Α. Τ. Ε. Ι. ΠΑΤΡΑΣ

ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ & ΟΙΚΟΝΟΜΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κατασκευή δυναμικού web site με χρήση της τεχνικής Ajax

Κατσαδωράκη Μαλβίνα(1740)
Μήλιου Ευαγγελία(1710)
Ρουμπάκου Ευγενία(1686)

Επιβλέπων: Στάμος Κωνσταντίνος
Υπεύθυνος: Στάμος Κωνσταντίνος

Πάτρα, Ιούλιος 2011

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Εισαγωγή στο Ajax

1.1 Ιστορική αναδρομή

Ajax (Asynchronous JavaScript και XML) είναι περισσότερο ένας όρος που χρησιμοποιείται για να περιγράψει το πώς οι τεχνολογίες JavaScript, DOM (Document Object Model) και XML (Extensible Markup Language) όταν συνδυαστούν για την δημιουργία διαδραστικών εφαρμογών web καταργούν την ανάγκη της ανανέωσης μίας ιστοσελίδας όταν μόνο μέρος της αλλάζει.

Την δεκαετία του 1990 οι περισσότερες ιστοσελίδες ήταν βασισμένες μονάχα σε HTML. Οποιαδήποτε ενέργεια του χρήστη μέσα στην σελίδα στην οποία πλοηγούταν επέφερε ως αποτέλεσμα την ολόκληρη ανανέωση της σελίδας (refresh) αυτής. Το 1995 όμως με τον ερχομό της γλώσσας προγραμματισμού Java έγινε δυνατή για πρώτη φορά η ασύγχρονη φόρτωση εφαρμογών(java applets). Ασύγχρονη φόρτωση βλέπουμε και στον Internet Explorer με την εμφάνιση του στοιχείου i frame στην HTML το 1996. Το 1999 η Microsoft προχώρησε στην δημιουργία του ActiveX (XML + HTTP) και στην εισαγωγή του στον Internet Explorer 5 όπου έπειτα υιοθετήθηκε και από άλλους φυλλομετρητές όπως Mozilla,Safari και Opera. Την μεγάλη όμως εφαρμογή της ασύγχρονης φόρτωσης την βλέπουμε σε εφαρμογές από το 2000 και μετά:

2000: Outlook Web Access

2002:Oddpost

2004:Gmail

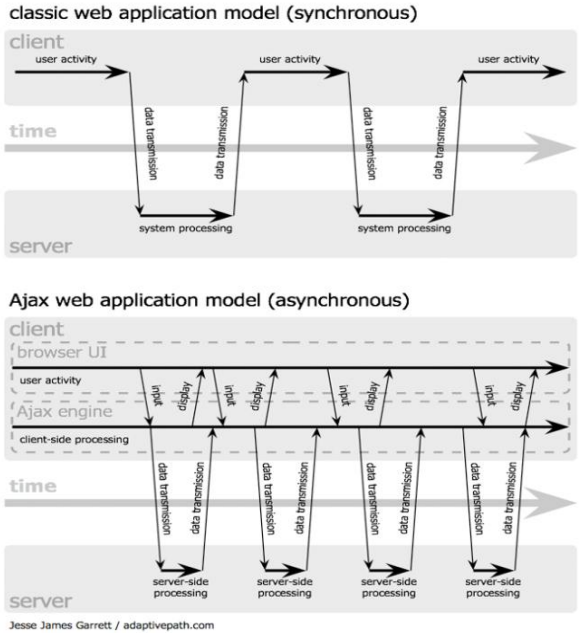
2005:Google Maps

1.2 Παρουσιάζοντας το Ajax

Τον Φεβρουάριο του 2005 ο Jesse Garrett εξέδωσε ένα άρθρο σχετικά με το Ajax "Ajax: A New Approach to Web Applications" (www.adaptivepath.com/publications/essays/archives/000385.php) όπου εκεί υποστήριζε ότι

με την χρήση αυτής, οι διαδικτυακές εφαρμογές έρχονταν πολύ κοντά στις εφαρμογές που τρέχουν από τον υπολογιστή του χρήστη (desktop applications) [1]. Βάση του άρθρου του αναφέρει ότι οι περισσότερες πράξεις του χρήστη μέσα σε μια ιστοσελίδα απαιτούσαν αποστολή αιτημάτων δια μέσου της HTML προς στον διαδικτυακό εξυπηρετητή (wed server). Ο web server κάνει κάποια επεξεργασία των δεδομένων και αποστέλλει μια νέα HTML σελίδα πίσω στον χρήστη. Αυτό καθιστά την όλη αλληλεπίδραση του χρήστη με την σελίδα αρκετά αργή. Μια εφαρμογή Ajax διορθώνει το πρόβλημα αυτό καθώς δημιουργεί έναν ενδιάμεσο Ajax μηχανισμό μεταξύ χρήστη και εξυπηρετητή. Αυτός ο μηχανισμός είναι υπεύθυνος τόσο για να αποδίδεται σωστά η διεπαφή της σελίδας με τον χρήστη, όσο και για να επικοινωνεί με τον εξυπηρετητή από μέρους του χρήστη. Πιο συγκεκριμένα, επιτρέπει η διεπαφή του χρήση με την εφαρμογή να γίνεται ασύγχρονα, ανεξάρτητα από την επικοινωνία του με τον εξυπηρετητή. Έτσι ο χρήστης δεν θα χρειάζεται ποτέ να μένει κοιτώντας μια λευκή σελίδα στον φυλομετρητή του, περιμένοντας να γίνει η φόρτωση της επόμενης από τον εξυπηρετητή.

Η παραπάνω επεξήγηση μπορεί να αναπαρασταθεί και στο παρακάτω σχέδιο το οποίο είχε στο άρθρο του ο Jesse Garrett:



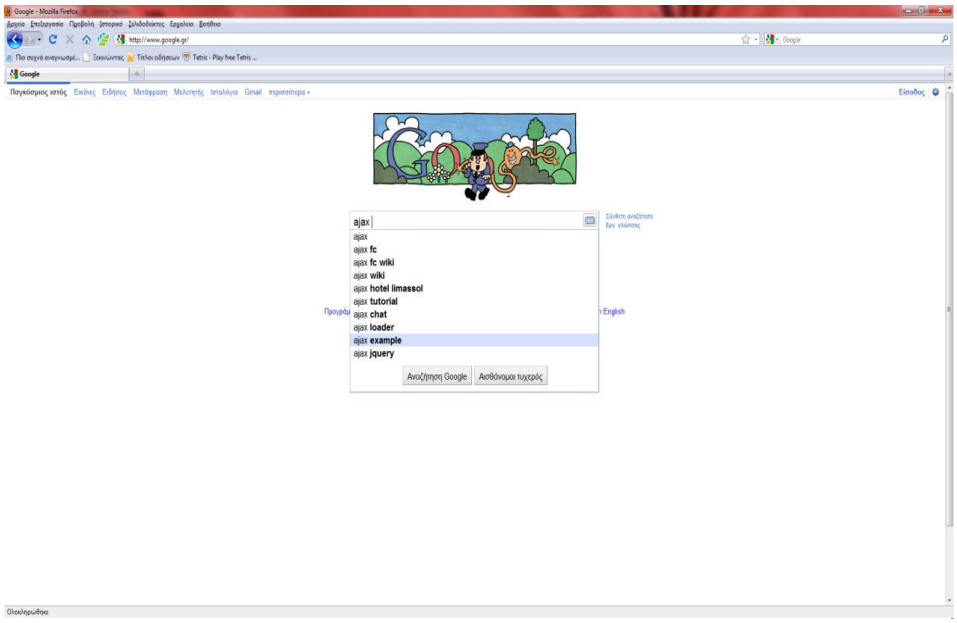
Εικόνα 1: Συγκριτική παρουσίαση του σύγχρονου και του ασύγχρονου μοντέλου μιας δικτυακής εφαρμογής

Το Ajax δεν είναι μια νέα γλώσσα προγραμματισμού και δεν είναι κάτι καινούριο. Δεν είναι τίποτα άλλο παρά ένας συνδυασμός ήδη υπαρχόντων τεχνολογιών [2]:

- HTML/XHTML: Primary content representation languages
- CSS: Provides stylistic formatting to XHTML
- DOM: Dynamic updating of a loaded page
- XML: Data exchange format
- XSLT: Transforms XML into XHTML (styled by CSS)
- XML HTTP: Primary communication broker
- JavaScript: Scripting language used to program an Ajax engine

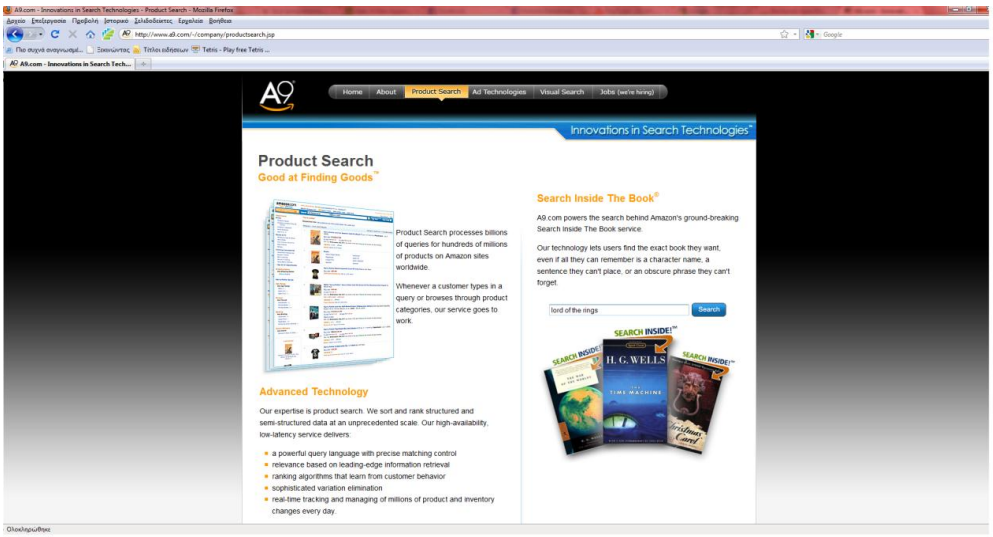
1.3 Παραδείγματα εφαρμογής

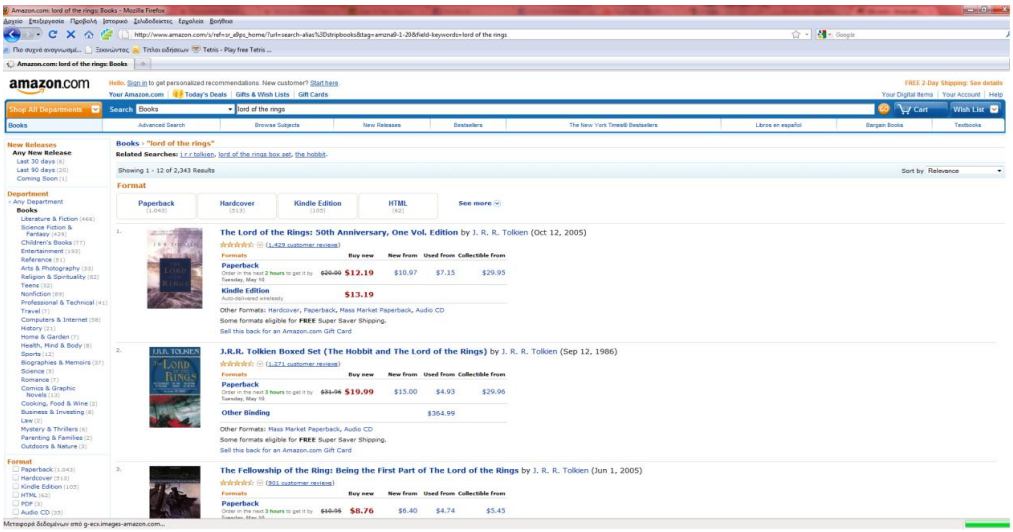
Ένα από τα σημαντικότερα εναύσματα για την χρήση του Ajax ήταν η ενσωμάτωσή της από την Google όπου εκεί βρήκε εφαρμογή στην σελίδα αναζήτησής της, στο ηλεκτρονικό της ταχυδρομείο (g mail) αλλά και στους χάρτες της (google maps) [3].



Εικόνα 2: Google search- Google suggest

Στις παρακάτω εικόνες φαίνεται μια στατική σελίδα αναζήτησης. Η αναζήτηση από την σελίδα www.a9.com είναι στατική και μας μεταφέρει μετά από έκλιση στον εξυπηρετητή της Amazon στην www.amazon.com.





Εικόνα 3: Μία στατική σελίδα αναζήτησης www.a9.com - Amazon

1.4 Οι βασικές αρχές του Ajax

Το Ajax βρίσκεται ακόμα σε πρώιμο στάδιο ως μοντέλο ανάπτυξης εφαρμογών. Ωστόσο αρκετοί προγραμματιστές έχουν αποδεχτεί την πρόκληση και προσπαθούν να προσδιορίσουν ποια είναι τα χαρακτηριστικά εκείνα που ξεχωρίζουν μία καλή από μία

κακή ή από μία μέτρια δικτυακή εφαρμογή. Οι κατά κοινή ομολογία βασικότερες αρχές των πιο αξιόλογων Ajax εφαρμογών είναι οι παρακάτω [2]:

- **Ελαχιστοποίηση του traffic**: Οι εφαρμογές σε Ajax θα πρέπει να στέλνουν και να λαμβάνουν από και προς τον server όσο το δυνατό λιγότερη πληροφορία. Εν συντομία, το Ajax μπορεί να ελαχιστοποιήσει την ποσότητα του traffic ανάμεσα στον client και στον server, και μία εφαρμογή γίνεται σίγουρα πολύ πιο συμπαγής εφόσον ο developer βεβαιωθεί ότι αυτή δεν στέλνει ούτε λαμβάνει άχρηστη πληροφορία.

- **Συνέπεια:** Οι εφαρμογές σε Ajax συνήθως εισάγουν διαφορετικά μοντέλα αλληλεπίδρασης με το χρήστη συγκριτικά με τις παραδοσιακές web εφαρμογές. Σε αντίθεση με το σύνηθες δικτυακό πρότυπο όπου κλικάρουμε και περιμένουμε μερικές Ajax εφαρμογές χρησιμοποιούν άλλα user interface παραδείγματα όπως το drag-and-drop ή το double-clicking. Θα πρέπει λοιπόν σε μία αξιόλογη εφαρμογή να μένουμε συνεπείς σε οποιοδήποτε μοντέλο αλληλεπίδρασης με το χρήστη επιλέξουμε έτσι ώστε ο χρήστης να γνωρίζει τι να κάνει στη συνέχεια.

- **Καθιερωμένες συμβάσεις:** Μη σπαταλάτε χρόνο προσπαθώντας να εφεύρετε νέα μοντέλα αλληλεπίδρασης με τα οποία ο χρήστης δεν θα είναι εξοικειωμένος. Δανειστείτε μοντέλα κυρίως από παραδοσιακές web και desktop εφαρμογές, έτσι ώστε να υπάρχει ελάχιστοποίηση της μαθησιακής δυσκολίας.

- **Όχι περισπασμοί:** Αποφύγετε όσα page elements δεν είναι απαραίτητα και αποσπούν την προσοχή όπως τα επαναλαμβανόμενα animations και τα τμήματα σελίδων που αναβοσβήνουν. Τέτοιοι μηχανισμοί αποσπούν την προσοχή του χρήστη από ότι προσπαθεί να επιτύχει.

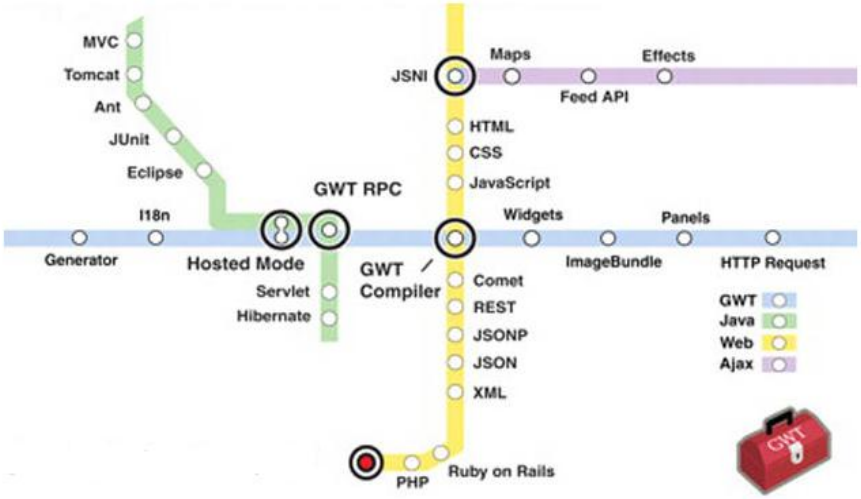
- **Προσβασιμότητα:** Θα πρέπει να εξετάζουμε από την αρχή ποιοι θα είναι οι πρωτεύοντες και ποιοι οι δευτερεύοντες χρήστες και με ποιο τρόπο είναι πιθανότερο αυτοί να αποκτούν πρόσβαση στην Ajax εφαρμογή. Θα πρέπει

επομένως κατά το σχεδιασμό της εφαρμογής να λαμβάνουμε υπόψη ότι ορισμένοι χρήστες μπορεί να χρησιμοποιούν παλαιότερους φυλλομετρητές ή εξειδικευμένο λογισμικό.

- **Αποφεύγουμε το κατέβασμα ολόκληρης σελίδας:** Ολόκληρη την επικοινωνία με τον server μετά από το κατέβασμα της αρχικής σελίδας θα πρέπει να τη διαχειρίζεται ο μηχανισμός του Ajax.

Το σημείο αναφοράς όλων των παραπάνω χαρακτηριστικών είναι η ευκολία στη χρήση. Ο βασικός ρόλος του Ajax είναι να βελτιωθεί η δικτυακή εμπειρία για τους χρήστες και αυτό θα πρέπει να το θυμόμαστε συνεχώς κατά την ανάπτυξη τέτοιων εφαρμογών.

Google Web Toolkit



2.1 Εισαγωγή στο GWT

Η ανάπτυξη ενός μεγάλου πλήθους από καινούρια toolkits και βιβλιοθήκες έχει βοηθήσει τον τελευταίο καιρό τους developers να απαλλαχθούν από την ανάγκη να γνωρίζουν τις ιδιαιτερότητες όλων των browsers για να φέρουν σε πέρας την κατασκευή και τη διαχείριση μεγάλων projects σε Ajax (Asynchronous JavaScript and XML). Ως γνωστόν το Javascript μπορεί να απαιτεί εξαιρετικά εξειδικευμένη γνώση και πειθαρχία ενώ οι browsers έχουν πολλές διαφορετικές υλοποιήσεις και σύνολα χαρακτηριστικών. Την ίδια στιγμή το tooling δεν φαίνεται να είναι αρκετά ώριμο και το debugging σε πολλαπλά διαφορετικά περιβάλλοντα είναι προβληματικό. Ένα πλήθος από βιβλιοθήκες όπως οι Script.aculo.as, Ext JS και η Yahoo User Interface Library αλλά και από projects όπως το Direct Web Remoting (DWR) και τεχνικές όπως αυτές που χρησιμοποιούνται από τα XML11, Echo2 συνεισφέρουν στην αντιμετώπιση ορισμένων από τα παραπάνω προβλήματα [4].

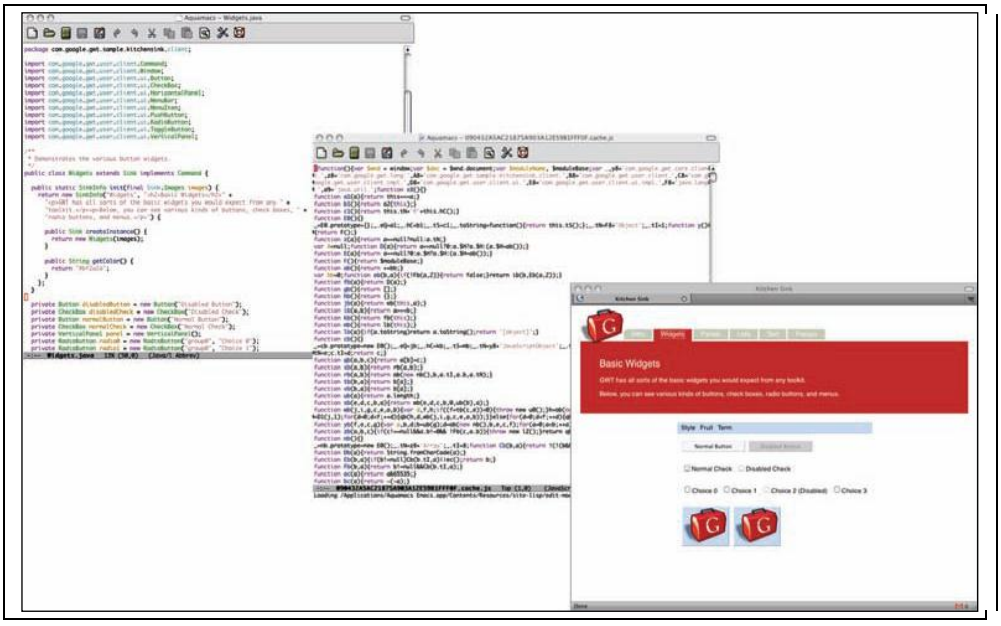
Τον Μάϊο του 2006 στο ετήσιο JavaOne συνέδριο στο San Franscisco παρουσιάστηκε το Google Web Toolkit (GWT), ένα εργαλείο πίσω από το οποίο η βασική ιδέα ήταν απλή˙ να γίνει ευκολότερη η ανάπτυξη σε Ajax κρύβοντας από τον προγραμματιστή τις ασυμβατότητες του browser και επιτρέποντας στον developer να εργαστεί σε ένα οικείο, περιβάλλον ανάπτυξης σε Java. Από τότε η GWT τεχνολογία έχει χρησιμοποιηθεί σε πλήθος διαφορετικών εφαρμογών, από gaming μέχρι mortgage calculators και το gwtPowered community site απαριθμεί περισσότερα από 130 παραδείγματα, άρθρα, widgets και άλλους πόρους [5].

Το Google Web Toolkit (GWT) είναι ένα framework ανοικτού κώδικα για web development που επιτρέπει την εύκολη δημιουργία AJAX εφαρμογών υψηλής απόδοσης με χρήση της Java. Μόλις ολοκληρωθεί η συγγραφή του κώδικα σε Java, ο GWT μεταγλωττιστής μεταφράζει τον πηγαίο κώδικα της Java σε βελτιστοποιημένη και συμβατή με τον browser HTML και JavaScript. Ο compiler δημιουργεί browser-specific

HTML και JavaScript για να μπορεί να υποστηρίζει ορθά όλους τους βασικούς browsers. Το GWT υποστηρίζει ένα συγκεκριμένο σύνολο από UI (user interface) widgets, το back button του browser και ένα test framework βασισμένο στο Junit.

Από πολλούς υποστηρίζεται ότι το Google Web Toolkit είναι το καλύτερο όσον αφορά την ανάπτυξη web 2.0 εφαρμογών. Ουσιαστικά πρόκειται για ένα Java Framework που κρύβει τον περίπλοκο κώδικα που χρειάζεται για την δημιουργία σύνθετων και δυναμικών συστατικών που μπορούν να συνδυαστούν και να δημιουργήσουν εντυπωσιακές και εύχρηστες οθόνες. Το συγκριτικό πλεονέκτημά του είναι ότι το GWT δημιουργεί τα συστατικά αυτά στο "Compile" εν αντιθέσει με άλλα frameworks που δημιουργούν τα δυναμικά συστατικά κάθε φορά που τα ζητάει ο χρήστης, οπότε και έχουν περισσότερες απαιτήσεις. Εκτός απ' το γεγονός ότι το GWT οδηγεί στην εφαρμογή του γνωστού MVC, παρέχει επιπροσθέτως plugin για το Eclipse και δίνει τη δυνατότητα εφαρμογής obsfunction στον compiled κώδικα, δηλαδή στις HTML και τα JavaScript ώστε να μην μπορεί εύκολα κάποιος κακόβουλος να καταλάβει πως λειτουργεί το σύστημα και το 'ρίξει'.

Στην Εικόνα 2.1 απεικονίζεται με σαφήνεια ο τρόπος λειτουργίας του GWT που παίρνει τον κώδικα σε Java και τον μεταγλωττίζει σε JavaScript για να τρέξει στον browser [4].



Εικόνα 2.1 Συνολική παρουσίαση της προσέγγισης του Google Web Toolkit. Ο πηγαίος κώδικας σε Java μεταγλωτίζεται σε JavaScript, και στη συνέχεια τρέχει σε έναν web browser όπως JavaScript/HTML/CSS.

Υπάρχουν πολλοί λόγοι για τους οποίους το GWT κατασκευάστηκε ώστε να χρησιμοποιεί αρχικά τη «στιβαρή» αυτή γλώσσα προγραμματισμού -τη Java- η οποία παρέχει υψηλές δυνατότητες υποστήριξης για tooling και testing και στη συνέχεια να «παράγει» διαφορετικές versions JavaScript εφαρμογών για όλους τους σημαντικούς browsers σε ένα μόνο βήμα του compilation. Ο βασικότερος από αυτούς είναι με τα JavaScript, ξεκινώντας από μία βάση όπου έχουμε μόνο τον κώδικα και παράγοντας όλες τις απαιτούμενες παραλλαγές, διευκολύνουμε πάρα πολύ τόσο τον developer όσο και τον χρήστη.

Άλλα χαρακτηριστικά που κάνουν το GWT να ξεχωρίζει είναι μία ρουτίνα για να πραγματοποιείται άμεσο debugging του Java byte code καθώς εκετελείται σε ένα προσομοιωμένο περιβάλλον browser, ένα σύνολο από core UI και layout widgets με τα οποία χτίζονται διάφορες εφαρμογές, ένα RPC (Remote Procedure Call) σύστημα για τη διαχείριση της επικοινωνίας με έναν host web server, λειτουργίες για υποστήριξη

διεθνοποίησης (στο εξής αυτό θα το αποκαλούμε με την ορολογία ‘internationalization support’), και ποικίλους μηχανισμούς για testing [4].

Το Google Web Toolkit παρέχει επιπλέον μία πλατφόρμα για τη δημιουργία πραγματικών “Rich” εφαρμογών διαδικτύου (Rich Internet Applications RIA’s). Ο όρος «rich» χρησιμοποιείται για να περιγράψει το γεγονός κατά το οποίο στον client επιτρέπεται να διατηρήσει την κατάστασή του, αλλά και να εκτελέσει τοπικά υπολογισμούς, με ένα πλήρες data model και χωρίς να απαιτείται η μετάβαση στον server για κάθε update στο interface. Τα πλεονεκτήματα του παραπάνω γεγονότος είναι πολλά τόσο για το χρήστη όσο και για τον developer. Ο χρήστης έχει με αυτό τον τρόπο μία πολύ πιο άμεσα ανταποκρινόμενη εφαρμογή, ενώ ο developer είναι σε θέση να κατανείμει το φορτίο της εφαρμογής. Το GWT παρέχει επίσης μία «πλούσια» πλατφόρμα όσον αφορά τα User Interface elements και τις δυνατότητες που παρέχει: sliders, reflections, drag-and-drop υποστήριξη, suggest boxes, data-bound πίνακες και άλλα. Αυτή η «πλούσια» client πλατφόρμα, η οποία στην ουσία χρησιμοποιεί μόνο HTML, JavaScript και Cascading Style Sheets (CSS) έχει πλήρη πρόσβαση με διάφορους τρόπους σε back-end server πόρους όποτε απαιτείται [4].

Το GWT παρέχει δύο διαφορετικά modes:

- Το Development Mode, το οποίο επιτρέπει στο χρήστη να κάνει debugging στον κώδικά (Java) της εφαρμογής του απευθείας μέσω του στάνταρ Java debugger.

- Το Web Mode, στο οποίο η εφαρμογή μεταφράζεται σε HTML και Javascript κώδικα και μπορεί να αναπτυχθεί σε έναν web server.

2.2 Η Δομή του Google Web Toolkit

Το GWT παρέχει ένα πλήθος από τεχνολογίες για την κατασκευή Ajax εφαρμογών: τον GWT μεταγλττιστή, ένα User Interface επίπεδο, ένα RPC σύστημα,

ορισμένες πρόσθετες ιδιότητες που κάνουν ευκολότερα διαχειρίσιμες τις διαβαθμίσεις στο δίκτυο, και το GWT shell. Στη συνέχεια αυτής της ενότητας θα εξετάσουμε αναλυτικά τις παραπάνω εφαρμογές και τον τρόπο που αυτές χρησιμοποιούνται. Στον πυρήνα του GWT βρίσκεται ο compiler του [4].

2.2.1 Ο GWT Compiler

Ο GWT Java compiler παίρνει κώδικα σε Java και τον μεταγλωττίζει σε JavaScript κάνοντας χρήση ορισμένων εξειδικευμένων κανόνων. Με τον ορισμό των compilation tasks του GWT ως modules, ο μεταγλωττιστής είναι σε θέση να εκτελεί περισσότερη ανάλυση στον κώδικα καθώς αυτός υφίσταται επεξεργασία, και να δημιουργεί πολλαπλά compilation artifacts για διαφορετικούς output targets. Αυτό σημαίνει, ότι καθώς γίνεται compile μία κλάση, ο προγραμματιστής μπορεί να καθορίσει διαφοροποιημένες υλοποιήσεις βάσει γνωστών παραμέτρων, όπου το προφανές switch point είναι ο agent ή client browser του χρήστη στον οποίο στοχεύει. Αυτό είναι άλλωστε το χαρακτηριστικό που προσδίδει στον πυρήνα του GWT συμβατότητα με όλους τους browsers.

Σε αντίθεση με τα περισσότερα Ajax toolkits που είναι γραμμένα σε JavaScript και τα οποία χρησιμοποιούν πολύπλοκη λογική για να προσαρμοστούν σε διαφορετικά περιβάλλοντα browsers, το GWT θα χρησιμοποιήσει την υλοποίηση των core classes που βασίζονται στον user agent για τον οποίο προορίζεται το compilation task. Εν συντομία κάθε browser λάμβάνει μία συγκεκριμένη, ισχυρή version της εκάστοτε εφαρμογής και δεν είναι αναγκασμένος να κατεβάζει κάθε φορά κώδικα για όλους τους browsers που μπορεί να υποστηρίξει η εφαρμογή. Μέσω αυτού του μηχανισμού το GWT υλοποιεί ένα UI Toolkit για όλους τους browsers [4].

2.2.2 Το User Interface επίπεδο

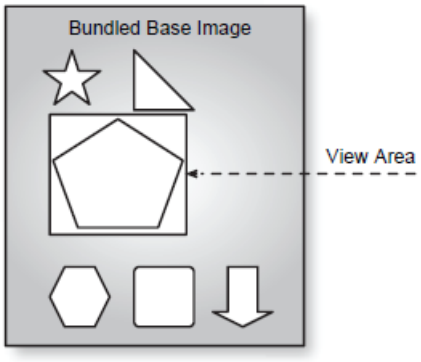
Χτισμένο ουσιαστικά πάνω στο ευφυές compilation σύστημα του GWT το User Interface Layer ανταπεξέρχεται στις ανάγκες όλων των browsers. Αυτό που κάνει τη

διαφορά στη συγκεκριμένη περίπτωση είναι η υλοποίηση των UI συστατικών στοιχείων σε Java και στη συνέχεια η χρησιμοποίηση μίας browser-specific υλοποίησης του πυρήνα DOM για την κατασκευή των επιμέρους συστατικών στοιχείων κάθε browser όπως αυτά θα ζητηθούν από κάποιο ανώτερο Java επίπεδο. Ενώ ορισμένες βιβλιοθήκες Ajax δίνουν περισσότερη σημασία σε UI widgets, το GWT στοχεύει στην παροχή ενός πυρήνα από UI λειτουργικότητες πάνω στις οποίες μπορούν να στηριχτούν οι χρήστες και η κοινότητα.

Το GWT User Interface Layer παρέχει μία ευρεία ποικιλία από layout-related panels, από constructs για αναπαράσταση δεδομένων (όπως είναι το Tree και το Grid), ένα σύνολο από user input στοιχεία και πολλά άλλα. Η 1.4 έκδοση του GWT άρχισε να επεκτείνει το UI Toolkit ώστε να περιλαμβάνει ορισμένα νέα, εξελιγμένα στοιχεία, όπως έναν πλούσιο text editor και ένα suggest box. Η ίδια έκδοση ξεκίνησε επίσης να ενσωματώνει ορισμένα εξαιρετικά βελτιωμένα UI elements τα οποία στηρίζονται στον πρωτοποριακό μεταγλωττιστή του GWT, όπως το ImageBundle.

Το Image Bundle παίρνει τις στατικές εικόνες από μία εφαρμογή και τις συνενώνει σε ένα ενιαίο graphic file. Κατόπιν, χρησιμοποιώντας το τοποθετημένο background mode στο CSS box μοντέλο, εμφανίζει μόνο το κομμάτι της μεγάλης ενιαίας εικόνας που απαιτείται σε οποιοδήποτε σημείο, όπως φαίνεται άλλωστε στην παρακάτω Εικόνα 2.2. Αυτό σημαίνει ότι ο client browser μπορεί να κατασκευάσει ένα μόνο request για να πάρει όλες τις εικόνες στην εφαρμογή, και δε χρειάζεται να διαπραγματεύτεαι για πολλαπλούς HTTP request-response cycles, βελτιώνοντας κατ΄ αυτόν τον τρόπο τον χρόνο έναρξης της εφαρμογής.

Επιπροσθέτως στο core UI και στο υποσύνολο των UI συστατικών στοιχείων που παρέχονται, το GWT περιλαμβάνει επιπλέον ορισμένα μέσα επικοινωνίας με τους πόρους του server. Ο βασικότερος από αυτούς τους μηχανισμούς είναι ο GWT RPC [4].



