ΤΜΗΜΑ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΠΟΛΥΜΕΣΩΝ

ΤΕΙ ΚΡΗΤΗ

Διαχείριση session σε Web Services

Εισηγητής: Μάρης Ιωάννης

Επιμελητής: Αθανάσιος Μαλάμος

Περιεχόμενα

Web Services

Εισαγωγική Περίληψη

Η επιχειρήσεις απομακρύνονται όλο και περισσότερο από τις βασισμένες σε συναλλαγές επιχειρησιακές διαδικασίες σε μια προσανατολισμένη στις υπηρεσίες προσέγγιση, περιλαμβάνοντας συχνά τη συνεργασία των πολλαπλών οργανώσεων για να παρέχει μια ολοκληρωμένη υπηρεσία στους πελάτες. Επιπλέον, οι υπάρχουσες οικονομικές μέθοδοι αυξάνουν την πίεση ώστε να παγιωθούν συστήματα πληροφοριών και να κεφαλαιοποιηθούν οι υπάρχουσες επενδύσεις. Συνεπώς, είναι απαραίτητος ένας αποτελεσματικός τρόπος ώστε να επαναχρησιμοποιηθούν τα ήδη υπάρχοντα συστατικά και να ολοκληρωθούν καινούργιες υπηρεσίες, μέσα στις ενδοεπιχειρησιακές εφαρμογές και υπηρεσίες αλλά και στις εφαρμογές και υπηρεσίες που αφορούν πελάτη-επιχείρηση αλλά και εσωτερικά μέσα στις ίδιες τις επιχειρήσεις.
Οι υπάρχουσες μέθοδοι ολοκλήρωσης και επαναχρησιμοποίησης εφαρμογών απαιτούν δαπανηρές, χρονοβόρες, επί παραγγελία αναπτύξεις νέων εφαρμογών, απαιτώντας συνήθως όλα τα ενδιαφερόμενα μέρη να εφαρμόσουν τις συμβατές, πρότυπες υποδομές ανάπτυξης. Η πρόκληση για τη βιομηχανία IT (Information Technology) ήταν να αναπτύξει ένα ελαφρύ πλαίσιο που θα μπορούσε να χρησιμοποιηθεί για να ικανοποιήσει την ικανότητα επαναχρησιμοποίησης και τις προσαρμόσιμες απαιτήσεις κατά τρόπο οικονομικώς αποδοτικό.
Το αναδυόμενο πλαίσιο των υπηρεσιών διαδικτύου (Web Services Technologies - WST) ενδεχομένως προσφέρει έναν τρόπο να επιτευχθεί αυτό. Το WST, που είναι ουσιαστικά μια τεχνολογία ενεργοποίησης παρά μια πλήρως νέα μέθοδος, είναι μια κατευθυνόμενη από τη βιομηχανία πρωτοβουλία με εμπλεκόμενους, πολλούς από τους κύριους φορείς όπως η

Microsoft, η Sun, η HP και η IBM. Αυτό φέρνει και το πλεονέκτημα της γρήγορης εξέλιξης και το μειονέκτημα της πιθανής εμπορικής σύγκρουσης.
Στην ουσία, το WST είναι ένα πλαίσιο των προσαρμοζόμενων εφαρμογών, οι οποίες μπορούν να ανακαλυφθούν και να εκτελεσθούν σε ένα δίκτυο υπολογιστών από απομακρυσμένα προγράμματα. Αυτές οι διανεμημένες εφαρμογές δημιουργούνται χρησιμοποιώντας απλά πρωτόκολλα, τα οποία επιτρέπουν στις οργανώσεις να χτίσουν τις εφαρμογές επικοινωνίας χωρίς την απαίτηση ότι και οι δύο άκρες τρέχουν τα ίδια περιβάλλοντα.
Η αρχιτεκτονική τωνWeb Services (WSs) παρέχει δημόσιες, εμφανείς και αλληλεπιδράσιμες δυνατότητες. Οι Web Services είναι εφαρμογές που στέλνουν και λαμβάνουν τα μηνύματα σε ένα δίκτυο τα οποία είναι τυποποιημένου σχήματος. Τα μηνύματα αποκωδικοποιούνται και περνούν στις εσωτερικές εφαρμογές του συστήματος για εκτέλεση. Οι περιγραφές των WSs, συμπεριλαμβανομένης της μεθόδου σύνδεσης με αυτές, δημοσιεύονται στις καταχωρίσεις της WS, οι οποίες μπορούν μετά να ανακτηθούν με τη χρήση ερωτημάτων. Αυτό επιτρέπει στους προγραμματιστές την ανεξάρτητη ανάπτυξη των εφαρμογών πελατών για να αναζητήσουν και να αλληλεπιδράσουν με μια WS μέσω του Διαδικτύου, ελαττώνοντας έτσι το κόστος ανάπτυξης της εφαρμογής.
Η ανάπτυξη των προτύπων WST έχει φθάσει σε μια σημαντική κρίσιμη καμπή. Ενώ τα βασικά πρότυπα SOAP (Simple Object Access Protocol), UDDI(Universal Data Description and Integration) και WSDL (Web Services Definition Language) έχουν συγχωνευτεί σε χρησιμοποιήσιμες μορφές, η λειτουργία τους είναι περιορισμένη. Η προσπάθεια της βιομηχανίας επικεντρώνεται τώρα στην εξέταση της ροής της επιχειρησιακή λογικής, της ασφάλειας και του συντονισμού των υπηρεσιών, οι οποίες θα επιτρέψουν την καλύτερη λειτουργικότητα, επεκτείνοντας έτσι τους τομείς για εφαρμογή των συγκεκριμένων ιδεών. Πρόσθετες απεικονίσεις των δεδομένων και των διαγραμμάτων ροής θα απαιτηθούν για να ικανοποιήσουν την απαίτηση των επιχειρήσεων για διαφορετικά περιβάλλοντα εφαρμογών.

Αν και το πλαίσιο εξελίσσεται ακόμα, οι σημαντικότεροι φορείς IT έχουν ήδη ενεργοποιημένα τα περιβάλλοντα εφαρμογών WS. Ομοίως, οι σημαντικότερες επιχειρήσεις που συμμετέχουν στην επιχειρηματική ολοκλήρωση εφαρμογών και την παροχή βάσεων δεδομένων έχουν ενσωματώσει το WST στο σύνολο των προϊόντων τους, με τους υπόλοιπους να αναμένεται να το εφαρμόσουν στο εγγύς μέλλον. Οι εμπορικές υπηρεσίες που βασίζονται στη WST αρχίζουν να εμφανίζονται.
Η WST έχει διάφορους πιθανούς τομείς εφαρμογής συγκεκριμένου ενδιαφέροντος. Μέσα στα ιδρύματα, η WST θα προσφέρει έναν οικονομικό και αποτελεσματικό τρόπο ανάπτυξης νέων υπηρεσιών, επιτρέποντας την ενσωμάτωσή τους σε υπάρχουσες εφαρμογές.
Συνοψίζοντας, αν και ακόμα εξελίξιμη, η WST προσφέρει αξιόλογη προοπτική. Οι WSs έχουν αναπτυχθεί αρκετά και προσφέρουν μία εύκολη και οικονομική λύση στην ανάπτυξη δικτυακών εφαρμογών.

Η τεχνολογία

Οι WSs περιγράφονται καλύτερα ως ένα πλαίσιο εφαρμογών που μπορούν να αναζητηθούν και να εκτελεσθούν σε ένα δίκτυο υπολογιστών από τα απομακρυσμένα προγράμματα.
Η ένωση διαδικτύου W3C (World Wide Web Consortium) αναπτύσσει μια αρχιτεκτονική Web Services για να βοηθήσει την ευκολότερη κατανόηση και εκμάθηση τους η οποία βασίζεται στις λειτουργίες ενσωμάτωσηςς (publish), αναζήτησης (discover) και αλληλεπίδρασης (interact).

Ενσωμάτωση (Publish)

Εάν μια οργάνωση, που αναπτύσσει μια WS, επιθυμεί να την καταστήσει διαθέσιμη προς χρήση από άλλες υπηρεσίες είτε μέσω του Διαδικτύου είτε μέσα στο ενδοδίκτυό της, πρέπει

πρώτα να καταστήσει διαθέσιμες τις λεπτομέρειες στους πιθανούς χρήστες-πελάτες. Η αρχιτεκτονική της WS παρέχει ένα χαρακτηριστικό γνώρισμα “publish” για να επιτευχθεί αυτό.
Ο παροχέας της WS "δημοσιεύει" πληροφορίες για τον εαυτό του, τις υπηρεσίες που προσφέρει και πώς είναι δυνατό να συνδεθεί κάποιος με αυτές, αποστέλλοντας πληροφορίες στις καταχωρήσεις της WS, όπου αυτές οι πληροφορίες στη συνέχεια αποθηκεύονται. Οι καταχωρήσεις της WS, συνήθως βασισμένες στο πρότυπο UDDI (Universal Data Description and Integration) το οποίο παρέχει τις κίτρινες (Περιέχει διάφορες κατηγοριοποιήσεις βασισμένες σε νόμους και διατάξεις), πράσινες (Περιέχουν τεχνικές πληροφορίες σχετικά με τις υπηρεσίες που παρέχονται) και άσπρες (Περιέχουν διεύθυνση, και γνωστά προσδιοριστικά για την WS) σελίδες για την WS, οι οποίες επιτρέπουν την αναζήτηση και απαρίθμηση των υπηρεσιών και παρέχουν λεπτομέρειες σχετικά με την απομακρυσμένη επαφή για περισσότερες πληροφορίες. Ο προμηθευτής της WS μπορεί να ενεργήσει ως καταχωρίσεις της WS.
Η περιγραφή της WS, που περιλαμβάνει τα μηνύματα που η υπηρεσία αναμένει και επιστρέφει και πώς να τα κωδικοποιήσει και πού να τα στείλει, γράφεται κανονικά στη γλώσσα περιγραφής υπηρεσιών διαδικτύου WSDL (Web Service Description Language), ένα XML έγγραφο για την περιγραφή της WS. Επιπρόσθετες πληροφορίες για τη WS όπως η ποιότητα των παρεχόμενων υπηρεσιών ή περαιτέρω πληροφορίες για την επιχειρήση μπορεί επίσης να δημοσιευθεί στις καταχωρίσεις της WS χρησιμοποιώντας τις δομές δεδομένων UDDI.

Αναζήτηση (Discover)

Προτού να μπορέσει να χρησιμοποιήσει η εφαρμογή χρήστη-πελάτη μια απομακρυσμένη WS, πελάτης δεν πρέπει μόνο να επιλέξει μια κατάλληλη WS από όσες είναι διαθέσιμες, αλλά πρέπει επίσης να μάθει πώς να συνδεθεί με αυτή. Αυτό επιτυγχάνεται χρησιμοποιώντας το χαρακτηριστικό γνώρισμα “discover” της WS.
Για να ανακαλύψει μια υπηρεσία, ένας πελάτης στέλνει ένα αίτημα "διαθεσιμότητας των υπηρεσιών" στο δίκτυο. Οι καταχωρήσεις της WS (WS Registry) απαντά με τις λεπτομέρειες των WSs και πώς μπορούν να κληθούν, με ένα τυποποιημένο σχήμα (συνήθως WSDL). Τα αιτήματα και οι απαντήσεις στέλνονται κανονικά χρησιμοποιώντας το πρωτόκολλο SOAP(Simple Object Access Protocol).
Ένας πελάτης μπορεί επίσης να καταχωρηθεί στη WS Registry έτσι ώστε να ενημερώνεται για τις αλλαγές σε οποιαδήποτε από τις WSs για τις οποίες ενδιαφέρεται.
Τότε μπορεί να γραφτεί κατάλληλο λογισμικό χρήστη-πελάτη για να ενσωματώσει τις κλήσεις στην επιλεγμένη απομακρυσμένη WS.

Αλληλεπίδραση (Interact)

Μόλις κωδικοποιηθεί η κλήση σε μια επιλεγμένη απομακρυσμένη WS, ο χρήστης-πελάτης μπορεί απομακρυσμένα να καλέσει τη WS μέσω δικτύου. Ο παροχέας της WS εκτελεί την καλούμενη WS χρησιμοποιώντας δεδομένα εισόδου που διευκρινίζονται από τον πελάτη. Η εκτέλεση της WS μπορεί να είναι μια απλή ασύγχρονη λειτουργία ή μπορεί να χρειασθεί ανταλλαγή πληροφοριών μεταξύ του πελάτη και της απομακρυσμένης WS κατά τη διάρκεια της εκτέλεσης. Η παραγωγή αιτημάτων σε άλλες WSs μπορεί επίσης να εμφανιστεί.
Η επικοινωνία μεταξύ του πελάτη και της απομακρυσμένης WS πραγματοποιείται με την ανταλλαγή μηνυμάτων SOAP.
Η χρήση της WS Registry και τυποποιημένων μεθόδων επικοινωνίας, περιγραφής και ενσωμάτωσηςς στο δίκτυο σημαίνει ότι οι εφαρμογές μπορούν να γραφτούν ανεξάρτητα για να αναζητήσουν, να βρουν και να εκτελέσουν τις κατάλληλες WSs. Αυτό αφαιρεί την ανάγκη για ακριβή, χρονοβόρα, παραγόμενη επί παραγγελία ανάπτυξη εφαρμογών και από τον

πελάτη και από τους προμηθευτές της WS και επιτρέπει στις υπηρεσίες να επαναχρησιμοποιηθούν σε διαφορετικές καταστάσεις, ενδεχομένως εκτός πλάνου σχεδίου.

Σενάρια Εφαρμογής

Για να δώσει έμφαση στις πιθανές χρήσεις της WST, η W3C έχει αναπτύξει δύο σενάρια χρήσης. Δεδομένου ότι αυτά δεν έχουν ιδιαίτερη σχέση στην ευρωπαϊκή και παγκόσμια κοινότητα, δύο παραδείγματα της πιθανής χρήσης εντός αυτής της κοινότητας παρέχονται παρακάτω.

Εγγραφή σπουδαστών

Στο μέλλον, μια εφαρμογή υποστήριξης για τους νέους σπουδαστές θα μπορεί αυτόματα να εγγράφει τους σπουδαστές στα εκπαιδευτικά ιδρύματα, να τακτοποιήσει τη στέγαση και την ασφάλεια, να οργανώσει τη χρηματοδότηση, να καταβάλει τα δίδακτρα, να προγραμματίσει τα ταξίδια, και να προτείνει μια περιήγηση στην περιοχή του εκπαιδευτικού ιδρύματος για την εισαγωγική εβδομάδα. Αυτό το σενάριο θα περιλάβει τις αιτήσεις ενσωμάτωσης από το αρχείο καταχωρήσεων των εκπαιδευτικών ιδρυμάτων, τα γραφεία ενοικίασης, τις τράπεζες, τις ασφαλιστικές εταιρείες, τις αρχές επιχορηγήσεων, τα δάνεια σπουδαστών και τους τοπικούς κεντρικούς υπολογιστές πληροφοριών με κάποια λογική μέθοδο. Αν και αυτήν την περίοδο, η περιορισμένη αυτοματοποιημένη αλληλεπίδραση των εφαρμογών μπορεί να υπάρξει όπου οι συμμαχίες τον μεγάλων εταιριών έχουν διαμορφωθεί και οι όπου αναπτύσσονται μεμονωμένα εφαρμογές, αυτό δεν επιτρέπει σε έναν σπουδαστή να εφοδιαστεί με την πιο κατάλληλη επιλογή για κάθε μία από τις παραπάνω διαδικασίες. Επιπλέον, η εφαρμογή είναι δαπανηρή και πολύ χρονοβόρα.

Διαχείριση της δια βίου μάθησης

Δεδομένου ότι η αληθινή δια βίου μάθηση εξελίσσεται, η εκμάθηση αυτή μπορεί να παρασχεθεί από τα σχολεία, τα εκπαιδευτικά ιδρύματα, από επαγγελματικούς οργανισμούς, ιδιωτικές επιχειρήσεις εκμάθησης, κοινοτικά προγράμματα ή εκπαιδευτικά τμήματα μέσα στις επιχειρήσεις. Οι μαθητευόμενοι όχι μόνο θα επιθυμήσουν να ταιριάξουν τη διαθεσιμότητα των σειρών μαθημάτων με τις απαιτήσεις τους, αλλά θα καταγράφουν τα επιτεύγματά τους, ίσως χτίζοντας ένα online χαρτοφυλάκιο με τις εμπειρίες τους από αυτά τα μαθήματα. Για να το επιτύχουν αυτό, πολλές διαφορετικές οργανώσεις, με διάφορες μεθόδους καταγραφής των λεπτομερειών εκμάθησης των μαθητευόμενων θα πρέπει να συνεργαστούν μεταξύ τους. Στην πληρότητα του χρόνου, οι έξυπνες υπηρεσίες θα μπορούν να αναζητήσουν τις πιο κατάλληλες σειρές μαθημάτων βασισμένες στο προφίλ, τις απαιτήσεις και τις πληροφορίες του μαθητευόμενου.

Σύνδεση με άλλες σχετικές τεχνολογίες

Όσον αφορά την επόμενη γενιά του παγκόσμιου διαδικτύου, η WST στηρίζεται στην τεχνολογία XML.Η εξέλιξη της WST αφορά πλήρως τη μελλοντική ανάπτυξη του παγκόσμιου ιστού, συχνά καλούμενη ως σημαντικός ιστός (semantic web), ο οποίος θα βασιστεί σε τεχνολογίες όπως XML, RDF, DAML+OIL και Intelligent Agents. Η ανάπτυξη της WST επίσης θα συσχετιστεί βαθειά με την ανάπτυξη των μοντέλων και των προτύπων ηλεκτρονικού εμπορίου όπως το ηλεκτρονικό εμπόριο XML (ebXML).

Οι τεχνολογίες υπηρεσιών Ιστού (WST) προσφέρουν τη δυνατότητα στο να εξομαλύνουν τη διαλειτουργικότητα μεταξύ των παροχέων υπηρεσιών και των εφαρμογών, και να εξετάσουν

τα εσωτερικά ζητήματα ολοκλήρωσης των συστημάτων πληροφοριών (IS) μέσα στις οργανώσεις.
Η αρχική εισαγωγή του WST έχει περιοριστεί κυρίως στον εμπορικό τομέα. Η ανάγκη να προσφερθούν οι εσωτερικές υπηρεσίες που είναι ενσωματωμένες σε ένα σύστημα και να συνδεθούν με μια ευρεία ποικιλία εξωτερικών υπηρεσιών μέσω διάφανων διεπαφών, παράγει το ενδιαφέρον για τη WST. Αυτό είναι προφανές στο συνυπολογισμό της WST σε ένα πλήθος χρηματοδοτούμενων προγραμμάτων της κοινότητας ενωμένων πληροφοριακών συστημάτων JISC (Join Information Systems Committee).
Πριν αρχίσει οποιαδήποτε εφαρμογή, είναι σημαντικό για τα ιδρύματα να ερευνήσουν την αγορά μάρκετινγκ, ώστε να αξιολογήσουν την καταλληλότητα της WST μέσα στον τομέα ενδιαφέροντος και να αναλύσουν τα θέματα εφαρμογής της. Ειδικότερα, η ωριμότητα της τεχνολογίας, οι αναμενόμενες υποδείξεις ως προς το χρόνο ανάπτυξης, τα τεχνικά ζητήματα και οι έκβαση των προγραμμάτων JISC, πρέπει να αξιολογηθούν.

Προϊόντα

Κύρια πρωτόκολλα της τεχνολογίας υπηρεσιών ιστού (WST)

Η ένωση παγκόσμιου ιστού (W3C) δεν ορίζει συνθήκες οι οποίες θα χρησιμοποιούνται στην αρχιτεκτονική μιας WS, παρά μόνο τη λειτουργικότητα που απαιτείται. Τα πρότυπα που θα αναλυθούν παρακάτω διαμορφώνουν τα εξ' ορισμού βασικά στοιχεία στην διαμόρφωση των WSs.

