vary: X-CDN,Accept-Encoding

...[υπόλοιπο περιεχόμενο]...
```

Σημείωση: Cross-Origin Resource Sharing (CORS).

Στο προηγούμενο παράδειγμα εμφανίζεται η επικεφαλίδα απόκρισης:

```
Access-Control-Allow-Origin: *
```

Περί τίνος πρόκειται; Οι φυλλομετρητές, για λόγους ασφαλείας, δεν επιτρέπουν στις εφαρμογές που εκτελούνται σε αυτούς να έχουν πρόσβαση σε δεδομένα με διαφορετική προέλευση από εκείνη από την οποία προήλθαν οι εφαρμογές. Εάν η σημασιολογική σας εφαρμογή τρέχει στον φυλλομετρητή, τότε θα πέσει θύμα αυτού του υποχρεωτικού περιορισμού της «ίδιας προέλευσης». Το πρότυπο Cross-Origin Resource Sharing (CORS) επιτρέπει τη χαλάρωση του περιορισμού:

μια σειρά από επικεφαλίδες στις αιτήσεις και αποκρίσεις του πρωτοκόλλου HTTP επιτρέπουν την ελεγχόμενη πρόσβαση σε δεδομένα από διαφορετικές πηγές. Όταν ένας εξυπηρετητής στέλνει την προηγούμενη επικεφαλίδα δηλώνει ότι τα δεδομένα του μπορούν να χρησιμοποιηθούν από οποιονδήποτε.

6.3 Διαπραγμάτευση μορφής επιστρεφόμενων δεδομένων

Η κλασική αρχιτεκτονική του παγκόσμιου ιστού χρησιμοποιεί προσπελάσεις σε διευθύνσεις *πόρων* (resources), όπως είναι οι ιστοσελίδες ή άλλα ψηφιακά έγγραφα, απ' όπου ο χρήστης μπορεί να λάβει μια *αναπαράσταση* (representation) του αντικειμένου που ζήτησε. Η αναπαράσταση αυτή μπορεί να είναι διαθέσιμη σε διάφορες μορφές, ανάλογα με τις ανάγκες του χρήστη. Το πρωτόκολλο HTTP παρέχει τη δυνατότητα να ζητήσετε έναν πόρο σε μια επιλεγμένη μορφή μέσω των επικεφαλίδων αίτησης και απόκρισης. Η λειτουργικότητα αυτή ονομάζεται *διαπραγμάτευση (επιστρεφόμενου) περιεχομένου* (content negotiation) και έχει ιδιαίτερη σημασία για τον Σημασιολογικό Ιστό, όπου

> «τα δεδομένα και η σημασιολογία τους είναι το ίδιο διαθέσιμα για τον άνθρωπο αλλά και τη μηχανή.»

Θα πρέπει εδώ να τονίσουμε ότι ο μηχανισμός διαπραγμάτευσης είναι πολύ ελαστικός: κανείς δεν υποχρεώνει τον εξυπηρετητή να ανταποκριθεί στο αίτημά σας για διαφορετικό είδος περιεχομένων. Συνήθως υπάρχει μία εξ ορισμού μορφή (π.χ. απλή HTML), η οποία επιστρέφεται όταν ο εξυπηρετητής δεν μπορεί να εξυπηρετήσει (ή αποφασίζει να αγνοήσει) το αίτημά σας!

Ας δούμε τη διαπραγμάτευση περιεχομένου μέσω ενός παραδείγματος. Αν επισκεφτούμε τη διεύθυνση `http://xmlns.com/foaf/spec/` (το έγγραφο περιγράφει το λεξιλόγιο FOAF) χωρίς να προσδιορίσουμε επιθυμητή μορφή περιεχομένου, η αίτηση μέσω του πρωτοκόλλου HTTP θα είναι παρόμοια με την ακόλουθη:

```
GET /foaf/spec/ HTTP/1.1
User-Agent: curl/7.35.0
Host: xmlns.com
Accept: */*
```

Η απόκριση περιέχει την περιγραφή του λεξιλογίου FOAF σε μορφή ιστοσελίδας:

```
HTTP/1.1 200 OK
Date: Sun, 04 Oct 2015 12:22:08 GMT
```

```
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Tue, 14 Jan 2014 19:54:30 GMT
ETag: "3672a-4eff38f801066"
Accept-Ranges: bytes
Content-Length: 223018
Vary: Accept-Encoding
Content-Type: text/html

<!DOCTYPE html>
...[υπόλοιπο περιεχόμενο ιστοσελίδας]...
```

Αν όμως δηλώσετε ότι προτιμάτε δεδομένα RDF (σε μορφή XML) μέσω της πρόσθετης επικεφαλίδας `Accept: application/rdf+xml`

```
GET /foaf/spec/ HTTP/1.1
User-Agent: curl/7.35.0
Host: xmlns.com
Accept: application/rdf+xml
```

τότε θα λάβετε μια τελείως διαφορετική απόκριση:

```
HTTP/1.1 200 OK
Date: Sun, 04 Oct 2015 12:20:33 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept
Last-Modified: Tue, 14 Jan 2014 19:16:42 GMT
ETag: "acb1-4eff30855fec5"
Accept-Ranges: bytes
Content-Length: 44209
Content-Type: application/rdf+xml