Εικόνα 2.2 Το ImageBundle ενώνει πολλές εικόνες σε μία μεγάλη ενιαία και στη συνέχεια επιστρέφει μία μοναδική εικόνα στη σελίδα τοποθετώντας την compiled εικόνα στο background.

2.2.3 Απομακρυσμένες κλήσεις διεργασίας (Remote Procedure Calls)

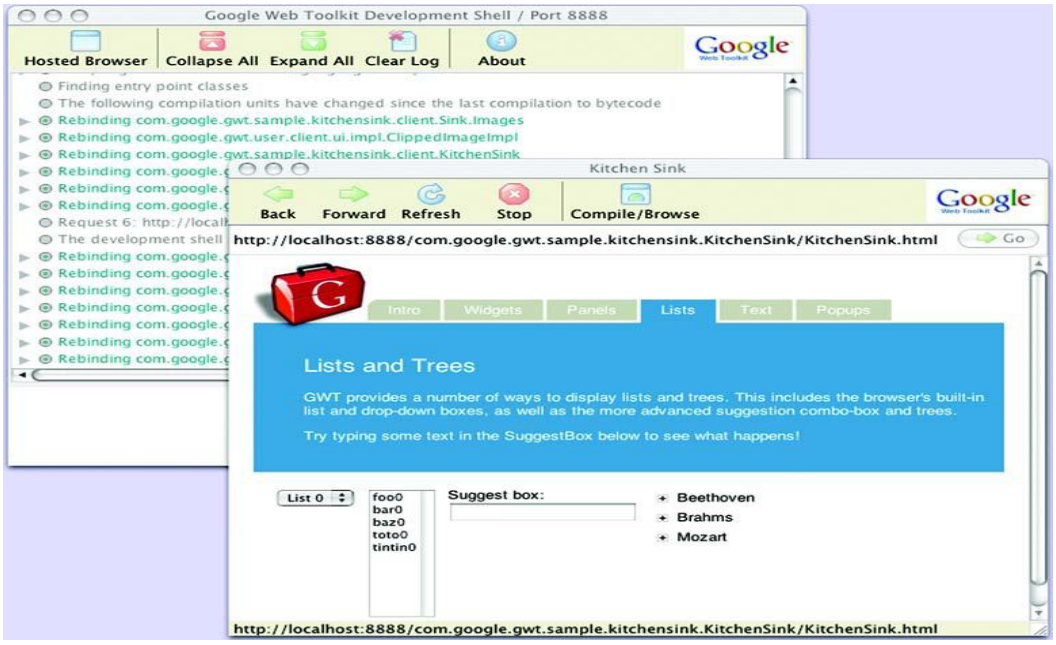
Ένα άλλο σημαντικό χαρακτηριστικό του GWT που σχετίζεται άμεσα με τις plugin δυνατότητες του compiler είναι η RPC (Remote Procedure Call) λειτουργία. Το σύστημα επιτρέπει serialization (σειριοποίηση) και deserialization του Java αντικειμένου από υλοποιήσεις απομακρυσμένων υπηρεσιών που έχουν γίνει από τη μεριά του server, οι οποίες μπορούν στη συνέχεια να καλούνται ασύγχρονα από τον client.

Για να πραγματοποιήσει τα παραπάνω, ο compiler παράγει κώδικα κατά τη διάρκεια της μεταγλώττισης για να χειριστεί το serialization σε χαμηλό επίπεδο. Κατά τη διάρκεια του compile δημιουργείται καινούρια version των serialized αντικειμένων, γεγονός που έχει δύο πλεονεκτήματα. Πρώτον, η συμφωνία client και server είναι εγγυημένη από τη στιγμή που εφαρμόζονται καινούριες versions και δεύτερον η υλοποίηση του server μπορεί να ‘συμπιέσει’ την κατάσταση των Java αντικειμένων σε arrays από JavaScript primitives. Αυτό το πέρασμα απλών array επιτρέπει τη διαμόρφωση περισσότερο περιεκτικών δεδομένων απ’ ότι το JavaScript Object Notation (JSON), το οποίο διάφοροι εξυμνούν για την απλότητα και την αποδοτικότητά του. Εκτός από το δικό του RPC σύστημα, το GWT περιλαμβάνει επίσης ένα σύνολο από πρόσθετες ωφέλειες οι οποίες κάνουν το web development απλούστερο [4].

2.2.4 GWT Shell

Όλα τα σημαντικά χαρακτηριστικά που περιλαμβάνει το GWT είναι χτισμένα στον πυρήνα της αρχιτεκτονικής προσέγγισης και του GWT compiler. Πέραν τούτου όμως το GWT περιλαμβάνει επιπροσθέτως ένα εξαιρετικό σύνολο από development εργαλεία ξεκινώντας από το GWT shell και τον hosted mode browser που φαίνονται στην Εικόνα 2.3.

Το GWT shell επιτρέπει το τεστάρισμα της εφαρμογής σε έναν browser καθώς εκτελείται ο native Java byte code. Αυτό παρέχει στον developer τη δυνατότητα χρησιμοποίησης όλων των αγαπημένων του Java tools έτσι ώστε να επιβλέπει την εκτέλεση της εφαρμογής, συμπεριλαμβανομένων των profilers, του debugging ανά βήμα και JTI-based monitors. Αυτός ο hosted mode browser σε συνδιασμό με τον ενσωματωμένο Apache Tomcat server είναι επίσης χαρακτηριστικό στοιχείο που επιτρέπει το τεστάρισμα του compiled JavaScript με JUnit. Καταλαβαίνουμε λοιπόν ότι το κέλυφος είναι εξαιρετικά σημαντικό για όλα τα project με GWT development [4].



Εικόνα 2.3 Το GWT shell και ο hosted mode browser. Το κέλυφος περιλαμβάνει ένα ιεραρχικό log view και έναν custom web browser.

2.2.4 Πρόσθετες λειτουργίες

Το GWT περιλαμβάνει ένα πλήθος από ωφέλιμα χαρακτηριστικά τα οποία είναι σχεδιασμένα με τέτοιο τρόπο ώστε να διευκολύνουν τη δημιουργία εφαρμογών για διαφορετικά web επίπεδα. Σε αυτά περιλαμβάνονται εργαλεία για διεθνοποίηση, εργαλεία για υποστήριξη επιπρόσθετων μεθόδων για επικοινωνία με τους servers, ένα history management σύστημα, καθώς και testing support.

Το GWT παρέχει επίσης κάποιες client-side βιβλιοθήκες ένα βήμα παραπέρα από τις GWT RPC και επικοινωνία με XML και JSON-based υπηρεσίες. Με αυτό τον τρόπο γίνεται ευκολότερη είτε η ενσωμάτωση πολλών υπαρχόντων web APIs είτε η υλοποίηση ενός πλήρους backend που δεν είναι Java-based.

Υπάρχει επίσης μία compile-time-checked internationalization βιβλιοθήκη που κάνει ευκολότερη και πιο αξιόπιστη την παροχή υποστήριξης πολλών γλωσσών. Το history management σύστημα από την άλλη διευκολύνει με τη σειρά του το bookmarking

και το deep-linking στις Ajax εφαρμογές. Το σύστημα αυτό μπορεί επίσης να υπερφορτωθεί όταν ο χρήστης χρειάζεται να συνδεθεί μεταξύ αρκετών deployed GWT εφαρμογών.

Τέλος, παρέχεται τυπική υποστήριξη testing. Το GWT παρέχει στο χρήστη τα μέσα να τεστάρει τον κώδικά του γράφοντας test cases είτε ως Java είτε ως JavaScript. Όλες οι πρόσθετες λειτουργίες που παρέχονται από το GWT στοχέυουν στο να διευκολύνουν τον κύκλο του development και να τον κάνουν περισσότερο προβλέψιμο στο web tier [4].

2.3 Οι βασικές έννοιες του GWT

Τα GWT projects συνθέτονται από κάποια βασικά κομμάτια. Στη συνέχεια θα εξετάσουμε τα βασικά χαρακτηριστικά ενός τέτοιου project για να οικειοποιηθούμε με τις βασικές έννοιες που χρησιμοποιούνται κατά την εργασία με το toolkit.

Αρχικά, τα GWT projects ορίζονται με τους όρους κάποιων modules, που συνθέτονται από resources, configuration και source. Το configuration του module προσδιορίζει πληροφορία σχετικά με τον χρόνο του compile ενός project και προσδιορίζει τους πόρους που χρειάζονται κατά το χρόνο εκτέλεσης.Πέρα από το configuration, τα modules καθιστούν δυνατή τη λειτουργία ενός μηχανισμού έντονης κληρονομικότητας (rich inheritance mechanism). Εξαιτίας αυτής της δυνατότητας, τα διάφορα projects μπορούν να είναι ολοκληρωμένες web εφαρμογές, μπορεί να έχουν κάποια library nature ή να κινούνται ανάμεσα στα δύο.

Ένα πράγμα που προσδιορίζεται οπωσδήποτε από κάθε module είναι το σημείο έναρξης του κώδικα του project, γνωστό ως entry point.Οι classes των entry points γράφονται σε Java, δηλώνονται από έναν ορισμό του module και στη συνέχεια μεταγλωττίζονται σε JavaScript. Τα ίδια τα modules μαζί με τα entry points που ορίζουν

ξεκινούν να λειτουργούν μέσω ενός <script> reference σε μία HTML σελίδα, γνωστή ως host page. Οι host pages θέτουν σε λειτουργία, ενεργοποιούν τα GWT projects και επίσης υποστηρίζουν μερικά ειδικά <meta tags>. Σε υψηλό επίπεδο αυτά είναι τα τρία σημαντικότερα συστατικά στοιχεία ενός GWT project: ένα module configuration, μία entry point class και μία HTML host page. Στη συνέχεια θα εξετάσουμε τη λειτουργία και των τριών αναλυτικά [4].

2.3.1 Modules και κληρονομικότητα

Καθώς εξετάζουμε το GWT ορισμένες πλευρές του φαίνονται εντελώς ξένες. Η βασικότερη από αυτές είναι η χρήση της έννοιας του module κατά τον προσδιορισμό των GWT εφαρμογών. Ένα από τα σημαντικότερα προβλήματα του web development με την πάροδο των ετών έχει υπάρξει η πραγματική επαναχρησιμοποίηση από εφαρμογές και συστατικά στοιχεία. Το GWT σύστημα των modules επιτρέπει στα συστατικά στοιχεία μιας εφαρμογής να ενώσει σε ένα πακέτο που εύκολα επαναδιανέμεται τον client-side κώδικα της εφαρμογής, τις server-side service υλοποιήσεις και πόρους όπως τα γραφικά και τα CSS αρχεία. Τα modules με την διευκόλυνση για inheritance που παρέχουν αποτελούν επίσης, ένα σημαντικό κομμάτι του GWT bootstrap και της διαδικασίας του deployment.

Ένα module ορίζεται από ένα XML αρχείο μαζί με τον πηγαίο κώδικα σε Java που υλοποιεί το module. Αυτό το XML αρχείο δηλώνει ορισμένα πρωταρχικά στοιχεία: inherited modules, servlet deployments, compiler plug-ins και entry points. Η Εικόνα 2.4 δείχνει ένα πρότυπο ενός πολύ απλού module file [4].

Listing 1.1 Calculator.gwt.xml module definition

```
<module>
   <inherits name='com.google.gwt.user.User'/>
   <entry-point
      class='com.manning.gwtip.calculator.client.Calculator'/>
   <stylesheet
      src='com.manning.gwtip.calculator.style.css'/>
</module>
```

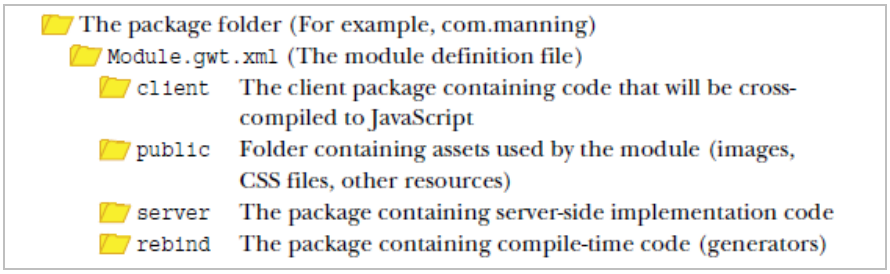
1 Inherit core GWT classes
2 Define EntryPoint
3 Specify CSS file to use

Εικόνα 2.4 Παράδειγμα ενός module file όπου φαίνονται καθαρά τα συστατικά του στοιχεία.

Το <inherits> tag λέει στο GWT να κάνει inherit το core User module b. Αυτό περιλαμβάνει τα GWT UI elements καθώς και τα custom compiler elements που παράγουν την κατάλληλη έκδοση για τους Firefox, Safari, Internet Explorer και Opera. Θα πρέπει να παρατηρήσουμε ότι όταν ένα module γίνεται inherited φέρνει μαζί του όλα τα συστατικά στοιχεία του module και όχι απλώς τον ποηγαίο κώδικα. Αυτός είναι και ο βασικός λόγος ύπαρξης του module system. Ο compiler μπορεί να βρει πρόσθετα Java sources που αναφέρονται από το project, αλλά το module system επιτρέπει να γίνονται inherited τόσο το build-time όσο και η server-side behavior. Στη συνέχεια ακολουθεί η δήλωση ενός <entry-point> c. Όπως αναφέραμε και πριν, ένα entry-point είναι το σημείο έναρξης από το οποίο ξεκινάει ο κώδικας σε ένα GWT project.

Στο τέλος, περιλαμβάνεται και ένα stylesheet για την εφαρμογή d. Αν και δεν είναι απαραίτητη η παροχή style πληροφορίας σε αυτό το επίπεδο, το stylesheet που δηλώνεται στο module το ακολουθεί κιόλας μέσω της κληρονομικότητας, κάνοντας ευκολότερο το κοινό formatting για τη μεταφορά από εφαρμογή σε εφαρμογή καθώς επαναχρησιμοποιούνται τα κοινά components.

Αυτός ο ορισμός του module κάνει ορισμένες βασικές υποθέσεις, η βασικότερη από τις οποίες είναι η δομή του directory των source folders. Αν και η δομή αυτή προσαρμόζεται, η default συμβατική δομή/διάταξη για GWT εφαρμογές ακολουθεί το format που φαίνεται στην Εικόνα 2.5 [4].



Εικόνα 2.5 Η default διάταξη των folders μιας GWT εφαρμογής

Στο προηγούμενο παράδειγμα, το reference στο com.manning.gwtip.calculator.style.css αναφέρεται στο com.manning.gwtip.calculator.style.css αρχείο που είναι στον public φάκελο του module package. Βλέπουμε επίσης ότι ο ορισμός του <entry-point> αναφέρει μία κλάση στο client-package.

2.3.2 Host pages

Για να τρέξουμε μία GWT εφαρμογή ξεκινάμε με μία HTML σελίδα. Η HTML host page όπως θα τη λέμε από δω και στο εξής, περιέχει απλώς κάποια elements που απαιτούνται για την ενεργοποίηση της εφαρμογής. Πρώτη είναι μία αναφορά σε ένα nocache JavaScript αρχείο που παράγει το GWT για κάθε module. Αυτό είναι ένα JavaScript αρχείο που ανιχνέυει την κατάλληλη version του application module, βάσει του user agent και άλλων παραγόντων και φορτώνει τελικά την compiled application. Η Εικόνα 2.6 δείχνει ένα παράδειγμα μίας HTML host page.

Το σημαντικό σε αυτό το σημείο είναι ότι το host page περιλαμβάνει το bootstrap JavaScript file που αρχικοποιεί την εφαρμογή b. Όταν μεταγλωττίζεται μία GWT application προκύπτουν δύο versions αυτού του αρχείου, που ονομάζονται [ModuleName].nocache.js, και [ModuleName]-xs.nocache.js, αντίστοιχα. Αυτά τα δύο αρχεία τυπικά αντιπροσωπεύουν το μόνο κομμάτι της GWT application που θα πρέπει να σταλεί στον client χωρίς να ενεργοποιηθεί πρώτα HTTP-level caching. Όλα τα υπόλοιπα αρχεία που παράγει το GWT ονομάζονται βάσει μίας παραλλαγής του πηγαίου κώδικα που τα δημιούργησε, γεγονός που σημαίνει ότι μπορούν να γίνουν cached αόριστα από τον client χωρίς άλλες παρενέργειες. Τα δύο nocache αρχεία καθορίζουν ποια παραλλαγμένη έκδοση χρειάζεται να φορτωθεί όταν ζητηθεί η σελίδα.

Η cross-site xs version του nocache αρχείου μπορεί να χρησιμοποιηθεί από άλλα domains που μπορεί να περιλαμβάνουν την εφαρμογή. Το αρχείο αυτό επιτρέπει στα

remote domains να επικοινωνήσουν με τα server-side resources, ακόμη και αν η HTML page που περιλαμβάνει το script δεν εξυπηρετήθηκε από το ίδιο host με τα server-side resources. Αυτή η τακτική της ίδιας προέλευσης όμως είναι αρκετά επικίνδυνη, αφού καθιστά την εφαρμογή ανοικτή σε πιθανά Cross Site Scripting (XSS) προβλήματα ασφαλείας. Κατά συνέπεια η χρήση αυτού του script θα πρέπει να γίνε αι με εξαιρετικά μεγάλη προσοχή. Κάθε φορά που ένα module nocache script (είτε είναι version-standard είτε είναι cross-site) φορτώνεται από μία host page, το υπόλοιπο της GWT bootstrap process μπαίνει σε λειτουργία και φορτώνεται μία entry-point κλάση [4].

Listing 1.2 The calculator HTML host page

```
<html>
    <head>
        <title>Wrapper HTML for Calculator</title>
    </head>
    <body>
        <script language="javascript"
            src=
"com.manning.gwtip.calculator.Calculator.nocache.js"
            >
            </script>
        <iframe id="__gwt_historyFrame"
                style="width:0;height:0;border:0"></iframe>
        <h1>Calculator</h1>

    </body>
</html>
```

1 Bootstrap JavaScript file

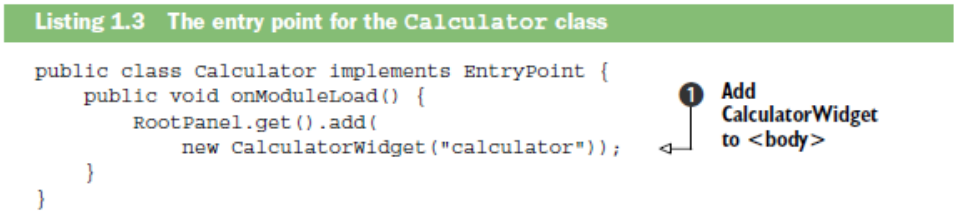
Special invisible iframe for history management

Εικόνα 2.6 Παράδειγμα μίας HTML host page

2.3.3 Entry-point classes

Μία entry point κλάση είναι μία απλή client-side κλάση που υλοποιεί τη διεπαφή com.google.gwt.core.client.EntryPoint η οποία ορίζει μία μόνο μέθοδο: την onModuleLoad (). Αυτό είναι το GWT ισοδύναμο μίας κλάσης με τη μέθοδο public static void main (), όπως δηλώνεται στη Java. Η Εικόνα 2.7 δείχνει ένα παράδειγμα μίας EntryPoint υλοποίησης.

Η onModuleLoad () μέθοδος είναι απλή αλλά το λειτουργικό χαρακτηριστικό εδώ είναι η χρήση της κλάσης GWT's RootPanel. Αυτή αποτελεί το μέσο πρόσβασης της εφαρμογής στην HTML host page. Η CallingRootPanel.get() επιστρέφει ένα default container που εισάγεται ακριβώς πριν από το τελικό tag (</body>) στην HTML page b. Εάν απαιτείται μεγαλύτερος και πιο προσεκτικός έλεγχος στο σημείο όπου εισάγεται στη σελίδα ένα GWT widget, καλείται η RootPanel.get ("ElementID") όπου ElementID είναι μία string τιμή που δηλώνει το ID του στοιχείου στην HTML σελίδα, τυπικά δηλαδή ένα tag [4].



Εικόνα 2.7 Μία Entry Point υλοποίηση

2.4 Ο κύκλος ζωής μίας default GWT εφαρμογής

Στην ενότητα αυτή και αφού έχουμε εξετάσει τα βασικά χαρακτηριστικά του Google Web Toolkit και έχουμε αναλύσει τις βασικές του έννοιες θα προχωρήσουμε στην παρουσίαση ενός τυπικού κύκλου ανάπτυξης μίας GWT εφαρμογής [6].

Οι GWT εφαρμογές μπορούν να δημιουργηθούν με τρεις διαφορετικούς τρόπους. Ο πρώτος τρόπος είναι με τη χρησιμοποίηση του συνόλου των creation tools που παρέχονται από το GWT download. Τα εργαλεία αυτά επιτρέπουν την γρήγορη δημιουργία μίας δομής από directories και files κατάλληλης για την εκάστοτε GWT εφαρμογής, η οποία δομή είναι επίσης συμβατή με τη σομή που περιμένει ο GWT μεταγλωττιστής. Με χρήση των παραπάνω εργαλείων το αποτέλεσμα γίνεται ανεξάρτητο από οποιοδήποτε IDE μπορεί να χρησιμοποιήσουμε. Παρέχοντας επιπλέον το flag –

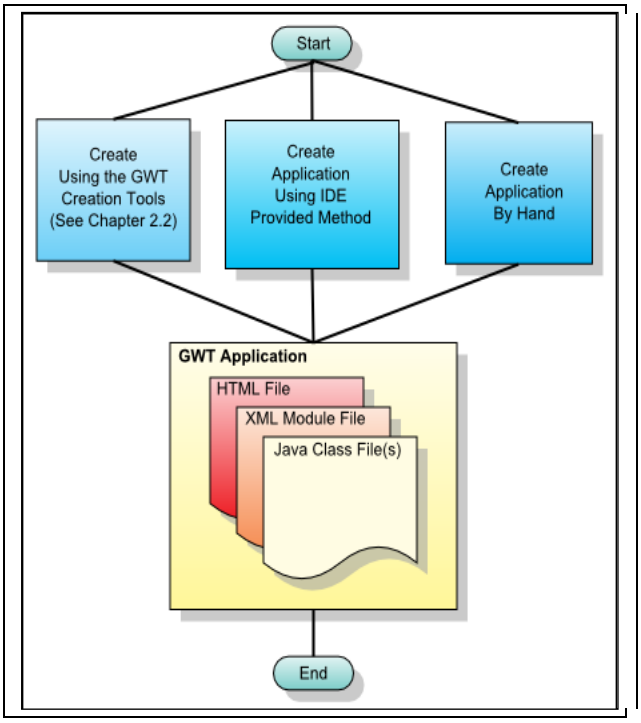
eclipse στα creation tools, τα οδηγούμε στη δημιουργία ενός πρόσθετου συνόλου από αρχεία που καθιστούν όλη τη δομή εύκολη κατά την εισαγωγή της στο Eclipse IDE (Για το GWT development απαιτείται εκτός από ένα copy της Java, για παράδειγμα το SunJDK 5.0, και ένα copy του Eclipse IDE).Το Eclipse δεν είναι το μόνο IDE που μπορεί να χρησιμοποιηθεί, απλώς το GWT διευκολύνει την ενσωμάτωση με το Eclipse, χωρίς τη χρήση ενός IDE specific wizard. Υπάρχει επίσης δυνατότητα να εισαχθούν τα Eclipse projects σε ορισμένα άλλα περισσότερα κοινότυπα IDEs.

Καθώς το GWT ωριμάζει, όλο και περισσότερα 3rd party tools βγαίνουν στην αγορά και καλύπτουν την ανάγκη χρησιμοποίησης των GWT creation tools. Σε ορισμένες περιπτώσεις αυτά τα 3rd party tools βγαίνουν σε μορφή wizards την οποία υποστηρίζουν τα IDEs, ή σε άλλες περιπτώσεις ολόκληρα καινούρια IDEs κατασκευάζονται ή παραλάσσονται ειδικά για να υποστηρίξουν τις ανάγκες ανάπτυξης GWT εφαρμογών. Σε γενικές γραμμές τα εργαλεία αυτά παρέχονται μαζί με οδηγίες για το πώς χρησιμοποιούνται αλλά αρκετά συχνά το αποτέλεσμα της εκτέλεσης ενός IDE specific wizard θα είναι μία default εφαρμογή παρόμοια με αυτή που παράγεται από τα GWT creation tools.

Εάν κάποιος δεν θέλει να χρησιμοποιήσει τα εργαλεία που παρέχονται από το GWT ή κάποιο εργαλείο που παρέχει ο εκάστοτε IDE, υπάρχει η δυνατότητα δημιουργίας νέας δομής των directories και των βασικών αρχείων. Αυτό συνηθίζεται σε περιβάλλοντα στα οποία οι περιορισμοί του συστήματος αποτρέπουν τη χρήση των συνηθισμένων δομών. Με αυτή τη προσέγγιση θα πρέπει να δημιουργηθούν υπεύθυνα και προσεκτικά η δομή του directory και το βασικό σύνολο των αρχείων και να γίνει λεπτομερής διαχείριση του Module XML αρχείου της εφαρμογής για να μάθει ο compiler όλα τα μονοπάτια προς τα απαραίτητα αρχεία. Το διάγραμμα της Εικόνας 2.8 συνοψίζει τις 3 διαφορετικές μεθόδους δημιουργίας.

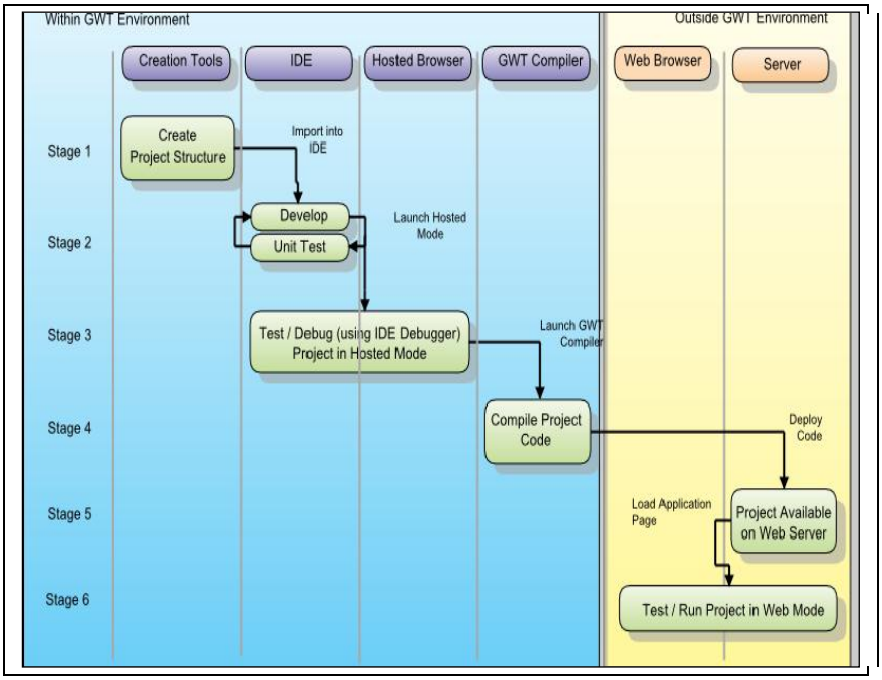
Οι τρεις αυτές προσεγγίσεις οδηγούν στην παραγωγή του ίδιου συνόλου από βασικά αρχεία που αναπαριστούν τη GWT default application, γεγονός το οποίο φαίνεται θετικό αφού όλα αυτά τα αρχεία είναι απαραίτητα για την ορθή λειτουργία της εφαρμογής. Αν δούμε αναλυτικότερα τη δεύτερη και την τρίτη προσέγγιση θα λέγαμε ότι πιθανότατα ο πιο ασφαλής τρόπος να δημιουργήσουμε τη δομή και τα αρχεία της GWT είναι να τα κάνουμε μόνοι μας από την αρχή.

Πρέπει επίσης να αναφερθεί ότι το GWT δεν είναι άρρηκτα συνδεδεμένο με το Eclipse, αφού τα αρχεία που παράγονται μπορούν να εισαχθούν σε οποιοδήποτε IDE και η εφαρμογή μπορεί να δημιουργηθεί ακόμα και από εργαλεία του command line. Το Eclipse, όπως οποιοδήποτε IDE, έχει θετικά και αρνητικά στοιχεία, είναι όμως δωρεάν και κυρίως είναι ευρέως χρησιμοποιούμενο [6].



Εικόνα 2.8 Οι τρεις διαφορετικοί τρόποι για τη δημιουργία μιας GWT εφαρμογής: χρησιμοποιώντας τα GWT tools, χρησιμοποιώντας ένα plug-in για το IDE και δημιουργώντας τη δομή και τα αρχεία από την αρχή.

Στην ενότητα αυτή θα εξετάσουμε μόνο τη δημιουργία εφαρμογών μέσω των GWT tools, τα οποία μαζί με το GWT Hosted Browser, το GWT Compiler και τον κατάλληλα επιλεγμένο web server και browser παρέχουν πλήρεις δυνατότητες ανάπτυξης και test environments. Στον Πίνακα 2.1 αναλύονται οι χρήσεις και τα οφέλη από κάθε εργαλείο ενώ στην Εικόνα 2.9 βλέπουμε μία απλουστευμένη προσέγγιση των σταδίων του κύκλου ζωής της ανάπτυξης μίας εφαρμογής στην οποία τα creation tools και τα υπόλοιπα tools που αναφέραμε χρησιμοποιούνται τυπικά [6].



Εικόνα 2.10 Ο γενικός κύκλος ζωής της ανάπτυξης μίας GWT εφαρμογής, όπου φαίνονται από επάνω ποια εργαλεία χρησιμοποιούνται και κάτω στο πλάϊ τα στάδια χρήσης. Ορισμένα από τα στάδια, για παράδειγμα το στάδιο 2, μπορεί να επαναληφθεί πολλές φορές προτού γίνει η μετάβαση στο επόμενο βήμα. Η προσέγγιση αυτή είναι waterfall και μπορεί να ακολουθηθεί μία περισσότερο ταχεία διαδικασία cycling για το application development ανάμεσα στα στάδια 2,3,4,5 και 6 όπως απαιτείται [6].

.

Στάδιο	Περιγραφή
1	Στο Στάδιο 1 εγκαθίσταται η δομή των φακέλων και του κώδικα του project. Αυτό μπορεί να εκτελεστεί χρησιμοποιώντας μία από τις τρεις προσεγγίσεις που προτείνονται στην Εικόνα 2.8 – χρησιμοποιώντας τα creation tools του GWT, ένα IDE specific wizard ή με το χέρι. Εάν χρησιμοποιηθούν τα GWT creation tools τότε δημιουργείται μία default εφαρμογή. Αρκετά συχνά, εάν χρησιμοποιηθούν είτε τα GWT creation tools είτε η μέθοδος με το χέρι, θα πρέπει μετά να εισάγουμε το project μας σε ένα Ολοκληρωμένο Περιβάλλον Ανάπτυξης (IDE) προτού ξεκινήσουμε κανονικά την ανάπτυξη. Εάν χρησιμοποιούμε τα GWT creation tools και το Eclipse IDE, τότε προσθέτοντας το eclipse flag στα εργαλεία του command line θα δημιουργήσουμε όλα τα απαραίτητα πρόσθετα αρχεία που απαιτούνται για την εύκολη εισαγωγή της δομής των φακέλων και του κώδικα στο Eclipse ώς ένα Eclipse project.
2	Μόλις δημιουργηθεί η δομή των φακέλων και του κώδικα τότε μπορεί να συνεχίσει η ανάπτυξη της εφαρμογής. Τυπικά σε αυτό το σημείο θα αντικαταστήσουμε τα default αρχεία που δημιούργησαν για εμάς τα GWT creation tools ή το IDE's wizard κι επίσης θα προσθέσουμε τα υπόλοιπα αρχεία που απαιτούνται για την εφαρμογή, για παράδειγμα πρόσθετες Java classes, Cascading Style Sheets, εικόνες κλπ.
3	Η περίοδος του development τυπικά θα περιλαμβάνει αρκετούς κύκλους, όπου θα κινούμαστε ανάμεσα στο γράψιμο της εφαρμογής και στο τεστάρισμά της σε Hosted mode, χρησιμοποιώντας τον Hosted browser

	(Στάδιο 3). Το Hosted mode μπορεί να φορτωθεί είτε άμεσα από το shell window είτε από το IDE μας. Λειτουργεί ως ένα διαχειριζόμενο περιβάλλον που εκτελεί τον κώδικά μας ως καθαρό κώδικα Java και που εφαρμόζει οποιοδήποτε server side Java κώδικα που έχουμε αναπτύξει στον δικό του εσωτερικό web server. Τα σφάλματα και τα exceptions που προκύπτουν στο Hosted mode, καθώς επίσης και οποιαδήποτε έξοδος από το GWT logging που μπορεί να έχουμε συμπεριλάβει στον κώδικά μας, γίνονται captured με ασφάλεια από αυτό το διαχειριζόμενο περιβάλλον. Εάν συγκρίνουμε αυτό με το Web mode όπου η εφαρμογή έχει γίνει JavaScript κώδικας και εκτελείται άμεσα από έναν web browser, βλέπουμε ότι το δεύτερο δεν έχει εγγυημένα ασφαλή διαχείριση των εξαιρέσεων και των σφαλμάτων. Άλλο ένα πλεονέκτημα του Hosted Mode προκύπτει από την δυνατότητα να το συνδέσουμε με τον debugger του IDE τόσο για client όσο και για server side Java code debugging.
4	Όταν ολοκληρώσουμε την ανάπτυξη του κώδικα σε Hosted mode θα πρέπει να κάνουμε compile τον Java κώδικα σε JavaScript για να τον χρησιμοπιήσουμε σε Web mode (Στάδιο 4). Το compilation ξεκινάει θέτοντας σε λειτουργία τον GWT compiler: τα αρχεία που προκύπτουν είναι στη συνέχεια έτοιμα για να τα δούμε στον web browser. Εάν σε αυτό το σημείο έχουμε μόνο client side κώδικα τότε μπορούμε να ανοίξουμε την εφαρμογή άμεσα από το σύστημα των αρχείων, αλλά εάν έχουμε sever side κώδικα τότε θα πρέπει να εφαρμόσουμε τον κώδικα σε έναν web server. Η διαδικασία του compilation παράγει ένα πλήθος από απαιτούμενα αρχεία.

