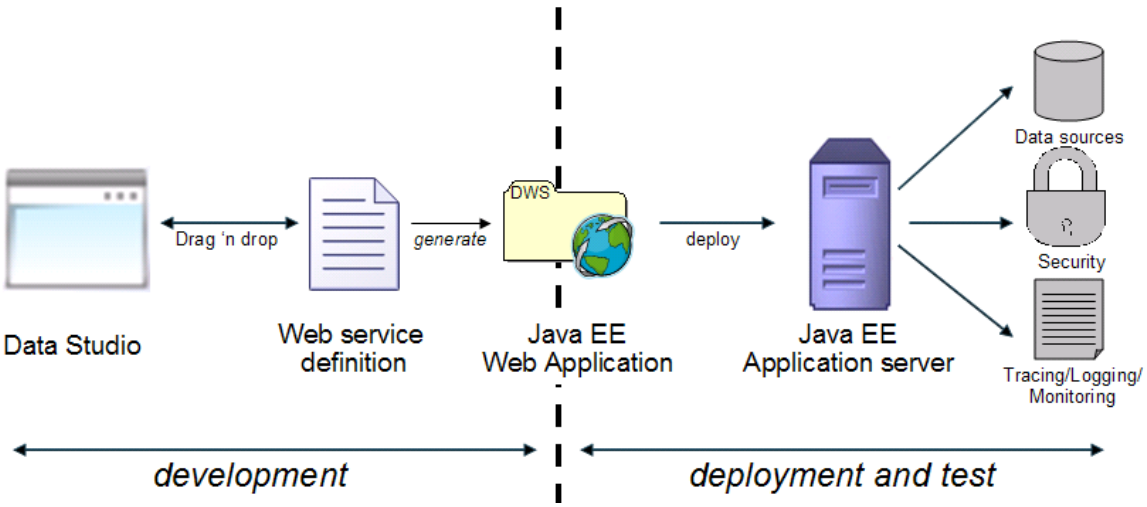
ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

Data Web Services με χρήση του IBM Data Studio



Του φοιτητή
Χρήστου Μάνιου
Αρ. μητρώου: 06/3018

Επιβλέπων Καθηγητής
Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2010

Ευχαριστίες

Θα ήθελα να καταρχήν ευχαριστήσω τους γονείς μου για την αέναη υπομονή που δείχνουν σε κάθε βήμα που κάνω στη ζωή μου. Επιπροσθέτως θα ήθελα να ευχαριστήσω όλους τους καθηγητές μου στο ΤΕΙ Πληροφορικής Θεσσαλονίκης και κυρίως τους κους Κεραμόπουλο Ευκλείδη και Δέρβο Δημήτριο για τις γνώσεις και την αγάπη για αναζήτηση που μου εμφύσησαν πάνω στο πεδίο των Βάσεων Δεδομένων.

Abstract

Web 2.0 is growing while technologies associated with web services are improved. Simultaneously, access to structured data which are stored in databases is a quite difficult and tricky process to implement in a web service. Writing code to access a database requires repetitive open connection, execute statement, collect results, close connection commands that become quite difficult to maintain. IBM Data Studio IDE has the solution: Creation of Data Web Services, web services oriented to structured database data, with no need of writing a single line of code. The developer implements his own stored procedures or SQL scripts targeting to an IBM Database Server (i.e. IBM DB2) and with a few drag and drop steps creates SOAP and REST Data Web Services which are deployed on a Java application server (i.e. IBM WebSphere). Developing a Data Web Service becomes an easy task allowing the developer to focus on database logic.

This paper explains how the technologies, programming platforms and databases cooperate to create an IBM Data Web Service. Through easy steps we provide the amount of information needed to get started with IBM Data Studio, create and consume a Data Web Service that targets to an IBM DB2 database server.

Ευρετήριο Περιεχομένων

Ευρετήριο Σχημάτων

1. Εισαγωγή

Η εργασία αυτή έχει ως σκοπό τη μελέτη και την παρουσίαση των Data Web Services με τη χρήση του IBM Data Studio. Με τη χρήση των Data Web Services παρέχονται εργαλεία για τη δημιουργία Web Services τα οποία βασίζονται σε SQL εντολές και Stored Procedures. Με αυτόν τον τρόπο δίνεται εύκολα πρόσβαση σε δομημένα δεδομένα τα οποία είναι αποθηκευμένα σε IBM database servers όπως η DB2. Η διαδικασία παραγωγής μιας web service αυτή απαιτεί γενικώς πολύ κώδικα, αλλά με τη χρήση του Data Studio αρκούν κάποιες SQL εντολές ή stored procedures και μερικά drag and drop.

Στο 2^{ο} κεφάλαιο γίνεται μια σύντομη εισαγωγή στις Web Services και την αρχιτεκτονική υπηρεσιών ιστού. Αναλύονται οι τεχνολογίες που χρησιμοποιούνται καθώς και ο τρόπος ανάλωσης των υπηρεσιών ιστού.

Στο 3^{ο} κεφάλαιο περιγράφουμε τις Data Web Services, μια νέα κατηγορία υπηρεσιών ιστού προσανατολισμένη στα δεδομένα. Παρατίθενται οι ορισμοί που δίνονται από την IBM και τη Microsoft για την υλοποίηση των Data Web Services. Ακολούθως γίνεται εισαγωγή στις τεχνολογίες και τις πλατφόρμες λογισμικού που χρησιμοποιούν οι IBM Data Web Services.

Στο 4^{ο} κεφάλαιο περιγράφουμε αναλυτικά τα βήματα για την προετοιμασία του IBM data Studio, την προετοιμασία της βάσης δεδομένων, του WebSphere server για τη δημιουργία των Data Web Services. Ακολούθως δημιουργούμε βήμα – βήμα μια Data Web Service που αναζητά φοιτητές, την findstudents, βασισμένη σε πραγματικά δεδομένα φοιτητών τα οποία έχουμε λάβει από δημόσιο ιστοτόπο του TEI (http://hydra.it.teithe.gr/s).

Στο 5^{ο} κεφάλαιο περιγράφουμε τους τρόπους ανάλωσης των Data Web Services χρησιμοποιώντας ως πηγή την DWS findstudents που έχουμε δημιουργήσει στο 4^{ο} κεφάλαιο. Περιγράφουμε τις προϋποθέσεις και τους περιορισμούς που εμπεριέχει ο κάθε τρόπος ανάλωσης μιας Data Web Service.

Στο 6^{ο} κεφάλαιο περιγράφουμε τις δοκιμαστικές Data Web Services που δημιουργήθηκαν καθώς και τις ιστοσελίδες που υλοποιήθηκαν σε PHP προκειμένου να τις αναλώσουν. Επίσης περιλαμβάνονται οδηγίες χρήσης του αρχείου VMWare image με το οποίο μπορείτε να δείτε στην πράξη τις υπηρεσίες και τις ιστοσελίδες που δημιουργήθηκαν στα πλαίσια αυτής της πτυχιακής.

2. Web Services

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια εισαγωγή στις υπηρεσίες ιστού, την αρχιτεκτονική τους, τα πρότυπα και στις τεχνολογίες που τις υποστηρίζουν. Παράλληλα αναλύονται τα μέρη των τεχνολογιών που είναι απαραίτητα για την κατανόηση της data web service που θα δημιουργήσουμε στο 4^{ο} κεφάλαιο. Για την καλύτερη κατανόηση κάποιων ενοτήτων χρησιμοποιούμε ως παραδείγματα δεδομένα και στοιχεία από το 4^{ο} κεφάλαιο.

2.2 Ορισμός

Σύμφωνα με το W3C, «.. Υπηρεσία ιστού ονομάζεται ένα σύστημα λογισμικού που υποστηρίζει την διαλειτουργικότητα μεταξύ μηχανών οι οποίες επικοινωνούν μέσω δικτύου. Κάθε υπηρεσία ιστού έχει μια διεπαφή επικοινωνίας η οποία περιγράφεται από μια γλώσσα κατανοητή στα μηχανήματα και είναι γνωστή ως WSDL (web service description language). ..» [13] Μια υπηρεσία ιστού μπορεί να κληθεί από ένα οποιοδήποτε υπολογιστικό σύστημα μέσω του πρωτοκόλλου SOAP ή επικοινωνίας REST ανταλλάσσοντας μηνύματα με βάση την WSDL περιγραφή της. Τα μηνύματα αυτά μεταφέρονται μέσω του HTTP, μορφοποιημένα σε XML σε συνδυασμό με άλλα πρότυπα στον παγκόσμιο ιστό (web).

Στην πράξη μια υπηρεσία ιστού είναι ένα πρόγραμμα γραμμένο σε κάποια γλώσσα ή πλατφόρμα προγραμματισμού (Java, .NET, PHP κ.ο.κ) το οποίο εκθέτει τις λειτουργίες του στον παγκόσμιο ιστό. Με τον όρο λειτουργίες εννοούμε τις μεθόδους που υλοποιεί και είναι προσβάσιμες μέσω πρωτοκόλλων SOAP ή της αρχιτεκτονικής REST. Μια απλή web service μπορεί να είναι μια εφαρμογή που μετατρέπει θερμοκρασία από βαθμούς Κελσίου σε βαθμούς Fahrenheit και το αντίστροφο. Για να την υλοποιήσουμε χρειαζόμαστε δύο μεθόδους μια που μετατρέπει σε βαθμούς Κελσίου και μια που μετατρέπει σε βαθμούς Fahrenheit. Έτσι έχουμε την web service με όνομα tempratureConverter που εκθέτει (exposes) προς ανάλωσιν δύο μεθόδους: την celsiusTofahrenheit και την fahrenheitTocelsius. Μπορείτε να δοκιμάσετε αυτήν την υπηρεσία στο http://www.w3schools.com/webservices/tempconvert.asmx.

Ο κώδικας που υλοποιεί αυτή την απλή μετατροπή, γραμμένος σε Visual Basic .NET είναι ο ακόλουθος:

```
1. <%@ WebService Language="VBScript" Class="TempConvert" %>
2.
3. Imports System
4. Imports System.Web.Services
5.
6. Public Class TempConvert :Inherits WebService
7.
8. <WebMethod()> Public Function FahrenheitToCelsius
9. (ByVal Fahrenheit As String) As String
```

```
10.    dim fahr
11.    fahr=trim(replace(Fahrenheit,",",".",))
12.    if fahr="" or IsNumeric(fahr)=false then return "Error"
13.    return ((((fahr) - 32) / 9) * 5)
14. end function
15.
16. <WebMethod()> Public Function CelsiusToFahrenheit
17. (ByVal Celsius As String) As String
18.    dim cel
19.    cel=trim(replace(Celsius,",",".",))
20.    if cel="" or IsNumeric(cel)=false then return "Error"
21.    return ((((cel) * 9) / 5) + 32)
22. end function
23.
24. end class
```

2.3 Οι ανάγκες που οδήγησαν στις υπηρεσίες ιστού

Στην δεκαετία του ‘90 οι επιχειρήσεις προκειμένου να περικόψουν έξοδα που αφορούσαν σε μηχανογραφικό εξοπλισμό, υπαλλήλους και υλοποίηση εφαρμογών που επιλύουν θέματα ή εκτελούν διαδικασίες, άρχισαν να αναθέτουν μέρος ή ολόκληρες τις διαδικασίες σε εξειδικευμένους εξωτερικούς συνεργάτες. Οι συνεργάτες οι οποίοι ονομάστηκαν πάροχοι υπηρεσιών εφαρμογών (application service providers) αναλαμβάνουν την περαίωση των δραστηριοτήτων και διαδικασιών που αφορούν τις επιχειρήσεις – πελάτες τους με την παραγωγή προγραμμάτων – υπηρεσιών για αυτές. Με την εξάπλωση του παγκοσμίου ιστού και την δημιουργία των standards για τα πρωτόκολλα επικοινωνίας (όπως το HTTP) τα προγράμματα – υπηρεσίες έγιναν διαθέσιμα σε ένα συγκεκριμένο ακραίο κόμβο του διαδικτύου. Με αυτόν τον τρόπο άρχισε να υφαίνεται η έννοια της αρχιτεκτονικής υπηρεσιών ιστού (SOA: Service Oriented Architecture).

Για την επίτευξη της επικοινωνίας μέσω υπηρεσιών, οι οποίες ενδεχομένως εντοπίζονται δυναμικά, απαιτείται χαλαρή σύζευξη (loose coupling) μεταξύ εκείνου που κάνει την αίτηση και της υπηρεσίας που θα χρησιμοποιηθεί. Η έννοια της χαλαρής σύζευξης αποκλείει οποιαδήποτε γνώση ή υποθέσεις σχετικά με τις συγκεκριμένες πλατφόρμες όπου εκτελείται ο αιτών ή ο πάροχος υπηρεσιών, την υλοποίηση που χρησιμοποιεί ο εκάστοτε πάροχος, ή τις μορφές και τα πρωτόκολλα που χρησιμοποιούν για τη μεταξύ τους επικοινωνία.

2.4 Χαρακτηριστικά

Οι υπηρεσίες ιστού έχουν κάποια βασικά στοιχεία τα οποία τις χαρακτηρίζουν:

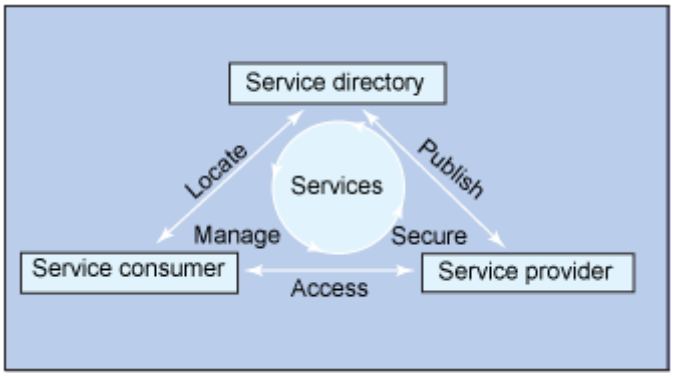
1. Οι υπηρεσίες είναι οντότητες λογισμικού διαθέσιμες σε ένα συγκεκριμένο ακραίο σημείο-κόμβο του δικτύου (ένας server). Λαμβάνουν και στέλνουν **μηνύματα** σύμφωνα με τη συμπεριφορά που έχουν ορίσει στις προδιαγραφές τους μέσω της WSDL.

2. Οι υπηρεσίες κρύβουν μέσα τους το μοντέλο προγραμματισμού και τις λεπτομέρειες της υλοποίησης τους. Εκθέτουν μόνο ότι είναι χρήσιμο στους πελάτες για να τις καλέσουν και να τις χρησιμοποιήσουν.
3. Οι υπηρεσίες ιστού μπορούν να κληθούν από οποιαδήποτε πλατφόρμα λογισμικού ή υλικού. Τα πρωτόκολλα επικοινωνίας τους αποτελούν πρότυπα που υλοποιούνται από όλους τους κατασκευαστές. Έτσι όταν καλούμε μια web service δεν γνωρίζουμε τι λειτουργικό σύστημα κρύβεται από πίσω αλλά ούτε μας ενδιαφέρει αν το λειτουργικό μας σύστημα είναι συμβατό.
4. Οι πληροφορίες που δημοσιεύονται για μια υπηρεσία μέσω της WSDL αρκούν για την περιγραφή της και τον τρόπο ανάλωσης της (αίτηση προς αυτήν, λήψη απάντησης, δικλείδες ασφαλείας). Όλα αυτά ισχύουν υπό την προϋπόθεση ότι ο αιτών γνωρίζει πώς να τα χρησιμοποιήσει.
5. Μια υπηρεσία ιστού μπορεί να ενσωματώσει ή να χρησιμοποιεί άλλες υπηρεσίες.
6. Κάθε υπηρεσία ιστού έχει υποδοχείς (endpoints) οι οποίοι δέχονται τα μηνύματα των αιτήσεων προς την υπηρεσία και αναλαμβάνουν να στείλουν τις απαντήσεις στους πελάτες. Τα endpoints είναι δύο κατηγοριών, SOAP αν χρησιμοποιείται το πρωτόκολλο soap για την επικοινωνία ή REST HTTP GET ή POST αν η υπηρεσία χρησιμοποιείται σαν πόρος του δικτύου.

2.5 Αρχιτεκτονική υπηρεσιών ιστού

Η αρχιτεκτονική υπηρεσιών ιστού ή αρχιτεκτονική προσανατολισμένη στις υπηρεσίες (SOA) αποτελεί ένα συγκεκριμένο αρχιτεκτονικό ύφος το οποίο έχει ως στόχο τη χαλαρή σύνδεση και δυναμική σύζευξη μεταξύ των υπηρεσιών. Για να λειτουργήσει αποδοτικά η SOA πρέπει να διαθέτει ορισμένους συγκεκριμένους παράγοντες.

Καταρχάς είναι απαραίτητο να παρασχεθεί ένας αφηρημένος ορισμός της υπηρεσίας, ο οποίος θα συμπεριλαμβάνει τις λεπτομέρειες που επιτρέπουν σε οποιονδήποτε θέλει να χρησιμοποιήσει την υπηρεσία να ***συνδεθεί*** (bind) καταλλήλως με εκείνη. Δεύτερον οι πάροχοι των υπηρεσιών πρέπει να ***δημοσιεύουν*** (publish) τις λεπτομέρειες των υπηρεσιών έτσι ώστε οι πελάτες τους να μπορούν να κατανοήσουν τον σκοπό, τον τρόπο χρήσης και σύνδεσης με την υπηρεσία. Τρίτον, εκείνη που ζητούν υπηρεσίες, πρέπει να έχουν κάποιο τρόπο να ***εντοπίζουν*** (find) ποιες υπηρεσίες είναι διαθέσιμες και εξυπηρετούν τις ανάγκες τους. Για να λειτουργήσει η προσέγγιση «σύνδεση/δημοσίευση/εντοπισμός» πρέπει να καθοριστούν τα πρότυπα τα οποία διέπουν τι θα δημοσιεύεται και πώς, τον τρόπο που θα εντοπίζονται οι πληροφορίες καθώς και τον τρόπο σύνδεσης σε αυτές.



Σχήμα 2.1 Η προσέγγιση σύνδεση/δημοσίευση/εντοπισμός

Ο πάροχος μια υπηρεσίας περιγράφει τη σημασιολογία της, δηλαδή τις λειτουργίες που υποστηρίζει και τα χαρακτηριστικά σχετικά με τον τρόπο προσπέλασής της σε ένα έγγραφο WSDL (web service description language). Οι πάροχοι των υπηρεσιών δημοσιεύουν τα wsdl έγγραφα των υπηρεσιών τους σε καταλόγους υπηρεσιών ιστού UDDI όπου οι επιχειρήσεις μπορούν να εγγραφούν και να αναζητούν υπηρεσίες ιστού ανάλογα με τις ανάγκες τους.

2.6 Είδη interface Υπηρεσιών

Οι υπηρεσίες ιστού χωρίζονται σε δύο μεγάλες κατηγορίες ανάλογα με την αρχιτεκτονική υλοποίησης που χρησιμοποιούν ως προς τον τρόπο επικοινωνίας με τους πελάτες τους. Η έννοια τρόπος επικοινωνίας συχνά συναντάται στη βιβλιογραφία ως το interface ή τα endpoints της υπηρεσίας ιστού. Η υπηρεσία μπορεί να είναι μία αλλά να έχει παραπάνω από μία υλοποίηση ως προς τον τρόπο επικοινωνίας. Υπάρχουν οι υπηρεσίες που επικοινωνούν μέσω του πρωτοκόλλου SOAP και οι υπηρεσίες που υλοποιούν τα πρότυπα της αρχιτεκτονικής REST στην οποία βασίζεται ολόκληρος ο παγκόσμιος ιστός. Μια υπηρεσία μπορεί να επικοινωνεί με έναν από τους δύο τρόπους ή και με τους δύο. Αυτό εναπόκειται στην ευχέρεια του προγραμματιστή και τις ανάγκες των πελατών του. Στην πτυχιακή αυτή θα υλοποιήσουμε και τους δύο τρόπους επικοινωνίας. Στις επόμενες δύο ενότητες εξηγούμε τα βασικά χαρακτηριστικά του πρωτοκόλλου SOAP και της αρχιτεκτονικής REST.

2.7 SOAP

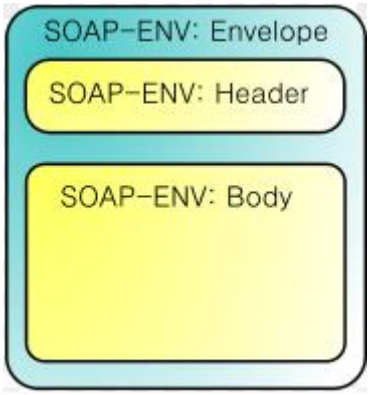
Το πρωτόκολλο Simple Object Access Protocol αποτελεί το πρότυπο με το οποίο αποκτούμε πρόσβαση και καλούμε μια web service. Αποτελεί πρόταση (recommendation) του W3C από το 2003. Βασίζεται στην XML, μορφοποιείται με αυτήν και έχει άμεση σχέση με τη WSDL. Χρησιμοποιεί το πρωτόκολλο HTTP ή το SMTP για τη μεταφορά των μηνυμάτων του. Ένα μήνυμα SOAP εσωκλείεται σε ένα πακέτο HTTP και αποστέλλεται στον παραλήπτη του.

Εν συντομία τα βασικά χαρακτηριστικά του SOAP είναι τα εξής:

- Είναι ένα πρωτόκολλο για την επικοινωνία με web services

- Ανήκει στο application layer του OSI
- Χρησιμοποιεί μηνύματα γραμμένα σε XML
- Χρησιμοποιεί το πρωτόκολλο HTTP ως φέρον
- Είναι ανεξάρτητο πλατφόρμας λογισμικού
- Είναι ανεξάρτητο γλώσσας προγραμματισμού
- Είναι απλό και επεκτάσιμο
- Αποτελεί πρότυπο του W3C

Η δομή του έχει ως εξής:
Το μήνυμα SOAP έχει (Σχήμα 2.2) ως στοιχείο ρίζα τον φάκελο (envelope), ο οποίος περιέχει μια κεφαλίδα (header) με μεταδεδομένα και ένα σώμα (body) με τις κλήσεις των μεθόδων μιας web service και τις παραμέτρους αν είναι μήνυμα αίτησης (request) ή την απάντηση από αυτήν αν είναι μήνυμα απάντησης (response message).



Σχήμα 2.2 Η δομή ενός μηνύματος SOAP

Η κεφαλίδα είναι ένα προαιρετικό στοιχείο και περιέχει πληροφορίες σχετικά με την εφαρμογή που θα εκτελεστεί, όπως κρυπτογράφηση, σύνδεση μέσω λογαριασμού (authentication).

Το σώμα μπορεί να περιέχει κλήση προς μεθόδους της web service αν το μήνυμα είναι αίτηση προς τη web service. Στο παρακάτω κομμάτι κώδικα έχουμε το μήνυμα αίτησης προς την data web service findstudents, το οποίο ζητά την κλήση της μεθόδου searchstudents (γραμμή 3). Η searchstudents δέχεται ως όρισμα ένα string και εμείς περνούμε σε αυτό το string chr (γραμμή 4). Στην πράξη, ζητούμε από την web service να μας εμφανίσει τα στοιχεία των φοιτητών που έχουν μέσα στο όνομά τους ή το επίθετό τους ή το username στον aeto, το string “chr”.

```
1. <soapenv:Envelope
   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:q0="http://it.teithe.findstudents"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
2. <soapenv:Body>
3.  <q0:SEARCHSTUDENTS>
4. <NAMEIN>chr</NAMEIN>
5. </q0:SEARCHSTUDENTS>
6. </soapenv:Body>
7. </soapenv:Envelope>
```

Το σώμα ενός μηνύματος SOAP μπορεί να περιέχει την απάντηση από την web service. Στο παρακάτω κομμάτι κώδικα βλέπουμε την απάντηση της data web service, η οποία μας επέστρεψε δύο rows με φοιτητές οι οποίοι έχουν το string “chr”.

```
1. <soapenv:Envelope
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">
2. <soapenv:Body>
3. <SEARCHSTUDENTSResponse xmlns:ns1="http://it.teithe.findstudents">
4.     <rowset>
5.         <row>
6.             <USERNAME>chrandr</USERNAME>
7.             <NAME>Christos</NAME>
8.             <SURNAME>Andrianos</SURNAME>
9.             <URL>http://www.it.teithe.gr/~chrandr</URL>
10.             <EMAIL>chrandr@it.teithe.gr</EMAIL>
11.         </row>
12.         <row>
13.             <USERNAME>chargir</USERNAME>
14.             <NAME>Christos</NAME>
15.             <SURNAME>Argiriadis</SURNAME>
16.             <URL>http://www.it.teithe.gr/~chargir</URL>
17.             <EMAIL>chargir@it.teithe.gr</EMAIL>
18.         </row>
19.     </rowset>
20. </SEARCHSTUDENTSResponse>
21. </soapenv:Body>
22. </soapenv:Envelope>
```

Παρατηρούμε ότι τόσο η αίτηση, όσο και η απάντηση είναι μορφοποιημένες σε XML, κάτι το οποίο μας βοηθά στην ανάλωσή του. Οι περισσότερες αντικειμενοστρεφείς γλώσσες προγραμματισμού (Java, C++, C#, Visual Basic, PHP, Ruby) έχουν υλοποιήσει κλάσεις προκειμένου να μπορούν να στέλνουν και να λαμβάνουν μηνύματα SOAP μέσω του HTTP πρωτοκόλλου.

2.8 REST

Μία εναλλακτική υλοποίηση έναντι του πρωτοκόλλου SOAP είναι η υλοποίηση των Restful web services. Η βασική έννοια στην αρχιτεκτονική REST (Representational State Transfer) είναι ο **πόρος**. Ένας πόρος είναι ένα οποιοδήποτε κομμάτι πληροφορίας, το οποίο μπορεί να προσδιοριστεί με μοναδικό τρόπο. Στην αρχιτεκτονική REST, οι αιτούντες και οι υπηρεσίες ανταλλάσσουν μηνύματα τα οποία μπορούν να έχουν τόσο δεδομένα όσο και μεταδεδομένα. Για να αναπαρασταθεί ένας πόρος με μοναδικό τρόπο χρησιμοποιούμε URIs. Για παράδειγμα ένας πόρος μπορεί να είναι ο http://www.go.gr/kilometers/count.

Οι υπηρεσίες που είναι υλοποιημένες με REST δεν διατηρούν την κατάσταση μιας αλληλεπίδρασης (stateless). Αυτό βοηθά τη γρήγορη ανάκαμψή τους σε περίπτωση σφάλματος. Σε περίπτωση που η υπηρεσία που βρίσκεται στο uri http://www.go.gr/kilometers/count παρουσιάσει σφάλμα ενώ την καλούμε, μπορούμε να την ξανακαλέσουμε χωρίς να μας περιορίσει κάποιος. Το γεγονός ότι

οι RESTful web services δεν διατηρούν την κατάσταση της αλληλεπίδρασης πελάτη – υπηρεσίας, μας υποχρεώνει να αποστέλλουμε ότι είναι απαραίτητο στην κάθε αίτηση προς αυτές. Κάθε μήνυμα πρέπει να είναι αυτοπεριγραφικό, δηλαδή να περιέχει όλες τις πληροφορίες που χρειάζεται η web service για να εκτελεστεί.

Τα βασικά χαρακτηριστικά των υπηρεσιών REST είναι τα εξής:

- Η web service είναι προσβάσιμη μέσω του πρωτοκόλλου HTTP με τις μεθόδους GET ή POST ή PUT ή DELETE.
- Κάθε web service έχει ένα μοναδικό base uri στο οποίο προστίθενται στο τέλος το όνομα της μεθόδου που θέλουμε να καλέσουμε, μαζί με τις παραμέτρους της.
- Κάθε web service υποστηρίζει δεδομένα τα οποία είναι κωδικοποιημένα με βάση ένα MIME type (JSON, XML, YAML ή άλλο).

Η χρήση των RESTful web services έχει αυξηθεί δραματικά εξαιτίας του γεγονότος ότι είναι απλές. Αρκεί μόνο το URL της υπηρεσίας και η γνώση των παραμέτρων που πρέπει να περαστούν σε αυτήν. Μια υπηρεσία με REST endpoint με μέθοδο HTTP GET μπορεί να κληθεί ακόμη και από έναν web browser. Η απλότητα της την έχουν κάνει τόσο δημοφιλή ώστε μεγάλες εταιρίες έχουν δημιουργήσει REST endpoints για τις ήδη υπάρχουσες web services που υποστήριζαν μόνο SOAP.

Η Google έχει αναπτύξει το AJAX API το οποίο είναι μια RESTful υλοποίηση των προϊόντων της. Μέσω αυτού μπορούμε να χρησιμοποιήσουμε υπηρεσίες όπως το Google Search, Google Maps, Google Translate, Google Ajax Feed και άλλες.

Η Yahoo έχει αναπτύξει το αντίστοιχο Yahoo API, το οποίο παρέχει υπηρεσίες για το Yahoo Search και άλλες. Η Yahoo έχει αναπτύξει την YQL (Yahoo Query Language), μια γλώσσα που μοιάζει με SQL και εκτελεί ερωτήματα σε web services. Κάθε ερώτημα της YQL μπορεί να αναπαρασταθεί με ένα URL και να κληθεί ως RESTful web service από οποιονδήποτε. Για παράδειγμα αν θέλουμε να λάβουμε σε JSON format τα πρώτα 10 αποτελέσματα για τον όρο «cat» από το Flickr αρκεί το url με το ερώτημα YQL: http://query.yahooapis.com/v1/public/yql?q=select%20*%20from%20flickr.photos.search%20where%20text%3D%22Cat%22%20limit%2010&format=json&diagnostics=true&callback=cbfunc. Η Yahoo επίσης παρέχει υποστήριξη σε όσους developers θέλουν να χρησιμοποιήσουν τις Web Services από διάφορες πλατφόρμες (Java, .NET, JavaScript, Ruby, Silverlight, PHP, Python) της μέσω άρθρων στο Yahoo Developer Network (YDN).

Η Amazon αντίστοιχα έχει αναπτύξει τις Amazon Web Services για το Amazon Cloud. Σύμφωνα με τον Jeff Barr ο οποίος είναι senior manager στις Amazon Web Services, η χρήση των υπηρεσιών της εταιρίας γίνεται κατά 85% από τις REST interfaces.

Στην εργασία αυτή θα δημιουργήσουμε data web services οι οποίες έχουν SOAP και REST interfaces ή endpoints και θα δείξουμε την ευκολία με την οποία καλείται μια REST service. Εντούτοις, όταν χρειαζόμαστε να συμπεριλάβουμε ασφάλεια

(κρυπτογράφηση) ή θέματα ποιότητας (quality of service) σε μια υπηρεσία, αυτό μπορεί να επιτευχθεί καλύτερα μέσω του SOAP.

2.9 WSDL

Η WSDL (web service description language) αποτελεί την γλώσσα περιγραφής των χαρακτηριστικών μιας web service. Επίσης Είναι μια γλώσσα περιγραφής σε XML και τα έγγραφα που παράγονται είναι αρχεία XML με κατάληξη .wsdl.

Σε γενικές γραμμές το WSDL είναι:

- WSDL σημαίνει web service description language
- Γράφεται σε XML
- Ένα .wsdl αρχείο είναι ένα XML αρχείο
- Χρησιμοποιείται για να περιγράψει web services.
- Χρησιμοποιείται επίσης για τον εντοπισμό μιας web service
- Η WSDL αποτελεί πρόταση (recommendation) του W3C

Όπως αναφέρεται και παραπάνω, ένα έγγραφο wsdl χρησιμοποιείται για τον εντοπισμό της web service και περιέχει τις functions ή methods που εκθέτει στον παγκόσμιο ιστό.

2.9.1 Η δομή του εγγράφου WSDL

Το έγγραφο wsdl αποτελείται από τεσσάρων ειδών tags τα οποία περιγράφουν την web service ως προς τη συμπεριφορά της και την τοποθεσία όπου μπορούμε να τη βρούμε.

- <types>: Οι τύποι δεδομένων που χρησιμοποιούνται για είσοδο και έξοδο από την web service.
- <message>: περιγραφή των μηνυμάτων
- <portType>: οι λειτουργίες (methods or functions) που υλοποιούνται από web service.
- <binding>: τα πρωτόκολλα επικοινωνίας που χρησιμοποιούνται

Θα αναλύσουμε τα παραπάνω συστατικά του εγγράφου στις επόμενες ενότητες χρησιμοποιώντας ένα πραγματικό αρχείο wsdl που προκύπτει από την web service που θα δημιουργήσουμε στο κεφάλαιο 3 (Δημιουργία DWS με τη χρήση του Data Studio). Το έγγραφο αυτό αφορά στην υπηρεσία findstudents η οποία έχει μόνο μία μέθοδο την SEARCHSTUDENTS η οποία αναζητά φοιτητές στη βάση δεδομένων μέσω της παραμέτρου NAMEIN τύπου string (κλειδί αναζήτησης).