<!-- This is the FOAF formal vocabulary description,
expressed using W3C RDFS and OWL markup -->
...[υπόλοιπο περιεχόμενο RDF σε σύνταξη XML]...
```

Παρατηρήστε την επικεφαλίδα `Vary: Accept`, η οποία απευθύνεται πρωτίστως σε ενδιάμεσους μηχανισμούς προσωρινής αποθήκευσης (HTTP caching) και προειδοποιεί ότι η απόκριση δεν εξαρτάται μόνο από τη διεύθυνση URL αλλά και από την επικεφαλίδα `Accept` της αίτησης. Έτσι, αν ζητηθεί ξανά η ίδια διεύθυνση, θα πρέπει να ελεγχθεί και η επικεφαλίδα `Accept`, πριν επιστραφεί η προσωρινά αποθηκευμένη απόκριση. Το θέμα δεν αφορά άμεσα τη σημασιολογική εφαρμογή σας, αν και μπορείτε να τη χρησιμοποιήσετε ως ένδειξη για την ύπαρξη περιεχομένου σε διάφορες μορφές αναπαράστασης.

6.4 Ελέγξτε τις γνώσεις σας

1. Ποιο το κύριο χαρακτηριστικό μιας σημασιολογικής εφαρμογής;
    i. Μπορεί να εκμεταλλευτεί σημασιολογικά δεδομένα.
    ii. Έχει προγραμματιστεί να αναγνωρίζει αποκλειστικά δεδομένα RDF.
    iii. Απαιτεί τουλάχιστον δύο διαφορετικές πηγές σημασιολογικών δεδομένων.
2. Η διαδικασία berry-picking, όσον αφορά τη συλλογή πληροφορίας
    i. περιγράφει τη σταδιακή επίσκεψη σε διάφορες πηγές γνώσης.
    ii. περιγράφει τη σταδιακή επίσκεψη σε διάφορες πηγές γνώσης με βάση ένα προκαθορισμένο σχέδιο δράσης.
    iii. περιγράφει τη σταδιακή επίσκεψη σε διάφορες πηγές γνώσης και τη δυναμική αναπροσαρμογή των επόμενων ενεργειών αναζήτησης με βάση τη νεοαποκτηθείσα γνώση.
3. Μια τυπική ανταλλαγή δεδομένων σύμφωνα με το πρωτόκολλο HTTP
    i. αποτελείται από την αίτηση προς την εφαρμογή-πελάτη (web client) και την απόκριση προς την εφαρμογή-εξυπηρετητή (web server).
    ii. αποτελείται από την αίτηση προς την εφαρμογή-εξυπηρετητή και την απόκριση προς την εφαρμογή-πελάτη.
    iii. επιτρέπει την απευθείας αποστολή δεδομένων προς την εφαρμογή-πελάτη, χωρίς να προηγηθεί αίτηση λήψης.
4. Οι επικεφαλίδες μιας αίτησης ή απόκρισης HTTP
    i. προσδιορίζουν τη διεύθυνση του πόρου που επισκεπτόμαστε.
    ii. προσδιορίζουν τις παραμέτρους της αίτησης ή της απόκρισης.
    iii. προσδιορίζουν την έκδοση του πρωτοκόλλου HTTP.
5. Δεδομένα μπορούν να μεταφερθούν
    i. μόνο στην αίτηση HTTP.
    ii. μόνο στην απόκριση HTTP.
    iii. τόσο στην αίτηση όσο και στην απόκριση HTTP.
6. Ο μηχανισμός διαπραγμάτευσης περιεχομένου
    i. επιτρέπει τη δήλωση προτίμησης για τη λήψη συγκεκριμένης μορφής περιεχομένου από την εφαρμογή-πελάτη.
    ii. επιτρέπει την επιλογή συγκεκριμένης μορφής περιεχομένου από την εφαρμογή-πελάτη.
    iii. επιτρέπει την ανακάλυψη συγκεκριμένης μορφής περιεχομένου από την εφαρμογή-πελάτη.

6.5 Έμμεση προσπέλαση (dereferencing)

Σύμφωνα με τις αρχές των Συνδεδεμένων Δεδομένων, τα αναγνωριστικά ονόματα URIs που αντιπροσωπεύουν κάθε οντότητα στον Σημασιολογικό Ιστό, είναι «επισκέψιμα»:

> Κανόνας 2: «*Χρησιμοποιήστε HTTP URIs, έτσι ώστε να αποτελούν επισκέψιμες διευθύνσεις στον παγκόσμιο ιστό*».

Η τήρηση του προηγούμενου κανόνα επιτρέπει πρακτικά σε μία σημασιολογική εφαρμογή να μάθει περισσότερα πράγματα για μία οντότητα επισκεπτόμενη το URI της. Ο μηχανισμός επίσκεψης και λήψης πληροφορίας μέσω του αντίστοιχου URI αποτελεί τη βάση του παγκόσμιου ιστού και ονομάζεται «dereferencing», σε αντιστοιχία με την έμμεση προσπέλαση μεταβλητών μέσω δεικτών ή αναφορών στις γλώσσες προγραμματισμού. Θα πρέπει όμως να θυμόμαστε πάντοτε μια λεπτή διαφορά:

- Στον παγκόσμιο ιστό το URI αντιπροσωπεύει έναν *πληροφοριακό πόρο* (π.χ., μια ιστοσελίδα) και είναι αυτός ακριβώς ο πόρος που θα λάβουμε, σε κάποια ψηφιακή μορφή, εάν επισκεφτούμε το URI.
- Με τα Συνδεδεμένα Δεδομένα, το URI μπορεί να αντιπροσωπεύει οποιαδήποτε *οντότητα*! Αν η οντότητα δεν είναι πληροφοριακός πόρος (δεν είναι, δηλαδή, έγγραφο στον παγκόσμιο ιστό), προφανώς αυτό που θα λάβετε ως απάντηση δεν είναι η ίδια η οντότητα, αλλά *ένα έγγραφο με πληροφορία σχετική με την οντότητα* που αντιπροσωπεύει το URI.

Η λειτουργία της επίσκεψης σε ένα URI οντότητας του Σημασιολογικού Ιστού είναι στενά συνδεδεμένη με τη μορφή του χώρου ονομάτων, στον οποίο ανήκει το URI. Υπενθυμίζεται ότι ο χώρος ονομάτων είναι μία ομάδα URIs με κοινό πρόθεμα. Οι χώροι ονομάτων που θα συναντήσουμε στα Συνδεδεμένα Δεδομένα έχουν δύο μορφές:

- Ο χώρος ονομάτων καταλήγει στον χαρακτήρα `#` (hash), ο οποίος εισάγει το *απόσπασμα* (fragment) ενός μεγαλύτερου εγγράφου. Για παράδειγμα, το URI:

  `http://www.w3.org/2000/01/rdf-schema#label`

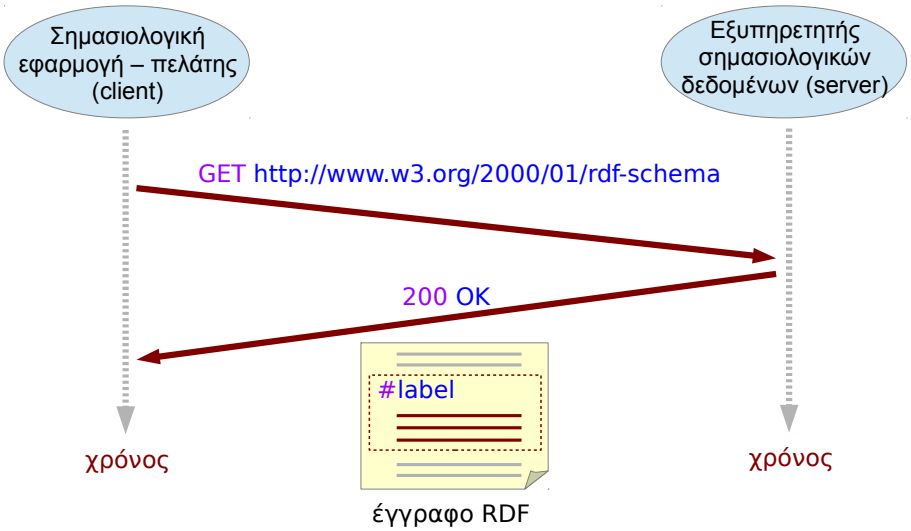
  μπορεί να θεωρηθεί ότι αντιπροσωπεύει την οντότητα `label`, η οποία ανήκει σε ένα ευρύτερο λεξιλόγιο (εδώ, το λεξιλόγιο RDFS) στον χώρο ονομάτων `http://www.w3.org/2000/01/rdf-schema#`.

- Ο χώρος ονομάτων καταλήγει στον χαρακτήρα `/` (slash). Για παράδειγμα, το URI

  `http://dbpedia.org/resource/Lodovico_Giustini`

αναγνωρίζει την οντότητα `Lodovico_Giustini` μέσα στον χώρο ονομάτων `http://dbpedia.org/resource/`.

Στο σχήμα 6.4 βλέπουμε την προσπέλαση ενός *hash URI* μιας οντότητας που δεν αποτελεί πληροφοριακό πόρο (`http://www.w3.org/2000/01/rdf-schema#label`).



Σχήμα 6.4: Προσπέλαση οντότητας με hash URI

Παρατηρήστε ότι:

1. Το πρωτόκολλο HTTP επιτάσσει *την αφαίρεση του αποσπάσματος* πριν από την αποστολή της αίτησης! Συνεπώς, η σημασιολογική εφαρμογή *δεν επισκέπτεται το URI που ζητήθηκε, αλλά ένα έγγραφο μέσα στο οποίο περιλαμβάνεται αυτό που ζητήθηκε*.

2. Σύμφωνα πάντα με το πρωτόκολλο HTTP, η αίτηση έγινε για ένα έγγραφο και αυτό ακριβώς το έγγραφο στέλνεται ως απάντηση. Η αίτηση ήταν επιτυχής και ο κωδικός κατάστασης `200 OK`.

3. Το πρωτόκολλο HTTP *δεν καθορίζει το πώς θα χειριστεί το απόσπασμα ο λήπτης*. Στην περίπτωση των σημασιολογικών εφαρμογών που χειρίζονται δεδομένα RDF, μπορούν, για παράδειγμα, να αναζητηθούν όλες οι τριάδες που έχουν το πλήρες URI (μαζί με το απόσπασμα) ως υποκείμενο ή αντικείμενο.

Το πλεονέκτημα χρήσης hash URIs έγκειται στην απλότητα της υλοποίησης. Συχνά το έγγραφο που συγκεντρώνει τις τριάδες για όλα τα URIs του χώρου ονομάτων είναι μια στατική ιστοσελίδα. Έτσι δεν απαιτούνται ειδικές ρυθμίσεις

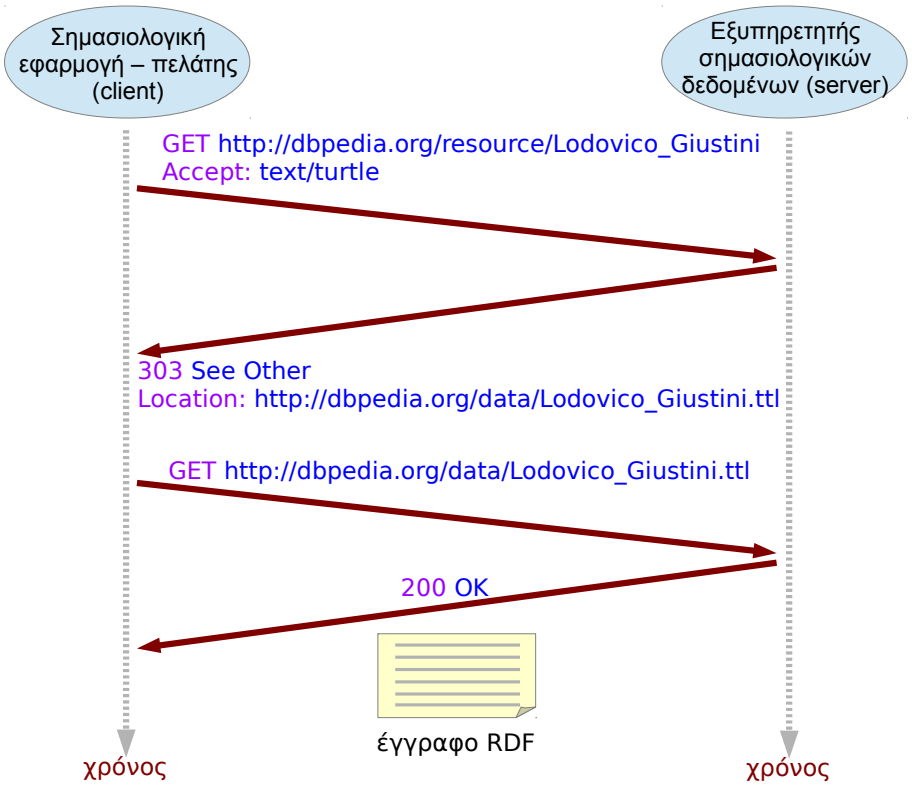
στην πλευρά του εξυπηρετητή (web server) για την εξυπηρέτηση του εγγράφου αυτού. Μέσω των hash URIs περιγράφονται συνήθως τα κατηγορήματα και οι κλάσεις ενός λεξιλογίου ή μίας οντολογίας.

Από την άλλη πλευρά, η μέθοδος του «ενός εγγράφου» είναι ακατάλληλη όταν τα URIs του χώρου ονομάτων είναι χιλιάδες ή εκατομμύρια: θα έπρεπε να λάβετε ένα τεράστιο έγγραφο, να εξαγάγετε μια μικροσκοπική ποσότητα πληροφορίας και να απορρίψετε τα υπόλοιπα, προτού μάθετε αυτό που θέλετε!

Στην περίπτωση των slash URIs, τα πράγματα είναι διαφορετικά. Δείτε για παράδειγμα στο σχήμα 6.5 τι συμβαίνει κατά την επίσκεψη του URI