5	Μία compiled εφαρμογή τυπικά θα εφαρμοστεί στο στάνταρντ περιβάλλον ελέγχου του web server έτσι ώστε να μπορούμε να ελέγξουμε το deployment process καθώς επίσης και να διασφαλίσουμε ότι ο κώδικας εκτελείται σωστά. Η εφαρμογή εξαρτάται κατά πολύ από από τον web server, αλλά εάν δεν έχουμε καθόλου server side κώδικα τότε μπορούμε πάντα να ελέγξουμε το Web mode απευθείας από το file system κάνοντας διπλό κλικ στο HTML αρχείο της εφαρμογής σας από το compiled directory.
6	Για να ολοκληρώσουμε το development θα πρέπει να ελέγξουμε τη λειτουργικότητα σε ένα πλήθος από browsers σε Web mode πριν από το τελικό production release.

Πίνακας 2.1 Περιγραφή των σταδίων που εμπλέκονται σε μία τυπική ανάπτυξη μίας GWT εφαρμογής. Τα στάδια είναι αυτά που φαίνονται στην Εικόνα 2.10 [6]

2.4.1 Βήμα 1 - Δημιουργία μιας GWT εφαρμογής

Σε όλες τις distributions του GWT μπορούμε να δούμε τέσσερις command line applications τα ονόματα των οποίων τελειώνουν με τη λέξη 'Creator'. Η λειτουργικότητα αυτών των εργαλείων συνοψίζεται παρακάτω στον Πίνακα 2.2. Μας βοηθούν να δημιουργήσουμε τη δομή των directory και των αρχείων της GWT εφαρμογής και επιπροσθέτως δημιουργούν ένα σύνολο από default files που όλα μαζί κάνουν τη default εφαρμογή [6].

Ενώ ο Πίνακας 2.2. παρέχει μία πολλή καλή θεώρηση όλων των εργαλείων δεν εξηγεί ποια εργαλεία είναι προαιρετικά και με ποια σειρά θα πρέπει να χρησιμοποιηθούν.

Στάδιο	Όνομα Εργαλείου	Περιγραφή
1A	projectCreator	Το GWT παρέχει στενή σχέση/ενσωμάτωση με το Eclipse IDE, και εάν χρησιμοποιούμε αυτό το IDE θα πρέπει πρώτα να εκτελέσουμε πρώτα αυτό το εργαλείο – εάν δεν πρόκειται να χρησιμοποιήσουμε το Eclipse μπορούμε να αγνοήσουμε αυτό το στάδιο. Αυτό το εργαλείο εγκαθιστά τα απαραίτητα αρχεία που χρειάζονται για να φορτωθεί εύκολα το directory και το file structure στο Eclipse IDE ως Eclipse project. Σε αυτό το στάδιο αυτό σημαίνει τη δημιουργία των απαραίτητων .project και .classpath αρχείων.
1B	applicationCreator	Εκτελεί τις παρακάτω τρεις λειτουργίες: -Δημιουργεί τη δομή πακέτων της Java μέσα σε ένα directory που θα περιέχει την GWT εφαρμογή. -Δημιουργεί default HTML και Java αρχεία μαζί με ένα βασικό Module XML αρχείο το οποίο χρησιμοποιείται για να ενοποιήσει την GWT εφαρμογή. Για τις περισσότερες εφαρμογές θα πρέπει να ξαναγράψουμε όλα αυτά τα αρχεία στο Στάδιο 2 του lifecycle. -Δημιουργεί τα command line scripts που μπορούν να χρησιμοποιηθούν για να φορτώσουμε την GWT εφαρμογή σε Hosted mode και για να το μεταγλωττίσουμε για το Web mode.
1C	i18nCreator	Το I18n είναι ένα ακρωνύμιο για διεθνοποίηση και αυτό το εργαλείο εκτελεί τις ακόλουθες δύο

		λειτουργίες: -Δημιουργεί ένα πολύ απλό property file που περιέχει τα key/value ζευγάρια που δρουν ως σταθερές ή ως μηνύματα σε μία εφαρμογή που χρησιμοποιεί τις GWT δυνατότητες του i18n. -Δημιουργεί ένα καινούριο command line εργαλείο ειδικά για την GWT εφαρμογή που δημιουργήθηκε, το οποίο θα χρησιμοποιήσουμε στο στάδιο 1D.
1D	Appl-i18n	Αυτή η command line εφαρμογή δημιουργείται από το εργαλείο i18nCreator στο στάδιο 1C. Στόχος του είναι να παίρνει το property file που περιέχει σταθερές ή μηνύματα και να δημιουργεί ένα αντίστοιχο Java interface αρχείο. Το interface αρχείο που προκύπτει θα χρησιμοποιείται στον κώδικα της GWT εφαρμογής μας όταν θα χρειαζόμαστε πρόσβαση στις i18n σταθερές και/ή μηνύματα από τα property αρχεία. Στην διαδικασία του compilation το GWT ενώνει το interface αρχείο με το property αρχείο έτσι ώστε η λειτουργία να είναι ενιαία.
1E	junitCreator	Εάν πρόκειται να εκτελέσετε unit testing της GWT εφαρμογής χρησιμοποιώντας το Junit tool τότε το junitCreator tool δημιουργεί μία κατάλληλη δομή φακέλων και αρχείων. Τα unit tests θα γράφονται μέσα στο αρχείο που δημιουργείται από αυτό το εργαλείο.
1F	Import to IDE	Το τελικό προεραιτικό βήμα στο Στάδιο 1 είναι η εισαγωγή της δομής των φακέλων και του κώδικα

		που δημιουργήθηκε από τα creation tools μέσα σε ένα IDE.

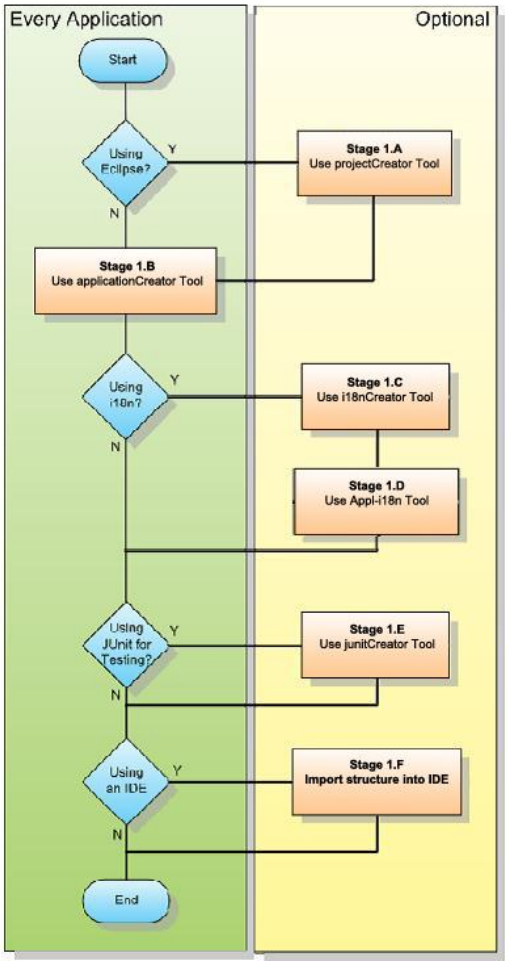
Πίνακας 2.2 Το GWT παρέχει ένα πλήθος διαφορετικών creation tools που μπορούν να χρησιμοποιηθούν για τη γρήγορη ανάπτυξη μίας default GWT εφαρμογής. Τα εργαλεία αυτά χρησιμοποιούνται σε διάφορα στάδια κατά τον τυπικό κύκλο ζωής του development.

Στην Εικόνα 2.11 παρουσιάζεται ένα διάγραμμα που βοηθάει να καταλάβουμε πότε και πού χρησιμοποιούνται συνήθως καθένα από τα εργαλεία αυτά.

2.4.2 Βήμα 2 – Ανάπτυξη της εφαρμογής

Στο πρώτο στάδιο της διαδικασίας μπαίνουν τα θεμέλια της εφαρμογής. Στο δεύτερο στάδιο προχωρούμε στην εναλλαγή των αρχείων που παρέχονται by default για να χτίσουμε τη δική μας GWT application (το στάδιο αυτό είναι αναγκαίο για όλες τις GWT εφαρμογές που χτίζονται με τα GWT creation tools και για τις περισσότερες από αυτές που χτίζονται χρησιμοποιώντας IDE specific tools) [6].

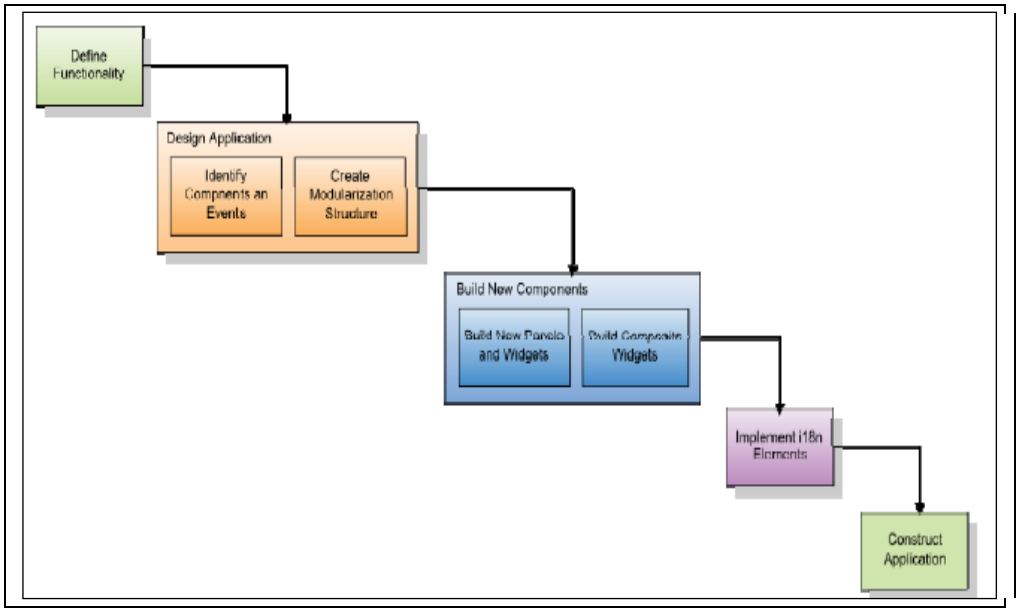
Καθώς οι εφαρμογές που κατασκευάζουμε γίνονται μεγαλύτερες θα πρέπει να λαμβάνουμε υπόψην πιο καλά δομημένες προσεγγίσεις έτσι ώστε να είμαστε σε θέση να έχουμε συνεχώς επίγνωση σε ποιο στάδιο βρισκόμαστε, και ταυτόχρονα να καθίσταται δυνατό για πολλαπλούς developers να εργάζονται επάνω στον κώδικα και γενικότερα να δημιουργείται μία εφαρμογή με δυνατότητες συντήρησης στο χρόνο.



Εικόνα 2.11 Ο πιο κοινός τρόπος χρήσης των διαφόρων GWT creation tools κατά την ανάπτυξη της δομής και των αρχείων.

Στην Εικόνα 2.12 παρουσιάζεται σχηματικά η προσέγγιση που ακολουθείται συνήθως για το χτίσιμο μιας εφαρμογής. Ξεκινάει από τον ορισμό της λειτουργίας που θέλουμε να υλοποιήσουμε και συνεχίζει με το σχεδιασμό της εφαρμογής με τρόπο ώστε να απαντώνται οι λειτουργικές ανάγκες προσδιορίζοντας τα συστατικά στοιχεία και τα γεγονότα που θα χρησιμοποιηθούν. Πρέπει επίσης να δοθεί ιδιαίτερη προσοχή στον τρόπο με τον οποίο θα αξιοποιηθούν οι δυνατότητες που παρέχει το GWT για modularization. Στη συνέχεια δημιουργείται το design της εφαρμογής και το στάδιο αυτό

ολοκληρώνεται με ένα βήμα κατασκευής που περιλαμβάνει το χτίσιμο ποικίλων συστατικών στοιχείων που συνθέτουν την εφαρμογή.



Εικόνα 2.12 Τα τυπικά βήματα του σταδίου ανάπτυξης της εφαρμογής.

2.4.3 Βήμα 3 – Testing και Debugging σε hosted mode

Το hosted mode είναι ένα managed περιβάλλον που παρέχεται ως τμήμα του GWT distribution. Είναι επίσης το περιβάλλον όπου εκτελείται η πλειοψηφία του testing και του debugging του συστήματος [6].

Σε λειτουργικό σύστημα Windows το hosted mode χρησιμοποιεί την version του Internet Explorer εκείνη που είναι εγκατεστημένη στον υπολογιστή ως η υπάρχουσα display technology, ενώ στις άλλες GWT distributions χρησιμοποιείται ένα pre-built Mozilla instance. Όποτε τρέχει ο κώδικας σε σε Hosted mode, ο κώδικας της Java απλώς μεταφράζεται και δεν μεταγλωττίζεται σε JavaScript. Ως αποτέλεσμα, παρέχεται το σημαντικό πλεονέκτημα της δυνατότητας σύνδεσης της εκτέλεσης της εφαρμογής σε Hosted mode με τις debugging δυνατότητες του IDE του συστήματος.

Μία εφαρμογή μπορεί να τρέξει σε hosted mode τόσο από το command line όσο και από τη launch δυνατότητα του Eclipse IDE προγράμματος. Σε Hosted mode μία εφαρμογή εκτελείται μέσω ενός managed περιβάλλοντος που παρέχεται από τη GWT distribution. Ο client-side κώδικας εκτελείται σε έναν managed web browser, όπου ο κώδικας σε Java μεταφράζεται αποτελεσματικά και τυχόν λάθη ή προβλήματα γίνονται captured και κατόπιν αναφέρονται σε ένα παράθυρο ελέγχου. Το Hosted mode περιλαμβάνει επίσης μία ενσωματωμένη έκδοση της Tomcat Servlet engine μέσα στην οποία αναπτύσσεται αυτόματα ο server-side Java κώδικας.

Εάν χρησιμοποιείται, όπως άλλωστε είναι και το πιο συνηθισμένο, ο Eclipse editor τότε το πρόγραμμα είναι δυνατό να εκκινήσει από το Eclipse. Για να γίνει αυτό το μόνο που χρειάζεται είναι να τρέξουμε το πρόγραμμα κανονικά ως μία απλή Java εφαρμογή.

Το Debugging στο Eclipse δουλεύει με τον ίδιο ακριβώς τρόπο όπως το debugging για οποιαδήποτε άλλη Java εφαρμογή (και αυτό εφαρμόζεται τόσο σε client όσο και σε server-side code). Ο Πίνακας 2.3 δείχνει τα βήματα που ακολουθούνται για να γίνει το debugging μίας εφαρμογής.

2.4.4 Βήμα 4 – Compilation του κώδικα

Ενώ το Hosted mode παρέχει ένα ασφαλές περιβάλλον για την εκτέλεση του system testing αφότου ο κώδικας σε Java μεταφράζεται και μπορεί να γίνει debugged, το Web mode είναι εκείνο στο οποίο τρέχει στην πραγματικότητα η εφαρμογή. Στο Web mode η εφαρμογή έχει γίνει compiled σε web technologies και η εφαρμογή δεν είναι πλέον απλός Java κώδικας αλλά πλέον είναι ένα πλήθος από JavaScript files (στο GWT ονομάζονται μεταλλάξεις) για κάθε browser και για κάθε locale συνδιασμό [6].

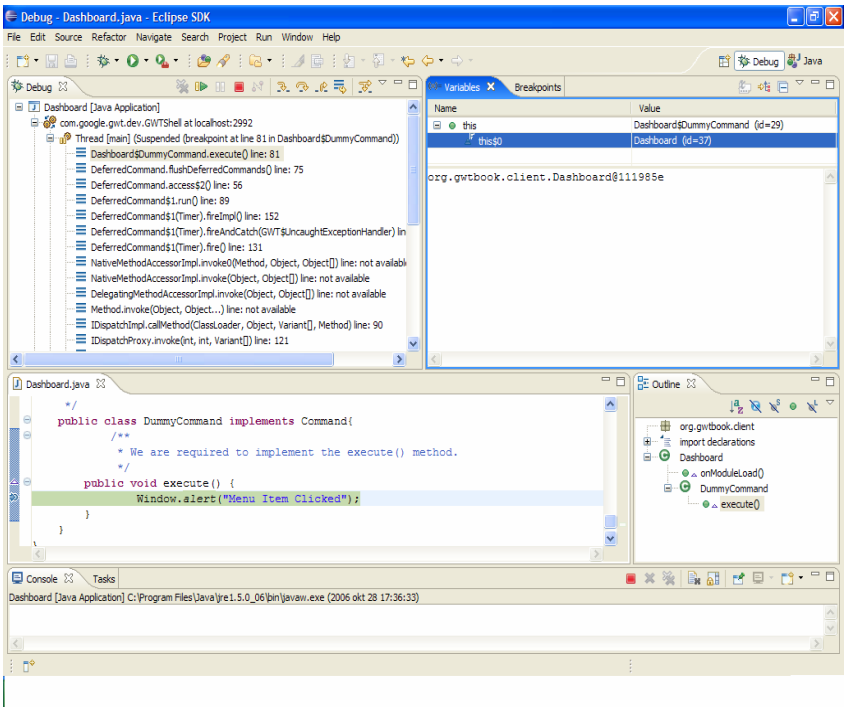
Καθώς δημιουργούνται όλο και πιο περίπλοκες εφαρμογές, που συμπεριλαμβάνουν και διαχειρίζονται διαφορετικές GWT ιδιότητες, το πλήθος των μεταλλάξεων μεγαλώνει. Κατά συνέπεια το matrix space των μεταλλάξεων θα καλύπτει browsers, locales και external visibilities, και το GWT θα τα διαχειρίζεται και θα επιλέγει κάθε φορά να δείξει στο χρήστη τη σωστή 'μετάλλαξη'.

Το πρώτο βήμα για την εκτέλεση της εφαρμογής σε Web mode είναι το compilation του Java κώδικα.

Βήμα	**Περιγραφή**

1	**Παράθεση των breakpoints στον κώδικα**
	
	Το πρώτο πράγμα που πρέπει να κάνουμε είναι να ορίσουμε τα breakpoints στον κώδικά μας στο σημείο όπου επιθυμούμε να εισέλθει ο debugger. Στην Εικόνα μας έχουμε θέσει ένα breakpoint στο σημείο όπου εμφανίζεται ένα μήνυμα στην οθόνη όταν ένας χρήστης επιλέγει κάτι από το μενού (φαίνεται ως η αριστερή μπλε μπάλα στην αριστερή μπάρα).
	
	`public class DummyCommand implements Command{`
	`/**`
	`* We are required to implement the execute() method.`
	`*/`
	`public void execute() {`
	`Window.alert("Menu Item Clicked");`
	`}`
	`}`

2	**Εκτέλεση της εφαρμογής σε debug mode** Το επόμενο βήμα είναι η εκτέλεση του project σε debug mode. Στο Eclipse αυτό εκτελείται με τον ίδιο τρόπο όπως όταν φορτώνουμε το Hosted mode, με εξαίρεση ότι τώρα επιλέγουμε "Debug as.." και όχι "Run as..".
3	**Ενεργοποίηση του break point** Τώρα όποτε εκτελούμε την εφαρμογή σε debug mode φορτώνεται ο hosted browser



και τα πάντα δουλεύουν φυσιολογικά μέχρις ότου πατήσουμε το κουμπί.

4	**Βηματικός έλεγχος του κώδικα**

Στο Web mode, μία εφαρμογή εκτελείται μέσω ενός κανονικού web browser και κατά συνέπεια η GWT εφαρμογή πρέπει να γίνει compiled από Java σε JavaScript προτού συμβεί αυτό. Το compilation γίνεται με την εκτέλεση του compilation script το οποίο έχει επίσης δημιουργηθεί πολύ προσεκτικά από το creator command της εφαρμογής.

Όταν καλεστεί ο compiler θα πάρει τις κλάσεις της Java και θα τις μεταφράσει σε ένα πλήθος από JavaScript αρχεία. Κάθε JavaScript αρχείο (μία μετάλλαξη δηλαδή σε GWT ορολογία) αναπαριστά μία διαφορετική version της εφαρμογής, όπου κάθε μία έχει

διαφοροποιημένες ιδιότητες οι οποίες είναι καθορισμένες ανάλογα με τον τύπο του browser και το internationalization που πιθανώς απαιτείται.

2.4.5 Βήμα 5 – Deployment του κώδικα

Μόλις ολοκληρωθεί η μεταγλώττιση του κώδικα, ακολουθεί η εκτέλεσή του σε Web mode κάνοντάς τον deploy στον κατάλληλο κάθε φορά web server. Όσο πιο πολύπλοκη γίνεται η εφαρμογή τόσο πιο πολύπλοκο μπορεί να γίνει και το deployment.

Όταν υπάρχει server-side κώδικας (σε Java) τότε θα πρέπει να διασφαλισθεί η δημιουργία ενός web.xml αρχείου που αναλύει λεπτομερώς τις υπηρεσίες που γίνονται deployed στο Servlet engine (σε Hosted mode όλη αυτή η διαδικασία γίνεται αυτόματα χωρίς επιπλέον παρέμβαση). Εφόσον στο GWT Module XML file έχουν συμπεριληφθεί service endpoints θα πρέπει να δημιουργηθεί ένα web.xml file για να είναι αυτά ορατά στο web mode deployment [6].

2.4.6 Βήμα 6 – Εκτέλεση σε Web mode

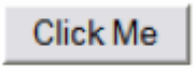
Η εκτέλεση μίας εφαρμογής σε web mode αποτελεί μία περίπτωση navigating προς το σωστό URL στο web server. Εάν σε αυτό το στάδιο παρουσιαστούν λάθη ή προβλήματα το πιθανότερο είναι ότι αυτά συνέβησαν στο ακριβώς προηγούμενο βήμα, κατά το deployment που έγινε [6].

2.5 Τα σημαντικότερα συστατικά στοιχεία μιας GWT εφαρμογής

Ολοκληρώνοντας την παρουσίαση του Google Web Toolkit θα δούμε τα βασικότερα συστατικά στοιχεία μίας GWT εφαρμογής, εκείνα από τα οποία προκύπτουν τα ορατά χαρακτηριστικά της εφαρμογής, όπως για παράδειγμα buttons, labels, images ή το menu system που προκύπτουν από τα widgets. Τα στοιχεία αυτά είναι τα widgets, panels, events, composite widgets και το JavaScript Native Interface (JSNI).

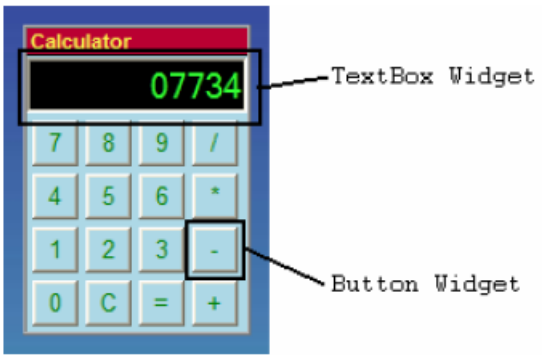
2.5.1 Τα widgets

Τα widgets είναι τα ορατά συστατικά στοιχεία μιας GWT εφαρμογής που μπορεί να δει ένας χρήστης στο browser page. Αποτελούν το ένα από τα τέσσερα θεμελιώδη κομμάτια των GWT applications –τα υπόλοιπα είναι τα panels, τα events και το server communication (που περιλαμβάνει τα RPC, Form, JSON, XML handling και το AJAX HTTPRequest). Όταν ένας χρήστης ενεργοποιεί μία GWT εφαρμογή, κοιτάζει ένα σύνολο από widgets τα οποία έχουν τοποθετηθεί εκεί από panels και τα οποία αντιδρούν σε events. Τα widgets, λειτουργούν όπως τα κουμπιά στο τηλεχειριστήριο της τηλεόρασης, αποτελούν δηλαδή το στοιχείο με το οποίο αλληλεπιδρά ο χρήστης. Ευτυχώς, το GWT παρέχει δωρεάν πολλά διαφορετικά widgets και αυτά περιλαμβάνουν τα συνηθέστερα buttons, textboxes και menus. Στην Εικόνα 2.13 βλέπουμε ένα τέτοιο button [6].



Εικόνα 2.13 Ένα button widget

Οι πιο πολλές εφαρμογές κατασκευάζοναι με τη χρήση πολλαπλών widgets, τα οποία τοποθετούνται μέσα σε panels για να παρέχεται κάποια δομή.

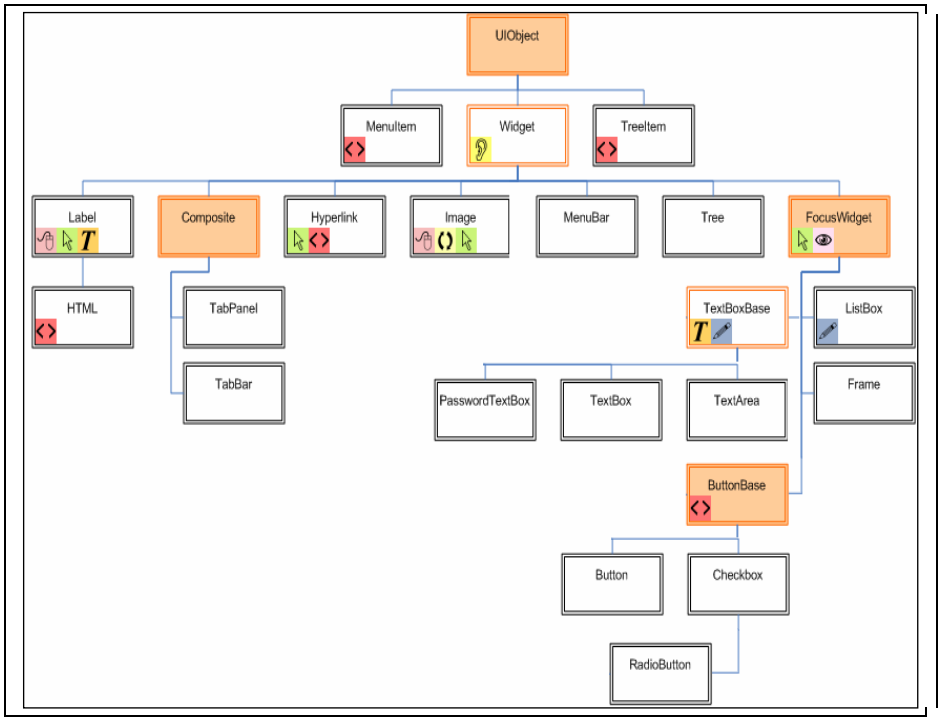


Εικόνα 2.14 Μία εφαρμογή ενός απλού CALCULATOR, στην οποία φαίνεται πως είναι δυνατό να τοποθετηθούν μαζί ένα πλήθος από widgets για να δημιουργήσουν μία ολοκληρωμένη εφαρμογή.

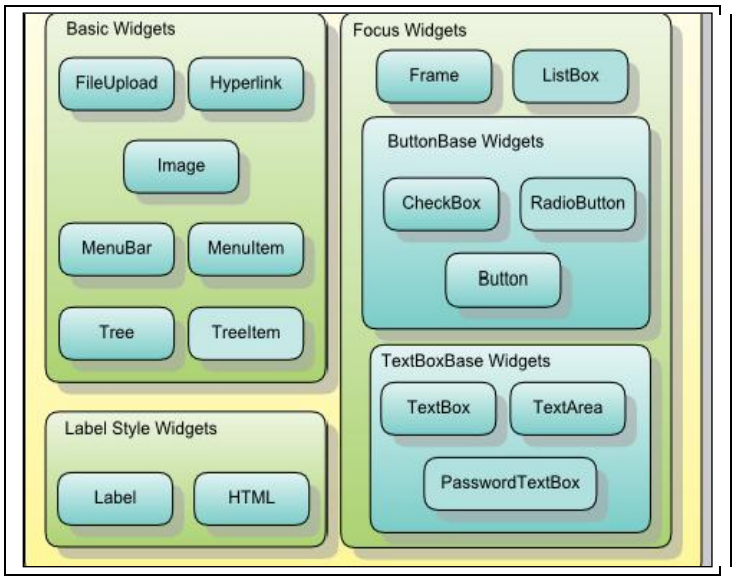
Η στάνταρ GWT distribution παρέχει μία ευρεία γκάμα από widgets για να χρησιμοποιηθούν στις εφαρμογές. Τα widgets αυτά καλύπτουν σε γενικές γραμμές τους τύπους που θα περίμενε κανείς, με buttons, text boxes και άλλα τέτοια widgets. Υπάρχουν ωστόσο ορισμένα αρκετά κοινότυπα τα οποία λείπουν όπως για παράδειγμα τα progress bars και τα sliders.

Μέσα στο σύνολο των widgets οι σχεδιαστές έχουν υλοποιήσει και ενσωματώσει μία πολλή σφιχτή ιεραρχία από κλάσεις Java με στόχο να παρέχουν το στοιχείο της σταθερότητας και της συνέπειας. Η ιεραρχία αυτή παρουσιάζεται αναλυτικά στην Εικόνα 2.15.

Στην Εικόνα 2.16 φαίνονται όλα τα widgets που συμπεριλαμβάνονται με τη GWT distribution ομαδοποιημένα [6].



Εικόνα 2.15 Παρουσίαση της ιεραρχίας των κλάσεων που αναδεικνύει επίσης τους τύπους των event listeners που μπορούν να γίνουν registered σε κάθε widget.



Εικόνα 2.16 Ομαδοποιημένη παρουσίαση των GWT widgets

2.5.2 Τα Panels

Τα panels παρέχουν τα μέσα για να οργανωθεί οπτικά και να δομηθεί μία GWT εφαρμογή. Αποτελούν τα βασικά συστατικά της δομής και πολλές φορές και της λειτουργίας μιας GWT εφαρμογής. Επιτρέπουν την τοποθέτηση των widgets στη θέση που θα πρέπει να βρίσκονται για να καλύπτεται η λειτουργικότητα της εφαρμογής.

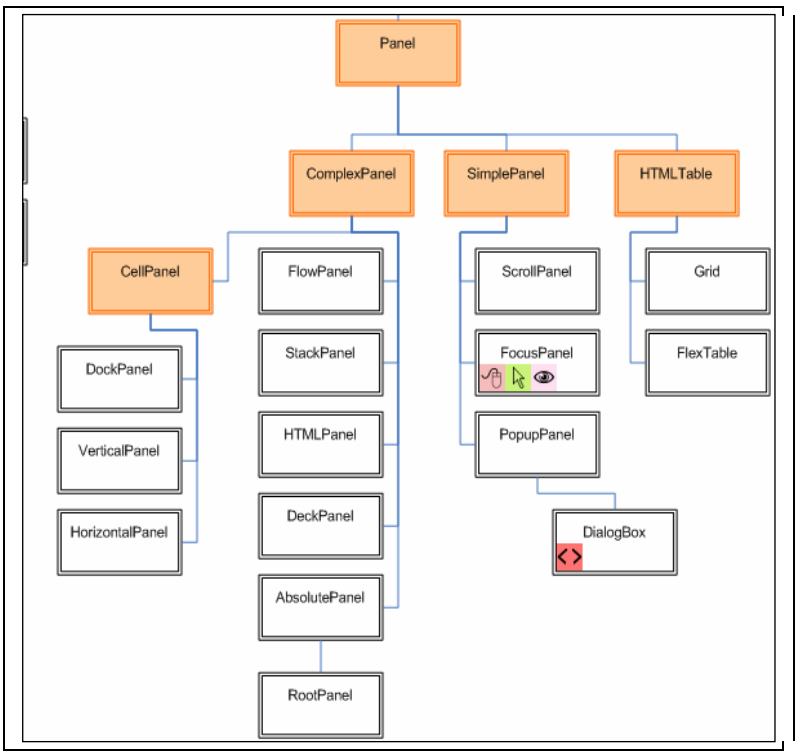
Το GWT παρέχει ένα πλήθος από διαφορετικά panels που ποικίλλουν από το απλό FlowPanel όπου τα δομικά στοιχεία είναι σε διάταξη από επάνω αριστερά προς κάτω δεξιά, μέχρι πιο περίπλοκα panels όπως το DeckPanel όπου μόνο ένα widget είναι ορατό κάθε φορά. Έχουν επίσης διπλή υπόσταση αφού μπορούν να θεωρηθούν είτε Java objects είτε DOM elements.

Η στάνταρ GWT distribution βγαίνει με μία μεγάλη γκάμα από panels που καλύπτουν πολλές διαφορετικές περιστάσει [6].

2.5.3 Τα Events

Τα widgets σε μία GWT εφαρμογή είναι δομημένα μέσω των panels και κάθε φορά που ο χρήστης αλληλεπιδρά μαζί τους θα πρέπει να γίνει κάποια διαχείριση των events [6].

Το event handling στις web applications δένει στην ουσία τα οπτικά στοιχεία που βλέπει ο χρήστης στη λειτουργικότητα της εφαρμογής. Αυτό μπορεί να συμβεί με το κλικάρισμα ενός button, με το component dragging, με την αλλαγή μιας τιμής ή με ένα άλλο από τα events. Το GWT υποστηρίζει όλα τα events που μπορεί να διαχειριστεί ένας browser, όπως φαίνεται στον Πίνακα 2.6. Για να καταφέρουμε ένα widget/panel να ακούσει κάποια γεγονότα θα πρέπει να κατανοήσουμε τον κύκλο ζωής του event handling ο οποίος παρουσιάζεται συνοπτικά στην Εικόνα 2.18 [6].