2.9.2 Σήμανση types

Σε αυτό το xml tag συμπεριλαμβάνεται ένα xml schema στο οποίο περιγράφονται οι τύποι δεδομένων που θα χρησιμοποιηθούν για την είσοδο και την έξοδο. Στο

συγκεκριμένο παράδειγμα (το κομμάτι xml που ακολουθεί) έχουμε την μέθοδο SEARCHSTUDENTS η οποία παίρνει ως παράμετρο την NAMEIN η οποία είναι τύπου xsd:string και μπορεί να είναι ίση με null (nillable=true). Η μέθοδος αυτή είναι η stored procedure που έχουμε μέσα στη βάση και χρησιμοποιείται από την data web service μας.

Η απάντηση της μεθόδου είναι η SEARCHSTUDENTSResponse και θα περιέχει ένα anonymousResultSetType με μία ή περισσότερες γραμμές (row) οι οποίες περιέχουν τις στήλες που ταιριάζουν στο select ερώτημα που τρέχει η stored procedure μας.

```
<wsdl:types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://it.teithe.findstudents">
      <complexType name="anonymousResultSetType">
        <sequence>
          <element maxOccurs="unbounded" minOccurs="0" name="row">
            <complexType>
              <sequence maxOccurs="unbounded" minOccurs="0">
                <any processContents="skip"/>
              </sequence>
            </complexType>
          </element>
        </sequence>
      </complexType>
      <element name="SEARCHSTUDENTS">
        <complexType>
          <sequence>
            <element name="NAMEIN" nillable="true" type="xsd:string"/>
          </sequence>
        </complexType>
      </element>
      <element name="SEARCHSTUDENTSResponse">
        <complexType>
          <sequence>
            <element maxOccurs="1" minOccurs="0" name="rowset"
type="tns:anonymousResultSetType"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </wsdl:types>
```

2.9.3 Σήμανση message

Σε αυτό το xml tag περιγράφονται τα μηνύματα που ανταλλάσσουν τα endpoints της web service. Παρατηρούμε ότι έχουμε τέσσερεις τύπους μηνυμάτων:

- ***searchstudentsSoapInput***: το μήνυμα εισόδου για την κλήση της μεθόδου searchstudents μέσω πρωτοκόλλου SOAP. Παρατηρούμε ότι ως είσοδο-αίτηση (request) δέχεται τον τύπο SEARCHSTUDENTS όπως έχει δηλωθεί στις σημάνεις types παραπάνω.
- ***searchstudentsSoapOutput***: το μήνυμα απάντησης που αποστέλλει στον πελάτη η μέθοδος searchstudents μέσω πρωτοκόλλου SOAP. Η απάντηση παράγει τον τύπο δεδομένων SEARCHSTUDENTSResponse όπως έχει δηλωθεί πιο πάνω.

- ***searchstudentsGetPostUrlInput***: το μήνυμα εισόδου για την κλήση της searchstudents μέσω REST HTTP GET ή POST μεθόδου. Παρατηρούμε ότι δέχεται ως είσοδο-αίτηση (request) την παράμετρο NAMEIN που είναι τύπου xsd:string
- ***searchstudentsGetPostUrlOutput***: το μήνυμα απάντησης για την κλήση της searchstudents μέσω REST HTTP GET ή POST μεθόδου. Η απάντηση παράγει τον τύπο δεδομένων SEARCHSTUDENTSResponse όπως έχει δηλωθεί πιο πάνω.

```
<wsdl:message name="SEARCHSTUDENTSSoapInput">
  <wsdl:part element="tns:SEARCHSTUDENTS" name="request" />
    </wsdl:message>
<wsdl:message name="SEARCHSTUDENTSSoapOutput">
  <wsdl:part element="tns:SEARCHSTUDENTSResponse" name="response" />
    </wsdl:message>
<wsdl:message name="SEARCHSTUDENTSGetPostUrlInput">
  <wsdl:part name="NAMEIN" type="xsd:string" />
    </wsdl:message>
<wsdl:message name="SEARCHSTUDENTSGetPostOutput">
  <wsdl:part element="tns:SEARCHSTUDENTSResponse" name="response" />
    </wsdl:message>
```

2.9.4 Σήμανση portType

Σε αυτό το xml tag περιγράφονται οι μέθοδοι ή οι functions της web service. Κάθε υπηρεσία ιστού έχει μια σειρά από μεθόδους όπως ορίζονται στην Java, .NET και κάθε άλλη αντικειμενοστρεφή γλώσσα προγραμματισμού οι οποίες δέχονται καμία, μία ή περισσότερες παραμέτρους και επιστρέφουν κάποια δεδομένα. Στην περίπτωσή μας η μέθοδος της web service είναι μία, η SEARCHSTUDENTS, η οποία είναι μια stored procedure που μας επιστρέφει αποτελέσματα από τη βάση δεδομένων. Παρατηρούμε ότι έχουμε δύο portTypes ανάλογα με το πρωτόκολλο επικοινωνίας που χρησιμοποιείται (SOAP ή REST):

- ***findstudents***: η port που αφορά την επικοινωνία μέσω SOAP δηλώνει μόνο μια μέθοδο την SEARCHSTUDENTS, η οποία έχει ως μήνυμα εισόδου το searchstudentsSoapInput και μήνυμα απάντησης το searchstudentsSoapOutput. Τα δύο αυτά μηνύματα έχουν εξηγηθεί στην προηγούμενη ενότητα.
- ***findstudentsGetPostUrl*** η port που αφορά την επικοινωνία μέσω REST HTTP δηλώνει μόνο μια μέθοδο την SEARCHSTUDENTS, η οποία έχει ως μήνυμα εισόδου το searchstudentsGetPostUrlInput και μήνυμα απάντησης το searchstudentsGetPostUrl. Τα δύο αυτά μηνύματα έχουν εξηγηθεί στην προηγούμενη ενότητα.

```
<wsdl:portType name="findstudents">
    <wsdl:operation name="SEARCHSTUDENTS">
          <wsdl:input message="tns:SEARCHSTUDENTSSoapInput" />
          <wsdl:output message="tns:SEARCHSTUDENTSSoapOutput" />
    </wsdl:operation>
  </wsdl:portType>
<wsdl:portType name="findstudentsGetPostUrl">
    <wsdl:operation name="SEARCHSTUDENTS">
```

```
        <wsdl:input message="tns:SEARCHSTUDENTSGetPostUrlInput" />
        <wsdl:output message="tns:SEARCHSTUDENTSGetPostOutput" />
    </wsdl:operation>
  </wsdl:portType>
```

2.9.5 Σήμανση binding

Σε αυτό το xml tag γίνεται η σύνδεση του endpoint (SOAP ή REST) με τις μεθόδους του όπως έχουν ορισθεί στα portTypes και με τα πρωτόκολλα επικοινωνίας που θα χρησιμοποιηθούν.

```
<wsdl:binding name="findstudentsSOAP" type="tns:findstudents">
  <soap:binding style="document"
     transport="http://schemas.xmlsoap.org/soap/http" />
<wsdl:operation name="SEARCHSTUDENTS">
  <soap:operation soapAction="http://it.teithe.findstudents/SEARCHSTUDENTS" />
<wsdl:input>
  <soap:body use="literal" />
     </wsdl:input>
<wsdl:output>
     </wsdl:operation>
     </wsdl:binding>
```

2.9.6 Σήμανση service

Σε αυτό το xml tag δίνεται η πραγματική θέση του κάθε endpoint της web service μέσα στον παγκόσμιο ιστό. Όπως φαίνεται στο παρακάτω κομμάτι κώδικα, για να καλέσουμε την μέθοδο SEARCHSTUDENTS μέσω SOAP πρέπει να στείλουμε ένα μήνυμα στη διεύθυνση http://localhost:8080/Project1findstudents/services/findstudents.

Αν θέλουμε να χρησιμοποιήσουμε την REST HTTP GET έκδοση της υπηρεσίας ψάχνοντας κάποιον που το όνομά του, ή το επίθετο ή το username του περιέχει του string “chr” μπορούμε να το κάνουμε γράφοντας http://localhost:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=chr

```
<wsdl:service name="findstudents">
<wsdl:port binding="tns:findstudentsSOAP" name="findstudentsSOAPHTTP">
  <soap:address
     location="http://localhost:8080/Project1findstudents/services/findstudents"
     />
     </wsdl:port>
<wsdl:port binding="tns:findstudentsHTTPGET" name="findstudentsHTTPGET">
  <http:address
     location="http://localhost:8080/Project1findstudents/rest/findstudents/" />
     </wsdl:port>
<wsdl:port binding="tns:findstudentsHTTPPOST" name="findstudentsHTTPPOST">
  <http:address
     location="http://localhost:8080/Project1findstudents/rest/findstudents/" />
```

```
</wsdl:port>
</wsdl:service>
```

2.10 XML

Η γλώσσα Extensible Markup Language (XML) είναι ένα πρότυπο για τη δόμηση και την ανταλλαγή δεδομένων μέσω ετερογενών συσκευών, πλατφορμών και λειτουργικών συστημάτων. Αποτελεί πρότυπο του W3C και χρησιμοποιείται ευρέως με κύρια χρήση στον παγκόσμιο ιστό. Δημιουργήθηκε με σκοπό να παρέχει έναν κοινό τρόπο ανταλλαγής δεδομένων μεταξύ διαφορετικών συσκευών που βασίζονται σε διαφορετικά λειτουργικά συστήματα. Η XML ανεπτύχθη με βάση την SGML.

Οι web services χρησιμοποιούν την XML για τη μεταφορά δεδομένων. Το πρωτόκολλο SOAP είναι ένα πρωτόκολλο που βασίζεται και δομείται με XML. Οι REST style web services χρησιμοποιούν την XML ή τη JSON για την μορφοποίηση των δεδομένων εντός του HTTP πακέτου.

2.10.1 Το ιεραρχικό μοντέλο της XML

Η γλώσσα XML χρησιμοποιείται για να περιγράψει τη **δομή** που έχουν τα δεδομένα και κατ' επέκτασιν τα δεδομένα που περιέχονται σε έγγραφα. Δεν έχει καμία σχέση με τον τρόπο αναπαράστασης ενός εγγράφου, το οποίο είναι δουλειά HTML. Η ευρεία χρήση της έχει οδηγήσει στην αποθήκευση XML εγγράφων μέσα σε βάσεις δεδομένων και τη χρήση ερωτημάτων πάνω σε αυτά μέσω της γλώσσας αιτημάτων XQuery.

".. Βασική μονάδα δόμησης ενός XML εγγράφου είναι το element (στοχείο) το οποίο αποτελείται από ετικέτες (tags) και οι ιδιότητες - γνωρίσματα (attributes). Οι ετικέτες μπορούν να έχουν οποιοδήποτε όνομα θέλουμε. Με αυτόν τον τρόπο μπορούμε να δημιουργήσουμε όποια ετικέτα θέλουμε, γι' αυτό και η XML χαρακτηρίζεται ως **αυτοπεριγραφόμενη**. .." [5] Στον αντίποδα, η HTML χρησιμοποιεί μόνο έναν πεπερασμένο αριθμό ετικετών που έχουν οριστεί από το W3C.

- Κάθε στοιχείο της XML αποτελείται από μια ετικέτα αρχής, τις ιδιότητές της (αν υπάρχουν) και την ετικέτα τέλους.
- Τα ονόματα των ετικετών αρχής περιβάλλονται από γωνιακές αγκύλες (π.χ. <ptyxiaki>)
- Τα ονόματα των ετικετών τέλους περιβάλλονται από γωνιακές αγκύλες και μια slash (π.χ. </ptyxiaki>)
- Τα σχόλια περιβάλλονται από τα σύμβολα <!--και --> όπως και στην HTML.
- Τα πολύπλοκα στοιχεία κατασκευάζονται από άλλα στοιχεία, ιεραρχικώ τω τρόπω.
- Τα απλά στοιχεία περιέχουν τιμές δεδομένων (αριθμοί,strings,Boolean values κλπ)

- Στην XML τα ονόματα των ετικετών περιγράφουν τη σημασία των δεδομένων (π.χ. <book>), ενώ στην HTML κάθε στοιχείο περιγράφει τη μορφή εμφάνισης του κειμένου.
- Το μοντέλο που υλοποιεί η XML ονομάζεται **ιεραρχικό** ή **δενδρικό**.

Η XML έχει τα εξής χαρακτηριστικά:

1. Είναι case-sensitive γι' αυτό η ετικέτα αρχής και τέλους πρέπει να γραμμένες ακριβώς με τον ίδιο τρόπο. Το <book> είναι διαφορετική ετικέτα από το <Book>.
2. Οι τιμές κειμένου των στοιχείων (strings) δεν περικλείονται από εισαγωγικά.
3. Οι τιμές των ιδιοτήτων (attribute) περικλείονται από μονά (π.χ. 'velos') ή διπλά εισαγωγικά (π.χ. "velos")

2.10.2 Ορθώς ορισμένο XML έγγραφο

Ένα ορθώς ορισμένο (well formed) XML θα πρέπει να ακολουθεί τις ακόλουθες συνθήκες:

- Να αναφέρει στην πρώτη γραμμή την έκδοση της XML που ακολουθεί.
- Να ξεκαθαρίζει με το γνώρισμα standalone αν το έγγραφο ακολουθεί κάποιο schema ή όχι.
  - standalone="yes" δεν ακολουθεί κάποιο σχήμα
  - standalone="no" ακολουθεί συγκεκριμένο σχήμα
- Θα πρέπει να ακολουθεί το συντακτικό του δενδρικού μοντέλου:
  - Θα πρέπει να υπάρχει ένα στοιχείο (element) ρίζα
  - Κάθε στοιχείο να περιλαμβάνει ένα ζεύγος ετικετών αρχής και τέλους
  - Κάθε στοιχείο να περικλείεται μέσα στις ετικέτες του κόμβου γονέα.
- Δεν υπάρχει προκαθορισμένο σύνολο στοιχείων.

2.10.3 Έγκυρο XML έγγραφο

Ένα XML έγγραφο είναι έγκυρο (valid) όταν:

- Είναι ορθώς ορισμένο (well formed)
- Τα ονόματα των στοιχείων του ακολουθούν κάποιο XML schema.

2.10.4 XML schema

Ένα XML schema (σχήμα) είναι ένα πρότυπο για τον προσδιορισμό της δομής των XML εγγράφων. Κάθε σχήμα είναι γραμμένο σε XML και αποτελεί ένα έγγραφο XML. Κάθε μήνυμα αίτησης ή απάντησης του πρωτοκόλλου SOAP και αρχείο WSDL ακολουθεί ένα schema τα οποίο έχει οριστεί από το W3C.

2.11 JSON

Η JSON ή αλλιώς JavaScript Object Notation αποτελεί ένα «ελαφρύ» πρότυπο ανταλλαγής δεδομένων μεταξύ γλωσσών προγραμματισμού. Βασίζεται σε ένα υποσύνολο της γλώσσας JavaScript σύμφωνα με το πρότυπο ECMA-262 3rd Edition - December 1999. Ο δημιουργός της JSON είναι ο Douglas Crockford, ο οποίος είναι μηχανικός πληροφορικής στη Yahoo και ιδρυτής της ιστοσελίδας json.org

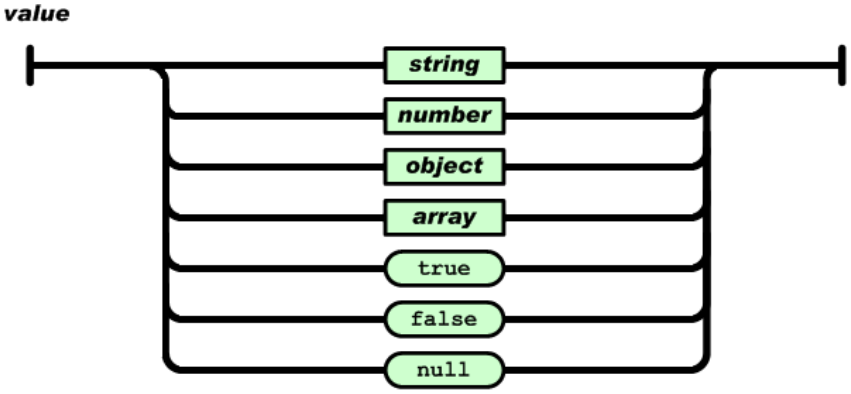
Με την εξάπλωση της χρήσης του παγκοσμίου ιστού και την εμφάνιση του AJAX και των Web Services η χρήση της JSON έχει αυξηθεί ραγδαία και αποτελεί μια πιο ελαφριά (lightweight) εναλλακτική για την XML. Ένα αρχείο με δεδομένα μορφοποιημένα σε JSON είναι πιο μικρό σε σχέση με ένα αντιστοιχο XML, εξαιτίας του γεγονότος ότι τα δεδομένα σε JSON δεν περικλείονται από tags.

Η Yahoo την υιοθέτησε το 2005 για την ανάκτηση πληροφοριών στις Web Services της που παρέχονται από το Yahoo! API. Ακολούθησε η Google το 2006 η οποία το χρησιμοποιεί στα διάφορα APIs του Google Code.

Στην πράξη η JSON είναι μια μορφή κειμένου για την σειριοποίηση (serialization) δομημένων δεδομένων.
Η JSON ορίζει ένα μικρό σύνολο κανόνων μορφοποίησης για την αναπαράσταση δομημένων δεδομένων. Σύμφωνα με το RFC 4627 μπορούν να αναπαρασταθούν τέσσερις βασικοί τύποι δεδομένων (strings, Boolean, null, numbers) και δύο δομημένοι τύποι δεδομένων (αντικείμενα και πίνακες).

2.11.1 Values

Οι βασικοί τύποι αποκαλούνται values (τιμές) και μπορούν να πάρουν τις ακόλουθες μορφές όπως αυτές αναπαρίστανται στο Σχήμα 2.3



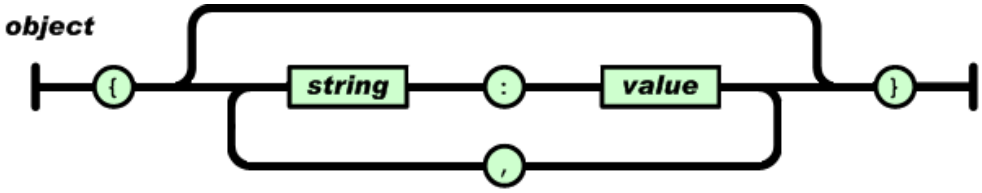
Σχήμα 2.3 Οι δυνατές τιμές των βασικών τύπων σε JSON

Ειδικά για τα strings αυτά πρέπει να περικλείονται από διπλά εισαγωγικά " και μπορούν να έχουν ως τιμή οποιονδήποτε χαρακτήρα που περιλαμβάνεται στο πρότυπο Unicode εκτός από τους χαρακτήρες " (διπλά εισαγωγικά) ή \

(backslash). Με τη χρήση της backslash μπορούν να αναπαρασταθούν ειδικοί χαρακτήρες όπως το backspace (\b) ή το carriage return (\r).

2.11.2 Object

Ένα αντικείμενο είναι ένα σύνολο από ζευγάρια ονομάτων (names) και τιμών (values) . Περικλείεται από άγκιστρα { και }. Κάθε όνομα ακολουθείται από τον χαρακτήτα **:** (διπλή τελεία) και την τιμή του (value). Οι τιμές που μπορεί να δεχτεί είναι αυτές που έχουν οριστεί για τους βασικούς τύπους δεδομένων (values). Κάθε ζεύγος ονόματος/τιμής διαχωρίζεται με κόμμα. Η γενική μορφή του αντικειμένου δίνεται στο Σχήμα 2.4



Σχήμα 2.4 Η μορφή του αντικειμένου σε JSON

Τα αντικείμενα υλοποιούνται χωρίς να υφίσταται κάποια κλάση που τα ορίζει, όπως είναι συνηθισμένο στις γλώσσες C++ και Java. Απλώς τα υλοποιούμε με τη δομή και το περιεχόμενο που θέλουμε και στην περίπτωση της Javascript τα αναθέτουμε σε μία μεταβλητή έτσι ώστε να μπορούμε να τα προσπελάσουμε.

Για παράδειγμα αν θέλουμε να γράψουμε ένα αντικείμενο που αναπαριστά ένα πρόσωπο (Person) μπορούμε να επιλέξουμε την ακόλουθη μορφή:

```
var ob1 = {
        “name“: “Christos”,
        “age“: 14,
        “height“: 1.75
}
```

Για να προσπελάσουμε ή να αλλάξουμε την ιδιότητα name του αντικειμένου ob1, αρκεί να γράψουμε σε Javascript: ob1**.**name

Ένα αντικείμενο μπορεί να είναι σύνθετο και να περιλαμβάνει άλλα αντικείμενα όπως για παράδειγμα

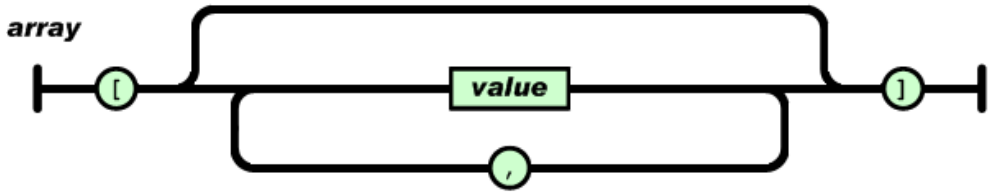
```
var ob2 = {
        “name“: “Christos”,
        “age“: 14,
        “height“: 1.75
        “address“: {
                “road“: “Papadopoulou”
                “number“: 45
                “city“: “Thessaloniki”
        }
```

}

Για να προσπελάσουμε ή να αλλάξουμε την ιδιότητα number της διεύθυνσης address του αντικειμένου ob2, αρκεί να γράψουμε σε Javascript: ob2**.**address.number = 12

2.11.3 Arrays

Οι πίνακες είναι δομές δεδομένων που αναπαρίστανται από συλλογές τιμών (values) ή αντικειμένων. Ένας πίνακας ξεκινά με αριστερή αγκύλη [και τελειώνει με δεξιά αγκύλη]. Κάθε τιμή διαχωρίζεται με κόμμα **,** από μία άλλη (Σχήμα 2.5).

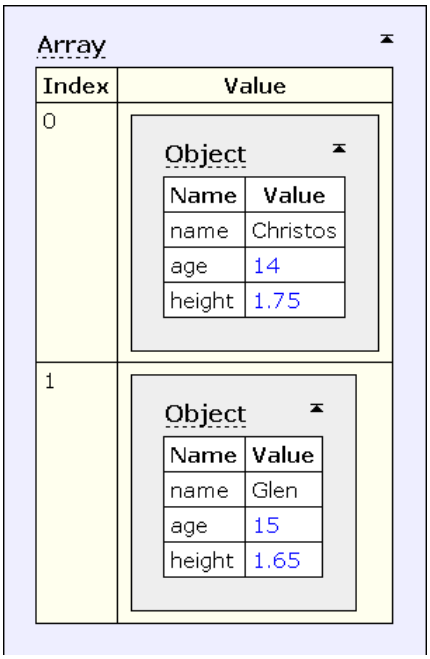


Σχήμα 2.5 Η γενική μορφή ενός πίνακα στη JSON

Η JSON επιτρέπει να έχουμε πίνακες τιμών,πίνακες αντικειμένων και μεικτούς πίνακες (με τιμές και αντικείμενα). Δεν υπάρχει περιορισμός στον τύπο των τιμών και των αντικειμένων. Ένας πίνακας μπορεί να περιέχει δεδομένα διαφορετικού τύπου.

```
Var arr1 = [2,3,4,5,6]; //πίνακας ακεραίων

Var arr2 = [“chris”,”nick”,”volotnikov”] //πίνακας strings

Var arr3 = [3,4,5, “chris”, [12,34,5.4], 3] //πίνακας δεδομένων διαφορετικών τύπων

//πίνακας αντικειμένων
var ob1 = [ {
        “name”: “Christos”,
        “age”: 14,
        “height”: 1.75
},
{
        “name“: “Glen”,
        “age“: 15,
        “height“: 1.65
}]
```

Ο παραπάνω πίνακας αντικειμένων μπορεί να αναπαρίσταθεί γραφικά στο Σχήμα 2.6 Για την οπτικοποίηση χρησιμοποιήθηκε το εργαλείο json2html (http://json.bloople.net/#_output).



Σχήμα 2.6 Γραφική αναπαράσταση πίνακα αντικειμένων σε JSON

2.11.4 JSONP

Η JSONP (json with padding) είναι μια επέκταση στο ήδη υπάρχον πρότυπο. Δημιουργήθηκε από τον Bob Ippolito το 2005 και δεν αποτελεί πρότυπο όπως η JSON αλλά τεχνική. Βασική της λειτουργία είναι ότι επιτρέπει ένα client side script (javascript) που έχει φορτωθεί από μια σελίδα να ζητά (request) δεδομένα σε μορφή JSON από έναν server διαφορετικό από αυτόν που είναι αποθηκευμένη η σελίδα.

Οι web browsers ακολουθούν την πολιτική ασφαλείας **same origin policy**. Σύμφωνα με αυτήν ένα client side script που βρίσκεται σε μια σελίδα μπορεί να ανακτά ή να μεταχειρίζεται μέσω AJAX δεδομένα μόνο από σελίδες και scripts που βρίσκονται στο ίδιο domain. Έτσι αν το script βρίσκεται στο domain example1.com, δεν μπορεί να ανακτήσει δεδομένα από ιστοσελίδες πού βρίσκονται στο domain example2.com. Το θέμα αυτό περιορίζει τους προγραμματιστές που θέλουν να λάβουν δεδομένα από Web Services ή ιστοσελίδες τρίτων με τη χρήση AJAX.

Έχουν χρησιμοποιηθεί διάφοροι τρόποι για την λύση του θέματος αλλά εμπεριέχουν περιορισμούς. Το W3C έχει εκδώσει ένα draft paper στον οποίο προτείνει τον μηχανισμό Cross-Origin Resource Sharing το οποίο όμως δεν έχει ακόμη τελειοποιηθεί για να αποτελέσει πρότυπο.

Προς το παρόν η επικρατούσα λύση είναι η δυναμική φόρτωση script (dynamic script insertion) μιας και εξαιρούνται από την πολιτική ασφαλείας τα html tags <script>. Με αυτόν τον τρόπο μπορούμε να εισάγουμε δυναμικά σε ένα <script> tag, δεδομένα σε μορφή JSONP από οποιοδήποτε domain θέλουμε.

Το JSONP είναι δεδομένα μορφοποιημένα σε JSON που περιβάλλονται από παρενθέσεις ( ) και αποτελούν ορίσματα μιας συνάρτησης επιστροφής (callback function). Έτσι τα δεδομένα φορτώνονται από το client script δυναμικά σαν κλήση της συνάρτησης που έχουμε ορίσει ως callback με όρισμα τα δεδομένα που αιτούμαστε από το δικό μας ή ένα ξένο domain.

Με τον όρο συνάρτηση callback εννοούμε τη συνάρτηση η οποία θα κληθεί όταν τα δεδομένα τα οποία αιτηθήκαμε φτάσουν στον client. Τα δεδομένα αυτά θα περάσουν ως ορίσματα στην συνάρτηση αυτή, από όπου μπορούμε να τα επεξεργαστούμε και να τα μετασχηματίσουμε κατά το δοκούν.

Έτσι αν μέσα στον κώδικά μας θέλουμε να καλέσουμε μια web service μέσω Javascript με μέθοδο HTTP GET (XMLHTTPRequest object) που δέχεται ως όρισμα ένα όνομα και επιστρέφει αντικείμενα που έχουν αυτό το όνομα σε μορφοποίηση JSON π.χ.

{ “name“: “Christos”, “age“: 14}

και έστω ότι το URL της web service είναι http://www.testws/getobject . Το url πέρνει ως παραμέτρους το callback και το όνομα namein που ψάχνουμε. Για να βρούμε όλα τα αντικείμενα με όνομα christos θα πρέπει στο url της web service να ορίσουμε ένα όνομα για την callback function π.χ. ptyxiaki και το κλειδί αναζήτησης π.χ. namein=christos.

Το url θα είναι το http://www.testws/getobject?callback=ptyxiaki&namein και η απάντηση θα έχει τη μορφή:

ptyxiaki ({ “name“: “Christos”, “age“: 14, “height“: 1.75 })

Ο πλήρης κώδικας με τη χρήση της JQuery θα έχει την εξής μορφή, όπου ως callback βάζουμε τον χαρακτήρα ?:

```
jQuery.getJSON(http://www.testws/getobject&callback=?namein=Christos",
function(data) {
    alert("Name: " + data.name + ", Age: " + data.age);
});
```

Η απάντηση από τη web service θα μας εμφανισθεί μέσω ενός μηνύματος στον browser και θα είναι αυτή που απεικονίζεται στο Σχήμα 2.7:

Σχήμα 2.7 Η απάντηση της web service σε μήνυμα

2.11.5 Θέματα ασφαλείας

Η JSONP μας βοηθά να καλέσουμε REST web services εύκολα και γρήγορα, αλλά έχει «τρύπες» ασφαλείας που την καθιστούν ευάλωτη σε επιθέσεις.

Το γεγονός ότι χρησιμοποιεί το tag <script> και δεν σέβεται την πολιτική same origin policy την καθιστά ευάλωτη σε επιθέσεις τύπου cross site request forgery (CSRF). Έτσι μια κακόβουλη σελίδα μπορεί να ζητήσει δεδομένα μέσω JSONP από μια άλλη σελίδα και να αποκαλύψει ευαίσθητα δεδομένα που μπορεί να έχει αποθηκευμένα ο χρήστης στην κληθείσα δεύτερη σελίδα. Για αυτόν το λόγο συστήνεται προσοχή στη χρήση της JSONP και κρίνεται καλό να μην χρησιμοποιείται για τη μεταφορά ευαίσθητων δεδομένων.

2.12 Επίλογος

Στο κεφάλαιο αυτό κάναμε μια σύντομη εισαγωγή στις υπηρεσίες ιστού ή web services. Επίσης αναλύσαμε τις τεχνολογίες και τα πρωτόκολλα που χρησιμοποιούνται για την επιτυχή δημιουργία, δημοσίευση και ανάλωση μιας web service. Η μικρή δόση θεωρίας θα μας βοηθήσει να κατανοήσουμε τη δημιουργία μιας Data Web Service στο 3^{ο} Κεφάλαιο.

3. Data Web Services (DWS)

3.1 Εισαγωγή

Οι Web Services αποτελούν έναν δοκιμασμένο πια μηχανισμό για την πρόσβαση, διαχείριση και ανταλλαγή δεδομένων. Τα πρότυπα που χρησιμοποιούνται με τα θετικά στοιχεία που εμπεριέχουν όπως η χαμηλή σύζευξη (loose coupling) και το virtualization δείχνουν το μέλλον που είναι η Αρχιτεκτονική Υπηρεσιών Ιστού (Service Oriented Architecture). Κάποιες μελέτες δείχνουν ότι προκύπτουν δυσκολίες στην υλοποίηση επειδή πολλά SOA projects καθυστερούν ή αποτυγχάνουν.

Πολλές Web Services προσανατολισμένες στα δεδομένα (data oriented) έχουν μια μοναδική λειτουργία: Εκτελούν ένα ερώτημα σε μια βάση δεδομένων ή καλούν μια stored procedure το οποίο ωθεί στο να ενσωματώνουμε όλες τις συνδιαλλαγές με τη βάση δεδομένων μέσα στη λογική του κώδικα του προγράμματος. Έτσι η υλοποίηση μιας εφαρμογής ιστού προσανατολισμένης στα δεδομένα αποτελεί πρόκληση, επειδή απαιτεί γνώσεις πάνω στις βάσεις δεδομένων αλλά και σε Web προγραμματισμό.

Οι προγραμματιστές που δημιουργούν data oriented εφαρμογές παραδέχονται ότι η διαδικασία δημιουργία τους περικλείει πολλά DML (data manipulation language) statements τα οποία πρέπει να τοποθετηθούν μέσα στον κώδικα. Επιπλέον η δημιουργία μιας σύνδεσης με τη βάση, η εκτέλεση ενός DML, DDL ή SQL statement, η επεξεργασία του συνόλου των αποτελεσμάτων (resultset) και ο τερματισμός της σύνδεσης αποτελεί μια επαναληπτική διαδικασία η οποία καταλαμβάνει πολλές γραμμές κώδικα και απαιτεί ιδιαίτερη προσοχή. Θεμιτό είναι να μπορούμε να χρησιμοποιούμε τις λειτουργίες της βάσης δεδομένων και να προβάλλουμε τα δεδομένα τους στο Web χωρίς να μας ταλαιπωρεί η δομή του προγράμματος που θα το υλοποιήσει. Για να επιτευχθεί αυτός ο στόχος, η IBM προτείνει τις Data Web Services.

3.2 Ορισμός

Γενικά ο όρος Data Web Service δεν έχει επίσημο ορισμό και εμφανίζεται και σαν Web Data Service. Παραθέτουμε τους ορισμούς της Microsoft και της IBM που έχουν αναπτύξει εργαλεία για να δώσουν πρόσβαση μέσω Web σε δεδομένα που είναι αποθηκευμένα στα DBMS τους. Εμείς επειδή ασχολούμαστε με τα προϊόντα της IBM θα δώσουμε περισσότερη σημασία στη δική της εκδοχή.