XML

Η XML (Extensive Markup Language) είναι ένα απλό, πολύ προσαρμοστικό σχηματισμός κειμένων. Αρχικά σχεδιασμένος για να αντιμετωπίσει τις προκλήσεις των μεγάλης κλίμακας ηλεκτρονικών εκδόσεων, η XML διαδραματίζει επίσης έναν όλο και περισσότερο σημαντικό ρόλο στην ανταλλαγή μιας ευρείας ποικιλίας στοιχείων όσον αφορά τον παγκόσμιο Ιστό και αλλού.
Για τον λόγο αυτό η XML αποτελεί ένα από τα κύρια στοιχεία των WSs και χρησιμοποιείται τόσο στην διαμόρφωση των μηνυμάτων αποστολής και απάντησης των πελατών και των εξυπηρετητών, όσο και στην αποθήκευση δεδομένων και ρυθμίσεων των τεχνολογιών που διαχειρίζονται και φιλοξενούν τις WSs.
Ένα XML κείμενο έχει μια συγκεκριμένη μορφή και πρέπει να τηρεί αυστηρά αυτή την μορφή για να είναι έγκυρο. Οποιοδήποτε λάθος κατά την διαμόρφωση του XML θα προκαλέσει λάθη κατά την χρήση του ειδικά όταν χρησιμοποιείται σε WSs.
Ωστόσο, κάθε τεχνολογία που χρησιμοποιεί το XML επιβάλει τα δικά της πρότυπα διαμόρφωσής του, τα οποία όμως δεν απέχουν και πολύ από τα αρχικά πρότυπα του.
Ένα απλό παράδειγμα της γλώσσας XML για την περιγραφή ενός βιβλίου είναι το ακόλουθο.

```
<? xml version="1.0" encoding="UTF-8"?>
        <book>
                <name>Digital Image Processing</name>
                <author>Gonzales and Woods</author>
                <publisher>Prentice Hall</publisher>
                <description>
                        Basic and advanced techniques in digital image processing
                </description>
        </book>
```

Οι λέξεις που περικλείονται από " < > " στοιχεία (elements). Κάθε στοιχείο πρέπει να ανοίγει και να κλείνει με ένα " / " χρησιμοποιώντας πάντα την ίδια λέξη. Τα στοιχεία που περιέχουν

υποστοιχεία λέγονται κόμβοι (nodes). Το πρώτο στοιχείο στο έγγραφο λέγεται ρίζα (root element) εξαιρώντας το στοιχείο που δηλώνει τον τύπο του εγγράφου (<? xml version=”1.0” encoding=”UTF-8”?>). Η διαμόρφωση του XML εγγράφου γίνεται σε επίπεδα. Έτσι στο παραπάνω παράδειγμα το στοιχείο <book> είναι η ρίζα του εγγράφου και περιέχει τα στοιχεία παιδιά (child elements) <name>,<author>,<publisher> και <description>. Επίσης είναι και κόμβος και στοιχείο πατέρας (parent element).

SOAP

Οι περισσότερες τρέχουσες εφαρμογές της WST χρησιμοποιούν το πρωτόκολλο SOAP για την επικοινωνία μεταξύ των WS Registries, των απομακρυσμένων WSs και των εφαρμογών χρηστών-πελατών. Το SOAP, τυπικά γνωστό ως απλό πρωτόκολλο πρόσβασης αντικειμένων (Simple Object Access Protocol), αναπτύχθηκε αρχικά από τις Microsoft, Userland και DevelopMentor, αλλά τώρα εξελίσσετε από την W3C. Είναι ένα ελαφρύ πρωτόκολλο, βασισμένο στην XML γλώσσα για τη μεταφορά δομημένων δεδομένων και πληροφοριών μέσω ενός δικτύου υπολογιστών κατά τρόπο κάθε φορά μοναδικό για κάθε τύπο δεδομένων (stateless). Τα μηνύματα SOAP μπορούν να μεταφερθούν, συχνά μέσω ενδιάμεσων σταθμών, χρησιμοποιώντας το πρωτόκολλο HTTP, SMTP ή άλλα κατάλληλα πρωτόκολλα δικτύων

Το SOAP 1.2, είναι η τελευταία έκδοση, αποτελείται από έναν φάκελο (envelope) στον οποίο εσωκλείονται τα δεδομένα (WSDL στην περίπτωση WST) και ένα μηχανισμό αίτησης-απάντησης για τη μεταφορά των δεδομένων. Τα προαιρετικά συστατικά του περιλαμβάνουν τα χαρακτηριστικά επέκτασης, όπως το RPC (Remote Procedure Call) για την κλήση των απομακρυσμένων υπηρεσιών και τις μεθόδους ένωσης με τα πρωτόκολλα δικτύων. Επίσης παρέχεται ένας μηχανισμός για να εξετάζει τα σφάλματα που μπορεί να προκύψουν.
Αν και πλήρως μοναδικό για κάθε διαδικασία μετάδοσης (stateless), το SOAP μπορεί να χρησιμοποιηθεί για να υποστηρίξει πολύπλοκες αλληλεπιδράσεις που περιλαμβάνουν τον αιτούντα χρήστη στο SOAP, τους ενδιάμεσους σταθμούς και τους κόμβους αποδεκτών, μέσω της χρήσης των προτύπων ανταλλαγής μηνυμάτων MEPs(Message Exchange Patterns).
Τα MEPs χρησιμοποιούν μηχανισμούς ελέγχου στα κατώτερα πρωτόκολλα μεταφοράς ή συγκεκριμένες πληροφορίες που υπάρχουν στις επικεφαλίδες των SOAP μηνυμάτων, για να επιτρέψουν πιο πολύπλοκες αλληλεπιδράσεις. Ένα παράδειγμα SOAP μηνύματος είναι τα παρακάτω.

Μήνυμα SOAP πελάτη:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetails xmlns="http://warehouse.example.com/ws">
      <productID>827635</productID>
    </getProductDetails>
  </soap:Body>
 </soap:Envelope>
```

Μήνυμα SOAP εξυπηρετητή:

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <getProductDetailsResponse xmlns="http://warehouse.example.com/ws">
      <getProductDetailsResult>
        <productName>Toptimate 3-Piece Set</productName>
        <productID>827635</productID>
        <description>3-Piece luggage set. Black Polyester.</description>
        <price currency="NIS">96.50</price>
        <inStock>true</inStock>
      </getProductDetailsResult>
```

```
    </getProductDetailsResponse>
  </soap:Body>
</soap:Envelope>
```

WSDL

Η WSDL (Web Service Definition Language) παρέχει τις λεπτομέρειες για το πώς θα γίνει η επικοινωνία με μια απομακρυσμένη WS. Χρησιμοποιώντας τη γλώσσα XML, περιγράφει πώς να ερμηνευτούν τα μηνύματα, πώς να έρθουν σε επαφή με τη WS και ποια πρωτόκολλα θα χρησιμοποιηθούν. Διευκρινίζοντας ακριβώς πώς να ερμηνεύσουν τα στοιχεία στα SOAP μηνύματα που συνδέονται με μια δεδομένη WS, η WSDL βοηθάει ώστε να αποφευχθεί η παρερμηνεία των στοιχείων μεταξύ του πελάτη και της υπηρεσίας.

UDDI

Το UDDI (Universal Data Description and Integration) παρέχει μια αποθήκη πληροφοριών και μια υπηρεσία ερωτημάτων (query service) για WSs.
Αναπτύχθηκε αρχικά με στόχο ένα κεντρικό "καθολικό επιχειρησιακό αρχείο καταχωρήσεων" για να ενεργήσει ως κύριος κατάλογος όλων των υπηρεσιών ηλεκτρονικού εμπορίου διαθέσιμων μέσω του Διαδικτύου. Εντούτοις, αναγνωρίζοντας ότι θα υπάρξουν πολλές διαφορετικές εφαρμογές της WST, μερικές διαθέσιμες μέσω του Διαδικτύου και άλλες διαθέσιμες μόνο σε ενδοδίκτυα και προσωπικά δίκτυα που χρησιμοποιούν τα πρωτόκολλα του Διαδικτύου (Extranets) , το UDDI έχει εξελιχθεί για να υποστηρίζει μια ομοσπονδία από WS Registries με ένα κεντρικό επιχειρησιακό UDDI αρχείο καταχωρίσεων (UBR).
Οι WS Registries είναι αποθήκες δεδομένων XML, οι οποίες περιέχουν διάφορους διαφορετικούς τύπους εγγραφών. Όπως και οι μεμονωμένες περιγραφές μιας WS, έτσι και αυτές αποθηκεύουν τις πληροφορίες σχετικά με τον παροχέα της WS όπως λεπτομέρειες επικοινωνίας με αυτόν, τύπους υπηρεσιών και λεπτομέρειες των τυποποιημένων πρωτοκόλλων που χρησιμοποιούνται.
Η πρόσβαση στις εγγραφές της γίνεται μέσω ενός κλειδιού, το οποίο είναι συχνά, αν και όχι απαραιτήτως, βασισμένο σε DNS (Domain Name Server) εγγραφές. Εξαρτάται από τους κατόχους των WS Registries να εφαρμόσουν τις κατάλληλες μεθόδους για να εξασφαλίσει την μοναδικότητα των δημόσια διαθέσιμων κλειδιών.
Οι παροχείς και οι πελάτες της WS αλληλεπιδρούν με το UDDI μέσω ενός συνόλου του APIs (Application Programming Interface) για να δημοσιεύσουν και να ανακαλύψουν τις λεπτομέρειες της WS χρησιμοποιώντας το SOAP. Τα APIs στα τρέχοντα πρότυπα (UDDI 3.0) μπορούν να ταξινομηθούν: στην έρευνα, τη ενσωμάτωση, την ασφάλεια, τη επιτηρούμενη μεταφορά, τη συνδρομή-εγγραφή, την αντένσταση, τον διαχειριστή συνδρομών-εγγραφών και το σύνολο τιμών.

Επιχειρησιακές διαδικασίες, δεδομένα και πρότυπα ελέγχου

Αν και το SOAP και το WSDL παρέχουν έναν μηχανισμό για την τεχνική διαλειτουργικότητα, αυτό δεν είναι επαρκές για να επιτρέψει την ολοκλήρωση των σύνθετων επιχειρησιακών διαδικασιών. Απαιτούνται υψηλότερου επιπέδου μηχανισμοί διαλειτουργικότητας. Η ακριβής μορφή θα εξαρτηθεί από τον τύπο της επιχειρησιακής εφαρμογής.
Η σημασιολογική διαλειτουργικότητα είναι ένα από τα βασικά ζητήματα. Παραδείγματος χάριν, ο όρος “φοιτητής” αναφέρεται μόνο σε έναν πλήρους απασχόλησης προπτυχιακό φοιτητή ή αυτό περιλαμβάνει επίσης τους μαθητευόμενους από απόσταση; Ένας μηχανισμός για τη σημασιολογία των στοιχείων στο μήνυμα SOAP ή των παραμέτρων SOAP-RPC

πρέπει να δοθεί.
Διαφορετικά σενάρια εφαρμογών θα απαιτήσουν διαφορετικά σχέδια ανταλλαγής μηνυμάτων. Παραδείγματος χάριν, οι μακρινές κλήσεις διαδικασίας RPC, που παρέχονται από το Soap-rpc είναι χρήσιμες σε σενάρια όπου οι πελάτες επιθυμούν να επικαλεσθούν τις υπηρεσίες που προσφέρονται από τους μακρινούς προμηθευτές της WS. Εντούτοις, στα κρίσιμα περιβάλλοντα συστημάτων, μια αρχιτεκτονική βασισμένη σε γεγονότα που πραγματοποιούνται μπορεί να απαιτηθεί. Σε τέτοια σενάρια ο μηχανισμός αίτησης-απάντησης του SOAP μπορεί να χρησιμοποιηθεί μαζί με τα στοιχεία εφαρμογών και τους ενδιάμεσους σταθμούς SOAP για να παραχθεί η επιθυμητή αρχιτεκτονική ελέγχου.

Πλαίσια ανάπτυξης της Web Service Technology

Ενώ η WST σχεδιάζεται για να είναι διαλειτουργική και επαναχρησιμοποιήσιμη, οι ενότητες που γράφονται σε ένα περιβάλλον ανάπτυξης εφαρμογών δεν μπορούν να μεταφερθούν εύκολα σε άλλο. Οι υπεύθυνοι για την ανάπτυξη πρέπει να κάνουν μια στρατηγική επιλογή ως προς το ποιο εκ των δύο ανταγωνιστικών πλαισίων εφαρμογής που υποστηρίζουν το WST ταιριάζει περισσότερο στις ανάγκες τους.
Η εφαρμογή Microsoft .NET., που σχεδιάστηκε συγκεκριμένα έχοντας υπόψη τις WS, είναι ένα σχετικά νέο, ιδιόκτητο σύστημα ανάπτυξης εφαρμογών που αποτελείται από κεντρικούς υπολογιστές, πελάτες, και υπηρεσίες, βιβλιοθήκες εκτέλεσης κώδικα, γλώσσες προγραμματισμού όπως C#, VB.NET και APIs. Διάφορα προϊόντα τρίτων που στηρίζονται στη .NET είναι επίσης διαθέσιμα.
Η πλατφόρμα της SUN, JAVA 2 Enterprise Edition (J2EE) είναι ένα δωρεάν για όλους σύνολο προδιαγραφών που επιτρέπει τη σχεδίαση, την ανάπτυξη και την εκτέλεση των διανεμημένων εφαρμογών της JAVA. Το J2EE είναι μια πλατφόρμα δημιουργίας εφαρμογών, η οποία είναι περισσότερο καιρό στον τομέα της πληροφορικής από ότι η .NET. Εντούτοις, χρειάζεται συνεχώς την προσθήκη νέων APIs για να ικανοποιήσει τις ανάγκες της WST. Το J2EE παρέχει μια ευρεία βάση εφαρμογών προς πώληση και αρκετές δωρεάν εφαρμογές είναι διαθέσιμες. Τα περισσότερα ιδρύματα ήδη έχουν επιλέξει ένα εσωτερικό περιβάλλον ανάπτυξης και ένα εκπαιδευμένο προσωπικό σε αυτό. Δεν υπάρχει κανένα σημαντικό πλεονέκτημα από καθεμία πλατφόρμα που θα έκανε αίσθηση ως προς την εισαγωγή στην WST.

Εμπορικά Προϊόντα

Τα προϊόντα της WST είναι πια πάρα πολλά για να απαριθμηθούν μέσα σε αυτήν την αναφορά. Αξίζει εντούτοις να τονισθούν μερικοί σημαντικοί φορείς που πρωταγωνιστούν στον χώρο αλλά και μερικά προϊόντα που είναι διαθέσιμα . Οι σημαντικότεροι φορείς πίσω από τη WST είναι η Microsoft, η IBM, η SUN, τα συστήματα BEA και η HP. Αυτές οι επιχειρήσεις προσφέρουν ένα ευρύ φάσμα λογισμικού και υλικού σχετικό με τη WST, συμπεριλαμβανομένων των εφαρμογών της WS, των κεντρικών υπολογιστών, των πυλών, των περιβαλλόντων ανάπτυξης και ένα σύνολο εργαλείων. Είναι ενεργοί στους διάφορους οργανισμούς που συμμετέχουν στην τυποποίηση, την ανάπτυξη και την εφαρμογή της WST.
Η WST έχει εισβάλει επίσης σε περισσότερο ειδικευμένες εφαρμογές στην αγορά. Εκτός από τις προαναφερθείσες επιχειρήσεις, άλλοι σημαντικοί φορείς στην αγορά των EAI (Enterprise Application Integration) όπως η Iona, η SAP, η Sybase και η See Beyond προσφέρουν ήδη την λειτουργικότητα τωνWSs στα προϊόντα τους. Ομοίως, οι σημαντικότερες εταιρίες παραγωγής προϊόντων βάσεων δεδομένων όπως η Oracle προσφέρουν τη λειτουργία των WSs στα σύνολα των προϊόντων τους, με άλλες εταιρίες να ακολουθούν.
Οι εμπορικές εφαρμογές των WSs έχουν ήδη αναπτύχθεί. Οι κύριοι φορείς παροχής υπηρεσιών όπως η BT (Broadband Technologies) έχουν αναγγείλει τα σχέδια για να αναπτύξουν τις προσεγγίσεις της WST στον τομέα εξυπηρετήσεις πελατών. Η ανάπτυξη

εμπορικών WSs έχει οδηγήσει σε μία νέα διάδοση των διοικητικών υπηρεσιών των WSs που προσφέρονται από τις επιχειρήσεις.

Πρότυπα

Τα WST προγράμματα που αναπτύχθηκαν αρχικά εξελίχθηκαν μέσω ομάδων προμηθευτών που σκοπό είχαν την δημοσίευση τους. Ενώ αυτό έχει επιτρέψει τη γρήγορη πρόοδο, η έλλειψη γενικού συντονισμού των προτύπων και του φάσματος ανταγωνιστικών εμπορικών ενδιαφερόντων ήταν ένας λόγος ανησυχίας.
Το 2002, ο συνεταιρισμός του παγκόσμιου ιστού (W3C) ανέλαβε την ολική διαχείριση των εμπλεκόμενων προτύπων ανάπτυξης. Ενώ αυτό έπρεπε να εξασφαλίσει ότι οι υπηρεσίες Ιστού (WSs) θα ενσωματώνονταν στη γενική δομή του παγκόσμιου Ιστού, η επίσημη δομή επιτροπών παρουσίασε καθυστερήσεις στην πρόοδο των προτύπων. Το W3C αναπτύσσει ενεργά την αρχιτεκτονική των WSs,το πρωτόκολλο XML (SOAP 1.2), το WSDL και το WS Choreography (WSCI), το οποίο στοχεύουν να αντιμετωπίσει τα ζητήματα ροής και ελέγχου μεταξύ των WSs.
Το W3C, που αναγνωρίζει την ανάγκη για επιχειρησιακή λογική, ερευνά αυτό το ζήτημα και διάφορα, υψηλότερου επιπέδου, πρότυπα εξετάζονται για τον έλεγχο, τη λογική, τη σημασιολογία και την ασφάλεια κατά τη λειτουργία τους στη WS Chorography.
Ο OASIS (Organization for the Advancement of Structured Information Standards) είναι ένας άλλος βασικός φορέας στην ανάπτυξη της WST. Ενώ ο OASIS ήταν αρχικά αντίθετος στις εξελίξεις των WSs, έχει αναλάβει από τότε την ανάπτυξη του UDDI καθώς επίσης και την προώθηση των νέων πρωτοβουλιών WSRP (Web Services for Remote Portal) και WSS(Web Services Security). Επίσης, τα πρότυπα προχωρούν μέσω των επιτροπών, οι οποίες μπορούν να ορίσουν τις καθυστερήσεις ανάπτυξης.
Η προοπτική της επιβράδυνσης της ανάπτυξης των WSs, λόγω των όλο και περισσότερο περίπλοκων διαδικασιών τυποποίησης, έχει οδηγήσει στο σχηματισμό, με την καθοδήγηση των προμηθευτών και των υπεύθυνων ανάπτυξης, της οργάνωσης διαλειτουργικότητας υπηρεσιών Ιστού WS-I (Web Services Interoperability Organization). Η WS-J δεν στοχεύει στην ανάπτυξη προτύπων, αλλά μάλλον στην συλλογή των προφίλ των σχετικών προτύπων των WSs για να επιτρέψουν τη δοκιμή απόδοσης και την πιστοποίηση της συμβατότητας, συνδέοντας κατά συνέπεια το χάσμα μεταξύ των οργανισμών προτύπων και των εμπορικών εφαρμογών.
Κάθε ένα από τα διάφορα διαφορετικά επιχειρησιακά σενάρια θα απαιτήσουν τις δικές του διεπαφές της WST. Άλλα πρότυπα, που εξετάζουν τις συγκεκριμένες "επιχειρησιακές διαδικασίες" όπως τα πρότυπα εκμάθησης IMS, αναμένονται να αναπτύξουν τα πρότυπα διεπαφών της WST.

Εκτιμήσεις

Τομείς πιθανής εφαρμογής

Η WST θα αποδείξει έναν οικονομικώς αποδοτικό τρόπο στην ανάπτυξη νέων υπηρεσιών για το προσωπικό και τους σπουδαστές, που παραδίδεται συχνά μέσω μιας θεσμικής πύλης, οι οποία ενσωματώνετε με υπάρχουσες εφαρμογές. Τα ιδρύματα μπορούν εγκαίρως να πάρουν μια στρατηγική απόφαση να χρησιμοποιήσουν τη WST ως τμήμα των στρατηγικών πληροφοριών τους όπως στην περίπτωση του MIT και την πρωτοβουλία του στο πρόγραμμα πλαισίου iCampus. Άλλες πιο διαφορετικές χρήσεις, όπως οι εφαρμογές ερευνητικών οργανισμών και εφαρμογών μάρκετινγκ, μπορούν επίσης τελικά να προκύψουν.
Η ηλεκτρονική εκμάθηση (e-learning) είναι επίσης ένας τομέας με προοπτική εφαρμογής της WST. Τα αποτελέσματα από αυτά τα προγράμματα και οι αναδυόμενες απαιτήσεις από τα ιδρύματα, έχουν οδηγήσει στην εκτίμηση της ευρύτερης ιδέας μιας αρχιτεκτονικής

διαλειτουργικότητας. Ενώ καμία απόφαση δεν έχει ληφθεί ακόμα σχετικά με τις τεχνολογίες, το WST θα μπορούσε να παρέχει πολλά από τα χαρακτηριστικά γνωρίσματα που απαιτούνται αν και η έλλειψη αφθονίας της διαθέσιμης λειτουργικότητας προς το παρόν είναι ένα πιθανό πρόβλημα. Είναι. Περισσότερη εμπειρία της WST απαιτείται προτού να μπορέσει να ληφθεί οποιαδήποτε τελική απόφαση.