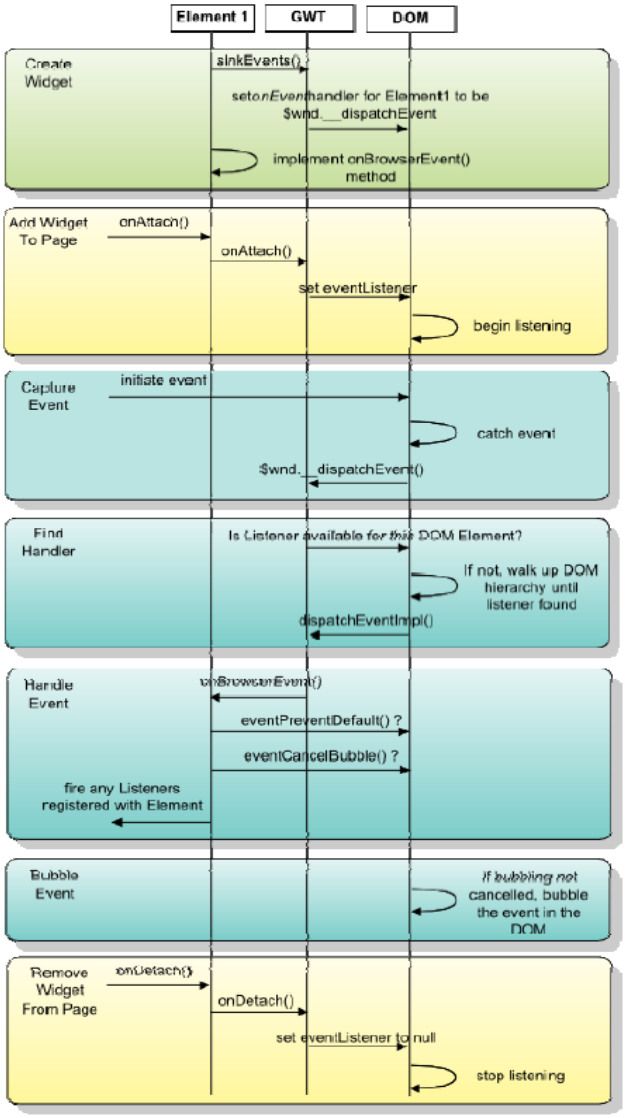


Εικόνα 2.17 Το class hierarchy των panels του GWT framework

Event	Description	GWT Value
BUTTON_LEFT	The left mouse button has been pressed.	N/A
BUTTON_MIDDLE	The middle mouse button has been pressed.	N/A
BUTTON_RIGHT	The right mouse button has been pressed.	N/A
ONBLUR	Occurs when an element loses keyboard focus.	N/A
ONCHANGE	We get this event when the value of an input element changes.	N/A
ONCLICK	When a user clicks on an element this event is fired.	0x00001
ONDBLCLICK	If the user double clicks on an element we get notified of this event.	0x00002
ONERROR	Fired when a JavaScript error occurs (often used when loading IMG which fails).	0x10000
ONFOCUS	This is the opposite of the ONBLUR event and gets fired when an element receives keyboard focus.	0x00800
ONKEYDOWN	Fired when the user depresses a key.	0x00080
ONKEYPRESS	Fired when a character is generated from a keypress (either directly or through auto-repeat).	0x00100
ONKEYUP	Fired when the user releases a key.	0x00200
ONLOAD	Fired when an element (normally an IMG) finishes loading THIS EVENT GETS FIRED.	0x08000
ONLOSECAPTURE	Fired when an element that has mouse capture loses it.	0x02000
ONMOUSEDOWN	Fired when the user depresses a mouse button over an element.	0x00004
ONMOUSEMOVE	Fired when the mouse is moved within an element's area.	0x00040
ONMOUSEOUT	Fired when the mouse is moved out of an element's area.	0x00020
ONMOUSEOVER	Fired when the mouse is moved into an element's area.	0x00010
ONMOUSEUP	Fired when the user releases a mouse button over an element.	0x00200
ONSCROLL	Fired when a scrollable element's scroll offset changes.	0x04000

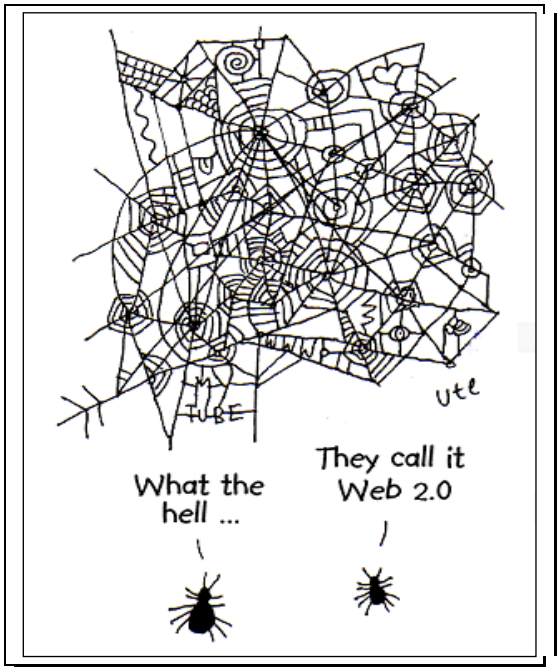
FOCUSEVENTS	A bit-mask covering both the focus and blur events.	ONFOCUS | ONBLUR
KEYEVENTS	A bit-mask covering the down, up and press keyboard events.	ONKEYDOWN | ONKEYPRESS | ONKEYUP
MOUSEEVENTS	A bit-mask covering the down, up, move, over and out mouse events. Note that this does not include the click or double click events.	ONMOUSEDOWN | ONMOUSEUP | ONMOUSEMOVE | ONMOUSEOVER | ONMOUSEOUT

Πίνακας 2.6 Μία λίστα από όλα τα browser events που μπορεί να επιλέξει να διαχειριστεί μαζί μία GWT application μαζί με πιθανές GWT τιμές που μπορούν να τους ανατεθούν.



Εικόνα 2.18 Ο κύκλος ζωής της διαδικασίας διαχείρισης ενός event για κάποιο widget/panel (το Element) στο GWT.

Dojo toolkit



3.1 Εισαγωγή στο Dojo

Τα τελευταία χρόνια έχει εμφανιστεί ένα πλήθος από νέες τεχνολογίες οι οποίες επαναπροσδιορίζουν τους κανόνες του server-side web application development. Με το Dojo μπαίνουμε σε μία νέα εποχή με browser-based εφαρμογές [7].

Το Dojo είναι ένα σύνολο εργαλείων που μας βοηθούν να χτίσουμε καλύτερες browser-based εφαρμογές. Κατασκευασμένο κυρίως με τη χρήση client-side JavaScript, είναι σε θέση να επεκτείνει τις δυνατότητες ενός σύγχρονου browser (ακόμα και του Internet Explorer) σε σημείο που να εξαφανίζεται η διαχωριστική γραμμή μεταξύ των local, native εφαρμογών και των browser-based εφαρμογών με συνέπεια, να μην μπορεί κανείς να διακρίνει τις browser-based (και επομένως και τις web-based) διεπαφές του χρήστη από εκείνες των καλύτερων local native εφαρμογών. Επίσης, το user interface των τοπικών εφαρμογών μπορεί να υλοποιηθεί στον browser και όχι διαμέσου ενός από τα βαριά, platform-sensitive και πολύπλοκα native GUI frameworks (δηλαδή Windows, Tk, Qt, Fox, AWT, SWT, Swing, Cocoa κ.ά).

Ο ευρέως χρησιμοποιούμενος browser γίνεται η user interface πλατφόρμα και πλέον δεν παίζει κανένα ρόλο το πού βρίσκεται το back end ˙ σε ένα δίκτυο από απομακρυσμένους HTTP servers ή σε ένα μικρό τοπικό πρόγραμμα που υλοποιεί το HTTP πρωτόκολλο. Δυστυχώς, ο σύγχρονος browser παρέχει ένα προγραμματιστικό περιβάλλον που είναι ημιτελές, άβολο και ασύμβατο. Το πρόβλημα αυτό μπορεί να αντιμετωπιστεί με τη χρήση ενός plug-in όπως το ActionScript, με το βασικό όμως μειονέκτημα ότι έτσι παραβιάζεται ο πολύ σημαντικός κανόνας που απαγορεύει αυστηρά την εγκατάσταση software τοπικά. Αν και δεν θα υπάρξει σημαντικό πρόβλημα από την παραβίαση αυτού του κανόνα για μία τοπική εφαρμογή, σίγουρα θα υπάρξει για μία web-based εφαρμογή.

Το Dojo διορθώνει τα ελαττώματα-δυσλειτουργίες του browser όπως για παράδειγμα τις ασυμβατότητες και τις απώλειες μνήμης και του προσδίδει σημαντικές

δυνατότητες όπως HTML user interface controls και DOM quering. Αν και υπάρχουν διαθέσιμες πολλές JavaScript βιβλιοθήκες, οι περισσότερες επικεντρώνονται σε μία συγκεκριμένη ιδέα, όπως αυτές που περιλαμβάνουν μόνο effects libraries ή υλοποιούν ένα ή περισσότερα HTML widgets (user interface controls). Αντίθετα, το Dojo προσανατολίζεται προς όλες αυτές τις λειτουργικές περιοχές –και σε πολλές ακόμα- και υποστηρίζεται λόγω αυτών ότι έχει πλάτος και ύψος όπως καμία άλλη open source λύση. Για παράδειγμα, παρατηρώντας το βάθος του Dojo, παρατηρείται ότι κανονικοποιεί το event system των δημοφιλέστερων browsers (ο Internet Explorer δεν υλοποιεί το W3C event model και έχει διαρροές μνήμης, ενώ οι περισσότεροι άλλοι browsers κάνουν το αντίθετο) [7].

Επιπλέον, το εργαλείο αυτό έχει μεγάλο πλάτος. Περιλαμβάνει ορισμένα user interface controls, ένα cometd support πλαίσιο γραφικών (δηλαδή μία τεχνική τηλεπικοινωνιών χαμηλού latency που επιτρέπει στο server να σπρώχνει τα δεδομένα στον browser), ένα packaging system και πολλά άλλα. Σήμερα, το Dojo ξεχωρίζει για τις τεράστιες δυνατότητές του.

Παρά τις μεγάλες δυνατότητές του δεν είναι πολύπλοκο αφού είναι οργανωμένο με σωστή ιεράρχηση των λειτουργιών του. Αυτό επιτρέπει στο χρήστη και στον προγραμματιστή να επικεντρωθούν στην περιοχή που χρειάζονται για την εργασία πάνω στην οποία δουλεύουν. Καθώς οι ανάγκες αλλάζουν και εξελίσσονται το Dojo είναι σε θέση να καλύψει με ακρίβεια τις ανάγκες αυτές εξαιτίας του πλάτους και του βάθους του. Συγκρίνοντας το Dojo με μία μικρότερη, λιγότερο φιλόδοξη βιβλιοθήκη μπορεί η δεύτερη να είναι πολύ πιο εύκολη στην κατανόηση –αν και ακόμα κι αυτό είναι αμφίβολο- αλλά μετά από τρεις ή έξι μήνες χρήσης όταν ο προγραμματιστής χρειαστεί κάποια δυνατότητα που δεν παρέχεται από τη βιβλιοθήκη θα πρέπει είτε να μάθει να χρησιμοποιεί μία καινούρια είτε να υλοποιήσει μόνος του ότι χρειάζεται. Οι δύο αυτές επιλογές απαιτούν σίγουρα πολύ περισσότερο κόστος και πολυπλοκότητα από το να μάθει κανείς το Dojo από την αρχή [7].

Μία από τις βασικές ιδέες της κοινότητας του Dojo είναι η ελάττωση της πολυπλοκότητας. Όλα τα βασικά χαρακτηριστικά του σχεδιασμού του Dojo έχουν επανειλημμένως συζητηθεί, υλοποιηθεί, επανασχεδιασθεί, τεσταριστεί και χρησιμοποιηθεί έχοντας πάντα αυτή τη βασική ιδέα κατά νου. Αντί να αγνοήσει τα ελαττώματα και να δημιουργήσει κάτι λαμπερό αλλά ασταθές η κοινότητα του Dojo έχει συνδιάσει την ώριμη, λογική και σκεπτική προσέγγιση του μηχανικού με την γρήγορη και πρωτοποριακή σκέψη του νέου χάκερ. Η κοινότητα του Dojo πέρασε σχεδόν όλόκληρο το έτος 2007 επανακατασκευάζοντας το core system του Dojo αποδεικνύοντας με αυτόν τον τρόπο την αφοσίωσή της στην τελειότητα και τον επαγγελματισμό που μπορεί μερικές φορές να λείπει από τα open source projects. Τέλος, θα πρέπει να επισημάνουμε ότι το Dojo δεν είναι ακαδημαϊκό project και χρησιμοποιείται σε εκατοντάδες projects σε εκατοντάδες εταιρείες [7].

Συνοψίζοντας τα βασικά πλεονεκτήματα του Dojo που το καθιστούν ένα από τα ποιοτικότερα και πληρέστερα εργαλεία σε JavaScript αναφέρουμε τα παρακάτω [8]:

- Το Dojo είναι ένα πλήρες πακέτο. Αντί να συνδυάζει απλώς διάφορα component από διαφορετικές πηγές, το Dojo επιτρέπει το κάθε component να στηρίζεται πάνω σε μία υψηλής ποιότητας και πολλά υποσχόμενη πλατφόρμα η οποία παρέχει μία ευρεία γκάμα από προαιρετικά modules. Αυτά τα components αποτελούν πολύ καλές λύσεις σε πολλά από τα προβλήματα που μπορεί να συναντήσουμε ενώ μπορούν πολύ εύκολα να μετατραπούν και να προσαρμοστούν στις ανάγκες μας. Οι δυνατότητες που μας παρέχουν είναι από απλές φόρμες σύνταξης περιεχομένου μέχρι δημιουργία διαγραμμάτων στην πλευρά του πελάτη.
- Περιέχει ενσωματωμένα στοιχεία για διεθνοποίηση (internationalization) του περιεχομένου και για πρόσβαση (accessibility) ατόμων με ειδικές ανάγκες.

- Όλα τα components συνδυάζονται άριστα μεταξύ τους ενώ υπάρχει και η δυνατότητα παραμετροποίησης με χρήση CSS, χωρίς να απαιτούνται πολλές αλλαγές για να πάρουμε ένα εμφανίσιμο User Interface.
- Το Dojo έχει φτιαχτεί με γνώμονα την απόδοση. Υπάρχουν πολλές ιστοσελίδες υψηλής επισκεψιμότητας που βασίζονται σε αυτό. Από την έκδοση 0.9 και ύστερα οι προγραμματιστές επικεντρώθηκαν στην δημιουργία κώδικα υψηλής απόδοσης και μικρότερου μεγέθους.
- Το Dojo αποτελεί μία ανοικτή κοινότητα ταχύτατα αναπτυσσόμενη. Έτσι πολλοί ανεξάρτητοι προγραμματιστές αλλά και εταιρείες μπορούν να συνεργαστούν και να δημιουργήσουν εργαλεία που μπορούν να χρησιμοποιηθούν από όλους.

3.2 Τα βασικά χαρακτηριστικά του Dojo

Είναι toolkit και όχι απλή βιβλιοθήκη

Το Dojo είναι μία συλλογή από στατικά, client-side JavaScript scripts, που δεν περιλαμβάνει client-side plug-in ή server-side συστατικά. Περιλαμβάνει τα παρακάτω χαρακτηριστικά [7]:

- Σχεδιασμός και υλοποίηση που κανονικοποιούν τον browser, επιτρέποντας τη λειτουργία του ίδιου πηγαίου κώδικα σε πολλούς διαφορετικούς browsers, βάζοντας τέλος σε καταστάσεις με browser/feature sniffing και σε κώδικα που εξαρτάται άμεσα από τον εκάστοτε browser.
- Συναρτήσεις/βιβλιοθήκες που συνοψίζουν το ορισμένες φορές μη-έξυπνο, κρυμμένο και άβολο μοντέλο προγραμματισμού W3C DOM σε ένα interface που είναι βολικό, λιτό και αποδοτικό.
- Συναρτήσεις και βιβλιοθήκες που διορθώνουν ποικίλλα σφάλματα από browsers όπως τις διαρροές μνήμης και άλλες που παρέχουν λειτουργικότητα που πιθανότατα θα είναι πρωτογενώς διαθέσιμη στον browser στο μέλλον.

- Μία βιβλιοθήκη που αποτελείται από το μεγαλύτερο σήμερα διαθέσιμο ενιαίο σύνολο από HTML widgets.
- Ένα module system μαζί με ένα build system που επιτρέπει τη διαίρεση του κώδικα σε μικρότερα, ευκολότερα διαχειρίσιμα κομμάτια κατά τη διάρκεια του development και στη συνέχεια το packaging του release system για βέλτιστη απόδοση στο download χωρίς να γίνουν τροποποιήσεις στον πηγαίο κώδικα. Το build system στην ουσία επιτρέπει τη 'διαίρεση' του Dojo σε τμήματα με τρόπο βέλτιστο για το εκάστοτε project.
- Ανεξάρτητες βιβλιοθήκες (δηλαδή βιβλιοθήκες που μπορεί να φορτωθούν κατά απαίτηση) που υλοποιούν και παρέχουν ποικίλες εξελιγμένες δυνατότητες.

Αρκετές από τις βιβλιοθήκες καταλήγουν σε frameworks για την υλοποίηση των παρακάτω:

- Χτίσιμο των συνηθισμένων HTML widgets.
- Internationalization (i18n).
- Localization (l10n).
- Προσβασιμότητα (a11y).

Περιλαμβάνει επίσης μία εφαρμογή με utilities που λέγεται build system το οποίο πακετάρει μεγάλα projects τα οποία μπορεί να περιλαμβάνουν εκατοντάδες αρχεία σε μικρά, βέλτιστα σύνολα από συμπιεσμένα αρχεία για εφαρμογή σε production servers. Το Dojo υποστηρίζει επισήμως τους Internet Explorer (6+), Firefox (1.5+), Safari (3+) και Opera (9+, το Dijit δεν υποστηρίζει τον Opera).

Εφόσον το Dojo είναι καθαρή JavaScript μπορεί να χρησιμοποιηθεί σε non-browser, Spider-Monkey και Rhino-embedded περιβάλλοντα. Φυσικά μεγάλο κομμάτι του Dojo δεν είναι εφαρμόσιμο σε αυτά τα περιβάλλοντα. Ακόμη υπάρχει ένας εσωτερικός πυρήνας λειτουργιών ο οποίος είναι χρήσιμος. Ορισμένες λειτουργίες που

εξαρτώνται μόνο από JavaScript και μπορούν να χρησιμοποιηθούν σε αυτά τα non-browser περιβάλλοντα είναι: loader, language extensions, ασύγχρονος προγραμματισμός, αντικειμενοστρεφής προγραμματισμός και Common Locale Data Repository.

Απευθύνεται στο ευρύ κοινό

Το Dojo απευθύνεται κατηγορηματικά σε μία ευρεία γκάμα χρηστών, από σχεδιαστές απλών ιστοσελίδων μέχρι επιχειρήσεις που ασχολούνται με ανάπτυξη εφαρμογών. Υπάρχει μεγάλη δυσκολία ώστε να αντεπεξέλθει στις απαιτήσεις όλων των παραπάνω αφού οι σχεδιαστικές αποφάσεις που είναι βέλτιστες για τη μία ομάδα μπορεί να μην ανταποκρίνονται στις ανάγκες της άλλης. Τα περισσότερα web design tasks είναι δυνατό να έρθουν σε πέρας απλώς φορτώντας το script dojo.js. Από αυτή την άποψη το Dojo είναι τόσο 'ελαφρύ' και εύκολο στη χρήση όσο οι πιο ανταγωνιστικές βιβλιοθήκες. Από την άλλη πλευρά, μεγαλύτερα projects έχουν και μεγαλύτερες απαιτήσεις. Το Dojo περιλαμβάνει μηχανισμούς για να φορτώνει κατά απαίτηση διαφορετικές οικογένειες λειτουργιών. Αυτός ο σχεδιασμός δίνει στους χρήστες του Dojo την πολυτέλεια να είναι σε θέση να αφομοιώσουν ακριβώς όση πολυπλοκότητα χρειάζονται για να λύσουν το πρόβλημα [7].

Στοχεύει στο μέλλον

Η κατάσταση του browser-based προγραμματιστικού περιβάλλοντος είναι μία άλλη βασική δύναμη πίσω από τη φιλοσοφία και το περιεχόμενο του Dojo. Εάν όλοι οι browsers ακολουθήσαν κάποια συγκεκριμένα standards –ή αν έστω λήταν συμβατοί με αυτά-, εάν τα JavaScript και DOM API ήταν σε θέση να διορθώσουν ορισμένα ολοφάνερα μειονεκτήματα και εάν η HTML περιελάμβανε user interface controls πιο σύγχρονα από το 1989, τότε μεγάλο μέρος του Dojo θα ήταν άχρηστο. Οι σχεδιαστές του Dojo αναγνωρίζουν ότι αυτά τα ελαττώματα θα διορθωθούν επιτέλους. Το Dojo δημιουργήθηκε για να λύσει τα κύρια προβλήματα στα browser-based προγραμματιστικά περιβάλλοντα έτσι ώστε τα μοντέρνα προγράμματα υψηλών δυνατοτήτων (στην

πραγματικότητα προγράμματα αντίπαλα των native εφαρμογών) να είναι στραμμένα στον browser, καθώς προετοιμάζεται ένα forward upgrade μονοπάτι όσο βελτιώνεται το native browser περιβάλλον.

Εν συντομία το Dojo παρέχει ένα σταθερό browser-based προγραμματιστικό περιβάλλον, ακόμα και καθώς μπαίνουμε στον επόμενο γύρο πολέμου μεταξύ των browsers [7].

Το Dojo ως open source εργαλείο

Ο πηγαίος κώδικας του Dojo διατίθεται δωρεάν. Είναι dual-licensed υπό τους όρους είτε της τροποποιημένης BSD άδειας είτε της Academic Free License version 2.1. Η BSD license είναι πολλή φιλική για τα εμπορικά προϊόντα και επιτρέπει τη χρήση ή την τροποποίηση του Dojo στα προϊόντα του κάθε χρήστη χωρίς καθόλου προϋποθέσεις. Φυσικά μπορεί ο καθένας να αλλάξει οτιδήποτε θέλει σε τέτοια προϊόντα.

Η διαδικασία της ανάπτυξης είναι ακριβής και ανοιχτή. Ο πηγαίος κώδικας διατηρείται σε μία SVN αποθήκη και τα μειονεκτήματα και τα πλεονεκτήματα ανιχνεύονται από το Trac. Επιπλέον επιβάλλεται η τήρηση κατευθυντήριων γραμμών στο coding style ενώ ο κώδικας πρέπει υποχρεωτικά να συνοδεύεται από unit tests πριν από την ένταξη στα βασικά release σύνολα. Στο διαδίκτυο υπάρχει δωρεάν υποστήριξη μέσω forums και mailing lists καθώς επίσης και μέσω εταιρειών που παρέχουν υπηρεσίες με πληρωμή.

Σε αντίθεση με πολλά open source projects, το Dojo υποστηρίζεται από ένα ίδρυμα. Το Dojo Foundation είναι ένα μη κερδοσκοπικό ίδρυμα που έχει οργανωθεί για να βοηθήσει στην προώθηση της υιοθέτησης του Dojo και για να παρέχει ένα υγιές περιβάλλον για JavaScript engineering κάθε είδους. Ένα από τα βασικά οφέλη που πηγάζουν από το ίδρυμα αυτό είναι η ικανότητα να προστατεύσει τους χρήστες από

κάποιες μη-εμφανείς επιβαρύνσεις (για παράδειγμα πατέντες ή καταπάτηση των πνευματικών δικαιωμάτων) όσον αφορά τη χρήση του κώδικα [7].

Διαίρεση σε μικρότερα projects

Το Dojo περιλαμβάνει τρία κύρια projects που θα αναλύσουμε παρακάτω:

- ***Dojo*** Το θεμέλιο project πάνω στο οποίο χτίστηκε οτιδήποτε άλλο. Συνολικά περιλαμβάνει περίπου πενήνητα JavaScript σκριπτάκια και αρκετούς άλλους πόρους που διαχειρίζονται την κανονικοποίηση του browser, JavaScript modularization, επεκτάσεις στην κύρια JavaScript βιβλιοθήκη, επεκτάσεις στο W3C DOM API (που περιλαμβάνουν ανάλυση και querying στο DOM), remote scripting, Firebug Lite, drag and drop, ένα API για αποθήκευση δεδομένων, localization και internationalization και διάφορες άλλες συναρτήσεις.

- ***Dijit*** Αποτελεί το Dojo widget framework και built-in widgets (περίπου σαράντα HTML user interaface widgets).

- ***Dojox*** Dojo extensions. Σε αυτό περιλαμβάνονται τα πάντα από το grid widget μέχρι τις βιβλιοθήκες των γραφικών, ενώ υπάρχουν μερικές πολύ εξεζητημένες και σταθερές βιβλιοθήκες οι οποίες εφαρμόζονται στον πραγματικό κόσμο σε κερδοσκοπικά συστήματα καθώς επίσης και σε πλήρως πειραματικά συστήματα. Κάθε βιβλιοθήκη περιλαμβάνει ένα readme αρχείο το οποίο περιγράφει το project.

Καθένα από αυτά τα τρία project βρίσκεται στο δικό του code tree [7].

Υψηλή εννοιολογική ακεραιότητα

Παρά το μέγεθος του Dojo, ο σχεδιασμός και η υλοποίηση ενέχουν ένα υψηλό επίπεδο εννοιολογικής ακεραιότητας, η οποία αποτελεί το λόγο της λειτουργικότητας

προς την πολυπλοκότητα και είναι το βασικότερο χαρακτηριστικό οποιουδήποτε project με προγραμματισμό. Αυτό είναι άλλωστε και το βασικό χαρακτηριστικό των πακέτων λογισμικού που ζούνε πολλά χρόνια, το οποίο όμως συχνα λείπει από αυτά. Τονίσαμε ήδη ότι η καταπολέμηση της πολυπλκότητας είναι η μία από τις βασικές ιδέες που διέπουν τα project του Dojo. Πηγαίνοντας ένα βήμα παραπέρα θα λέγαμε ότι τα project trees των Dojo και Dijit τα διαχειρίζεται το καθένα ξεχωριστά ένα μόνο άτομο το οποίο καθοδηγεί και συντονίζει την εξέλιξη του project. Με αυτό τον τρόπο αναδεικνύεται μία ενιαία αρχιτεκτονική συστήματος και προχωρούμε ένα βήμα πιο κοντά προς την εννοιολογική ακεραιότητα [7].

Ασύγχρονη επικοινωνία

Ένα σημαντικό χαρακτηριστικό των Ajax εφαρμογών είναι η ασύγχρονη επικοινωνία του browser με τον sever, κατά την οποία ανατλλάσσεται πληροφορία και η παρουσίαση της σελίδας γίνεται updated χωρίς να χρειάζεται να ξαναφορτωθεί ολόκληρη η σελίδα. Παραδοσιακά αυτό γίνεται με το JavaScript object XMLHttpRequest. Το Dojo παρέχει ένα abstracted wrapper (το *dojo.xhr*) γύρω από ποικίλες web browser υλοποιήσεις του XMLHttpRequest και το dojo.io υποστηρίζει επίσης άλλες μεταφορές (όπως κρυφά IFrames) και μία ποικιλία από data formats. Χρησιμοποιώντας αυτή την προσέγγιση είναι εύκολο να έχουμε τα δεδομένα που εισάγει ένας χρήστης σε μορφή κατάλληλη για αποστολή στον server ‘στο παρασκήνιο’. Στη συνέχεια ο server μπορεί να απαντήσει με JavaScript κώδικα ο οποίος κάνει update την παρουσίαση της σελίδας.

Packaging System

Το Dojo παρέχει ένα packaging system για να διευκολύνει την τμηματική ανάπτυξη της λειτουργικότητας σε ατομικά πακέτα και υπο-πακέτα. Το βασικό Dojo ‘bootstrap’ script αρχικοποιεί ένα σύνολο από ιεραρχικά package namespaces, όπως ‘io’,

‘event’ κ.ά., κάτω από ένα root ‘dojo’ namespace. Μετά από την αρχικοποίηση του root namespace, μπορεί να φορτωθεί οποιοδήποτε Dojo package (μέσω XMLHttpRequest ή άλλης παρόμοιας μεταφοράς) με τη χρησιμοποίηση utility συναρτήσεων που παρέχονται από το bootstrap. Υπάρχει επίσης δυνατότητα να αρχικοποιηθούν πρόσθετα namespaces μέσα ή παράλληλα στο ‘dojo’ namespace, επιτρέποντας τα extensions του Dojo ή την ανάπτυξη private namespaces διαχειριζόμενα από το Dojo για third-party βιβλιοθήκες και εφαρμογές.

Τα Dojo πακέτα μπορεί να αποτελούνται από πολλαπλά αρχεία και μπορούν να καθορίσουν ποια αρχεία θα αποτελούν ολόκληρο το πακέτο. Οποιοδήποτε πακέτο ή αρχείο μπορεί επίσης να καθορίσει την ύπαρξη κάποιας εξάρτησης από άλλα πακέτα ή αρχεία και όποτε φορτώνεται το πακέτο θα φορτώνονται και όλες οι εξαρτήσεις που έχει καθορίσει. Παρέχονται επίσης workarounds για cross-domain φόρτωση των περισσότερων πακέτων Dojo (αν και αυτό απαιτεί εξειδικευμένη κατασκευή του Dojo).

Το Dojo παρέχει επίσης έναν μηχανισμό για την κατασκευή προφιλ. Το σύστημα κατασκευής παίρνει ως είσοδο μία λίστα από πακέτα και χρησιμοποιεί το Rhino για τη δημιουργία ενός συμπιεσμένου JavaScript αρχείου που παριέχει τα πακέτα εκείνα μαζί με όλες τις εξαρτήσεις τους. Αυτό επιτρέπει να φορτώνεται και να αρχικοποιείται με τη μία όλος ο απαραίτητος κώδικας και επιτρέπει το caching του κώδικα (οι περισσότεροι web browsers δεν κάνουν cache αρχεία που φορτώνονται μέσω του XMLHttpRequest). Υπάρχουν επίσης διαθέσιμα για download από την ίδια τοποθεσία με το πλήρες toolkit όλα τα προκατασκευασμένα προφίλ.

Client-side αποθήκευση δεδομένων

Επιπροσθέτως στην παροχή υποστηρικτικών συναρτήσεων για ανάγνωση και συγγραφή cookies, το Dojo παρέχει επίσης μία δμή για τοπική, client-side αποθήκευση που ονομάζεται Dojo Storage. Το Dojo Storage επιτρέπει στις web εφαρμογές να

αποθηκεύουν δεδομένα στην πλευρά του client συνεχόμενα, με ασφάλεια και με την άδεια του χρήστη. Η εφαρμογή αυτή δουλεύει σε υπάρχοντες web browsers συμπεριλαμβανομένων των Internet Explorer, Firefox και Safari. Όταν συμπεριλαμβάνεται σε μία ιστοσελίδα, το Dojo Storage καθορίζει την καλύτερη μέθοδο για σταθερή αποθήκευση πληροφορίας. Στον Firefox 2 χρησιμοποιεί native browser persistence, ενώ σε άλλους browsers χρησιμοποιεί ένα κρυφό Flash applet. Με το Flash 6+ εγκατεστημένο σε περίπου το 95% των υπολογιστών που είναι συνδεδεμένοι στο web, ο μηχανισμός αποθήκευσης γίνεται προσβάσιμος για μεγάλο τμήμα της εγκατεστημένης βάσης του ιστού. Για μία δικτυακή εφαρμογή που φορτώνεται από το file system (δηλαδή από ένα αρχείο ://URL), το Dojo Storage θα χρησιμοπιήσει ξεκάθαρα το XPCOM στον Firefox και το ActiveX στον Internet Explorer για τη διατήρηση της πληροφορίας. Το Dojo Storage δεν υποστηρίζεται από την έκδοση 1.3 και τις μεταγενέστερες.

Server-side αποθήκευση δεδομένων

Από τον Ιανουάριο του 2007 το Dojo περιλαμβάνει τις παρακάτω server-side υλοποιήσεις για αποθήκευση δεδομένων στο dojo.data namespace:

- **CsvStore**: μία αποθήκη μόνο για ανάγνωση που διαβάζει δεδομένα πινάκων από comma-separated values αρχεία
- **OpmlStore:** μία αποθήκη μόνο για ανάγνωση που διαβάζει ιεραρχικά δεδομένα από αρχεία τύπου OPML
- **YahooStore:** μία αποθήκη μόνο για ανάγνωση που μεταφέρει αποτελέσματα αναζήτησης από την διαδικτυακή υπηρεσία του Yahoo! Search
- **DeliciousStore:** μία read-only αποθήκη που μεταφέρει bookmarks από την διαδικτυακή υπηρεσία του del.icio.us

- **RdfStore:** μία αποθήκη για γραφή και για ανάγνωση που χρησιμοποιεί SPARQL για να επικοινωνήσει με τους data servers RDF, συμπεριλαμβανομένου για παράδειγμα του Rhizome RDF application server.

Υποστήριξη του Adobe Integrated Runtime (AIR)

Το Dojo μπορεί να χρησιμοποιηθεί σε βασισμένες σε JavaScript Adobe AIR εφαρμογές. Έχει τροποποιηθεί ώστε να καλύπτει τις απαιτήσεις ασφαλείας του AIR.

3.3 Το Dojo ως client-side βιβλιοθήκη για την ανάπτυξη Ajax εφαρμογών

Το εργαλείο που εξετάζουμε αποτελεί μία πολλή ισχυρή JavaScript βιβλιοθήκη που περιλαμβάνει ένα σύνολο από core libraries για HTML, JavaScript, και για το DOM, καθώς επίσης και βιβλιοθήκες για δομές δεδομένων, RPC, JSON, animation, drag-and-drop, validation, κρυπτογραφία, μαθηματικά καθώς και μία εκτεταμένη συλλογή από widgets [9]. Οι πιο πολλές από τις special editions του Dojo (για παράδειγμα μία Ajax edition, event edition ή widget edition) υπάρχουν διαθέσιμες στο site http://dojotoolkit.org.