Σύμφωνα με τη Microsoft οι Data Services είναι ο τρόπος πρόσβασης, διαχείρισης και μεταβολής δεδομένων, αποθηκευμένων σε βάσεις δεδομένων μέσω web services που υλοποιούν το πρωτόκολλο OData (open data). Μέσω αυτού του πρωτοκόλλου δίνεται η δυνατότητα στα προϊόντα της Microsoft να έχουν πρόσβαση σε δομημένα δεδομένα μέσω REST HTTP web services που υλοποιούν το OData. Αξίζει να σημειωθεί ότι η Microsoft χρησιμοποιεί κατά κύριο

λόγο το πρωτόκολλο SOAP στις Web Services της και οι Data Services έρχονται να συμπληρώσουν το κενό που υπήρχε όσον αφορά στις REST.

Σύμφωνα με την IBM «.. οι Data Web Services (DWS) είναι ο νέος τρόπος υλοποίησης (development), ανάπτυξης (deployment) και διαχείρισης (management) της πρόσβασης μέσω web services σε δεδομένα που είναι αποθηκευμένα σε DB2 και IDS database servers ..» [16] .

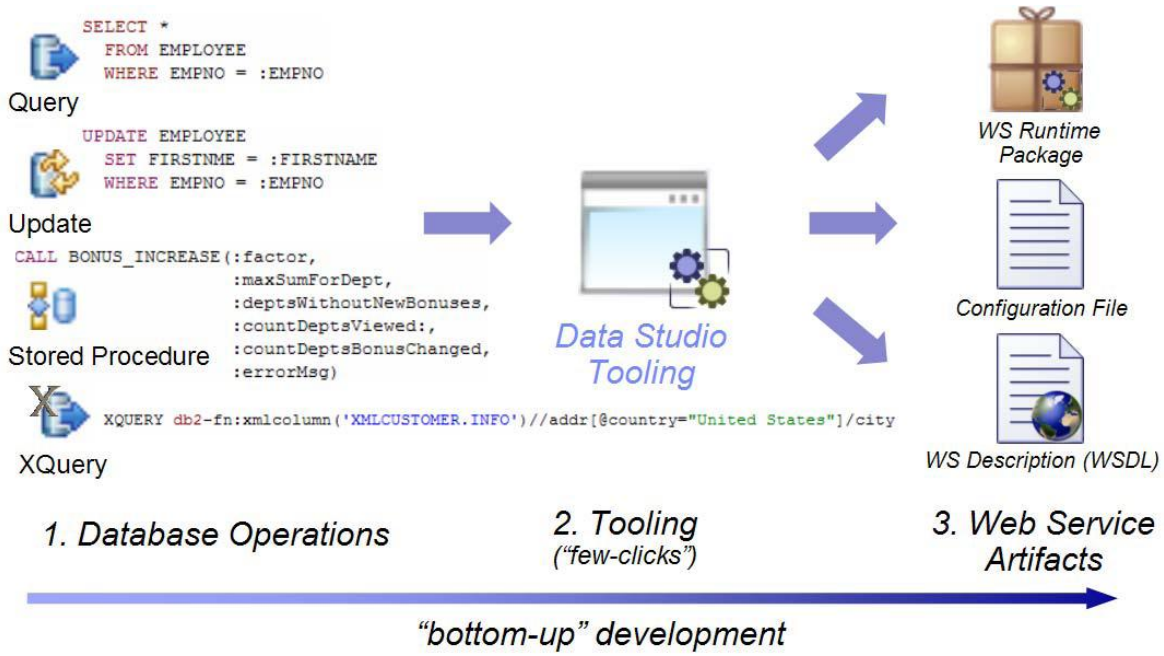
Αυτό σημαίνει ότι μπορούμε να δημιουργήσουμε web services χωρίς να γράψουμε γραμμή κώδικα, έχοντας απλά μόνο τα DML statements (select,insert,update,delete,XQuery) ή καλώντας τις stored procedures που θέλουμε στη βάση δεδομένων. Οι DWS μας επιτρέπουν να δημιουργήσουμε web services οι οποίες μέσω SOAP αλλά και REST HTTP. Οι παραγόμενες web services είναι συσκευασμένες σε μια ready-to-deploy μορφή και μπορούν να γίνουν deployed σε υποστηριζόμενους application servers. Οι δυνατότητες αυτές παρέχονται από το user interface του IBM Data Studio IDE.

Τα βασικά στοιχεία των IBM DWS είναι τα εξής:

- Δεν χρειάζεται καθόλου προγραμματισμός
  - Με τη χρήση drag-drop interface μπορούμε εύκολα να σύρουμε DML statements ή stored procedure σε ένα web service container το οποίο είναι ready-to-deploy σε application server.
  - Το Data Studio παρέχει περιβάλλον με το οποίο μπορούν να δοκιμαστούν οι δημιουργηθείσες DWS.
- Οι DWS υποστηρίζουν το SOAP over HTTP και δημιουργούν WSDL αρχεία
  - Το Data Studio δημιουργεί αυτόματα WSDL αρχείο για κάθε SOAP web service που περιέχει την περιγραφή της
- Οι DWS υποστηρίζουν REST-styled service interface
  - Παράλληλα με το SOAP υποστηρίζονται και HTTP GET/POST bindings για την υποστήριξη REST web services.
- Οι DWS μπορούν να μετασχηματίσουν στον server τα δεδομένα των web services μέσω XSLT
  - Οι DWS επιτρέπουν μετασχηματισμούς XSLT στα δεδομένα πριν φύγουν από τον server (server side) ανάλογα με τη μορφοποίηση που θέλουμε να έχουν.
- Οι DWS δεν παράγουν κώδικα αλλά μια απλή J2EE εφαρμογή που μπορεί να γίνει deployed σε έναν application server [16]

Η δημιουργία των Data Web Services απαιτεί τρία διακριτά βήματα (Σχήμα 3.1)

1. Συγγραφή των DML statements, XQuery, stored procedures που θα προσπελαύνουν ή τροποποιούν ή μετασχηματίζουν τα δεδομένα στη βάση δεδομένων.
2. Δημιουργία των data web services με λίγα κλικ και λειτουργίες drag and drop.
3. Deployment application server, δοκιμή και χρήση.



Σχήμα 3.1 Βήματα δημιουργίας των data web services

3.3 Η αρχιτεκτονική των IBM DWS

Όπως προαναφέρθηκε, οι Data Web Services υποστηρίζουν SOAP και REST υλοποιήσεις ως προς τον τρόπο αίτησης (request) και ανάκτησης της υπηρεσίας. Η αρχιτεκτονική θα μπορούσε να χωριστεί σε 3 επίπεδα από πάνω προς τα κάτω όπως φαίνεται στο σχήμα 3.2:

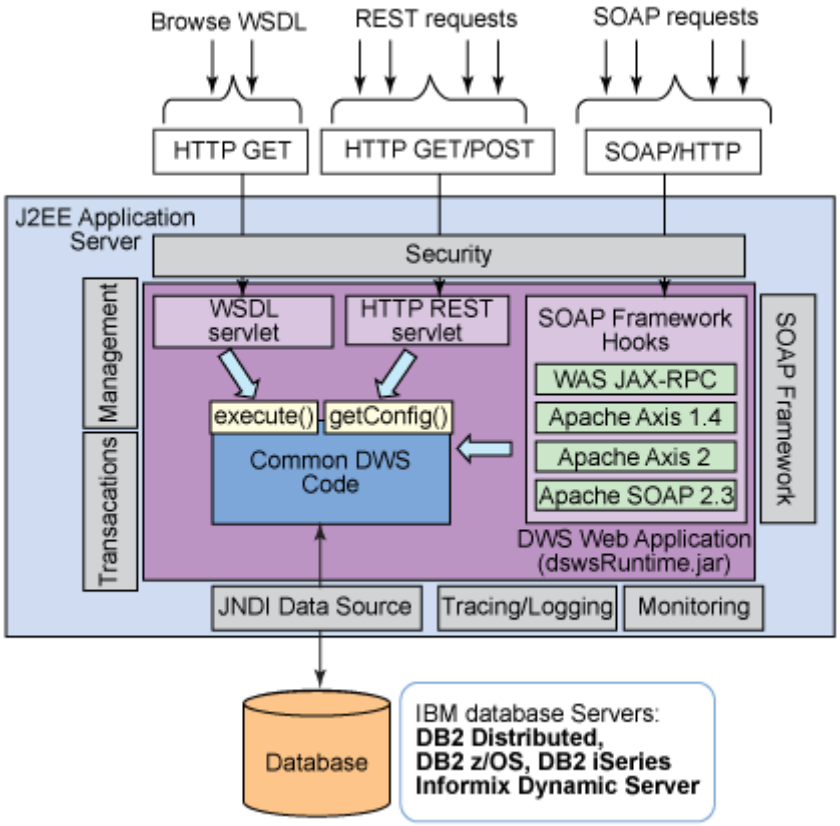
1. Αιτήματα web service που έρχονται από τον έξω κόσμο στους υποδοχείς αιτημάτων του application server
2. Ο J2EE application server όπου γίνεται η επεξεργασία των μηνυμάτων και η προετοιμασία τους για τη βάση
3. Η βάση δεδομένων από όπου ο application server θα τροφοδοτηθεί με δεδομένα για να απαντήσεις στις αιτήσεις των web services.

Δεν πρέπει να παραλείψουμε ότι για να λειτουργήσουν τα παραπάνω απαιτείται να υπάρχει εγκατεστημένο το Java Runtime Environment που θα υποστηρίζεται από τον application server και το dbms. Το runtime environment μπορεί να είναι το JRE της Oracle Sun ή το JRE της IBM αρκεί να υποστηρίζει τις λειτουργίες του application server.

Στο ανώτερο επίπεδο βρίσκονται οι αιτήσεις που γίνονται από διάφορους πελάτες (browsers, applications, web servers) προς τους υποδοχείς (endpoints) της data web service. Κάθε τύπος αίτησης εξυπηρετείται από κώδικα ο οποίος είναι συγκεκριμένος για κάθε τύπο (WSDL servlet, HTTP REST servlet, SOAP framework hooks). Έτσι κάθε φορά που έρχεται μια αίτηση web service υφίσταται επεξεργασία και παράγεται μια κοινή αναπαράσταση της αίτησης. Αυτό σημαίνει ότι η μια αίτηση web service που ζητά τον Α πόρο από τη βάση δεδομένων, θα έχει ίδια αναπαράσταση μέσα στον application server άσχετα με το αν εκλήθη μέσω SOAP ή REST.

Κατόπιν στο δεύτερο επίπεδο (J2EE application server) , η κοινή αναπαράσταση περνά από ένα common metadata driven runtime το οποίο αποφασίζει για το πώς θα χαρτογραφηθεί (map) το αίτημα της web service σε αίτημα βάσης δεδομένων. Οι παράμετροι των αιτήσεων (αν υπάρχουν) περνούν στην εφαρμογή και μετατρέπονται σε τύπους δεδομένων (data type) της DB2 ή του εκάστοτε database server. Δεν παρεμβάλλεται mapping σε τύπους δεδομένων της Java. Οι XML τύποι δεδομένων του μηνύματος της web service μετατρέπονται απευθείας σε data types της βάσης δεδομένων. Έτσι ένας xs:int μετατρέπεται απευθείας σε SQL Integer. Κατόπιν το αίτημα μεταφράζεται σε αίτημα βάσης δεδομένων και αποστέλλεται στο DBMS. Με το απευθείας mapping σε τύπους της βάσης δεδομένων χωρίς ενδιάμεσες μετατροπές πετυχαίνουμε την γρήγορη και αποτελεσματική λειτουργία της εφαρμογής που τρέχει μέσα στον application server.

Τέλος στο τρίτο και κατώτερο επίπεδο βρίσκεται ο database server. Το αίτημα αποστέλλεται εκεί, η απάντηση από τη βάση δεδομένων επιστρέφει στον application server, μορφοποιείται σε XML και αποστέλλεται πίσω στον αιτούντα client. Η απάντηση μπορεί να μορφοποιηθεί και σε JSON αν η αίτηση είναι REST HTTP GET και στο url της web service προστεθεί η παράμετρος _outputFormat=JSON. Σε επόμενο κεφάλαιο θα δούμε τον τρόπο χρήσης του υποδοχέα REST με μέθοδο HTTP GET και JSON output.

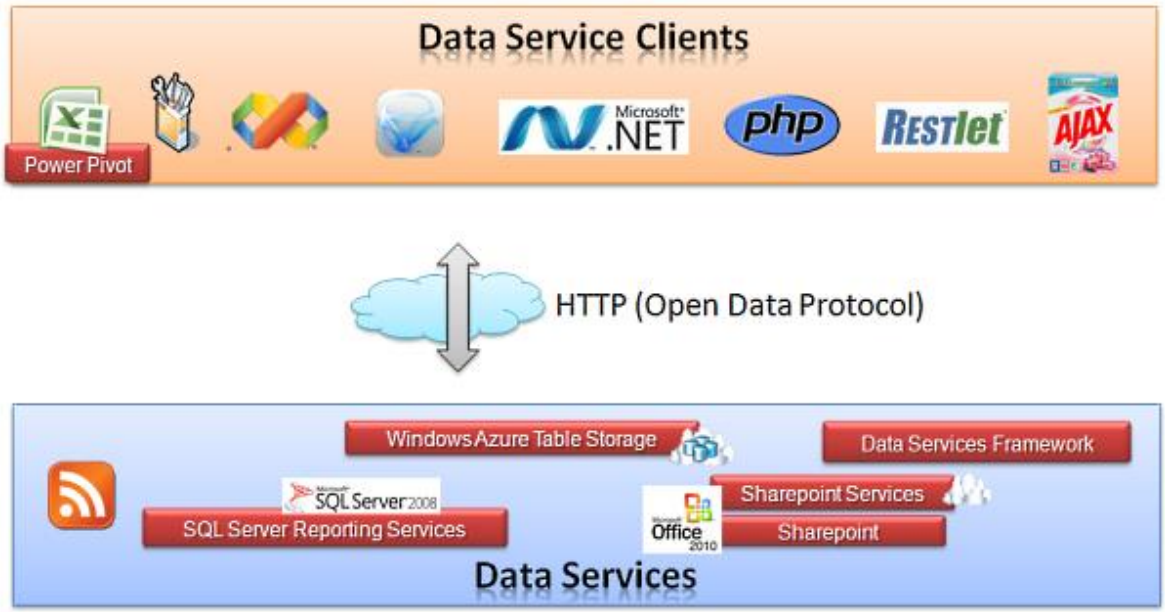


Σχήμα 3.2 Η αρχιτεκτονική δομή των IBM Data Web Services

3.4 Η αρχιτεκτονική των Microsoft Data Services

Η Microsoft ακολουθεί λίγο διαφορετική αρχιτεκτονική σε σχέση με την IBM. Παρέχει το WCF (Windows Community Foundation) Data Services framework, το οποίο αποτελείται από ένα σύνολο βιβλιοθηκών, μέσω των οποίων διάφορες εφαρμογές και ιστοσελίδες μπορούν να λαμβάνουν δεδομένα από βάσεις δεδομένων και άλλους πόρους μέσω REST web services που υλοποιούν το πρωτόκολλο OData.

Σε αντίθεση με την IBM, η Microsoft εκτός από την πρόσβαση σε βάση δεδομένων MS SQL server δίνει τη δυνατότητα πρόσβασης και σε δεδομένα που είναι αποθηκευμένα σε πλατφόρμες cloud computing, όπως το Azure καθώς και σε δεδομένα που διαχειρίζεται ο Sharepoint server. Επίσης όπως φαίνεται στο Σχήμα 3.3 οι data services μπορούν να χρησιμοποιηθούν από server side scripts ASP,PHP και client side JavaScript μέσω AJAX. Παράλληλα η πλατφόρμα με το Power Pivot παρέχεται η δυνατότητα άμεσης σύνδεσης του MS Excel με τις data services.



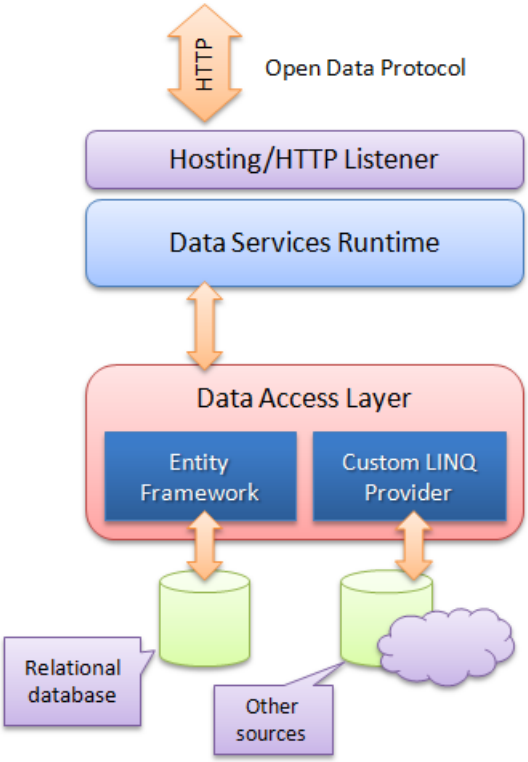
Σχήμα 3.3 Ο τρόπος σύνδεσης των MS Data Services με τον έξω κόσμο

Το Data Services framework αποτελεί μέρος του .NET framework και δεν αποτελεί ξεχωριστή οντότητα όπως ο application server για τις IBM Data Web Services. Διευκολύνει τη δημιουργία ευέλικτων data services που είναι άμεσα συνυφασμένες με το Web. Οι Data Services είναι REST style web services οι οποίες χρησιμοποιούν URIs για να προσπελάσουν τους πόρους και τα δεδομένα. Τα δεδομένα που επιστρέφονται ως απάντηση στις αιτήσεις είναι μορφοποιημένα με formats ευρέως χρησιμοποιούμενα όπως η JSON και το ATOM (XML based feed format).

Προκειμένου το σύστημα να προσθέσει και να διαχειριστεί τα σημασιολογικά μεταδεδομένα (semantics) επί των δεδομένων που επεξεργάζεται, οργανώνει τα δεδομένα με βάση του Entity Data Model (EDM) που αποτελεί παράγωγο του

Entity-Relationship Model (ER). Τα δεδομένα είναι οργανωμένα σε στιγμιότυπα τύπων οντοτήτων (instances of entity types or entities) και φέρουν σχέσεις μεταξύ τους.

Όσον αφορά σε δεδομένα σχεσιακών βάσεων δεδομένων το μοντέλο EDM δημιουργείται μέσω του ADO .NET Entity Framework και ενσωματώνεται στην data service. Το EDM υποστηρίζει και δεδομένα που δεν είναι οργανωμένα με το σχεσιακό μοντέλο και μπορούν να προέρχονται από οποιαδήποτε άλλη πηγή δεδομένων (Σχήμα 3.4)



Σχήμα 3.4 Η αρχιτεκτονική του WCF Data Services framework

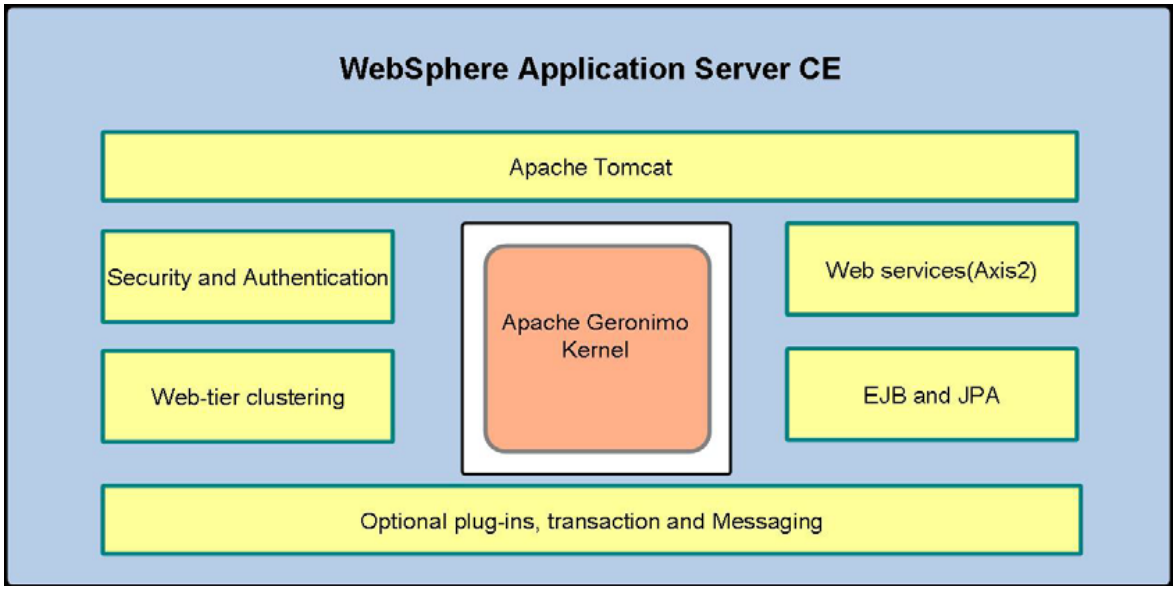
3.5 IBM WebSphere Application Server CE

Ο IBM WebSphere Application Server Community Edition είναι ο J2EE application server της ομώνυμης εταιρίας για δημιουργία Java Enterprise Edition εφαρμογών, Web services και EJB (Enterprise Java Beans). Είναι γραμμένος σε Java και αποτελεί τη βάση για πολλά προϊόντα της IBM όπως το ILOG BRMS (business rule management system). Εμείς θα τον χρησιμοποιήσουμε ως application server στον οποίο θα κάνουμε deploy τις data web services μας.

Η έκδοση Community Edition είναι δωρεάν και μπορεί να χρησιμοποιηθεί σε συνεργασία με τα IBM Data Studio IDE και τη βάση δεδομένων IBM DB2 9.7 Express C τα οποία είναι και αυτά δωρεάν κατόπιν registration στην IBM.
Μπορεί να συνεργαστεί και με άλλα DBMS όπως IBM Informix, MS SQL server, Oracle, MySql και PostgreSQL

Τα κύρια συστατικά του WebSphere είναι τρία όπως φαίνονται στο Σχήμα 3.5 :

- **Apache Geronimo Kernel**: το framework που παρέχει τη δυνατότητα δημιουργίας εφαρμογών στους χρηστές του server.
- **Developer components**: Συστατικά του server που παρέχουν στους developers εργαλεία για τη δημιουργία web services και την υλοποίηση ασφάλειας και ανταλλαγής μηνυμάτων (security and messaging) μεταξύ των εφαρμογών Java EE. Τα εργαλεία αυτά είναι δοκιμασμένα στην έκδοση Community Edition και δεν έχουν προβλήματα συμβατότητας με άλλες εκδόσεις του server.
- **Administrator console**: web κονσόλα διαχείρισης του server που είναι προσβάσιμη μέσω browser. Βοηθά τους προγραμματιστές να ορίσουν συνδέσεις για τον Tomcat (HTTP,HTTPS,AJP), database pools, να κάνουμε deploy/undeploy Java EE applications και web services και να ορίσουμε κανόνες ασφαλείας, χωρίς να επανεκκινήσουμε τον server ή να γράψουμε κώδικα. Η κονσόλα είναι προσβάσιμη αν πληκτρολογήσουμε τη διεύθυνση http://localhost:8443/console σε έναν οποιονδήποτε web browser.



Σχήμα 3.5 Τα κύρια συστατικά του WebSphere

Μια σύντομη περιγραφή των συστατικών του WebSphere παρουσιάζεται στον πίνακα 2.1

Components or Features	Description
Apache Geronimo Kernel	Java EE 5 Application server ανοικτού κώδικα που καθιστά το project ικανό για να είναι Community Edition
Apache Derby	Ενσωματωμένη βάση δεδομένων για μικρά και απλά projects. Για μεγαλύτερα και πολυπλοκότερα projects συστήνεται η IBM DB2
Apache Tomcat	Ένα web-tier container που

	χρησιμοποιείται από τις τεχνολογίες Java Servlet και Java server pages (JSP)
Apache OpenEJB	Ένα ενσωματωμένο ελαφρύ EJB 3.0 (enterprise java beans) framework
Apache OpenJPA	Υλοποίηση του Java Persistence API για την αποθήκευση Java Objects σε βάσεις δεδομένων
Apache Active MQ	Υλοποίηση του Java Messaging services
WebSphere Application Server Community Edition server adapter (Eclipse plug-in)	Plug-in για το Eclipse IDE για την εύκολη δημιουργία, αποσφαλμάτωση και deployment Java EE εφαρμογών
Built-in support for third part RDBMSs	Υποστήριξη για DB2, MS SQL server, Oracle, MySql
JVM support	Υποστήριξη για το IBM Java Virtual Machine και για το Oracle Sun JVM

Πίνακας 2.1 Σύντομη περιγραφή των υπομονάδων του WAS CE

3.6 IBM Data Studio IDE

Το Data Studio είναι ένα Integrated Development Environment (IDE) φτιαγμένο πάνω στο Eclipse IDE με το οποίο δημιουργούμε τις IBM Data Web Services.

Παρέχει τις εξής δυνατότητες:

- μπορούμε εύκολα και γρήγορα να συνδεθούμε και να διαχειριστούμε μία ή περισσότερες βάσεις δεδομένων τοπικά ή απομακρυσμένα. Η IBM το προτείνει ως εργαλείο διαχείρισης της βάσης δεδομένων DB2 μιας και θα σταματήσει να υποστηρίζει το DB2 Administration Control Panel.
- Δημιουργία data web services με λίγα βήματα και χωρίς συγγραφή κώδικα
- Δημιουργία Java EE εφαρμογών που γίνονται deploy στον WebSphere μέσω των plug-in που μας παρέχονται από τον server και ενσωματώνονται γρήγορα και ευκολα με τον Eclipse Plug-In manager

 Αξίζει να σημειώσουμε ότι το Data Studio IDE για να δουλέψει χωρίς να κολλά και να καθυστερεί χρειάζεται να χρησιμοποιήσει 1 GB RAM. Έτσι είναι καλό να εκτελείται σε υπολογιστές με μνήμη μεγαλύτερη του 1,5 GB RAM. Σε υπολογιστές που δεν πληρούν αυτές τις προϋποθέσεις και έχουν μικρότερη μνήμη θα παρατηρήσετε σημαντική καθυστέρηση στη λειτουργία του

3.7 Apache Web Server

Ο πιο γνωστός open source web server χρησιμοποιείται προκειμένου να παρέχουμε τη δυνατότητα στις REST web services να μπορούν να δημοσιεύουν τα δεδομένα τους με διαμόρφωση JSONP προκειμένου να μπορούν να κληθούν μέσω AJAX. Χρησιμοποιούμε την διανομή xampp των apache friends, όπου μέσα σε ένα πακέτο εγκαθίσταται ο Apache, η PHP, η βάση δεδομένων MySQL 5, τον FileZilla ftp server και τον Mercury mail server.

Ο WebSphere στις REST εκδόσεις των data web services μορφοποιεί τα εξαγόμενα δεδομένα σε XML ή JSON. Πολλές web services καλούνται από Javascript μέσω AJAX το οποίο περιορίζεται από την πολιτική ασφαλείας same origin policy των web browsers με αποτέλεσμα να μην μπορούν να κληθούν αν τα δεδομένα δεν είναι μορφοποιημένα σε JSONP.

3.8 IBM DB2 9.7 Express-C

Για τη δημιουργία των Data Web Services χρησιμοποιούμε ως πηγή δεδομένων την βάση δεδομένων DB2 που μας παρέχεται δωρεάν. Μέσα σε αυτή θα δημιουργήσουμε πίνακες δεδομένων και stored procedures οι οποίες χρησιμοποιούνται αργότερα από τις data web services για να μας φέρουν δεδομένα από τη βάση. Η βάση δεδομένων που χρησιμοποιούμε ονομάζεται Ptyxiaki και η κωδικοποίηση χαρακτήρων που χρησιμοποιεί είναι UTF-8.

3.9 Επίλογος

Σε αυτό το κεφάλαιο δώσαμε έναν σύντομο ορισμό στον όρο Data Web Services. Επίσης περιγράψαμε τις τεχνολογίες που εμπλέκονται και τα εργαλεία που θα χρησιμοποιήσουμε. Στο επόμενο κεφάλαιο θα δούμε πώς όλα αυτά θα συνεργαστούν αρμονικά για να δημιουργήσουν Data Web Services.

4. Δημιουργία Data Web Services με το Data Studio

4.1 Εισαγωγή

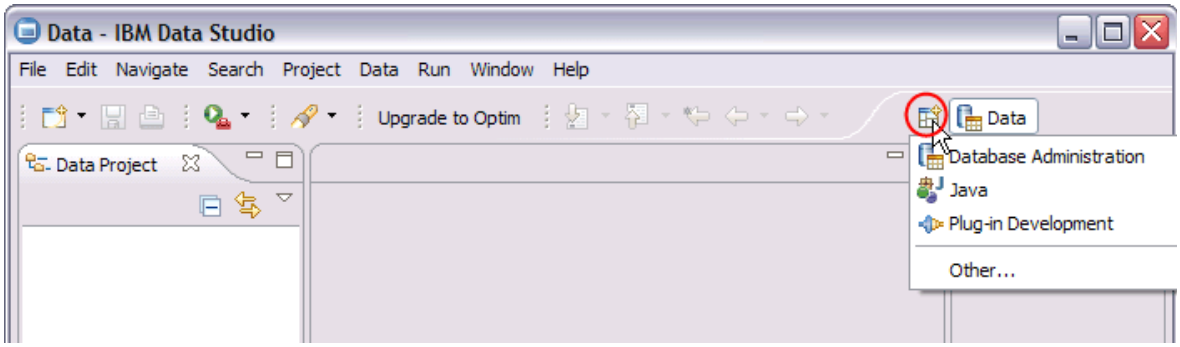
Στο κεφάλαιο αυτό θα δείξουμε τα βήματα με τα οποία δημιουργούμε εύκολα και γρήγορα Data Web Services με τη χρήση του Data Studio. Για να γίνουν πιο κατανοητά θα χρησιμοποιήσουμε ένα παράδειγμα που αφορά μια λίστα φοιτητών και μια web service που αναζητά φοιτητές με βάση κάποια κριτήρια όπως όνομα, επίθετο, email.

Στις ενότητες 2 και 3 περιγράφουμε το πρόβλημα που αντιμετωπίζουμε καθώς και το πώς καταγράφονται τα δεδομένα μέσα στη βάση δεδομένων DB2.
Στις ενότητες 4,5,6 κάνουμε μια μικρή εισαγωγή στο Data Studio και περιγράφουμε με απλά βήματα τις αρχικές ρυθμίσεις που πρέπει να γίνουν.
Στις ενότητες 7 έως 10 δημιουργούμε τη web service, την κάνουμε deploy στον WebSphere και τη δοκιμάζουμε.

4.2 Το περιβάλλον του Data Studio

Το περιβάλλον του Data Studio IDE περιέχει μία ή περισσότερες perspectives (προοπτικές). Κάθε perspective περιέχει views (όψεις) και εργαλεία με βάση το ρόλο ή την εργασία που έχει να κάνει. Έτσι αν θέλουμε να γράψουμε κώδικα Java επιλέγουμε το Java perspective, αν θέλουμε να διαχειριστούμε μια βάση δεδομένων, επιλέγουμε το Database Administration perspective. Το περιβάλλον του Data Studio διαφοροποιείται προκειμένου να μας βοηθήσει στην εργασία που αφορά στο επιλεχθέν perspective.

Εμείς θα χρησιμοποιήσουμε το Data perspective (Σχήμα 4.1) το οποίο μας παρέχει views για τη δημιουργία Data Development Projects μέσω των οποίων δημιουργούμε data web services,stored procedures, functions και πολλά άλλα.

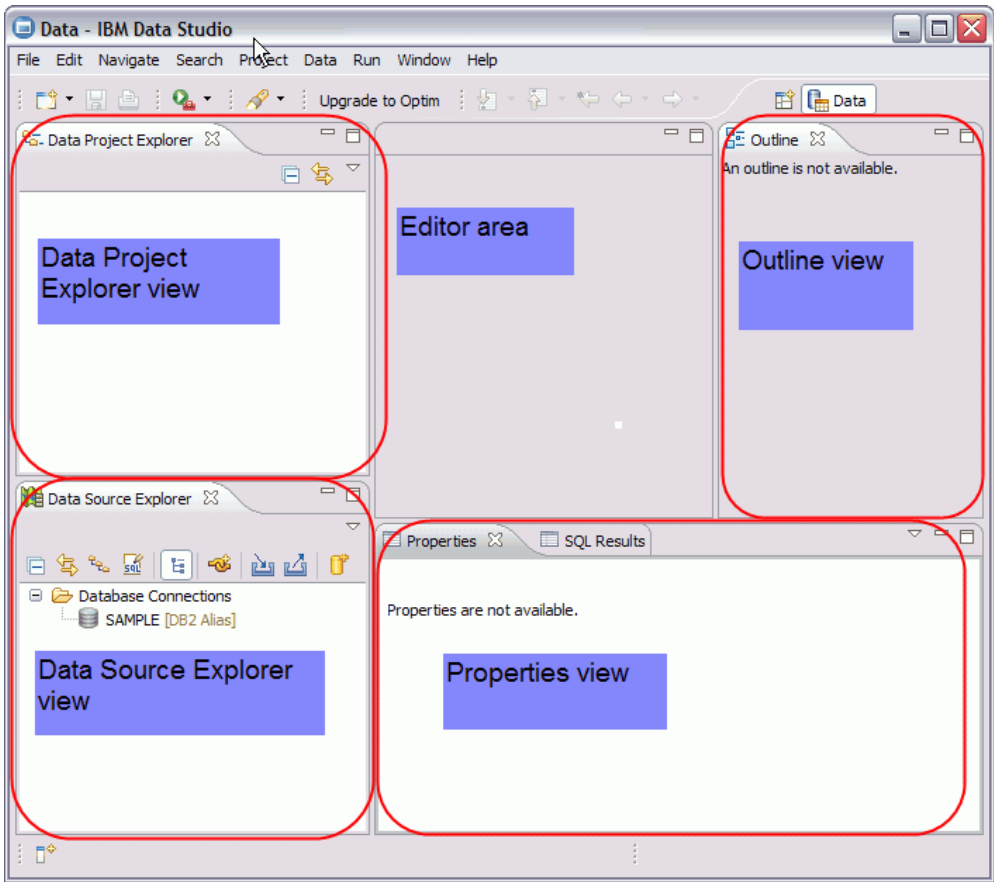


Σχήμα 4.1 Εναλλαγή perspectives στο Data Studio

Επιλέγοντας το Data Perspective το παράθυρο του Data Studio αποκτά τη μορφή του σχήματος 3.2. Σε αυτήν παρατηρούμε κάποια views. Με τον όρο view εννοούνται τα παράθυρα ή toolboxes τα οποία μας βοηθούν να προηγηθούμε στην ιεραρχία μιας πληροφορίας, να μεταβάλλουμε κάποιο έγγραφο ή να δούμε πληροφορίες σχετικά με κάποιο επιλεχθεν στοιχείο.