Πλεονεκτήματα

Η αποτελεσματικότητα ευκολίας και δαπανών της ανάπτυξης και της εφαρμογής των νέων υπηρεσιών, που ενδεχομένως ενσωματώνει ποικίλους φορείς παροχής υπηρεσιών, θα είναι ένα από τα κύρια οφέλη της WST.
Με τις μεταβαλλόμενες απαιτήσεις και αυξημένη ανάγκη για αλληλεπίδραση, η δυνατότητα να αναπτυχθούν βασικές προτυποποιημένες υπηρεσίες για τη μεταφορά των δεδομένων και πληροφοριών και η διανομή των πόρων μέσω του τομέα, χρησιμοποιώντας τη WST θα είναι συμφέρουσα.
Κοιτάζοντας στο μέλλον, οι επιχειρησιακές απαιτήσεις θα γίνουν όλο και περισσότερο σύνθετες και προσανατολισμένες στις υπηρεσίες, απαιτώντας αλληλεπίδραση μεταξύ πολλών διαφορετικών συνεργατών, συχνά μέσω παροδικών συμμαχιών. Τα μελλοντικά συστήματα πληροφοριών θα απαιτείται να είναι ευκίνητα, ευφυής, ικανά να παρέχουν έγκαιρες, προσαρμοσμένες, εξατομικευμένες υπηρεσίες με ελάχιστη ανθρώπινη επέμβαση, μέσω ενός πλήθους αλληλεπιδρώντων δικτυωμένων εφαρμογών από τους ποικίλους φορείς παροχής υπηρεσιών.

SESSIONS

Στην επιστήμη υπολογιστών, και ειδικότερα στα δίκτυα υπολογιστών, το session αναφέρεται στην πεπερασμένη σύνδεση ενός χρήστη (client) και ενός εξυπηρετητή (server) κατά την οποία γίνεται ανταλλαγή αρκετών μηνυμάτων-πακέτων μεταξύ τους . Ένα session συνήθως εγκαθίσταται σαν ένα επιπλέον επίπεδο σε ένα πρωτόκολλο δικτύου υπολογιστών.

Στην περίπτωση των πρωτόκολλων μεταφοράς, όπου δεν υπάρχει κάποιο τυπικό επίπεδο Session (π.χ.UDP), ή στην περίπτωση όπου τα sessions, σε ένα επίπεδο Session, έχουν μικρή διάρκεια ζωής (π.χ.HTTP), τα sessions τοποθετούνται σε μία εφαρμογή η οποία είναι τοποθετημένη σε κάποιο υψηλότερο επίπεδο και διαχειρίζονται χρησιμοποιώντας μεθόδους, οι οποίες ορίζονται στα δεδομένα τα οποία ανταλλάσσονται μεταξύ των δύο πλευρών. Για παράδειγμα, σε μία κοινή HTTP συναλλαγή μεταξύ ενός web browser και ενός απομακρυσμένου εξυπηρετητή μπορεί να συμπεριλαμβάνονται τα γνωστά HTTP Cookies τα οποία προσδιορίζουν μία κατάσταση όπως είναι η μοναδική ταυτότητα του session η αλλιώς το Session ID, το οποίο περιέχει πληροφορίες σχετικά με τον χρήστη η το επίπεδο εξουσιοδότησης και πρόσβασης του χρήστη στον συγκεκριμένο εξυπηρετητή.

Κατά την σύνδεση όμως ενός χρήστη σε ένα εξυπηρετητή, ο οποίος βρίσκεται σε ένα σύμπλεγμα εξυπηρετητών, δημιουργείται ένα πρόβλημα διατήρησης της συνοχής της σύνδεσης καθώς οι εξυπηρετητές πρέπει να διατηρήσουν ίδια την κατάσταση ενός session. Ο χρήστης θα πρέπει να οδηγείται στον ίδιο εξυπηρετητή κατά την διάρκεια που είναι ανοιχτό

το session, ή ο εξυπηρετητής θα πρέπει να μεταδίδει πληροφορίες σχετικά με το session στο οποίο βρίσκετε ο χρήστης. Στην περίπτωση που δεν πραγματοποιηθεί αυτό ο χρήστης θα επανασυνδεθεί σε διαφορετικό εξυπηρετητή από αυτόν που είχε ξεκινήσει το session, προκαλώντας έτσι πρόβλημα αφού ο νέος εξυπηρετητής δεν έχει πρόσβαση στις πληροφορίες του εξυπηρετητή στον οποίο είχε ξεκινήσει το session.

Τα sessions χωρίζονται σε αυτά του χρήστη (Client Side) και σε αυτά του εξυπηρετητή (Server Side).

Server Side Sessions

Τα sessions από την πλευρά του εξυπηρετητή είναι βολικά και αποτελεσματικά, αλλά μπορούν να γίνουν δύσκολα στη διαχείριση όταν έχουμε συνδέσεις σε συστήματα υψηλής διαθεσιμότητας και ζήτησης και καθόλου εύχρηστα σε εμφωλευμένα συστήματα χωρίς δυνατότητα τοπικής αποθήκευσης σε κάποιο μέσο.

Το πρόβλημα με τα συστήματα υψηλής διαθεσιμότητας και ζήτησης μπορεί να λυθεί χρησιμοποιώντας κάποιο κοινόχρηστο σύστημα αποθήκευσης ή εφαρμόζοντας μια εξαναγκασμένη αναζήτηση μεταξύ χρήστη και κάθε εξυπηρετητή σε μία συστάδα από υπολογιστές, κάνοντας έτσι έναν συμβιβασμό μεταξύ της αποδοτικότητας του συστήματος και του φόρτου εργασίας σε αυτό όσον αφορά τη διανομή των δεδομένων. Από την άλλη, το πρόβλημα που παρουσιάζεται στα συστήματα χωρίς μέσα αποθήκευσης μπορεί να λυθεί δεσμεύοντας μία ποσότητα από την κεντρική μνήμη του συστήματος και χρησιμοποιώντας την σαν μέσο αποθήκευσης. Ωστόσο αυτή η μέθοδος είναι εφαρμόσιμη σε συστήματα με περιορισμένο αριθμό χρηστών.

Και στα δύο προηγούμενα σενάρια που αναφέρθηκαν, η χρησιμοποίηση sessions από την πλευρά του χρήστη (Client Side Sessions) θα μας παρείχε δικτυακά πλεονεκτήματα σε σχέση με τα session από την πλευρά του εξυπηρετητή (Server Side Session). Στην πρώτη περίπτωση θα μείωνε τους περιορισμούς που θα έπρεπε να εφαρμοστούν στους αλγορίθμους διαχείρισης της κίνησης και επιφόρτισης του συστήματος, και στην δεύτερη περίπτωση θα επέτρεπε την χρήση sessions σε εφαρμογές οι οποίες δεν μπορούν να χρησιμοποιήσουν μέσα αποθήκευσης η να δεσμεύσουν μνήμη από το σύστημα.

Client Side Sessions

Χρησιμοποιώντας session στην πλευρά του πελάτη θα πρέπει με κάποιον τρόπο να αποθηκεύονται οι πληροφορίες του session. Υπάρχουν διάφορες τεχνικές που μπορούν να εφαρμοστούν γι' αυτό τον σκοπό, οι ποιο δημοφιλής όμως είναι οι δύο που περιγράφονται παρακάτω.

- **Αποθήκευση τον πληροφοριών σε Cookies**: Τα cookies είναι μικρά κομμάτια δεδομένων τα οποία ενσωματώνουν οι εξυπηρετητές στα μηνύματα απάντησης προς τους χρήστες και περιέχουν όλες τις πληροφορίες που είναι απαραίτητες για την επικοινωνία μεταξύ τους. Κάθε φορά που ο χρήστης-πελάτης αποστέλλει πληροφορίες στον εξυπηρετητή, ενσωματώνει στην επικεφαλίδα του μηνύματος όλα τα προηγούμενα αποθηκευμένα cookies που έχει δεχθεί από αυτόν. Οι πληροφορίες του session μπορούν να αποθηκευθούν σε cookies τοπικά στον χρήστη έτσι ώστε ο εξυπηρετητής να μπορεί να τις χρησιμοποιήσει κάθε φορά που διαμορφώνει ένα μήνυμα απάντησης, αφού αυτές αποστέλλονται σε αυτόν μέσω των μηνυμάτων του χρήστη.
- **Επανεγγραφή URLs (Uniform Recourse Locators) ώστε να εμπεριέχουν πληροφορίες του session:** Η συγκεκριμένη τεχνική κωδικοποιεί κάθε διεύθυνση αποστολής (URL) έτσι ώστε να περιέχει τις πληροφορίες του session. Κάθε φορά που μία τέτοια διεύθυνση δεν αναφέρεται κάπου, ο χρήστης-πελάτης αποστέλλει μία αίτηση επιστροφής στον εξυπηρετητή, ο οποίος λαμβάνει τις πληροφορίες που του

έστειλε ο χρήστης-πελάτης σαν παραμέτρους στο μήνυμα, τις οποίες πρέπει να ανακτήσει από το μήνυμα και να τις χρησιμοποιήσει για να ολοκληρώσει την ζητούμενη υπηρεσία.

Είναι πολύ σημαντικό οι πληροφορίες αυτές να παραμένουν εμπιστευτικές και μακριά από παρεμβάσεις τρίτων. Για το λόγο αυτό χρησιμοποιούνται πάντα τεχνικές κρυπτογράφησης έτσι ώστε να αποφευχθεί οποιαδήποτε ανεπιθύμητη υποκλοπή.

Session Token

Το session token είναι μοναδική ταυτότητα η οποία δημιουργείτε και αποστέλλεται από έναν εξυπηρετητή σε έναν χρήστη-πελάτη έτσι ώστε να είναι δυνατή η αναγνώριση του session. Ο χρήστης-πελάτης συνήθως αποθηκεύει και αποστέλλει αυτή την μοναδική ταυτότητα ως cookie ή μέσω παραμέτρου στις αιτήσεις επιστροφής (GET) στον εξυπηρετητή. Ο σκοπός για τον οποίο χρησιμοποιούνται τα session tokens είναι επειδή ο χρήστης-πελάτης (Client) το μόνο που έχει να κάνει είναι να διαχειρισθεί την ταυτότητα αφού όλα τα δεδομένα που αφορούν το session αποθηκευμένα στον εξυπηρετητή (server) συνδεδεμένα με αυτήν την ταυτότητα.
Αυτό που ενδιαφέρει στην δική μας περίπτωση, είναι η διαχείριση των sessions σε μια εφαρμογή και κατά πόσο είναι εφικτό πολλαπλοί χρήστες να μπορούν να χρησιμοποιήσουν την ίδια υπηρεσία, επικοινωνώντας με αυτή μέσω του ίδιου session έτσι ώστε ο κάθε χρήστης να έχει την δυνατότητα να βλέπει τις αλλαγές που πραγματοποιούνται στις μεταβλητές και τα αντικείμενα της υπηρεσίας αυτής. Για να πραγματοποιηθεί αυτό πρέπει είτε να αποθηκεύονται τοπικά στον εξυπηρετητή σε μια βάση έτσι ώστε να είναι δυνατή η ανάκτησή τους κάθε φορά που ζητηθούν από κάποιον χρήστη, είτε να αποθηκεύονται στην μνήμη του εξυπηρετητή και να χρησιμοποιούνται όποτε ζητηθούν.
Στην πρώτη περίπτωση, οι πληροφορίες θα παραμείνουν αποθηκευμένες τοπικά στον εξυπηρετητή σε μια βάση κάτι το οποίο δίνει επιπλέον ασφάλεια στην διατήρηση των πληροφοριών που μας ενδιαφέρουν, σε περίπτωση δυσλειτουργίας του συστήματος. Στην δεύτερη περίπτωση, το σύστημα θα λειτουργήσει ποιο αποδοτικά, αφού όλες οι πληροφορίες θα είναι άμεσα προσβάσιμες από τη μνήμη του συστήματος. Σε περίπτωση όμως δυσλειτουργίας θα έχουμε απώλεια δεδομένων και θα χρειαστεί η επαναποστολή τους από τους χρήστες. Και στις δύο περιπτώσεις, για να λειτουργήσει το σωστά το σύστημα θα πρέπει να υπάρχει μόνο ένα αντίγραφο (Instance) της υπηρεσίας κάθε φορά και όλοι οι χρήστες να μπορούν να συνδεθούν με αυτή. Η υπηρεσία πρέπει υποστηρίζει ασύγχρονη μετάδοση και έτσι ώστε όλοι οι χρήστες να μπορούν να έχουν πρόσβαση ταυτόχρονα σε όλους τους πόρους της υπηρεσίας. Η διάρκεια ζωής της υπηρεσίας πρέπει να είναι ίση με αυτή του εξυπηρετητή στον οποίο θα είναι αποθηκευμένη και θα λειτουργεί.

Τεχνολογίες ανάπτυξης Web Services

Για την ανάπτυξη εφαρμογών βασισμένων στις Web Services εκτός από τις πλατφόρμες ανάπτυξης, οι οποίες αναφέρθηκαν παραπάνω (Microsoft .NET, C#, VB.NET, JAVA 2 Enterprise Edition J2EE), με τις οποίες θα γίνει η σχεδίαση, η ανάπτυξη και η εκτέλεση τους, απαιτείται και η χρησιμοποίηση μιας ενδιάμεσης τεχνολογίας η οποία θα φιλοξενεί της υπηρεσίες και θα παρέχει τις κατάλληλες μεθόδους και διαδικασίες για την διαχείριση των συνδέσεων μεταξύ των πελατών και των παροχέων των WSs και θα επιτρέψουν την διαχείριση, την κωδικοποίηση-αποκωδικοποίηση, την δημιουργία και την διαμεταγωγή των μηνυμάτων πληροφορίας από και προς τους πελάτες και τους εξυπηρετητές. Επίσης κάθε τέτοια ενδιάμεση τεχνολογία προαπαιτεί την χρήση περαιτέρω τεχνολογιών και προτύπων για την λειτουργία της.

Υπάρχουν αρκετές τέτοιες ενδιάμεσες τεχνολογίες από τις οποίες οι πιο γνωστές είναι οι Apache AXIS (και η τελευταία έκδοσή του AXIS2),AJAX,C WS Core,TEN Technology. Η κάθε μια από αυτές αποσκοπεί και σε κάποιον πιο ειδικό σκοπό για την ανάπτυξη WSs δημιουργώντας έτσι μια αρκετά μεγάλη γκάμα επιλογής ανάλογα με την επιθυμία των ομάδων ανάπτυξης και το επιθυμητό αποτέλεσμα.
Η τεχνολογίες αυτές αναφέρονται και σαν στοίβες υπηρεσιών ή web service containers διότι σε αυτές θα αποθηκευθούν οι WSs που θα δημοσιοποιηθούν στο δίκτυο η τον παγκόσμιο ιστό. Έτσι κάθε μια από αυτές προσφέρουν μία αποθήκη στην οποία θα αποθηκευθούν οι υπηρεσίες και ένα σύνολο ρυθμίσεων οι οποίες θα δηλώνουν τον τρόπο με τον οποίο θα πραγματοποιούνται οι απομακρυσμένες συνδέσεις, τον τρόπο λειτουργίας και συμπεριφοράς των υπηρεσιών, τα πρωτόκολλα με τα οποία θα συνεργάζονται, τους χρόνους αναμονής για την ολοκλήρωση των sessions με τους πελάτες και ένα σύνολο πιο ειδικών ρυθμίσεων που αφορούν την όλη συμπεριφοράς της λειτουργίας των εφαρμογών που θα χρησιμοποιούν τις υπηρεσίες.
Ο τρόπος με τον οποίο δημοσιεύονται οι υπηρεσίες είναι συγκεκριμένος και πρέπει να πραγματοποιηθεί ακριβώς όπως το απαιτεί η συγκεκριμένη τεχνολογία υποστήριξης και δημοσίευσής της. Ειδικό ρόλο παίζει σε αυτό και η πλατφόρμα ανάπτυξης των υπηρεσιών, καθώς ανάλογα με την γλώσσα προγραμματισμού με την οποία έχουν αναπτυχθεί απαιτείται να εκτελεσθούν και οι ανάλογες διαδικασίες για την τοποθέτησή της στον web service container.
Είναι ξεκάθαρο ότι η επιλογή πλατφόρμας πάνω στην οποία θα γίνει η ανάπτυξη των υπηρεσιών, και για μεγαλύτερη ευκολία και καλύτερη συμβατότητα και η εφαρμογές που θα τις χρησιμοποιούν, αν και αυτό δεν είναι απαραίτητο καθώς υπάρχει μεγάλη υποστήριξη των WSs ανάμεσα σε διαφορετικές τεχνολογίες, παίζει αρκετά σημαντικό ρόλο.
Άρα οι επιλογή της πλατφόρμας ανάπτυξης της web service και του web service container στον οποίο θα φιλοξενηθεί, είναι αυτή που θα καθορίσει τον τρόπο με το οποίο θα αναπτυχθεί και θα ολοκληρωθεί η υπηρεσία.
Στη αναφορά αυτή θα ασχοληθούμε με την τεχνολογία AXIS2 και σαν πλατφόρμα ανάπτυξης της web service θα χρησιμοποιήσουμε την JAVA πάνω στην οποία θα είναι γραμμένα και όλα τα παραδείγματα κώδικα.

Apache AXIS2

Εισαγωγική Περίληψη

Ο Web Service Stack ή αλλιώς Web Service Container έχει εξελιχθεί γρήγορα κατά τη διάρκεια των τελευταίων τριών ετών, και στο διάστημα αυτό η ομάδα του AXIS άρχισε την ανάπτυξη της επόμενης γενεάς του Apache Axis. Το Apache Axis2 είναι βασισμένο σε μια νέα αρχιτεκτονική, παγιώνοντας την εμπειρία των προκατόχων, του Apache SOAP και του Apache Axis. Η πρώτη βασική έκδοση του Axis2 κυκλοφόρησε τον Φεβρουάριο του 2005, και η ομάδα ανάπτυξής του κυκλοφόρησε την έκδοση 1.0 στους επόμενους μήνες.
Η πρώτη γενιά των web services, που προέκυψε στις αρχές του 2000, ασχολούνταν με την εκτέλεση μερικών ελεγχόμενων αλληλεπιδράσεων μέσω του παγκόσμιου Ιστού. Αυτή τη στιγμή, ο στόχος των μηχανών SOAP (π.χ.Apache Soap) ήταν να καταδειχθεί ότι οι υπηρεσίες παγκόσμιου Ιστού ήταν κάτι περισσότερο από μια έννοια. Επομένως, το υλικο και λογισμικό των WSs που αναπτύχθηκε σε αυτήν την εποχή εστίασε στην ακρίβεια αποτελεσμάτων παρά την απόδοση.
Η δεύτερη γενιά των WSs θεωρήθηκε εφικτή τεχνολογία. Οδήγησε στην κατασκευή πολλών μηχανών SOAP όπως το Apache Axis, το Systinet WASP και το gSOAP, οι οποίες ήταν ικανές πολλών διαφορετικών μορφών αλληλεπίδρασης. Οι προδιαγραφές τέθηκαν σε ισχύ για σημαντικές πτυχές των WSs και οι περισσότερες μηχανές SOAP ενέμειναν σε αυτές τις κοινές προδιαγραφές, επιτρέποντας κατά συνέπεια τις παράλληλες αλληλεπιδράσεις

διαφορετικών πλατφόρμων. Το Axis ήταν μια της δεύτερης γενιάς μηχανές SOAP που έγιναν σύντομα δημοφιλείς μεταξύ των εργαλείων WSs. Αλλά οι WSs στερούνταν ακόμα την αυτοδυναμία και την αποδοτικότητα που απαιτούσε μια αληθινή πλατφόρμα υλικο-λογισμικού (middleware platform).
Αυτό το πρόβλημα τίθεται ως στόχος προς λύση από την τρίτη γενεά των WSs που αναμένονται να είναι γρήγορες, σταθερές, αποδοτικές, και αξιόπιστες. Είναι καλά ωριμασμένες και οι προδιαγραφές που ελέγχουν τις περισσότερες πτυχές των WSs έχουν βελτιωθεί. Οι δεύτερης γενιάς μηχανές SOAP όπως το Axis δεν είναι πλέον επαρκείς για να τροφοδοτήσουν τις υψηλής απαίτησης WSs νέας γενεάς. Το Axis2 στοχεύει να καλύψει αυτό το κενό.
Το Axis2 διαμορφώνεται από την εμπειρία που αποκτήθηκε από το Axis 1.x και τις προόδους του Web Services Stack στα τελευταία έτη. Στο Axis2 οι υπεύθυνοι για την ανάπτυξη έχουν καταβάλει μια συνειδητή προσπάθεια να εξασφαλίσουν ότι παραδίδει την καλύτερες ταχύτητες απόκρισης και καλύτερη απόδοση μνήμης, με τα πρόσθετα χαρακτηριστικά γνωρίσματα και τις λειτουργίες στην πιο πρόσφατη έκδοση. Ο σχεδιασμός του Axis2 έχει χαρακτηριστικές αλλαγές στους ακόλουθους τομείς:

- Νέο μοντέλο SOAP
- Καλύτερη υποστήριξη των MEPS (Message Exchange Patterns)
- Σύγχρονη και ασύγχρονη συμπεριφορά
- Εξελιγμένο μοντέλο ενσωμάτωσηςς των WSs (deployment method)
- Ενσωματώσιμη δέσμευση δεδομένων (Pluggable Data Binding)
- Εξελιγμένοι διαχειριστές (Πλατφόρμα φιλτραρίσματος)

Το νέο Μοντέλο SOAP

Η κύρια πρόκληση οποιουδήποτε υλικο-λογισμικού συστήματος βασισμένου στον παγκόσμιο Ιστό, είναι να αποφύγει να κρατάει το πλήρες μήνυμα SOAP στη μνήμη. Κρατώντας ολόκληρο το μοντέλο αντικειμένου στη μνήμη κάνει το προγραμματιστικό χειρισμό των αντικειμένων XML ευκολότερο. Εντούτοις, η μνήμη επιβαρύνεται πολύ με αυτή την διαδικασία αφού το μοντέλο αντικειμένου χρειάζεται μια αρκετά μεγάλη ποσότητα μνήμης η οποία ισούται με το μέγεθος του XML εγγράφου που επεξεργάζεται την συγκεκριμένη στιγμή.
Μια από τις μηχανές SOAP πρώτης γενεάς, το Apache SOAP, χρησιμοποιεί, εσωτερικά, ένα μοντέλο αντικειμένου βασισμένο στο DOM (Document Object Model) για να αναπαραστήσει τα έγγραφα XML, όπου οι τεχνικές επεξεργασίας XML αναγκάζουν ολόκληρο το πρότυπο αντικειμένου XML να διαμορφωθεί αμέσως. Το Apache Axis δεύτερης γενιάς άλλαξε το μοντέλο αντικειμένου σε SAX για να αποφύγει τις καταχωρίσεις όλων των πληροφοριών στην μνήμη. Το SAX, εντούτοις, έχει έναν σημαντικό περιορισμό , έχει χτιστεί γύρω από μια τεχνική "ώθησης", κατά την οποία, μόλις ξεκινήσει η διαδικασία ανάλυσης του XML κειμένου τότε αυτή δεν μπορεί να διακοπεί . Για να ξεπεράσει αυτό το εμπόδιο, το Apache Axis έπρεπε να καταγράψει τα γεγονότα συμβάντων του SAX. Έτσι, αποτελεσματικά, το μήνυμα XML πρέπει να κρατηθεί στη μνήμη υπό μορφή γεγονότων SAX, κάνοντας κατά συνέπεια επιφορτίζοντας το Apache Axis με άλλο ένα μοντέλο το οποίο επιβαρύνει την μνήμη.

Το μειονέκτημα αυτών προγραμματιστικών μοντέλων τα οποία επιβαρύνουν την μνήμη γίνεται εμφανές όταν πρέπει να χειριστούν αρχεία XML μεγάλου μεγέθους. Για XML αρχεία μεσαίου μεγέθους η απόδοση είναι αποδεκτή, αλλά όχι για τα μεγαλύτερα έγγραφα XML. Μερικές ταυτόχρονες αιτήσεις με μεγάλου μεγέθους έγγραφα XML είναι ικανές να προκαλέσουν κατάρρευση της μηχανής του SOAP.
Το μειονέκτημα αυτών των εντατικών προτύπων προγραμματισμού μνήμης γίνεται εμφανές όταν πρέπει να χειριστούν τα μεγάλα αρχεία XML. Για XML τεκμηριώνει την απόδοση είναι αποδεκτός, αλλά όχι έτσι για τα μεγαλύτερα έγγραφα XML. Μερικά ταυτόχρονα αιτήματα με τα μεγάλα έγγραφα XML μπορούν να αναγκάσουν τη μηχανή ΣΑΠΟΥΝΙΩΝ για να συντρίψουν.
Το μειονέκτημα αυτών των εντατικών προτύπων προγραμματισμού μνήμης γίνεται εμφανές όταν πρέπει να χειριστούν τα μεγάλα αρχεία XML. Για XML τεκμηριώνει την απόδοση είναι αποδεκτός, αλλά όχι έτσι για τα μεγαλύτερα έγγραφα XML. Μερικά ταυτόχρονα αιτήματα με τα μεγάλα έγγραφα XML μπορούν να αναγκάσουν τη μηχανή ΣΑΠΟΥΝΙΩΝ για να συντρίψουν.
Το Axis2 αποφεύγει το να κρατάει το πλήρες μήνυμα SOAP στη μνήμη με την εισαγωγή ενός νέου προτύπου αντικειμένου για την αναπαράσταση των SOAP μηνυμάτων, το AXIOM. Το AXIOM (ή OM για ευκολία) υιοθετεί μια εξολοκλήρου νέα μέθοδο. Αν και το AXIOM έχει μια "εξωτερική" ομοιότητα με το DOM, η διαφορά επίκειται στο ότι παράγει τα αντικείμενα μόνο σε περίπτωση ανάγκης. Αυτό το χαρακτηριστικό γνώρισμα "on-demand building (παραγωγή κατά απαίτηση)" δίνει στο AXIOM την αιχμή που απαιτείται για να υπερνικήσει το εμπόδιο μνήμης που οι πρόωρες μηχανές SOAP απέτυχαν να περάσουν.
Ένα ενδιαφέρον χαρακτηριστικό γνώρισμα του AXIOM είναι ότι είναι βασισμένο στην ανάλυση των XML εγγράφων με την μέθοδο του “pull parsing”. Είναι σε θέση να παράγει γεγονότα “pull” για την ανάλυση του μοντέλου αντικειμένου που χτίζεται . Εάν το μοντέλο αντικειμένου τυχαίνει να είναι ολοκληρωμένο κατά το ήμισυ, τότε AXIOM είναι σε θέση να παράγει γεγονότα “pull” για να το αναλύσει, κατευθείαν από τα δεδομένα που καταφθάνουν από το δίκτυο εκείνη τη στιγμή .
Παραδείγματος χάριν, ας υποθέσουμε ότι έχουμε ένα μήνυμα SOAP που έχει τρία στοιχεία στην επικεφαλίδα του μηνύματος SOAP και 100 στοιχεία παιδιά στο κυρίως “σώμα” του μηνύματος SOAP. Εάν ο χρήστης ζητήσει το τρίτο στοιχείο παιδί στην επικεφαλίδα του μηνύματος SOAP, το μοντέλο αντικειμένου (Object Model) θα χτιστεί μέχρι το τρίτο στοιχείο της επικεφαλίδας αφήνοντας το κυρίως “σώμα” του μηνύματος. Τώρα, αν ο χρήστης ζητήσει με τα γεγονότα “pull” τον φάκελο του μηνύματος SOAP (SOAP envelope-είναι όλο το μήνυμα SOAP στο Axis2) τότε θα παραχθούν γεγονότα “pull” (από το μοντέλο αντικειμένου) μόνο μέχρι το τρίτο στοιχείο της επικεφαλίδας που χτίστηκε από πριν και τα άλλα γεγονότα θα ληφθούν απευθείας από το επερχόμενο XML. Δεδομένου ότι στην επεξεργασία μηνυμάτων SOAP είναι συνηθισμένο να υποβληθούν σε επεξεργασία οι επικεφαλίδες των SOAP μηνυμάτων μέσω των κατάλληλων DOM (Document Object Model-μοντέλο αντικειμένου εγγράφων) και να επεξεργαστεί το κυρίως “σώμα” του SOAP μηνύματος μέσω των εισερχόμενων δεδομένων που φτάνουν εκείνη την στιγμή μέσω δικτύου, οι παραπάνω ικανότητες επιτρέπουν στο Axis2 να κάνει μια αξιοσημείωτη διαφορά στην απόδοση.
Η καρδιά του AXIOM είναι η ανάλυση του XML με την μέθοδο “pull”, δεδομένου ότι είναι το μόνο πρότυπο ανάλυσης που υποστηρίζει τη διακοπή της διαδικασίας ανάλυσης. Το AXIOM χρησιμοποιεί την εισερχόμενη ροή API για XML (StAX), που καθιστά εύκολο να χειριστεί και να χρησιμοποιήσει μόνο ένα μέρος της μνήμης που χρησιμοποιεί ένα συμβατικό μοντέλο αντικειμένου. Συνδυασμένο με την ταχύτητα του αναλυτή επερχόμενης κίνησης δεδομένων με την μέθοδο “pull”, το AXIOM ωθεί το Axis2 μπροστά από τους προκατόχους του από την άποψη της αποδοτικότητας και της ταχύτητας.

Εξελιγμένο μοντέλο ενσωμάτωσης των WSs (deployment method)

Στην προηγούμενη έκδοση Axis 1.x, η ενσωμάτωση (WS deployment) μιας WS απαιτούσε την ενημέρωση του αρχείου παραμέτρων και την τοποθέτηση των αρχείων class (τα αρχεία class είναι τα παραγόμενα αρχεία από την Java όταν συντάσσεται και εκτελείτε ένα πρόγραμμα περνώντας από τον συντακτικό και λογικό έλεγχο-compiling) στον κατάλογο που περιέχει τα αρχεία εκτέλεσης της Java και ένα πρόγραμμα μπορεί να τα φορτώσει κατά την διάρκεια εκτέλεσής του (class path) .Το πρώτο μπορεί να επιτευχθεί με την εκτέλεση ενός εργαλείου που ονομάζεται AdminClient, αλλά στη νεότερη έκδοση ήταν δραματικά πιο απλό. Ο απλούστερος τρόπος για να ενημερωθεί το class path ήταν με το να αντιγράψει ο χρήστης τα αρχεία class στη σωστή θέση και επανεκκινήσει το Axis . Αυτό το δυσκίνητο deployment model είχε κατηγορηθεί ως σημαντική δυσχέρεια στη δυνατότητα χρησιμοποίησης του Axis.

Το μοντέλο ενσωμάτωσης(deployment model) του Axis2 είναι βασισμένο σε ένα αρχειοθετημένο σύστημα αρχείων. Κάθε συστατικό των παραμέτρων του Axis2 αποθηκεύεται σε ένα αρχείο ή ένα αρχείο παραμέτρων, το οποίο είναι παρόμοιο με το μηχανισμό deployment του J2EE. Με αυτόν τον νέο μηχανισμό, τα δύο βήματα στο μοντέλο deployment του Axis 1.x υστερούν στο να δημοσιεύσουν (deploy) ένα απλό αρχείο WSs. Για να καταστήσει την αρχειακή κατανομή (archive distribution) περισσότερο χρησιμοποιήσιμη, το Axis2 εισάγει τον όρο “hot deployment”. Κατά την διαδικασία αυτή το Axis2 ανιχνεύει την παρουσία του αρχείου της WS στον κατάλογο τον οποίο χρησιμοποιεί σαν αποθήκη των WSs και φορτώνει τα καινούργια αρχεία WS αυτόματα. Αν και αυτό το χαρακτηριστικό γνώρισμα είναι απίθανο να χρησιμοποιηθεί σε ένα περιβάλλον παραγωγής, είναι ένα σπουδαίο χαρακτηριστικό για τους αρχαρίους.
Η πραγματική αξία του νέου τρόπου deployment βρίσκεται στο γεγονός ότι οι πληροφορίες για κάθε WS ομαδοποιούνται σε ένα ενιαίο αρχείο. Το Axis2 παρέχει επίσης την απομόνωση των εφαρμογών που είναι βασιζόμενο σε κάθε αρχείο. Τα αρχεία των WSs είναι φορητά ανάμεσα στους διαφορετικούς Axis2 containers και η φόρτωση των αρχείων class των WSs απλοποιείται. Τα αρχεία του Axis μπορούν να τοποθετηθούν χειροκίνητα στον κατάλογο αποθηκών ή να φορτωθούν μέσω της web εφαρμογής του Axis για ένα J2EE servlet container. Για να καταστήσει τα πράγματα ευκολότερα, το Axis2 παρέχει επίσης διάφορα εργαλεία GUI για να βοηθήσει τη δημιουργία των αρχείων WS.

Καλύτερη υποστήριξη των MEPS (Message Exchange Patterns)

Όταν παρουσιάστηκε το Axis1.x , ο κύριος σκοπός των WSs ήταν να προσομοιώσουν το RPC στο SOAP. Έτσι το Axis1.x λανσαρίστηκε με ενσωματωμένο μοντέλο αλληλεπίδρασης RPC, το οποίο απαιτεί από το σύστημα να απαντήσει σε κάθε αίτημα. Αυτή η συμπεριφορά προκαλεί προβλήματα στις αλληλεπιδράσεις με συστήματα που δεν είναι βασισμένα στο RPC όπως στην μονόδρομη διαμεταγωγή μηνυμάτων . Εντούτοις, το ενδιαφέρον έχει επικεντρωθεί τώρα στην διαμεταγωγή μηνυμάτων, που αναγκάζει τις σημερινές μηχανές SOAP να υποστηρίζουν τις αλληλεπιδράσεις τύπου μηνύματος ως φυσικό τους χαρακτηριστικό.
Οι αλληλεπιδράσεις τύπου μηνύματος καθορίζονται από τα πρότυπα ανταλλαγής μηνυμάτων (MEPs) Η προδιαγραφή WSDL2.0 καθορίζει οκτώ τύπους MEPs. Από αυτούς, οι τρεις ουσιαστικοί για την ανάπτυξη της βασικής λύσης όσον αφορά την διαμεταγωγή μηνυμάτων υποστηρίζονται άμεσα από το Axis2. Ο προγραμματιστής της WS θα πρέπει να πραγματοποιήσει κάποιες επιπλέον αλλαγές για να υποστηρίξει και τους υπόλοιπους.
Τα τρία βασικά πρότυπα MEPs είναι τα “in-only”, ”Robust-In” και “In-Out”.

Στην "In-Only" υπάρχει μόνο μήνυμα αίτησης SOAP .
Στην "Robust-In" υπάρχει επίσης μήνυμα αίτησης SOAP, και μήνυμα απάντησης SOAP αποστέλλεται μόνο όταν προκύψει κάποιο σφάλμα.
Στην "In-Out" πάντα υπάρχει μήνυμα αίτηση και απάντησης SOAP.

Αντίθετα από τους προγόνους του, το Axis2 χτίστηκε έχοντας υπόψη την μονόδρομη ανταλλαγή μηνυμάτων. Ο πυρήνας του Axis2 υποστηρίζει και τις αλληλεπιδράσεις τύπου RPC και τις αλληλεπιδράσεις τύπου μηνύματος βασισμένες στο μονόδρομο μοντέλο μηνύματος. Περαιτέρω, οι αλληλεπιδράσεις ρπθ-ύφους μπορούν να χτιστούν επάνω στο πρότυπο αλληλεπίδρασης μήνυμα-ύφους. Ως εκ τούτου το Axis2 παρέχει την υποστήριξη του μοντέλου RPC διατηρώντας την υποστήριξη του μοντέλου μηνύματος στον πυρήνα του, που καθιστά πιθανό να συνυπάρξει με άλλες βασισμένες στο RPC μηχανές SOAP, ιδιαίτερα στο Axis 1.x.

Μονόδρομο μοντέλο ανταλλαγής μηνυμάτων

Το μονόδρομο μοντέλο ανταλλαγής μηνυμάτων έχει σκοπό μόνο την παράδοση ενός ενιαίου μηνύματος. Το μήνυμα αποστέλλεται και ξεχνιέται όσο το πλαίσιο του μοντέλου ανταλλαγής μηνυμάτων παραμένει ενεργό. Εάν υπάρχει οποιοσδήποτε συσχετισμός μεταξύ των μηνυμάτων, αφήνεται στα ανώτερα στρώματα του πλαισίου για να τον χειριστεί.

Σύγχρονος και ασύγχρονος προσανατολισμός

Η διανεμημένη πληροφοριακή επιστήμη υπολογιστών έχει εστιάσει στο RPC αρκετό καιρό. Κατά συνέπεια, η ιδέα ενός μηνύματος απάντησης (response) ακολουθούμενη αμέσως από ένα μήνυμα αίτησης (request), απασχολεί πάρα πολύ τους προγραμματιστές των WSs . Στα σενάρια του πραγματικού κόσμου, εντούτοις, το μήνυμα απάντησης μπορεί να μην εμφανιστεί καθόλου ή να μην εμφανιστεί μετά από κάποιο χρονικό διάστημα (που μπορεί να είναι είτε ένα λεπτό είτε ένα έτος). Ανάλογα με το χρόνο που λαμβάνεται για να αποκριθεί το σύστημα, η απάντηση μπορεί να σταλεί μέσω της σύνδεσης εισερχομένων ή μιας διαφορετικής σύνδεσης.
Στο RPC, η μέθοδος κλήσης της WS (invocation method) δεσμεύει τη γραμμή και περιμένει την απάντηση να φθάσει. Η μέθοδος απομακρυσμένης κλήσης αποδεσμεύει τη γραμμή μόνο όταν φθάσει η απάντηση. Όπως αναφέρθηκε προηγουμένως, τα σενάρια του πραγματικού κόσμου, είναι πολύ πιο σύνθετα και το μοντέλο ανάπτυξης της εφαρμογής πελάτη πρέπει να είναι αρκετά πλούσιο για να τα συμπεριλάβει όλα. Όσον αφορά στους χρήστες των WSs, το τι περιμένουν από μία ανταλλαγή μηνυμάτων είναι τα ακόλουθα.

Ο χρήστης μπορεί να μην αναμείνει μια απάντηση , έτσι η μέθοδος που κάλεσε την WS πρέπει απλά να επιστρέψει στην κανονική της λειτουργία.
Ο χρήστης μπορεί να θελήσει να αποστείλει την αίτηση και η μέθοδος που κάλεσε την απομακρυσμένη WS να περιμένει μέχρι να έρθει η απάντηση. Αυτή είναι μια συνηθισμένη λειτουργία RPC.
Ο χρήστης μπορεί να θελήσει η μέθοδος που αποστέλλει την αίτηση να επιστρέψει αμέσως στην κανονική της λειτουργία, και να αφήσει στο ενδιάμεσο υλικο-λογισμικό-middleware (Axis2) να αναλάβει την διαμεταγωγή των μηνυμάτων και να τον αφήσει να χειριστεί το μήνυμα απάντησης όταν αυτό φτάσει. Τα τελευταία μπορούν να γίνουν με τις κλήσεις με επιστροφή (callbacks), γεγονότα που πληροφορούν το χρήστη όταν ένα μήνυμα απάντησης φτάσει, ή από έναν μηχανισμό καταμετρήσεις όπου ο χρήστης ελέγχει περιοδικά για την παρουσία μηνύματος απάντησης.
Δυστυχώς, η εικόνα περιπλέκεται περισσότερο από την πολυπλοκότητα που προκύπτει από τους διαφορετικούς τρόπους μεταφοράς. Υπάρχουν πρώτιστα δύο τρόποι μεταφοράς, ο μονόδρομος και ο αμφίδρομος. Η χρήση των αμφίδρομων όπως το HTTP μειώνετε με το ενδιαφέρων να μετατοπίζεται στα μονόδρομα πρωτόκολλα μεταφορών όπως το SMTP. Οι μελλοντικές WSs είναι πιθανό να είναι ασύγχρονης φύσης και το HTTP μπορεί να σταματήσει να είναι η πρώτιστη επιλογή.
Τα σύγχρονα και ασύγχρονα μοντέλα προγραμματισμού προορίζονται να καλύψουν όλα τα

παραπάνω σενάρια. Αν και η οικοδόμηση μιας μηχανής SOAP που υποστηρίζει όλες αυτές τις πτυχές της διαδικασίας ανταλλαγής μηνυμάτων δεν είναι εύκολη, το Axis2 αναπτύσσεται σε αυτό το περιβάλλον. Δεδομένου ότι το Axis2 είναι βασισμένο σε έναν ουδέτερο πυρήνα, δεν γνωρίζει τίποτα για τις μορφές αλληλεπίδρασης μηνυμάτων και επίσης δεν είναι ανάγκη να γνωρίζει τη συμπεριφορά της μεταφοράς. Το Axis2 επιτρέπει στους χρήστες να προγραμματίσουν WSs σε οποιαδήποτε από τις παραπάνω μορφές.