Το Dojo αποτελείται από διάφορα πακέτα συμπεριλαμβανομένων πακέτων για Ajax, animation, drag-and-drop, events και widgets. Μόλις ο χρήστης αντιγράψει το dojo.js στον Web server του θα έχει αμέσως πρόσβαση από τη σελίδα του σε ένα Dojo package απλώς περιλαμβάνοντας ένα script element μαζί με το URL για το dojo.js file, το οποίο είναι επίσης γνωστό ως Dojo bootstrap [9]:

```
<script type="text/javascript" src="dojo.js"></script>
```

Στη συνέχεια θα πρέπει να προσθέσει άλλο ένα script element που προσδιορίζει ποια Dojo packages χρησιμοποιούνται στη συγκεκριμένη σελίδα:

```
<script type="text/javascript">
    dojo.require("dojo.event.*");
    dojo.require("dojo.widget.*");
</script>
```

Τα Ajax requests γίνονται διαχειρίσιμα με τη χρήση της μεθόδου dojo.io.bind():

```
dojo.io.bind ({
    url: "http://example.com/Data.txt",
    load: processResults,
    mimetype: "text/plain"
 });
```

Η μέθοδος αυτή περιλαμβάνει το URL για το server request, μία callback συνάρτηση, και ένα MIME type για το request. Για τη χρησιμοποίηση αυτής της μεθόδου, θα πρέπει να συμπεριληφθεί το io package, ως εξής:

```
<script type="text/javascript">
    dojo.require("dojo.io.*");
</script>
```

Η μέθοδος io.bind() μπορεί επίσης να χρησιμοποιηθεί για την υποβολή form data. Το formNode αναφέρεται στο form element χρησιμοποιώντας την id τιμή του.

```
dojo.io.bind({
    url: "http://example.com/form.php",
    load: processResults,
    formNode: document.getElementById('myForm')
});
```

Το Dojo περιλαμβάνει αρκετές μεθόδους για εργασίες με το DOM, όπως την createDocument() η οποία επιστρέφει ένα XML document object, την getAncestorsByTag() η οποία επιστρέφει όλους τους προγόνους με ένα συγκεκριμένο tag name, την copyChildren() η οποία αντιγράφει τους child nodes από ένα source node σε

ένα destination node και προσφέρει την επιλογή για trimming των text nodes, καθώς και πολλές άλλες [9].

Το Dojo παρέχει πρόσθετα λειτουργικά χαρακτηριστικά για την προσθήκη back και forward buttons στον browser επιτρέποντας στο developer να προσδιορίσει κώδικα που να εκτελείται όταν ο χρήστης κάνει κλικ σε αυτά τα buttons. Επίσης υποστηρίζει bookmarking με την changeURL παράμετρο του io.bind(). Η παράμετρος αυτή καθιστά δυνατή την προσθήκη ενός string στο URL [9]:

```
dojo.io.bind({
    url: "http://example.com/form.php",
    load: processResults,
    changeURL: "contactInfo"
});
```

3.4 Η δομή του πηγαίου κώδικα

Όπως αναφέραμε και νωρίτερα το module decomposition για το Dojo toolkit χωρίζεται σε τρία δέντρα [7]:

- Στο Dojo που είναι το κύριο toolkit και περιλαμβάνει βιβλιοθήκες που είναι χρήσιμες σε σχεδόν όλα τα προγράμματα. Τα περιεχόμενα αυτού του δέντρου είναι γνωστά ως Dojo Core.
- Στο Dijit (Dojo widgets), το οποίο είναι ένα framework για την κατασκευή HTML user interface controls (δηλαδή widgets) καθώς επίσης και μία βιβλιοθήκη από πολλά προκατασκευασμένα widgets. Το Dijit εξαρτάται άμεσα από το Dojo.
- Στο Dojox (Dojo extensions) που περιλαμβάνει projects που μπορεί να μην είναι απαραίτητα σε κάθε μία εφαρμογή ή που μπορεί να μην είναι 100% σταθερά ακόμη επειδή εξωθούν στα άκρα τις δυνατότητες των σημερινών browsers (πχ

GFX). Πολλά από τα projects στο Dojox θα μπορούσαν να θεωρηθούν highly-mature στα πλαίσια άλλων toolkits. Όμως το Dojo βάζει τον πήχη τόσο ψηλά που για τα projects είναι σε πολλές περιπτώσεις ευκολότερο να συνεχίσουν να 'ωριμάζουν' ως extensions. Κάθε project στο Dojo περιλαμβάνει ένα readme αρχείο που εξηγεί την κατάσταση του project.

Το root directory του dojo/tree περιλαμβάνει το αρχείο dojo.js, ένα αρχείο που κάθε script element πρέπει να φορτώσει προτού γίνει οποιαδήποτε άλλη διεργασία με το toolkit. Το dojo.js είναι το module primary script για το module dojo. Όταν γίνεται evaluated το dojo.js δημιουργείται η global μεταβλητή dojo (ένα αντικείμενο), και προστίθενται σε αυτό ιδιότητες που αναπληρώνουν τον ελάχιστο χρόνο εκτέλεσης που ορίζεται από το Dojo.

Πρόσθετα στο dojo.js η ρίζα του dojo/tree ορίζει και ορισμένα άλλα modules. Κάθε ένα από αυτά είναι ένας απόγονος του dojo module, αφού τα modules μπορεί να περιέχουν και άλλα modules. Για παράδειγμα, το module primary script number.js είναι τοποθετημένο στο root directory του dojo/tree και ορίζει το module dojo.number. Με αυτή την έννοια το dojo.js είναι διαφορετικό από κάθε άλλο αρχείο στο root, το dojo.js προκαλεί τη δημιουργία του module dojo, και όλα τα άλλα 'αδέρφια' του dojo.js προκαλούν τη δημιουργία modules που είναι απόγονοι του dojo (για παράδειγμα το dojo.number). Παρομοίως το root directory του Dijit tree περιλαμβάνει το αρχείο dijit.js το οποίο παίζει το ρόλο του entry point για το Dijit και επιδεικνύει το ίδιο είδος εξειδικευμένης συμπεριφοράς.

Όλα τα directory trees, κατά το δυνατό, πιάνουν τα dependencies. Έτσι για παράδειγμα ισχύει πάντοτε ότι για κάποια τυχαία modules dojo.x και dojo.x.y το dojo.x.y εξαρτάται από το module dojo.x. Φυσικά δεν υπάρχει λόγος να μάθει κανείς όλα τα module

dependencies αφού κάθε module θα εξασφαλίσει ότι θα φορτωθούν όλα τα προαπαιτούμενα modules του [7].

3.4.1 Τα Dojo Modules

Μέσα στο Dojo υπάρχει το υπάρχει το module dojo το οποίο περιλαμβάνει τις ακόλουθες λειτουργίες [7]:

1. Environment properties: Ένα μικρό σύνολο από μεταβλητές που αναδεικνύουν διάφορες ιδιότητες του runtime περιβάλλοντος (για παράδειγμα την έκδοση και τις δυνατότητες του browser)
2. Language extensions: Συναρτήσεις που συμπληρώνουν ορισμένα στοιχεία που λείπουν από την κύρια JavaScript βιβλιοθήκη
3. Asynchronous programming: Συναρτήσεις για προγραμματιστικά events και ασύγχρονες αναδρομικές κλήσεις
4. DOM programming: Μία βιβλιοθήκη από συναρτήσεις που κάνει πολύ πιο ευχάριστο τον προγραμματισμό του DOM
5. XHR programming: Συναρτήσεις για επικοινωνία με τον server διαμέσου κάποιων XHR αντικειμένων
6. Αντικειμενοστρεφής προγραμματισμός: Μία συνάρτηση για τη δημιουργία ισχυρών, ευέλικτων ιεραρχιών κλάσεων
7. Ο Dojo loader: Συναρτήσεις που φορτώνουν JavaScript scripts και Dojo modules.

Το Dojo/tree περιλαμβάνει αρκετά άλλα modules που δεν φορτώνονται αυτόματα ως αποτέλεσμα του γεγονότος ότι το dojo.js φορτώνεται σε ένα script tag, και τα οποία θα πρέπει να φορτωθούν μέσω της κλήσης μίας συνάρτησης dojo.require. Σε αυτά τα modules περιλαμβάνονται τα παρακάτω [7]:

- dojo.back: υλοποιεί τη λειτουργία του back button για κάποιες εφαρμογές του browser
- dojo. behavior: ‘συμπεριφορές’ που μπορεί να προστεθούν σε σύνολα κόμβων

- dojo.cldr: υλοποίηση της Common Locale Data Repository (CLDR)
- dojo.colors: συναρτήσεις για τον εύκολο χειρισμό του CSS color
- dojo.cookie: απλή διαχείριση HTTP cookie
- dojo.currency: συναρτήσεις για νομισματικά δεδομένα
- dojo.data: συναρτήσεις για την πρόσβαση σε μία γενικευμένη πηγή δεδομένων μαζί με μία υλοποίηση για ορισμένους τύπους data sources
- dojo.date: συναρτήσεις για δεδομένα ημερομηνίας
- dojo.dnd: DOM drag-and-drop
- dojo.fx: DOM effects
- dojo.i18n: κάνει support σε πολλαπλά locales, ενώ δουλεύει μαζί με το dojo.cldr
- dojo.io: συναρτήσεις για επικοινωνία με έναν server μέσω script και/ή iFrame elements
- dojo.number: συναρτήσεις για αριθμητικά δεδομένα
- dojo.parser: HTML parser
- dojo.regexp: συναρτήσεις που βοηθούν στην δημιουργία regular expressions
- dojo.rpc: πλαίσιο για Remote Procedure Calls (RPC)
- dojo.string: κάποιες συνηθισμένες string functions

3.4.2 Τα Dijit Modules

Τα Dijit modules περιέχονται σε ένα χωριστό source code tree, ορίζονται από το root module dijit και όχι από το dojo.dijit και υλοποιούν το HTML widget system του Dojo [7].

3.5 Widgets

Τα Dijit components περικλείουν και ενσωματώνουν με εύχρηστο τρόπο τη λειτουργικότητα του user interface και αυτό είναι το χαρακτηριστικό που έκανε την

τεχνολογία που είναι βασισμένη σε Visual Basic components τόσο ελκυστική είκοσι χρόνια πριν. Αυτή η δύναμη έχει χαθεί στην εποχή του Διαδικτύου και εάν κάποιος αναζητά αυτή την ευκολία και την τμηματοποίηση κατά το development θα βρει αυτή τη μέθοδο προγραμματισμού οικεία και ενδιαφέρουσα [7].

Με την αφηρημένη έννοια τα widgets είναι components αλλά πώς είναι όσον αφορά τη συγγραφή κώδικα; Μέχρι τώρα γνωρίζαμε ότι ένα widget είναι ένα τμήμα HTML που έχει dojoType:

```
<div dojoType="dijit.layout.ContentPane"
    href="http://localhost/too/many/slashes.html" ></div>
```

Η dijit.layout.ContentPane είναι στην πραγματικότητα μία Dojo class. Οι μέθοδοί της στέλνουν Dojo API κλήσεις να διαχειριστούν την επικοινωνία με τον browser, το DOM tree και τον server και ο προγραμματιστής μπορεί να γράψει JavaScript για να καλέσει αυτές τις μεθόδους.

Στη συνέχεια παραθέτουμε κάποιους ορισμούς για να κατανοήσουμε καλύτερα την σημαντικότερη ορολογία [7].

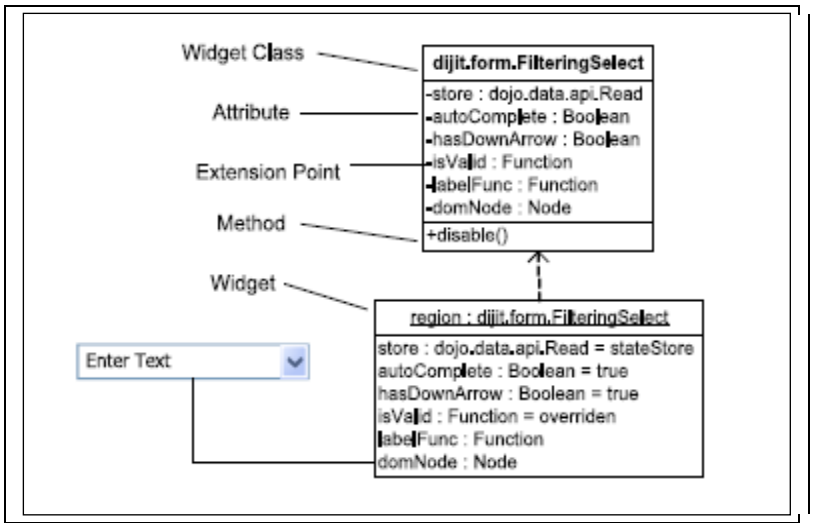
Widget class: Μία Widget class είναι οποιαδήποτε κλάση προέρχεται από το dijit_Widget. Μπορεί να είναι άμεση υπο-κλάση του dijit_Widget ή μπορεί να είναι υπο-κλάση ενός άλλου widget.

Widget: Ένα widget είναι ένα JavaScript αντικείμενο, ένα instance μίας widget class. Εάν μία σελίδα έχει για παράδειγμα δώδεκα validation textboxes τότε έχει δώδεκα widgets, και όλα αποτελούν instances της widget class dijit.form.ValidationTextBox.

Attribute: Το attribute αποτελεί property σε μία widget class. Ο όρος attribute οφείλεται στην ορολογία της HTML. Αρχικοποιείται κατά το χρόνο δημιουργίας του widget αλλά δεν είναι έτοιμο προς διαχείριση αμέσως μετά. Κατά συνέπεια δεν μπορεί κανείς να γράψει *myWidget.disabled = true* για να απενεργοποιήσει ένα widget, και αντί γι αυτό θα πρέπει να χρησιμοποιήσει μία method call όπως η myWidget.disable().

Extension point: Ένα extension point είναι μία μέθοδος που ο προγραμματιστής μπορεί να την παρακάμψει. Για παράδειγμα, μπορεί να παρακάμψει το extension point isValid της dijit.form.ValidationTextBox με μία δική του συνάρτηση. Η συνάρτηση αυτή μπορεί να εφαρμοστεί σε ένα widget χρησιμοποιώντας διάφορες τεχνικές, ή σε μία ολόκληρη widget class. Όσον αφορά τη JavaScript τόσο οι methods όσο και τα extension points είναι απλώς widget class methods. Η διαφορά έγκειται στο ότι οι μέθοδοι έχουν πλήρη υλοποίηση ενώ οι extension point methods έχουν φτωχή ή κενή υλοποίηση. Μερικά extension points παίρνουν το όνομά τους από events όπως το onClick ή το onChange. Αυτά τα extension points παιρνούν περισσότερο εξειδικευμένες παραμέτρους απότι απλώς ένα μεγάλο DOM Level 2 event object, όπως θα έκανε ένα αντίστοιχο event.

Στην Εικόνα 3.1 μπορεί κανείς να δει το widget region και την αντίστοιχη widget class με UML σημειογραφία [7].



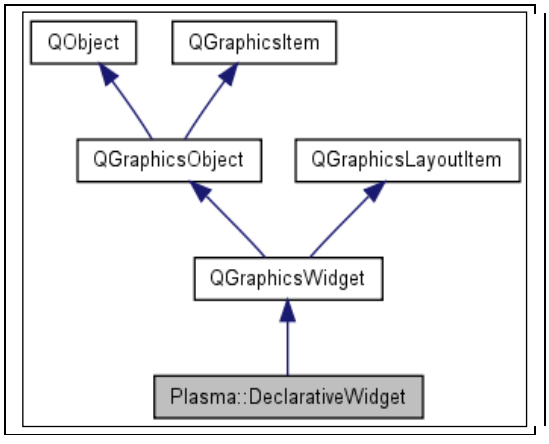
Εικόνα 3.1 Οι σχέσεις μεταξύ των widget elements

3.5.1 Declarative Widgets

Τα δηλωτικά (declarative) widgets αποτελούν μία κατηγορία widgets που έχουν δημιουργηθεί με HTML markup και με ένα attribute *dojoType*=. Υπάρχουν επίσης τα programmatic widgets τα οποία διαφέρουν από τα πρώτα μόνο στον τρόπο που δημιουργούνται και μόλις ολοκληρωθεί το setup δεν υπάρχει καμία απολύτως διαφορά ανάμεσα στις δύο κατηγορίες. Θα πρέπει να παρατηρήσουμε επίσης ότι οποιαδήποτε widget class μπορεί να δημιουργήσει είτε declarative είτε programmatic widgets [7].

Ο parser του Dojo μετατρέπει τα declarative widgets σε JavaScript αντικείμενα, σκανάροντας όλο το HTML source, συλλέγοντας όλα τα tags που έχουν dojoType= attribute και δημιουργώντας με αυτόν τον τύπο ένα widget object [7].

Αν και τεχνικά είναι λανθασμένο, είναι αρκετά δελεαστικό να αποκαλεί κανείς το HTML tag που έχει dojoType= attribute ως widget. Παρά το ότι υπάρχει μία ένα-προς-ένα αντιστοιχία μεταξύ των tags και των declarative widgets, το tag είναι ένα JavaScript DOM object. Ο DOM node έχει μία συγκεκριμένη style property την οποία το widget δεν έχει. Επίσης, τα widgets έχουν widget class μεθόδους όπως η setLabel ενώ οι DOM nodes δεν έχουν. Γνωρίζουμε τέλος ότι η μετατροπή της μίας κατηγορίας στην άλλη δεν είναι δύσκολη.



Εικόνα 3.2 Ένα inheritance διάγραμμα για το declarative plasma widget

3.5.2 Programmatic Widgets

Στην ουσία τα programmatic widgets παραλείπουν το parsing step που χρησιμοποιείται στα declarative widgets. Η δημιουργία ενός widget προγραμματιστικά γίνεται καλώντας τον καινούριο τελεστή σε μία κλάση widget και αναθέτοντας το αποτέλεσμα σε μία μεταβλητή. Δεν υπάρχει στην πραγματικότητα καμία διαφορά μεταξύ του να γίνει η διαδικασία αυτή από τον προγραμματιστή και του να γίνει από τον Dojo parser όπως γίνεται δηλαδή για τα declarative widgets [7].

Τα programmatic widgets είναι καλύτερα όποτε χρειαζόμαστε ακαθόριστο αριθμό από widgets. Επιπλέον τα programmatic widgets μπορούν να χρησιμοποιήσουν JavaScript εκφράσεις για να αρχικοποιήσουν τα attributes ενώ τα declarative δεν μπορούν [7].

3.6 Trees

Γνωρίζουμε ότι εκτός από ομαλά data stores με στοιχεία που έχουν μόνο πρωτεύοντα, απλά χαρακτηριστικά το dojo.data είναι πιθανό να χρειαστεί να δουλέψει και με σύνθετα δεδομένα. Για παράδειγμα τα XML data sources μπορεί να έχουν εμφωλευμένα elements και τα JSON data stores μπορεί να έχουν εμφωλευμένα arrays και hashes.

Οι άνθρωποι φαίνεται ότι προτιμούν τα δέντρα (trees) για την οπτικοποίηση ιεραρχικών δεδομένων. Τα user interfaces σε προγράμματα ηλεκτρονικής αλληλογραφίας αλλά και το file manipulation portion ενός λειτουργικού συστήματος χρησιμοποιούν δέντρα για να δείξουν στοιχεία που περιέχονται μέσα σε φακέλους, οι οποίοι με τη σειρά τους περιέχονται σε άλλους φακέλους κ.ο.κ. Το tree widget του Dijit που λέγεται dijit.Tree απλοποιεί αρκετά αυτή την οπτικοποίηση αφού η ισχυρή ενσωμάτωση με το dojo.data εμφωλευμένων data sources διασφαλίζει την ακριβή αναπαράσταση στην οθόνη σε οποιαδήποτε στιγμή. Αξιοποιώντας τα πολλά extension

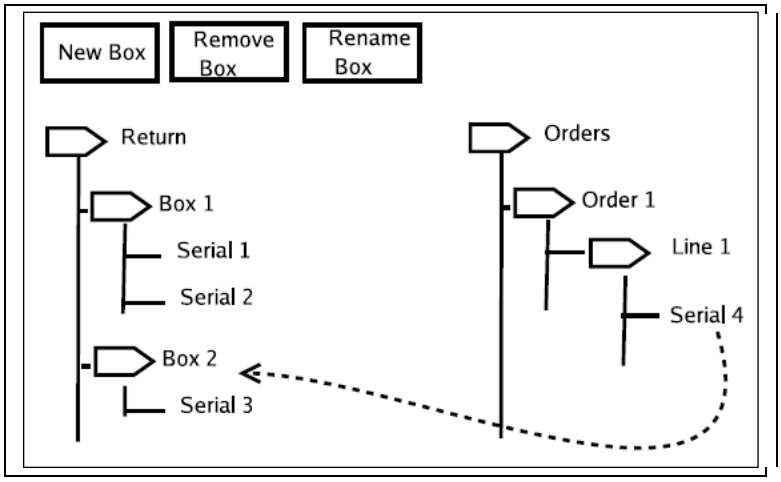
points του dijit.Tree καθώς και τη λειτουργία drag-and-drop που διαθέτει, μπορεί κανείς να δημιουργήσει μία δυνατή εμπειρία για το χρήστη [7].

Στη συνέχεια θα δημιουργήσουμε μία web εφαρμογή για Return Material Authorizations (RMAs) για να επιδείξουμε την προσαρμοστικότητα των Trees. Υποθέτουμε την ύπαρξη ενός επιχειρηματία ο οποίος ειδικεύεται στα δώρα-έκπληξη και σπάνια του επιστρέφει κάποιος ένα δώρο και ο οποίος επιθυμεί να κάνει τη διαδικασία επιστροφής γρήγορη και ανώδυνη. Ο επιχειρηματίας πληρώνει όλα τα έξοδα της επιστροφής επιλέγοντας την εταιρεία αποστολής της αρεσκείας του. Η εταιρεία αποστολής έχει ένα web service για να κανονίζει τη συλλογή των δώρων και τη δημιουργία ετικετών για την αποστολή. Η εταιρεία αποστολής έχει επίσης ορισμένους περιορισμούς ο σημαντικότερος από τους οποίους είναι το όριο των 15 ευρώ για κάθε πακέτο. Ο επιχειρηματίας θα σιαχειριστεί το shipper integration και εμείς θα πρέπει να φτιάξουμε το User Interface. Ένας πελάτης θα κάνει log-in στο site, θα δει τη λίστα των παραγγελιών του μαζί με τους σειριακούς αριθμούς και θα επιλέξει τα είδη για το RMA.

Θα σχεδιάσουμε το παραπάνω με δύο δέντρα (trees). Το δεξί tree, το οποίο θα αποκαλείται order tree, θα περιέχει τις πρόσφατες παραγγελίες, τα είδη που παραγγέλθηκαν κάθε φορά καθώς και τους σειριακούς αριθμούς. Το αριστερό tree, το οποίο θα αποκαλείται box tree, θα περιέχει κουτιά από σειριακούς αριθμούς προς επιστροφή. Ο χρήστης θα μπορεί να σύρει είδη που θέλει να επιστρέψει από το order tree προς το box tree. Η εφαρμογή θα πρέπει να είναι σε θέση να αποτρέψει το χρήστη από τη δημιουργία κουτιών που να είναι πιο ακριβά από 15 ευρώ. Ο χρήστης μπορεί επίσης να δημιουργήσει, να ανανεώσει και να διαγράψει κουτιά καθώς τα μετακινεί για να ταιριάξουν στο δικό του packing list. Στην Εικόνα 3.3 μπορεί κανείς να δει μία αναπαράσταση της εφαρμογής [7].

Σε πρώτη φάση μοντελοποιούμε τους κύριους κόμβους του order tree. Το δέντρο αποτελεί ένα dijit.Tree widget το οποίο απαιτεί για την λειτουργία του ένα dojo.data

store. Κάθε στοιχείο στο data store γίνεται κόμβος του δέντρου. Ο κορυφαίος κόμβος καλείται ρίζα (root), δεν έχει αντίστοιχο item (στοιχείο) στο data store και ο προγραμματιστής θα πρέπει να παρέχει το root label στο tag dijit.tree.ForestStoreModel. Υπάρχει επίσης η δυνατότητα το tree να μην έχει καθόλου ρίζα θέτοντας στο dijit.Tree widget το showRoot=”false”.



Εικόνα 3.3 Το διάγραμμα της εφαρμογής της επιστροφής

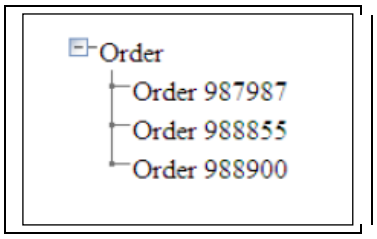
Μπορούμε να χρησιμοποιήσουμε οποιονδήποτε dojo.data driver για να κάνει feed το δέντρο, όμως στο παρακάτω παράδειγμα θα χρησιμοποιήσουμε το πιο γνωστό το ItemFileReadStore. Ένα στατικό αρχείο θα παρέχει τα sample data και στη συνέχεια θα αντικατασταθεί από μία δυναμική υπηρεσία. Στη συνέχεια παραθέτουμε το datasource/order.json [7]:

```
{
   identifier: 'id',
   label: 'description',
   items: [
          { id: 987987, description:"Order 987987",
                priority:"Next Day Air"
          },
          { id: 988855, description:"Order 988855",
                priority:"2nd Day Air"
```

```
        },
        { id: 988900, description:"Order 988900" ,
                priority:"2nd Day Air"
}]}
```

Με αυτό το data source οι παρακάτω πέντε γραμμές κώδικα θα δημιουργήσουν την Εικόνα 3.4.

```
<div dojoType="dojo.data.ItemFileReadStore" url="datasources/order_header.json"
        jsId="ordJson" ></div>

<div dojoType="dijit.tree.ForestStoreModel" rootLabel="Order" store="ordJson"
        jsId="ordModel" ></div>

<div dojoType="dijit.Tree" id="ordTree" model="ordModel"></div>
```



Εικόνα 3.4 Ένα dijit.Tree widget

Η ροή των δεδομένων πραγματοποιείται από το data source έως το Tree σε τρία βήματα [7]:

- Το tag dojo.data.ItemFileReadStore ορίζει την πηγή των δεδομένων.
- Ένα μοντέλο λειτουργεί ως ο adapter που συνδέει το data source με το Tree. Η είσοδος του προέρχεται από το jsId= του data store και η έξοδος του περνάει από το δικό του jsId=. Σε αυτό το σημείο μπορεί να πραγματοποιηθεί φιλτράρισμα των δεδομένων μέσω του dojo.data query. Κάθε δέντρο έχει δύο adapters, ο πιο γνωστός εκ των οποίων είναι ο dijit.tree.ForestStoreModel ο οποίος παίρνει ένα store με πολλά στοιχεία, κάνει όλα τα στοιχεία παιδιά και προσθέτει ένα τεχνητό root node. Η ετικέτα αυτού του root node καθορίζεται στο attribute root-Label=.

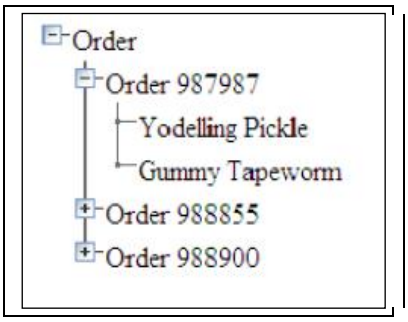
Το dijit.tree.TreeStoreModel παίρνει το root από το store και περιμένει το query dojo.data store να επιστρέψει ένα στοιχείο, με προφανώς πολλά στοιχεία-παιδιά.

- Το dijit.Tree tag συνδέεται στο jsld= του μοντέλου μέσω του attribute model=.

Ιεραρχική αποθήκευση δεδομένων

Το προηγούμενο παράδειγμα αναφερόταν σε δέντρο ενός επιπέδου. Εάν το backing data store υποστηρίζει ιεραρχία των δεδομένων, όπως για παράδειγμα τα ItemFileReadStore και το XMLStore, τότε ένα Tree μπορεί να απαριθμήσει οποιονδήποτε αριθμό επιπέδων. Η εμφώλευση αυτή ελαττώνει τον αριθμό των επισκέψεων στον server. Για παράδειγμα, μία παραγγελία έχει header πληροφορίες (συμπεριλαμβάνοντας τον αριθμό της παραγγελίας και τον πελάτη) και μηδενικά σε πολλές γραμμές (πχ την ποσότητα). Ο προγραμματιστής μπορεί αντί να χρησιμοποιήσει δύο ξεχωριστά requests για την επικεφαλίδα και για τις σχετιζόμενες γραμμές να συμπεριλάβει τις γραμμές μέσα στην επικεφαλίδα [7].

Στην Εικόνα 3.5 βλέπουμε ένα δέντρο όπου φαίνεται η ιεράρχηση.



Εικόνα 3.5 Ένα hierarchical tree

3.7 Κατασκευή της διεπαφής του χρήστη με το Dijit

Η πλούσια component βιβλιοθήκη του Dojo, το Dijit, επιτρέπει το χτίσιμο ολοκληρωμένων user interafaces με ελάχιστο κώδικα και ελάχιστη ταλαιπωρία. Τα Dijit components μπορεί να χρησιμοποιηθούν με δύο τρόπους: είτε προγραμματιστικά με τη χρήση JavaScript είτε δηλωτικά με τη χρήση HTML-style elements με attribute *dojoType*. Στην ενότητα αυτή θα χρησιμοποιήσουμε το δηλωτικό τρόπο και οποιοδήποτε JavaScript logic πίσω από αυτά τα widgets θα γίνει σε ξεχωριστό source file [10].

Ο κώδικας της βασικής εφαρμογής περιέχεται σε τρία αρχεία. Οι layout και widget δηλώσεις πραγματοποιούνται στο βασικό index.html αρχείο. Ένας υπο-φάκελος που ονομάζεται css περιέχει ένα μόνο αρχείο το style.css, το οποίο περιέχει όλα τα stylesheets που απαιτούνται για τη συγκεκριμένη εφαρμογή. Τέλος όλο το JavaScript logic αποθηκεύεται στο αρχείο script.js στον υποκατάλογο js. Στην ενότητα αυτή εξηγούμε πώς θα χτίσουμε το user interface για την εφαρμογή χρησιμοποιώντας HTML και Dojo widgets [10].

3.7.1 Η βασική δομή

Η εφαρμογή ξεκινάει με τη δημιουργία της βασικής HTML δομής. Αυτή περιλαμβάνει όλα τα σχετικά stylesheets, τον καθορισμό του Dijit theme που θα χρησιμοποιηθεί και το φόρτωμα όλων των JavaScript source αρχείων που θα είναι απαραίτητα. Για την εφαρμογή που δημιουργούμε σε αυτή την ενότητα, θα πρέπει να φορτώσουμε για ευκολία το Dojo από το CDN του Google. Κατά συνέπεια η εφαρμογή χρησιμοποιείται μόνο εφόσον υπάρχει διαθέσιμη σύνδεση στο Internet. Εάν κάποιος θέλει να χρησιμοποιήσει την εφαρμογή offline θα πρέπει απλώς να κατεβάσει την βιβλιοθήκη του Dojo και να την φορτώσει τοπικά. Οι γραμμές κώδικα που ακολουθούν δείχνουν το σημείο έναρξης του main αρχείου της εφαρμογής του index.html και όλη τη δομή του [10].

Όπως μπορούμε να δούμε η εφαρμογή φορτώνει τέσσερα stylesheets. Τα τρία από αυτά είναι από το ίδιο το Dojo: το πρώτο είναι το βασικό θέμα του CSS αρχείου για το Claro Dijit theme και τα άλλα δύο χρησιμοποιούνται για το Data Grid DojoX extension [10].

```
<!doctype html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>Dojo Contacts</title>
      <link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/dojo/1.5/dijit/themes/
claro/claro.css">
      <link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/dojo/1.5/dojox/grid/re
sources/Grid.css">
      <link rel="stylesheet"
href="https://ajax.googleapis.com/ajax/libs/dojo/1.5/dojox/grid/re
sources/claroGrid.css">
      <link rel="stylesheet" href="css/style.css">
   </head>
   <body class="claro">
      <!-- Content goes here -->
      <script type="text/javascript"
      src="https://ajax.googleapis.com/ajax/libs
/dojo/1.5/dojo/dojo.xd.js" djConfig="parseOnLoad: true"></script>
      <script type="text/javascript" src="js/script.js"></script>
   </body>
</html>
```

Ένα σημαντικό σημείο που μπορούμε να παρατηρήσουμε είναι ότι έχουμε προσθέσει μία κλάση claro στο κύριο <body> element αυτού του document και αυτή είναι που λέει στον Dojo parser ότι θα έπρεπε να χρησιμοποιήσει το Claro theme. Χωρίς αυτό η εφαρμογή δεν θα δείχνει σωστή οπότε είναι κάτι που δεν μπορούμε να παραλείψουμε. Στη συνέχεια φορτώνουμε το <script> blocks για την εφαρμογή στην κορυφή του αρχείου, ακριβώς πριν από το τελικό </body> element. Αυτός είναι ο γενικότερα προτεινόμενος τρόπος ενσωμάτωσης και συμπερίληψης των πηγαίων αρχείων

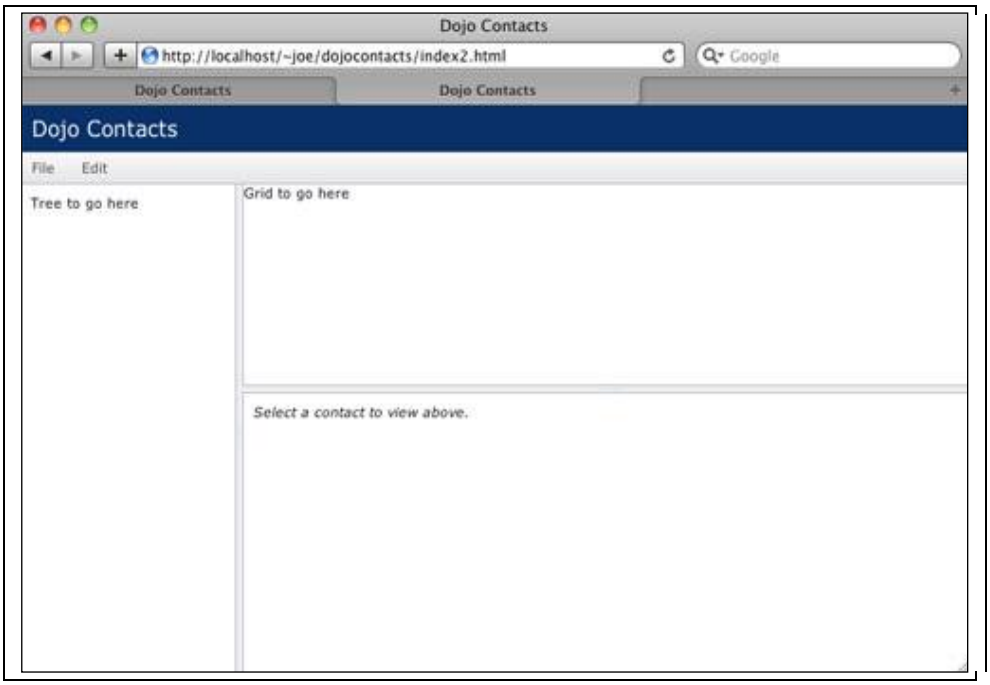
σε JavaScript, καθώς έτσι δεν θα εμποδιστεί το φόρτωμα της υπόλοιπης σελίδας προτού τα scripts έχουν ολοκληρώσει τη διαδικασία του loading.