Στο Data Perspective παρατηρούμε 5 views:

- Data Project Explorer: δενδροειδής δομή που μας δείχνει τα ανοικτά projects και τα περιεχόμενά τους.
- Data Source Explorer: παράθυρο που μας βοηθά να διαχειριστούμε τις βάσεις δεδομένων που έχουμε δηλώσει. Μπορούμε να προσθέσουμε και να αφαιρέσουμε συνδέσεις με βάσεις δεδομένων. Το view αυτό είναι το ίδιο με το view του database administration perspective.
- Properties view: εμφανίζει τις ιδιότητες του αντικειμένου που έχει επιλεχθεί στον Data Project ή τον Data Source Explorer. Επίσης στα SQL results δείχνει τα αποτελέσματα των εκτελεσθέντων εντολών SQL.
- Outline View: περιέχει το περίγραμμα και τη δομή του τρέχοντος αρχείου. Για παράδειγμα αν έχουμε ανοικτό ένα αρχείο XML, μας εμφανίζει τη δομή του με τα elements και τα attributes του.
- Editor area: το view στο οποίο μπορούμε να επεξεργαστούμε stored procedures, να γράψουμε sql statements και γενικά να επεξεργαστούμε ότι έχει επιλεχθεί στον Data Project ή τον Data Source Explorer



Σχήμα 4.2 Τα views του Data perspective

4.3 Το παράδειγμα που θα χρησιμοποιήσουμε

4.3.1 Περιγραφή προβλήματος

Για να γίνει κατανοητός ο τρόπος δημιουργίας των data web services θα δημιουργήσουμε μία Data Web Service η οποία θα αναζητά φοιτητές με κλειδί αναζήτησης το όνομα, επίθετο και το username που έχει ο φοιτητής στον server aetos.

Χρησιμοποιώντας τη σελίδα που περιέχει τις Προσωπικές Ιστοσελίδες Φοιτητών http://aetos.it.teithe.gr/pagelist/students.html με regular expressions εξάγαμε για κάθε φοιτητή το όνομα, το επίθετο και το username στον aeto.

4.3.2 Η βάση δεδομένων

Η βάση δεδομένων πρέπει να υποστηρίζει ελληνικά και να μπορεί να θεωρεί δύο χαρακτήρες ίσους στην Ελληνική γλώσα, άσχετα αν είναι κεφαλαίοι ή μικροί, τονιζόμενοι ή μη. Δηλαδή θέλουμε το Χρήστος, χρηστος, χΡήστος να θεωρούνται ίσα. Για να το κάνουμε αυτό δημιουργούμε τη βάση δεδομένων με UTF-8 κωδικοποίηση με την εξής εντολή:

```
CREATE DATABASE ptyxiaki USING CODESET utf-8
    TERRITORY el-gr
    COLLATE USING UCA500R1_LEL_AS_CX_EO_FX_HX_NO_S1;
```

Ορίζουμε τον πίνακα students:

```
create table administrator.students (
    username varchar (30)  not null ,
    name varchar (50) ,
    surname varchar (50),
    url varchar (40),
    email varchar (40),
    constraint pk_students_username primary key ( username)  ) ;
```

Ορίζουμε και τον trigger studenti με τον οποίο γεμίζουμε τις στήλες url και email όταν κάνουμε insert μια νέα γραμμή.

```
--#set terminator /
create trigger administrator.studenti
after  insert  on administrator.students  referencing  new as nrow
for each row  mode db2sql
  begin atomic
  update administrator.students
  set email = username  || '@it.teithe.gr' ,
  url = 'http://www.it.teithe.gr/~' || username ;
end/
--#set terminator ;
```

Έτσι όταν κάνουμε insert έναν νέο φοιτητή αρκεί να εισάγουμε το όνομα, επίθετο και το username. Το email και το url της ιστοσελίδας του, δημιουργούνται από τον trigger μετά, με βάση το username που έχει δοθεί.

```
Insert into  administrator.students (name,surname,username) values ('Manios', 'Christos', 'chrman')
```

Παρατηρούμε ότι πριν και μετά τη δήλωση του trigger έχουμε εισάγει το ειδικό σχόλιο **--#set terminator /** .Με αυτό το ειδικό σχόλιο αλλάζουμε τον χαρακτήρα τερματισμού εντολής στην DB2.

Όταν έχουμε να τρέξουμε σε batch mode πολλαπλές εντολές που περιέχουν και δηλώσεις triggers ή stored procedures, χρειάζεται να αλλάζουμε τον χαρακτήρα τερματισμού πριν και μετά τις δηλώσεις. Οι δηλώσεις των triggers και των stored procedures στην DB2 χρειάζονται διαφορετικό χαρακτήρα τερματισμού στο σώμα και διαφορετικό στην εξωτερική δήλωση.

Αυτή η ιδιομορφία της DB2 **δυστυχώς** δεν παρουσιάζεται πουθενά στο online documentation παρά μόνο σε μια σελίδα χωρίς να δοθεί ιδιαίτερη σημασία (http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/r0010410.htm) με αποτέλεσμα να χαθεί πολύς χρόνος αναζήτησης στο internet για να καταφέρουμε να τρέξουμε εντολές σε batch mode.

Άλλα συστήματα βάσεων δεδομένων όπως η MySQL και ο MS SQL server χρησιμοποιούν τον ίδιο terminator για όλες τις εντολές.

Τέλος παραθέτουμε και τον κώδικα της stored procedure την οποία θα δημιουργήσουμε αργότερα μέσα στο data studio.

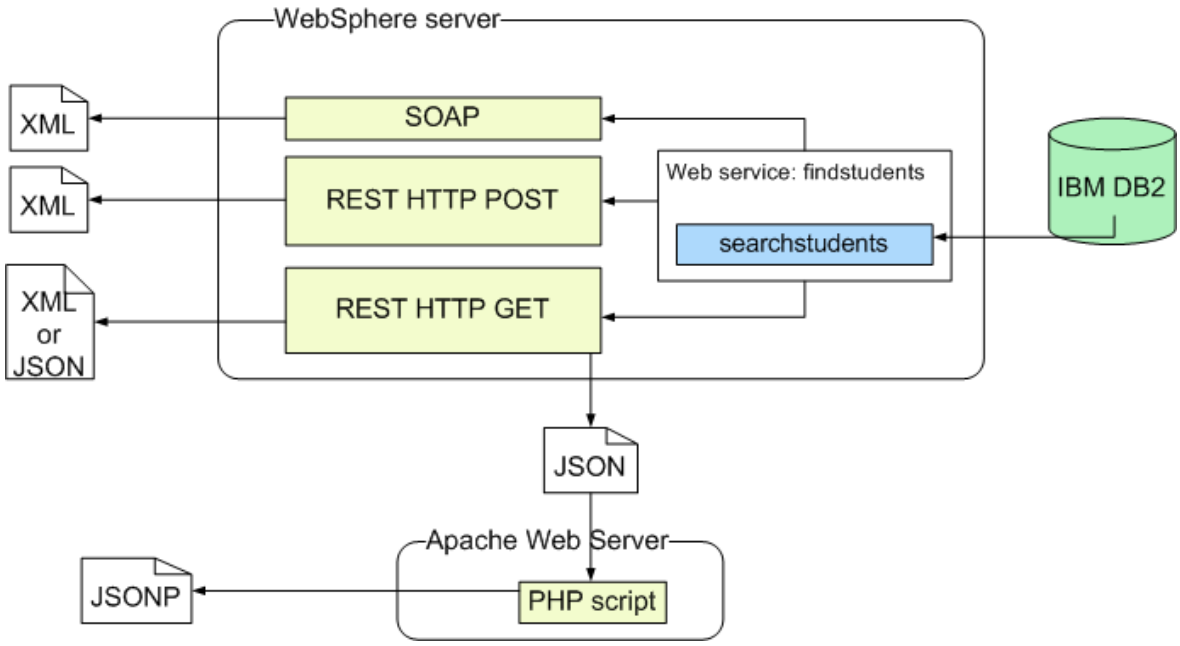
```
--#SET TERMINATOR ;
create PROCEDURE SEARCHSTUDENTS (IN NAMEIN VARCHAR(50))
    DYNAMIC RESULT SETS 1
P1: BEGIN
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        select *
        from students
        where lower(name) like '%' || lower(namein) || '%'
        or lower(surname) like '%' || lower(namein) || '%'
        or lower(username) like '%' || lower(namein) || '%';

    -- Cursor left open for client application
    OPEN cursor1;
END P1/
--#SET TERMINATOR ;
```

4.3.3 Η δομή της Web Service

Η data web service που θα δημιουργήσουμε θα ονομάζεται findstudents και θα έχει μια μοναδική μέθοδο η οποία αντιστοιχίζεται στην stored procedure searchstudents. Η web service θα εξάγει τα δεδομένα μέσω SOAP και REST HTTP.

Όπως φαίνεται στο Σχήμα 4.3 το πρωτόκολλο SOAP, REST HTTP POST και GET μορφοποιούν την απάντηση σε XML και μόνο για τη REST HTTP GET μπορούμε να έχουμε μορφοποίηση σε JSON. Εμείς επειδή θέλουμε να καλέσουμε τη web service μέσω AJAX από το JQuery framework χρειαζόμαστε JSONP. Η μορφοποίηση αυτή όμως δεν δημιουργείται αυτόματα από την υπάρχουσα έκδοση του Data Studio.

Για να μορφοποιήσουμε την απάντηση της web service σε JSONP καλούμε την REST HTTP GET έκδοση με απάντηση σε μορφοποίηση σε JSON από μια PHP σελίδα και τη μετατρέπουμε σε JSONP προσθέτοντας το padding. Έτσι αν θέλουμε να καλέσουμε την web service και να μας επιστρέψει JSONP θα καλέσουμε την ιστοσελίδα searchstudents.php με παραμέτρους το κλειδί αναζήτησης (namein) και το όνομα της callback function.

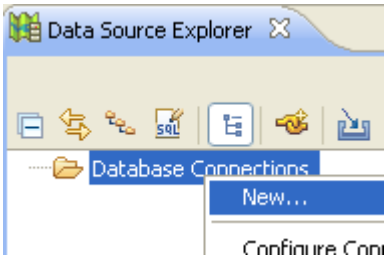


Σχήμα 4.3 Η δομή της web service searchstudents

4.4 Σύνδεση με τη βάση δεδομένων

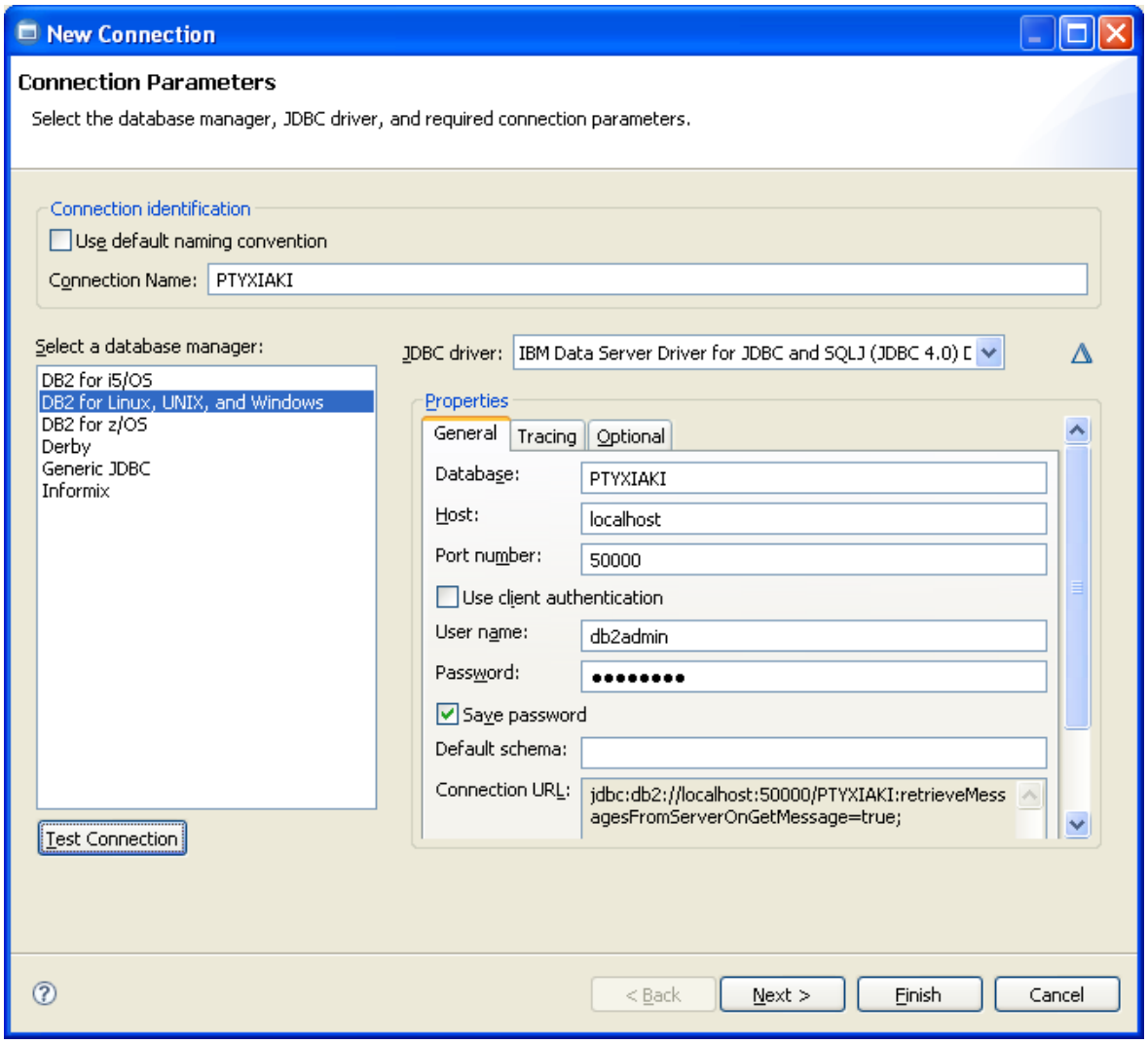
Για να δημιουργήσουμε data web services πρέπει πρώτα να είμαστε συνδεδεμένοι με τη βάση δεδομένων από όπου θα αντλούμε ή θα εισάγουμε δεδομένα. Για να δημιουργήσουμε μια νέα σύνδεση ακολουθούμε τα παρακάτω βήματα:

1. Στον *Data Sourse Explorer* κάνουμε δεξί κλικ στο Database connections και επιλέγουμε New.. (Σχήμα 4.4)



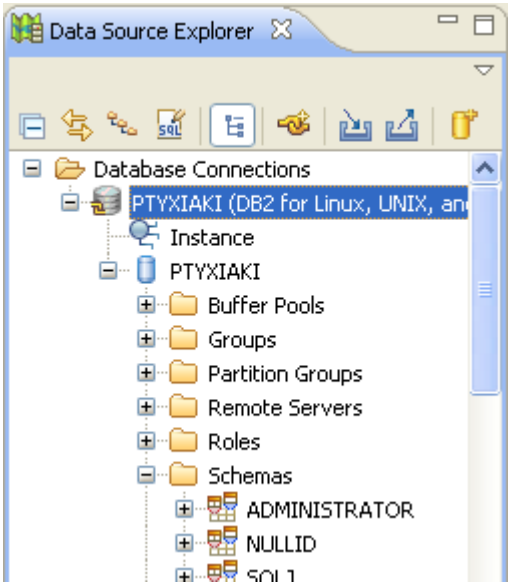
Σχήμα 4.4 New database connection menu

2. Στο παράθυρο που μας εμφανίζεται (Σχήμα 4.5) εισάγουμε ένα όνομα της αρεσκείας μας για τη σύνδεση (Connection name), επιλέγουμε ως database manager την DB2 for Linux/Unix and Windows. Εισάγουμε το όνομα της βάσης που θέλουμε να χρησιμοποιήσουμε καθώς και σε ποιον host και σε ποια θύρα (port number) ακούει. Τέλος εισάγουμε όνομα και κωδικό χρήστη. Αν θέλουμε να επιβεβαιώσουμε ότι τα δεδομένα είναι σωστά επιλέγουμε κάτω δεξιά το πλήκτρο *Test Connection*. Τέλος πατούμε το *Finish*.



Σχήμα 4.5 Παράμετροι σύνδεσης βάσης δεδομένων

3. Η διαδικασία έχει ολοκληρωθεί. Στον *Data Sourse Explorer* (Σχήμα 4.6) μπορούμε να δούμε τη βάση μας και να τη διαχειριστούμε.

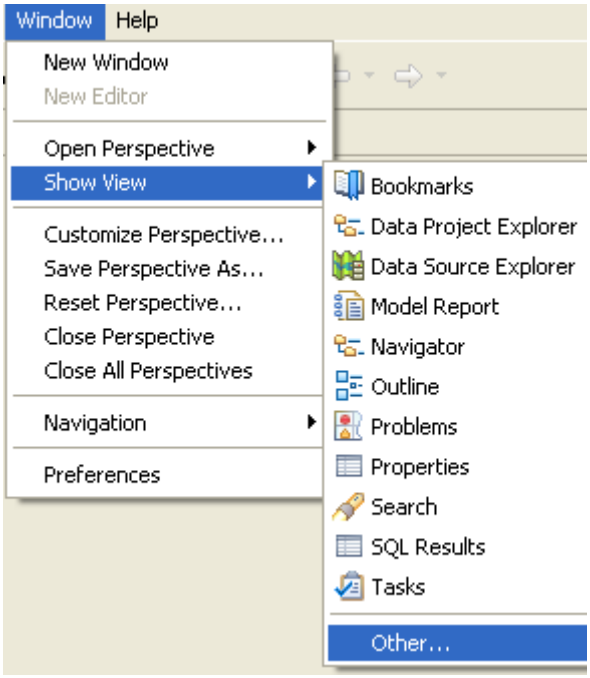


Σχήμα 4.6 Η βάση δεδομένων στον Data Source Explorer

4.5 Προσθήκη του WebSphere

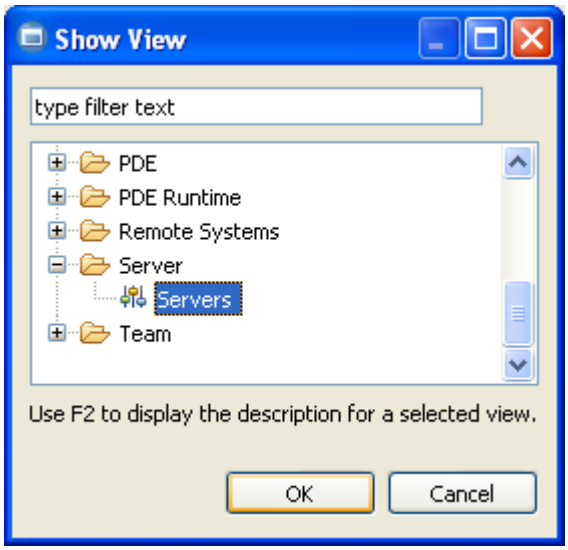
Όπως έχουμε προαναφέρει, ο J2EE application server στον οποίο θα κάνουμε deploy και θα χρησιμοποιούμε τις data web services είναι ο IBM WebSphere Application Server Community Edition. Για να τον προσθέσουμε ως server στο Data Studio, ακολουθούμε τα εξής βήματα:

1. Στο κεντρικό μενού επιλέγουμε **Window > Show View > Other..** (Σχήμα 4.7)



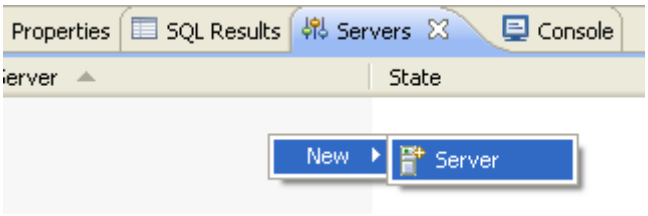
Σχήμα 4.7 Προσθήκη του server view

2. Στο παράθυρο των Views που εμφανίζεται (Σχήμα 4.8) επιλέγουμε **Server > servers** και κάνουμε κλικ στο *OK*.



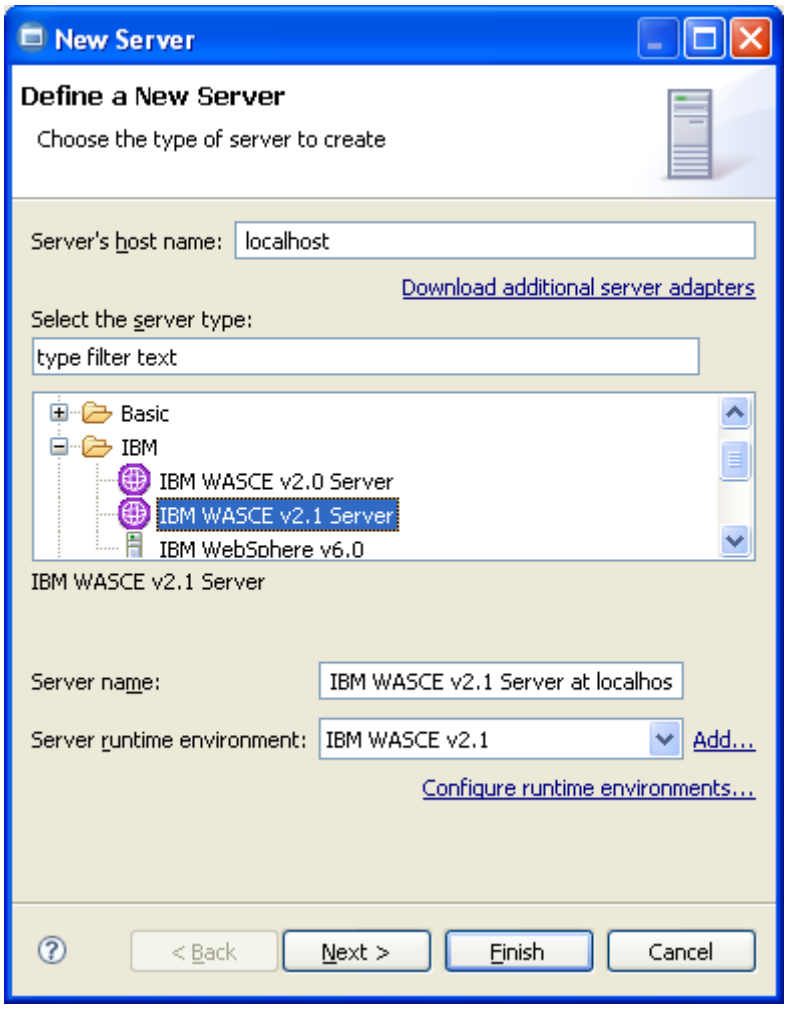
Σχήμα 4.8 Παράθυρο επιλογής Views

3. Στο Properties View (κάτω στο κέντρο του παραθύρου) εμφανίζονται δύο επιπλέον tabs με ονομασία Servers και console (σχήμα .9). Κάνουμε δεξί κλικ μέσα στην καρτέλα Servers και επιλέγουμε **New>Server**.



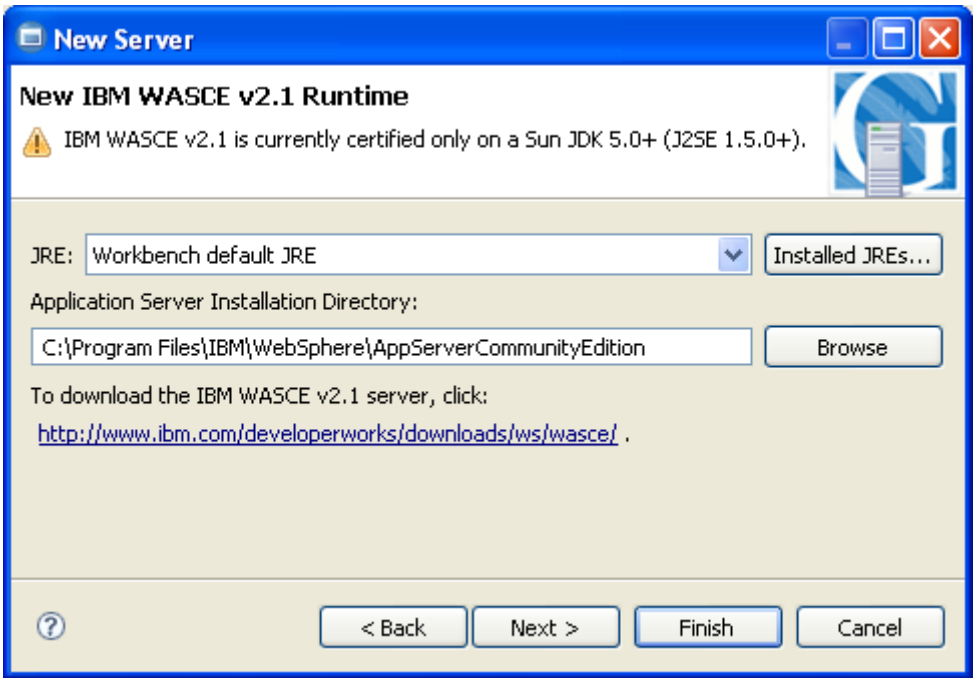
Σχήμα 4.9 Εισαγωγή νέου server

4. Στο παράθυρο που εμφανίζεται (Σχήμα 4.10), επιλέγουμε τον WebSphere v2.1 server και πατούμε *Next*.



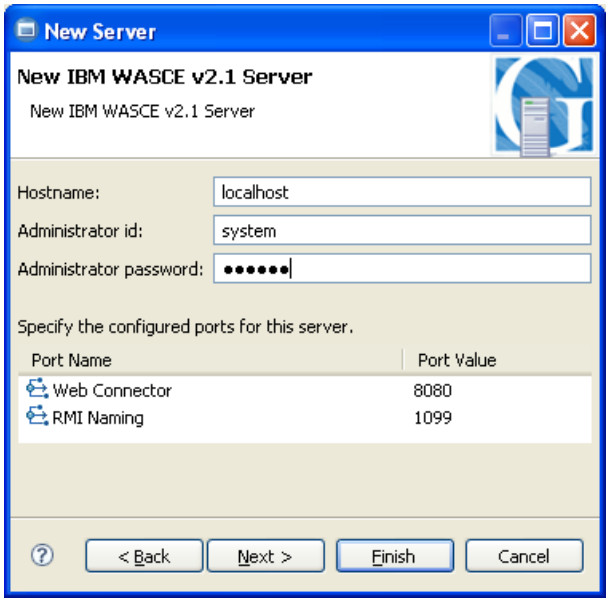
Σχήμα 4.10 Ορισμός ιδιοτήτων server

5. Στο παράθυρο που εμφανίζεται (Σχήμα 4.11), επιλέγουμε το WorkBench default JRE ως Java Runtime Environment, εισάγουμε το path του φακέλου που έχει εγκατασταθεί ο WebSphere και πατούμε *Next*.



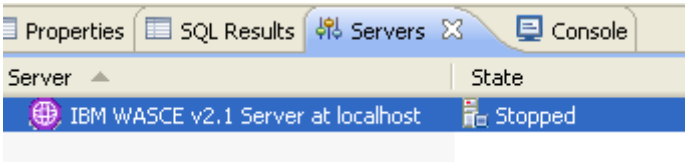
Σχήμα 4.11 Ορισμός του JRE και του installation path του server

6. Στο παράθυρο που εμφανίζεται (Σχήμα 4.12), εισάγουμε το username και τον κωδικό του administrator του server και πατούμε *Finish*.



Σχήμα 4.12 Εισαγωγή administrator user και password

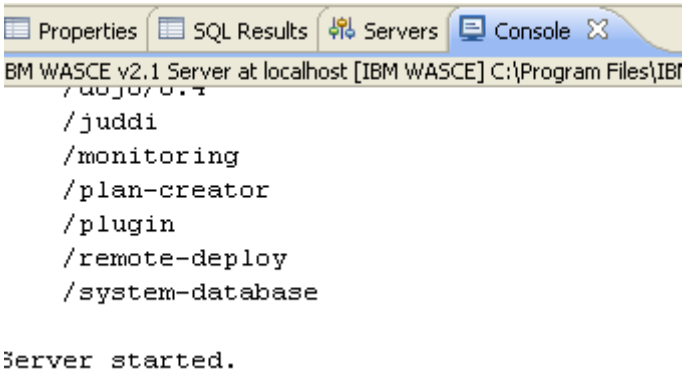
7. Στο Properties View (κάτω στο κέντρο του παραθύρου) μέσα στην καρτέλα servers (Σχήμα 4.13) έχει προστεθεί ο WebSphere. Παρατηρούμε ότι ο server είναι απενεργοποιημένος. Αν θέλουμε να τον εκκινήσουμε χειροκίνητα προχωρούμε στο επόμενο βήμα. Αλλιώς όταν κάνουμε deploy μια web service το Data Studio πραγματοποιεί έλεγχο και εκκινεί τον server αν είναι σβηστός.



Σχήμα 4.13 Ο WebSphere στο servers view

7. Για να εκκινήσουμε τον WebSphere, κάνουμε δεξί κλικ πάνω του (Σχήμα 4.14) και επιλέγουμε Start. Η διαδικασία εκκίνησης κρατά κάποια δευτερόλεπτα. Μπορείτε να δείτε την εξέλιξή της στο Console View (Σχήμα 4.15). Στο τέλος εμφανίζεται στο Console το μήνυμα «Server started.» το οποίο μας δείχνει ότι η εκκίνηση ολοκληρώθηκε επιτυχώς.

Σχήμα 4.14 Χειροκίνητη εκκίνηση του WebSphere

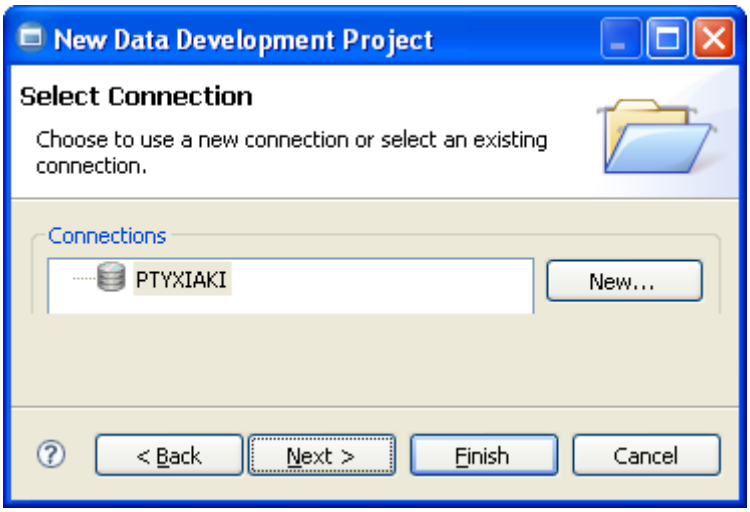


Σχήμα 4.15 Η ολοκλήρωση της εκκίνησης του WebSphere

4.6 Δημιουργία νέου Data Project

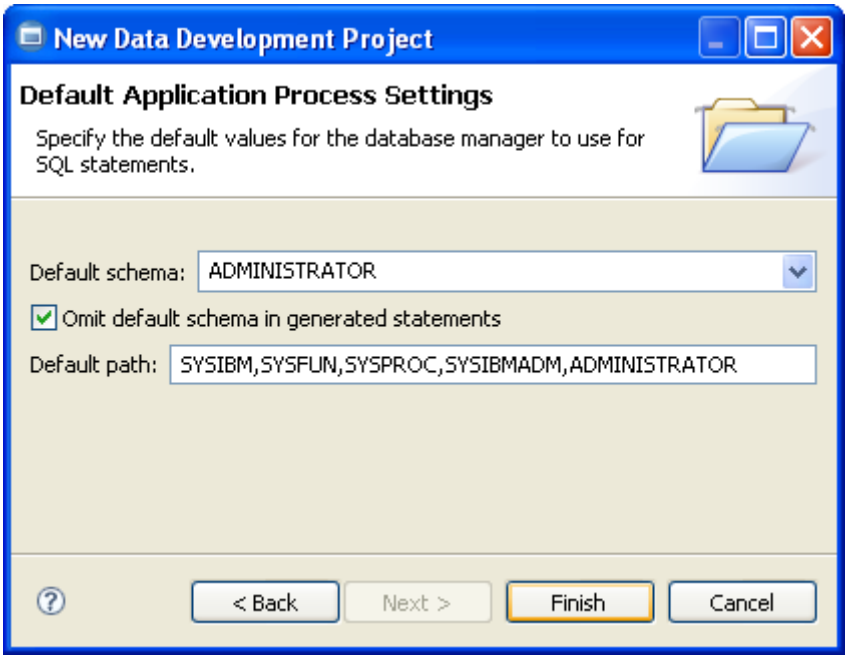
Ακολουθούν τα βήματα δημιουργίας νέου Data Development Project μέσα στο οποίο θα δημιουργήσουμε την stored procedure και την data web service μας.