```
http://dbpedia.org/resource/Lodovico_Giustini
```

(το URI αυτό αντιπροσωπεύει έναν μη πληροφοριακό πόρο).



Σχήμα 6.5: Προσπέλαση οντότητας με slash URI

Σε αντίθεση με την προηγούμενη περίπτωση του hash URI, εδώ η επίσκεψη γίνεται σε *δύο στάδια*:

1. Στην πρώτη αίτηση, η οποία ζητάει δεδομένα σε μορφή Turtle, η απόκριση

δεν μεταφέρει δεδομένα. Αντιθέτως, ο επιστρεφόμενος κωδικός κατάστασης είναι

`303 See Other`

και εμφανίζεται η επικεφαλίδα

`Location: http://dbpedia.org/data/Lodovico_Giustini.ttl`

η οποία παρέχει την *πραγματική* διεύθυνση του εγγράφου, όπου βρίσκεται η πληροφορία που ζητήθηκε.

2. Στη συνέχεια, η δεύτερη αίτηση στη διεύθυνση που έχει υποδειχτεί φέρνει τα σχετικά με το αρχικό URI δεδομένα RDF.

Η υλοποίηση ενός χώρου ονομάτων με slash URIs απαιτεί έναν πολυπλοκότερο μηχανισμό στην πλευρά του εξυπηρετητή. Από την άλλη πλευρά, η μέθοδος αυτή είναι πολύ ευέλικτη και επιτρέπει κάθε πιθανό τρόπο επιλογής δεδομένων προς αποστολή: φανταστείτε ότι τα δεδομένα δεν είναι ανάγκη να προέρχονται από στατικά έγγραφα· συνήθως εξάγονται δυναμικά από βάσεις δεδομένων μέσω ερωτημάτων SQL ή SPARQL.

Στο σημείο αυτό ολοκληρώνεται η παρουσίαση των δύο βασικών μεθόδων προσπέλασης των σημασιολογικών δεδομένων μέσω της επίσκεψης των αντίστοιχων URIs. Για μια πλήρη ανάπτυξη των προτεινόμενων μεθόδων προσπέλασης δείτε στο [2].

Σημείωση: Οι άγγελοι, η καρφίτσα και το httpRange-14.

Κάτω από την κρυπτική ονομασία «httpRange-14» βρίσκεται μία από τις πιο διάσημες υποθέσεις που απασχόλησε για αρκετά χρόνια τις επιτροπές του W3C, μέχρι την οριστική λύση της το 2005. Το ερώτημα ήταν:

> «Εφόσον το HTTP προβλέπει την επιστροφή του κωδικού `200 OK` εάν βρεθεί ο πληροφοριακός πόρος (το έγγραφο) που ζητήσατε, είναι σωστό να επιστρέφεται το ίδιο για ένα URI που δεν αντιπροσωπεύει πληροφοριακό πόρο (π.χ. το URI ενός ανθρώπου, ενός συναισθήματος ή μιας γενικότερης αφηρημένης έννοιας);»

Η λύση που προτάθηκε αποτελεί τη βάση για τη λειτουργία των slash URIs που είδαμε προηγουμένως:

- Αν σε αίτηση GET ληφθεί `200 OK`, τότε το URI που ζητήθηκε αντιπροσωπεύει έναν πληροφοριακό πόρο.

- Αν ληφθεί `303 See Other`, τότε το URI αντιπροσωπεύει οποιονδήποτε πόρο (ακόμα και μη πληροφοριακό).
- Τέλος, αν ληφθεί `4xx` (σφάλμα επίσκεψης), δεν μπορείτε να βγάλετε κανένα συμπέρασμα για τη φύση της οντότητας που αντιπροσωπεύει το URI.

Σήμερα φαίνεται απίστευτος ο χρόνος και ο κόπος ενασχόλησης[2] με ένα θέμα, το οποίο έχει τόση πρακτική σημασία όσο και η ατέρμονη διαφωνία των θεολόγων του μεσαίωνα σχετικά με το πόσοι άγγελοι χωράνε στη μύτη μιας καρφίτσας! Υποθέστε, όμως, ότι η σημασιολογική εφαρμογή σας λαμβάνει πίσω τα δεδομένα RDF που ζήτησε, με τον κωδικό `200 OK` αντί του ορθού `303 See Other`. Θα συμφωνήσετε μαζί μας ότι δεν έχετε την πολυτέλεια να τα απορρίψετε μόνο και μόνο γιατί δεν «σερβίρονται» με τον σωστό τρόπο! Αντιθέτως η εφαρμογή σας θα πρέπει να κινηθεί ευριστικά, προσπαθώντας να διαπιστώσει από το περιεχόμενο αν τα δεδομένα σάς είναι χρήσιμα ή όχι.

6.6 Ποια είναι η «σχετική με ένα URI» πληροφορία;

Στην προηγούμενη ενότητα αναλύθηκε λεπτομερώς ο μηχανισμός dereferencing, μέσω του οποίου μία σημασιολογική εφαρμογή μπορεί να αποκτήσει νέα πληροφορία (τριάδες RDF) σχετική με μία οντότητα, επισκεπτόμενη το URI που αναφέρεται στην οντότητα αυτή. Από τι απαρτίζεται όμως η «σχετική πληροφορία»;

Τα πρότυπα του Σημασιολογικού Ιστού αφήνουν αναπάντητο το προηγούμενο ερώτημα. Είναι στη διακριτική ευχέρεια του παρόχου των σημασιολογικών δεδομένων να αποφασίσει ποια είναι η σχετική πληροφορία. Ιδανικά, όταν προσπελαύνουμε ένα URI, θα θέλαμε να πάρουμε ως απάντηση όλη την πληροφορία που αφορά την αντιπροσωπευόμενη οντότητα. Αν δούμε ένα σύνολο δεδομένων RDF ως έναν γράφο, η πληροφορία που ζητάμε είναι ένας υπογράφος του συνολικού. Ο υπογράφος θα πρέπει να είναι περιορισμένου εύρους (μικρό, δηλαδή, υποσύνολο του συνολικού γράφου τριάδων), έτσι ώστε να μπορεί να αναζητηθεί και αποσταλεί με αποδοτικό τρόπο. Ταυτόχρονα όμως θα πρέπει να περιγράφει ικανοποιητικά τη ζητούμενη οντότητα.

Οι συνήθεις μορφές «σχετικής πληροφορίας» ονομάζονται *συνοπτικές περιγραφές περιορισμένου εύρους* (concise bounded descriptions) και περιλαμβάνουν:

- Το σύνολο των τριάδων με τη ζητούμενη οντότητα σε θέση υποκειμένου.
- Επιπλέον, όπου το αντικείμενο των προηγούμενων τριάδων είναι ανώνυμος κόμβος, προστίθενται αναδρομικά και όλες οι τριάδες που έχουν τον ανώνυμο αυτόν κόμβο ως υποκείμενο.

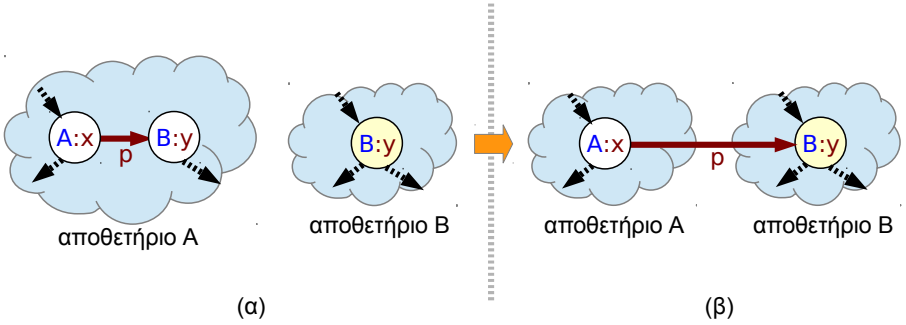
[2] Μπορείτε να δείτε τα πλήρη πρακτικά στο http://www.w3.org/2001/tag/issues.html#httpRange-14.

- Προαιρετικά, τα δύο προηγούμενα επαναλαμβάνονται με τη ζητούμενη οντότητα σε θέση αντικειμένου και οι τριάδες που προκύπτουν προστίθενται στην επιστρεφόμενη περιγραφή.

Έχοντας στη διάθεσή της την περιγραφή, μία σημασιολογική εφαρμογή μπορεί να μάθει για όλες τις ιδιότητες κάποιας οντότητας, καθώς επίσης και με ποιες άλλες οντότητες σχετίζεται. Το τελευταίο είναι ζωτικής σημασίας για την ανάπτυξη των Συνδεδεμένων Δεδομένων και θα αναλυθεί στη συνέχεια.

6.7 Το νέφος των Ανοικτών Συνδεδεμένων Δεδομένων

Ας υποθέσουμε ότι υπάρχουν στον παγκόσμιο ιστό δύο αποθετήρια (repositories) σημασιολογικών δεδομένων, τα αποθετήρια `A` και `B` όπως φαίνεται στο σχήμα 6.6(α).



Σχήμα 6.6: Διασύνδεση αποθετηρίων RDF

Τα αποθετήρια αυτά παρέχουν τα δεδομένα τους σε μορφή γράφου τριάδων RDF. Στο αποθετήριο A υπάρχει μέσα στον γράφο των δεδομένων η εξής τριάδα:

```
A:x p B:y .
```

όπου το υποκείμενο `A:x` ανήκει στο αποθετήριο A (έχει δημιουργηθεί και ανήκει στον χώρο ονομάτων που διαχειρίζεται το αποθετήριο A) και το αντικείμενο `B:y` ανήκει στο αποθετήριο `B` (με άλλα λόγια, το αποθετήριο `A` έχει «υιοθετήσει» το URI του αποθετηρίου `B`).

Μια σημασιολογική εφαρμογή που επισκέπτεται το αποθετήριο A, ζητώντας πληροφορία για την οντότητα `A:x`, μπορεί μέσω της προηγούμενης τριάδας (`A:x p B:y .`) *να ανακαλύψει* το αποθετήριο B και *να μετακινηθεί* εκεί, επισκεπτόμενη

την οντότητα `B:y`. Στην ουσία, η τριάδα λειτουργεί ως *σύνδεσμος μεταξύ των αποθετηρίων A και B* (σχήμα 6.6(β)).

Η νοητή διασύνδεση των δύο αποθετηρίων μέσω του κατηγορήματος p επιτρέπει την ανακάλυψη όχι μόνο νέων ενδιαφερουσών οντοτήτων αλλά και νέων αποθετηρίων για τα οποία η σημασιολογική εφαρμογή μας ενδεχομένως δεν γνωρίζει τίποτα όταν ξεκινά να εκπληρώσει τον στόχο της. Η διασύνδεση των αποθετηρίων είναι ουσιαστική προϋπόθεση για την επιτυχία του berry-picking. Οποιοδήποτε κατηγόρημα p μπορεί να χρησιμοποιηθεί ως σύνδεσμος, αν και είναι σύνηθες να συναντάμε στον ρόλο αυτόν το `owl:sameAs` ως δηλωτικό ισοδυναμίας μεταξύ οντοτήτων.

Η αυξημένη διασύνδεση μεταξύ των διαφόρων πηγών (αποθετηρίων) RDF έχει μεγάλη αξία για την ανάπτυξη του Σημασιολογικού Ιστού, σύμφωνα με το κίνημα των Συνδεδεμένων Δεδομένων. Αρκεί όμως αυτό;

- Θα πρέπει να υπάρξει και ο αντίστοιχος μεγάλος αριθμός αποθετηρίων RDF, τα οποία *θα παρέχουν στον παγκόσμιο ιστό* τα σημασιολογικά δεδομένα τους προς τις εφαρμογές.
- Τα δεδομένα θα πρέπει να παρέχονται με *άδειες ανοικτής χρήσης*, οι οποίες θα επιτρέπουν την ελεύθερη ενσωμάτωση των δεδομένων σε παράγωγα έργα.

Αναγνωρίζοντας τη σημασία της «ανοικτότητας» για την ανάπτυξη των Συνδεδεμένων Δεδομένων, ο Tim Berners-Lee πρόσθεσε το 2010 στο ιστορικό κείμενο των τεσσάρων κανόνων τις «Εντολές των 5 άστρων»:

★☆☆☆☆ «Διαθέστε τα δεδομένα σας στο Web, σε οποιαδήποτε μορφή, με ανοικτή άδεια.»

★★☆☆☆ «Διαθέστε τα δεδομένα σας σε δομημένη μορφή, έστω και με κλειστό πρότυπο.»

★★★☆☆ «Χρησιμοποιήστε ανοικτά πρότυπα.»

★★★★☆ «Χρησιμοποιήστε HTTP URIs, για να μπορούν οι άλλοι να αναφερθούν στα δεδομένα σας.»

★★★★★ «Διασυνδέστε τα δεδομένα σας με άλλα δεδομένα τρίτων.»

Τα 5 άστρα απευθύνονται σε ένα λιγότερο «τεχνικό» κοινό αλλά δεν θα μπορούσαν να τονίσουν με μεγαλύτερη έμφαση την αξία της «ανοικτότητας», η οποία εμφανίζεται εδώ εξίσου σημαντική με τη «διασυνδεσιμότητα»! Πλέον αναφερόμαστε σε *Ανοικτά Συνδεδεμένα Δεδομένα* (Linked Open Data – LOD) αντί απλώς για Συνδεδεμένα Δεδομένα. Οι παραινέσεις απευθύνονται στους παραγωγούς των Ανοικτών Συνδεδεμένων Δεδομένων, αλλά είναι το ίδιο σημαντικές και για τους δημιουργούς των σημασιολογικών εφαρμογών: *αποκομίστε πληροφορία με ευέλικτο τρόπο από οποιαδήποτε μορφή σημασιολογικών δεδομένων*.

Όσο τα ανοικτά αποθετήρια RDF αυξάνονται στον παγκόσμιο ιστό και όσο τηρούνται οι καλές πρακτικές αποθήκευσης σε κάποιο ανοικτό πρότυπο και παροχής συνδέσμων και αδειών ανοικτής χρήσης, τόσο ο χώρος των Ανοικτών Συνδεδεμένων Δεδομένων μετασχηματίζεται σε ένα κατανεμημένο, και ταυτόχρονα αλληλοσυνδεόμενο, «νέφος» (Linked Open Data cloud – LOD cloud [3]). Στο νέφος αυτό μπορούν να κινηθούν οι σημασιολογικές εφαρμογές, αναζητώντας πληροφορία RDF για να εκπληρώσουν τον στόχο τους. Στο http://lod-cloud.net/ μπορείτε να δείτε εντυπωσιακά διαγράμματα με την πρόσφατη κατάσταση του νέφους των Ανοικτών Συνδεδεμένων Δεδομένων, να επισκεφτείτε τις αντίστοιχες πηγές RDF και να διαβάσετε στατιστικά μεγέθη για το νέφος.

Σημείωση: Ανάστροφοι σύνδεσμοι (backlinks).

Στο προηγούμενο σχήμα είδαμε πώς η τριάδα `A:x p B:y` στο αποθετήριο σημασιολογικών δεδομένων `A` αποτελεί εννοιολογικά έναν σύνδεσμο μεταξύ των αποθετηρίων `A` και `B`. Ο σύνδεσμος αυτός, μέσω του κατηγορήματος `p`, συνενώνει την οντότητα `A:x` (που «ανήκει» στο αποθετήριο `A`) με την οντότητα `B:y` (η τελευταία «ανήκει» στο αποθετήριο `B`).

Ο ιδεατός σύνδεσμος επιτρέπει τη μετακίνηση μιας σημασιολογικής εφαρμογής από το αποθετήριο `A` στο αποθετήριο `B`. *Δεν ισχύει όμως και το αντίστροφο*. Ακόμα κι αν η τριάδα του αποθετηρίου `A` ήταν

```
B:y p A:x .
```

τίποτα δεν μπορεί να υποδείξει σε μια εφαρμογή που βρίσκεται στο αποθετήριο `B` να μετακινηθεί προς το αποθετήριο `A`. Αυτό είναι δυνατό μόνο εάν υπάρχει κάποια από τις δύο προηγούμενες τριάδες *και* στο αποθετήριο `B`. Αν υπάρχει, ο ανάστροφος σύνδεσμος που οδηγεί πίσω σε εκείνον που έκανε την πρώτη σύνδεση ονομάζεται ανεπίσημα *backlink* (δανειζόμαστε αυτή την ορολογία από τα δίκτυα κοινωνικής δικτύωσης).

Δεν υπάρχει κάποιο πρότυπο ή κάποια καλή πρακτική για τη δημιουργία και την ενημέρωση των ανάστροφων συνδέσμων, παρά την αδιαμφισβήτητη συμβολή τους στην αύξηση της διασυνδεσιμότητας μεταξύ των διάφορων πηγών σημασιολογικών δεδομένων. Τα διάσημα αποθετήρια δεδομένων RDF ανταλλάσσουν συχνά μεταξύ τους λίστες με ισοδύναμα URIs που αναφέρονται στην ίδια οντότητα. Στη συνέχεια, οι ισοδυναμίες αυτές εμφανίζονται και στα δύο αποθετήρια ως σύνδεσμοι `owl:sameAs`, επιτρέποντας την αμφίδρομη μετακίνηση μεταξύ των αποθετηρίων.

Από την άλλη πλευρά, είναι μάλλον απίθανο τα μεγάλα αποθετήρια (όπως η DBpedia), των οποίων τα URIs χρησιμοποιούνται σε πολλά άλλα μικρότερα αποθετήρια

RDF, να δημιουργήσουν και να ενημερώνουν backlinks προς κάθε μικρότερη πηγή που χρησιμοποιεί τα URIs τους!

6.8 Εξερευνώντας ένα αποθετήριο RDF

Η λειτουργία μιας σημασιολογικής εφαρμογής, όπως έχει παρουσιαστεί στις προηγούμενες ενότητες, προϋποθέτει την ύπαρξη της «ελάχιστης αρχικής γνώσης» κατά την έναρξη λειτουργίας: η σημασιολογική εφαρμογή θα πρέπει να γνωρίζει *τουλάχιστον ένα URI ή ένα τελικό σημείο εξυπηρέτησης SPARQL* για να ξεκινήσει την αναζήτηση νέας γνώσης.

Αυτή η ελάχιστη αρχική γνώση προκύπτει συνήθως με τη βοήθεια του (ανθρώπου) χρήστη, κατά τη διατύπωση του ερωτήματός του. Αυτό συχνά προϋποθέτει την εξερεύνηση των χαρακτηριστικών και των υπηρεσιών που παρέχει κάποιο νέο, άγνωστο αποθετήριο RDF. Αν και η εξερεύνηση διενεργείται πρωτίστως από τον άνθρωπο, υπάρχει η δυνατότητα να επωφεληθούν από τη διαδικασία αυτή και οι σημασιολογικές εφαρμογές, όπως θα φανεί στη συνέχεια.

Το λεξιλόγιο VoID (Vocabulary of Interlinked Datasets) επιτρέπει την άντληση μεταδεδομένων σχετικών με μία πηγή δεδομένων RDF. Δεν θα παραθέσουμε αναλυτικά τις κλάσεις και τα κατηγορήματα του λεξιλογίου εδώ· αντί γι' αυτό θα δούμε παραδείγματα από το έγγραφο RDF (σε μορφή Turtle) που επιστρέφεται όταν επισκεφτούμε τη διεύθυνση http://data.nobelprize.org/dataset.

Η οντότητα με URI `http://data.nobelprize.org/dataset` είναι στιγμιότυπο της κλάσης `void:Dataset`:

```
@prefix void:    <http://rdfs.org/ns/void#> .
<http://data.nobelprize.org/dataset>
      a void:Dataset .
```

Ο χώρος ονομάτων (namespace) των οντοτήτων είναι:

```
<http://data.nobelprize.org/dataset>
      void:uriSpace "http://data.nobelprize.org/resource/" .
```

Η πληροφορία RDF παρέχεται σε διάφορες μορφές:

```
<http://data.nobelprize.org/dataset>
      void:feature <http://www.w3.org/ns/formats/Turtle> ,
                   <http://www.w3.org/ns/formats/RDF_XML> ,
                   <http://www.w3.org/ns/formats/N3> .
```

Παρέχεται τελικό σημείο εξυπηρέτησης SPARQL:

```
<http://data.nobelprize.org/dataset>
      void:sparqlEndpoint <http://data.nobelprize.org/sparql> .
```

Χρησιμοποιούνται τα εξής λεξιλόγια:

```
<http://data.nobelprize.org/dataset>
      void:vocabulary rdf: , rdfs: , foaf: ,
                      <http://www.w3.org/2004/02/skos/core#> ,
                      <http://purl.org/dc/terms/> ,
                      <http://dbpedia.org/ontology/> ,
                      <http://data.nobelprize.org/terms/> ,
                      <http://www.w3.org/2002/07/owl#> ,
                      <http://dbpedia.org/property/> .
```

Εκτός από τα προηγούμενα, μέσω του λεξιλογίου VoID περιγράφονται οι χρησιμοποιούμενες κλάσεις (ακολουθεί απόσπασμα μόνο από τη συνολική λίστα):

```
<http://data.nobelprize.org/dataset>
      void:classPartition
              [ void:class
↪   <http://data.nobelprize.org/terms/LaureateAward>
              ] ;
      void:classPartition
              [ void:class <http://data.nobelprize.org/terms/NobelPrize>
              ] ;
      void:classPartition
              [ void:class <http://data.nobelprize.org/terms/PrizeFile>
              ] ;
      void:classPartition
              [ void:class <http://dbpedia.org/ontology/Country>
              ] ;
      void:classPartition
              [ void:class <http://dbpedia.org/ontology/Award>
              ] ;
      void:classPartition
              [ void:class <http://data.nobelprize.org/terms/Laureate>
              ] ;
```

και ιδιότητες (κατηγορήματα, παρατίθενται πάλι ορισμένα μόνον από αυτά):

```
<http://data.nobelprize.org/dataset>
      void:propertyPartition
              [ void:property <http://purl.org/dc/terms/subject>
              ] ;
      void:propertyPartition
              [ void:property
↪  <http://www.w3.org/2004/02/skos/core#hiddenLabel>
              ] ;
      void:propertyPartition
              [ void:property <http://www.w3.org/2002/07/owl#sameAs>
              ] ;
      void:propertyPartition
              [ void:property <http://purl.org/dc/terms/hasPart>
              ] ;
      void:propertyPartition
              [ void:property rdfs:label
              ] ;
      void:propertyPartition
              [ void:property rdfs:seeAlso
              ] ;
```

Θυμάστε τη συντομογραφία των ανώνυμων κόμβων στην Turtle; οι προηγούμενες δηλώσεις λένε ότι το σύνολο δεδομένων χωρίζεται:

- σε υποσύνολα οντοτήτων με κοινή κλάση (class partitions) και
- σε υποσύνολα τριάδων με κοινό κατηγόρημα (property partitions).

Τέλος, το λεξιλόγιο VoID περιγράφει ορισμένες οντότητες-κατηγορίες, μέσω της προσπέλασης των οποίων μπορούμε να φτάσουμε σε κάθε άλλη οντότητα σε λίγα βήματα (και εδώ δίνουμε μερικές μόνο από τις οντότητες αυτές):

```
<http://data.nobelprize.org/dataset>
      void:rootResource
        <http://data.nobelprize.org/all/university> ,
        <http://data.nobelprize.org/all/city> ,
        <http://data.nobelprize.org/all/laureateaward> ,
        <http://data.nobelprize.org/all/country> .
```

Έχοντας στη διάθεσή της τις πιο πάνω πληροφορίες, μια σημασιολογική εφαρμογή θα μπορούσε, αν όχι να πάρει αυτόνομες αποφάσεις, τουλάχιστον να βοηθήσει τον χρήστη να διατυπώσει το ερώτημά του και στη συνέχεια να ξεκινήσει την αναζήτηση. Αξίζει να σημειωθεί ότι το λεξιλόγιο VoID μπορεί επίσης να εκφράσει στατιστικά στοιχεία (πόσες τριάδες, με ποια άλλα αποθετήρια υπάρχει σύνδεση και μέσω ποιου κατηγορήματος κ.ο.κ.).

6.9 Εναλλακτικές μέθοδοι προσπέλασης δεδομένων RDF

Μέχρι τώρα έχουμε γνωρίσει τρεις τρόπους προσπέλασης της σημασιολογικής πληροφορίας:

- *Μέσω του μηχανισμού dereferencing, επισκεπτόμενοι το URI μίας οντότητας*. Είναι η κύρια μέθοδος που συνιστούν οι πρακτικές των Συνδεδεμένων Δεδομένων και αντιστοιχεί στην «πλοήγηση» ενός χρήστη από ιστοσελίδα σε ιστοσελίδα, ακολουθώντας τους υπερσυνδέσμους της HTML. Στην περίπτωση των Συνδεδεμένων Δεδομένων, το έγγραφο που λαμβάνεται περιέχει τριάδες RDF, όπου εμφανίζονται και άλλα σχετιζόμενα URIs. Τα νέα URIs χρησιμεύουν ως υπερσύνδεσμοι σε νέα έγγραφα RDF. Η μέθοδος είναι αποκλειστικά για ανάγνωση, υλοποιείται εύκολα στην πλευρά του εξυπηρετητή, μεταφέρει μικρό όγκο δεδομένων σε κάθε προσπέλαση και απαιτεί από την εφαρμογή-πελάτη να μπορεί να επεξεργάζεται τοπικά τη λαμβανόμενη πληροφορία. Η εκπλήρωση των στόχων της εφαρμογής πιθανόν να απαιτήσει πολλαπλά βήματα επίσκεψης και επεξεργασίας.

- *Μέσω ενός τελικού σημείου εξυπηρέτησης SPARQL*. Εδώ η πολυπλοκότητα υλοποίησης και το βάρος της επεξεργασίας των ερωτημάτων μεταφέρεται στην πλευρά του εξυπηρετητή. Η εφαρμογή-πελάτης αρκεί να διαμορφώσει το κατάλληλο ερώτημα σύμφωνα με τους στόχους της. Η μέθοδος αυτή προσπέλασης προϋποθέτει τη χρήση του μοντέλου του γράφου, τόσο από την εφαρμογή-πελάτη όσο και τον εξυπηρετητή. Η εφαρμογή θα λάβει έτοιμη την πληροφορία που έχει ακριβώς ζητήσει, συνεπώς δεν χρειάζεται τοπική επεξεργασία μετά τη λήψη. Η χρήση της SPARQL 1.1 επιτρέπει όχι μόνο την ανάγνωση αλλά και την εγγραφή σημασιολογικών δεδομένων.

- *Με τη λήψη ολόκληρων συνόλων δεδομένων (RDF data dumps)*. Πολλά αποθετήρια RDF παρέχουν το σύνολο των σημασιολογικών δεδομένων τους σε ένα ή περισσότερα αρχεία τριάδων, τα οποία πιθανόν να έχουν μέγεθος αρκετών GBytes. Η σημασιολογική εφαρμογή πρέπει να «κατεβάσει» όλο το αρχείο, να το αποθηκεύσει τοπικά και στη συνέχεια να προχωρήσει σε όποια επεξεργασία απαιτείται. Η μέθοδος αυτή προϋποθέτει τη μετακίνηση όλης της πληροφορίας στην πλευρά του πελάτη και ενδείκνυται όταν απαιτείται τοπική μαζική (bulk) επεξεργασία των σημασιολογικών δεδομένων. Από την άλλη πλευρά, χάνετε τη σύνδεση με το πρωτότυπο σετ δεδομένων: αν το τελευταίο ενημερωθεί, δεν θα συμβεί το ίδιο στο δικό σας τοπικό αντίγραφο!

Στον παγκόσμιο ιστό, οι μέθοδοι προσπέλασης της πληροφορίας εξελίσσονται διαρκώς. Νέες τάσεις εμφανίζονται, κυρίως ως αποτέλεσμα της εξάπλωσης των «εφαρμογών web»: προγράμματα που εκτελούνται στον φυλλομετρητή του χρήστη και προσπελαύνουν τα δεδομένα τους μέσω προγραμματιστικών διεπαφών

(web APIs). Δεν θα φανεί λοιπόν περίεργο το γεγονός ότι και τα πρότυπα και οι πρακτικές των Συνδεδεμένων Δεδομένων ακολουθούν αυτό ακριβώς το προγραμματιστικό παράδειγμα προσπέλασης. Πριν προχωρήσουμε όμως στην παράθεση παραδειγμάτων για το νέο, εναλλακτικό μοντέλο προσπέλασης, ας δούμε τι ακριβώς συμβαίνει στον παγκόσμιο ιστό και τα web APIs.

Η φράση-κλειδί είναι *Representational State Transfer* (REST) [4]: Αν και είναι αδύνατο να περιγράψουμε με ακρίβεια σε μία μόνο παράγραφο την αρχιτεκτονική λογισμικού REST, τα κύρια χαρακτηριστικά της είναι τα ακόλουθα:

- Πρόκειται για μια κατανεμημένη αρχιτεκτονική client-server.
- Η εφαρμογή στην πλευρά του πελάτη (client) βρίσκεται σε διάφορες καταστάσεις (application state), όπως τις αντιλαμβάνεται ο χρήστης.
- Στη μεριά του εξυπηρετητή (server) αποθηκεύονται οι πόροι (resources) που διαχειρίζεται η εφαρμογή. Οι πόροι βρίσκονται και αυτοί σε διάφορες καταστάσεις (resource state), ως προς το περιεχόμενό τους.
- Ο χρήστης μέσω υπερμέσων (hypermedia), όπως π.χ. οι σύνδεσμοι HTML, προκαλεί αλλαγές στην κατάσταση των πόρων του εξυπηρετητή.
- Με τη σειρά του ο εξυπηρετητής στέλνει στον πελάτη αναπαραστάσεις της κατάστασης των πόρων, προκαλώντας την αλλαγή της κατάστασης της εφαρμογής του πελάτη.

Στον παγκόσμιο ιστό η αρχιτεκτονική REST υλοποιείται μέσω των χαρακτηριστικών του πρωτοκόλλου HTTP. Το κύριο γνώρισμα υλοποίησης είναι η χρήση των διαφόρων μεθόδων του HTTP για τη διαχείριση των πόρων. Το μοντέλο οργάνωσης δεδομένων προβλέπει τη χρήση συλλογών (collections), μέσα στις οποίες μπορούμε να εισάγουμε, να τροποποιήσουμε ή να διαγράψουμε άλλους πόρους. Οι κύριες μέθοδοι HTTP χρησιμοποιούνται σύμφωνα με τον πίνακα 6.2:

Πίνακας 6.2: Η χρήση των μεθόδων HTTP σύμφωνα με την αρχιτεκτονική REST

Μέθοδος	Χρήση
`GET`	Προσπέλαση ενός απλού πόρου ή μιας συλλογής. Η μέθοδος GET δεν προκαλεί αλλαγές στην κατάσταση των πόρων (safe) και πολλαπλές συνεχόμενες αιτήσεις GET για τον ίδιο πόρο θα φέρουν το ίδιο αποτέλεσμα (idempotent).
`POST`	Δημιουργία νέου πόρου μέσα σε μια συλλογή. Η μέθοδος POST προκαλεί αλλαγές στην κατάσταση των πόρων.

Μέθοδος	Χρήση
`PUT`	Αντικατάσταση της πληροφορίας ενός πόρου. Όμοιες συνεχόμενες αιτήσεις PUT για τον ίδιο πόρο θα έχουν το ίδιο αποτέλεσμα (idempotent). Χρησιμοποιείται και για τη δημιουργία νέων πόρων.
`DELETE`	Διαγραφή ενός πόρου. Πολλαπλές συνεχόμενες αιτήσεις DELETE για τον ίδιο πόρο θα έχουν το ίδιο αποτέλεσμα (idempotent).

Θα ολοκληρώσουμε την (υπερβολικά απλουστευτική) σύνοψη της αρχιτεκτονικής REST με το θέμα των *υπερμέσων* (hypermedia). Στον παγκόσμιο ιστό γνωρίζουμε όλοι ένα κλασικό υπερμέσο: τους υπερσυνδέσμους που οδηγούν από ιστοσελίδα σε ιστοσελίδα. Αν το δούμε αφαιρετικά, τα υπερμέσα είναι εκείνα που περιγράφουν στην εφαρμογή-πελάτη –είτε πρόκειται για τον φυλλομετρητή είτε για μια σημασιολογική εφαρμογή– τον τρόπο με τον οποίο θα αλληλεπιδράσει με τον εξυπηρετητή για να εκπληρώσει τους στόχους της.

Είμαστε πλέον σε θέση να γνωρίσουμε ένα πολύ πρόσφατο[3] πρότυπο προσπέλασης των Συνδεδεμένων Δεδομένων, το πρότυπο Linked Data Platform (LDP, 2015). Το πρότυπο αυτό περιγράφει μεθόδους προσπέλασης σημασιολογικών (και όχι μόνο) δεδομένων για ανάγνωση και εγγραφή, σύμφωνα με την τεχνοτροπία της αρχιτεκτονικής REST.

Το πρότυπο περιγράφει δύο είδη *πόρων* (Linked Data Platform Resources – LDPRs): το πρώτο είδος μπορεί να αναπαρασταθεί σε μορφή RDF (LDP-RS) και το δεύτερο σε οποιαδήποτε άλλη μορφή, όπως οι ιστοσελίδες και οι εικόνες (LDP-NR).

Ένα ιδιαίτερο είδος πόρων σε μορφή RDF είναι οι *περιέκτες* (containers): πόροι που οργανώνουν άλλους πόρους σε συλλογές, παρέχοντας τρόπους για εισαγωγή, ανάκτηση, ενημέρωση και διαγραφή των περιεχόμενων πόρων. Το πρότυπο προβλέπει διάφορους τύπους περιεκτών, προσπαθώντας να καλύψει κάθε ανάγκη διαχείρισης Συνδεδεμένων Δεδομένων.

Στο πρότυπο LDP χρησιμοποιούνται οι μέθοδοι HTTP σύμφωνα με την αρχιτεκτονική REST και προσδιορίζονται οι επικεφαλίδες αίτησης και απόκρισης του πρωτοκόλλου HTTP που πρέπει να υποστηρίζει κάθε σύστημα που υλοποιεί το πρότυπο. Ο εξυπηρετητής θα πρέπει να μπορεί να απαντήσει σε μορφή Turtle όταν πρόκειται για δεδομένα RDF, ενώ συνιστάται και η μορφή JSON-LD.

Όταν προσπελάσετε έναν πόρο, η επικεφαλίδα απόκρισης `Link` RFC5988 πρέπει να προσδιορίζει το είδος του πόρου. Για παράδειγμα, το απόσπασμα της επόμενης απόκρισης δηλώνει ότι προσπελαύνουμε έναν γενικό πόρο:

[3] Πρόσφατο κατά τη συγγραφή του παρόντος βιβλίου.

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
```