3.7.2 Καθορισμός τη διάταξης της εφαρμογής

Στη συνέχεια δημιουργούμε την κύρια διάταξη της εφαρμογής. Ο στόχος είναι για την ώρα να καταφέρουμε να έχουμε το αποτέλεσμα που βλέπουμε στην Εικόνα 3.6 και μέχρι το τέλος της υποενότητας αυτής να μπορούμε να δούμε στον web browser ένα τέτοιο screenshot [10].

Για να χρησσιμοποιήσουμε τα Dijit components που χρησιμοποιήθηκαν στην εφαρμογή, θα πρέπει να πούμε στο Dojo να φορτώσει τα widgets χρησιμοποιώντας την dojo.require JavaScript συνάρτηση. Θα πρέπει λοιπόν να δημιουργήσουμε ένα αρχείο με το όνομα script.js και να το σώσουμε σε έναν υποκατάλογο με όνομα js στον φάκελο της εφαρμογής μας. Επίσης θα πρέπει στο αρχείο αυτό να προσθέσουμε τις παρακάτω γραμμές κώδικα.

```
dojo.require("dijit.dijit");
dojo.require("dijit.layout.BorderContainer");
dojo.require("dijit.layout.ContentPane");
dojo.require("dijit.MenuBar");
dojo.require("dijit.PopupMenuBarItem");
dojo.require("dijit.Menu");
dojo.require("dijit.MenuItem");
```



Εικόνα 3.6 Η διάταξη της βασικής εφαρμογής

Οι παραπάνω γραμμές κώδικα λένε στο Dojo ότι θα πρέπει να φορτώσει τα layout widgets BorderContainer και ContentPane, καθώς επίσης και διάφορα Menu widgets που θα χρησιμοποιηθούν στο menu bar της εφαρμογής. Εφόσον φορτωθούν αυτά τα στοιχεία, μπορούμε να τα προσθέσουμε στο index.html αρχείο της εφαρμογής μας. Μέσα στο αρχείο θα πρέπει απλώς να αντικαταστήσουμε το σχόλιο <!- - Content goes here - -> με τις παρακάτω γραμμές κώδικα [10].

Στις παρακάτω γραμμές προσθέτουμε Dijit components στη σελίδα μας χρησιμοποιώντας standard HTML elements με ένα ειδικό dojoType attribute. Αυτές οι δηλώσεις των widget χρησιμοποιούν επίσης ορισμένα χαρακτηριστικά, ευδιάκριτα attributes που είναι κατάλληλα για το συγκεκριμένο τύπο widget, για παράδειγμα το region attribute χρησιμοποιείται για να ορίσει σε ποιο σημείο θα πρέπει να φαίνεται ένα child element του BorderContainer στα περιεχόμενα του container [10].

```
<div dojoType="dijit.layout.BorderContainer" design="header" gutters="false"
liveSplitters="true" id="borderContainer">
    <div dojoType="dijit.layout.ContentPane" region="top" id="topBar">
        <h1>Dojo Contacts</h1>
        <div dojoType="dijit.MenuBar" id="navMenu">
            <div dojoType="dijit.PopupMenuBarItem">
                <span>File</span>
                <div dojoType="dijit.Menu" id="fileMenu">
                    <div dojoType="dijit.MenuItem"
```

```
jsId="mnuNewContact">New Contact</div>
                    <div dojoType="dijit.MenuItem"
jsId="mnuNewGroup">New Group</div>
                </div>
            </div>
            <div dojoType="dijit.PopupMenuBarItem">
                <span>Edit</span>
                <div dojoType="dijit.Menu" id="editMenu">
                    <div dojoType="dijit.MenuItem" jsId="mnuEditContact"
disabled="true">Edit Contact</div>
                    <div dojoType="dijit.MenuItem" jsId="mnuMoveContact"
disabled="true">Move Contact</div>
                    <div dojoType="dijit.MenuItem" jsId="mnuRenameGroup"
disabled="true">Rename Group</div>
```

```
                    <div dojoType="dijit.MenuSeparator"></div>
                    <div dojoType="dijit.MenuItem" jsId="mnuDeleteContact"
disabled="true">Delete Contact</div>
                    <div dojoType="dijit.MenuItem" jsId="mnuDeleteGroup"
disabled="true">Delete Group</div>
                </div>
            </div>
        </div>
    </div>
    <div dojoType="dijit.layout.BorderContainer" region="center" gutters="true"
liveSplitters="true" id="mainSection">
        <div dojoType="dijit.layout.ContentPane" splitter="true" region="left"
id="treeSection">
            Tree to go here
        </div>
        <div dojoType="dijit.layout.BorderContainer" design="header" gutters="true"
liveSplitters="true" id="mainContainer" region="center">
            <div dojoType="dijit.layout.ContentPane" region="top" splitter="true"
id="gridSection">
                Grid to go here
            </div>
            <div dojoType="dijit.layout.ContentPane" id="contactView"
jsId="contactView" region="center">
```

```
                <em>Select a contact to view above.</em>
            </div>
        </div>
    </div>
</div>
```

Υπάρχουν αρκετά components στον παραπάνω κώδικα και στη συνέχεια θα ρίξουμε μια ματιά στη βασική δομή των components [10].

- dijit.layout.BorderContainer
    1. dijit.layout.ContentPane (top)
- h1
- dijit.MenuBar
- dijit.PopupMenuBarItem
    1. dijit.Menu
- dijit.MenuItem
- dijit.MenuItem
- dijit.PopupMenuBarItem
    1. dijit.Menu
- dijit.MenuItem
- dijit.MenuItem
- dijit.MenuItem
    1. dijit.layout.BorderContainer (center)
- dijit.layout.ContentPane (left)
- dijit.layout.BorderContainer (center)
- dijit.layout.ContentPane (top)
- dijit.layout.ContentPane (center)

Στο top level υπάρχει ένα BorderContainer, το οποίο επιτρέπει στον προγραμματιστή να τοποθετήσει τα child components σε μία από τις πέντε περιοχές: στην κορυφή, στο κέντρο, αριστερά, δεξιά και στο κάτω μέρος. Κάτω από αυτό υπάρχουν δύο components, ένα ContenetPane στο επάνω μέρος και ένα BorderContainer στο κέντρο. Επειδή χρησιμοποιούνται μόνο δύο components , το component που βρίσκεται στο κέντρο θα καταλάβει όλο το χώρο που δεν χρησιμοποιείται από το component του επάνω μέρους. Στο επάνω τμήμα, υπάρχει ένα στάνταρ HTML h1 heading και μία Dijit Menu δομή. Στο κεντρικό τμήμα, υπάρχει ένα εμφωλευμένο BorderContainer component, με ένα ContentPane στα αριστερά, και άλλο ένα BorderContainer στο κέντρο (το οποίο καταλαμβάνει όσο χώρο δεν χρησιμοποιείται από το αριστερό component). Το Tree control θα μπει στο αριστερό ContentPane, ενώ το BorderContainet στο κέντρο έχει δύο child components –και τα δύο είναι ContentPanes- στην περιοχή που βρίσκεται επάνω και στο κέντρο. Στο σημείο αυτό είμαστε σε θέση να σώσουμε το αρχείο μας και να το φορτώσουμε στον web browser για να πάρουμε το αποτέλεσμα που φαίνεται στην Εικόνα 3.6.

3.7.3 Προσθήκη του Tree component

Στη συνέχεια προσθέτουμε ένα Tree component στην αριστερή πλευρά της εφαρμογής. Τα Dijit Trees είναι όπως αναφέραμε πολύπλοκα components και απαιτούν ορισμένα support components για να διασφαλιστεί η σωστή λειτουργία τους. Πρώτα χρειάζονται ένα data store που 'προσκολλάται' στο Identity API του Dojo και συνδέεται με κάποια JSON data source. Στο sample application, τα δεδομένα αυτά παράγονται από το PHP script data/groups.php, έτσι ώστε να μπορούμε να συνδέσουμε το data store του δέντρου με το API αυτό. Κατόπιν, το Tree χρειάζεται ένα model component για να ορίσει τη δομή του data store και τι δεδομένα θα είναι σε θέση να διαβάσει. Τέλος, προσθέτουμε το Tree component και το συνδέουμε σε αυτό το μοντέλο [10].

Για να ξεκινήσουμε θα πρέπει πρώτα να πούμε στο Dojo ποια components θα πρέπει να φορτώσει έτσι ώστε αυτά να είναι διαθέσιμα στην εφαρμογή. Προσθέτουμε συνεπώς τις παρακάτω δύο γραμμές κώδικα στο αρχείο script.js [10].

```
dojo.require("dojo.data.ItemFileWriteStore");
dojo.require("dijit.Tree");
```

Θα πρέπει σε αυτό το σημείο να εξηγήσουμε γιατί χρησιμοποιείται ένα writable store. Το Tree component του Dojo είναι αρκετά πολύπλοκο και στην πραγματικότητα δεν μπορεί να ανανεωθεί ασύγχρονα απευθείας (εκτός εάν το βάλουμε μέσα σε ένα ContentPane και ανανεώνουμε όλο το ContentPane), με αποτέλεσμα να χρειάζεται η σύνδεση με ένα writable store για την προσθήκη, την ανανέωση και τη διαγραφή στοιχείων.

Στη συνέχεια θα πρέπει να βρούμε το τμήμα 'Tree to go here' του index.html αρχείου μας και να το αντικαταστήσουμε με τον παρακάτω κώδικα. Ο κώδικας αυτός ορίζει το data store, το tree model καθώς και το ίδιο το tree και το τοποθετεί στην αριστερή πλευρά της εφαρμογής. Επίσης προσθέτει στο tree ένα context menu αν και τα

στοιχεία σε αυτή τη φάση θα είναι απενεργοποιημένα. Το context menu θα υλοποιηθεί σε άλλη υποενότητα [10].

```
<div dojoType="dojo.data.ItemFileWriteStore" jsId="groupsStore"
url="data/groups.php"></div>
<div dojoType="dijit.tree.TreeStoreModel" jsId="groupsModel"
childrenAttrs="groups" store="groupsStore" query="{id:0}"></div>
<div dojoType="dijit.Tree" id="groupsTree" jsId="groupsTree"
model="groupsModel">
   <div dojoType="dijit.Menu" targetNodeIds="groupsTree">
         <div dojoType="dijit.MenuItem" jsId="ctxMnuRenameGroup"
disabled="true">Rename Group</div>
         <div dojoType="dijit.MenuItem" jsId="ctxMnuDeleteGroup"
disabled="true">Delete Group</div>
   </div>
</div>
```

Στο παραπάνω τμήμα κώδικα παρατηρούμε ότι το attribute jsId χρησιμοποιείται σε κάθε component. Το attribute αυτό στην ουσία καλεί τον Dojo parser να δημιουργήσει μεταβλητές JavaScript για κάθε ένα από αυτά τα components και τα ονομάζει με την τιμή που τους ανατίθεται εδώ. Έτσι γίνεται πολύ ευκολότερη η διαχείριση αυτών των components αργότερα καθώς εξαλείφεται η ανάγκη χρήσης του dijit.byId για να κάνουμε query στο DOM στην προσπάθεια να βρούμε αυτά τα widgets. Επίσης βελτιώνεται σημαντικά η απόδοση καθώς το DOM querying μπορεί να είναι ακριβό σε όρους ταχύτητας και απόκρισης της εφαρμογής ιδιαίτερα σε πιο αργούς web browsers [10].

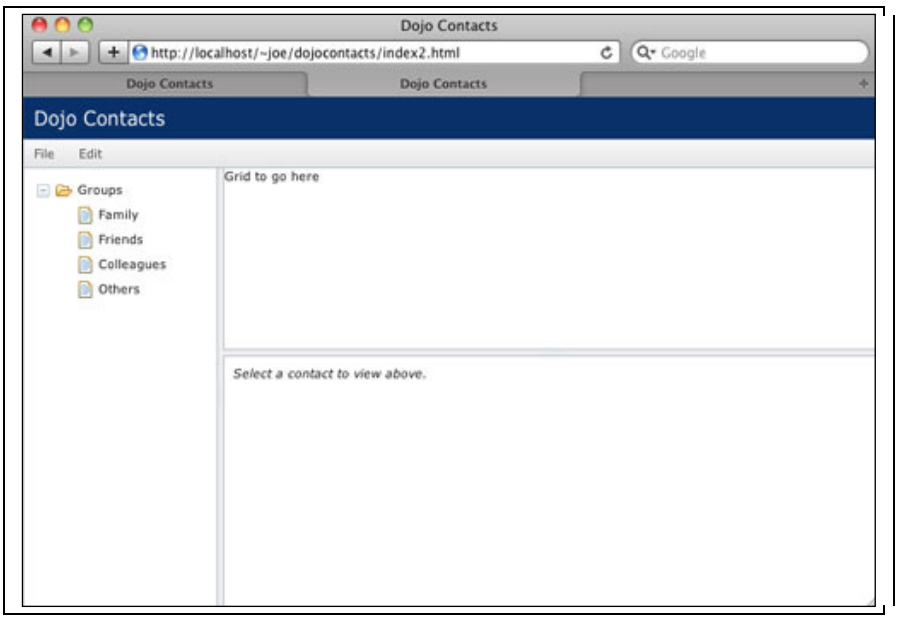
Εάν τώρα ξαναφορτώσουμε τον browser η εφαρμογή θα πρέπει να φαίνεται όπως το παράθυρο της Εικόνας 3.7 (υποθέτοντας ότι έχουμε αρχικοποιήσει σωστά τα PHP scripts και ότι τα δεδομένα για το δέντρο πράγματι φορτώνονται).

3.7.4 Προσθήκη του Grid component

Η διαδικασία της προσθήκης ενός grid στην εφαρμογή θα έπρεπε να μοιάζει με αυτήν την προσθήκης ενός tree. Σε αντίθεση με το tree όμως το grid μπορεί να επαναφορτωθεί ασύγχρονα με την ανανέωση του data store, έτσι δεν απαιτείται κανένα μοντέλο και μπορεί με ασφάλεια να χρησιμοποιηθεί ένα read-only store. Αρχικά θα

πρέπει να προσθέσουμε την ακόλουθη γραμμή κώδικα στο αρχείο script.js έτσι ώστε το grid component να είναι διαθέσιμο για χρήση στην εφαρμογή [10]:

```
dojo.require("dojox.grid.DataGrid");
```



Εικόνα 3.7 Το Dijit Tree

Θα πρέπει επίσης να αναφέρουμε ότι το Data Grid widget περιλαμβάνεται στο DojoX και όχι στη Dijit βιβλιοθήκη. Τα DojoX components συχνά απαιτούν τη φόρτωση πρόσθετων stylesheets (σε αυτή την περίπτωση αυτά τα styles έχουν ήδη συμπεριληφθεί στο βασικό template που δημιουργήσαμε νωρίτερα).

Τώρα μπορούμε να προσθέσουμε το grid στο βασικό αρχείο της εφαρμογής. Τα data grids στο Dojo μπορούν να εφαρμοσθούν σε ένα στάνταρ <table> element, επιτρέποντάς μας να ορίσουμε εύκολα τα headings που θέλουμε να χρησιμοποιήσουμε. Το grid μπορεί να ορίσει μία σειρά από attributes τα οποία θα καθορίσουν τον τρόπο λειτουργίας του, για παράδειγμα το εάν θα πρέπει να ενεργοποιηθεί το χαρακτηριστικό

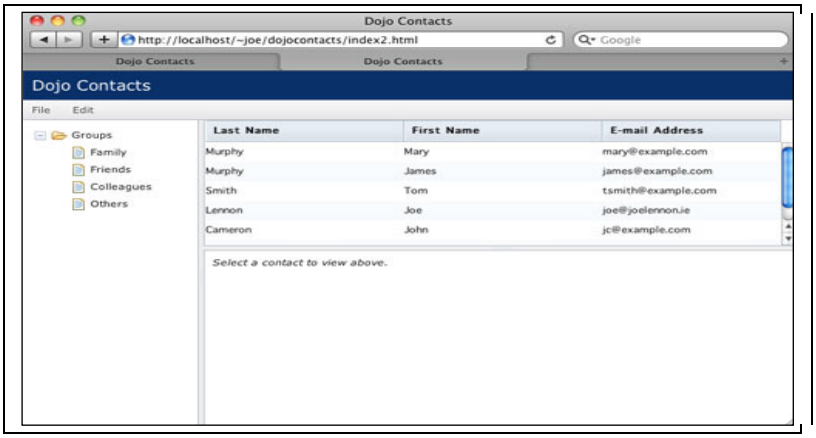
του client-side column sorting. Θα πρέπει στη συνέχεια να βρούμε την αναφορά ‘Grid to go here’ στο index.html αρχείο και να το αντικαταστήσουμε με τον κώδικα παρακάτω.

Το Grid όπως και το δέντρο προηγουμένως έχει επίσης διαθέσιμο context menu. Υπάρχει επίσης μία incine Dojo μέθοδος <script> block, η οποία οφείλει την ύοαρξή της σε ένα bug στο DataGrid component όπου το context menu δεν πρόκειται να εμφανίζεται χωρίς την ύπαρξη αυτού του άδειου block. Μπορούμε επίσης εφόσον θέλουμε να βάλουμε τον κώδικα JavaScript κατευθείαν μέσα στο layout code χρησιμοποιώντας αυτά τα blocks, αν και μάλλον είναι προτιμότερο να χωρίζουμε όπου είναι δυνατό το application logic από τα UI elements.

```
<span dojoType="dojo.data.ItemFileReadStore" jsId="contactsStore"
url="data/contacts.php"></span>
<table dojoType="dojox.grid.DataGrid" id="contactsGrid" jsId="contactsGrid"
columnReordering="true" sortFields="['last_name','first_name']" store="contactsStore"
query="{first_name: '*'}"
clientSort="true" selectionMode="single" rowHeight="25" noDataMessage="<span
class='dojoxGridNoData'>No contacts found in this group</span>">
    <thead>
        <tr>
            <th field="last_name" width="200px">Last Name</th>
            <th field="first_name" width="200px">First Name</th>
            <th field="email_address" width="100%">E-mail Address</th>
        </tr>
    </thead>

    <script type="dojo/method" event="onRowContextMenu" args="e">
    </script>
</table>
<div dojoType="dijit.Menu" id="gridMenu" targetNodeIds="contactsGrid">
    <div dojoType="dijit.MenuItem" jsId="ctxMnuEditContact"
disabled="true">Edit Contact</div>
    <div dojoType="dijit.MenuItem" jsId="ctxMnuMoveContact"
disabled="true">Move Contact</div>
    <div dojoType="dijit.MenuItem" jsId="ctxMnuDeleteContact"
disabled="true">Delete Contact</div>
</div>
```

Το grid φορτώνει τα δεδομένα χρησιμοποιώντας ένα read-only data store, το οποίο συνδέεται με το API data/contacts.php. Αυτό ταξινομεί αυτόματα τα δεδομένα στην πλευρά του χρήστη με το last name και στη συνέχεια με το first name. Το grid επιτρέπει να επιλέγεται μόνο μία γραμμή κάθε φορά και ορίζει ένα μήνυμα λάθους το οποίο εμφανίζεται εάν δεν βρεθούν καθόλου δεδομένα.

Σε αυτό το στάδιο η εφαρμογή μας θα πρέπει να δείχνει όπως η Εικόνα 3.8 (υποθέτουμε και αυτή τη φορά ότι έχουμε προσθέσει τα PHP scripts από τον πηγαίο κώδικα και έχουμε κάνει ρυθμίσει σωστά το server side) [10].



Εικόνα 3.8 Το Data Grid του Dojox

3.7.5 Κρυφά application components: dijit.Dialog windows

Σε αυτό το σημείο έχει πλέον ολοκληρωθεί η κύρια UI εφαρμογή [10]. Σε αυτή την υποενότητα θα προσθέσουμε μερικά κρυφά dialog windows που θα εμφανίζονται στο κύριο παράθυρο της εφαρμογής όταν επιλέγουμε μία επιλογή από το menu. Αυτά τα pop up παράθυρα τυπικά περιέχουν web forms που επιτρέπουν τον καθορισμό των διαφόρων ιδιοτήτων ενός group ή μίας επαφής, αλλά θα πρέπει επίσης να δημιουργηθεί ένα απλό παράθυρο με ένα OK μήνυμα το οποίο δείχνει πολύ καλύτερο από το στάνταρ JavaScript alert dialog box.

Υπάρχουν πέντε τέτοια dialogs συνολικά. Αυτά τα dialog windows περιέχουν διάφορα Dijit form πεδία μεταξύ των οποίων είναι τα ValidationTextBoxes, FilteringSelects και Buttons. Για να συμπεριλάβουμε όλα αυτά τα πρόσθετα widgets θα

πρέπει να τα φορτώσουμε χρησιμοποιώντας τη dojo.require, άρα θα πρέπει να προσθέσουμε τον παρακάτω κώδικα στο script.js file.

```
dojo.require("dijit.Dialog");
dojo.require("dijit.form.Form");
dojo.require("dijit.form.ValidationTextBox");
dojo.require("dijit.form.TextBox");
dojo.require("dijit.form.FilteringSelect");
dojo.require("dijit.form.Button");
```

Τώρα μπορούμε να αρχίσουμε να προσθέτουμε τα dialog boxes. Τα components αυτά θα πρέπει να μπουν έξω από την κύρια εφαρμογή BorderContainer για να εξασφαλίσουμε ότι θα εμφανίζονται πάνω από όλα τα άλλα elements στη σελίδα. Ξεκινάμε να τα προσθέτουμε ακριβώς πάνω από το πρώτο <script> element κοντά στο κάτω μέρος του αρχείου index.html.file. Πρώτα προσθέτουμε τα dialog boxes 'New Group' και 'Rename Group'. Ο κώδικας για αυτά τα παράθυρα φαίνεται παρακάτω.

```
<div id="newGroupDialog" jsId="newGroupDialog" dojoType="dijit.Dialog" title="Create
New Group" draggable="false">
    <div dojoType="dijit.form.Form" id="newGroupForm"
jsId="newGroupForm" action="data/new_group.php" method="post">
        <input type="hidden" name="new_group_ajax"
id="new_group_ajax" value="0">
        <label for="new_group_name">Group Name:</label>
        <input type="text" name="new_group_name"
id="new_group_name" required="true" dojoType="dijit.form.ValidationTextBox" />
        <button dojoType="dijit.form.Button" type="submit">Submit</button>
        <button dojoType="dijit.form.Button" jsId="newGroupCancel"
type="button">Cancel</button>
    </div>
</div>

<div id="editGroupDialog" jsId="editGroupDialog" dojoType="dijit.Dialog"
title="Rename Group" draggable="false">
    <div dojoType="dijit.form.Form" id="editGroupForm" jsId="editGroupForm"
action="data/edit_group.php" method="post">
        <input type="hidden" name="edit_group_ajax" id="edit_group_ajax"
value="0">
        <input type="hidden" name="edit_group_id" id="edit_group_id">
        <table cellspacing="4" cellpadding="4">
            <tr>
                <td><label for="edit_group_old">Old Group
Name:</label></td>
                <td><input type="text" name="edit_group_old" id="edit_group_old"
disabled="true" dojoType="dijit.form.TextBox" /></td>
```

```
            </tr>
            <tr>
                <td><label for="edit_group_name">New Group
Name:</label></td>
                <td><input type="text" name="edit_group_name"
id="edit_group_name" required="true" dojoType="dijit.form.ValidationTextBox"
style="margin-bottom: 6px" /></td>
            </tr>
            <tr>
                <td colspan="2" align="center">
                    <button dojoType="dijit.form.Button"
type="submit">Submit</button>
                    <button dojoType="dijit.form.Button" jsId="editGroupCancel"
type="button">Cancel</button>
                </td>
            </tr>
        </table>
    </div>
</div>
```

Το New Group dialog περιέχει μία φόρμα, την οποία θα υποβάλλει στο PHP script data/new_group.php. Αυτή η φόρμα περιέχει μία φόρμα ελέγχου ValidationTextBox, η οποία είναι ένα Dijit-styled text box με αυτόματο validation. Σε αυτή την περίπτωση ο έλεγχος έχει γίνει flagged ως υποχρεωτικός θέτοντας το απαιτούμενο attribute ως true. Το dialog περιέχει επίσης buttons για Submit και Cancel.

Το Rename Group dialog επιτρέπει στους χρήστες να αλλάξουν το όνομα μίας ομάδας. Εμφανίζει ένα read-only TextBox control με το εκάστοτε όνομα της ομάδας και παρέχει ένα ValidationTextBox για την παροχή καινούριου ονόματος στην ομάδα. Κατόπιν, προσθέτουμε το Move Contact dialog window, όπως φαίνεται στον παρακάτω κώδικα.

```
<div id="moveContactDialog" jsId="moveContactDialog" dojoType="dijit.Dialog"
title="Move Contact" draggable="false">
    <div dojoType="dijit.form.Form" id="moveContactForm" jsId="moveContactForm"
action="data/move_contact.php" method="post">
        <input type="hidden" name="move_contact_ajax"
id="move_contact_ajax" value="0">
        <input type="hidden" name="move_contact_id" id="move_contact_id">
        <table cellspacing="4" cellpadding="4">
            <tr>
                <td><label for="move_contact_name">Contact Name:
</label></td>
                <td><input type="text" name="move_contact_name"
id="move_contact_name" disabled="true" dojoType="dijit.form.TextBox" /></td>
            </tr>
            <tr>
                <td><label for="move_contact_old">Current
Group:</label></td>
                <td><input type="text" name="move_contact_old"
id="move_contact_old" disabled="true" dojoType="dijit.form.TextBox" /></td>
            </tr>
            <tr>
                <td><label for="move_contact_new">New Group:</label>
</td>
```

```
                <td><input dojoType="dijit.form.FilteringSelect"
name="move_contact_new" store="groupsStore" searchAttr="name" query="{type:'node'}"
id="move_contact_new" required="true" style="margin-bottom: 6px" /></td>
            </tr>
            <tr>
                <td colspan="2" align="center">
                    <button dojoType="dijit.form.Button"
type="submit">Submit</button>
                    <button dojoType="dijit.form.Button" jsId="moveContactCancel"
type="button">Cancel</button>
                </td>
            </tr>
        </table>
    </div>
</div>
```

Αυτό το dialog εμφανίζει το όνομα της επιλεγμένης επαφής και την ομάδα στην οποία ανήκουν αυτή τη στιγμή. Κατόπιν εμφανίζει μία drop-down λίστα FilteringSelect η οποία επιτρέπει στο χρήστη να επιλέξει την καινούρια ομάδα στην οποία θα μετακινήσει την επαφή. Αυτό το drop-down υποστηρίζεται στην πραγματικότητα από το ίδιο data store με το Tree control που εμφανίζει τη λίστα των ομάδων στην αριστερή πλευρά της διεπαφής. Παρόλα αυτά το query attribute του FilteringSelect component μας επιτρέπει να ‘πούμε’ στο Dojo να μην συμπεριλάβει το root elemet Groups, με συνέπεια ο χρήστης να μην μπορεί να το επιλέξει ως επιλογή για να μετακινήσει εκεί την επαφή [10].

Το επόμενο dialog είναι το EditContact dialog το οποίο στην πραγματικότητα χρησιμοποιείται τόσο για την προσθήκη νέων επαφών όσο και για την τροποποίηση των ήδη υπαρχόντων. Ο κώδικας για αυτό το dialog είναι ο παρακάτω [10]:

```
<div id="editContactDialog" jsId="editContactDialog" dojoType="dijit.Dialog"
title="Edit Contact" draggable="false">
    <div dojoType="dijit.form.Form" id="editContactForm" jsId="editContactForm"
action="data/edit_contact.php" method="post">
        <input type="hidden" name="edit_contact_ajax" id="edit_contact_ajax"
value="0">
        <input type="hidden" name="edit_contact_real_id"
id="edit_contact_real_id">
        <table cellspacing="4" cellpadding="4">
            <tr><td><label for="edit_contact_id">Contact ID:
</label></td>
            <td><input type="text" name="edit_contact_id" id="edit_contact_id"
disabled="true" dojoType="dijit.form.TextBox" /></td></tr>
            <tr><td><label for="move_contact_new">Group:
</label></td>
            <td><input dojoType="dijit.form.FilteringSelect"
name="edit_contact_group" store="groupsStore" searchAttr="name" query="{type:'node'}"
id="edit_contact_group" required="true" /></td></tr>
            <tr><td><label for="edit_contact_first_name">First Name:
</label></td>
            <td><input type="text" name="edit_contact_first_name"
id="edit_contact_first_name" required="true" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_last_name">Last Name:
</label></td>
            <td><input type="text" name="edit_contact_last_name"
id="edit_contact_last_name" required="true" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_email_address">
Email Address:</label></td>
            <td><input type="text" name="edit_contact_email_address"
id="edit_contact_email_address" required="true" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_home_phone">Home
Phone:</label></td>
            <td><input type="text" name="edit_contact_home_phone"
id="edit_contact_home_phone" required="false" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_work_phone">Work
Phone:</label></td>
            <td><input type="text" name="edit_contact_work_phone"
id="edit_contact_work_phone" required="false" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_twitter">Twitter:
</label></td>
            <td><input type="text" name="edit_contact_twitter"
id="edit_contact_twitter" required="false" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_facebook">Facebook:
</label></td>
            <td><input type="text" name="edit_contact_facebook"
id="edit_contact_facebook" required="false" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td><label for="edit_contact_linkedin">LinkedIn:
</label></td>
            <td><input type="text" name="edit_contact_linkedin"
id="edit_contact_linkedin" required="false" dojoType="dijit.form.ValidationTextBox"
/></td></tr>
            <tr><td colspan="2" align="center">
                <button dojoType="dijit.form.Button"
type="submit">Submit</button>
                <button dojoType="dijit.form.Button" jsId="editContactCancel"
type="button">Cancel</button>
            </td></tr>
        </table>
    </div>
</div>
```

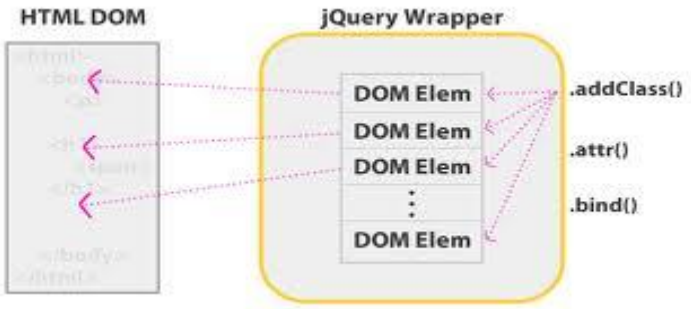
Η φόρμα αυτή έχει ένα display-only πεδίο για την Contact ID value, η οποία σε JavaScript θα λάβει την τιμή [NEW] στην περίπτωση δημιουργίας νέων επαφών (το ID θα δημιουργηθεί αυτόματα σε MySQL). Επίσης περιέχει form fields για το Group (το ίδιο drop-down με το Move Contact dialog), το μικρό όνομα, το επίθετο, την ηλεκτρονική διεύθυνση, το τηλέφωνο οικία και εργασίας, τον λογαριασμό στο Twitter, το λογαριασμό στο Face book και τον LinkedIn λογαριασμό.

Το τελευταίο dialog που πρέπει να δημιουργηθεί είναι ένα στάνταρ OK dialog box που θα χρησιμοποιηθεί ως το display window για το μήνυμα επιτυχίας ή λάθους κάθε φορά που πραγματοποιείται μία save operating με τη χρήση XHR/Ajax. Ο κώδικας για αυτό το dialog είναι ο παρακάτω [10]:

```
   <div id="okDialog" jsId="okDialog" dojoType="dijit.Dialog"
title="Title"
   draggable="false">
      <p id="okDialogMessage" style="margin-top: 5px">Message</p>
      <div align="center">
            <button dojoType="dijit.form.Button" jsId="okDialogOK"
   type="button">OK</button>
      </div>
   </div>
```

Δεν υπάρχει τίποτα περίπλοκο σε αυτό το dialog, απλώς περιέχει ένα μήνυμα και ένα Button για να το κλείσουμε. Με τη δημιουργία και αυτού του dialog έχει ολοκληρωθεί ο κώδικας της διεπαφής για την εφαρμογή. Μπορούμε να ξαναφορτώσουμε τη διεπαφή στον browser εφόσον επιθυμούμε αλλά δεν θα παρατηρήσουμε καμία αλλαγή σε σχέση με την Εικόνα 3.8 αφού αυτά τα dialogs πρέπει να συνδεθούν με τη χρήση JavaScript [10].

jQuery toolkit



4.1 Εισαγωγή στο jQuery

Ο πυρήνας της τεχνολογίας jQuery αποτελείται από την ομώνυμη βιβλιοθήκη JavaScript, σχεδιασμένη να επιφέρει νέα στοιχεία στον τρόπο που γράφεται ο κώδικας στο client side κομμάτι των Web Εφαρμογών. Η πρώτη έκδοση έγινε διαθέσιμη το 2006 από τον John Resig, αρκετά αργότερα δηλαδή σε σχέση με παρόμοιες τεχνολογίες. Παρόλα αυτά, σήμερα θεωρείται το πιο δημοφιλές εργαλείο JavaScript. Η τρέχουσα έκδοση, από το 2010, είναι η 1.4.2.