1. Από το κεντρικό μενού επιλέγουμε **File>New>Projects>Data Development Project**. Στο παράθυρο που εμφανίζεται ονοματίζουμε το project Project1 και πιέζουμε το *Next*.
2. Επιλέγουμε τη σύνδεση με τη βάση δεδομένων (Σχήμα 4.16). Αν δεν υπάρχει καμία σύνδεση πιέζουμε το πλήκτρο New. Πιέζουμε το πλήκτρο *Next*.



Σχήμα 4.16 Επιλογή βάσης δεδομένων

3. Τέλος επιλέγουμε το default σχήμα της βάσης πάνω στο οποίο θα βλέπει το project. Εμείς επιλέξαμε το σχήμα Administrator. Τέλος κάνουμε κλικ στο κουμπί *Finish* και το Project1 εμφανίζεται στον *Data Project Explorer*.

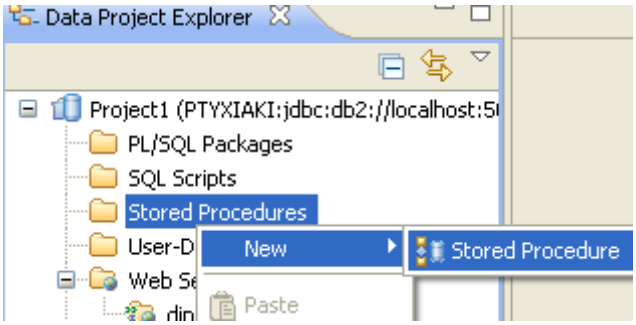


Σχήμα 4.17 Επιλογή default database schema

4.7 Δημιουργία νέας stored procedure

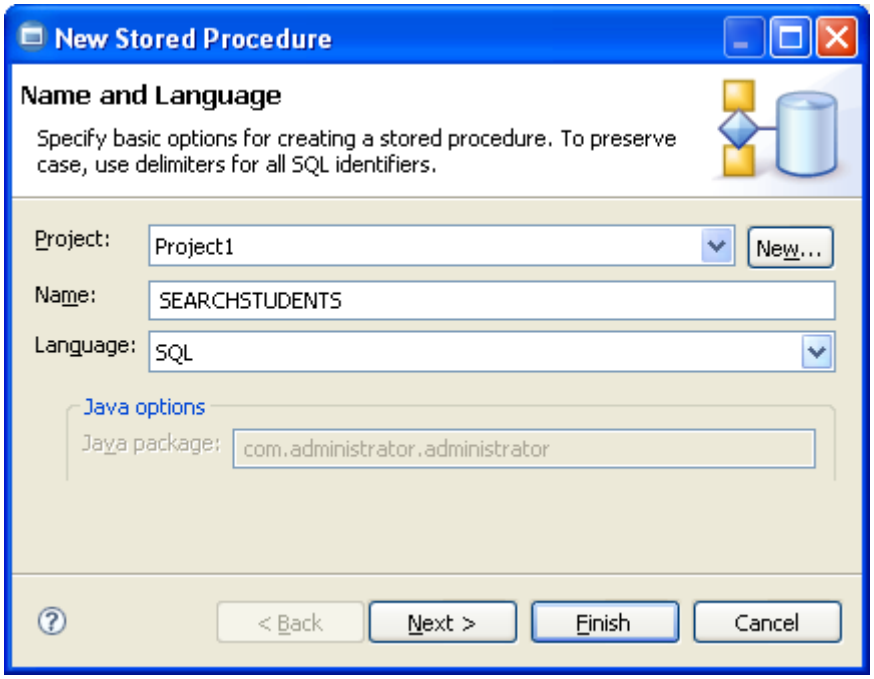
Σε αυτή την ενότητα θα περιγράψουμε τα βήματα με τα οποία δημιουργούμε stored procedures στο data studio. Μέσω αυτής θα δημιουργήσουμε την stored procedure SEARCHSTUDENTS η οποία θα καλείται από την web service που θα κατασκευάσουμε για να αναζητήσει φοιτητές με βάση ένα κλειδί αναζήτησης.

1. Στον Data Project Explorer (Σχήμα 4.18) δεξί κλικ πάνω στον φάκελο stored procedures και επιλέγουμε **New>Stored procedure**.



Σχήμα 4.18 New stored procedure

2. Στο παράθυρο που εμφανίζεται (Σχήμα 4.19) επιλέγουμε το Project1 ως project στο οποίο θα ανήκει η procedure. Κατόπιν εισάγουμε το όνομά της και επιλέγουμε ως γλώσσα προγραμματισμού την SQL. Μπορούμε να δημιουργήσουμε stored procedures γραμμένες σε καθαρή SQL, σε PL/SQL (γλώσσα της Oracle DB) και σε Java. Στην συγκεκριμένη περίπτωση επιλέγουμε SQL γιατί είναι πιο εύκολη και πιο γρήγορη. Τέλος επιλέγουμε το κουμπί *Finish*.

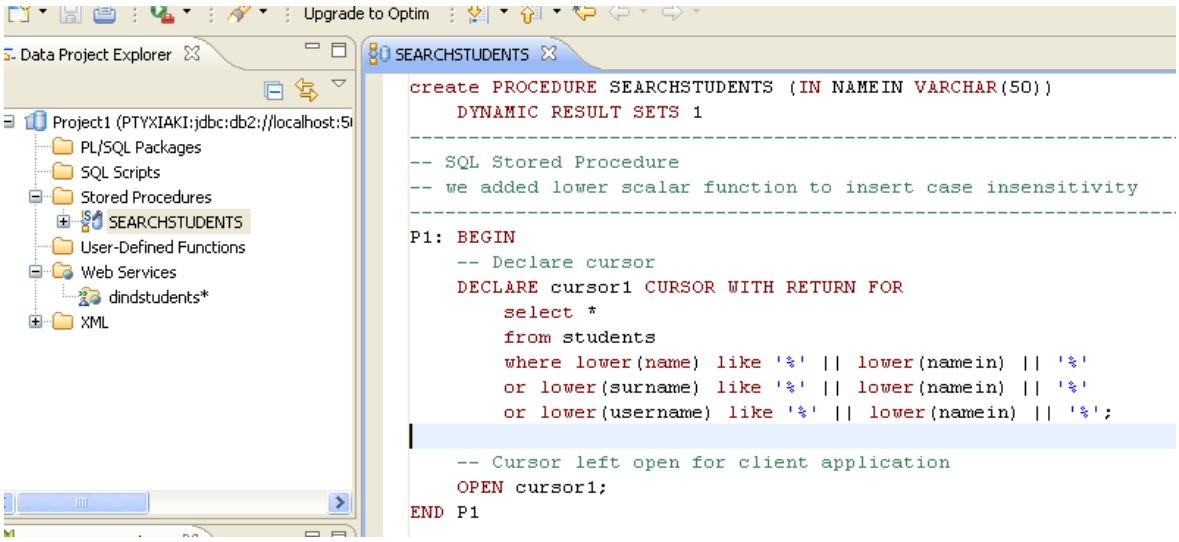


Σχήμα 4.19 Επιλογή παραμέτρων για την stored procedure

3. Στον editor εμφανίζεται η αρχική δήλωση της stored procedure (Σχήμα 4.20). Γράφουμε τον κώδικα της δικής μας stored procedure

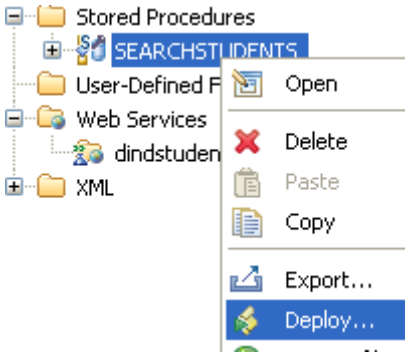
```
1. create PROCEDURE SEARCHSTUDENTS (IN NAMEIN VARCHAR(50))
2.     DYNAMIC RESULT SETS 1
3. P1: BEGIN
4.     -- Declare cursor
5.     DECLARE cursor1 CURSOR WITH RETURN FOR
6.         select *
7.         from students
```

```
8.          where lower(name) like '%' || lower(namein) || '%'
9.          or lower(surname) like '%' || lower(namein) || '%'
10.          or lower(username) like '%' || lower(namein) || '%';
11.
12.      -- Cursor left open for client application
13.      OPEN cursor1;
14. END P1;
```



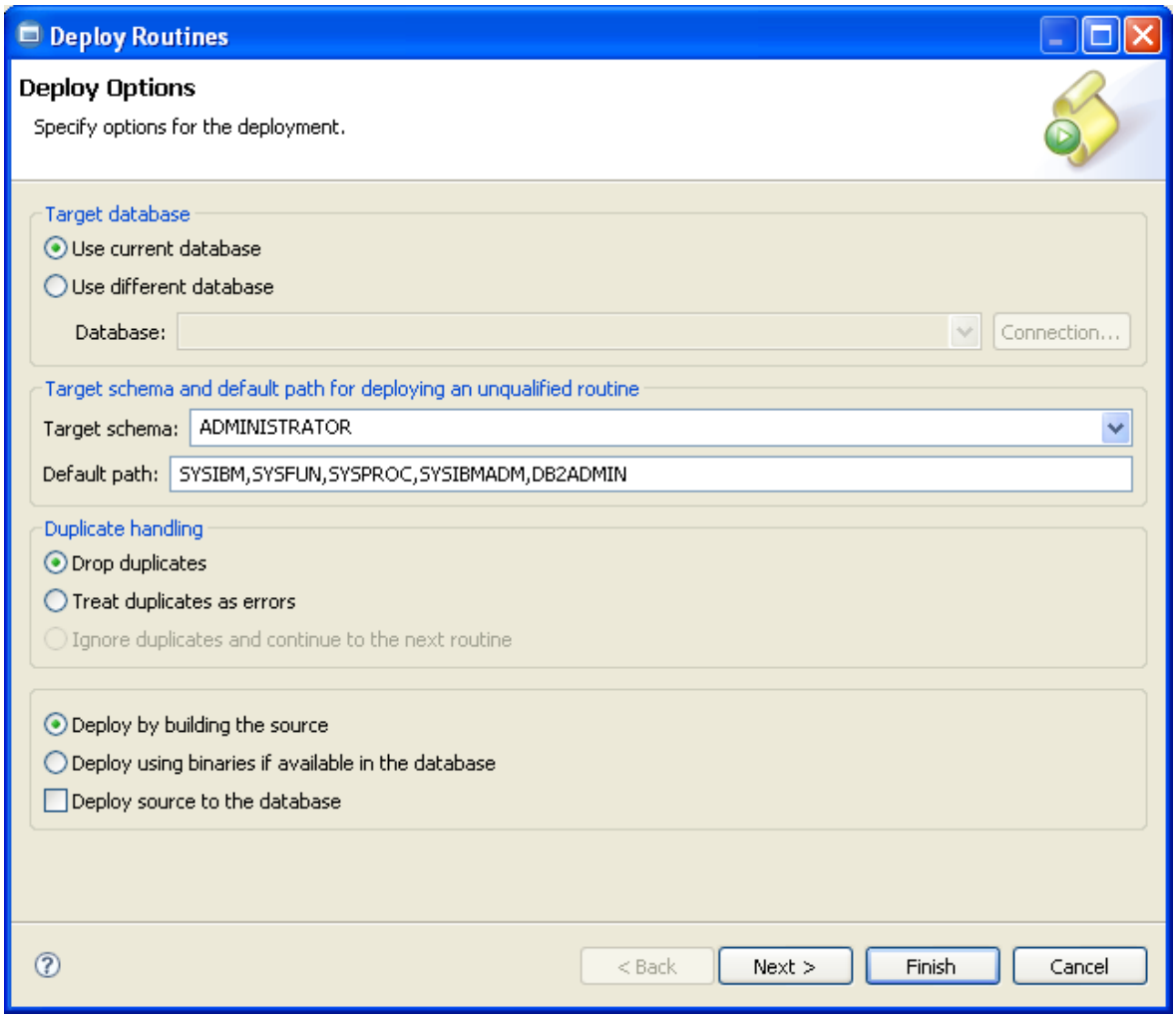
Σχήμα 4.20 Ο κώδικας της stored procedure στον editor

4. Αποθηκεύουμε τις αλλαγές μας στο project (Ctrl+S) και έτσι ο κώδικας της stored procedure είναι αποθηκευμένος μέσα στο project1.
5. Για να δημιουργήσουμε την stored procedure μέσα στη βάση δεδομένων πρέπει να την κάνουμε deploy σε αυτήν. Για να γίνει αυτό κάνουμε δεξί κλικ πάνω στην stored procedure (Σχήμα 4.21) και επιλέγουμε *Deploy..*



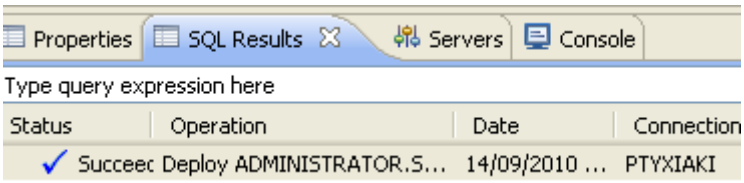
Σχήμα 4.21 Κάνοντας deploy μια stored procedure

6. Εμφανίζεται ένα παράθυρο επιλογών (Σχήμα 4.22), όπου επιλέγουμε ως Targer Database: Current Database επιλέγουμε ως targert schema το administrator (ή όποιο άλλο έχετε δημιουργήσει εσείς) και επιλέγουμε *Finish.*



Σχήμα 4.22 Επιλογές deployment για την stored procedure

7. Η διαδικασία ξεκινά και όταν ολοκληρωθεί εμφανίζεται στο SQL Results View μήνυμα επιτυχίας. Αν για κάποιο λόγο το deploy αποτύχει, εμφανίζεται μήνυμα σφάλματος και επεξήγηση.



Σχήμα 4.23 Επιτυχία του deploy στη βάση δεδομένων

⚠ Αξίζει να σημειώσουμε ότι το όνομα της stored procedure και τα ονόματα των παραμέτρων της αποθηκεύονται **με κεφαλαία γράμματα** στη βάση δεδομένων. Όταν θα χρειαστεί να χρησιμοποιήσουμε τη stored procedure ως μέθοδο της data web service πρέπει να προσέξουμε να γράφουμε το όνομα της stored procedure και τα ονόματα των παραμέτρων **με κεφαλαία γράμματα**.

4.8 Δημιουργία νέου SQL script

Σε αυτή την ενότητα θα περιγράψουμε τα βήματα με τα οποία δημιουργούμε ένα νέο SQL script στο data studio. Το SQL script είναι ένα αρχείο που περιέχει μια ή περισσότερες εντολές SQL είτε μια ή περισσότερες εντολές DML (insert, delete, update) ή DDL (create, drop, alter). Ένα αρχείο SQL script μπορεί να ενταχθεί μέσα σε μια web service και να αποτελέσει μέθοδο της. Το όνομα της μεθόδου θα είναι το όνομα του αρχείου. Αν εντάξουμε ένα τέτοιο αρχείο σε μια Data Web Service τότε ισχύουν τα παρακάτω:

- Οι εντολές που εμπεριέχονται στο αρχείο είναι αμετάβλητες και δεν παίρνουν παραμέτρους όπως οι stored procedures.
- Αν το αρχείο περιέχει μια εντολή SQL επιστρέφει ένα result set.
- Αν το αρχείο περιέχει περισσότερες από μία εντολές SQL επιστρέφει τόσα result sets όσες είναι και οι εντολές SQL.

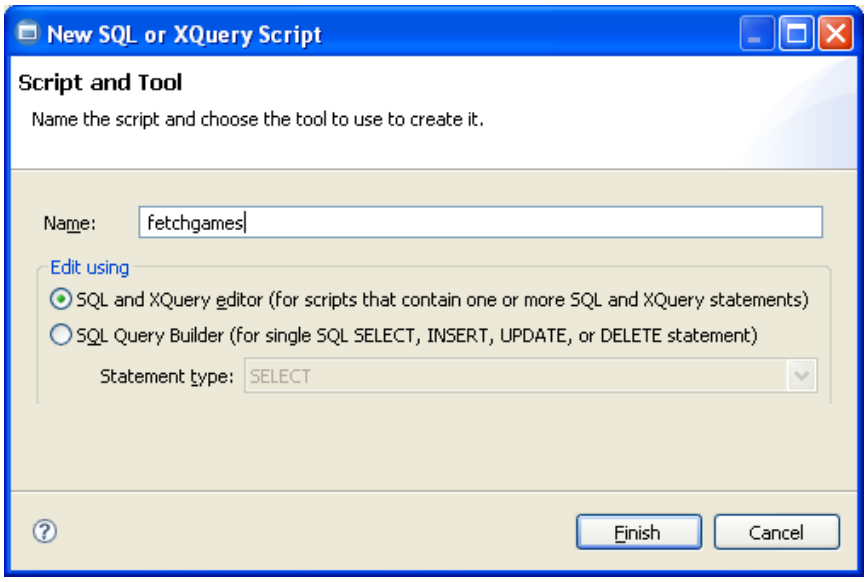
⚠ Αξίζει να σημειώσουμε ότι το όνομα του αρχείου του SQL script αποθηκεύεται μέσα σε μια stored procedure και αποτελεί και όνομα της μεθόδου που αντιπροσωπεύει αυτό το script. Όταν θα χρειαστεί να χρησιμοποιήσουμε τη μέθοδο της data web service που προέκυψε από το SQL script πρέπει να προσέξουμε να γράφουμε το όνομα της όπως το γράψαμε στο Data Studio επειδή οι κλήσεις και τα URLs είναι **case sensitive**.

Για να δημιουργήσουμε ένα SQL script στο IBM Data Studio προχωρούμε στα παρακάτω βήματα. ΠΡΟΣΟΧΗ! Το script που θα δημιουργήσουμε δεν αποτελεί μέρος της web service που θα δημιουργήσουμε αργότερα.

1. Στον Data Project Explorer κάνουμε δεξί κλικ στο φάκελο Web Services (Σχήμα 4.24) και επιλέγουμε *New > SQL or XQuery Script.*

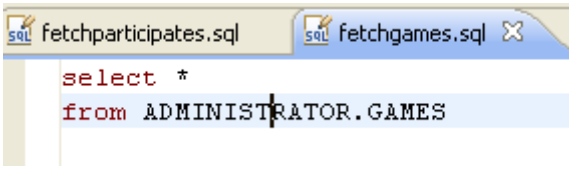
Σχήμα 4.24 Δημιουργία νέου SQL script

2. Στο παράθυρο που εμφανίζεται δίνουμε ένα όνομα στο script (Σχήμα 4.25) και επιλέγουμε να το διαμορφώσουμε με τον SQL Query Editor και επιλέγουμε Finish.



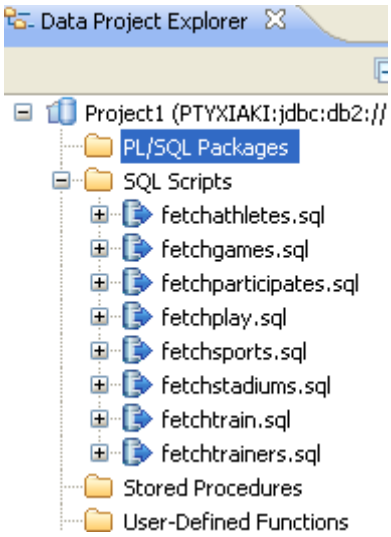
Σχήμα 4.25 Ρύθμιση ονόματος script

3. Στον editor (Σχήμα 4.26) πληκτρολογούμε το/τα SQL ή DML ή DDL statements και αποθηκεύουμε το αρχείο (Ctrl+S). Τα αρχεία φέρουν την κατάληξη sql και είναι καθαρά αρχεία κειμένου.



Σχήμα 4.26 Το sql statement στον editor του SQL script.

4. Τα SQL scripts αποθηκεύονται και είναι ορατά στον Data Project Explorer κάτω από τον ομώνυμο φάκελο (Σχήμα 4.27). Από εκεί μπορούμε με drag and drop να τα σύρουμε μέσα σε μια data web service και να μετατραπούν σε μέθοδο της.

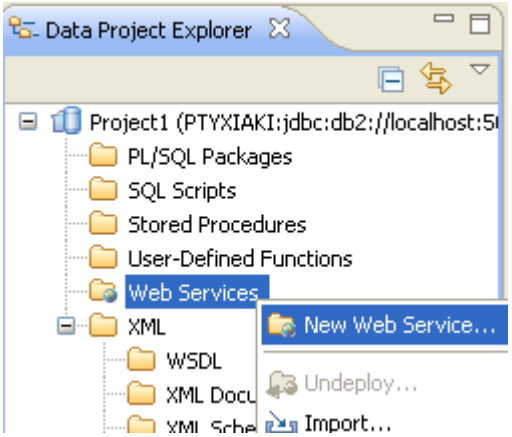


Σχήμα 4.27 Τα SQL scripts στον Data Project Explorer

4.9 Δημιουργία νέας Data Web service

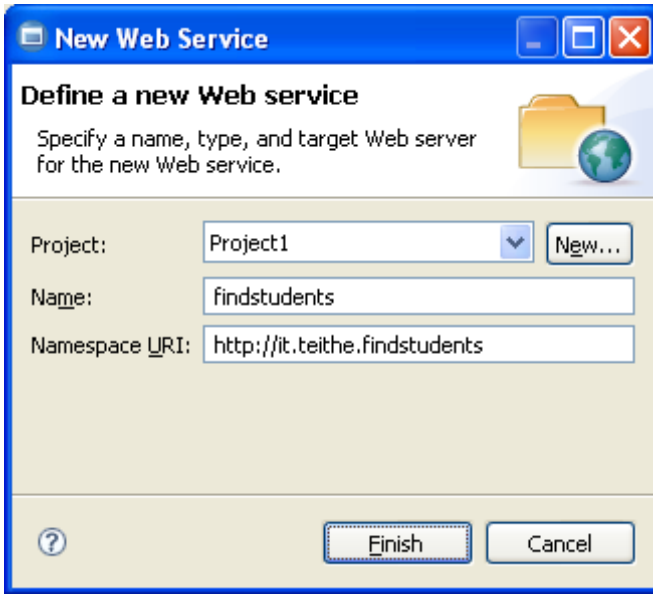
Για να δημιουργηθεί μια νέα Data Web Service πρέπει να προχωρήσουμε στα εξής βήματα:

1. Στον Data Project Explorer κάνουμε δεξί κλικ στο φάκελο Web Services (Σχήμα 4.28) και επιλέγουμε *New Web Service..*



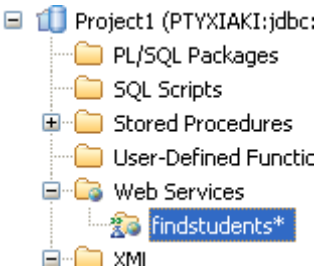
Σχήμα 4.28 Δημιουργία νέας web service στον Data Project Eplorer

2. Μας εμφανίζεται ένα παράθυρο επιλογών (Σχήμα 4.29) όπου επιλέγουμε το Project στο οποίο ανήκει η web service, εν προκειμένω το Project1. Δίνουμε ως όνομα για την Data Web Service το findstudents και θέτουμε ένα URI το οποίο θα χρησιμοποιηθεί στα μηνύματα του πρωτοκόλλου SOAP. Τέλος επιλέγουμε το *Finish*.



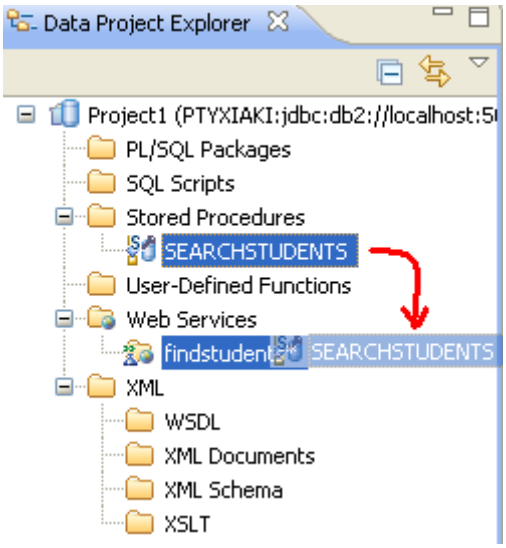
Σχήμα 4.29 Ρύθμιση ιδιοτήτων της web service

3. Η Data Web Service έχει δημιουργηθεί ως οντότητα μέσα στο project1 και μπορούμε να τη δούμε μέσα στο φάκελο Web Services. Παρατηρούμε ότι δίπλα το όνομα της Web Service έχει έναν αστερίσκο στα αριστερά του. Αυτό δηλώνει ότι η findstudents δεν έχει γίνει compiled (build) και δεν έχει γίνει deployed στον WebSphere application server.



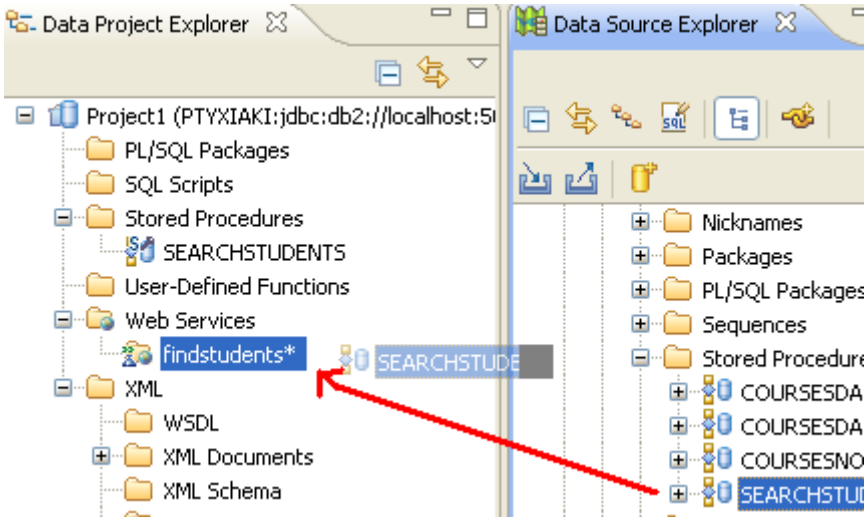
Σχήμα 4.30 Η findstudents όπως φαίνεται στον Data Project Explorer

4. Η data web service μέχρι στιγμής δεν έχει κάποια λειτουργικότητα. Για να την κάνουμε να αλληλεπιδρά με τη βάση και να μας επιστρέφει δεδομένα, προσθέτουμε μια stored procedure, την SEARCHSTUDENTS που ορίσαμε σε προηγούμενη ενότητα. Για να το επιτύχουμε αυτό στον Data Project Explorer σέρνουμε με drag and drop την stored procedure μέσα στην web service (Σχήμα 4.31).



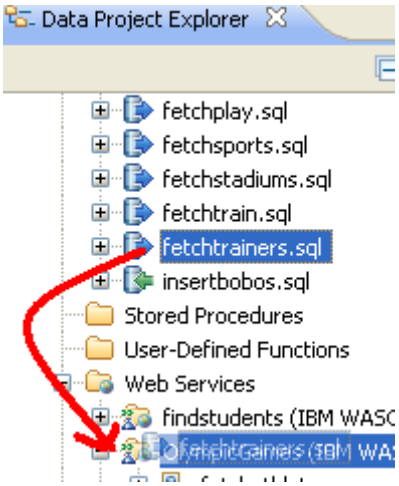
Σχήμα 4.31 Drag and drop της stored procedure από το Project στη Web Service

5. Αν έχουμε μια ήδη υπάρχουσα stored procedure μέσα στη βάση δεδομένων η οποία δεν έχει δημιουργηθεί μέσα σε αυτό το project μπορούμε να τη βρούμε μέσω του Data Source Explorer και να τη σύρουμε με drag and drop από τον Data Source Explorer στη web service μέσα στον Data Project Explorer (Σχήμα 4.32).



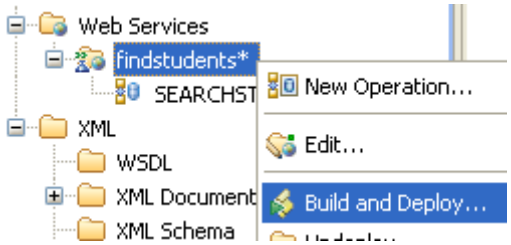
Σχήμα 4.32 Drag and drop της stored procedure από τον Data Source Explorer στη Web Service

6. Αν θέλουμε να συμπεριλάβουμε και SQL scripts μέσα στην Data Web Service, τότε με μπορούμε να τα σύρουμε με drag and drop από τον φάκελο τους μέσα στη web service στον Data Project Explorer (Σχήμα 4.33). ΠΡΟΣΟΧΗ!! Το βήμα αυτό είναι πληροφοριακό και δεν αφορά την τρέχουσα data web service (findstudents) η οποία έχει μόνο μια μέθοδο που προκύπτει από την stored procedure SEARCHSTUDENTS.



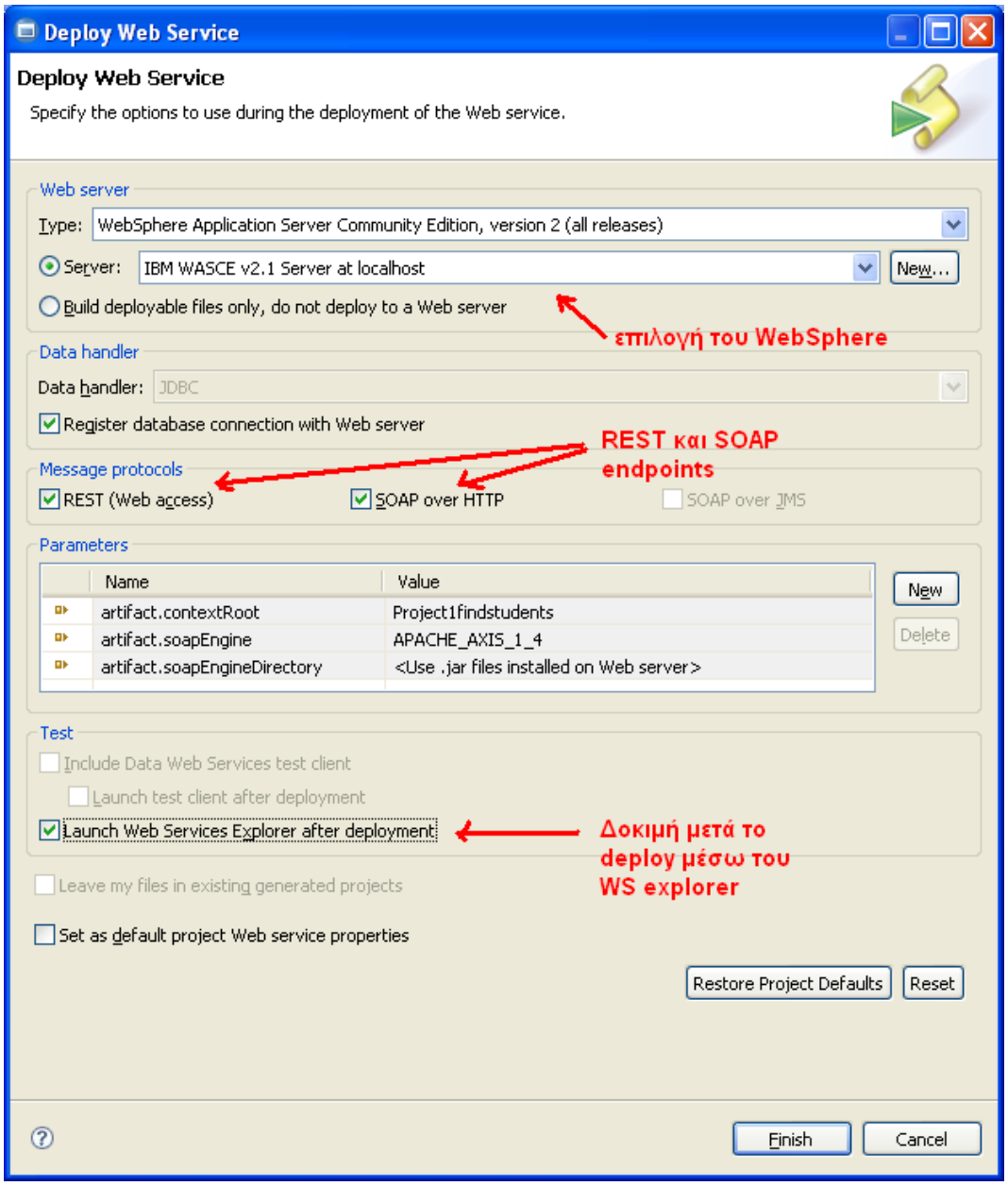
Σχήμα 4.33 Drag and drop ενός SQL script μέσα σε Web Service

7. Η web service μας είναι έτοιμη για μεταγλώττιση και deploy στον WebSphere application server. Έως τώρα δεν έχουμε γράψει καμιά γραμμή κώδικα. Την επίπονη αυτή εργασία την αναλαμβάνει στο παρασκήνιο το Data Studio. Κάνουμε save το project μας (Ctrl+S) για να αποθηκευθούν οι αλλαγές. Ακολούθως, με δεξί κλικ πάνω στην web service (Σχήμα 4.34), επιλέγουμε *Build and Deploy*.