Μία επέκταση του προτύπου (Linked Data Platform Paging) προβλέπει τη δυνατότητα σελιδοποίησης (paging) των αποτελεσμάτων, μια ιδιαίτερα χρήσιμη λειτουργικότητα για μεγάλα σετ δεδομένων. Παρατηρήστε πώς η επικεφαλίδα `Link` χρησιμοποιείται ως υπερμέσο, οδηγώντας τη σημασιολογική εφαρμογή στην επόμενη σελίδα (`rel="next"`):

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type",
      <http://www.w3.org/ns/ldp#Page>; rel="type"
Link: <http://example.org/episodes?p=2>; rel="next"
```

Αν και στο βιβλίο αυτό προτιμάμε πάντοτε πραγματικά παραδείγματα, δεν υπάρχουν ακόμη ρεαλιστικές χρήσεις του προτύπου LDP. Για τον λόγο αυτόν, θα παραθέσουμε ορισμένα παραδείγματα αλληλεπίδρασης με περιέκτες και πόρους σε ένα υποθετικό σύστημα που υλοποιεί το πρότυπο LDP. Θυμηθείτε ότι, αν και οι αναπαραστάσεις που εμφανίζονται είναι σε μορφή τριάδων RDF, τίποτα δεν υποχρεώνει το σύστημα να αποθηκεύει τους πόρους στη μορφή αυτή: πρόκειται μόνο για την αναπαράσταση RDF των πόρων, όπως θα σταλεί στην εφαρμογή-πελάτη!

Στο παράδειγμα θα χρησιμοποιήσουμε τη γνωστή ομάδα επεισοδίων της σειράς «Doctor Who» που είδαμε και σε προηγούμενο κεφάλαιο. Πρόκειται για τέσσερα επεισόδια που συγκροτούν την ομάδα· θεωρήστε ότι υπάρχει ένας περιέκτης (container) που περιλαμβάνει ήδη τα τρία από τα επεισόδια αυτά. Το URI του περιέκτη είναι

```
http://example.org/episodes/
```

και μια επίσκεψη με τη μέθοδο `GET` θα σας δώσει:

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#BasicContainer>; rel="type",
      <http://www.w3.org/ns/ldp#Resource>; rel="type"

@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
```

```
<> a ldp:Container, ldp:BasicContainer;
    dcterms:created "2011-08-19T13:51:03+01:00"^^xsd:dateTime ;
    dcterms:modified "2012-07-04T13:10:24+01:00"^^xsd:dateTime ;
    dcterms:title "Episodes of the series \"An Unearthly Child\"" ;
    ldp:contains <episode1>, <episode2>, <episode3> .
```

Σύμφωνα με την επικεφαλίδα `Link` πρόκειται για έναν *βασικό περιέκτη* (Basic Container – LDP-BC), ο οποίος δηλώνει τη σχέση με τα περιεχόμενά του μέσω της σχέσης `ldp:contains`. Πέρα από αυτό, ένας βασικός περιέκτης δεν προσφέρει καμία άλλη ευκολία. Παρατηρήστε ότι χρησιμοποιούνται σχετικά (relative) URIs, τα οποία υποστηρίζει η Turtle και υπολογίζονται πάντα σε σχέση με το URI του εγγράφου:

- το `<>` (δηλαδή «αυτό το έγγραφο») είναι συντομογραφία του `http://example.org/episodes/`.
- το `<episode1>` ισοδυναμεί με το απόλυτο URI `http://example.org/episodes/episode1`.

Μία επίσκεψη στο URI ενός επεισοδίου (ένας πόρος LDP-RS), για παράδειγμα του

```
http://example.org/episodes/episode2
```

θα σας επιστρέψει την αναπαράστασή του σε Turtle:

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"

@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix po: <http://purl.org/ontology/po/> .

<>
    dcterms:created "2013-11-08T22:31:09Z"^^xsd:dateTime ;
    dcterms:modified "2013-11-11T12:29:01Z"^^xsd:dateTime ;
    rdfs:label "Description of the episode \"The Cave of Skulls\"" ;
    foaf:primaryTopic <#programme> .

<#programme> a po:Episode ;
    dc:title "The Cave of Skulls" ;
    po:position "2"^^xsd:int ;
    po:short_synopsis "The TARDIS travels to Stone Age times, where the
↪   crew make the mistake of going outside." .
```

Αντίστοιχη πληροφορία περιλαμβάνεται στα έγγραφα των άλλων δύο επεισοδίων. Παρατηρήστε ότι το URI

```
http://example.org/episodes/episode2#programme
```

αναπαριστά το *επεισόδιο καθαυτό*, έναν μη πληροφοριακό, δηλαδή, πόρο.

Για να προσθέσει το τέταρτο επεισόδιο στον περιέκτη, μια εφαρμογή δημιουργεί αίτηση με τη μέθοδο `POST` στο URI του περιέκτη, στέλνοντας μαζί το νέο περιεχόμενο:

```
POST /episodes/ HTTP/1.1
Host: example.org
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
Content-Type: text/turtle

@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix po: <http://purl.org/ontology/po/> .