Τα ειδικά χαρακτηριστικά της βιβλιοθήκης, όπως θα περιγραφεί παρακάτω, καθιστούν πολύ εύκολη την επέκτασή της, με αποτέλεσμα τη δημιουργία εκατοντάδων επεκτάσεων (plug-ins) που προσφέρουν σημαντικές λειτουργίες και πρόσβαση σε έτοιμα ολοκληρωμένα διαδραστικά αντικείμενα διεπαφής, τα οποία μπορούν εύκολα να ενσωματωθούν σε σελίδες. Αυτά τα Plugins έχουν δημιουργηθεί από ανεξάρτητους προγραμματιστές καθώς η βιβλιοθήκη jQuery είναι προϊόν ανοιχτού κώδικα. Επίσης, δημιουργήθηκε η βιβλιοθήκη jQuery UI που επεκτείνει τον πυρήνα του jQuery, παρέχοντας λειτουργίες κυρίως χρήσιμες σε θέματα εμφάνισης, όπως οπτικές λειτουργίες, animation, κ.ά [11].

Συνεπώς, ο όρος jQuery Framework θα μπορούσε να αναφέρεται σε όλα τα παραπάνω, αντιπροσωπεύοντας έτσι έναν τεράστιο όγκο κώδικα με ενσωματωμένη λειτουργικότητα ικανή να εμπλουτίσει μια σελίδα Web εφαρμογής με ποικίλους τρόπους.

Φυσικά, η βιβλιοθήκη παρέχει πλήρη υποδομή για χειρισμό αντικείμενων DOM και μεθόδους για εκτέλεση ασύγχρονων Ajax αιτημάτων με εύκολο και αποδοτικό τρόπο. Ο πυρήνας της βιβλιοθήκης επικεντρώνεται, όπως και στην περίπτωση του Prototype, σε λειτουργίες ανάκτησης και διαχείρισης αντικειμένων DOM και των γεγονότων που συμβαίνουν πάνω σε αυτά, καθώς και στην παροχή διαδραστικής

λειτουργικότητας σε αυτά. Επιπλέον, εισάγει κάποια καινοτόμα στοιχεία που λύνουν προβλήματα και βελτιώνουν την ποιότητα του κώδικα.

Το πιο βασικό από αυτά τα στοιχεία αφορά στη δέσμευση των δημιουργών του jQuery να κινηθούν προς την κατεύθυνση της διακριτικής παρουσίας της JavaScript (Unobtrusive JavaScript), μια τάση που αφορά τον διαχωρισμό του κώδικα JavaScript από τα στοιχεία HTML της σελίδας. Παράλληλα, η jQuery παρέχει έναν ισχυρό μηχανισμό επιλογής και ανάκτησης αντικειμένων DOM, που έχει υιοθετηθεί από τον αντίστοιχο μηχανισμό της τεχνολογίας Cascading Style Sheets (CSS) [11].

Συγκριτικά με άλλα toolkits τα οποία επικεντρώνονται κυρίως στην εφαρμογή έξυπνων τεχνικών Javascript, το jQuery έρχεται να αλλάξει τον τρόπο που σκέφτονται οι web developers πάνω στην δημιουργία πλούσιων λειτουργικά σελίδων. Αντί να σπαταλούν χρόνο προσπαθώντας να ξεπεράσουν τις πολυπλοκότητες της JavaScript υψηλού επιπέδου, οι σχεδιαστές μπορούν να χρησιμοποιήσουν συνδιαστικά τις υπάρχουσες γνώσεις τους πάνω σε Cascading Style Sheets (CSS), Hypertext Markup Language (HTML), Extensible Hypertext Markup Language (XHTML) και στην γνωστή, άμεση JavaScript για να διαχειριστούν άμεσα τα page elements, κάνοντας έτσι το development πολύ γρηγορότερο [11].

Μπορεί κανείς να κατεβάσει την τελευταία έκδοση του jQuery από το site του: http://jquery.com/. Η εγκατάστασή του είναι εξαιρετικά απλή αφού αρκεί να το τοποθετήσουμε στο φάκελο της web εφαρμογής μας και να χρησιμοποιήσουμε το HTML <script> tag για να το συμπεριλάβουμε στις σελίδες μας:

```
<script type="text/javascript" src="scripts/jquery version.js">
</script>
```

4.2 Οικονομία στον κώδικα

Όποιος έχει καταναλώσει έστω και μία φορά χρόνο προσπαθώντας να προσθέσει δυναμική λειτουργικότητα σε κάποια σελίδα του γνωρίζει ότι θα πρέπει συνεχώς να ακολουθεί ένα συγκεκριμένο pattern επιλογής ενός element (ή ενός συνόλου από elements) και να ενεργεί αλλάζοντάς τα. Μπορεί για παράδειγμα να κρύβει ή να εμφανίζει τα elements, να τους προσθέτει μία CSS κλάση, να τους βάζει animations ή να επιθεωρεί τα attributes τους [14].

Η χρήση σκέτης JavaScript μπορεί να έχει ως αποτέλεσμα τη δημιουργία πολλών γραμμών κώδικα για κάθε ένα από αυτά τα tasks, και οι δημιουργοί του jQuery δημιούργησαν αυτήν συγκεκριμένα τη βιβλιοθήκη για να κάνουνε πολύ εύκολα τα πιο συνηθισμένα tasks. Σε αυτό βοήθησε πάρα πολύ η δημιουργία των Selectors (των επιλογέων) του jQuery. Η τεχνολογία CSS αναπτύχθηκε με σκοπό το διαχωρισμό των στοιχείων εμφάνισης (style) και την αποθήκευσή τους σε διαφορετικό αρχείο από τα HTML στοιχεία. Για να αντιμετωπιστεί το πρόβλημα αναφοράς σε αντικείμενα που βρισκόταν σε διαφορετικό αρχείο, δημιουργήθηκε η έννοια του επιλογέα (selector). Οι επιλογείς αναπαριστούν συλλογές στοιχείων που επιλέγονται βάσει ιδιοτήτων τους ή της θέσης τους σε ένα HTML έγγραφο. Για παράδειγμα ο επιλογέας “p a” αναφέρεται σε όλα τα στοιχεία τύπου υπερσύνδεσμος <a> που περιέχονται μέσα σε στοιχεία τύπου παραγράφου <p>.

Το jQuery επεκτείνει αυτόν το μηχανισμό και υποστηρίζει ακόμα πιο ισχυρούς επιλογείς αντικειμένων. Ο αντίστοιχος επιλογέας του jQuery για το παραπάνω παράδειγμα είναι ο: $(“p a”) ή jQuery(“p a”)

Η συνάρτηση δολλαρίου, συντομογραφία της συνάρτηση jQuery(), επιστρέφει ένα ειδικό αντικείμενο αποτελούμενο από έναν πίνακα με τα DOM στοιχεία που

επιλέγονται από τον επιλογέα. Αυτό το αντικείμενο, που ονομάζεται jQuery wrapper, διαθέτει χρήσιμες, προκαθορισμένες ή μη, μεθόδους ικανές να επενεργήσουν στο σύνολο των στοιχείων. Για παράδειγμα, η εντολή: $(".somecssclass").doSomething(); έχει σαν αποτέλεσμα να εφαρμοστεί η μέθοδος doSomething() για όλα τα στοιχεία HTML, στα οποία έχει αποδοθεί η CSS κλάση «somecssclass».

Μερικοί άλλοι βασικοί επιλογείς είναι οι παρακάτω [14]:

- Επιλογή όμοιων ετικετών (tags, π.χ. $("p"): ο παραπάνω επιλογέας θα επιλέξει όλα τα στοιχεία <p> που βρίσκονται στην ιστοσελίδα. Η τιμή που επιστρέφει ο επιλογέας είναι ένας πίνακας με τα αντίστοιχα DOM αντικείμενα.
- Επιλογή στοιχείου με βάση τον κωδικό id, π.χ. $("#some_id"):
- Επιλογή στοιχείων με βάση την ετικέτα και την κλάση.
- Επιλογή στοιχείων παιδιών, π.χ. $("tr > td"):
- Επιλογή επομένου στοιχείου, π.χ. $("table + p"):

Οι παραπάνω επιλογείς μπορεί να ανασυνδυαστούν με διάφορους τρόπους, ώστε τελικά ο μηχανισμός αυτός να καταστήσει δυνατή την απόκτηση πρόσβασης σε κάθε πιθανή συλλογή αντικειμένων της σελίδας, προκειμένου στη συνέχεια να εφαρμοστεί κάποια λογική λειτουργία σε αυτά.

4.3 Η χρήση της JavaScript στο jQuery

Η προσθήκη του CSS στα toolkits ανάπτυξης εφαρμογών μας επιτρέπει να διακρίνουμε τη στυλιστική πληροφορία από τη δομή του κειμένου με αποτέλεσμα τα κείμενά μας να γίνονται ευκολότερα διαχειρίσιμα και να έχουμε τη δυνατότητα να μεταβάλλουμε τη στυλιστική εικόνα μίας σελίδας απλώς εναλλάσσοντας διαφορετικά stylesheets [14].

Παρότι η συντριπτική πλειοψηφία των web developers κάθε άλλο παρά αναπολεί τις ημέρες εφαρμογής styles με HTML elements, η παρακάτω σημειολογία είναι ακόμη πολλή συνηθισμένη:

```
<button
    type="button"
    onclick="document.getElementById('xyz').style.color='red';">
     Click Me
</button>
```

Μπορεί κανείς εύκολα να δει ότι το στυλ αυτού του button element, συμπεριλαμβανομένου του φόντου της επικεφαλίδας του, δεν εφαρμόζεται μέσω του <font> tag ή άλλης παρόμοιας σημειολογίας , αλλά καθορίζεται από τους εκάστοτε CSS κανόνες που επιδρούν πάνω στη σελίδα. Όμως παρότι αυτή η δήλωση δεν ανακατεύει το style markup με τη δομή, ανακατεύει τη συμπεριφορά με τη δομή αφού περιλαμβάνει την εκτέλεση της JavaScript όταν ο χρήστης κλικάρει το button ως κομμάτι του markup του button element μέσω του onclick attribute (το οποίο σε αυτή την περίπτωση δίνει κόκκινο χρώμα στο Document Object Model-DOM με id value xyz).

Διαχωρισμός της συμπεριφοράς από τη δομή

Για τους ίδιους ακριβώς λόγους που είναι επιθυμητό να διαχωρίζουμε το στυλ από τη δομή μέσα σε ένα HTML document, θα πρέπει να διαχωρίζουμε τη συμπεριφορά από τη δομή. Ιδανικά μία HTML σελίδα θα πρέπει να δομείται με τον τρόπο που φαίνεται στην Εικόνα 4.1 με καθένα από τα συστατικά της δομής, του style και της συμπεριφοράς να παίρνει το τμήμα που του αναλογεί [14].

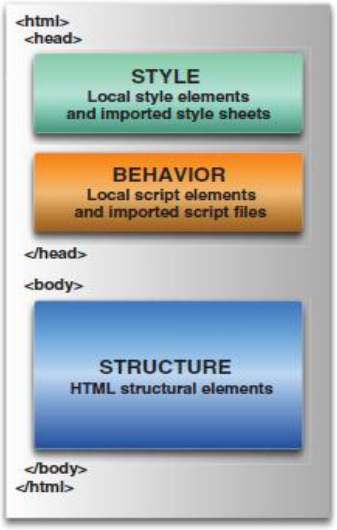
Η στρατηγική αυτή που είναι γνωστή ως Unobtrusive JavaScript, έγινε ευρύτερα γνωστή από τους δημιουργούς του jQuery και πλέον χρησιμοποιείται από όλες τις μεγάλες JavaScript βιβλιοθήκες βοηθώντας τους δημιουργούς ιστοσελίδων αυτόν τον άρτιο διαχωρισμό στις σελίδες τους. Ως η βιβλιοθήκη που έκανε γνωστή αυτή την τεχνική, ο πυρήνας του jQuery έχει βελτιωθεί τόσο ώστε να παράγει με μεγάλη ευκολία

Unobtrusive JavaScript. Η Unobtrusive JavaScript εκλαμβάνει ως σφάλμα οποιαδήποτε JavaScript έκφραση ή δήλωση που είναι ενσωματωμένη στο <body> της HTML, είτε ως attribute των HTML elements (όπως το onclick) είτε σε script blocks που βρίσκονται τοποθετημένα μέσα στο σώμα της σελίδας.

Για να υλοποιήσουμε και να χρησιμοποιήσουμε λοιπόν ένα onclick button κάνουμε την παρακάτω αλλαγή στο button element:

```
<button type="button" id="testButton">Click Me</button>
```

Η μορφή αυτή είναι σίγουρα πολλή πιο απλή και στη συνέχεια θα δούμε τι πρέπει να κάνουμε για να γίνεται η διεργασία όταν κλικάρουμε το button [14].



Εικόνα 4.1 Μεγιστοποίηση της αναγνωσιμότητας και της συντηρησιμότητας με τη σωστή τοποθέτηση της δομής του style και της συμπεριφοράς μέσα στη σελίδα.

Τμηματοποίηση του script

Αντί να ενσωματώνουμε τη συμπεριφορά του button μέσα στο markup του, τμηματοποιούμε το script μετακινώντας το σε ένα script block στο <head> section της σελίδας, έξω από την εμβέλεια του document body, όπως φαίνεται παρακάτω [14]:

```
  <script type="text/javascript">
     window.onload = function() {
     document.getElementById('testButton').onclick = function() {
     document.getElementById('xyz').style.color = 'red';
     };
  };
  </script>
```

Τοποθετούμε το script στον onload handler έτσι ώστε η σελίδα να αναθέσει μία inline συνάρτηση στο onclick attribute του button element. Προσθέτουμε αυτό το script στον onload handler επειδή θέλουμε να εξασφαλίσουμε ότι το button element υπάρχει προτού προσπαθήσουμε να το αυξήσουμε.

Η Unobtrusive JavaScript παρότι είναι μία ισχυρή τεχνική για τον καθαρό καταμερισμό των αρμοδιοτήτων μέσα σε μία web εφαρμογή, έχει ελαττώμματα. Αυξάνει για παράδειγμα το πλήθος των γραμμών του script που χρησιμοποιούμε για να πετύχουμε το στόχο μας σε σύγκριση με την απλή τοποθέτηση μέσα στο button markup. Μπορεί δηλαδή να αυξήσει το πλήθος των γραμμών που πρέπει να γραφτούν στο script και απαιτεί πειθαρχία και την εφαρμογή κατάλληλων coding patterns στο client-side script.

Παρολαυτά θα λέγαμε πως οτιδήποτε μας αναγκάζει να γράψουμε τον client-side κώδικά μας με το ίδιο επίπεδο προσοχής και σεβασμού που συνήθως κατανέμεται στον server-side κώδικα είναι ωφέλιμο, παρότι είναι αρκετή επιπλέον εργασία [14].

Προηγουμένως αναφέραμα ότι η ομάδα του jQuery έχει ειδικά επικεντρώσει τη λειτουργία του στο να το κάνει πιο εύκολο και ευχάριστο κατά τη συγγραφή κώδικα για τις σελίδες μας με τη χρήση τεχνικών Unobtrusive JavaScript, χωρίς να πρέπει να πληρώσουμε βαρύ αντάλλαγμα σε όρους προσπάθειας ή μεγέθους του κώδικα. Θα δούμε στη συνέχεια ότι η αποδοτική χρήση του jQuery μας καθιστά ικανούς να πετύχουμε πολύ περισσότερα στις σελίδες μας με την ταυτόχρονη συγραφή πολύ μικρότερου κώδικα.

4.4 Τα βασικά χαρακτηριστικά του jQuery

Η προσθήκη του CSS στα toolkits ανάπτυξης εφαρμογών μας επιτρέπει να διακρίνουμε τη στυλιστική πληροφορία από τη δομή του κειμένου με αποτέλεσμα τα κείμενά μας να γίνονται ευκολότερα διαχειρίσιμα και να έχουμε τη δυνατότητα να μεταβάλλουμε τη στυλιστική εικόνα μίας σελίδας απλώς εναλλάσσοντας διαφορετικά stylesheets [12], [13], [14].

4.4.1 Υποστήριξη Ajax

Το jQuery παρέχει ένα σύνολο αντικειμένων που υλοποιούν πλήρως τη λειτουργικότητα Ajax με εύχρηστο και πρακτικό τρόπο που αποκρύπτει εν μέρει τις λεπτομέρειες της υλοποίησης του XHR αντικειμένου από τον προγραμματιστή. Η βασική υποδομή αποτελείται από δύο αντικείμενα:

$.Ajax(): Η μέθοδος αυτή αποτελεί συνδυασμό των μεθόδων open() και send() του XHR αντικειμένου. Χρησιμοποιεί ως παράμετρο έναν πίνακα με ζεύγη ιδιότητας-τιμής, όπου οι ιδιότητες μπορεί να είναι κάποιες ή όλες από τις παρακάτω:

- url: Είναι η μόνη υποχρεωτική ιδιότητα και παριστάνει τη διεύθυνση αποστολής ενός Ajax αιτήματος.
- method: Η μέθοδος του αιτήματος, με τιμή POST ή GET.
- timeout: Ο χρόνος σε ms μέχρι να ακυρωθεί ένα αίτημα για το οποίο δεν έχει επιστραφεί απάντηση.
- data: Μία τιμή ή μια συλλογή τιμών που θα αποσταλούν μαζί με το αίτημα.
- async: Δηλώνει αν το αίτημα είναι ασύγχρονο.
- global: Λογική τιμή που δηλώνει αν η εξυπηρέτηση του αιτήματος θα εντοπιστεί από ειδικά αντικείμενα που ονομάζονται Καθολικοί Χειριστές Ajax και η λειτουργία τους περιγράφεται πιο κάτω.

- success: περιέχει μια callback συνάρτηση που εκτελείται σε περίπτωση επιτυχούς εξυπηρέτησης του αιτήματος.
- error: περιέχει μια callback συνάρτηση που εκτελείται σε περίπτωση ανεπιτυχούς εξυπηρέτησης του αιτήματος.
- complete: περιέχει μια callback συνάρτηση που εκτελείται όταν ολοκληρωθεί η εξυπηρέτηση του αιτήματος.
- beforeSend: περιέχει μια callback συνάρτηση που εκτελείται πριν την αποστολή του αιτήματος.

Ο κώδικας παρακάτω δείχνει τον τρόπο χρήσης της μεθόδου για να επιτύχει την εξυπηρέτηση ενός αιτήματος Ajax στην διεύθυνση «some_url» και να εμφανίσει ένα ενημερωτικό μήνυμα στον χρήστη, ανάλογο με την επιτυχή ή όχι απάντηση του διακομιστή.

```
$.Ajax(
      {
            url: 'some_url'
            method:'get',
            success: function(transport){
            var response = transport.responseText || "";
            alert("Επιτυχία " + response);
      },
     failure: function(){ alert('Σφάλμα') }
});
```

$.AjaxSettings(): καλείται με ίδια παράμετρο όπως η μέθοδος $.Ajax() για να αναθέσει τιμές, με καθολικό τρόπο, στις ιδιότητες που αναφέρθηκαν παραπάνω. Μετά την εκτέλεσή της, όλες οι επόμενες κλήσεις της μεθόδου $.Ajax() θα χρησιμοποιήσουν τις ίδιες τιμές των ιδιοτήτων που δηλώθηκαν.

Οι παραπάνω δύο μέθοδοι σπάνια χρησιμοποιούνται απευθείας, αλλά αποτελούν τη βάση πάνω στην οποία το jQuery παρέχει κάποιες πιο απλές και εύχρηστες μεθόδους για την εξυπηρέτηση αιτημάτων, όπως:

$.get(): αντιστοιχεί σε μια κλήση της μεθόδου $.Ajax() με προεπιλεγμένη τιμή ιδιότητας method την τιμή “GET”.
$.post(): αντιστοιχεί σε μία κλήση της μεθόδου $.Ajax() με προεπιλεγμένη τιμή ιδιότητας method την τιμή “POST”.
$.load(): εκτελείται ως μέθοδος κάποιο αντικειμένου DOM και αντιστοιχεί με μια κλήση της μεθόδου $.Ajax() και στη συνέχεια αντικατάσταση του HTML περιεχομένου του αντικειμένου με το περιεχόμενο της απάντησης στο αίτημα.

Οι Καθολικοί χειριστές Ajax αποτελούν ένα άλλο κομμάτι της υποδομής της jQuery για υποστήριξη Ajax. Σε γενικές γραμμές αντιστοιχούν στα αντικείμενα Ajax.Responders του Prototype και η λειτουργία τους είναι να αποκρίνονται σε αλλαγές της κατάστασης εξυπηρέτησης για όλα τα αιτήματα της σελίδας, εκτελώντας κάποια συνάρτηση.

4.4.2 Document Ready Handler

Η ισχυρή τάση του jQuery προς της κατεύθυνση της διακριτικής παρουσίας της JavaScript σημαίνει ότι πρέπει να βρεθεί κάποιος τρόπος ώστε οι συναρτήσεις και οι χειριστές γεγονότων που αφορούν τα HTML στοιχεία να μπορούν να δηλωθούν έξω από την περιοχή της σελίδας που περιέχει τα ίδια τα στοιχεία.

Ο τρόπος που επιλέχθηκε είναι η δήλωση όλων των απαραίτητων χειριστών γεγονότων σε μια κεντρική συνάρτηση, η οποία θα εκτελείται μετά τη φόρτωση της σελίδας, δηλαδή αφού έχουν ήδη δημιουργηθεί τα αντικείμενα τα οποία αφορούν οι χειριστές γεγονότων. Η μέθοδος onload ( ) του DOM αντικειμένου window είναι δυνατό να χρησιμοποιηθεί γι’ αυτό το σκοπό. Το γεγονός όμως ότι, σύμφωνα με τις προδιαγραφές, δεν εκτελείται από τον περιηγητή, παρά μόνο όταν έχει ολοκληρωθεί η διαδικασία φόρτωσης όλων των στοιχείων της σελίδας, σημαίνει ότι είναι δυνατόν να γίνει εμφανής η καθυστέρηση της εκτέλεσής της στον τελικό χρήστη, αν π.χ. η σελίδα περιέχει πολλές εικόνες που αργούν να εμφανιστούν.

Η εναλλακτική λύση της jQuery είναι να χρησιμοποιήσει την συνάρτηση-χειριστή φόρτωσης σελίδας (Document Ready Handler):

```
$(document).ready(function() {
//δήλωση χειριστών γεγονότων
…
});
```

Η συνάρτηση αυτή εκτελείται σε ένα χρονικό σημείο κατά το οποίο έχει ολοκληρωθεί η κατασκευή της ιεραρχίας αντικειμένων DOM από τον περιηγητή, ώστε τα αντικείμενα αυτά να είναι διαθέσιμα για την δήλωση των χειριστών γεγονότων, χωρίς όμως απαραίτητα να έχουν φορτωθεί πλήρως τα στοιχεία που απαρτίζουν την σελίδα, π.χ. εικόνες κ.α. Με αυτό τον τρόπο αποφεύγεται η ανεπιθύμητη συμπεριφορά που περιγράφηκε παραπάνω.

4.4.3 Δημιουργία DOM elements

Μπορούμε να δημιουργήσουμε DOM elements περνώντας στην συνάρτηση $( ) μία συμβολοσειρά που περιέχει το HTML markup για αυτά τα elements. Για παράδειγμα, μπορούμε να δημιουργήσουμε ένα καινούριο paragraph element ως εξής:

```
$("<p>Hi there!</p>")
```

Ωστόσο η δημιουργία ενός ‘αποκομμένου’ DOM element (ή ιεραρχία από elements) δεν είναι και τόσο χρήσιμη, αφού συνήθως η ιεραρχία που δημιουργείται από μία τέτοια κλήση εφαρμόζεται στη συνέχεια κατά τη χρήση κάποιας από τις DOM manipulation συναρτήσεις του jQuery. Το παρακάτω παράδειγμα είναι χαρακτηριστικό.

```
<html>
  <head>
    <title>Follow me!</title>
    <link rel="stylesheet" type="text/css"
          href="../styles/core.css"/>
    <script type="text/javascript" src="../scripts/jquery-1.4.js">
    </script>
    <script type="text/javascript">
      $(function(){
        $("<p>Hi there!</p>").insertAfter("#followMe");
      });
    </script>
  </head>

  <body>
    <p id="followMe">Follow me!</p>
  </body>
</html>
```

1 Ready handler that creates HTML element

2 Existing element to be followed

Το παραπάνω παράδειγμα εγκαθιστά ένα υπάρχον HTML paragraph element που ονομάζεται followMe στο document body. Στο script element μέσα στο τμήμα <head> εγκαθιστούμε έναν ready handler που χρησιμοποιεί την δήλωση που ακολουθεί για να εισάγει μία καινούρια παράγραφο μέσα στο DOM tree ακριβώς μετά από το υπάρχον element: `$("<p>Hi there!</p>").insertAfter("#followMe");`

Το αποτέλεσμα φαίνεται στην Εικόνα 4.2

Εικόνα 4.2 Η δυναμική δημιουργία και εισαγωγή elements που συνήθως απαιτεί πολλές γραμμές κώδικα, μπορεί να επιτευχθεί μέσα σε μία μόνο γραμμή στον κώδικα του jQuery.

4.4.4 Πρόσθετα Utilities

Εκτός από το μηχανισμό ανάκτησης συλλογών αντικειμένων DOM μέσω των επιλογέων, το jQuery υποστηρίζει επίσης μια σειρά μεθόδων για τη διαχείριση αυτών των αντικειμένων. Οι μέθοδοι αυτοί περιλαμβάνουν, μεταξύ άλλων:

- Διαχείριση των ιδιοτήτων DOM αντικειμένων: παρέχονται οι μέθοδοι attr(<όνομα ιδιότητας>) και attr(attribute) για να αναθέσουν και να ανακτήσουν αντίστοιχα την τιμή μιας ιδιότητας κάποιου αντικειμένου.
- Διαχείριση των ιδιοτήτων CSS αντικειμένων: παρέχονται οι μέθοδοι css(<όνομα ιδιότητας>,<νέα τιμή>) και css(<όνομα ιδιότητας>) για να αναθέσουν και να ανακτήσουν αντίστοιχα την τιμή μιας ιδιότητας CSS κάποιου αντικειμένου.
- Τροποποίηση του περιεχομένου και της τιμής των αντικειμένων της σελίδας:

1. Οι μέθοδοι html( ) και html(<νέο HTML περιεχόμενο>) ανακτούν και αντικαθιστούν με νέα τιμή, αντίστοιχα, τα HTML περιεχόμενα κάποιου στοιχείου που είναι κατάλληλο να περιέχει HTML περιεχόμενο, όπως π.χ. div, p, κ.α.
2. Οι μέθοδοι text( ) και text(<νέο κείμενο>) ανακτούν και αντικαθιστούν με νέα τιμή, αντίστοιχα, τα περιεχόμενα κειμένου κάποιου στοιχείου που είναι κατάλληλο να περιέχει τέτοιο περιεχόμενο.
3. Οι μέθοδοι val( ) και val(<νέα τιμή>) ανακτούν και αντικαθιστούν με νέα τιμή, αντίστοιχα, την τιμή (ιδιότητα “value”) σε κάποιο στοιχείο, όπως πεδία κειμένου, λίστες επιλογής, κ.ά.

- Αντιγραφή, τοποθέτηση, και διαγραφή αντικειμένων στη σελίδα:

1. Οι μέθοδοι prepend( ) και append( ) αναλαμβάνουν να τοποθετήσουν έναντικείμενο DOM ως πατέρα ή ως παιδί, αντίστοιχα, κάποιου άλλου αντικειμένου, ενώ οι μέθοδοι insertBefore( ) και insertAfter( ) αναλαμβάνουν να το τοποθετήσουν πριν η μετά από κάποιο άλλο αντίστοιχα.
2. Η μέθοδος clone( ) αντιγράφει ένα αντικείμενο DOM, με ή χωρίς όλη την ιεραρχία των αντικειμένων-παιδιών του.
3. Η μέθοδος remove( ) απομακρύνει ένα αντικείμενο από τη σελίδα.
4. Η μέθοδος empty( ) διαγράφει τα τυχόν περιεχόμενα του.

- Διάτρεξη Συλλογής (Iteration): η μέθοδος each(someFunction) παίρνει ως όρισμα μια συνάρτηση και η εκτέλεσή της πάνω σε μια συλλογή αντικειμένων έχει ως αποτέλεσμα την εκτέλεση της συγκεκριμένης συνάρτησης-ορίσματος διαδοχικά για κάθε ένα από τα στοιχεία της συλλογής.

- Χειριστές γεγονότων (Event Handlers): το jQuery περιλαμβάνει μια πλήρη συλλογή μεθόδων που αντιστοιχούν σε όλα τα πιθανά γεγονότα διεπαφής χρήστη που υποστηρίζει ένας περιηγητής. Το γεγονός στο οποίο αντιστοιχεί κάθε μέθοδος φαίνεται εύκολα από το όνομα της:

1. Γεγονότα σχετικά με την φόρτωση του εγγράφου: μέθοδοι load( ) και unload( ).
2. Γεγονότα σχετικά με τη χρήση του ποντικιού: παρέχονται οι μέθοδοι mouseover( ), mouseout( ), mousemove( ), hover (), toggle ( ), mouseup( ), mousedown( ), click( ), και dblclick( ).
3. Γεγονότα σχετικά με τη χρήση του πληκτρολόγιου: παρέχονται οι μέθοδοι keydown( ), keypress( ), και keyup( ).
4. Γεγονότα σχετικά με φόρμες HTML: παρέχονται οι μέθοδοι focus( ), blur( ), και submit( ).

Η πλειοψηφία των παραπάνω μεθόδων χρησιμοποιείται με δύο πιθανούς τρόπους:

- Η κλήση της μεθόδου χωρίς ορίσματα:

  $('#form').submit ( )

  έχει ως αποτέλεσμα την άμεση εκτέλεση του ίδιου του γεγονότος – την υποβολή μιας φόρμας στην προκειμένη περίπτωση.

- Η κλήση με όρισμα μία συνάρτηση:

  $('#form').submit (someFunction( ))

  έχει ως αποτέλεσμα την δήλωση της συνάρτησης αυτής ως χειριστή γεγονότων για το συγκεκριμένο γεγονός και το συγκεκριμένο στοιχείο που επιστρέφει ο επιλογέας. Εν προκειμένω, η μέθοδος «someFunction( )» θα εκτελεστεί όταν ο χρήστης υποβάλλει τα στοιχεία της φόρμας.

- Μέθοδοι για οπτικά αποτελέσματα: το jQuery περιλαμβάνει, μεταξύ άλλων, μεθόδους για σταδιακή εμφάνιση – fadeIn( ) – και απόκρυψη – fadeout( ) – αντικειμένων.

4.4.5 Οι επεκτάσεις του jQuery

Το jQuery παρέχει έναν εύκολο μηχανισμό επέκτασης και ενσωμάτωσης νέων μεθόδων που εμπλουτίζουν τη λειτουργικότητα του. Ο όρος που χρησιμοποιείται συχνά για την επέκταση είναι jQuery Plug in. Υπάρχουν εκατοντάδες διαθέσιμα Plugins, τα οποία μπορούν να συνεργαστούν ανά οποιοδήποτε πιθανό συνδυασμό για να επιτύχουν την επιθυμητή λειτουργικότητα. Ο μηχανισμός αυτός, στον οποίο η βιβλιοθήκη οφείλει μεγάλο μέρος της δημοτικότητάς της, περιγράφεται με ένα παράδειγμα.

Ας υποθέσουμε ότι θέλουμε να προσθέσουμε μια νέα μέθοδο σε όλα τα στοιχεία ενός jQuery wrapper αντικειμένου που επιστρέφεται από έναν επιλογέα. Η εντολή που το επιτυγχάνει είναι του τύπου:

```
jQuery.fn.methodName = methodDefinition;
```

όπου methodName είναι το όνομα της νέας μεθόδου και methodDefinition η υλοποίηση της. Τυπικά, ένα jQuery wrapper αποτελείται από μια συλλογή αντικειμένων, επομένως η υλοποίηση της συνάρτησης θα πρέπει να λάβει αυτό το γεγονός υπόψη και να επενεργήσει σε όλα τα αντικείμενα της συλλογής. Ο παρακάτω κώδικας:

```
jQuery.fn.showAlert = function() {
return this.each(function() {
alert("Αυτή είναι μια νέα μέθοδος ");
});
}
```

ορίζει μια νέα μέθοδο “showAlert”, η οποία, όταν εκτελεστεί, εμφανίζει ένα μήνυμα στο χρήστη. Η μεταβλητή “this” είναι αναφορά στο ίδιο το jQuery wrapper αντικείμενο, και η μέθοδος each() διατρέχει την συλλογή αντικειμένων του, εκτελώντας για κάθε ένα από αυτά τον κώδικα της συνάρτησης-όρισμα –στην προκειμένη περίπτωση την εμφάνιση του μηνύματος.

Η νέα μέθοδος που ορίστηκε έτσι μπορεί να χρησιμοποιηθεί στη συνέχεια ως μέθοδος από έναν επιλογέα: $('.someClass').showAlert();
με αποτέλεσμα το μήνυμα ”Αυτή είναι μια νέα μέθοδος” να εμφανιστεί διαδοχικά, μία φορά για κάθε στοιχείο HTML με CSS κλάση “someClass” στο έγγραφο.