Εξελιγμένοι διαχειριστές (Handlers)

Το Axis εισήγαγε την έννοια των χειριστών (Handlers) που παρέχουν τις επεκτάσεις ή το φιλτράρισμα στη Web Service stack . Οι χειριστές είναι διαμορφώσιμες κλήσεις με επιστροφή (Callbacks) που μπορούν να καταχωρηθούν πριν σπαταλήσουν ή στείλουν το μήνυμα SOAP. Στο πλαίσιο χειριστών του Axis 1.x, η διάταξη των χειριστών μπορεί να διευκρινιστεί μόνο την στιγμή που γίνετε το deploy της WS. Αυτό απαιτεί να γνωρίζουμε την ακριβή διάταξη των χειριστών την ώρα του deploy, αφήνοντας με αυτόν τον τρόπο την διαδικασία απόφασης του πως θα είναι η διάταξη των χειριστών στον προγραμματιστή της web service. Εντούτοις, οι περισσότεροι από τους χειριστές (π.χ. χειριστής ασφάλειας, αξιόπιστης ανταλλαγής μηνυμάτων και διαμεταγωγής) αναπτύσσονται από ξεχωριστά προγράμματα Web Service Stack. Δυστυχώς, οι υπεύθυνοι για την ανάπτυξη web services συχνά δεν καταλαβαίνουν τις σύνθετες απαιτήσεις που προσδιορίζουν την διάταξη των διαχειριστών.
Το Axis2 σκοπεύει να το αλλάξει αυτό με την εισαγωγή των "κανόνων φάσης" που επιτρέπουν στους προγραμματιστές ο χειριστής να διευκρινίζει τη "μερική διάταξη" των χειριστών. Αυτό επιτυγχάνεται με την εισαγωγή δύο εννοιών, των "φάσεων" και των "κανόνων φάσης". Ενώ οι φάσεις είναι ετικέτες που δίνονται σε συγκεκριμένο τομέα της εκτέλεσης της διάταξης χειριστών, οι κανόνες φάσης διευκρινίζουν τη θέση κάθε χειριστή σχετικά με συγκεκριμένες φάσεις. Όταν η εκτέλεση πραγματοποιείται, η μηχανή του Axis2 θα εξασφαλίσει ότι ο χειριστής τοποθετείται κατά τέτοιο τρόπο ώστε να ικανοποιούνται όλοι οι κανόνες φάσης.
Παραδείγματος χάριν, ας υποθέσουμε ότι το αξιόπιστο πρόγραμμα ανταλλαγής μηνυμάτων – RM(Reliable Messaging) παρήγαγε έναν χειριστή RM. Δεδομένου ότι αυτός ο χειριστής ότι θα εκτελεστεί με άλλους χειριστές και πρόκειται να χρησιμοποιηθεί σε πολλές διαφορετικές καταστάσεις, δεν υπάρχει κανένας τρόπος οι προγραμματιστές του χειριστή RM να μπορούν να διευκρινίσουν την απόλυτη θέση από την οποία ο χειριστής RM πρέπει να τρέξει. Αυτός ο στόχος μετατοπίζεται έπειτα στον υπεύθυνο για την ανάπτυξη των WSs, ο οποίος δεν χρειάζεται να γνωρίζει για τη συμπεριφορά του χειριστή RM.
Ακόμα κι έτσι, ο προγραμματιστής γνωρίζει μερικούς από τους κανόνες για το πώς ο χειριστής RM πρέπει να εκτελεστεί. Παραδείγματος χάριν, ένας απλός κανόνας που προσδιορίζει το χειριστή RM είναι ότι πρέπει να τρέξει μετά από το χειριστή ασφάλειας αλλά πριν τον χειριστή συναλλαγής. Ο χειριστής RM θα τεκμηριώσει αυτό το γεγονός για να βοηθήσει τον προγραμματιστή της WS. Οι κανόνες φάσης πηγαίνουν ένα βήμα περαιτέρω με το να αφήσουν τον υπεύθυνο για την ανάπτυξη χειριστών RM, να καθοδηγήσει άμεσα το Axis για την τοποθέτηση των χειριστών, παρά από το να τεκμηριώσουν απλά το γεγονός, για να εξασφαλίσουν την ομαλή λειτουργία.

Ενσωματώσιμη δέσμευση δεδομένων (Pluggable Data Binding)

Η δέσμευση των XML δεδομένων ή η μετατροπή XML στην Java και της Java σε XML είναι ένα γνωστό πρόβλημα. Έχει λυθεί επιτυχώς με τη χρήση του JAXB, του Castor και των XMLBeans. Το καλύτερο εργαλείο δέσμευσης δεδομένων για μια δεδομένη περίπτωση χρήσης αποφασίζεται από τη φύση των προτιμήσεων των περιπτώσεων χρήσης και των προγραμματιστών.

Η λύση για τις δεσμεύσεις δεδομένων του Axis2 είναι να παρασχεθεί ένα πλαίσιο που επιτρέπει την χρήση διαφορετικών εργαλείων δέσμευσης δεδομένων. Η ομάδα ανάπτυξης του Axis2 θεωρεί ότι αυτό θα παρήγε την πιο εύκαμπτη λύση, καθώς επίσης και να βοηθήσει στον προσανατολισμό στις Web Service Stacks παρά στην απλή και κοινή ιδέα της δέσμευσης δεδομένων.

Αρχιτεκτονική του Apache Axis2

Η αρχιτεκτονική του Axis2 εκθέτει μερικές αρχές για να διατηρήσει την ομοιομορφία. Είναι οι ακόλουθοι:
Η αρχιτεκτονική του Axis2 χωρίζει τη λογική και τις καταστάσεις. Ο κώδικας που κάνει την επεξεργασία είναι χωρίς σταθερή κατάσταση στο Axis2. Αυτό επιτρέπει στον κώδικα να εκτελεσθεί ελεύθερα από τα νήματα που εκτελούνται παράλληλα.
Όλες οι πληροφορίες κρατούνται σε ένα μοντέλο πληροφοριών, επιτρέποντας στο σύστημα να σταματήσει να λειτουργεί και να επανεκκινήσει.

Η αρχιτεκτονική του Axis2 είναι προσαρμόσιμη. Επομένως ο σκελετός του Axis2 είναι χτισμένος από ουσιώδη διαμορφώματα (modules) που αποτελούν συλλογικά τον πυρήνα της αρχιτεκτονικής του Axis2, και από μη θεμελιώδη διαμορφώματα που είναι τοποθετημένα σε μορφή επιπλέον στρώματος πάνω από αυτό τον πυρήνα μορφομάτων/αρσιτεκτονικής.

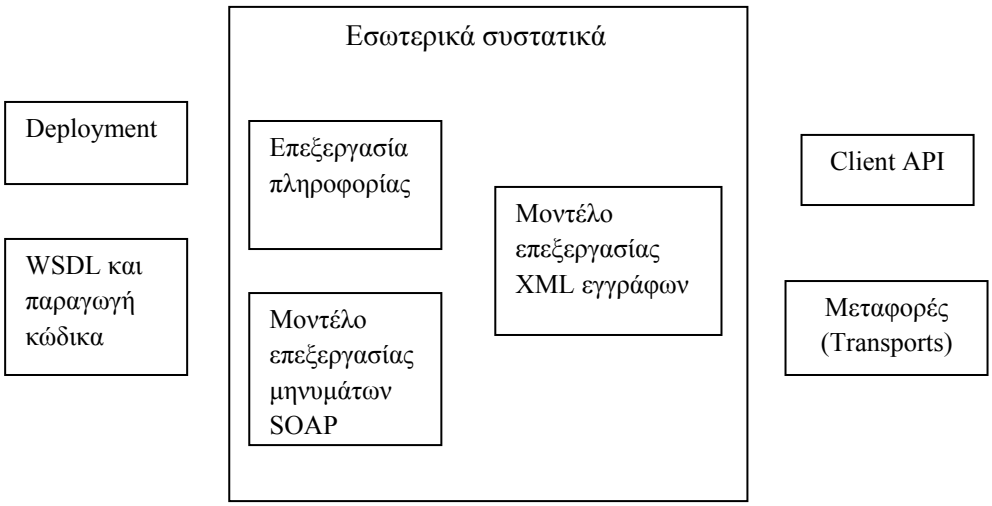
Core Modules:

- Μοντέλο πληροφοριών: Το Axis2 καθορίζει ένα μοντέλο για να χειριστεί τις πληροφορίες και όλες τις καταστάσεις που κρατιούνται σε αυτό το μοντέλο. Το μοντέλο έχει μια ιεραρχία για τις πληροφορίες. Το σύστημα διαχειρίζεται τον κύκλο ζωής των αντικειμένων σε αυτήν την ιεραρχία.
- Πρότυπο επεξεργασίας XML: Το να χειριστούν τα μηνύματα SOAP είναι ο σημαντικότερος και πιο σύνθετος στόχος. Η αποδοτικότητα αυτής της διαδικασίας είναι ο σημαντικότερος παράγοντας που αποφασίζει την απόδοση. Είναι λογικό να ανατεθεί αυτή η διαδικασία σε ένα χωριστό module(AXIOM) που επιτρέποντάς του να παρέχει ένα απλό API για το SOAP και το XML κρύβοντας δυσκολίες της αποδοτικής επεξεργασίας του XML μέσα στην εφαρμογή.
- Μοντέλο επεξεργασίας SOAP: αυτό ελέγχει την εκτέλεση της επεξεργασίας. Το μοντέλο αυτό καθορίζει τις διαφορετικές φάσεις από τις οποίες θα περάσει η εκτέλεση, και ο χρήστης μπορεί να επεκτείνει το μοντέλο επεξεργασίας σε μερικές συγκεκριμένες φάσεις.
- Μοντέλο δημοσιοποίησης: το πρότυπο δημοσιοποίησης του Axis2 επιτρέπει στο χρήστη για να δημοσιεύσει τις υπηρεσίες, να διαμορφώσει τις μεταφορές, να επεκτείνει το μοντέλο επεξεργασίας SOAP ανά σύστημα, βάση υπηρεσίας η λειτουργίας

- Το API του πελάτη: αυτό παρέχει ένα κατάλληλο API για τους χρήστες για να επικοινωνήσουν με τις WSs χρησιμοποιώντας το Axis2. Υπάρχει ένα σύνολο από κλάσεις για αλληλεπιδράσει ο χρήστης με IN-OUT και IN-ONLY MEPS όπου μπορούν να χρησιμοποιηθούν για να κατασκευάσουν οποιοδήποτε άλλο διαφορετικό MEP.
- Μεταφορές: Το Axis2 καθορίζει ένα μοντέλο μεταφορών που επιτρέπει στο χρήστη να χρησιμοποιήσει διαφορετικές μεταφορές. Οι μεταφορές ταιριάζουν σε συγκεκριμένες φάσεις του προτύπου επεξεργασίας SOAP. Η εγκατάσταση παρέχει μερικές κοινές μεταφορές και ο χρήστης μπορεί να γράψει νέες εάν και όταν απαιτείται.

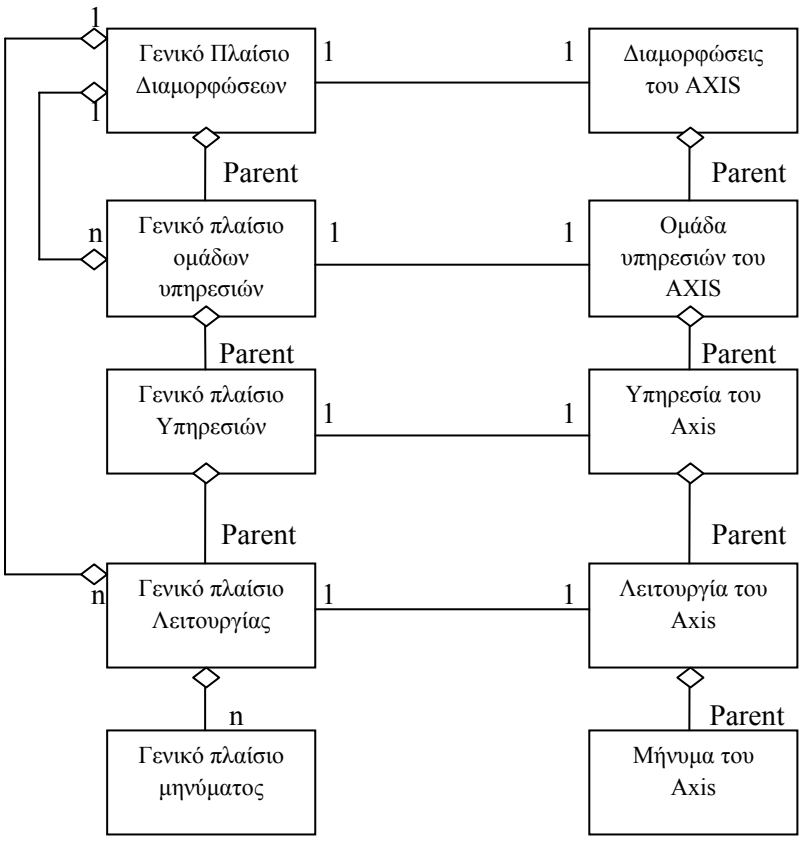
Non-core Modules:

- Αυτόματη παραγωγή κώδικα: Το Axis2 παρέχει ένα εργαλείο παραγωγής κώδικα που θα παραγάγει τον server side και client side κώδικα μαζί με μια δοκιμαστική περίπτωση. Ο παραγμένος κώδικας θα απλοποιούσε το deploy των υπηρεσιών το απομακρυσμένο κάλεσμα των υπηρεσιών. Αυτό θα αύξανε τη δυνατότητα χρησιμοποίησης του Axis2.
- Δέσμευση δεδομένων: Το βασικό Client API του Axis2 αφήνει τους χρήστες να επεξεργαστούν το SOAP στο επίπεδο ορισμού των πληροφοριών όπου η δέσμευση δεδομένων το επεκτείνει για να το καταστήσει καταλληλότερο στους χρήστες με την ομαδοποίηση του στρώματος ορισμού πληροφοριών και παρέχοντας μία γλώσσα προγραμματισμού συγκεκριμένου περιβάλλοντος.



Μοντέλο πληροφοριών

Το μοντέλο πληροφοριών έχει δύο ιεραρχίες, τα Γενικά Πλαίσια και τις Περιγραφές. Το μοντέλο αυτό περιγράφεται καλύτερα από το παρακάτω UML γράφημα.



Οι δύο ιεραρχίες συνδέονται όπως φαίνεται στο παραπάνω διάγραμμα. Η ιεραρχία περιγραφής αντιπροσωπεύει τα στατικά δεδομένα. Αυτά τα δεδομένα μπορούν να φορτωθούν από ένα αρχείο διαμόρφωσης που υπάρχει σε όλη τη διάρκεια ζωής του Axis2. Παραδείγματος χάριν, δημοσιευμένες WSs, λειτουργίες, κλπ. Αφ' ετέρου, η ιεραρχία γενικού πλαισίου φυλάσσει τις δυναμικότερες πληροφορίες για τα πράγματα που έχουν περισσότερα από ένα αντίγραφα κάθε φορά (π.χ. Γενικό πλαίσιο μηνύματος).

Αυτές οι δύο ιεραρχίες δημιουργούν ένα πρότυπο που παρέχει τη δυνατότητα να ψάξει για τα βασικά ζευγάρια τιμών. Όταν οι τιμές αναζητούνται σε ένα δεδομένο επίπεδο, αναζητούνται ανεβαίνοντας στην ιεραρχία έως ότου βρεθεί μια αντιστοιχία. Στο προκύπτον μοντέλο, τα χαμηλότερα επίπεδα αγνοούν τις τιμές των ανώτερων επιπέδων. Παραδείγματος χάριν, όταν μια τιμή αναζητείται στο γενικό πλαίσιο μηνύματος και δεν βρίσκεται , θα αναζητηθεί στο γενικό πλαίσιο λειτουργίας κλπ, ανεβαίνοντας στην ιεραρχία. Η αναζήτηση γίνεται αρχικά ανεβαίνοντας στην ιεραρχία, και εάν η αφετηρία είναι ένα γενικό πλαίσιο έπειτα αυτό αναζητεί στην ιεραρχία περιγραφής.

Αυτό επιτρέπει στο χρήστη να δηλώσει και να αγνοήσει τιμές. Αποτέλεσμα είναι ένα πολύ ευέλικτο μοντέλο διαμόρφωσης. Η ευελιξία θα μπορούσε να είναι αχίλλειος πτέρνα για το σύστημα δεδομένου ότι η αναζήτηση είναι ακριβή, ειδικά για κάτι που δεν υπάρχει. Ακόμα στην τελική ανάλυση οι προγραμματιστές θεωρούν ότι η ευελιξία θα εξυπηρετούσε καλύτερα αυτήν την στιγμή.

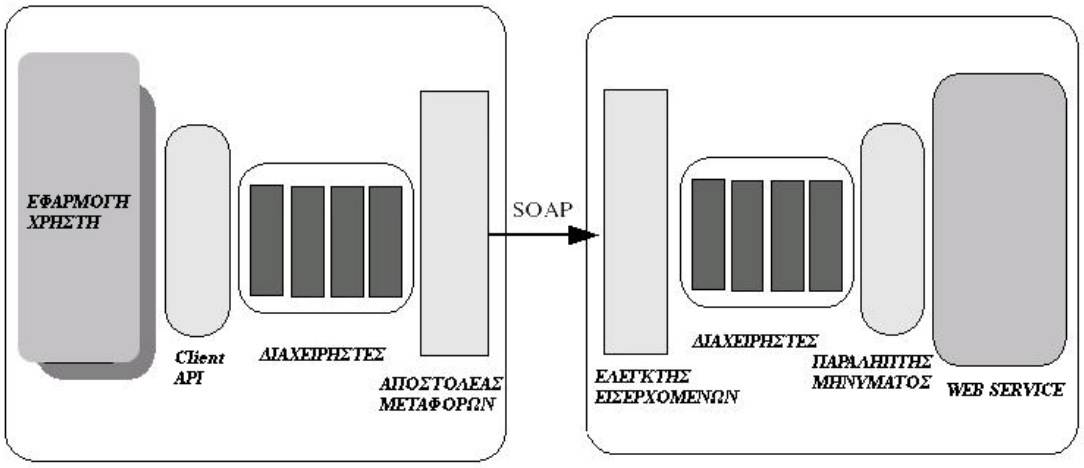
Γενικό Πλαίσιο διαμόρφωσης	Κρατά την κατάσταση κατά την διάρκεια εκτέλεσης. Ένα καλό αντίγραφο αυτού	Διαμορφώσεις του Axis (Axis	Κρατά όλες τις διαμορφώσεις που αφορούν την λειτουργία του

(Configuration Context)	θα έκανε ουσιαστικά ένα αντίγραφο του Axis2.	Configuration)	Axis. Μεταφορές, modules, παράμετροι και υπηρεσίες κλπ.
Γενικό πλαίσιο ομάδων υπηρεσιών (ServiceGroup Context)	Φυλάσσει τις πληροφορίες για μια συγκεκριμένη χρήση της αντίστοιχης ομάδας υπηρεσιών. Η ζωή ενός πλαισίου ομάδας υπηρεσιών αρχίζει όταν ο χρήστης αρχίζει να αλληλεπιδρά με μια υπηρεσία που ανήκει σε αυτή την ομάδα. Αυτό μπορεί να χρησιμοποιηθεί για να διαμοιράζει πληροφορίες μεταξύ των υπηρεσιών (μέσα στην ίδια ομάδα υπηρεσιών) σε μια ενιαία αλληλεπίδραση.	Ομάδα υπηρεσιών του Axis (AxisServiceGroup)	Κρατά πληροφορίες για την ώρα ενσωμάτωσηςς σχετικά με μία συγκεκριμένη ομάδα υπηρεσιών.
Γενικό πλαίσιο Υπηρεσιών (Service Context)	Αυτό το γενικό πλαίσιο είναι διαθέσιμο καθ' όλη τη διάρκεια χρήσης της αντίστοιχης υπηρεσίας. Αυτό μπορεί να χρησιμοποιηθεί για να διαμοιράσει πληροφορίες μεταξύ διάφορων MEPs της ίδιας υπηρεσίας, μέσα σε μια ενιαία αλληλεπίδραση.	Υπηρεσία του Axis (AxisService)	Κρατά της λειτουργίες και της διαμορφώσεις του επιπέδου των υπηρεσιών.
Γενικό πλαίσιο λειτουργίας (Operation Context)	Κρατά τις πληροφορίες για το τρέχων αντίγραφο MEP, διατηρεί τα μηνύματα στον τρέχων MEP κ.λπ.	Λειτουργία του Axis (AxisOperation)	Κρατά τις διαμορφώσεις του επιπέδου λειτουργίας.
Γενικό πλαίσιο μηνύματος (Message Context)	Κρατά όλες τις πληροφορίες για το μήνυμα που εκτελείται την συγκεκριμένη χρονική στιγμή.	Μήνυμα του Axis (AxisMessage)	Δεν κρατά κάποιες πληροφορίες μέχρι τώρα, αλλά μπορεί να χρησιμοποιηθεί ως σημείο για μελλοντική επέκταση.