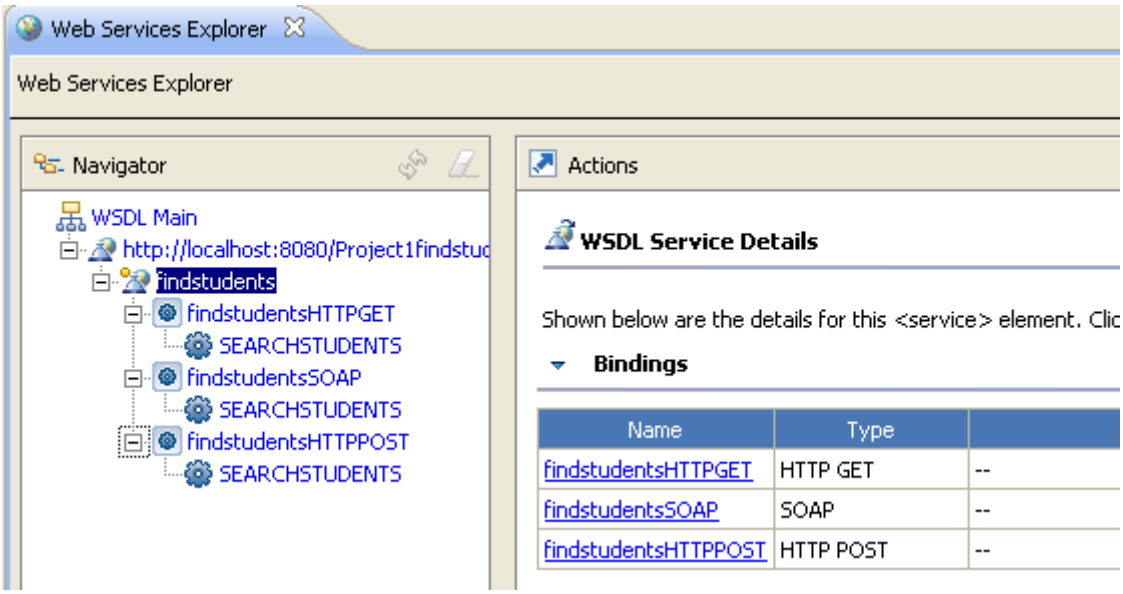
Σχήμα 4.34 Επιλογή Deploy στον WebSphere

8. Στο παράθυρο που εμφανίζεται (Σχήμα 4.35) πρέπει να ορίσουμε τις παραμέτρους του deploy. Επιλέγουμε να σταλεί η web service στον WebSphere application server. Κατόπιν επιλέγουμε να δημιουργηθούν SOAP και REST endpoints. Τέλος επιλέγουμε να φορτωθεί ο Web Services Explorer μετά το deploy. Ο WS explorer είναι μια διεπιφάνεια χρήστη που περιλαμβάνεται στο Data Studio με την οποία μπορούμε να δοκιμάσουμε τις web services που έχουμε δημιουργήσει. Τέλος επιλέγουμε το *Finish*.



Σχήμα 4.36 Επιλογές για το deployment της web service στον WebSphere

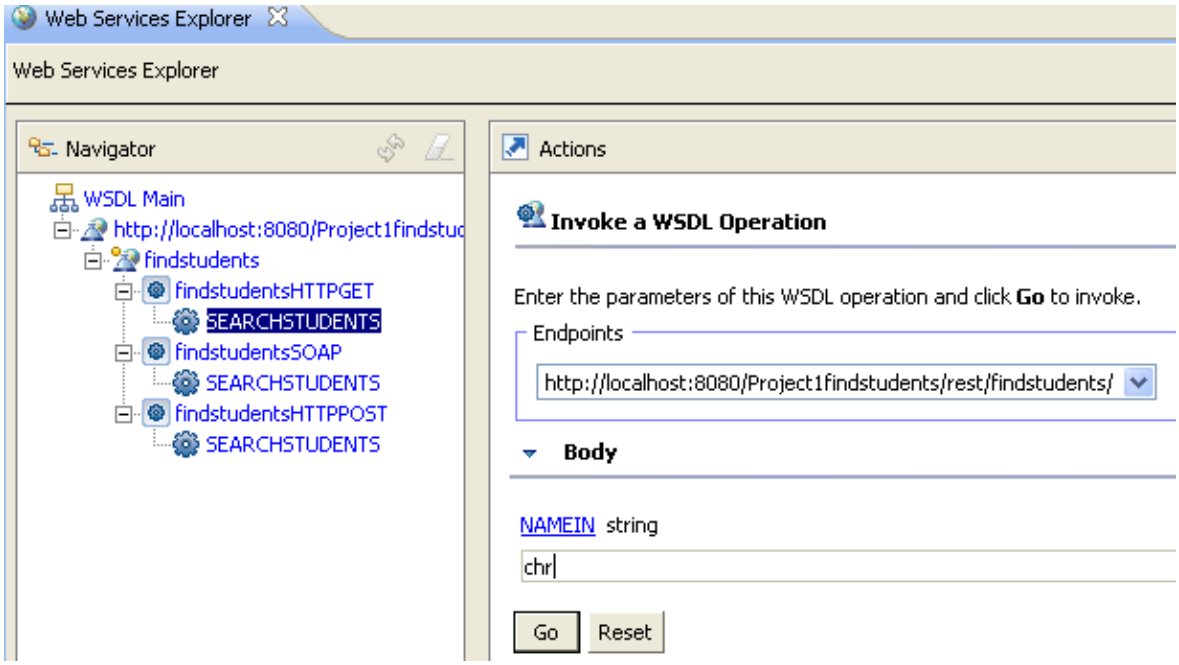
9. Εμφανίζεται ένα μήνυμα που δείχνει την πρόοδο του built and deploy. Αν δεν έχετε εκκινήσει τον WebSphere (στο servers view), το Data Studio τον εκκινεί αυτόματα προτού αρχίσει το deploy. Όταν ολοκληρωθεί το deploy, στο editors view ανοίγει ο Web Services Explorer, στον οποίο εμφανίζονται οι υποδοχείς (endpoints) της web service (soap, rest http get, rest http post) καθώς και οι stored procedures ή τα dml statements που εμπεριέχουν (Σχήμα 4.37)



Σχήμα 4.37 Η web service φορτωμένη στον ws explorer

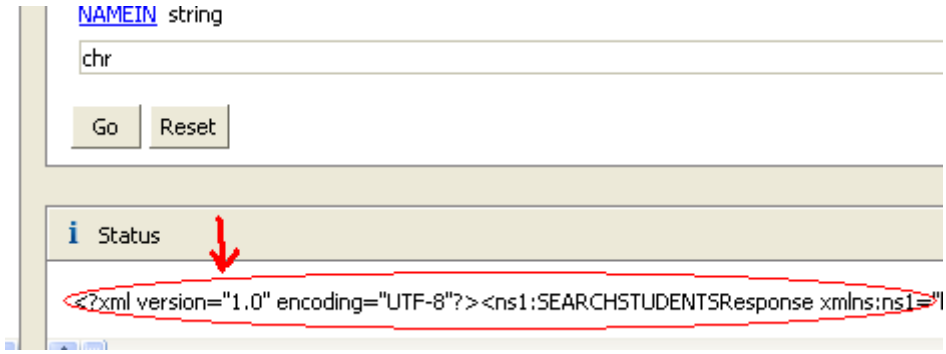
4.10 Δοκιμή της web service

1. Στον ws explorer μπορούμε να δοκιμάσουμε τη web service μας για να διαπιστώσουμε ότι λειτουργεί σωστά και παράγει τα επιθυμητά αποτελέσματα. Στην περίπτωση της findstudents έστω ότι αναζητούμε έναν φοιτητή ο οποίος έχει μέσα στο όνομά του ή το επίθετό του ή το username του το string «chr». Στη συγκεκριμένη περίπτωση θα δοκιμάσουμε την REST HTTP έκδοση της findstudents. Επιλέγουμε την stored procedure searchstudents στον υποδοχέα findstudentsHTTPGET του navigator (Σχήμα 4.38) Κατόπιν στο textbox στο παράθυρο actions συμπληρώνουμε το string “chr”. Παρατηρούμε ότι το πεδίο ονομάζεται NAMEIN και πιέζουμε το πλήκτρο *Go*. Το “namein” είναι και το όνομα της input παραμέτρου που περνούμε στην stored procedure όταν τη δηλώσαμε.



Σχήμα 4.38 Δοκιμή της findstudents με REST HTTP GET μέθοδο.

2. Κάτω από το Go εμφανίζεται η απάντηση με τους φοιτητές που βρέθηκαν σε μορφοποίηση XML (Σχήμα 4.39).



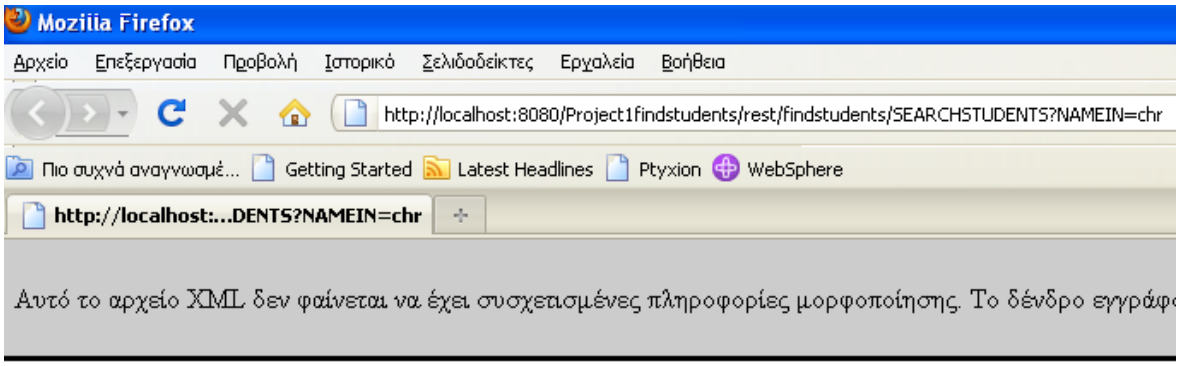
Σχήμα 4.39 Η απάντηση της web service

3. Στην περίπτωση του HTTP GET και POST μπορούμε να έχουμε απευθείας πρόσβαση στην web service μέσω του url της. Αν παρατηρήσουμε στο Σχήμα 4.38 μας δίνεται το url του endpoint το οποίο είναι το http://localhost:8080/Project1findstudents/rest/findstudents

Για να καλέσουμε την stored procedure με παράμετρο “chr”, προσθέτουμε στο url τον χαρακτήτα /, το όνομα της stored procedure με κεφαλαία γράμματα και τις παραμέτρους όπως θα κάναμε σε μια κλασσική κλήση μιας ιστοσελίδας με μέθοδο GET. Τα ονόματα των παραμέτρων γράφονται και αυτά με κεφαλαία γράμματα.

Έτσι το url θα είναι
http://localhost:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=chr

Αν αντιγράψουμε αυτό το Link και το τρέξουμε σε έναν web browser όπως ο Mozilla Firefox μας εμφανίζει (Σχήμα 4.40) την XML απάντηση όπως μας εμφανίστηκε και στον Web Services Explorer.




```
− <ns1:SEARCHSTUDENTSResponse>
  − <rowset>
    − <row>
        <USERNAME>chrandr</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Andrianos</SURNAME>
        <URL>http://www.it.teithe.gr/~chrandr</URL>
        <EMAIL>chrandr@it.teithe.gr</EMAIL>
      </row>
```

Σχήμα 4.40 Η απάντηση της findstudents σε HTTP GET κλήση

4.11 Επίλογος

Στο κεφάλαιο αυτό περιγράψαμε αναλυτικά όλα τα βήματα που χρειάζονται για την προετοιμασία, τη δημιουργία και τη δοκιμή μιας Data Web Service με τη χρήση του Data Studio. Στο επόμενο κεφάλαιο θα αναλύσουμε με ποιους τρόπους μπορούμε να επικοινωνήσουμε με μια Data Web Service μέσω των υποδοχέων (endpoints) που διατίθενται.

5. Ανάλωση των Web Services

5.1 Εισαγωγή

Στο προηγούμενο κεφάλαιο αναφερθήκαμε στα βήματα που πρέπει να ακολουθήσουμε για να δημιουργήσουμε Data Web Services με τη χρήση του Data Studio. Κατόπιν τις δοκιμάσαμε μέσω του Web Services explorer. Στο κεφάλαιο αυτό θα δείξουμε τη δομή των μηνυμάτων επικοινωνίας μεταξύ της web service και ενός προγράμματος πελάτη καθώς και το πώς μπορούμε να καλέσουμε τις DWS από ιστοσελίδες. Παράλληλα θα παρουσιάσουμε εφαρμογές και τεχνολογίες που μας βοήθησαν να χρησιμοποιήσουμε και να δοκιμάσουμε τις data web services.

5.2 Οι υποδοχείς

Οι υποδοχείς ή endpoints που μας δίνονται για μια web service είναι δύο ειδών: SOAP και REST. Όταν δημιουργούμε μια web service επιλέγουμε αν θέλουμε και τους δύο ή μόνον ένα τύπο endpoint (Σχήμα 4.36). Στην πτυχιακή μας επιλέγουμε και τους δύο τύπους. Συνολικά τα bindings για τα δύο endpoints που μπορούμε να χρησιμοποιήσουμε σε μια data web service είναι τα εξής 6:

- SOAP over HTTP: Αυτή είναι η πιο γνωστή σύνδεση (binding) που χρησιμοποιείται για εφαρμογές πελάτες βασισμένες σε WSDL και SOAP frameworks.
- REST HTTP GET με XML output: το binding αυτό χρησιμοποιείται από Web 2.0 clients και web browsers με απάντηση μορφοποιημένη σε XML.
- REST HTTP GET με JSON output: το binding αυτό χρησιμοποιείται από Web 2.0 clients και web browsers, JavaScript εφαρμογές και frameworks με απάντηση μορφοποιημένη σε JSON.
- REST HTTP POST με XML output: το binding αυτό χρησιμοποιείται από Web 2.0 clients και web browsers, JavaScript εφαρμογές και frameworks με απάντηση μορφοποιημένη σε XML.
- REST HTTP POST Url-encoded με XML output: το binding αυτό χρησιμοποιείται από Web 2.0 clients και web browsers, AJAX, JavaScript εφαρμογές και frameworks με απάντηση μορφοποιημένη σε JSON.
- REST HTTP POST με JSON output

Επίσης όσον αφορά στα endpoints ισχύουν τα παρακάτω:

- Όλα τα endpoints μπορούν να κληθούν από client εφαρμογές και server side scripts (PHP,ASP,JSP).
- Όλα τα endpoints μπορούν να κληθούν από JavaScript μέσω AJAX μόνο αν βρίσκονται στο ίδιο domain με το script που τα καλεί. Αλλιώς

δεν μπορούν να κληθούν γιατί προσκρούουν στο Same Origin Policy (βλέπε ενότητα 1.11.4 JSONP).
- Γι' αυτό εμείς σε μια ακόλουθη ενότητα μέσω της PHP θα καλέσουμε το JSON HTTP GET binding και θα παράγουμε JSONP, ώστε να ξεπεράσουμε αυτόν τον περιορισμό.

5.3 cURL

Το cURL είναι μια command line εφαρμογή γραμμένη σε C, η οποία δύναται να μεταφέρει δεδομένα μέσω πολλών πρωτοκόλλων μέσω ενός URL. Εμείς τη χρησιμοποιούμε για να δοκιμάσουμε τις web services μας στέλνοντας μηνύματα με το πρωτόκολλο HTTP.

Το cURL υποστηρίζει τα πρωτόκολλα DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET και TFTP. Έχει επίσης ενσωματωμένη βιβλιοθήκη για τη δημιουργία κρυπτογραφημένων συνδέσεων μέσω του πρωτοκόλλου SSL.
Είναι open source και μπορεί να δουλέψει σχεδόν στα περισσότερα λειτουργικά συστήματα και πλατφόρμες που υπάρχουν. Μπορείτε να το κατεβάσετε δωρεάν στη διεύθυνση http://curl.haxx.se/download.html

5.4 JQuery

Σχήμα 5.1 Το λογότυπο της JQuery

Η JQuery είναι μια open source, cross browser βιβλιοθήκη JavaScript η οποία έχει σχεδιαστεί για να απλοποιήσει τον client – side προγραμματισμό των HTML ιστοσελίδων. Παρουσιάστηκε για πρώτη φορά το 2006 και σήμερα χρησιμοποιείται από το 36,8% των 10.000 δημοφιλέστερων site παγκοσμίως.

Είναι πολύ ελαφριά και περιέχει functions που υλοποιούν λειτουργίες οι οποίες είναι απαραίτητες ή χρησιμοποιούνται συχνά για το σχεδιασμό μιας ιστοσελίδας. Για παράδειγμα μπορούμε εύκολα και γρήγορα να προσπελάσουμε όλα τα στοιχεία μιας ιστοσελίδας, να χρησιμοποιήσουμε AJAX calls ή απλά να χρωματίσουμε τις άρτιες και περιττές γραμμές ενός table με διαφορετικό χρώμα. Δεν είναι τυχαίο ότι το σύνθημα της JQuery είναι: Write less, do more!
Έτσι ο προγραμματιστής έχει εύκολα και γρήγορα εργαλεία που του επιτρέπουν τη δημιουργία δυναμικών ιστοσελίδων, με πλήρη έλεγχο του κώδικα.

Εμείς χρησιμοποιούμε τη JQuery μαζί με το JQuery UI για να χρησιμοποιήσουμε την data web service findstudents μέσω μιας ιστοσελίδας, όπως θα δούμε αργότερα. Για να χρησιμοποιήσετε την JQuery πρέπει να έχετε γνώση πάνω στα θέματα HTML, HTML DOM, XML, JavaScript, AJAX και λίγο XPath.

Τα βασικά χαρακτηριστικά της είναι τα εξής:

1. Επιλογή των αντικειμένων του HTML DOM (Document Object Model) μέσω της Sizzle selector engine. Μέσω αυτής μπορούμε με λίγο κώδικα να επιλέξουμε όποιο αντικείμενο του DOM θέλουμε, γράφοντας με μορφή που θυμίζει XPath.
2. Διάσχιση (traversal) και μεταβολή (modification) του DOM με υποστήριξη για CSS 1,2,3.
3. Διαχείριση των events
4. Διαχείριση των CSS φύλλων και ιδιοτήτων (style attribute) των HTML elements.
5. Εύκολη δημιουργία κλήσεων AJAX
6. Επεκτασιμότητα με τη χρήση και κατασκευή Plug-in.
7. Υποστήριξη όλων των web browser (cross browser support)

5.4.1 Element selection

Η επιλογή των στοιχείων – αντικειμένων του DOM στην JQuery γίνεται μέσω της μηχανής Sizzle. Το συντακτικό μοιάζει πολύ με XPath. Θα δούμε μερικά παραδείγματα για να αποδείξουμε την απλότητα της:

1. $(“p”) επιλέγει όλα τα <p> html στοιχεία
2. $(“p.bob”) επιλέγει όλα τα <p> html στοιχεία με class=”bob”
3. $(“p#nona”) επιλέγει το πρώτο <p> html στοιχείο με id=”nona”
4. $(“#aeras”) επιλέγει html στοιχείο με id=”aeras”
5. $(“.aeras”) επιλέγει όλα τα html στοιχεία με class=”aeras”
6. $(“[href]”) επιλέγει όλα τα html στοιχεία που έχουν href attribute
7. $(“p:first”) επιλέγει το πρώτο <p> html στοιχείο
8. $(“p:last”) επιλέγει το τελευταίο <p> html στοιχείο
9. $(“tr:even) επιλέγει τις ζυγές γραμμές ενός html table
10. $(“tr:odd) επιλέγει τις ζυγές περιττές γραμμές ενός html table
11. $(“ul li:eq(2)”) επιλέγει το 3^{o} στοιχείο μιας λίστας (το index ξεκινά από το 0)
12. $(“ul li:gt(2)”) επιλέγει τα στοιχεία μιας λίστας με index > 2
13. $(“ul li:lt(2)”) επιλέγει τα στοιχεία μιας λίστας με index < 2

Αν θέλαμε να επιλέξουμε τα στοιχεία των παραπάνω παραδειγμάτων χωρίς τη χρήση της JQuery, θα έπρεπε να γράφουμε αρκετό κώδικα σε JavaScript, ο οποίος τις περισσότερες φορές επαναλαμβάνεται, με αποτέλεσμα να μειώνεται η παραγωγικότητα. Οι προγραμματιστές της JQuery ομαδοποίησαν τις ποιο συχνές λειτουργίες και μας τις δίνουν έτοιμες σε μια ελαφριά βιβλιοθήκη.

5.4.2 JQuery and AJAX

Η JQuery μας βοηθά να κάνουμε κλήσεις AJAX (Asynchronous JavaScript and XML) με τις έτοιμες συναρτήσεις που μας δίνει χωρίς να μας νοιάζει σε ποιον browser θα εκτελεστούν. Η cross browser υλοποίηση της γλυτώνει τους προγραμματιστές από πολλές ώρες προγραμματισμού που χρειαζόταν έως τώρα για να μπορούμε να κάνουμε μια ιστοσελίδα να εμφανίζεται ίδια σε όλους τους

γνωστούς browsers. Ο κάθε browser (Internet Explorer, Mozilla Firefox, Opera, Safari, Chrome) έχει ιδιαιτερότητες ως προς την υλοποίηση του DOM και του AJAX framework.Το DOM και το AJAX τα οποία παρόλο έχουν προτυποποιηθεί από το W3C (το AJAX είναι σε φάση πρότασης) δεν υλοποιούνται με τον ίδιο τρόπο από τους browsers με αποτέλεσμα να αποτελούν βραχνά για τους προγραμματιστές διαδικτυακών εφαρμογών.

Για να καταλάβουμε την βοήθεια που μας δίνει η JQuery θα παραθέσουμε ένα παράδειγμα: Θέλουμε να καλέσουμε μέσω AJAX με μέθοδο HTTP GET την ιστοσελίδα gethint.asp η οποία παίρνει ως όρισμα ένα γράμμα και μας επιστρέφει μια πρόταση με ένα όνομα που ξεκινά με αυτό το γράμμα. Έτσι αν θέλουμε όλα τα ονόματα που αρχίζουν από S θα καλέσουμε την ιστοσελίδα ως εξής: gethint.asp?q=S

Η ακόλουθη υλοποίηση είναι γραμμένη με απλή JavaScript μέσα σε μια συνάρτηση. Αν καλέσουμε τη συνάρτηση με όρισμα το S, αυτή θα καλέσει την ιστόσελίδα gethint.asp?q= με το string που έχουμε περάσει στη συνάρτηση και θα γράψει την απάντηση από την ιστοσελίδα στο στοιχείο με id=”txtHint” της δικής μας ιστόσελίδας. Παρατηρούμε ότι ο κώδικας διαφοροποιείται για τις παλιές εκδόσεις 5 και 6 του Internet Eplorer (γραμμή 11,12) ενώ για τους υπόλοιπους browsers θα τρέξουν οι γραμμές 6,7,8. Το παράδειγμα πιάνει 23 γραμμές κώδικα με τα σχόλια.

```
1. function showHint(str){
2.     if (str.length==0){
3.         document.getElementById("txtHint").innerHTML="";
4.         return;
5.     }
6.     if (window.XMLHttpRequest){
7.         // code for IE7+, Firefox, Chrome, Opera, Safari
8.         xmlhttp=new XMLHttpRequest();
9.     }
10.     else{// code for IE6, IE5
11.         xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
12.     }
13.
14.     xmlhttp.onreadystatechange=function(){
15.         if (xmlhttp.readyState==4 && xmlhttp.status==200){
16.
document.getElementById("txtHint").innerHTML=xmlhttp.responseText;
17.             alert('Load was performed.');
18.         }
19.     }
20.
21.     xmlhttp.open("GET","gethint.asp?q="+str,true);
22.     xmlhttp.send();
23. }
```

Για να κάνουμε το ίδιο ακριβώς πράγμα χρησιμοποιώντας την JQuery χρειαζόμαστε να γράψουμε μόνο 6 γραμμές όπως φαίνεται στο ακόλουθο παράδειγμα κώδικα. Επίσης δεν χρειάζεται να ανησυχούμε για το cross browser support. Η JQuery το φροντίζει για εμάς. Το μόνο που χρειάζεται να μάθουμε είναι το συντακτικό της.

```
1. function showHint(str){
```

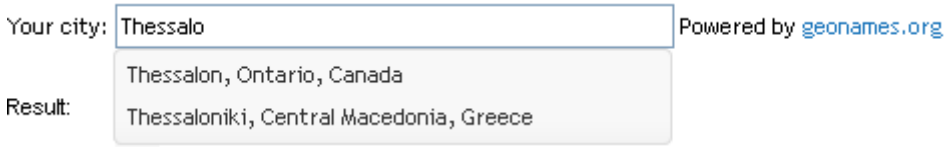
```
2.      $.get('gethint.asp?q='+str, function(data) {
3.          $('#txtHint').html(data);
4.          alert('Load was performed.');
5.      });
6.  }
```

5.4.3 JQuery UI

Σχήμα 5.2 Το λογότυπο του JQuery UI

Το JQuery UI (user interface) project είναι μια βιβλιοθήκη δημιουργημένη με JQuery με σκοπό την παροχή γραφικών, έτοιμων widgets και themes για μια ιστοσελίδα. Το JQuery UI σε συνδυασμό με την μαμά JQuery βοηθούν στην κατασκευή στιβαρών ιστοσελίδων, πλουσίων σε γραφικά και λειτουργικότητα.

Στην πτυχιακή εργασία χρησιμοποιούμε κυρίως δύο widgets τα AutoComplete και το DatePicker. Το AutoComplete widget (Σχήμα 5.3) το χρησιμοποιούμε για την εύρεση φοιτητών μέσω της web service findstudents την οποία θα αναλώσουμε σε επόμενη ενότητα. Το widget αυτό μας δίνει προτάσεις για την αναζήτησή μας ενώ πληκτρολογούμε, όπως γίνεται στις γνωστές μηχανές αναζήτησης Google, Yahoo.



Σχήμα 5.3 Το autocomplete widget κατά την ανάλωση της web service geonames.org

Το Date Picker widget (Σχήμα 5.4) το χρησιμοποιούμε για να επιλέγουμε γρήγορα και εύκολα ημερομηνίες στην ιστοσελίδα μας. Παρέχει έξυπνες συναρτήσεις και παραμέτρους για τον τρόπο εμφάνισης της ημερομηνίας και των μηνών.

Σχήμα 5.4 Το datepicker widget του JQuery UI

5.5 SOAP over HTTP

Στην ενότητα αυτή θα καλέσουμε μέσω του SOAP endpoint την μέθοδο SEARCHSTUDENTS της web service findstudents με τη χρήση του cURL. Κατόπιν θα δείξουμε πώς μπορούμε να κάνουμε το ίδιο πράγμα με τη χρήση του Web Services Explorer.

Το πρωτόκολλο SOAP λειτουργεί με ανταλλαγή μηνυμάτων XML τα οποία στέλνονται μέσω του πρωτοκόλλου HTML στον παγκόσμιο ιστό. Καταρχήν θα ετοιμάσουμε ένα μήνυμα αίτησης (request) προς τη web service το οποίο θα αποθηκεύσουμε στο αρχείο FoititisRequest.xml. Κατόπιν, με το cURL θα αποστείλουμε αυτό το μήνυμα μέσω HTTP και θα λάβουμε την απάντηση μορφοποιημένη σε XML. Έστω ότι θέλουμε να αναζητήσουμε τους φοιτητές που το όνομά τους ή το επίθετό τους ή το username περιέχει το string “chr”.

Για να προσπελάσουμε το SOAP endpoint στις Data Web Services θα πρέπει να γνωρίζουμε το URL του, το οποίο έχει την εξής μορφή:

http(s)://<server>:<port>/<contextRoot>/<services>/<ServiceName>

Για την web service findstudents το url του endpoint είναι το ακόλουθο: http://localhost:8080/Project1findstudents/services/findstudents.