<>
    rdfs:label "Description of the episode \"The Firemaker\"" ;
    foaf:primaryTopic <#programme> .

<#programme> a po:Episode ;
    dc:title "The Firemaker" ;
    po:position "4"^^xsd:int ;
    po:short_synopsis "The Doctor and his companions use guile to escape
↪  death at the hands of their captors." .
```

Ο εξυπηρετητής απαντά, δηλώνοντας ότι η εισαγωγή έγινε με επιτυχία και παρέχει το URI του νέου πόρου:

```
HTTP/1.1 201 Created
Location: http://example.org/episodes/episode4
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
```

Ταυτοχρόνως αλλάζει και η κατάσταση του περιέκτη για να συμπεριλάβει και το `episode4` ως μέλος. Αν ζητήσετε ξανά την αναπαράστασή του, θα λάβετε:

```
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<> a ldp:Container, ldp:BasicContainer;
    dcterms:created "2011-08-19T13:51:03+01:00"^^xsd:dateTime ;
    dcterms:modified "2012-07-04T13:10:24+01:00"^^xsd:dateTime ;
    dcterms:title "Episodes of the series \"An Unearthly Child\"" ;
    ldp:contains <episode1>, <episode2>, <episode3>, <episode4> .
```

Όπως είδαμε στα προηγούμενα παραδείγματα, ο βασικός περιέκτης διαχειρίζεται τη σχέση `ldp:contains`. Εάν θέλουμε να διαγράψουμε ένα από τα περιεχόμενα μέλη θα χρησιμοποιήσουμε τη μέθοδο `DELETE` του πρωτοκόλλου HTTP. Για παράδειγμα, για να διαγράψουμε το `episode3`:

```
DELETE /episodes/episode3 HTTP/1.1
Host: example.org
```

Μετά την αίτηση, εκτός από τη διαγραφή του `episode3`, θα ενημερωθεί ο περιέκτης έτσι ώστε να περιλαμβάνει μόνο τα υπόλοιπα επεισόδια:

```
<> a ldp:Container, ldp:BasicContainer;
    ldp:contains <episode1>, <episode2>, <episode4> .
```

Η αυτόματη διαχείριση του `ldp:contains` είναι η μόνη λειτουργικότητα που παρέχει ο βασικός περιέκτης. Αυτή καλύπτει αρκετές περιπτώσεις αποθήκευσης και οργάνωσης της σημασιολογικής πληροφορίας. Υπάρχουν όμως και περιπτώσεις που χρειαζόμαστε πρόσθετες ικανότητες από έναν περιέκτη:

- Πολλές πηγές σημασιολογικών δεδομένων χρησιμοποιούν ήδη *τα δικά τους κατηγορήματα* για να δηλώσουν την ιδιότητα του μέλους. Δεν θα ήταν χρήσιμο να μπορεί ο περιέκτης να ενημερώνει αυτόματα και εκείνα τα κατηγορήματα;
- Το `ldp:contains` συνδέει με την ιδιότητα του μέλους δύο πόρους: τον περιέκτη και το περιεχόμενο έγγραφο. Και οι δύο αυτοί πόροι είναι πληροφοριακές πηγές, με άλλα λόγια, έγγραφα RDF (ή άλλα έγγραφα). Σε πολλές περιπτώσεις οργάνωσης της σημασιολογικής πληροφορίας, οι σχέσεις μέλους *δεν αφορούν τα έγγραφα RDF, αλλά τις ίδιες τις μη πληροφοριακές οντότητες*. Πώς θα μπορούσε ένας περιέκτης να διαχειριστεί αυτόματα και αυτή την περίπτωση;

Για να καλύψει όσο το δυνατόν περισσότερες ανάγκες, το πρότυπο LDP παρέχει και δύο άλλους τύπους περιέκτη, τον *άμεσο περιέκτη* (Direct Container – LDP-DC)

και τον *έμμεσο περιέκτη* (Indirect Container – LDP-IC). Στη συνέχεια, θα δούμε το προηγούμενο παράδειγμα, χρησιμοποιώντας όμως έναν έμμεσο περιέκτη: το είδος που παρέχει τη μεγαλύτερη ευελιξία στη διαχείριση σημασιολογικών δεδομένων.

Στα πραγματικά δεδομένα RDF της σειράς «Doctor Who», η σχέση μέλους μεταξύ της ομάδας επεισοδίων και κάθε επεισοδίου δηλώνεται με τη σχέση:

```
po:episode
```

όπου το υποκείμενο και το αντικείμενο δεν είναι τα URIs των εγγράφων που μιλούν για τις οντότητες, αλλά οι οντότητες (ομάδας και επεισοδίων) καθαυτές. Αυτή ακριβώς τη σχέση, πέραν του βασικού `ldp:contains`, θα προσπαθήσουμε να εκφράσουμε στο επόμενο παράδειγμα.

Όπως περιγράψαμε και νωρίτερα, ο περιέκτης αρχικά περιέχει τα τρία πρώτα επεισόδια. Αν τον επισκεφτείτε μέσω της μεθόδου `GET` θα πάρετε το εξής:

```
HTTP/1.1 200 OK
Content-Type: text/turtle; charset=UTF-8
Link: <http://www.w3.org/ns/ldp#IndirectContainer>; rel="type",
      <http://www.w3.org/ns/ldp#Resource>; rel="type"


@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix po: <http://purl.org/ontology/po/> .

<> a ldp:Container, ldp:IndirectContainer;

    ldp:membershipResource <#programme>;
    ldp:hasMemberRelation po:episode;
    ldp:insertedContentRelation foaf:primaryTopic;

    dcterms:created "2011-08-19T13:51:03+01:00"^^xsd:dateTime ;
    dcterms:modified "2012-07-04T13:10:24+01:00"^^xsd:dateTime ;
    dcterms:title "Episodes of the series \"An Unearthly Child\"" ;
    ldp:contains <episode1>, <episode2>, <episode3> .

<#programme> a po:Series ;
    dcterms:title "An Unearthly Child" ;
    po:short_synopsis "Two school teachers' curiosity leads them to a
↪   junk yard where they meet the Doctor." ;

    po:episode <episode1#programme>, <episode2#programme>,
↪   <episode3#programme> .
```

Ο περιέκτης τώρα μοιάζει περισσότερο με το αυθεντικό έγγραφο RDF. Εκτός από το τεχνητό `ldp:contains` που επιβάλει το πρότυπο LDP, υπάρχει και η αρχική σχέση `po:episode`, και μάλιστα με τη σωστή της χρήση: μεταξύ μη πληροφοριακών οντοτήτων κι όχι μεταξύ εγγράφων RDF. Με ποιον τρόπο γνωρίζει όμως ο περιέκτης πώς να διαχειριστεί το `po:episode`; Κατά τη δημιουργία του περιέκτη, ο οποίος εισάγεται μέσω της μεθόδου POST όπως κάθε άλλος πόρος, έχουν οριστεί οι παρακάτω σχέσεις:

`ldp:membershipResource`: το αντικείμενο της τριάδας με τη σχέση αυτή θα είναι η οντότητα που περιέχει τις άλλες. Στο προηγούμενο παράδειγμα, η τριάδα:

```
<> ldp:membershipResource <#programme> .
```

περιγράφει ότι ο η οντότητα αυτή είναι η ακόλουθη:

```
http://example.org/episodes/#programme
```

`ldp:hasMemberRelation`: το αντικείμενο της τριάδας με τη σχέση αυτή θα αποτελέσει το κατηγόρημα των νέων τριάδων με τη σχέση μέλους. Η τριάδα του παραδείγματος:

```
<> ldp:hasMemberRelation po:episode .
```

περιγράφει τη δημιουργία τριάδων στη μορφή:

```
http://example.org/episodes/#programme po:episode episode-URI .
```

`ldp:insertedContentRelation`: το αντικείμενο της τριάδας είναι μια σχέση που θα αναζητηθεί κατά την εισαγωγή νέων οντοτήτων. Όταν βρεθεί τέτοια τριάδα στα δεδομένα της οντότητας προς εισαγωγή, το αντικείμενό της θα χρησιμοποιηθεί ως αντικείμενο στη νέα σχέση μέλους. Σας φαίνεται μπερδεμένο; Ακολουθήστε το παράδειγμα!

Ο περιέκτης έχει δημιουργηθεί με το εξής χαρακτηριστικό:

```
<> ldp:insertedContentRelation foaf:primaryTopic .
```

Έστω ότι εισάγετε το τέταρτο επεισόδιο, ακριβώς όπως προηγουμένως:

```
POST /episodes/ HTTP/1.1
Host: example.org
Link: <http://www.w3.org/ns/ldp#Resource>; rel="type"
Content-Type: text/turtle
```

```
@prefix ldp: <http://www.w3.org/ns/ldp#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix po: <http://purl.org/ontology/po/> .

<>
    rdfs:label "Description of the episode \"The Firemaker\"" ;
    foaf:primaryTopic <#programme> .

<#programme> a po:Episode ;
    dc:title "The Firemaker" ;
    po:position "4"^^xsd:int ;
    po:short_synopsis "The Doctor and his companions use guile to escape
↪   death at the hands of their captors." .
```

Κατά την εισαγωγή των δεδομένων αυτών, θα εντοπιστεί η εξής τριάδα με κατηγόρημα το `foaf:primaryTopic`:

```
<> foaf:primaryTopic <#programme> .
```

Κατά συνέπεια, ταυτόχρονα με την αποθήκευση του εγγράφου του νέου επεισοδίου, ο περιέκτης θα προσθέσει στα δικά του περιεχόμενα:

```
<> ldp:contains <episode4> .