Μοντέλο επεξεργασίας XML

Το μοντέλο επεξεργασία XML βασίζεται στο AXIOM το οποίο είναι ένα εξολοκλήρου ξεχωριστό κεφάλαιο του οποίου η ανάλυση δεν θα μας απασχολήσει στην συγκεκριμένη αναφορά. Το μόνο που χρειάζεται να γνωρίζουμε είναι ότι κάθε μήνυμα στο Axis2 είναι ένα XML το οποίο για να επεξεργαστεί πρέπει να αναλυθεί με διάφορους μεθόδους που μας προσφέρει το AXIOM.

Το μοντέλο επεξεργασίας του SOAP



Η αρχιτεκτονική προσδιορίζει δύο βασικές ενέργειες που ένας επεξεργαστής SOAP πρέπει να εκτελέσει, στέλνοντας και λαμβάνοντας τα μηνύματα SOAP. Η αρχιτεκτονική παρέχει δύο σωλήνες ("ροών), για να εκτελέσει αυτές τις δύο βασικές ενέργειες. Η μηχανή του Axis ή ο οδηγός του Axis2 καθορίζει δύο μεθόδους τις send() και receive() για να εφαρμόσει αυτούς τους δύο σωλήνες "ροής". Οι δύο σωλήνες ονομάζονται *in-pipe* και *out-pipe*, και τα σύνθετα MEPs κατασκευάζονται συνδυάζοντας αυτούς τους δύο σωλήνες *ροής*.
Η επεκτασιμότητα του μοντέλου επεξεργασίας SOAP παρέχεται μέσω των χειριστών (handlers). Όταν ένα μήνυμα SOAP υποβάλλεται σε επεξεργασία οι χειριστές που έχουν δηλωθεί θα εκτελεστούν. Οι χειριστές μπορούν να δηλωθούν ως global, service, ή operation πεδία και η τελική αλυσίδα χειριστών υπολογίζεται συνδυάζοντας τους χειριστές από όλα τα πεδία.
Οι χειριστές ενεργούν ως αναχαιτιστές τις λειτουργίας και επεξεργάζονται τα μέρη του μηνύματος SOAP και παρέχουν πρόσθετες υπηρεσίες. Συνήθως οι χειριστές εργάζονται στις επικεφαλίδες των μηνυμάτων SOAP αλλά μπορούν να έχουν πρόσβαση ή να αλλάξουν και το κυρίως μέρος ενός μηνύματος SOAP(SOAP Body).
Όταν ένα μήνυμα SOAP στέλνεται μέσω του client API, ένα *out-pipe* θα ξεκινήσει, το *out-pipe* καλεί τους χειριστές και τελειώνει με τον αποστολέα μεταφορών (Transport Sender) που στέλνει το μήνυμα SOAP στο τελικό παραλήπτη. Το μήνυμα SOAP παραλαμβάνεται από έναν δέκτη μεταφορών στον τελικό παραλλήπτη, το οποίο διαβάζει το μήνυμα SOAP και αρχίζει ένα *in-pipe*. Ο *in-pipe* αποτελείται από τους χειριστές και τελειώνει με τον δέκτη μηνυμάτων, ο οποίος παραλαμβάνει και αναλύει το μήνυμα SOAP.

Η επεξεργασία που περιγράφτηκε παραπάνω συμβαίνει για κάθε μήνυμα SOAP που συναλλάσσεται. Μετά από την επεξεργασία ενός μηνύματος το Axis2 μπορεί να αποφασίσει να δημιουργήσει άλλα μηνύματα SOAP, οπότε σ' αυτή την περίπτωση προκύπτουν τα πιο σύνθετα σχέδια μηνυμάτων. Εντούτοις το Axis2 βλέπει πάντα το μήνυμα SOAP από την άποψη της επεξεργασίας ενός ενιαίου μηνύματος. Η ένωση των μηνυμάτων τοποθετείτε ως επίπεδο πάνω από αυτό του βασικού σκιλετού.
Οι δύο σωλήνες ροής δεν διαφοροποιούνται μεταξύ του server και του client. Το μοντέλο επεξεργασίας SOAP χειρίζεται την πολυπλοκότητα και παρέχει δύο σωλήνες ροής στο χρήστη. Οι διαφορετικοί τομείς ή τα στάδια των σωλήνων ροής είναι ονόματα, και σύμφωνα με το ιδίωμα του Axis2 ονομάζονται "φάσεις". Ένας χειριστής τρέχει πάντα μέσα σε μια φάση, και η φάση παρέχει έναν μηχανισμό για να διευκρινίσει τη διάταξη των χειριστών. Και οι δύο σωλήνες ροής έχουν χτιστεί σε φάσεις, και οι δύο καθορίζουν τις περιοχές για τις "φάσεις χρηστών" που μπορούν να καθοριστούν από το χρήστη.

Παρακάτω παρουσιάζονται δύο σωλήνες *ροής* με τις προκαθορισμένες φάσεις τους και τις καθορισμένες από τον χρήστη φάσεις. Οι καθορισμένες από τον χρήστη φάσεις ταιριάζουν στις φάσεις χρηστών.

Εισερχόμενη ροή(In flow)

Μεταφορά	Προ-μεταφοράς	Μεταφορά	Μετα-μεταφοράς	Φάσεις χρηστών	Επικύρωση Μηνύματος	Επεξεργασία Μηνύματος

Εξερχόμενη ροή(Out Flow)

Αρχικοποίηση Μηνύματος	Φάσεις Χρηστών	Μεταφορά

Προεπιλεγμένο μοντέλο επεξεργασίας του Axis2

Το Axis2 έχει μερικούς ενσωματωμένους χειριστές οι οποίοι λειτουργούν σε φάσεις, εσωτερικά, και δημιουργούν τις προεπιλεγμένες διαμορφώσεις του Axis2.

1. Υπάρχουν τέσσερις ειδικοί χειριστές, ορισμένοι στο Axis2.
   Dispatchers (Υπεύθυνοι μεταφοράς): Βρίσκει την υπηρεσία και τη λειτουργία που το μήνυμα SOAP απευθύνεται. Οι αποστολείς τρέχουν πάντα στον *in*-pipe και μέσα στη φάση μεταφοράς. Η ενσωματωμένη μεταφορά των διαχειριστών μεταφοράς σε μια συγκεκριμένη λειτουργία ανάλογα με διάφορους όρους όπως οι πληροφορίες του WS-Addressing, οι πληροφορίες των URI, οι πληροφορίες της λειτουργίας του SOAP κλπ.
2. Δέκτης μηνυμάτων (Message Receiver): Παραλαμβάνει το μήνυμα SOAP και το παραδίδει στην εφαρμογή. Ο δέκτης μηνυμάτων είναι ο τελευταίος διαχειριστής του *in-pipe.*
3. Αποστολέας μεταφορών (Transport Sender): Αποστέλλει το μήνυμα SOAP στο τελικό σημείο (endpoint) του SOAP, για το οποίο προορίζεται το μήνυμα. Πάντα λειτουργεί ως ο τελευταίος διαχειριστής στον *out-pipe*.

Επεξεργασία ενός εισερχόμενου μηνύματος SOAP

Το εισερχόμενο μήνυμα SOAP παραλαμβάνεται πάντα από έναν δέκτη μεταφορών (Transport Receiver) ο οποίος περιμένει τα μηνύματα SOAP. Μόλις φθάσει το μήνυμα SOAP οι επικεφαλίδες της μεταφοράς αναλύονται και δημιουργείται ένα γενικό πλαίσιο μηνύματος (Message Context) για το εισερχόμενο μήνυμα SOAP. Κατόπιν ο *in-pipe* εκτελείται με το γενικό πλαίσιο μηνύματος.

Παρακάτω περιγράφεται τι συμβαίνει σε κάθε φάση της εκτέλεσης. Αυτή η διαδικασία μπορεί να συμβεί είτε στον server είτε στον client.

1. Φάση μεταφορών (Transport Phase): Οι χειριστές σε αυτή τη φάση πρέπει να επεξεργαστούν τις πληροφορίες μεταφοράς όπως είναι η επικύρωση του εισερχομένου μηνύματος με το να εξετάζουν διάφορες επικεφαλίδες μεταφοράς, η πρόσθεση δεδομένων στο γενικό πλαίσιο μηνύματος κλπ.
2. Φάση προ-μεταφοράς (Pre-dispatch phase): Η κύρια λειτουργία των χειριστών σε αυτήν την φάση είναι να διαβάσουν το γενικό πλαίσιο μηνυμάτων προκειμένου να

πραγματοποιηθεί η μεταφορά. Για παράδειγμα η επεξεργασία των επικεφαλίδων των διευθύνσεων συμβαίνει σε αυτήν την φάση, έτσι με την τους θα βρεθεί το όνομα της υπηρεσίας και της λειτουργίας.

3. Φάση μεταφοράς (Dispatch Phase): Οι Μεταφορείς (Dispatchers) λειτουργούν σε αυτήν την φάση και βρίσκουν την υπηρεσία εάν η υπηρεσία δεν έχει βρεθεί ήδη. Η προϋπόθεση της φάσης μεταφοράς λειτουργεί ως εξής: ελέγχτει εάν η υπηρεσία και η λειτουργία έχουν βρεθεί ή όχι. Εάν η υπηρεσία ή η λειτουργία δεν έχει βρεθεί μέχρι στιγμής εκτέλεση θα σταματήσει και θα στείλει ένα μήνυμα λάθους "service not found error".
4. Καθορισμένες από τον χρήστη φάσεις : Οι χρήστες εδώ έχουν την άδεια να χρησιμοποιήσουν τους προσωπικούς τους διαχειριστές.
5. Φάση επικύρωσης μηνυμάτων: Μόλις ολοκληρωθεί η εκτέλεση του επιπέδου χρηστών, αυτή η φάση επικυρώνει εάν η επεξεργασία μηνυμάτων SOAP έχει πραγματοποιηθεί σωστά.
6. Φάση επεξεργασίας μηνυμάτων: Εδώ εκτελείται η επιχειρησιακή λογική του μηνύματος SOAP. Ένας δέκτης μηνυμάτων καταχωρείται με κάθε λειτουργία. Αυτός ο δέκτης μηνυμάτων θα εκτελεστεί ως ο τελευταίος χειριστής αυτής της φάσης.

Μπορούν να υπάρξουν άλλοι χειριστές σε οποιεσδήποτε από αυτές τις φάσεις. Οι χρήστες μπορούν να χρησιμοποιήσουν τους προσωπικού τους χειριστές για να αγνοήσουν τους προεπιλεγμένους σε κάθε μια από αυτές τις φάσεις.

Επεξεργασία του εξερχόμενου μηνύματος

Ο *out-pipe* είναι απλούστερος επειδή η υπηρεσία και η λειτουργία που θα μεταφερθούν είναι γνωστές από την ώρα που ξεκινάει η εκτέλεση του. Ο *out-pipe* μπορεί να αρχίσει από τον δέκτης μηνυμάτων (Message Receiver) ή από το Client API. Οι φάσεις του out-pipe είναι:

1. Φάση αρχικοποίησης μηνύματος: Η πρώτη φάση του *out-pipe* . Χρησιμεύει ως υποδοχέας για τους προσωπικούς χειριστές του χρήστη.
2. Φάσεις χρηστών: Αυτή η φάση εκτελεί τους χειριστές στις προκαθορισμένες από τον χρήστη φάσεις.
3. Φάση μεταφοράς: Εκτελεί όσους χειριστές μεταφορών λαμβάνονται από τη δηλωμένη διαμόρφωση μεταφοράς. Ο τελευταίος χειριστής θα είναι ο αποστολέας μεταφορών που θα στείλει το μήνυμα SOAP στον τελικό αποδέκτη.

Ενσωμάτωση Υπηρεσίας (Service Deployment)

Καταρχήν για να λειτουργήσει το Axis2 πρέπει το σύστημα στο οποίο θα εγκατασταθεί να διαθέτει κάποια έκδοση του Java SDK. Το Axis2 χρησιμοποίει τις βιβλιοθήκες της Java και προαπαιτεί να υπάρχει ήδη εγκατεστημένη στο σύστημα. Επίσης υπάρχει έκδοση του Axis2 για την C, στην αναφορά αυτή όμως δεν θα ασχοληθούμε με αυτήν.

Η ενσωμάτωση υπηρεσιών του Axis2 εισαγάγει την έννοια ενός J2EE-ομοειδούς μηχανισμού ενσωμάτωσης, όπου ο προγραμματιστής μπορεί να συσσωρεύσει όλα τα αρχεία class, τα αρχεία βιβλιοθηκών, τα αρχεία πηγής, και τα αρχεία διαμόρφωσης που διαθέτει μαζί ως ένα αρχείο συμπίεσης , και να τα τοποθετήσει σε μια διευκρινισμένη θέση στο σύστημα αρχείων. Το Axis2 υποστηρίζει δύο νέες τεχνολογίες ενσωμάτωσης, το ***hot deployment*** και το ***hot update*** έννοιες που δεν είναι μια νέα ορολογία στα τεχνικά πρότυπα, ιδιαίτερα σε μια πλατφόρμα web services, αλλά για το Apache Axis, είναι ένα νέο χαρακτηριστικό.

Hot Deployment: Η ικανότητα ενσωμάτωσης της υπηρεσίας ενώ το σύστημα είναι σε λειτουργία. Σε ένα σύστημα ή επιχείρηση πραγματικού χρόνου, η διαθεσιμότητα του συστήματος είναι πολύ σημαντική. Εάν το σύστημα σταματήσει να λειτουργεί ακόμα και ελάχιστο χρόνο, η απώλεια είναι σημαντική και μπορεί να έχει επιπτώσεις στη διάρκεια ζωής της επιχείρησης. Εντούτοις, εάν είναι απαραίτητο να προστεθεί μια νέα υπηρεσία στο σύστημα και εάν αυτό μπορεί να γίνει χωρίς διακοπή των κεντρικών υπολογιστών, αυτό θα ήταν ένα μεγάλο επίτευγμα. Έτσι, το Axis2 στράφηκε προς αυτή την λύση και παρέχει την δυνατότητα *hot deployment*, όπου δεν χρειάζεστε η διακοπή της λειτουργίας του συστήματος ώστε να ενσωματωθεί μια νέα WS σε αυτό. Πρέπει απλά να τοποθετηθεί το απαραίτητο αρχείο της WS στον κατάλογο υπηρεσιών της αποθήκης του Axis2. Κατόπιν, το μοντέλο ενσωμάτωσης θα ενσωματώσει αυτόματα την υπηρεσία και θα την καταστήσει διαθέσιμη.

Hot Update: Η ικανότητα να γίνουν αλλαγές σε μία ήδη ενσωματωμένη υπηρεσία χωρίς να διακοπεί η λειτουργία του συστήματος. Αυτό είναι ένα σημαντικό χαρακτηριστικό γνώρισμα και απαιτείται σε ένα περιβάλλον ελέγχου. Εντούτοις, δεν είναι ενδεδειγμένο να χρησιμοποιηθεί σε ένα σε σύστημα πραγματικού χρόνου επειδή μπορεί να οδηγήσει το σύστημα σε μία ανεπιθύμητη και άγνωστη κατάσταση. Επιπλέον, υπάρχει η πιθανότητα τα ήδη υπάρχοντα δεδομένα της υπηρεσίας που θέλουμε να αλλάξουμε να χαθούν. Για να αποτραπεί αυτό, το Axis2 διαθέτει στο κεντρικό αρχείο διαμορφώσεών του την επιλογή απενεργοποιημένη από την αρχή.
Το Axis2 πραγματοποιεί το *hot deployment* και το *hot update* με τη χρησιμοποίηση ενός χρονοπρογραμματιστή και ενός παρατηρητή συστημάτων αρχείων (file system listener), όπου ο χρονοπρογραμματιστής προγραμματίζεται με τέτοιο τρόπο ώστε να ενημερώνει τον παρατηρητή να ψάξει την αποθήκη του Axis2 (Axis2 repository) για ενημερώσεις και, ανάλογα με τις διαμορφώσεις που έχουν γίνει στο κεντρικό αρχείο διαμορφώσεων του Axis2 το axis2.xml, η υπηρεσία θα τεθεί στην διάθεση του συστήματος.

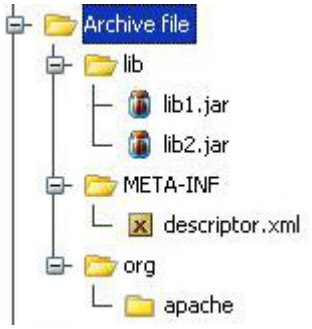
Μηχανισμός Ενσωμάτωσης

Όπως αναφέρθηκε παραπάνω το Axis2 χρησιμοποιεί την έννοια ενός J2EE-ομοειδούς μηχανισμού ενσωμάτωσης. Σε οποιονδήποτε κεντρικό υπολογιστή εφαρμογών J2EE η ενσωμάτωση υπηρεσιών μπορεί να γίνει μέσω αυτόνομων πακέτων , όπου μπορούν να συσσωρευτούν όλοι οι πόροι, τα αρχεία διαμόρφωσης, και τα δυαδικά αρχεία μαζί σε ένα αρχείο και να γίνει η ενσωμάτωσή τους. Αυτό είναι σαφώς χρήσιμο, και γι' αυτό το Axis2 έχει εισαγάγει επίσης τον ίδιο μηχανισμό για να ενσωματώνει τις υπηρεσίες (και τις ενότητες) βολικά.
Η εσωτερική δομή του αυτόνομου πακέτου του Axis2 (ή του συμπιεσμένου αρχείου) παρουσιάζεται στο παρακάτω σχήμα. Οι υπηρεσίες του Axis2,το συμπιεσμένο αρχείο και αρχείο ενότητας, είναι παρόμοια. Οι δευτερεύουσες διαφορές περιλαμβάνουν:

- Στην περίπτωση της υπηρεσίας του Axis, το αρχείο περιγραφής των υπηρεσιών είναι το services.xml, και στην περίπτωση του Axis module, το αρχείο περιγραφής είναι το module.xml.

- Η επέκταση των αρχείων για την υπηρεσία του Axis2 είναι .aar (π.χ. το όνομα αρχείου μιας υπηρεσίας θα είναι MyService.aar), και η επέκταση αρχείου για το module είναι .mar (π.χ. το όνομα αρχείου module θα είναι MyModule.mar).

Δομή ενός αρχείου συμπίεσης (archive file)



Η αποθήκη (repository) του Axis2

Η αποθήκη του Axis2 είναι ουσιαστικά ένας κατάλογος στο σύστημα αρχείων με μια συγκεκριμένη δομή. Μπορεί να βρεθεί είτε τοπικά είτε σε ένα απομακρυσμένο υπολογιστή. Η ιδέα της έννοιας αποθηκών εισήχθη για την κατάλληλη υποστήριξη συμπιεσμένων αρχείων και του *hot deployment*.
Ο κατάλογος αποθηκών αποτελείται από δύο κύριους υποκαταλόγους που ονομάζονται *services* και *modules*. Μπορεί επίσης, να υπάρχει και ένας προαιρετικός υποκατάλογος που ονομάζεται *lib*. Για να ενσωματώσουμε μια υπηρεσία, πρέπει να τοποθετηθεί το συμπιεσμένο αρχείο της υπηρεσίας στον κατάλογο *services*. Ομοίως, για να ενσωματώσουμε ένα module, απλά τοποθετούμε το συμπιεσμένο αρχείο του module στον κατάλογο *modules*. Η ιδέα πίσω από τον κατάλογο *lib* είναι ότι ενεργεί ως θέση για να κρατήσει τις βιβλιοθήκες τρίτων κοινές για τις υπηρεσίες και τα module. Μια γραφική αντιπροσώπευση του repository παρουσιάζεται στο παρακάτω σχήμα.