Το μήνυμα SOAP που θα αποθηκεύσουμε στο αρχείο FoititisRequest.xml θα είναι το ακόλουθο:

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://it.teithe.findstudents"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <soapenv:Body>
        <q0:SEARCHSTUDENTS>
              <NAMEIN>chr</NAMEIN>
          </q0:SEARCHSTUDENTS>
      </soapenv:Body>
</soapenv:Envelope>
```

Ανοίγουμε ένα νέο command line prompt (Εναρξη>Εκτέλεση>cmd) και πληκτρολογούμε τα εξής:

```
curl.exe -d @FoititisRequest.xml
         -H "Content-Type: text/xml"
         -H "SOAP-Action: \"http://it.teithe.findstudents\""
         -v
http://localhost:8080/Project1findstudents/services/findstudents
```

Η παράμετροι στην κλήση εξηγούνται ως εξής:

- -d @”αρχείο” : το όνομα του αρχείου που περιέχει το μήνυμα SOAP

- -H : η κεφαλίδα (header) του μηνύματος HTTP που περιγράφει τον τύπο MIME (text/html) που πρόκειται να μεταφέρει καθώς και το SOAP action (http://it.teithe.findstudents) όπως αυτό περιγράφεται στο WSDL αρχείο της data web service.
- -v : verbose mode που μας επιστρέφει μηνύματα στο stdout για κάθε πράξη στην οποία προβαίνει το cURL.
- Τέλος γράφουμε το URL του endpoint της data web service

Η απάντηση που θα λάβουμε είναι η εξής:

```
* About to connect() to localhost port 8080 (#0)
* Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 8080 (#0)
> POST /Project1findstudents/services/findstudents HTTP/1.1
>   User-Agent:   curl/7.18.2   (i386-pc-win32)   libcurl/7.18.2
OpenSSL/0.9.8h
libssh2/0.18
> Host: localhost:8080
> Accept: */*
> Content-Type: text/xml
> SOAPAction:"http://it.teithe.findstudents"
> Content-Length: 389
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Content-Type: text/xml;charset=utf-8
< Transfer-Encoding: chunked
< Date: Sun, 28 Jun 2009 04:21:21 GMT
<
<?xml version="1.0" encoding="UTF-8"?><soapenv:Envelope
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:Body> <SEARCHSTUDENTSResponse
xmlns:ns1=http://it.teithe.findstudents><rowset><row>
<USERNAME>chrandr</USERNAME><NAME>Christos</NAME>
<SURNAME>Andrianos</SURNAME>
<URL>http://www.it.teithe.gr/~chrandr</URL>
<EMAIL>chrandr@it.teithe.gr</EMAIL></row> <row>
<USERNAME>chargir</USERNAME><NAME>Christos</NAME>
<SURNAME>Argiriadis</SURNAME>
<URL>http://www.it.teithe.gr/~chargir</URL>
<EMAIL>chargir@it.teithe.gr</EMAIL></row> <row>
<USERNAME>chrath</USERNAME><NAME>Chrisa</NAME>
<SURNAME>Athanasiadou</SURNAME>
<URL>http://www.it.teithe.gr/~chrath</URL>
<EMAIL>chrath@it.teithe.gr</EMAIL> </row></rowset>
</SEARCHSTUDENTSResponse></soapenv:Body></soapenv:Envelope>
```

Παρατηρούμε ότι η απάντηση είναι ένα HTTP μήνυμα που περιέχει απόκριση (response) από την web service σε μορφή XML. Τα δεδομένα που επέστρεψε η stored procedure όπως ήταν μέσα στον πίνακα της βάσης δεδομένων

username	name	Surname	url	Email

chrandr	Christos	Andrianos	http://www.it.teithe.gr/~chrandr	chrandr@it.teithe.gr
chrath	Chrisa	Athanasiadou	http://www.it.teithe.gr/~chrath	chrath@it.teithe.gr
chargir	Christos	Argiriadis	http://www.it.teithe.gr/~chargir	chargir@it.teithe.gr

επιστρέφονται σε XML ως εξής

```
<rowset>
    <row>
        <USERNAME>chrandr</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Andrianos</SURNAME>
        <URL>http://www.it.teithe.gr/~chrandr</URL>
        <EMAIL>chrandr@it.teithe.gr</EMAIL>
    </row>
    <row>
        <USERNAME>chargir</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Argiriadis</SURNAME>
        <URL>http://www.it.teithe.gr/~chargir</URL>
        <EMAIL>chargir@it.teithe.gr</EMAIL>
    </row>
    <row>
        <USERNAME>chrath</USERNAME>
        <NAME>Chrisa</NAME>
        <SURNAME>Athanasiadou</SURNAME>
        <URL>http://www.it.teithe.gr/~chrath</URL>
        <EMAIL>chrath@it.teithe.gr</EMAIL>
    </row>
</rowset>
```

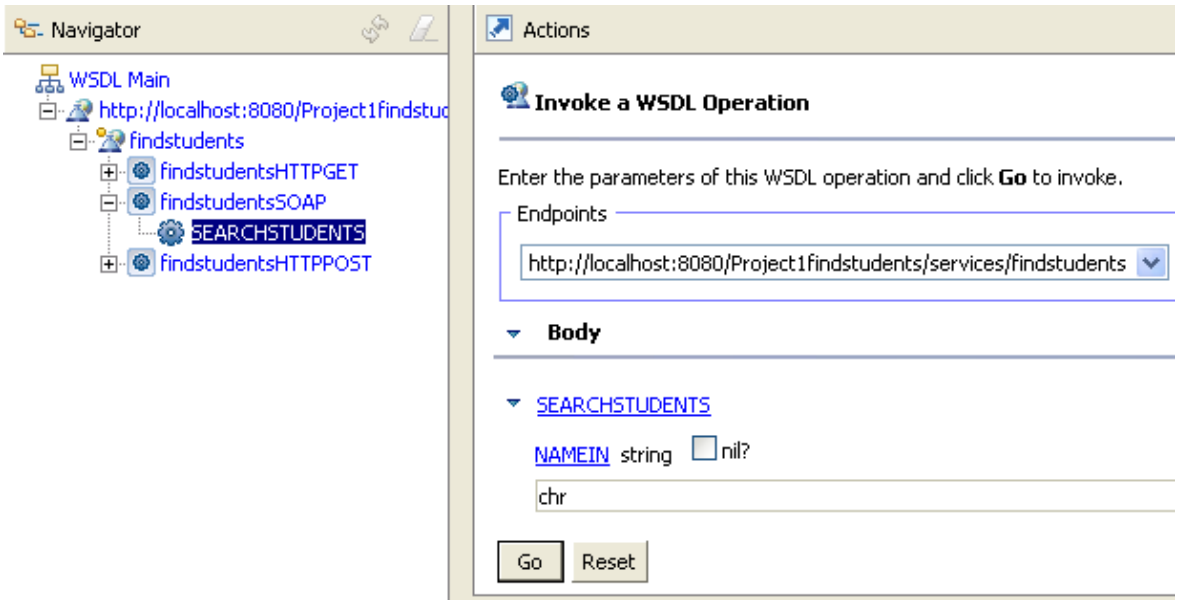
Σημείωση:
Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

Αν θέλουμε να κάνουμε την ίδια κλήση με τον Web Services Explorer, στον Data Project Explorer του Data Studio κάνουμε δεξί κλικ πάνω στην web service findstudents και επιλέγουμε το *Launch Web Services Explorer*.

Σχήμα 5.5 Δοκιμή της data web service στο Data Studio με χρήση του WS Explorer

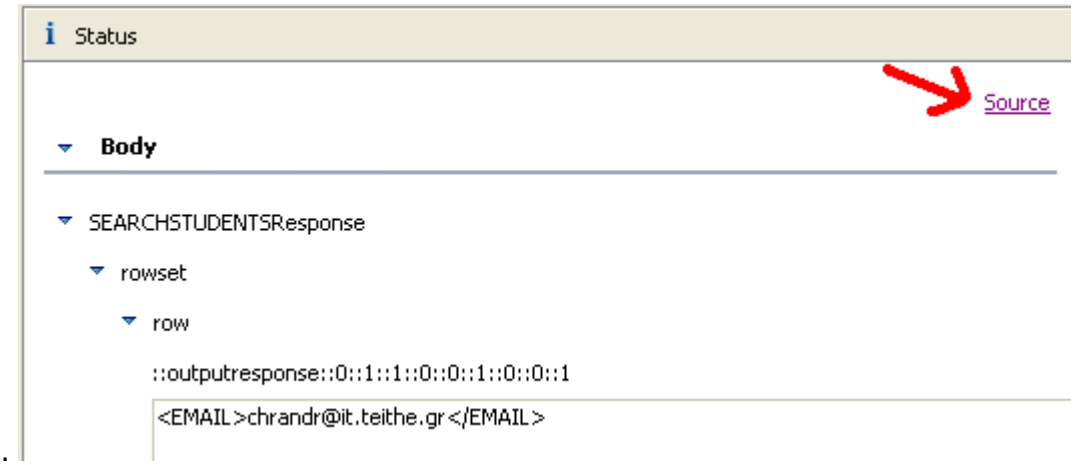
Ακολούθως πλοηγούμαστε στο δέντρο του Navigator (Σχήμα 5.6 αριστερά), επιλέγουμε το SOAP endpoint και επιλέγουμε την μέθοδο – stored procedure SEARCHSTUDENTS. Με την επιλογή, το παράθυρο Actions μεταβάλλεται και εμφανίζει (σχήμα. 6 δεξιά) το πλήρες URL του endpoint που χρησιμοποιήσαμε και

στο cURL και ένα από κάτω ένα textbox για να εισάγουμε το NAMEIN προς αναζήτηση και πιέζουμε το κουμπί *Go*.



Σχήμα 5.6 Ο WS Explorer με SOAP endpoint

Αμέσως αποστέλλεται ένα μήνυμα SOAP HTTP και η απάντηση μας εμφανίζεται στο Σχήμα 5.7 Αν κάνουμε κλικ στο *Source*, θα μας εμφανιστεί το μήνυμα SOAP request (αίτηση) και το response (απόκριση) όπως φαίνεται στο Σχήμα 5.8



Σχήμα 5.7 Η απάντηση της web service δενδρική απεικόνιση

Σχήμα 5.8 Τα πραγματικά μηνύματα αίτησης και απάντησης SOAP

Η SOAP σύνδεση (binding) της Web Service μπορεί να χρησιμοποιηθεί από διάφορες γλώσσες προγραμματισμού που υλοποιούν το πρωτόκολλο SOAP όπως η Java.

5.6 REST HTTP GET

Το HTTP GET binding του REST endpoint, αποτελεί τον πιο απλό τρόπο με τον οποίο μπορούμε να καλέσουμε μια data web service. Αυτό οφείλεται στο γεγονός ότι αρκεί να πληκτρολογήσουμε ένα URL σε έναν browser με τις κατάλληλες παραμέτρους και να έχουμε άμεσα μια απάντηση. Όπως έχει αναφερθεί, στην αρχιτεκτονική REST η σύνδεση δεν κρατά κατάσταση (stateless) Έτσι η κλήση προς κάποιον πόρο, ο οποίος στην προκειμένη περίπτωση είναι μια data web service, πρέπει να περιέχει όλα τα στοιχεία που απαιτούνται για να προσδιορίσουν τον πόρο. Η κλήση προς τη web service γίνεται μέσω πρωτοκόλλου HTTP με χρήση της μεθόδου GET.

Στο binding HTTP GET έχουμε τον ίδιο τρόπο κλήσης, αλλά τα δεδομένα μπορούν να επιστρέψουν είτε σε μορφοποίηση (output format) XML είτε σε JSON. Έτσι για την καλύτερη κατανόηση θα ορίσουμε δύο υποκατηγορίες ανάλογα με το output format, XML και JSON. Ακόμη για να χρησιμοποιήσουμε μέσω AJAX το JSON output δημιουργήσαμε και JSONP output έμμεσα, με τη χρήση της PHP.

Σημείωση:
Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

5.6.1 REST HTTP GET XML output

Για να χρησιμοποιήσουμε το binding αυτό χρειαζόμαστε μόνο το URL του REST endpoint. Σε αυτό προσθέτουμε το όνομα της μεθόδου που καλούμε, έπειτα ο χαρακτήρας ? και τις παραμέτρους που περνούμε στη μέθοδο χωρισμένες με τον χαρακτήρα &.

Η γενική μορφή του REST endpoint URL για τις IBM Data Web Services είναι :

http(s)://<server>:<port>/<contextRoot>/<services>/<ServiceName>/<methodName>?PARAM1=value&PARAM2=value2&….&PARAMn=valueN

Για παράδειγμα εμείς θέλουμε να χρησιμοποιήσουμε την μέθοδο searchstudents της υπηρεσίας findstudents που παίρνει ως παράμετρο το κλειδί αναζήτησης namein.

Το URL αναζήτησης για το string “chr” θα είναι το ακόλουθο:

http://localhost:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=chr

Μπορούμε να το δοκιμάσουμε με τη χρήση του cURL γράφοντας:

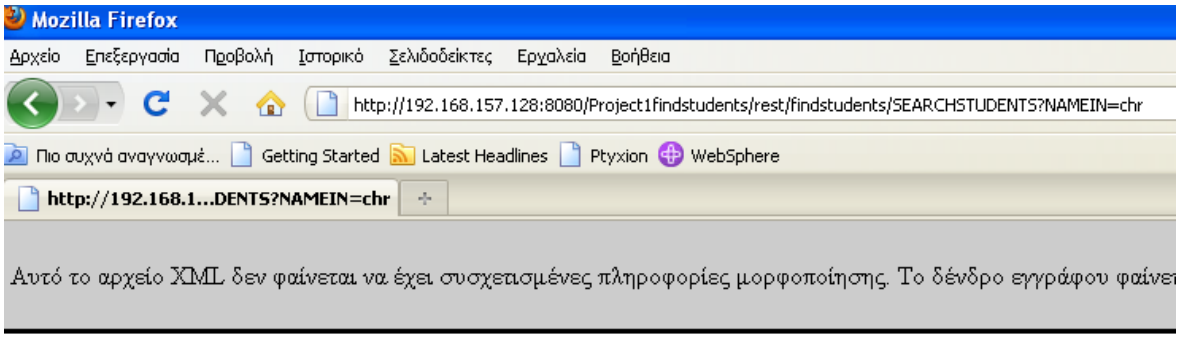
curl "http://127.0.0.1:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=chr"

Η απάντηση θα επιστρέψει σε XML μορφή όπως φαίνεται ακολούθως:

<?xml version="1.0" encoding="UTF-8"?><ns1:SEARCHSTUDENTSResponse xmlns:ns1="http://it.teithe.findstudents" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"><rowset><row><USERNAME>chrandr</USERNAME><NAME>Christos</NAME><SURNAME>Andrianos</SURNAME><URL>http://www.it.teithe.gr/~chrandr</URL><EMAIL>chrandr@it.teithe.gr</EMAIL></row>
……
</rowset></ns1:SEARCHSTUDENTSResponse>

Τα δεδομένα είναι μορφοποιημένα με τον ίδιο τρόπο όπως μορφοποιούνται μέσα στο σώμα (body) της απάντησης SOAP (βλέπε ενότητα 4.5)

Μπορούμε να ανοίξουμε ένα παράθυρο web browser και να πληκτρολογήσουμε το ίδιο URL και να έχουμε την ίδια απάντηση (Σχήμα 5.9).




```
− <ns1:SEARCHSTUDENTSResponse>
  − <rowset>
    − <row>
        <USERNAME>chrandr</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Andrianos</SURNAME>
        <URL>http://www.it.teithe.gr/~chrandr</URL>
        <EMAIL>chrandr@it.teithe.gr</EMAIL>
      </row>
    − <row>
        <USERNAME>chargir</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Argiriadis</SURNAME>
```


Σχήμα 5.9 Ανάλωση σε browser REST HTTP GET με XML output

Σημείωση:
Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

5.6.2 REST HTTP GET JSON output

Για να χρησιμοποιήσουμε το binding αυτό χρειαζόμαστε μόνο το URL του REST endpoint. Σε αυτό προσθέτουμε το όνομα της μεθόδου που καλούμε, έπειτα ο χαρακτήρας ? και τις παραμέτρους που περνούμε στη μέθοδο χωρισμένες με τον χαρακτήρα & Στις παραμέτρους προσθέτουμε ακόμη και την _outputFormat=JSON. Με αυτόν τον τρόπο διατάζουμε την web service να μας επιστρέψει τα δεδομένα σε JSON.

Η γενική μορφή του REST endpoint URL για τις IBM Data Web Services είναι :

http(s)://<server>:<port>/<contextRoot>/<services>/<ServiceName>/<methodName>?PARAM1=value&PARAM2=value2&….&PARAMn=valueN

Για παράδειγμα εμείς θέλουμε να καλέσουμε την μέθοδο searchstudents της υπηρεσίας findstudents που παίρνει ως παράμετρο το κλειδί αναζήτησης namein.

Το URL αναζήτησης για το string “chr” θα είναι το ακόλουθο:

http://localhost:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=chr_outputFormat=JSON

Μπορούμε να το δοκιμάσουμε με τη χρήση του cURL γράφοντας:

curl "http://127.0.0.1:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=chr&_outputFormat=JSON"

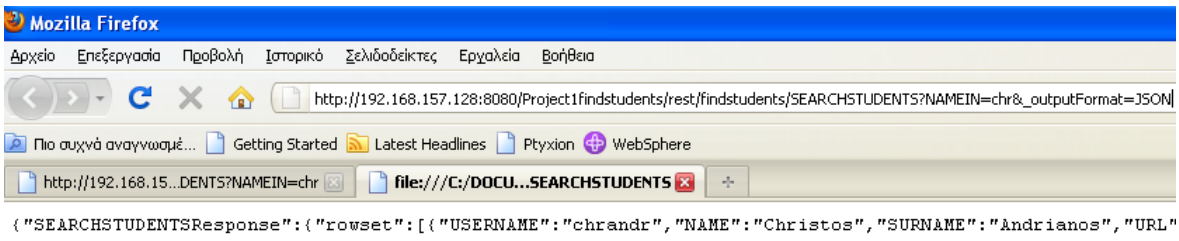
Η απάντηση έχει την εξής μορφή:

```
{"SEARCHSTUDENTSResponse":
    {"rowset": [
        {"USERNAME":"chrandr",
        "NAME":"Christos",
        "SURNAME":"Andrianos",
        "URL":"http:\/\/www.it.teithe.gr\/~chrandr",
        "EMAIL":"chrandr@it.teithe.gr"
        },
        {"USERNAME":"chargir",
        "NAME":"Christos",
        "SURNAME":"Argiriadis",
        "URL":"http:\/\/www.it.teithe.gr\/~chargir",
        "EMAIL":"chargir@it.teithe.gr"
        }]}}
```

Για να γίνει πιο κατανοητή η αντιστοιχία παραθέτουμε την ίδια απάντηση σε XML.

```
<rowset>
    <row>
        <USERNAME>chrandr</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Andrianos</SURNAME>
        <URL>http://www.it.teithe.gr/~chrandr</URL>
        <EMAIL>chrandr@it.teithe.gr</EMAIL>
    </row>
    <row>
        <USERNAME>chargir</USERNAME>
        <NAME>Christos</NAME>
        <SURNAME>Argiriadis</SURNAME>
        <URL>http://www.it.teithe.gr/~chargir</URL>
        <EMAIL>chargir@it.teithe.gr</EMAIL>
    </row>
</rowset>
```

Μπορούμε να δοκιμάσουμε το URL και σε έναν browser. Το αποτέλεσμα είτε θα εμφανιστεί κατευθείαν (Σχήμα 5.10) είτε θα σας ζητηθεί να κατεβάσετε ένα αρχείο με κατάληξη .json.



Σχήμα 5.10 Ανάλωση σε browser REST HTTP GET με JSON output

Σημείωση:
Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

5.6.3 REST HTTP GET JSONP output

Οι data web services δεν μας παρέχουν binding για το REST endpoint που να απαντά σε μορφοποίηση JSONP. Για να μπορέσουμε να αποφύγουμε τον σκόπελο του Same Origin Policy, δημιουργήσαμε ένα server side script γραμμένο σε PHP που παίρνει τις ίδιες παραμέτρους με το HTTP GET JSON, θα καλεί την υπηρεσία και θα μας επιστρέφει JSONP.
Η λογική είναι απλή:

- Καλούμε το REST HTTP GET endpoint της υπηρεσίας μέσω του PHP script
- Προσθέτουμε το padding
- Το επιστρέφουμε στο χρήστη

Με αυτόν τον τρόπο για να δοκιμάσουμε το JSONP δημιουργήσαμε μια ιστοσελίδα που καλεί με AJAX το php script (το οποίο καλεί την data web service) και έχει τη μορφή φόρμας αναζήτησης η οποία χρησιμοποιεί JQuery και το autocomplete widget της JQuery UI (Σχήμα 5.11).

Σχήμα 5.11 Η ανάλωση της υπηρεσίας από ιστοσελίδα μέσω JSONP

Σημείωση:
Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

5.7 REST HTTP POST

Το HTTP POST binding του REST endpoint, είναι ο δεύτερος τρόπος με τον οποίο μπορούμε να καλέσουμε μια data web service μέσω REST. Η κλήση προς τη web service γίνεται μέσω πρωτοκόλλου HTTP με χρήση της μεθόδου POST.Το Data Studio μας παρέχει τρεις τρόπους για τη μορφοποίηση των δεδομένων που θα αποσταλούν και θα ληφθούν.

Η γενική μορφή του REST endpoint URL για τις IBM Data Web Services είναι :

http(s)://<server>:<port>/<contextRoot>/<services>/<ServiceName>/<methodNam
e>

5.7.1 REST HTTP POST XML output

Το binding αυτό χρησιμοποιείται από client applications, JavaScript. Για να το καλέσουμε με το curl γράφουμε:

```
curl.exe -d "NAMEIN=chr"
         -v
http://localhost:8080/Project1findstudents/rest/findstudents/
SEARCHSTUDENTS
```

Η παράμετροι στην κλήση εξηγούνται ως εξής:

- -d οι παράμετροι που πρέπει να περαστούν στη μέθοδο χωρισμένες με & ή γράφουμε @"αρχείο" : το όνομα του αρχείου που περιέχει τις παραμέτρους
- -v : verbose mode που μας επιστρέφει μηνύματα στο stdout για κάθε πράξη στην οποία προβαίνει το cURL.
- Τέλος γράφουμε το URL του endpoint της data web service

Η απάντηση που θα λάβουμε περιέχει δεδομένα μορφοποιημένα σε XML όπως στην ενότητα 4.6.1:

```
* About to connect() to 127.0.0.1 port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
>    POST    /Project1findstudents/rest/findstudents/SEARCHSTUDENTS
HTTP/1.1
>    User-Agent:    curl/7.20.1    (i386-pc-win32)    libcurl/7.20.1
OpenSSL/0.9.8n zlib/1.2
.5
> Host: 127.0.0.1:8080
> Accept: */*
> Content-Length: 10
> Content-Type: application/x-www-form-urlencoded
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Cache-Control: no-cache, no-store, max-age=0
< Expires: Thu, 01 Jan 1970 00:00:01 GMT
< Content-Type: text/xml;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Sat, 25 Sep 2010 19:31:08 GMT
<
<?xml  version="1.0"  encoding="UTF-8"?><ns1:SEARCHSTUDENTSResponse
xmlns:ns1="htt
p://it.teithe.findstudents"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
"><rowset><row><USERNAME>chrandr</USERNAME><NAME>Christos</NAME><S
URNAME>Andrian
```

```
os</SURNAME><URL>http://www.it.teithe.gr/~chrandr</URL><EMAIL>chra
ndr@it.teithe.
gr</EMAIL></row><row>
…..
….
</row></rowset></ns1:SEARCHSTUDENTSResponse>* C
onnection #0 to host 127.0.0.1 left intact
* Closing connection #0
```

Σημείωση:
Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

5.7.2 REST HTTP POST URL-encoded

Το binding αυτό χρησιμοποιείται όταν έχουμε φόρμες HTML που αποστέλλουν τα δεδομένα τους με μέθοδο POST (method=”post”). Τα δεδομένα επιστρέφουν με XML μορφή, απλά αποστέλλονται κωδικοποιημένα με URL-encoding.

Όποιοι χαρακτήρες δεν ανήκουν στο ASCII, μετατρέπονται σε μια ακολουθία χαρακτήρων με αρχικό τον χαρακτήρα επί τις εκατό % ακολουθούμενα από δύο δεκαεξαδικούς χαρακτήρες. Αυτό οφείλεται στον ορισμό του URL (RFC 1738) που δεν επιτρέπει μη ASCII χαρακτήρες σε URL. Πριν από ένα χρόνο περίπου ανακοινώθηκε ότι θα αρχίσουν να χρησιμοποιούνται Unicode χαρακτήρες στα URL. Αυτό είναι ένα καλό βήμα, αλλά υπάρχουν τρύπες ασφαλείας. Έτσι το url-encoding συνεχίζει να χρησιμοποιείται.

Αν θέλουμε να καλέσουμε το HTTP POST URL-encoded με το cURL θα γράψουμε την ίδια εντολή με την προηγούμενη ενότητα. Θα προσθέσουμε επίσης και επιπλέον δεδομένα στην κεφαλίδα (header) του HTTP μηνύματος για να πούμε ότι ο MIME type του μηνύματος είναι url encoded. Έτσι γράφουμε στο command line prompt:

```
curl.exe -d "NAMEIN=chr"
         -v
        -H "Content-Type:application/x-www-form-urlencoded"
http://localhost:8080/Project1findstudents/rest/findstudents/
SEARCHSTUDENTS
```

Παρατηρούμε ότι έχουμε προσθέσει μια παράμετρο –H στο cURL, η οποία προσθέτει στον header του μηνύματος HTTP το contet-type που είναι x-www-form-urlencoded. Η απάντηση από την data web service θα είναι ίδια με την απάντηση που πήραμε στην προηγούμενη ενότητα (4.7.1). Αν πάντως θέλουμε να χρησιμοποιήσουμε URL encoded links θα πρέπει να μεταβάλλουμε κάποιες ρυθμίσεις στον WebSphere server προκειμένου να το επιτρέπει. Αυτές οι ρυθμίσεις θα περιγραφούν σε επόμενη ενότητα.

Σημείωση:

Οι τιμές SQL NULL αντιμετωπίζονται ως απούσες. Αυτό σημαίνει ότι παράμετροι που δεν εμφανίζονται στα ζεύγη key/value θεωρούνται ως NULL. Μια παράμετρος χωρίς τιμή θεωρείται ότι στέλνει ένα κενό string και ΟΧΙ NULL.

5.7.3 REST HTTP POST JSON output

Το τελευταίο binding για την HTTP POST έκδοση του REST endpoint παίρνει ως είσοδο δεδομένα με μορφοποίηση JSON και απαντά με JSON. Το binding αυτό σύμφωνα με τους προγραμματιστές της IBM ως το πλέον κατάλληλο για χρήση από JavaScript εφαρμογές που χρησιμοποιούν AJAX, δηλαδή το XMLHttpRequest. Οι κανόνες δημιουργίας ενός JSON μηνύματος αίτησης (request message) για τις Data Web Services είναι οι εξής:

{ “<methodName>” : { “<parameter1>”:<value1>, “<parameter2>”:<value2>, …}}

Έτσι για να καλέσουμε την searchstudents της υπηρεσίας findstudents το μήνυμα θα είναι το εξής:

{ “SEARCHSTUDENTS”: { “NAMEIN” : “chr” }}

Για να χρησιμοποιήσουμε με το cURL το HTTP POST JSON binding, γράφουμε μέσα σε ένα αρχείο το παραπάνω JSON μήνυμα και το αποθηκεύουμε. Έστω ότι το αρχείο είναι το studentJSON.txt

Η κλήση του cURL προς τη data web service είναι η ακόλουθη:

```
curl.exe -d @"studentJSON.txt"
         -H "Content-Type:application/json;charset=utf-8"
         -v
http://localhost:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS
```

Η απάντηση θα είναι η ακόλουθη και θα εμφανιστεί στο command prompt. Παρατηρούμε ότι τα δεδομένα επιστρέφουν και αυτά μορφοποιημένα σε JSON.

```
* About to connect() to 127.0.0.1 port 8080 (#0)
*   Trying 127.0.0.1... connected
* Connected to 127.0.0.1 (127.0.0.1) port 8080 (#0)
>    POST    /Project1findstudents/rest/findstudents/SEARCHSTUDENTS
HTTP/1.1
>    User-Agent:    curl/7.20.1    (i386-pc-win32)    libcurl/7.20.1
OpenSSL/0.9.8n zlib/1.2
.5
> Host: 127.0.0.1:8080
> Accept: */*
> Content-Type:application/json;charset=utf-8
> Content-Length: 37
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Cache-Control: no-cache, no-store, max-age=0
```

```
< Expires: Thu, 01 Jan 1970 00:00:01 GMT
< Content-Type: application/json;charset=UTF-8
< Content-Length: 2788
< Date: Sat, 25 Sep 2010 19:39:27 GMT
<
{"SEARCHSTUDENTSResponse":{"rowset":[{"USERNAME":"chrandr","NAME":
"Christos","SU
RNAME":"Andrianos","URL":"http:\/\/www.it.teithe.gr\/~chrandr","EM
AIL":"chrandr@
it.teithe.gr"},{"USERNAME":"chargir","NAME":"Christos","SURNAME":"
Argiriadis","U
RL":"http:\/\/www.it.teithe.gr\/~chargir","EMAIL":"chargir@it.teit
he.gr"},{"USER
NAME":"chrath","NAME":"Chrisa","SURNAME":"Athanasiadou","URL":"htt
p:\/\/www.it.t
eithe.gr\/~chrath","EMAIL":"chrath@it.teithe.gr"}]}}
* Connection #0 to host 127.0.0.1 left intact
* Closing connection #0
```

Σημείωση πάνω στη μορφοποίηση των δεδομένων JSON:
Οι τύποι δεδομένων που περνιούνται ως παράμετροι στο μήνυμα JSON πρέπει να ακολουθούν τις προδιαγραφές του προτύπου JSON. Έτσι οι τύποι δεδομένων time, date, timestamp πρέπει να εισάγονται με βάση τη μορφοποίηση XSD: xsd:date, xsd:time, xsd:timestamp. Δυαδικά δεδομένα πρέπει να είναι μορφοποιημένα σε Base64 encoded strings. Οι τιμές SQL NULL πρέπει να αναπαρίστανται με το JSON null.

5.7.4 Ρύθμιση WebSphere για URL encoded links

Για να μπορέσουμε να περάσουμε URL encoded links πρέπει να προβούμε πρώτα σε κάποιες ρυθμίσεις. Ο WebSphere εκ προεπιλογής δεν τα υποστηρίζει, αλλά πειράζοντας το configuration file του application server, τον διατάζουμε να το κάνει.

Αν χρησιμοποιούμε τον Apache Tomcat ως application server, τότε προσθέτουμε το xml attribute URIEncoding=”UTF-8” στις ρυθμίσεις <Connector> μέσα στο αρχείο server.xml

Αν χρησιμοποιούμε τον WebSphere Application Server CE, όπως κάνουμε σε αυτήν την πτυχιακή, ο Tomcat αποτελεί web container του WebSphere. Έτσι υπάρχει ένα configuration file στη διαδρομή $WASCE_HOME\var\config\config.xml Με $WASCE_HOME εννούμε το φάκελο εγκατάστασης του server. Μέσα σε αυτό το αρχείο υπάρχει ένα element με υπογραφή <gbean name="TomcatWebConnector"> και μέσα σε αυτό προσθέτουμε ως παιδί του το <attribute name="uriEncoding">UTF-8</attribute>.

Αποθηκεύουμε το αρχείο και κάνουμε restart τον server για να υλοποιηθεί η ρύθμιση.

5.8 Επίλογος

Σε αυτό το κεφάλαιο αναλύσαμε τα bindings που μας προσφέρουν οι Data Web Services για τα SOAP και REST endpoints. Επίσης δείξαμε τον τρόπο ανάλωσης μιας web service μέσω της command line εφαρμογής cURL και μέσω ενός web browser. Τέλος περιγράψαμε τις τεχνικές λεπτομέρειες που πρέπει να προσεχθούν πάνω στη μορφοποίηση των μηνυμάτων αίτησης και απάντησης που χρησιμοποιούνται για την ανάλωση μιας Data Web Service.

6. Κατασκευή ιστοσελίδων που χρησιμοποιούν DWS

6.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναφερθούμε στις Data Web Services που δημιουργήσαμε προκειμένου να δοκιμάσουμε το IBM Data Studio. Επίσης για κάθε web service θα περιγράψουμε τις ιστοσελίδες που υλοποιήσαμε για να καλέσουν τις μεθόδους τους. Τέλος θα εξηγήσουμε πώς μπορείτε να χρησιμοποιήσετε οι ίδιοι το VMware Image που συνοδεύει την πτυχιακή εργασία σε DVD και περιέχει όλα αυτά τα παραδείγματα.

Οι Data Web Services που δημιουργήθηκαν για τους σκοπούς αυτής τη τπυχιακής εργασίας είναι τρεις στον αριθμό και βασίζονται δεδομένα που πήραμε από το ΤΕΙ Πληροφορικής.

1. Η πρώτη web service είναι η searchstudents η οποία έχει μια μέθοδο την SEARCHSTUDENTS η οποία αναζητά φοιτητές.
2. Η δεύτερη είναι η teischedule η οποία μας εμφανίζει τα μαθήματα που λαμβάνουν χώρα στο τμήμα Πληροφορικής σε συγκεκριμένη ώρα και ημερομηνία με βάση το πρόγραμμα του Εαρινού Εξαμήνου 2010.
3. Η τρίτη data web service ονομάζεται OlympicGames και χρησιμοποιεί την ομώνυμη βάση δεδομένων που δίδεται ως παράδειγμα στα μαθήματα Βάσεις Δεδομένων ΙΙ.

Στις επόμενες ενότητες θα δώσουμε μια πιο λεπτομερή περιγραφή των παραπάνω υπηρεσιών καθώς και των ιστοσελίδων που τις χρησιμοποιούν. Και οι τρεις έχουν SOAP και REST endpoints με όλα τα bindings. Εμείς χρησιμοποιούμε μόνο το REST endpoint επειδή η κλάση που υλοποιεί η PHP για το πρωτόκολλο SOAP (SoapClient) αναγνωρίζει τα δεδομένα της απάντησης από την web service ως Anonymous Objects με αποτέλεσμα να μην μπορούμε να τα διαχειριστούμε.

6.2 DWS searchstudents

Η Data Web service με ονομασία searchstudents έχει αναφερθεί στο προηγούμενο κεφάλαιο. Στόχος της είναι να αναζητά στον πίνακα students της βάσης δεδομένων φοιτητές που στο όνομα ή επίθετο ή username στον aetos έχουν το string που δίνεται ως κλειδί αναζήτησης.

Ο πίνακας φέρει δεδομένα τα οποία ανακτήσαμε από την δημόσια ιστοσελίδα http://aetos.it.teithe.gr/pagelist/students.html κάνοντας parsing τον HTML κώδικα και μετατρέποντας τα δεδομένα σε DML insert statements.

```
CREATE TABLE ADMINISTRATOR.STUDENTS (
    USERNAME VARCHAR (30)  NOT NULL ,
    NAME VARCHAR (50) ,
    SURNAME VARCHAR (50),
    URL VARCHAR (40),
    EMAIL VARCHAR (40),
    CONSTRAINT PK_STUDENTS_USERNAME PRIMARY KEY ( USERNAME)  ) ;
```

Για να γεμίσουμε τα πεδία του δώσαμε insert statements που περιέχουν το username,name και surname. Τα πεδία url και email γεμίζουν αυτόματα με έναν trigger μιας και περιέχουν το username του κάθε φοιτητή.