<#programme> po:episode <episode4#programme> .
```

Η αντίστοιχη λειτουργία ενημέρωσης τόσο στις τριάδες του `ldp:contains` όσο και στις τριάδες με το `po:episode` θα εκτελεστεί σε περίπτωση διαγραφής κάποιου επεισοδίου.

Στο σημείο αυτό ολοκληρώνεται η σύντομη παρουσίαση του προτύπου LDP. Στα προηγούμενα παραδείγματα είδαμε κυρίως το νέο στοιχείο της χρήσης περιεκτών για την αποθήκευση σημασιολογικών πόρων. Το πρότυπο LDP περιγράφει αναλυτικότερα τις τεχνικές πτυχές των παραπάνω. Για περισσότερα παραδείγματα μπορείτε να ανατρέξετε στο [5].

Σημείωση: Πατώντας στη στέρεα βάση του HTTP.

Ίσως έχετε αναρωτηθεί εάν το πρότυποLDP προσδιορίζει μεθόδους πιστοποίησης και ελέγχου πρόσβασης στα δεδομένα. Η απάντηση είναι αρνητική: το πρότυπο

δεν επανεφευρίσκει τον τροχό! Αντιθέτως, βασίζεται στους υπάρχοντες μηχανισμούς ασφάλειας του παγκόσμιου ιστού και του πρωτοκόλλου HTTP. Πατώντας πάνω στη στέρεα βάση του HTTP, έχουμε, ως πρόσθετο πλεονέκτημα, έτοιμες λύσεις για:

- την επιλογή προτιμήσεων κατά τις αιτήσεις, όπως η διαπραγμάτευση περιεχομένου
- την πιστοποίηση και τον έλεγχο πρόσβασης στους πόρους του συστήματος
- τη διαχείριση πολλαπλών ταυτόχρονων αλλαγών στον ίδιο πόρο
- τη χρήση υπερμέσων για την καθοδήγηση της εφαρμογής στα επόμενα βήματα

6.10 Ελέγξτε τις γνώσεις σας

1. Τι συμβαίνει κατά την επίσκεψη ενός URI που δεν αναπαριστά πληροφοριακό πόρο;

    i. Δεν είναι δυνατή η επίσκεψη σε κάτι που δεν είναι έγγραφο του παγκόσμιου ιστού. Επιστρέφεται σφάλμα.

    ii. Επιστρέφεται ένα έγγραφο με πληροφορία σχετική με την οντότητα του URI.

    iii. Επιστρέφεται η ψηφιακή αναπαράσταση του πόρου.

2. Κατά την πρακτική των Συνδεδεμένων Δεδομένων, οι χώροι ονομάτων που χρησιμοποιούνται

    i. τελειώνουν πάντοτε με τον χαρακτήρα #.

    ii. τελειώνουν είτε με # είτε με /.

    iii. δεν έχουν κανέναν περιορισμό ως προς το ποιος θα είναι ο τελευταίος χαρακτήρας.

3. Πόσες προσπελάσεις απαιτούνται για την επίσκεψη ενός URI από χώρο ονομάτων που τελειώνει σε #;

    i. Μία προσπέλαση, για την επίσκεψη του URI αφού αφαιρεθεί το απόσπασμα πριν γίνει η αίτηση.

    ii. Μία προσπέλαση, για την επίσκεψη του URI που περιέχει και το απόσπασμα. Ο εξυπηρετητής θα αφαιρέσει το απόσπασμα πριν απαντήσει.

    iii. Δύο προσπελάσεις, μία φορά για το URI χωρίς το απόσπασμα, τη δεύτερη για το απόσπασμα μόνο.

4. Τι θα λάβετε κατά την επίσκεψη ενός URI από χώρο ονομάτων που τελειώνει σε #;

    i. Ένα έγγραφο με τριάδες σχετικές με την οντότητα που αντιπροσωπεύει το URI.

    ii. Ένα έγγραφο με τριάδες όπου το συγκεκριμένο URI βρίσκεται σε οποιαδήποτε θέση υποκειμένου, κατηγορήματος ή αντικειμένου.

    iii. Ένα έγγραφο με διάφορες τριάδες, μέσα στις οποίες βρίσκονται και ορισμένες με το συγκεκριμένο URI σε θέση υποκειμένου ή αντικειμένου.

5. Γιατί χρησιμοποιείται η ανακατεύθυνση `303 See Other` στα Συνδεδεμένα Δεδομένα;

    i. Για να ανακατευθύνει την αίτηση στην κατάλληλη αναπαράσταση του URI που επισκεπτόμαστε.

    ii. Για να ανακατευθύνει την επίσκεψη από URI μη πληροφοριακού πόρου στο URI ενός εγγράφου με πληροφορία σχετική με τον μη πληροφοριακό πόρο.

    iii. Για να ανακατευθύνει την επίσκεψη από ένα έγγραφο του παγκόσμιου ιστού σε έναν μη πληροφοριακό πόρο.

6. Πότε μια τριάδα RDF μπορεί να αποτελέσει σύνδεσμο μεταξύ δύο αποθετηρίων RDF;

    i. Όταν η ίδια τριάδα βρίσκεται και στα δύο αποθετήρια.

    ii. Όταν τα δύο αποθετήρια έχουν την ίδια τριάδα αλλά αντεστραμμένη (στο δεύτερο αποθετήριο το υποκείμενο και το αντικείμενο της τριάδας έχουν αντιστραφεί).

    iii. Όταν η τριάδα, σε οποιοδήποτε από τα δύο αποθετήρια, συνδέει μία οντότητα που ανήκει στο πρώτο αποθετήριο με κάποια οντότητα του δεύτερου αποθετηρίου.

7. Το λεξιλόγιο VoID

    i. παρέχει κλάσεις και κατηγορήματα για την αποθήκευση δεδομένων σε ένα αποθετήριο RDF.

    ii. επιτρέπει την έκφραση μεταδεδομένων σχετικών με ένα αποθετήριο RDF.

    iii. διασυνδέει ομάδες αποθετηρίων RDF.

8. Ποια η διαφορά λήψης σημασιολογικών δεδομένων μέσω επίσκεψης URI και μέσω ερωτημάτων SPARQL;

i. Με το ερώτημα SPARQL η επεξεργασία γίνεται στην πλευρά του εξυπηρετητή, ενώ με την επίσκεψη URI γίνεται στην πλευρά της εφαρμογής-πελάτη.

ii. Με το ερώτημα SPARQL η μεταδιδόμενη πληροφορία είναι πολύ περισσότερη απ' ό,τι με την επίσκεψη σε URI.

iii. Με την επίσκεψη σε URI οι εφαρμογές ανακαλύπτουν νέα URIs για επόμενες επισκέψεις, κάτι που δεν συμβαίνει με το ερώτημα SPARQL.

9. Το πρότυπο Linked Data Platform

i. επιτρέπει την ανάγνωση και την εγγραφή τριάδων RDF σε βάσεις γράφων.

ii. επιτρέπει την ανάγνωση και την εγγραφή εγγράφων RDF, τα οποία οργανώνονται σε περιέκτες.

iii. επιτρέπει την ανάγνωση και την εγγραφή εγγράφων RDF ή άλλης μορφής, τα οποία οργανώνονται σε περιέκτες.

6.11 Σύνοψη

Με το κεφάλαιο αυτό ολοκληρώσαμε την παρουσίαση των προτύπων και των μεθόδων προσπέλασης των Συνδεδεμένων Δεδομένων. Γνωρίζουμε πλέον τον τυπικό τρόπο λειτουργίας μιας εφαρμογής, η οποία καταναλώνει σημασιολογικά δεδομένα, εξάγει συμπεράσματα και αναπροσαρμόζει δυναμικά τους στόχους της. Οι σημασιολογικές εφαρμογές κινούνται στον παγκόσμιο ιστό σε έναν χώρο ανοικτών και αλληλοσυνδεόμενων αποθετηρίων RDF, ο οποίος εύστοχα ονομάζεται νέφος των Ανοικτών Συνδεδεμένων Δεδομένων (LOD cloud). Ο χώρος των σημασιολογικών δεδομένων διαμορφώνεται διαρκώς: με το πέρασμα του χρόνου εμφανίζονται νέα πρότυπα, όπως το πρότυπο Linked Data Platform, ενώ συγχρόνως αλλάζουν και οι τάσεις στον τρόπο χρήσης των σημασιολογικών δεδομένων από τις εφαρμογές. Στο τελευταίο κεφάλαιο θα παραθέσουμε πληροφορίες για το πώς θα παρακολουθήσετε την έρευνα και τις νέες τάσεις που διαμορφώνουν τη σύγχρονη όψη των Ανοικτών Συνδεδεμένων Δεδομένων.

6.12 Βιβλιογραφικές αναφορές

[1] Bates, M. J. (1989). *The design of browsing and berrypicking techniques for the on-line search interface*, Online Review, 13(5), pp. 407-431.

[2] *Cool URIs for the Semantic Web*, W3C Interest Group Note, 03 December 2008, διαθέσιμο από: http://www.w3.org/TR/cooluris/, ημερομηνία πρόσβασης: 11.10.2015.

[3] Max Schmachtenberg, Christian Bizer, Anja Jentzsch and Richard Cyganiak,

The Linking Open Data cloud diagram, διαθέσιμο από: http://lod-cloud.net/, ημερομηνία πρόσβασης: 11.10.2015.

[4] Fielding, R.-T. 2000. *Architectural Styles and the Design of Network-based Software Architectures* (διδ. διατρ.), διαθέσιμο από: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm, ημερομηνία πρόσβασης: 11.10.2015.

[5] *Linked Data Platform 1.0 Primer*, 23 April 2015, διαθέσιμο από: http://www.w3.org/TR/ldp-primer/, ημερομηνία πρόσβασης: 11.10.2015.