Η αποθήκη του Axis2

Νέοι περιγραφείς ενσωμάτωσης

Η ευελιξία και οι προοπτικές εξέλιξης του Axis2 στρέφονται στους περιγραφείς ενσωμάτωσής του. Εκτός από την παραμετροποίηση του Axis2 με ένα αρχείο διαμόρφωσης, υπάρχουν και διαφορετικά αρχεία διαμόρφωσης για διαφορετικά επίπεδα διαμόρφωσης. Για παράδειγμα, εάν θέλουμε να προσθέσουμε έναν χειριστή στο σύστημα, δεν είναι ανάγκη να

αλλαχτεί το κεντρικό αρχείο διαμόρφωσης του Axis2, αλλά μπορεί να γίνει με την αλλαγή μόνο του αρχείου διαμόρφωσης υποεφαρμογών (modules configuration file). Υπάρχουν τρεις τύποι περιγραφέων ή αρχείων διαμόρφωσης στοAxis2:

- Κεντρικός περιγραφέας (axis2.xml)
- Περιγραφέας υπηρεσιών (services.xml)
- Περιγραφέας υποεφαρμογών (module.xml)

Στον κεντρικό περιγραφέα, όλες οι διαμορφώσεις του συστήματος δίνονται από το αρχείο axis2.xml, συμπεριλαμβανομένων των εξής:

- Παράμετροι
- Αποστολείς μεταφορών
- Παρατηρητές μεταφορών
- Φάσεις
- Δημόσια υποεφαρμογή

Το Axis2 διαθέτει το axis2.xml με τις προεπιλεγμένες διαμορφώσεις από την Apache. Περιέχει τις ελάχιστες διαμορφώσεις που απαιτούνται για να λειτουργήσει το Axis2, αλλά μπορούμε να το διαμορφώσουμε ανάλογα με τις επιθυμίες μας και να λειτουργήσουμε το Axis2 με το δικό μας axis2.xml. Είναι σημαντικό να σημειωθεί ότι εάν κάνουμε οποιαδήποτε αλλαγή στο axis2.xml, πρέπει να επανεκινήσουμε το σύστημα για να πραγματοποιήθούν οι αλλαγές.

Στον περιγραφέα υπηρεσιών, η διαμόρφωση σχετικά με την υπηρεσία δίνεται από το αρχείο services.xml. Για να είναι έγκυρη η υπηρεσία, πρέπει να συμπεριλάβουμε το αρχείο services.xml στο συμπιεσμένο αρχείο της υπηρεσίας μας. Το αρχείο διαμόρφωσης υπηρεσιών περιέχει τα εξής:

- Παράμετροι επιπέδου υπηρεσίας
- Περιγραφή της υπηρεσίας
- Δέκτες μηνυμάτων
- Λειτουργίες που χρειάζονται για να δημοσιοποιήσουμε σαν υπηρεσίες διαθέσιμες στον δίκτυο (λειτουργίες στην υπηρεσία)
- Υποεφαρμογές επιπέδου υπηρεσιών

Το αρχείο περιγραφής υποεφαρμογών, ή module.xml, αποτελείται από τα δεδομένα διαμόρφωσης που απαιτούνται για να συνδέσουν την υποεφαρμογή με το σύστημα. Οι βασικές διαμορφώσεις περιλαμβάνουν τα εξής:

- Χειριστές και οι κανόνες φάσης τους
- Παράμετροι υποεφαρμογής

Είναι σημαντικό να σημειωθεί ότι το module.xml μπορεί επίσης να περιλάβει τα ακόλουθα στοιχεία:

- Μια περιγραφή για την υποεφαρμογή (και την προδιαγραφή που εκπληρώνει)
- Σημεία παραλαβής (Endpoints)

Διαθέσιμες μέθοδοι ενσωμάτωσης στο Axis2

Στο Axis2, υπάρχουν τρεις κύριοι τρόποι να ενσωματωθεί μια υπηρεσία:

- Τοποθέτηση του συμπιεσμένου αρχείου υπηρεσιών στην αποθήκη.

- Δημιουργία υπηρεσίας προγραμματιστικά χρησιμοποιώντας συμπιεσμένο αρχείο.
- Ενσωμάτωση υπηρεσίας ως Plain Old Java Object (POJO).

Η πιο κοινή προσέγγιση στην ενσωμάτωση μιας υπηρεσίας στο Axis2 είναι απλά να τοποθετήσουμε το συμπιεσμένο αρχείο υπηρεσιών στην αποθήκη (ο κατάλογος *services*).

Τρόποι Ανάπτυξης Υπηρεσιών

Το Axis2 παρέχει διάφορους τρόπους να δημιουργηθεί μια υπηρεσία, όπως:

- Δημιουργία υπηρεσίας από την αρχή. Σε αυτήν την περίπτωση, χτίζουμε το αρχείο υπηρεσίας class της Java προσθέτοντας κατά την διάρκεια που γράφουμε τον κώδικα υποστήριξη στα αντικείμενα στοιχείων του AXIOM (OMElements) , και ύστερα δημιουργούμε το αρχείο services.xml και το τοποθετούμε σε ένα συμπιεσμένο αρχείο JAR για να το ενσωματώσουμε.
- Ενσωμάτωση των POJOs ως υπηρεσία.
- Δημιουργία της υπηρεσίας από WSDL. Από τι στιγμή που μπορούμε να δημιουργήσουμε clients από αρχείο WSDL, μπορούμε να δημιουργήσουμε και τον σκελετό της υπηρεσίας.

Στην αναφορά αυτή θα ασχοληθούμε με την δημιουργία υπηρεσίας από την αρχή, διότι προσφέρει πλήρη έλεγχο του παραγόμενου αποτελέσματος, κάτι το οποίο μας δίνει την δυνατότητα να διαμορφώσουμε εμείς όπως θέλουμε την υπηρεσίας μας και να έχουμε απόλυτο έλεγχο των αποτελεσμάτων των λειτουργιών της και των μεθόδων της.

Τύποι sessions στο Axis2

Μια επιχειρηματική web service δεν μπορεί να στηριχτεί από μόνη της εκτός αν η έννοια του session μεταβιβασθεί στις μηχανές των web services. Με άλλα λόγια, είναι λανθασμένο να πούμε ότι κάθε μηχανή web service βασισμένη στο SOAP πρέπει να διατηρήσει τα sessions επειδή μπορούμε να έχουμε υπηρεσίες που δεν απαιτούν την διατήρηση των sessions. Ένα καλό παράδειγμα θα μπορούσε να είναι η υπηρεσία αναζήτησης της Google και υπηρεσία διόρθωσης ορθογραφικού λάθους της Google.
Η απευθείας διαχείριση των sessions προκαλεί την αύξηση της χρησιμοποίησης της μνήμης και μειώνει την απόδοση. Όμως, το Axis2 προορίζεται να είναι μια επιχειρηματική μηχανή web services, έτσι πρέπει να υποστηρίξει τη διαχείριση sessions.
Όταν εξετάζονται τα sessions, μπορούν επίσης να υπάρξουν διαφορετικοί τύποι sessions, και η διάρκεια ζωής του session μπορεί να ποικίλει. Μερικά session διαρκούν μερικά δευτερόλεπτα ενώ άλλα διαρκούν όσο η διάρκεια ζωής του συστήματος. Στο Axis2 η αρχιτεκτονική έχει ως σκοπό να υποστηρίξει τέσσερις τύπους συνόδων. Υπάρχουν μερικές διαφορές από τη μια στην άλλη. Ενδεχομένως μπορούμε να αναπτύξουμε οποιοδήποτε είδος web service με τη χρησιμοποίηση των ακόλουθων τεσσάρων τύπων πλαισίων sessions (scopes).

- Request
- SOAP Session
- Application
- Transport

Request session

Το πλαίσιο sessions Request είναι το προεπιλεγμένο session του Axis2. Όταν ενσωματώνουμε μια υπηρεσία χωρίς να γνωρίζουμε τίποτα για τη διαχείριση sessions, η

υπηρεσία θα ενσωματωθεί στο συγκεκριμένο πλαίσιο session. Η διάρκεια ζωής του συγκεκριμένου session περιορίζεται στη διάρκεια ζωής της κλήσης μεθόδου, ή το χρόνο επεξεργασίας της αίτησης. Έτσι, όταν επεκτείνετε μια υπηρεσία στο πλαίσιο Request, δεν έχουμε τρόπο διαχειριστούμε το session, είναι το ίδιο σαν να μην είχαμε session καθόλου.

Μόλις ενσωματώσουμε μια υπηρεσία στο πλαίσιο sessions Request, για κάθε κλήση της υπηρεσίας, θα δημιουργείται και από ένα αντίγραφο της υπηρεσίας . Για παράδειγμα έχουμε ενσωματώσει μια υπηρεσία με όνομα "foo" στο πλαίσιο Request. Αν ένας πελάτης καλέσει την υπηρεσία 10 φορές, θα υπάρξουν 10 αντίγραφα της υπηρεσίας.
Ακόμα και αν ενσωματώσουμε μια υπηρεσία στο πλαίσιο Request, υπάρχουν πάρα πολλοί τρόποι να διαχειριστούμε τα sessions στο Axis2. Ο ένας είναι να αποθηκεύσουμε την κατάσταση του session στο γενικό πλαίσιο του Axis2 (configuration context) και να το επαναφέρουμε όποτε θελήσουμε.

Session SOAP

Το πλαίσιο session SOAP έχει μια ελαφρώς πιο μεγάλη διάρκεια ζωής από πλαίσιο Request, και η ενσωματώνοντας μια υπηρεσία σε ένα SOAP session απαιτείται να αλλάξει το αρχείο services.xml. Για να διαχειριστούμε ένα SOAP session απαιτεί από τον πελάτη και από την υπηρεσία να παραμένουν ενήμεροι για το session. Με άλλα λόγια, ο πελάτης πρέπει να στείλει τα σχετικά με το session δεδομένα, εάν θέλει να έχει πρόσβαση στο ίδιο session, και η υπηρεσία πρέπει να επικυρώσει το χρήστη με τη χρήση αυτών των δεδομένων.
Για να διαχειριστεί ένα SOAP session, ένας πελάτης πρέπει να στείλει μια παράμετρο αναφοράς μέσω της επικεφαλίδας του SOAP. Η παράμετρος αυτή ονομάζεται serviceGroupId. Το SOAP session παρέχει έναν τρόπο να ρυθμιστεί το session όχι μόνο για μια μόνο κλήση υπηρεσίας αλλά και για πολλαπλές υπηρεσίες σε μια ομάδα υπηρεσιών. Εφ' όσον είμαστε στο ίδιο session, μπορούμε να διαχειριστούμε τα σχετικά με την υπηρεσία δεδομένα σε ένα γενικό πλαίσιο υπηρεσιών. Εάν θέλουμε να μοιραστούμε τα δεδομένα με άλλες υπηρεσίες στην ομάδα, μπορούμε να χρησιμοποιήσουμε την μέθοδο serviceGroupContext.
Όταν ενσωματώνουμε μια υπηρεσία στο SOAP session και όταν ένας πελάτης προσπαθεί να έχει πρόσβαση στην υπηρεσία την πρώτη φορά, το Axis2 θα παράγει ένα serviceGroupId και θα το στείλει στον πελάτη ως παράμετρο στο στοιχείο wsa:ReplyTo του XML μηνύματος, όπως παρουσιάζεται παρακάτω:

```
<wsa:ReplyTo>
   <wsa:Address>
      http://www.w3.org/2005/08/addressing/anonymous
   </wsa:Address>
   <wsa:ReferenceParameters>
      <axis2:ServiceGroupId xmlns:axis2=
         "http://ws.apache.org/namespaces/axis2">
            urn:uuid:65E9C56F702A398A8B11513011677354
      </axis2:ServiceGroupId>
   </wsa:ReferenceParameters>
</wsa:ReplyTo>
```

Εάν ο πελάτης θέλει να παραμείνει στο ίδιο session, πρέπει να αντιγράψει αυτή την παράμετρο και να την στέλνει πίσω στον εξυπηρετητή όταν καλέσει την υπηρεσία για δεύτερη φορά. Εφ' όσον ένας πελάτης στέλνει έγκυρο serviceGroupId, μπορεί να χρησιμοποιήσει το ίδιο session, και η υπηρεσία μπορεί να διατηρήσει σχετικά με το session δεδομένα. Αντίθετα με το session Request, ένα SOAP session έχει μια προεπιλεγμένη περίοδο παύσης, έτσι εάν ο πελάτης δεν χρησιμοποιήσει την υπηρεσία για 30 δευτερόλεπτα, το session θα λήξει, και εάν ο πελάτης στείλει μετά το παλαιό serviceGroupId, θα παραλάβει σαν απάντηση ένα μήνυμα λάθους AxisFault.

Transport session

Στην περίπτωση ενός Application session, το Axis2 χρησιμοποιεί τις σχετικές με το Transport session τεχνικές για να διαχειριστεί το session. Για παράδειγμα, στην περίπτωση του HTTP, χρησιμοποιεί τα HTTP cookies για να διαχειριστεί το session. Η διάρκεια ζωής του session ελέγχεται από τη μεταφορά, και όχι από το Axis2. Το Axis2 αποθηκεύει τις σχετικές πληροφορίες με την υπηρεσία και το serviceGroupContext στο Transport session αντικείμενο, έτσι ώστε η υπηρεσία να μπορεί να έχει πρόσβαση σε αυτές τις πληροφορίες όσο το session παραμένει ανοιχτό.
Ένα από τα βασικά πλεονεκτήματα του Transport session σε σχέση με άλλα sessions είναι ότι μπορούμε να μιλήσουμε σε πολλαπλές ομάδες υπηρεσιών μέσα από ένα μόνο Transport session. Σε ένα SOAP session, δεν έχουμε την δυνατότητα να επικοινωνήσουμε μεταξύ δύο ομάδων υπηρεσιών, αλλά με Transport session έχουμε αυτή την δυνατότητα. Σε αυτήν την περίπτωση, ο αριθμός των αντίγραφων των υπηρεσιών που δημιουργούνται εξαρτάται από τον αριθμό των Transport session που δημιουργούνται.

Application session

Το Application session έχει την πιο μεγάλη διάρκεια ζωής έναντι των άλλων. Η διάρκεια ζωής του είναι ίση με τη διάρκεια ζωής του συστήματος. Εάν ενσωματώσετε μια υπηρεσία σε Application session, θα υπάρξει μόνο ένα αντίγραφο της υπηρεσίας που θα εκτελείτε και προφανώς θα υπάρξει μόνο ένα γενικό πλαίσιο της υπηρεσίας για αυτή την υπηρεσία. Επίσης το Application session εξαναγκάζει τους πελάτες που καλούν την υπηρεσία να παραμένουν κάθε φορά στο ίδιο session. Αυτό το χαρακτηριστικό είναι πολύ χρήσιμο αφού δεν χρειάζεται να αποστέλλουμε κάθε φορά το serviceGroupId για να παραμείνουμε στο ίδιο session. Στο Axis2, εάν λάβουμε υπόψη μας την χρήση της μνήμης τότε το Application session.

Για να εφαρμόσουμε κάποιον από αυτούς τους τύπους session πρέπει να το δηλώσουμε στο αρχείο services.xml. Το πώς θα γίνει αυτό θα το εξηγήσουμε παρακάτω.

Όπως παρατηρούμε το Axis2 είναι ένα πλήρες παραμετροποιήσημο ενδιάμεσο υλι/σμικό , το οποίο μας δείνει την δυνατότητα να αναπτύξουμε μία web service και να την ενσωματώσουμε σε αυτό και μέσω των διαφόρων παραμέτρων του να διαμορφώσουμε τον τρόπο με τον οποίο θα γίνει η επικοινωνία και η διαμεταγωγή των μυνημάτων ανάμεσα στην web service και τους πελάτες.

Ανάπτυξη υπηρεσίας

Θα αναπτύξουμε μία υπηρεσία και στη συνέχεια θα την ενσωματώσουμε στο Axis2. Η υπηρεσία αυτή θα μετράει τις κλήσεις των πελατών και θα επιστρέφει τον αριθμό των κλήσεων σε όποιον πελάτη την καλέσει. Για να είναι αυτό εφικτό θα πρέπει η υπηρεσία να μπορεί να παραμένει σε λειτουργία ανάμεσα στις κλήσεις πελατών και να μπορεί να προσφέρει στους πελάτες τα σωστά αποτελέσματα. Με την υπηρεσία αυτή θα δούμε ότι με το Axis2 μπορούμε με διαφορετικούς πελάτες, να καλέσουμε την web service και να πάρουμε δεδομένα τα οποία πριν έχουν αλλαχτεί από κάποια άλλη κλήση πελάτη.
Θα ενσωματώσουμε την υπηρεσία χρησιμοποιώντας το Application session, το οποίο μας προσφέρει μόνο ένα αντίγραφο της υπηρεσία το οποίο θα καλούν όλοι οι πελάτες. Αφού το Application session αναγκάζει τους πελάτες να παραμένουν στο ίδιο session σε κάθε κλήση στην web service, δεν χρειάζεται να ανησυχήσουμε γι’ αυτό.

Χρησιμοποιώντας την JAVA δημιουργούμε το αρχείο class της υπηρεσίας. Αυτό είναι πάρα πολύ απλό. Δημιουργήσουμε μια μέθοδο, η οποία θα είναι και η λειτουργία της υπηρεσίας μας. Στην μέθοδο αυτή θα περνάμε σαν παράμετρο ένα στοιχείο του AXIOM (OMElement) το οποίο, όπως αναφέραμε, είναι το πρότυπο το οποίο χρησιμοποιεί το Axis2 για να χτίζει τα μηνύματα που θα μεταφερθούν. Την μέθοδο θα την ονομάσουμε echo και θα επιστρέφει έναν ακέραιο. Η μέθοδος της υπηρεσίας μας θα έχει την εξής απλη μορφή:

```
public int echo(OMElement element)
   {
      counter++;
      return counter;
   }
```

Όπως παρατηρούμε, η λειτουργία της υπηρεσίας είναι απλά να επιστρέψει στο πελάτη που την κάλεσε ένα ακέραιο ο οποίος είναι ένας μετρητής των κλήσεων που έχει δεχθεί η υπηρεσία.
Επίσης, παρατηρούμε ότι από την μεριά της υπηρεσίας δεν χρειάζεται να διαχειριστούμε τις συνδέσεις καθώς ο πελάτης θα αποφασίσει με ποιον τρόπο θα συνδεθεί με την υπηρεσία. Αυτό δίνει μεγάλη ευελιξία στην υπηρεσία αφού δεν χρειάζεται να ανησυχεί για του διαφορετικού τύπου συνδέσεις που θα δέχεται. Το μόνο που έχουμε να ανησυχούμε για την υπηρεσία είναι την σωστή επιστροφή της απάντησης και την διαχείριση του session αν ο πελάτης επιθυμεί να γίνεται διαχείριση του session. Στην περίπτωσή μας, το Application scope, όπως είπαμε, εξαναγκάζει τους πελάτες να συνδέονται στο ίδιο session κάθε φορά άρα δεν χρειάζεται να το διαχειριστούμε. Αφού χτίσουμε την υπηρεσία, η Java θα μας επιστρεψει ένα αρχείο class. Έστω ότι ονομάσαμε την υπηρεσία SimpleService τότε θα πάρουμε ένα αρχείο SimpleService.class.
Στην συνέχεια πρέπει να δημιουργήσουμε το αρχείο services.xml με το οποίο θα περιγράψουμε την υπηρεσία μας. Το services.xml είναι ένα XML αρχείο το οποίο περιέχει όλες τις απαραίτητες πληροφορίες για να μπορέσει να ενσωματώσει το Axis2 την υπηρεσία μας. Έτσι τα περιεχόμενα του αρχείου services.xml θα είναι:

```
<service name="SimpleService" scope="application">
   <Description>
      A simple Echo action
   </Description>
   <parameter name="ServiceClass"
         locked="false">SimpleService.SimpleService</parameter>
   <operation name="echo">
         <messageReceiver  class="org.apache.axis2.rpc.receivers.RPCMessageReceiver" />
   </operation>
</service>
```

Το στοιχείο *service* περιέχει την παράμετρο *name* οποία είναι το όνομα της υπηρεσίας μας και το *scope* το οποίο δηλώνει το session το οποίο θα χρησιμοποιήσουμε το οποίο στην περίπτωσή μας είναι το Application. Στο *description* μπορούμε να δώσουμε μια περιγραφή της υπηρεσίας μας. Το στοιχείο *parameter* περιέχει την παράμετρο *name* με την οποία δηλωνουμε τον τύπο του αρχείου που θέλουμε να ενσωματώσουμε, στην περίπτωση μας είναι ένα αρχείο class υπηρεσίας άρα θα πρέπει να δηλώσουμε *ServiceClass*. Το στοιχείο *operation* περιέχει μία παράμετρο με όνομα *name* οι οποία δηλώνει το όνομα της λειτουργία που προσθέσαμε στην υπηρεσία. Στην περίπτωσή μας ονομάσαμε την λειτουργία *echo* άρα θα πρέπει να το δηλώσουμε εδώ. Το στοιχείο αυτό περιέχει δύο στοιχεία παιδιά, το *messageReceiver* όπου πρέπει να δηλώσουμε την τοποθεσία του αποδέκτη μυνημάτων (ένα URI).