```
1. --#SET TERMINATOR /
2.
3. CREATE TRIGGER ADMINISTRATOR.STUDENTI
4. AFTER  INSERT  ON ADMINISTRATOR.STUDENTS  REFERENCING  NEW AS nrow
5. FOR EACH ROW  MODE DB2SQL
6. begin atomic
7.     update administrator.students
8.     set email = username  || '@it.teithe.gr' ,
9.             url = 'http://www.it.teithe.gr/~' || username ;
10. end/
11.
12. --#SET TERMINATOR ;
```

Η μοναδική μέθοδος που υλοποιεί είναι η **SEARCHSTUDENTS** η οποία

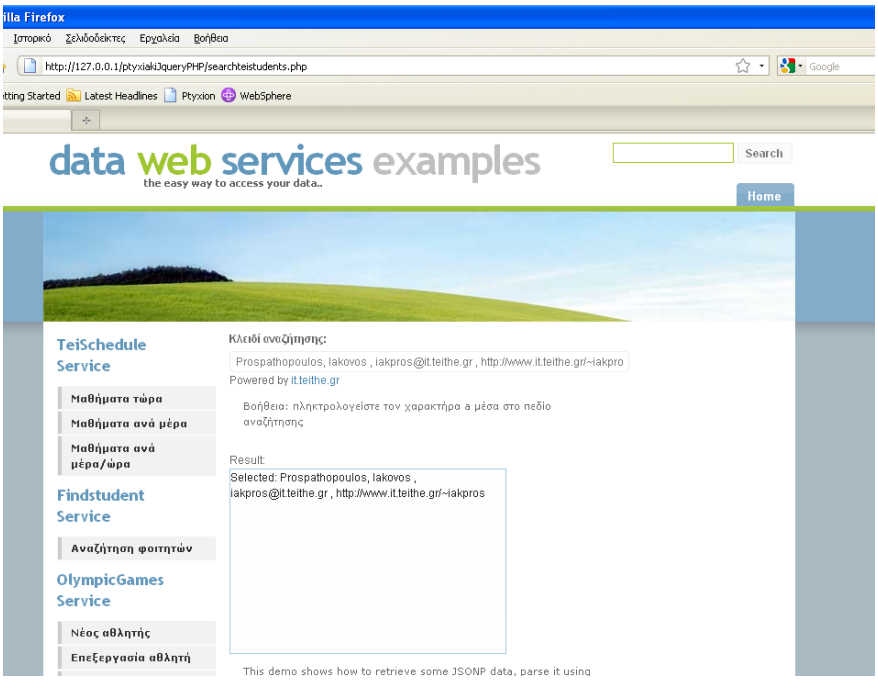
Επιστρέφει: τα πλήρη στοιχεία των φοιτητών που περιέχουν το string που δίνεται ως παράμετρος, μέσα στο όνομα ή στο επίθετο ή στο username του aetos.

Παράμετροι:

- **NAMEIN**: το string προς αναζήτηση

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1findstudents/rest/findstudents/SEARCHSTUDENTS?NAMEIN=pro

Η ιστοσελίδα που την καλεί (Σχήμα 6.1) μέσω REST HTTP GET JSONP είναι η http://127.0.0.1/ptyxiakiJqueryPHP/searchteistudents.php



Σχήμα 6.1 Η ιστοσελίδα που αναζητά φοιτητές

6.3 DWS teischedule

Η Data Web service με ονομασία teischedule είναι μια υπηρεσία που έχει ως σκοπό να μας ενημερώνει για το ποια μαθήματα λαμβάνουν χώρα στο τμήμα Πληροφορικής μια συγκεκριμένη ώρα σε μια συγκεκριμένη ημερομηνία.

Τα δεδομένα του ημερησίου προγράμματος μαθημάτων ελήφθησαν από την ιστοσελίδα http://hydra.it.teithe.gr/s στην καρτέλα Πρόγραμμα Μαθημάτων. Κάθε ιστοσελίδα που περιέχει το πρόγραμμα του κάθε εξαμήνου αποθηκεύτηκε σε λειτουργικό σύστημα Linux όπου μετονομάστηκε σε sem1.html έως sem7.html Κατόπιν τα δεδομένα του προγράμματος που είναι αποθηκευμένα μέσα σε HTML κώδικα πέρασαν από parser δική μας κατασκευής και εξήγαγαν τις DML insert εντολές για το γέμισμα του πίνακα pogramma. Οι parsers είναι δύο bash scripts με που χρησιμοποιούν την εντολή gawk.

Ο πίνακας που περιλαμβάνει το ημερήσιο πρόγραμμα είναι ο ακόλουθος:

```
CREATE TABLE  programma(
  aa INT NOT NULL GENERATED ALWAYS AS IDENTITY (START WITH 1, INCREMENT
BY 1) PRIMARY KEY,
  mday varchar(15) ,
  mdayn int ,
  starth time ,
  endh time ,
  course varchar(100) ,
  class varchar(10) ,
  lectype varchar(10),
  semester int
);
```

Οι μέθοδοι που υλοποιούνται στην data web service teischdule είναι τρεις:

1. Η μέθοδος **COURSESNOW** είναι μια stored procedure που επιστρέφει τα μαθήματα που γίνονται αυτή τη χρονική και ημερολογιακή στιγμή στο τμήμα Πληροφορικής.

Παράμετροι: δεν έχει

Παράδειγμα χρήσης (REST HTTP GET):
http://192.168.157.128:8080/Project1teischedule/rest/teischedule/COURSESNOW

2. Η μέθοδος **COURSESDAY** είναι μια stored procedure που επιστρέφει τα μαθήματα που γίνονται την παρούσα ώρα (δηλαδή τώρα είναι 11:30) σε μια δοσμένη ημερομηνία που περνιέται ως παράμετρος

Παράμετροι:

- **DATEIN**: η ημερομηνία σε μορφή ISO (έτος-μήνας-μέρα)

Παράδειγμα χρήσης (REST HTTP GET):
http://192.168.157.128:8080/Project1teischedule/rest/teischedule/COURSESDAY?DATEIN=2010-10-1

3.Η μέθοδος COURSESDAYTIME είναι μια stored procedure που επιστρέφει τα μαθήματα που γίνονται μια συγκεκριμένη ώρα σε μια δοσμένη ημερομηνία που περνιούνται ως παράμετροι.

Παράμετροι:

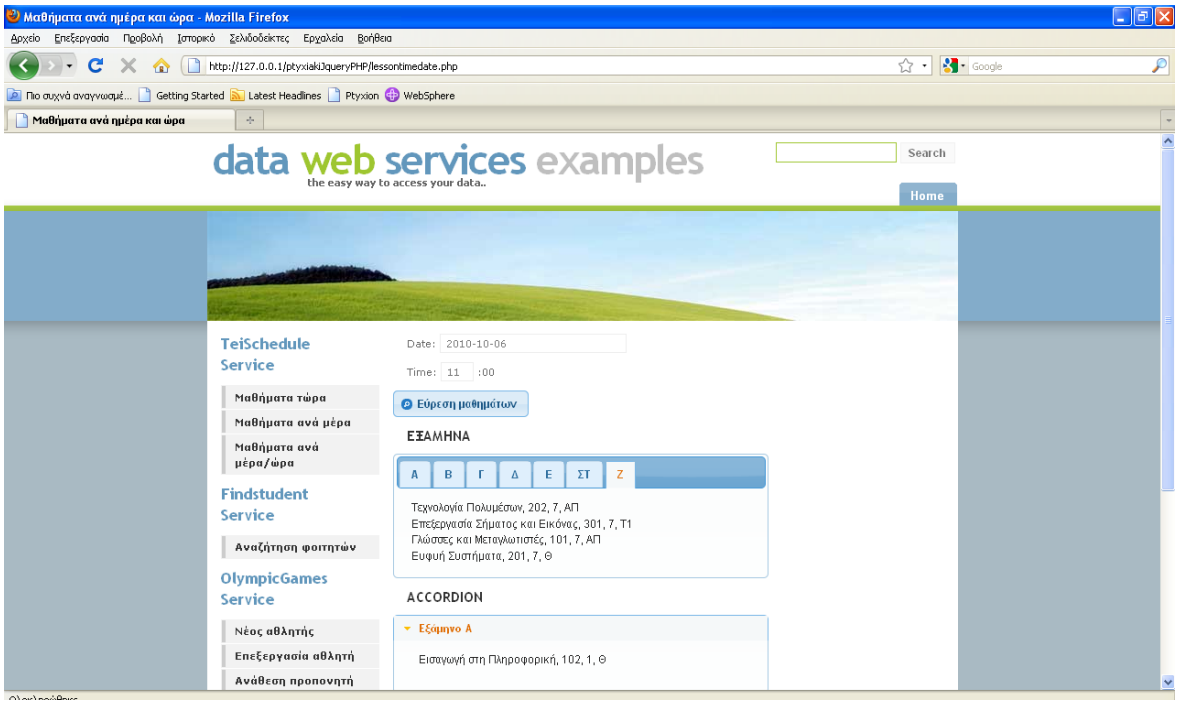
- **DATEIN**: η ημερομηνία σε μορφή ISO (έτος-μήνας-μέρα)
- **TIMEIN**: η ώρα έναρξης του μαθήματος (00 έως 23)

Παράδειγμα χρήσης (REST HTTP GET):
http://192.168.157.128:8080/Project1teischedule/rest/teischedule/COURSESDAYTIME?DATEIN=2010-10-1&TIMEIN=11

Οι ιστοσελίδες που υλοποιηθήκαν για να αναλώσουν τις 3 αυτές μεθόδους χρησιμοποιουν το REST HTTP GET JSONP binding που έχουμε κατασκευάσει εμείς από το REST HTTP GET JSON binding και είναι οι παρακάτω:

http://127.0.0.1/ptyxiakiJqueryPHP/lessonnow.php
http://127.0.0.1/ptyxiakiJqueryPHP/lessondate.php
http://127.0.0.1/ptyxiakiJqueryPHP/lessontimedate.php

Η lessontimedate.php φαίνεται στο παρακάτω Σχήμα 6.2

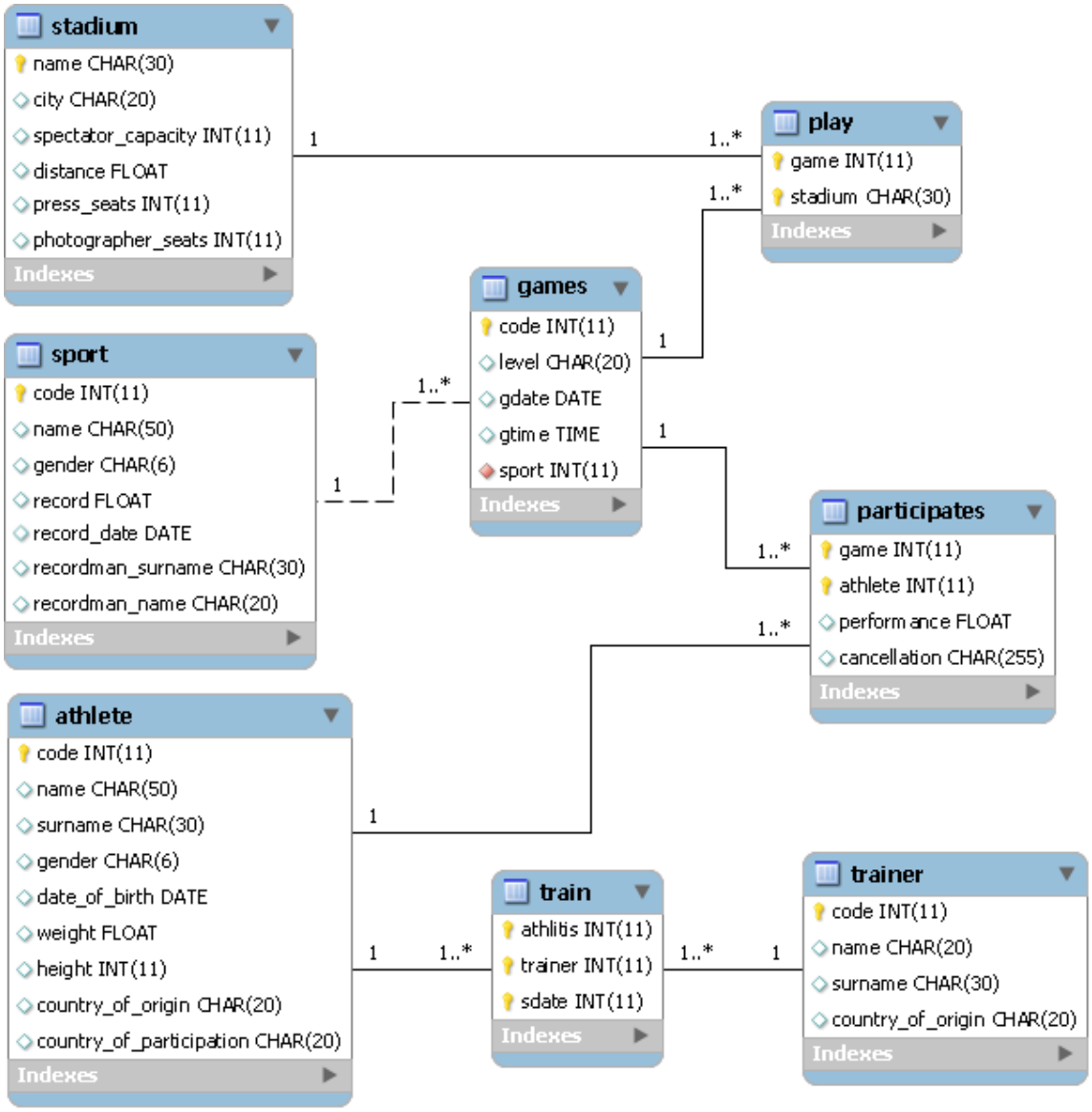


Σχήμα 6.2 Η ιστοσελίδα lessontimedate.php

6.4 DWS OlympicGames

Η Data Web service με ονομασία OlympicGames σε αντίθεση με τις προηγούμενες web services που ήταν read only, περιλαμβάνει μεθόδους που μεταβάλλουν τα δεδομένα της βάσης δεδομένων. Η ομώνυμη βάση δεδομένων την οποία προσπελαύνει και μεταβάλλει μας δίνεται στα μαθήματα Βάσεις Δεδομένων Ι και ΙΙ ως παράδειγμα για την επίλυση ασκήσεων. Εμείς χρησιμοποιούμε μόνο τρεις από τους 8 πίνακες, τον αθλητή (athlete), τον προπονητή (trainer) και την προπόνηση (train)

Το σχήμα της βάσης δεδομένων φαίνεται παρακάτω (Σχήμα 6.3) :



Σχήμα 6.3 Η βάση δεδομένων Olympic Games

Οι μέθοδοι που υλοποιεί η data web service είναι οι ακόλουθες:

1. Μέθοδος: fetchathletes

Πληροφορίες: η μέθοδος αυτή προέκυψε από ένα SQL script και μας εμφανίζει όλους τους αθλητές.
Επιστρέφει: όλα τα δεδομένα του πίνακα ATHLETE
Παράμετροι: δεν έχει

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1OlympicGames/rest/OlympicGames/fetchathletes

2. Μέθοδος: fetchtrainers
Πληροφορίες: η μέθοδος αυτή προέκυψε από ένα SQL script και μας εμφανίζει όλους τους προπονητές
Επιστρέφει: όλα τα δεδομένα του πίνακα TRAINERS (προπονητές)
Παράμετροι: δεν έχει

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1OlympicGames/rest/OlympicGames/fetchtrainers

3. Μέθοδος: INSERTATHLETE
Πληροφορίες: η μέθοδος αυτή είναι μια stored procedure που εισάγει έναν νέο αθλητή στον πίνακα athlete.
Επιστρέφει: τίποτα
Παράμετροι:

- NAMEM: όνομα αθλητή
- SURNAMEM: επώνυμο αθλητή
- GENDERM: φύλο τιμές: Male ή Female
- BIRTHM: η ημερομηνία γέννησης σε μορφή ISO (έτος-μήνας-μέρα)
- WEIGHTM: βάρος (float)
- HEIGHTM: ύψος σε cm(int)
- ORIGINM: το όνομα της χώρας προέλευσης
- PARTICIPATIONM: το όνομα της χώρας με την οποία συμμετάσχει

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1OlympicGames/rest/OlympicGames/INSERTATHLETE?NAMEM=Haros&SURNAMEM=Maros&GENDERM=Male&BIRTHM=2010-10-1&WEIGHTM=101&HEIGHTM=190&ORIGINM=Gana&PARTICIPATIONM=Gana
3. Μέθοδος: UPDATEATHLETE
Πληροφορίες: η μέθοδος αυτή είναι μια stored procedure που τροποποιεί τα στοιχεία του αθλητή με κωδικό CODEM στον πίνακα ATHLETE
Επιστρέφει: τίποτα
Παράμετροι:

- CODEM: κωδικός αθλητή (int)
- NAMEM: όνομα αθλητή
- SURNAMEM: επώνυμο αθλητή
- GENDERM: φύλο τιμές: Male ή Female
- BIRTHM: η ημερομηνία γέννησης σε μορφή ISO (έτος-μήνας-μέρα)
- WEIGHTM: βάρος (float)
- HEIGHTM: ύψος σε cm(int)
- ORIGINM: το όνομα της χώρας προέλευσης
- PARTICIPATIONM: το όνομα της χώρας με την οποία συμμετάσχει

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1OlympicGames/rest/OlympicGames/UPDATEATHLETE?CODEM=79&NAMEM=Peras&SURNAMEM=Maros&GENDERM=Male&BIRTHM=2010-10-1&WEIGHTM=101&HEIGHTM=190&ORIGINM=Gana&PARTICIPATIONM=Gana

4. Μέθοδος: DELETEATHLETE
Πληροφορίες: η μέθοδος αυτή είναι μια stored procedure που διαγράφει τον αθλητή με κωδικό AID από τον πίνακα ATHLETE και καταργεί τις προπονήσεις του στον πίνακα TRAIN
Επιστρέφει: τίποτα
Παράμετροι:

- AID: κωδικός αθλητή (int)

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1OlympicGames/rest/OlympicGames/DELETEATHLETE?AID=79

5. Μέθοδος: ASSIGNTRAINERTOATHLETE
Πληροφορίες: η μέθοδος αυτή είναι μια stored procedure που αναθέτει στον αθλητή με κωδικό AID να τον προπονεί ένας προπονητής με κωδικό TID για ένα συγκεκριμένο έτος. Η εγγραφή γίνεται στον πίνακα TRAIN.
Επιστρέφει: τίποτα
Παράμετροι:

- AID: κωδικός αθλητή (int)
- TID: κωδικός προπονητή (int)
- YE: έτος (0000 έως 9999)

Παράδειγμα χρήσης (REST HTTP GET):
http://localhost:8080/Project1OlympicGames/rest/OlympicGames/ASSIGNTRAINERTOATHLETE?AID=79&TID=2&YE=2010

Οι ιστοσελίδες που καλούν τις μεθόδους αυτής της Data Web service είναι οι παρακάτω:
http://127.0.0.1/ptyxiakiJqueryPHP/newathlete.php: η ιστοσελίδα αυτή μας παρέχει μια φόρμα στην οποία εισάγουμε τα στοιχεία του νέου αθλητή. Η υποβολή της φόρμας καλεί τη μέθοδο INSERTATHLETE με τη χρήση του REST HTTP POST URL encoded binding.

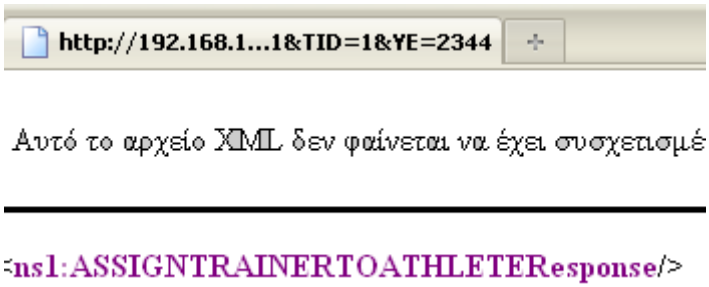
http://127.0.0.1/ptyxiakiJqueryPHP/editathlete.php: η ιστοσελίδα αυτή καλεί τη μέθοδο fetchathletes (μέσω REST HTTP GET) για να μας τους παρουσιάσει σε ένα drop down list. Επιλέγοντας έναν αθλητή βλέπουμε και μπορούμε να μεταβάλλουμε τα στοιχεία του μέσα σε μια φόρμα. Η υποβολή της φόρμας καλεί τη μέθοδο UPDATEATHLETE με τη χρήση του REST HTTP POST URL encoded binding

http://127.0.0.1/ptyxiakiJqueryPHP/deleteathlete.php: η ιστοσελίδα αυτή καλεί τη μέθοδο fetchathletes (μέσω REST HTTP GET) για να μας τους παρουσιάσει σε ένα combobox. Επιλέγοντας έναν αθλητή και πατώντας το πλήκτρο *Διαγραφή*

υποβάλλουμε μια φόρμα η οποία καλεί τη μέθοδο DELETEATHLETE με τη χρήση του REST HTTP POST URL encoded binding.

http://127.0.0.1/ptyxiakiJqueryPHP/assignathlete.php: η ιστοσελίδα αυτή καλεί τις μεθόδους fetchathletes και fetchtrainers (μέσω REST HTTP GET) για να μας τους παρουσιάσει σε δύο drop down lists. Επιλέγουμε τον αθλητή και τον προπονητή που θέλουμε να τον προπονήσει, το έτος προπόνησης και επιλέγουμε Ανάθεση Προπόνησης.

⚠ Αξίζει να σημειώσουμε ότι όταν κάνουμε submit τις φόρμες των 4 άνω σελίδων αυτές μας επιστρέφουν μια κενή απάντηση που μοιάζει με αυτήν στο Σχήμα 6.4. Αυτή είναι η κενή απάντηση που επιστρέφει η web service αν δεν έχει κάποιο SQL select statement που επιστρέφει κάτι πίσω. Φυσικά σε μια κανονική ιστοσελίδα θα το χειριστούμε με τέτοιο τρόπο ώστε να μην φαίνεται.



Σχήμα 6.4 Η κενή απάντηση της web service

6.5 Οδηγίες χρήσης VMware Image

Μέσα στο DVD περιέχεται ένας φάκελος με όνομα ptyxiakiF1 η οποία περιέχει ένα εικονικό μηχάνημα που φορτώνεται στο VMware Workstation ή Player (έκδοση 6.5 και άνω).

Το image αυτό περιέχει έναν εικονικό υπολογιστή που έχει εγκατεστημένα όλα τα προγράμματα και τους servers που αναφέραμε σε αυτήν την πτυχιακή.

Τεχνικά χαρακτηριστικά:

- OS: Windows XP Pro SP3
- .NET framework: version 3.5
- RAM memory: 1 Gb
- Max HDD: 10 Gb
- Network: NAT (static IP address 192.168.157.128)
- HDD persistence: Non persistent (οι αλλαγές διαγράφονται μετά το shut down)

Εγκατεστημένα προγράμματα και servers:

- IBM Data Studio IDE
- IBM DB2 Express C 9.7
- IBM WebSphere Application Server CE

- XAMPP (Apache, MySQL, PHP, Mercury Mail, Filezilla FTP)
- Mozilla Firefox

Ρυθμίσεις χρηστών:

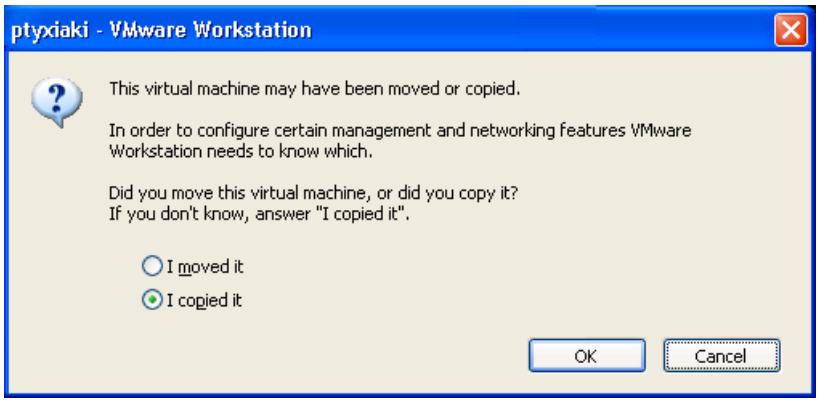
- Windows
  - User: Administrator, password: χωρίς κωδικό
  - User: db2admin, password: db2admin
- DB2:
  - User: Administrator, password: χωρίς κωδικό
  - User: db2admin, password: db2admin
- WebSphere:
  - Console: http://127.0.0.1:8080/console/portal/Welcome
  - User: system, password: system
- Apache:
  - Admin page: http://127.0.0.1/xampp
  - User: root, password: χωρίς κωδικό

Εκκίνηση του μηχανήματος

Όταν εκκινήσουμε το μηχάνημα για πρώτη φορά, υπάρχει η πιθανότητα να μας εμφανίσει ένα μήνυμα σαν αυτό που φαίνεται στο Σχήμα 6.5. Εμείς επιλέγουμε το I copied it και κάνουμε κλικ στο OK.

Για να δούμε τις ιστοσελίδες των παραδειγμάτων πρέπει πρώτα να εκκινήσουμε τον WebSphere και τον Apache server. Στην επιφάνεια εργασίας υπάρχουν δύο εικονίδια Start Apache και Start WebSphere. Κάνουμε διπλό κλικ και στα δύο. Εμφανίζονται για το καθένα από ένα παράθυρο command line όπου φορτώνουν διάφορα modules. Όταν ολοκληρωθεί το παράθυρο του Apache το κλείνουμε. Το παράθυρο στο οποίο φορτώνει ο WebSphere δεν πρέπει να το κλείσουμε γιατί θα κλείσει και ο WebSphere μαζί.

Κατόπιν ανοίγουμε τον Mozilla Firefox και επιλέγουμε από την μπάρα σελιδοδεικτών τον σελιδοδείκτη Ptyxion ο οποίος φορτώνει τη σελίδα http://127.0.0.1/ptyxiakiJqueryPHP/index.php Η ιστοσελίδα περιέχει διάφορα πλευρικά μενού με τα οποία μπορείτε να πλοηγηθείτε στις σελίδες που έχουν προαναφερθεί.



Σχήμα 6.5 Μήνυμα μεταφοράς του VMware Image

6.6 Επίλογος

Στο κεφάλαιο αυτό έγινε μια σύντομη περιγραφή για τις data web services που υλοποιήθηκαν στα πλαίσια της πτυχιακής αυτής. Κατόπιν γίνεται αναφορά στις ιστοσελίδες που τις χρησιμοποιούν. Τέλος δίνονται οδηγίες χρήσης του εικονικού μηχανήματος VMware που περιέχει όλα αυτά τα παραδείγματα.

7. Συμπεράσματα – Προτάσεις

Η χρήση του IBM Data Studio είναι αρκετά απλή και μας διευκολύνει στη δημιουργία των Data Web Services. Το γεγονός ότι δεν γράφουμε καθόλου κώδικα και δεν μπορούμε να τον δούμε μας περιορίζει σε συγκεκριμένους τύπους Data Web Service. Πιστεύω ότι θα ήταν χρήσιμο ο προγραμματιστής να έχει τη δυνατότητα να βλέπει τον κώδικα ώστε να μπορεί να προσθέσει επιπλέον λειτουργικότητα στην Data Web Service.

Οι υπηρεσίες που δημιουργούνται με το Data Studio δεν μπορούν να υποστηρίξουν ασφάλεια (authentication via Kerberos,RADIUS etc.). Επίσης δεν απαντούν σε μορφή JSONP γεγονός που αποτρέπει έναν προγραμματιστή να τις καλέσει μέσω AJAX. Εγώ αναγκάστηκα να καλέσω το JSON binding και μέσω PHP να δημιουργήσω ένα δικό μου JSONP binding για να καταστήσω δυνατή την επικοινωνία μέσω AJAX και HTTPXMLRequest. Αυτά τα στοιχεία θα μπορούσαν να προστεθούν σε κάποια επόμενη έκδοση του IDE.

Γενικά η χρήση των Data Web Services μας βοηθά στο γεγονός ότι δεν χρειάζεται να ασχολούμαστε με θέματα βάσης δεδομένων όταν δημιουργούμε μια ιστοσελίδα που χρησιμοποιεί DWS. Κάθε συναλλαγή με τη βάση γίνεται με κλήση μεθόδων μιας web service ελαφρύνοντας έτσι τον προγραμματιστή από το φόρτο της διαδικασίας open connection/execute statements/get results/close connection η οποία περικλείει πολλές γραμμές κώδικα.

Οι Data Web Services θα μπορούσαν να χρησιμοποιηθούν για το TEI σε δεδομένα που δεν εμπεριέχουν πολιτική απορρήτου όπως μέσοι όροι μαθημάτων, αποτελέσματα αξιολόγησης, πρόγραμμα μαθημάτων. Δεδομένου ότι δεν υποστηρίζονται ακόμη πολιτικές ασφαλείας από το Data Studio οι DWS θα ήταν απαγορευτικές για δεδομένα όπως προσωπικά στοιχεία φοιτητών, βαθμολογίες και απουσίες.

Τέλος τα προϊόντα της IBM κατά τη γνώμη μου απαιτούν ως διαχειριστή κάποιον ο οποίος είναι καλός γνώστης της γλώσσας προγραμματισμού Java και των υλοποιήσεων που έχει δημιουργήσει η IBM για τα προϊόντα της οι οποίες μπορεί να διαφοροποιούνται από αυτές της Oracle Sun.

7.Βιβλιογραφία

1. Weerawarana,Curbera,Leymann,Stor, 2008. Αρχιτεκτονική Πλατφόρμας Υπηρεσιών Ιστού.Κλειδάριθμος.

2. Papazoglou Michael, 2007. Web Services: Principles and Technology. Pearson, Prentice Hall.

3. Debra Eaton,Vitor Rodrigues, Manoj K. Sardana, Michael Schenker, Kathryn Zeidenstein,Raul F. Chong. (Δεκέμβριος 2009). Getting Started with IBM Data Studio for DB2. In IBM DeveloperWorks. Retrieved 2 Οκτωβρίου 2010, from http://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+IBM+Data+Studio+for+DB2.

4. Whei-Jen Chen, Art Sammartino, Dobromir Goutev, Felicity Hendricks, Ippei Komi, Ming-Pang Wei, et al. (23 Ιανουάριος 2007). *DB2 9 pureXML Guide*. IBM.

5. Κεραμόπουλος Ευκλείδης (2009) , Σημειώσεις μαθήματος Βάσεις Δεδομένων ΙΙ, ΑΤΕΙ Θεσσαλονίκης

6. Douglas Crockford. (undefined). Introducing JSON. In JSON. Retrieved 2 Οκτωβρίου 2010, from http://www.json.org/.

7. Seda Özses, Salih Ergül. (24 Φεβρουαρίου 2009). Cross-domain communications with JSONP, Part 1: Combine JSONP and jQuery to quickly build powerful mashups. In IBM DeveloperWorks. Retrieved 2 Οκτωβρίου 2010, from http://www.ibm.com/developerworks/library/wa-aj-jsonp1/.

8. Seda Özses, Salih Ergül. (3 Μαρτίου 2009). Cross-domain communications with JSONP, Part 2: Building mashups with JSONP, jQuery, and Yahoo! Query Language. In IBM DeveloperWorks. Retrieved 2 Οκτωβρίου 2010, from http://www.ibm.com/developerworks/library/wa-aj-jsonp2/.

9. Douglas Crockford. (Ιούλιος 2006). The application/json Media Type for JavaScript Object Notation (JSON). In Network Working Group . Retrieved 2 Οκτωβρίου 2010, from http://www.ietf.org/rfc/rfc4627.txt?number=4627.

10. Bob Ippolito. (5 Δεκεμβρίου 2005). Remote JSON - JSONP. In from __future__ import * . Retrieved 2 Οκτωβρίου 2010, from http://bob.pythonmac.org/archives/2005/12/05/remote-json-jsonp/.

11. Άγνωστος συγγραφέας. (28 Σεπτεμβρίου 2010). JSON. In Wikipedia. Retrieved 2 Οκτωβρίου 2010, from http://en.wikipedia.org/wiki/JSON.

12. W3C Working Group. (27 Ιουλίου 2010). Cross-Origin Resource Sharing. In W3C. Retrieved 2 Οκτωβρίου 2010, from http://www.w3.org/TR/cors/.

13. W3C. (nd). WSDL Tutorial. In W3schools. Retrieved 2 Οκτωβρίου 2010, from http://www.w3schools.com/wsdl/default.asp.

14. W3C. (nd). SOAP intriduction. In W3schools. Retrieved 2 Οκτωβρίου 2010, from http://www.w3schools.com/soap/soap_intro.asp

15. David Booth, Hugo Haas, Francis McCabe, Eric Newcomer Michael Champion ,Chris Ferris ,David Orchard. (11 Φεβρουαρίου 2004). Web Services Architecture. In W3C. Retrieved 2 Οκτωβρίου 2010, from http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/wsa.pdf.

16. Mohamed I. Mabrouk . (5 Σεπτεμβρίου 2008). SOA fundamentals in a nutshell. In IBM DeveloperWorks. Retrieved 2 Οκτωβρίου 2010, from http://www.ibm.com/developerworks/webservices/tutorials/ws-soa-ibmcertified/section8.html.

17. Tim O'Reilly. (3 Απριλίου 2003). REST vs. SOAP at Amazon. In O'Reilly XML.com. Retrieved 2 Οκτωβρίου 2010, from http://www.oreillynet.com/pub/wlg/3005.

18. Άγνωστος συγγραφέας. (). Creating a REST Request. In Yahoo Developer Network. Retrieved 2 Οκτωβρίου 2010, from http://developer.yahoo.com/search/rest.html.

19. Vijay Bommireddipalli . (6 Δεκεμβρίου 2007). Data Web Services: Build Web services the new way to access IBM database servers. In IBM DeveloperWorks. Retrieved 2 Οκτωβρίου 2010, from http://www.ibm.com/developerworks/data/library/techarticle/dm-0712bommireddipalli/.

20. Άγνωστος συγγραφέας. (). WCF Data Services and OData At-a-Glance . In Data Developer Center. Retrieved 2 Οκτωβρίου 2010, from http://msdn.microsoft.com/en-us/data/aa937697.aspx.

21. Άγνωστος συγγραφέας. (12 Σεπτεμβρίου 2006). Microsoft Open Specification Promise. In Microsoft Interoperability. Retrieved 2 Οκτωβρίου 2010, from http://www.microsoft.com/interop/osp/default.mspx.

22. Chris Sells. (Μάρτιος 2010). Open Data Protocol by Example. In Microsoft Developer Network. Retrieved 2 Οκτωβρίου 2010, from http://msdn.microsoft.com/en-us/library/ff478141.aspx.

23. John Resig. (). Tutorials:How jQuery Works. In jQuery Documentation. Retrieved 2 Οκτωβρίου 2010, from http://docs.jquery.com/How_jQuery_Works.

24. Bruce Schneier . (15 Ιουνίου 2000). Security Risks of Unicode. In Crypto-Gram Newsletter. . Retrieved 2 Οκτωβρίου 2010, from http://www.schneier.com/crypto-gram-0007.html#9.