Αφού ολοκληρώσουμε την διαμόρφωση του αρχείου services.xml μπορούμε να φτιάξουμε το αρχείο aar και να τοποθετήσουμε τα αρχεία SimpleService.class και services.xml με την διάταξη που περιγραψαμε παραπάνω. Στην συνέχεια τοποθετούμε το αρχείο στην αποθήκη του Axis2, όπως επίσης περιγράψαμε παραπάνω και η ενσωμάτωση έχει ολοκληρωθεί. Αξίζει να σημειωθεί ότι το Axis2 δημιουργεί αυτόματα το αρχείο WSDL με την ενσωμάτωση της υπηρεσίας. Αυτό είναι πολύ βολικό και εξοικονομεί χρόνο στον προγραμματιστή που θέλει απλά να ενσωματώσει μια απλή web service. Φυσικά μπορούμε να δημιουργήσουμε και το δικό μας WSDL.

Ανάπτυξη εφαρμογής πελάτη

Πολλές μηχανές web services παρέχουν στους χρήστες τις μεθόδους blocking και non-Blocking για τα APIs πελάτη.

- **Blocking API:** Μόλις κληθεί η μέθοδος κλήσης υπηρεσιών, η εφαρμογή πελατών σταματάει την λειτουργία της και επανακτά τον έλεγχο μόνο όταν η διαδικασία ολοκληρωθεί, και ο πελάτης λαμβάνει μια απάντηση ή ένα μήνυμα λάθους. Αυτόσ είναι ο απλούστερος τρόπος να κληθεί μία web service, και εξυπηρετεί αρκετές περιπτώσεις επιχειρήσεων.
- **Non-Blocking API:** Αυτό είναι μια κλήση με απάντηση (callback) ή ένα API βασισμένο στην καταμέτρηση. Ως εκ τούτου μόλις κληθεί μια υπηρεσία, η εφαρμογή πελατών επανακτά αμέσως τον έλεγχο και η απάντηση ανακτάται χρησιμοποιώντας το παρεχόμενο αντικείμενο αίτησης με απάντηση (callback). Αυτή η προσέγγιση παρέχει την ευελιξία στην εφαρμογή πελατών να καλεστούν διάφορες web services ταυτόχρονα χωρίς φράξιμο της λειτουργίας που έχει ήδη καλεστεί.

Και οι δύο μηχανισμοί λειτουργούν στο επίπεδο API. Ας ονομάσουμε την ασύγχρονη συμπεριφορά που μπορούμε να έχουμε με την χρήση του non-blocking API ως Ασύγχρονο Επίπεδο API.

Και οι δύο μηχανισμοί χρησιμοποιούν συνδέσεις μονής μεταφοράς για να στείλουν το αίτημα και για να λάβουν την απάντηση. Έχουν σοβαρή αδυναμία να χρησιμοποιήσουν διπλής σύνδεσης μεταφοράς για το αίτημα και την απάντηση (είτε μονόδρομης είτε διπλής κατεύθυνσης). Έτσι και οι δύο αυτοί μηχανισμοί αποτυγχάνουν να διευθετήσουν το πρόβλημα μακροχρόνιων συναλλαγών (η σύνδεση μεταφοράς μπορεί να τερματίσει προτού να ολοκληρωθεί η λειτουργία). Μια πιθανή λύση θα ήταν να χρησιμοποιηθούν δύο χωριστές συνδέσεις μεταφορών για το αίτημα και την απάντηση. Η ασύγχρονη συμπεριφορά που κερδίζουμε με τη χρησιμοποίηση αυτής της λύσης μπορεί να ονομαστεί Ασύγχρονο επίπεδο μεταφοράς

Με το συνδυασμό του ασύγχρονου επιπέδου API και του ασύγχρονου επιπέδου μεταφοράς, μπορούμε να έχουμε τέσσερα διαφορετικά σχέδια κλήσης για web services όπως φαίνεται στον ακόλουθο πίνακα.

API (Blocking/Non-Blocking)	**Ασύγχρονη Μεταφορά (Ναι/Οχι)**	**Description**

Blocking	Όχι	Το απλούστερο και πιό γνωστό σχέδιο κλησης
Non-Blocking	Όχι	Χρησιμοποιεί callbacks ή σύστημα καταμέτρησης
Blocking	Ναι	Αυτό είναι χρήσιμο όταν η λειτουργία της υπηρεσίας είναι in-out η μεταφορά που χρησιμοποιείται είναι μονόδρομη (π.χ. SMTP)
Non-Blocking	Ναι	Αυτό μπορεί να χρησιμοποιηθεί για να έχουμε την μέγιστη ασύγχρονη συμπεριφορά. Non-Blocking στο επίπεδο API αλλά και στο επίπεδο μεταφορών.

Υπάρχουν τέσσερεις διαφορετικοί τύποι εφαρμογών πελατών:

1. Απλή blocking εφαρμογή, αίτησης-απάντησης (request-responce)
2. Μονόδρομη εφαρμογή πελάτη
3. Non-blocking, Request-Response εφαρμογή πελάτη, που χρησιμοποιεί μία σύνδεση μεταφοράς.
4. Non-Blocking, Request-Response εφαρμογή πελάτη, που χρησιμοποιεί δύο συνδέσεις μεταφοράς.

Απλή blocking εφαρμογή, αίτησης-απάντησης (request-responce)

Το Axis2 παρέχει στο χρήστη διάφορα πρότυπα κλήσης για web services, που κυμαίνονται από τις καθαρές blocking κλήσεις μονού καναλιού ως τις non-blocking κλήσεις διπλών καναλιών. Πρώτα ας δούμε πώς μπορούμε να γράψουμε έναν πελάτη για να καλέσουμε τη λειτουργία "*echo*" της υπηρεσίας "*SimpleService*" που φτιάξαμε πριν χρησιμοποιώντας την απλή blocking κλήση. Ο κώδικας πελάτη είναι ο ακόλουθος.

```
try {
        OMElement payload = CreateMessage();

        Options options = new Options();
        options.setTo(targetEPR); // Η εντολή αυτή δηλώνει την τοποθεσία της SimpleService
        ServiceClient serviceClient = new ServiceClient();
        serviceClient.setOptions(options);
        OMElement result = serviceClient.sendReceive(payload);

        System.out.println(result);

    } catch (AxisFault axisFault) {
       axisFault.printStackTrace();
    }
}
```

1. Οι γραμμές που τονίζονται με πράσινο παρουσιάζουν το σύνολο των διαδικασιών που πρέπει να εκτελεστούν προκειμένου να κληθεί μια υπηρεσία Ιστού.

2. Το υπόλοιπο χρησιμοποιείται για να δημιουργήσει το OMElement που πρέπει να σταλεί και να τυπώσει την απάντηση που θα έρθει.

Μονόδρομη εφαρμογή πελάτη

Ας δούμε τώρα πως είναι ο κώδικας για την κλήση της λειτουργίας "*echo*" με *in-only* τρόπο.

```
try {
      OMElement payload = CreateMessage();
      Options options = new Options();
      options.setTo(targetEPR);
      ServiceClient serviceClient = new ServiceClient();
      serviceClient.setOptions(options);
      serviceClient.fireAndForget(payload);
       /**
        * Πρέπει να διακόψουμε την λειτουργία του νήματος μέχρι να στείλουμε την αίτηση,
        * το πρόβλημα είναι ότι εάν εξέλθουμε από το κεντρικό νήμα, τότε η αίτηση
        * δεν θα αποσταλεί, άρα πρέπει να περιμένουμε για λίγο
        */
      Thread.sleep(500);
    }
catch (AxisFault axisFault) {
          axisFault.printStackTrace();
    }
```

Δεδομένου ότι έχουμε πρόσβαση σε μια υποθετικά *in-only* λειτουργία, μπορούμε άμεσα να χρησιμοποιήσουμε την μέθοδο *fireAndForget()* στο αντικείμενο *ServiceClient* για να καλέσουμε αυτήν την λειτουργία. Αυτό δεν θα εμποδίσει την κλήση και θα επιστρέψει πίσω τον έλεγχο αμέσως στον πελάτη.

Non-blocking, Request-Response εφαρμογή πελάτη, που χρησιμοποιεί μία σύνδεση μεταφοράς.

Στην απλή blocking εφαρμογή, αίτησης-απάντησης μόλις η serviceClient.sendReceive(Payload) καλείται, ο πελάτης φράζεται μέχρις ότου η λειτουργία ολοκληρωθεί. Αυτή η συμπεριφορά δεν είναι επιθυμητή όταν υπάρχουν πολλές κλήσεις σε web service να πραγματοποιηθούν σε μια ενιαία εφαρμογή πελατών ή μέσα σε ένα GUI. Μια λύση θα ήταν να χρησιμοποιηθεί ένα non-Blocking API για να κληθούν οι web services. Το Axis2 παρέχει ένα βασισμένο στην αίτηση με απάντηση (callback) non-blocking API για τους χρήστες.
Ο κώδικας λοιπόν της απλής blocking εφαρμογής αίτησης-απάντησης θα αλλάξει λίγο και θα προστεθεί το callback του οποίου ο κώδικας θα είναι

```
Callback callback = new Callback() {
  public void onComplete(AsyncResult result) {
     System.out.println(result.getResponseEnvelope());
  }
public void onError(Exception e) {
     e.printStackTrace();}};
```

Ο χρήστης αναμένεται να ενσωματώσει τις μεθόδους *"onComplete"* και *"onError"* στην μέθοδο *callback*. Η μηχανή Axis2 καλεί τη μέθοδο "onComplete" μόλις παραληφθεί η απάντηση από το API πελάτη του Axis2 (ServiceClient). Αυτό αποβάλει την φραγή της κλήσης της web service και θα παρέχει στους χρήστες την ευελιξία να χρησιμοποιηθεί το non-blocking API για τους πελάτες της web service.
Επίσης θα πρέπει να αλλάξει και η μέθοδος κλήσης της web service serviceClient.sendReceive(payload) έτσι ώστε να είναι non-blocking και να περιέχει και το αντικείμενο callback στις παραμέτρους της για να μπορέσει να πραγματοποιηθεί η παραλαβή της επιστροφής. Άρα η μέθοδος θα γίνει:

```
sender.sendReceiveNonBlocking(payload, callback);
```

Non-Blocking, Request-Response εφαρμογή πελάτη, που χρησιμοποιεί δύο συνδέσεις μεταφοράς.

Η λύση που παρέχεται από το non-Blocking API έχει έναν περιορισμό όταν οι κλήσεις στις web services χρειάζονται αρκετό χρόνο να ολοκληρώσουν. Ο περιορισμός οφείλεται στη χρήση μονών συνδέσεων μεταφορών για την κλήση της web service και την ανακτήση της απάντηση. Με άλλα λόγια, το API πελάτη παρέχει έναν non-blocking μηχανισμό κλήσης για τους χρήστες, αλλά το αίτημα και η απάντηση έρχονται σε μια μονή σύνδεση μεταφορών (διπλής κατεύθυνσης μεταφορά, όπως το HTTP). Οι κλήσεις web services που έχουν μεγάλη διάρκεια ή οι κλήσεις των web services,που χρησιμοποιούν τις μονόδρομες μεταφορές (όπως το SMTP), δεν μπορούν να χρησιμοποιηθούν απλά κάνοντας χρήση μιας non-blocking κλήσης.
Η τετριμμένη λύση είναι να χρησιμοποιηθούν χωριστές συνδέσεις μεταφορών (είτε μονόδρομης είτε διπλής κατεύθυνσης) για το αίτημα και την απάντηση. Το επόμενο πρόβλημα που πρέπει να λυθεί είναι ο συσχετισμός (συσχετίζοντας το αίτημα και την απάντηση). Το WS-Addressing παρέχει μια κομψή λύση σε αυτό χρησιμοποιώντας τις επικεφαλίδες < *wsa:MessageID* > και < *wsa:RelatesTo* >. Το Axis2 παρέχει την υποστήριξη για έναν μηχανισμό συσχέτισης βασισμένο στην διεθυνσιοδότιση και ένα συμμορφωμένο API πελάτη για να καλέσει τις web services με δύο συνδέσεις μεταφορών. (Ο πυρήνας του Axis2 δεν βασίζεται στο WS-Addressing, αλλά περιέχει ένα σύνολο παραμέτρων, η οποίες μπορούν να υιοθετηθούν από οποιαδήποτε μέθοδο. Το WS-Addressing είναι μια από τις χρήσεις που μπορεί να τις υιοθετήσει. Ακόμη και οι μεταφορές μπορούν να τις υιοθετήσουν. Ως εκ τούτου, το Axis2 έχει την ευελιξία να χρησιμοποιήσει διαφορετικές εκδόσεις της διευθυνσιοδότισης).

Οι χρήστες μπορούν να επιλέξουν μεταξύ του blocking και non-Blocking APIs για τους πελάτες των web services με δύο συνδέσεις μεταφορών. Χρησιμοποιώντας απλά, μια δυαδική σημαία, το ίδιο API μπορεί να χρησιμοποιηθεί για να καλέσει τις web services (διαδικασίες in-out) χρησιμοποιώντας δύο χωριστές συνδέσεις μεταφορών. Ο ακόλουθος κώδικας επιδεικνύει πώς να κληθεί η λειτουργία *"echo"* χρησιμοποιώντας το non-Blocking API με δύο συνδέσεις μεταφορών. Η απόλυτη ασύγχρονη μεταφορά.

```
try {
        OMElement payload = CreateMessage();

        Options options = new Options();
        options.setTo(targetEPR);
        options.setTransportInProtocol(Constants.TRANSPORT_HTTP);
        options.setUseSeparateListener(true);
```

```
Callback callback = new Callback() {
            public void onComplete(AsyncResult result) {
               System.out.println(result.getResponseEnvelope());
            }

            public void onError(Exception e) {
               e.printStackTrace();
            }
         };
         //Non-Blocking Κλήση
          ConfigurationContext configContext =
          ConfigurationContextFactory.createConfigurationContextFromFileSystem("C:\\Progr
          am Files\\Apache Tomcat 6.0.10\\webapps\\axis2\\WEB-INF", null);
         sender = new ServiceClient();
         sender.engageModule(new QName(Constants.MODULE_ADDRESSING));
         sender.setOptions(options);
         sender.sendReceiveNonBlocking(payload, callback);
         //Αναμονή μέχρι το callback να λάβει τη απάντηση.
         while (!callback.isComplete()) {
          Thread.sleep(1000);
         }

         } catch (AxisFault axisFault)
```

Η δυαδική σημαία (value true) στη μέθοδο *options.setUseSeparateListener(...)* ενημερώνει τη μηχανή του Axis2 να χρησιμοποιήσει τις χωριστές συνδέσεις μεταφορών για την αίτηση και την απάντηση. Τέλος η μέθοδος *sender.cleanup()* ενημερώνει τη μηχανή του Axis2 να σταματήσει τον παρατηρητή γεγονότων της εφαρμογής πελατών, ο οποίος άρχισε να ανακτά την απάντηση.

Κατά την διάρκεια εκτέλεσης των παραπάνω εφαρμογών πελατών, μία κλήση θα πραγματοποιηθεί στην *SimpleService* και αυτή θα επιστρέψει μία απάντηση. Η κλήση θα είναι ένα μήνυμα SOAP το οποίο θα είναι ένα XML. Όπως είπαμε και πριν, το Axis2 κάνει χρήση του AXIOM για να διαχειριστεί τα μηνύματα σε XML που ανταλλάσσονται. Στον κώδικα καλούμε την μέθοδο *CreateMessage()* η οποία χτίζει το μήνυμα με την βοήθια τον μεθόδων του AXIOM. Το μήνυμα αίτησης που θα αποσταλεί στην υπηρεσία θα είναι το ακόλουθο.

```
<SimpleService:echo xmlns:SimpleService="http://SimpleService/xsd">
        <SimpleService:Text>
                Axis2 Echo String
        </SimpleService:Text>
</SimpleService:echo>
```

Η απάντηση θα είναι και αυτή ένα μήνυμα SOAP το οποίο στο στοιχείο <ns:return> του κορμού του μηνύματος (body), θα περιέχει τον ακέραιο που θα αντιστοιχεί στις κλήσεις που έγιναν στην υπηρεσία από όλους του πελάτες. Το μήνυμα SOAP που θα παραλάβουν οι πελάτες θα είναι το παρακάτω.

```
<?xml version='1.0' encoding='utf-8'?>
        <soapenv:Envelope xmlns:wsa="http://www.w3.org/2005/08/addressing"
                xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
                <soapenv:Header>
                        <wsa:To>
```

```
                    http://192.168.1.22:6060/axis2/services/anonService1/anonOutInOp
                </wsa:To>
                <wsa:ReplyTo>
                    <wsa:Address>
                        http://www.w3.org/2005/08/addressing/none
                    </wsa:Address>
                </wsa:ReplyTo>
                <wsa:MessageID>
                    urn:uuid:06C75745A9BD2B14121180254751436
                </wsa:MessageID>
                <wsa:Action>urn:echo</wsa:Action>
                <wsa:RelatesTo>
                    urn:uuid:760287FAEDF098D3C31180254750850
                </wsa:RelatesTo>
            </soapenv:Header>
            <soapenv:Body>
                <ns:echoResponse xmlns:ns="http://SimpleService/xsd">
                    <ns:return>1</ns:return>
                </ns:echoResponse>
            </soapenv:Body>
        </soapenv:Envelope>
```

Διατήρηση attachments αντικειμένων σε XML

Στο Axis2 όλα τα μηνύματα που διακινούνται είναι τύπου XML. Κατά την διάρκεια εκτέλεσης μιας εφαρμογής πελάτη πολλές φορές θα χρειαστεί να θέλουμε να στείλουμε μέσα στο μήνυμα SOAP και αντικείμενα της Java. Έτσι λοιπόν πρέπει το Axis2 να υποστηρίζει αυτή την λειτουργία έτσι ώστε να μπορέσουμε να ενσωματώσουμε τα αντικείμενα στο μήνυμα XML.
Από την στιγμή που η XML είναι η γλώσσα με την οποία χτίζονται τα μηνύματα SOAP, τότε μπορούμε να ενσωματώσουμε, σε μορφή δυαδικής σειράς, το αντικείμενο. Ένας κώδικας ο οποίος ενσωματώνει αντικείμενα σε ένα αρχείο XML σε μορφή δυαδικής σειράς, είναι ο ακόλουθος.

```
try{
    Vector myVector = new Vector();
    myVector.add("0");
    myVector.add("1");
    myVector.add("2");
    File myFile = new File("C:\\test1\\test.test");
    myFile.createNewFile();
    FileOutputStream f_out = new FileOutputStream(myFile);
    ObjectOutputStream objectOutputStream = new ObjectOutputStream(f_out);
    objectOutputStream.writeObject(myVector);
    objectOutputStream.close();
    FileInputStream fileInputStream = new FileInputStream("C:\\test1\\test.test");
     int k1 = fileInputStream.available();

    byte[] b = new byte[k1];
    fileInputStream.read(b);
    String tmp = new String();
    String tmp2 = "<xml><results><string>" + Base64.encode(b) +
                + "</string></results></xml>";
```

```
myFile = new File("C:\\test1\\georrge.xml");
myFile.createNewFile();
FileWriter fW = new FileWriter(myFile);
fW.write(tmp2);
fW.close();
}catch(Exception e){
   e.printStackTrace();
}
```

Στον παραπάνω κώδικα δημιουργούμε ένα αντικείμενο τύπου *vector* και στη συνέχεια με την μέθοδο *File myFile = new File("C:\\test1\\test.test");* δημιουργούμε ένα καινούργιο αρχείο με όνομα *test.test*.Στη συνέχεια τοποθετούμε το *vector* στο αρχείο που μόλις δημιουργήσαμε με τη μέθοδο *objectOutputStream.writeObject(myVector);*, και κλείνουμε το αρχείο για να αποθηκευτεί.
Στη συνέχεια ανοίγουμε το αρχείο που δημιουργήσαμε αλλά αυτή την φορά για διάβασμα μόνο *FileInputStream fileInputStream = new FileInputStream("C:\\test1\\test.test");*.
Μετατρέπουμε το αρχείο που ανοίξαμε σε δυαδική ακολουθία με τις μεθόδους *int k1 = fileInputStream.available();* και *byte[] b = new byte[k1];*. Δημιουργούμε το παρακάτω XML κείμενο:

```
<xml>
    <results>
        <string>
                <-- εδώ θα τοποθετηθεί η δυαδική ακολουθία -->
        </string>
   </results>
</xml>
```

Στο στοιχείο <*string*> τοποθετούμε την δυαδική ακολουθία του αρχείου που μόλις δημιουργήσαμε και κλείνουμε το αρχείο.

Όπως είδαμε είναι εφικτό να περάσουμε αντικείμενα σε ένα XML έγγραφο κάτι το οποίο μας δίνει πολλές δυνατότητες στην ανάπτυξη υπηρεσιών και εφαρμογών πελατών με αρκετές επιπλέον δυνατότητες