Το γεγονός ότι στην JavaScript μια συνάρτηση αποτελεί ταυτόχρονα και αντικείμενο, και επομένως μπορεί να έχει δικές της ιδιότητες και μεθόδους, σημαίνει ότι ο παραπάνω απλός μηχανισμός μπορεί να οδηγήσει σε Plugins με εξαιρετικά σύνθετη λειτουργικότητα.

4.4.6 To extension Form Plugin

Το Form Plugin αποτελεί μια από τις πιο χρήσιμες και διαδεδομένες επεκτάσεις του jQuery και προσφέρει ενισχυμένη λειτουργικότητα για τη διαχείριση HTML φορμών, καθώς και τη σύγχρονη ή ασύγχρονη υποβολή τους, εμπλουτίζοντας ταυτόχρονα την Ajax υποστήριξη του jQuery Framework. Η επέκταση παρέχει τις παρακάτω μεθόδους για ασύγχρονη υποβολή στοιχείων φόρμας:

- AjaxForm(): μέθοδος της φόρμας που προετοιμάζει ένα αίτημα υποβολής των στοιχείων της μέσω Ajax. Οι παράμετροι της κλήσης είναι ανάλογες με την περίπτωση κλήσης της $.Ajax().
- AjaxSubmit(): η μέθοδος της φόρμας που εκτελεί την υποβολή ενός αιτήματος υποβολής των στοιχείων της μέσω Ajax. Αν έχει προηγηθεί κλήση της μεθόδου AjaxForm() για αυτή τη φόρμα, δεν χρειάζεται να οριστούν παράμετροι. Αλλιώς οι παράμετροι της κλήσης είναι ανάλογες με την περίπτωση κλήσης της AjaxForm().
- AjaxFormUnbind(): η εκτέλεσή της ακυρώνει τα αποτελέσματα της μεθόδου AjaxForm(), αν αυτή έχει εκτελεστεί προηγουμένως.

Επίσης, παρέχονται κάποιες χρήσιμες μέθοδοι διαχείρισης της φόρμας και των στοιχείων της:

- .formToArray(): εκτελείται σε ένα αντικείμενο φόρμας και συγκεντρώνει όλες τις τιμές των στοιχείων μιας φόρμας σε ένα αντικείμενο τύπου πίνακα.
- .formSerialize(): εκτελείται σε ένα αντικείμενο φόρμας και συγκεντρώνει όλες τις τιμές των στοιχείων μιας φόρμας σε ένα ειδικό αλφαριθμητικό αντικείμενο, κατάλληλο να αποσταλεί στον διακομιστή ως μέρος ενός αιτήματος.

- .fieldSerialize(): εκτελείται σε μια συλλογή αντικειμένων πεδίου φόρμας και συγκεντρώνει όλες τις τιμές τους σε ένα ειδικό αλφαριθμητικό αντικείμενο κατάλληλο να αποσταλεί στον διακομιστή ως μέρος ενός αιτήματος.
- .fieldValue(): εκτελείται σε μια συλλογή αντικειμένων πεδίου φόρμας και συγκεντρώνει όλες τις τιμές τους σε ένα αντικείμενο τύπου πίνακα.
- .clearForm(): επαναφέρει τα στοιχεία της φόρμας στην αρχική τους κατάσταση.

4.5 Σχεδιασμός και υλοποίηση ενός ηλεκτρονικού καταστήματος φωτοβολταϊκών με τη χρήση του jQuery

Στην εργασία μας χρησιμοποιήσαμε τις ιδιότητες και τα πλεονεκτήματα του jQuery για να υλοποιήσουμε ένα ηλεκτρονικό κατάστημα (e-shop) φωτοβολταϊκών. Προσπαθήσαμε να ενσωματώσουμε όσο το δυνατό περισσότερα utilities του jQuery και να παρουσιάσουμε το σύνολο των προϊόντων του ηλεκτρονικού μας καταστήματος σε ένα απλό και λειτουργικό interface. Ο χρήστης μπορεί να βλέπει τα φωτοβολταϊκά προϊόντα, να μαθαίνει τα λειτουργικά χαρακτηριστικά και την τιμή τους καθώς και την κατηγορία προϊόντος στην οποία αυτά ανήκουν (πχ ανεμογεννήτριες).

Με τη χρήση ενός καλαθιού για αγορές ο χρήστης μπορεί, με ένα απλό κλικάρισμα επάνω στο προϊόν, να το εισάγει στη λίστα αγορών του. Παρέχεται επίσης η δυνατότητα διαγραφής ενός ή περισσότερων προϊόντων από τη λίστα σε περίπτωση που το προϊόν αυτό επιλέχθηκε κατά λάθος ή που ο χρήστης απλώς δεν επιθυμεί πλέον την αγορά του, καθώς και η δυνατότητα αύξησης των τεμαχίων που ο χρήστης θέλει να αγοράσει από ένα συγκεκριμένο είδος, πολύ εύκολα, κλικάροντας την εικόνα του προϊόντος τόσες φορές όσα είναι τα τεμάχια που θέλει να αγοράσει ο χρήστης.

Παρακάτω θα δούμε αναλυτικά και με εικόνες όλες τις λειτουργίες του ηλεκτρονικού μας καταστήματος, πρώτα όμως θα αναφέρουμε ορισμένα στοιχεία για το

Microsoft Visual Web Developer 2010 Express, το εργαλείο που μας βοήθησε να «στήσουμε» την εφαρμογή μας.

4.5.1 Microsoft Visual Web Developer 2010 Express

Το Microsoft Visual Studio Express είναι ένα σύνολο από freeware ολοκληρωμένα περιβάλλοντα ανάπτυξης (IDE's – Integrated Development Environments), που δημιουργήθηκαν από τη Microsoft και αποτελούν 'μικρότερες' εκδοχές της γραμμής παραγωγής του Microsoft Visual Studio. Οι Express εκδόσεις ξεκίνησαν την ύπαρξή τους με το Visual Studio 2005. Η ιδέα πίσω από τις Express εκδόσεις είναι σύμφωνα με τη Microsoft η παροχή σύγχρονων, εύκολων στη χρήση και στην κατανόηση IDE's σε χρήστους που δεν είναι επαγγελματίες web developers, αλλά απλώς ερασιτέχνες προγραμματιστές ή μαθητές.

Οι πρώτες versions του Visual Studio 2005 Express εμφανίστηκαν τον Οκτώβριο του 2005 και δύο χρόνια αργότερα, το Νοέμβριο του 2007 βγήκε το Visual Studio 2008 Express μαζί με το Service Pack 1 που το συνόδευε. Οι εκδόσεις Visual Studio Express του 2008 και του 2010 (την οποία και χρησιμοποιήσαμε) απαιτούν το Windows XP SP3 ή νεότερη έκδοση των Windows.

Για την υλοποίηση της εργασίας μας κατεβάσαμε και εγκαταστήσαμε από τον ιστότοπο της Microsoft http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-web-developer-express το Visual Studio 2010 Express, το οποίο δουλεύει για 30 ημέρες δοκιμάστικα και στη συνέχεια θα πρέπει να κάνει κανείς registration για να το αποκτήσει.

Το Visual Studio Express αποτελείται από τα παρακάτω ξεχωριστά προϊόντα:

- Visual Basic Express
- Visual Web Developer Express
- Visual C++ Express
- Visual C# Express
- SQL Server Express
- Express for Windows Phone

Το Visual Web Developer Express αποτελεί ένα δωρεάν διαθέσιμο εργαλείο για web development το οποίο επιτρέπει στους developers να αξιολογήσουν τις δυνατότητες ανάπτυξης, διαμόρφωσης και επεξεργασίας web εφαρμογών του Visual Studio 2008 χωρίς κανένα κόστος. Η βασική του λειτουργία είναι η δημιουργία ASP.NET ιστοτόπων. Περιέχει τα παρακάτω στοιχεία:

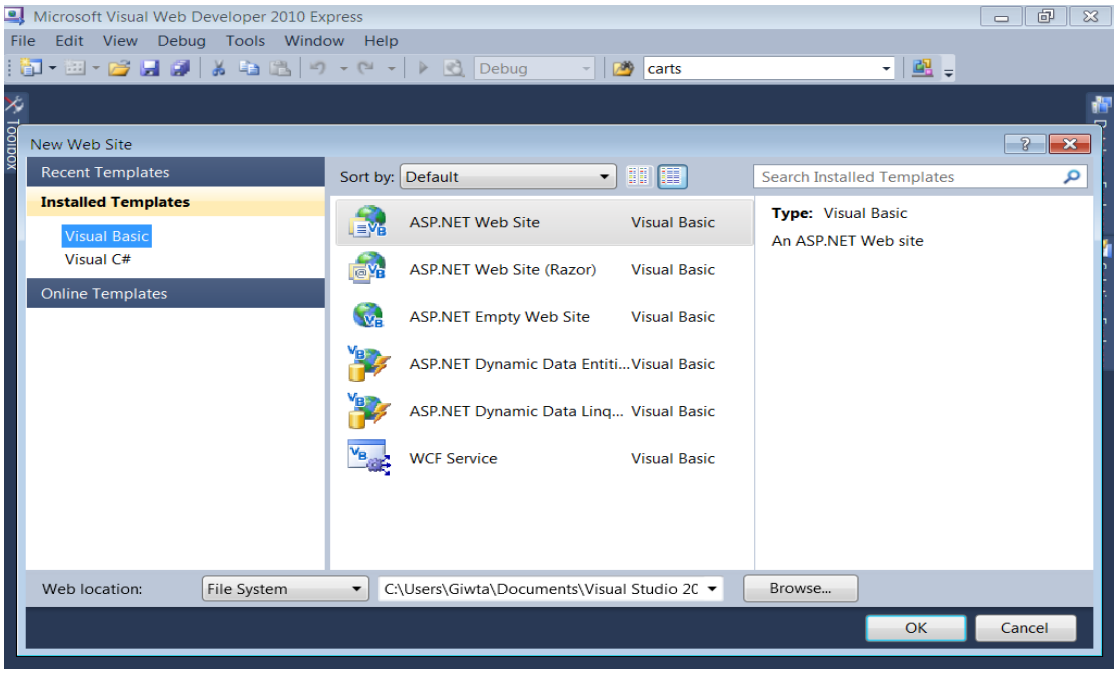
- Μία WYSIWYG διεπαφή,
- Έναν drag-and-drop user interface designer,
- Ενισχυμένους HTML και code editors,
- Έναν περιορισμένο database explorer,
- Λειτουργία υποστήριξης για άλλες web technologies (δηλαδή για CSS, JavaScript, XML),
- Ολοκληρωμένη, design-time validation για standards όπως το XHTML 1.0/1.1 και το CSS 2.1.

Εικόνα 4.3 Η αρχική σελίδα για το κατέβασμα του Microsoft Visual Studio Web Developer 2010 Express, του εργαλείου που χρησιμοποιήσαμε στη δημιουργία του e-shop.

Εικόνα 4.4 Ορισμένα από τα στοιχεία που ενσωματώνει το Microsoft Visual Studio Web Developer 2010 Express, και χρησιμοποιήθηκαν στην υλοποίησή μας.

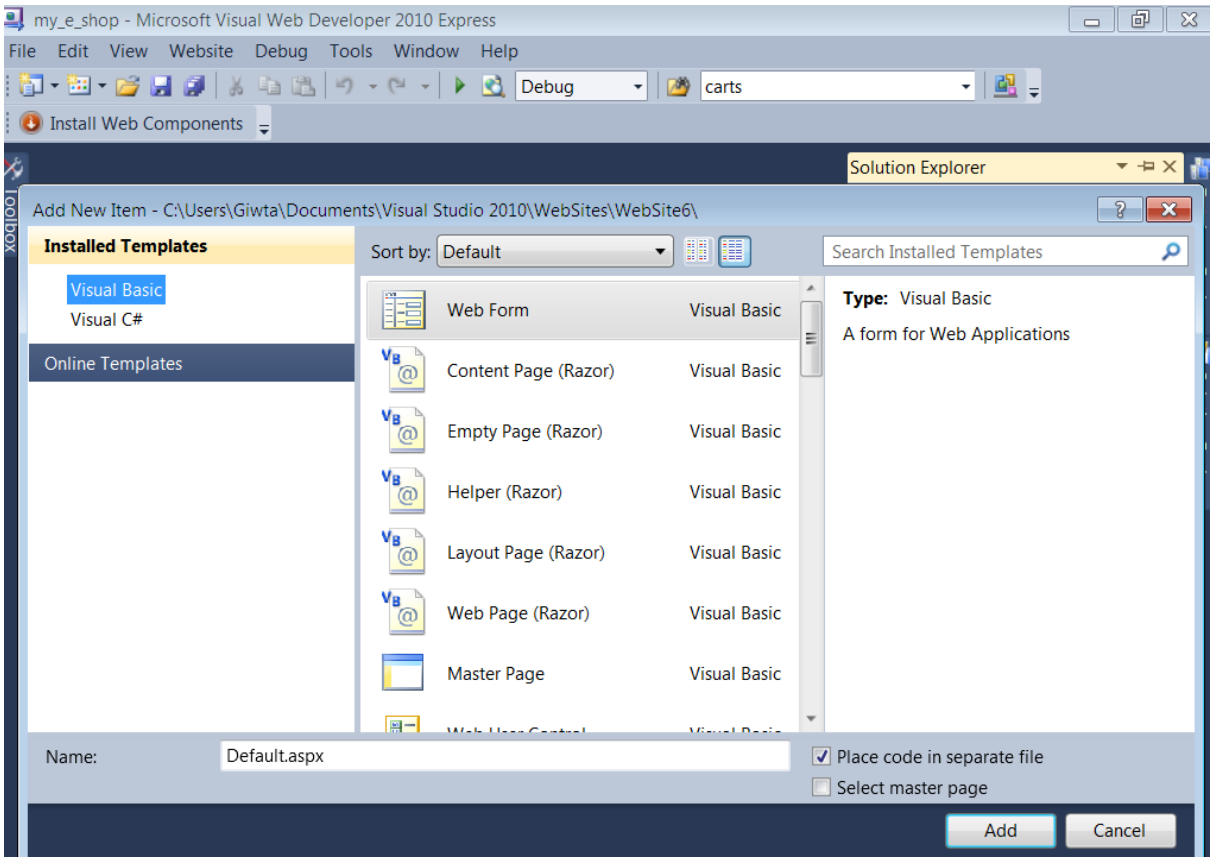
4.5.2 Η δομή των αρχείων του ηλεκτρονικού καταστήματος φωτοβολταϊκών

Για να εγκαταστήσουμε τα αρχεία της εφαρμογής μας ανοίγουμε το Visual Web Developer 2010 Express και πατάμε File → New Web Site → (Εικόνα 4.4).



Εικόνα 4.4 Δημιουργία νέου web site

Στη συνέχεια επιλέγουμε τη δημιουργία καινούριου ASP.NET Web Site και το ονομάζουμε my_e_shop. Για να δημιουργήσουμε ένα καινούριο αρχείο ακολουθούμε τη διαδρομή: Solution Explorer WebSite6 (δεξί κλικ) Add New Item. Σε αυτό το σημείο εμφανίζεται μία λίστα με όλους τους τύπους αρχείων που μπορούμε να δημιουργήσουμε στο IDE μας, για παράδειγμα HTML page, Jscript File, Style Sheet, XML file, SQL Server Database, Silverlight Application, Text File, Web Form, Master Page και πάρα πολλά άλλα (Εικόνα 4.5).



Εικόνα 4.5 Επιλογή του τύπου του αρχείου που επιθυμούμε να προσθέσουμε στο site μας

Το αρχείο index.html είναι εκείνο το οποίο ουσιαστικά υλοποιεί τον jQuery κώδικά μας και στο οποίο μπορεί κανείς να δει αναλυτικά πάρα πολλές από τις functionalities του jQuery που εξετάσαμε στις προηγούμενες ενότητες του κεφαλαίου (για παράδειγμα τους selectors, την χαρακτηριστική συνάρτηση $( ), τον wrapper κ.ά.).

Μεγάλο ενδιαφέρον έχει φυσικά και το αρχείο που υλοποιεί το style sheet του e-shop, το css.css. Όλα τα αρχεία και οι εικόνες βρίσκονται κάτω από το ίδιο directory, που δεν είναι άλλο από το WebSite μας. Στη συνέχεια παραθέτουμε μόνο το κομμάτι του κώδικα του jQuery, που φυσικά περιλαμβάνεται στο αρχείο index.html, ενώ όλα τα αρχεία και οι εικόνες που συνθέτουν το ηλεκτρονικό μας κατάστημα βρίσκονται στο

directory my_e_shop. Για να εκτελεστούν χρειάζεται να τοποθετηθούν κάτω από το ίδιο directory, και όχι μέσα σε ξεχωριστούς φακέλους για να γίνεται κατανοητό το path από τον κώδικα.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Φωτοβολταϊκά Συστήματα</title>
<script type="text/javascript" src="jquery-1.3.2.js"></script>
<script type="text/javascript" src="jquery.livequery.js"></script>
<link href="css.css" rel="stylesheet" />
```

Το παραπάνω αποτελεί το πρώτο από τα τρία κομμάτια στα οποία χωρίζουμε σε αυτό το σημείο τον κώδικα του index.html για να τον κατανοήσουμε καλύτερα. Εδώ δηλώνουμε τον τίτλο της σελίδας μας, και συμπεριλαμβάνουμε και το Style Sheet αρχείο (το css αρχείο) σύμφωνα με το οποίο θα διαμορφωθεί η σελίδα μας. Επίσης βλέπουμε και τη δήλωση του αρχείου της JavaScript του jQuery 1.3.2. αφού όπως είχαμε αναφέρει και στην ενότητα 4.1 η εγκατάσταση του jQuery είναι εξαιρετικά απλή αφού αρκεί να το τοποθετήσουμε στο φάκελο της web εφαρμογής μας και να χρησιμοποιήσουμε το HTML <script> tag για να το συμπεριλάβουμε στις σελίδες μας.

Στη συνέχεια παραθέτουμε τον jQuery κώδικα του αρχείου μας που υλοποιεί τις συναρτήσεις που δρουν πάνω στα δεδομένα μας (δηλαδή στη λίστα με τα φωτοβολταϊκά προϊόντα). Θα πρέπει να αναφέρουμε ότι μπορεί κανείς εφόσον επιθυμεί να βάλει τον κώδικα με τις jQuery συναρτήσεις σε ένα JavaScript αρχείο (έστω στο filename.js) ξεχωριστά εφόσον επιθυμεί, και να το δηλώσει sto html αρχείο όπως πιο πάνω ως <script type=”text/javascript” src=”filename.js”></script>.

```
<script type="text/javascript">
$(document).ready(function() {

	var Arrays=new Array();

	$('#wrap li').mousemove(function(){
		var position = $(this).position();

		$('#cart').stop().animate({


				left	: position.left+'px',

			},250,function(){

		});
	}).mouseout(function(){

	});

	$('#wrap li').click(function(){

		var thisID = $(this).attr('id');

		var itemname  = $(this).find('div .name').html();
		var itemprice = $(this).find('div .price').html();

		if(include(Arrays,thisID))
		{
			var price	= $('#each-'+thisID).children(".shopp-
price").find('em').html();
			var quantity = $('#each-'+thisID).children(".shopp-
quantity").html();
			quantity = parseInt(quantity)+parseInt(1);
			var total = parseInt(itemprice)*parseInt(quantity);

			$('#each-'+thisID).children(".shopp-
price").find('em').html(total);
			$('#each-'+thisID).children(".shopp-quantity").html(quantity);

			var prev_charges = $('.cart-total span').html();
			prev_charges = parseInt(prev_charges)-parseInt(price);

			prev_charges = parseInt(prev_charges)+parseInt(total);
			$('.cart-total span').html(prev_charges);
			$('#total-hidden-charges').val(prev_charges);
		}
		else
		{
			Arrays.push(thisID);

			var prev_charges = $('.cart-total span').html();
```

```
                    prev_charges = parseInt(prev_charges)+parseInt(itemprice);

                    $('.cart-total span').html(prev_charges);
                    $('#total-hidden-charges').val(prev_charges);

                    $('#left_bar .cart-info').append('<div class="shopp" id="each-
'+thisID+'"><div class="label">'+itemname+'</div><div class="shopp-price">
€<em>'+itemprice+'</em></div><span class="shopp-quantity">1</span><img
src="remove.png" class="remove" /><br class="all" /></div>');

                    $('#cart').css({'-webkit-transform' : 'rotate(20deg)','-moz-
transform' : 'rotate(20deg)' });
                }
                setTimeout('angle()',200);
        });


        $('.remove').livequery('click', function() {

                var deduct = $(this).parent().children(".shopp-
price").find('em').html();
                var prev_charges = $('.cart-total span').html();

                var thisID = $(this).parent().attr('id').replace('each-','');

                var pos = getpos(Arrays,thisID);
                Arrays.splice(pos,1,"0")

                prev_charges = parseInt(prev_charges)-parseInt(deduct);
                $('.cart-total span').html(prev_charges);
                $('#total-hidden-charges').val(prev_charges);
                $(this).parent().remove();

        });

        $('#Submit').livequery('click', function() {

                var totalCharge = $('#total-hidden-charges').val();

                $('#left_bar').html('Η Παραγγελία σας έχει καταχωρηθεί στο
σύστημα..Σας ευχαριστούμε!Το συνολικό ποσό είναι: €'+totalCharge);
                return false;

        });

});

function include(arr, obj) {
  for(var i=0; i<arr.length; i++) {
    if (arr[i] == obj) return true;
  }
}
```

```
function getpos(arr, obj) {
  for(var i=0; i<arr.length; i++) {
    if (arr[i] == obj) return i;
  }
}
function angle(){$('#cart').css({'-webkit-transform' : 'rotate(0deg)','-moz-
transform' : 'rotate(0deg)' });}

</script>
```

Το τελευταίο κομμάτι του index.html αρχείου μας παρατίθεται στη συνέχεια και είναι εκείνο στο οποίο εισάγεται η λίστα με τα προϊόντα του ηλεκτρονικού καταστήματος μαζί μετις ιδιότητές τους (τιμή, εικόνα κλπ.).

Κάθε φορά που ο χρήστης κλικάρει το όνομα του προϊόντος καλείται ένα αρχείο το οποίο περιλαμβάνει την περιγραφή του. Έχουμε επιλέξει να καλείται κάθε φορά ξεχωριστό αρχείο (τύπου htm) για την περιγραφή καθενός από τα προϊόντα, θα μπορούσε όμως να ήταν αρχείο διαφόρων άλλων τύπων (πχ pdf).

```
<body>

    <div class="container">
           <div id="titlos" align="center">
        <h1>Φωτοβολταϊκά Συστήματα...Μια έξυπνη επένδυση</h1>
        <img src="resolve_it_green.jpg" class="items" height="120" width="120"
alt="" />

        <h3>Νέα απελευθερωμένη αγορά ηλεκτρικής ενέργειας!</h3>
        </div>
    </div>

            <span class="top-label">
           <span class="label-txt">
            <div>Επιλέξτε από την μεγάλη γκάμα προϊόντων μας εκείνα που
ταιριάζουν στις ενεργειακές ανάγκες του κτιρίου σας..</div></span>
        </span>
       <div align="left">

       <div id="wrap" align="left">

             <ul>
                   <li id="1">
```

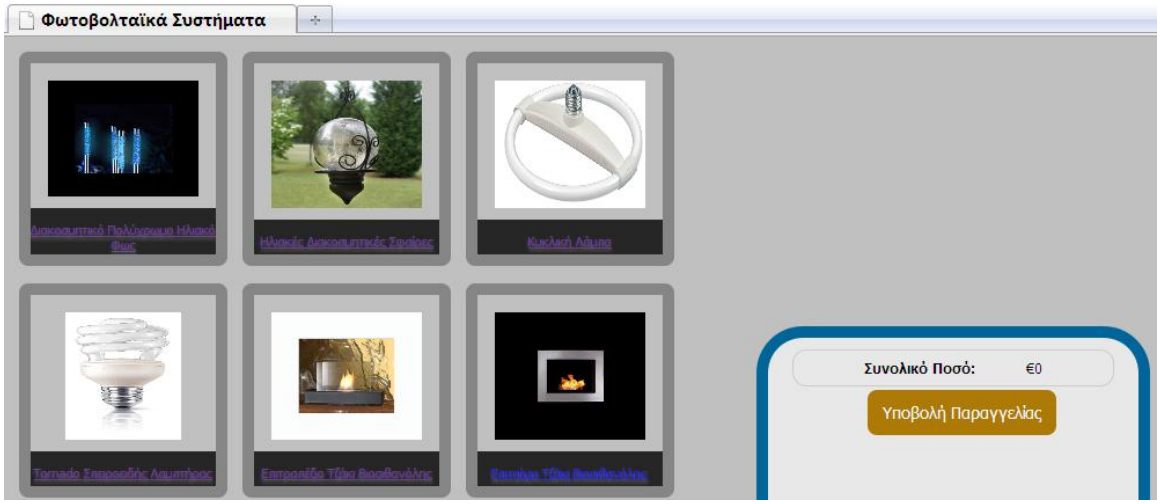
```
                    <img src="1.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a href="description_1.htm">Ηλιακός Προβολέας 50 LED</a></span></div>
                <div><span class="name">Ηλιακός Προβολέας 50 LED</span>: € <span class="price">100</span></div>

            </li>
                    <li id="2">
                    <img src="2.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a href="description_2.htm">Ηλιακός Προβολέας 28 LED</a></span></div>
                <div><span class="name">Ηλιακός Προβολέας 28 LED</span>: € <span class="price">50 </span></div>
                    </li>
                    <li id="3">
                    <img src="3.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a href="description_3.htm">Διακοσμητικό Ηλιακό Φως Κήπου</a></span></div>
                    <div><span class="name">Διακοσμητικό Ηλιακό Φως Κήπου</span>: € <span class="price">15</span></div>
                    </li>
                    <li id="4">
                    <img src="4.png" class="items" height="100" width="130" alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a href="description_4.htm">Διακοσμητικό Πολύχρωμο Ηλιακό Φως</a></span></div>
                        <div><span class="name">Διακοσμητικό Πολύχρωμο Ηλιακό Φως</span>: € <span class="price">20 </span></div>
                    </li>
                    <li id="5">
            <img src="5.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a href="description_5.htm">Ηλιακές Διακοσμητικές Σφαίρες</a></span></div>
                        <div> <span class="name">Ηλιακές Διακοσμητικές Σφαίρες</span>: € <span class="price">18</span></div>
                    </li>
                    <li id="6">
            <img src="6.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a href="description_6.htm">Κυκλική Λάμπα</a></span></div>
<div><span class="name">Κυκλική Λάμπα</span>: € <span class="price">25</span>
</div>
                    </li>
```

```
                    <li id="7">
               <img src="7.png" class="items" height="110" width="100" alt="" />
                    <br clear="all" />
                 <div><span class="perigrafi"><a href="description_7.htm">Tornado
Σπειροειδής Λαμπτήρας</a></span></div>
                              <div><span class="name">Tornado Σπειροειδής
Λαμπτήρας</span>: € <span class="price">21</span></div>
                    </li>
                    <li id="8">
               <img src="8.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                 <div><span class="perigrafi"><a
href="description_8.htm">Επιτραπέζιο Τζάκι Βιοαιθανόλης</a></span></div>
                              <div><span class="name">Επιτραπέζιο Τζάκι
Βιοαιθανόλης</span>: € <span class="price">80 </span></div>
                    </li>
                    <li id="9">
               <img src="9.png" class="items" height="110" width="130" alt="" />
                    <br clear="all" />
                 <div><span class="perigrafi"><a href="description_9.htm">Επιτοίχιο
Τζάκι Βιοαιθανόλης</a></span></div>
                              <div><span class="name">Επιτοίχιο Τζάκι
Βιοαιθανόλης</span>: € <span class="price">150</span></div>
                    </li>
                    <li id="10">
                    <img src="10.png" class="items" height="110" width="130"
alt="" />
                    <br clear="all" />
                 <div><span class="perigrafi"><a
href="description_10.htm">Επιδαπέδιο Τζάκι Βιοαιθανόλης</a></span></div>
                              <div><span class="name">Επιδαπέδιο Τζάκι
Βιοαιθανόλης</span>: € <span class="price">160</span></div>
                    </li>
                    <li id="11">
                    <img src="11.png" class="items" height="110" width="130"
alt="" />
                    <br clear="all" />
                 <div><span class="perigrafi"><a
href="description_11.htm">Ανεμογεννήτρια 600 Watt</a></span></div>
                              <div><span class="name">Ανεμογεννήτρια 600 Watt</span>:
€ <span class="price">500</span></div>
                    </li>
                    <li id="12">
                    <img src="12.png" class="items" height="110" width="130"
alt="" />
          <br clear="all" />
                 <div><span class="perigrafi"><a
href="description_12.htm">Ανεμογεννήτρια 450 watt</a></span></div>
                    <div><span class="name">Ανεμογεννήτρια 450 watt</span>: €
<span class="price">350</span></div>
                    </li>
            <li id="13">
```

```
                    <img src="13.png" class="items" height="110" width="130"
alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a
href="description_13.htm">Φωτοβολταϊκό πάνελ 50 Wp</a></span></div>
                    <div><span class="name">Φωτοβολταϊκό πάνελ 50 Wp</span>: €
<span class="price">200</span></div>
                    </li>
                    <li id="14">
                    <img src="14.png" class="items" height="110" width="130"
alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a
href="description_14.htm">Φωτοβολταϊκό πάνελ 230 Wp</a></span></div>
                        <div><span class="name">Φωτοβολταϊκό πάνελ 230
Wp</span>: € <span class="price">500</span></div>
                    </li>
                    <li id="15">
                    <img src="15.png" class="items" height="110" width="130"
alt="" />
                    <br clear="all" />
                <div><span class="perigrafi"><a
href="description_15.htm">Φωτοβολταϊκό πάνελ 10 Wp</a></span></div>
            <div><span class="name">Φωτοβολταϊκό πάνελ 10 Wp</span>: € <span
class="price">50</span></div>
                    </li>
            </ul>
            <br clear="all" />
    </div>

       <div id="left_bar" align="center">

            <form action="#" id="cart_form" name="cart_form">

            <div class="cart-info"></div>

            <div class="cart-total">

            <b>Συνολικό
Ποσό:          </b>
€<span>0</span>
            <input type="hidden" name="total-hidden-charges" id="total-hidden-
charges" value="0" />
</div>
            <button type="submit" id="Submit">Υποβολή Παραγγελίας</button>

            </form>
       </div>
</div>
</body>
</html>
```

4.5.3 Η διεπαφή του ηλεκτρονικού καταστήματος φωτοβολταϊκών

Στη συνέχεια θα δούμε αναλυτικά και με εικόνες το έτοιμο ηλεκτρονικό κατάστημα φωτοβολταϊκών.



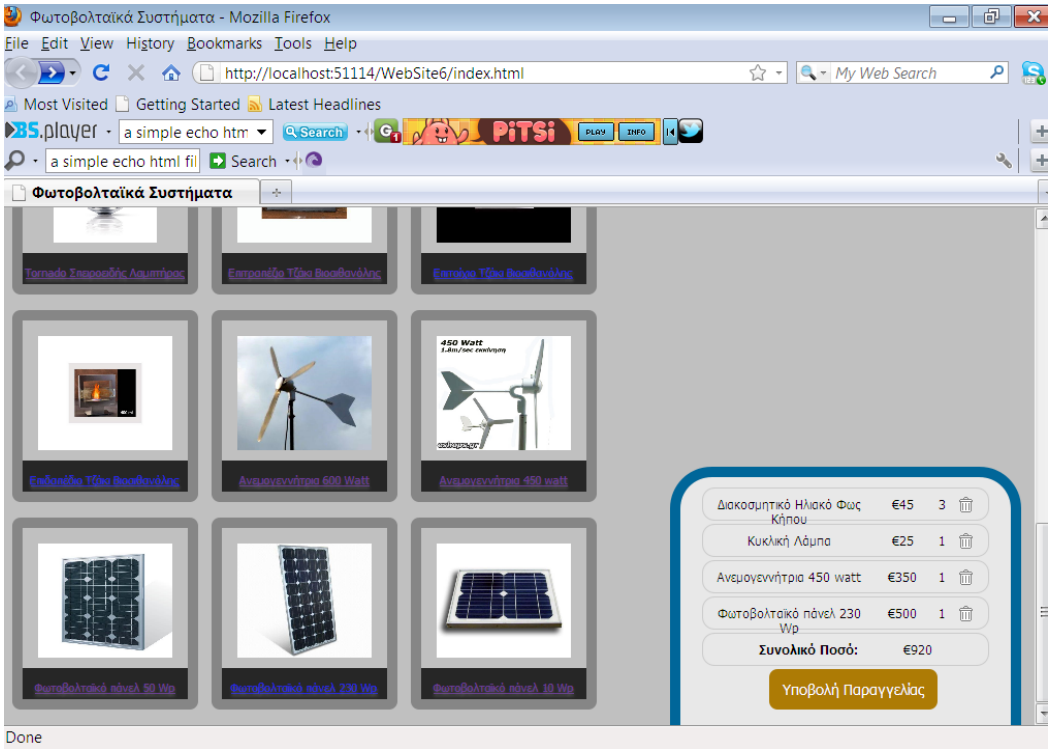
Εικόνα 4.6 Η αρχική σελίδα του e-shop. Παρουσίαση των προϊόντων με εικόνες.

Εφόσον πατήσουμε αριστερό κλικ επάνω στο όνομα ενός προϊόντος εμφανίζονται όλες οι πληροφορίες σχετικά με αυτό, δηλαδή η τιμή του, το όνομά του, η εικόνα του, η κατηγορία φωτοβολταϊκών στην οποία ανήκει και η περιγραφή του για να μπορεί ο χρήστης να πάρει περισσότερες πληροφορίες. Η Εικόνα 4.7 δείχνει το αποτέλεσμα για κλικάρισμα πάνω στο προϊόν «Ανεμογεννήτρια 600 Watt».

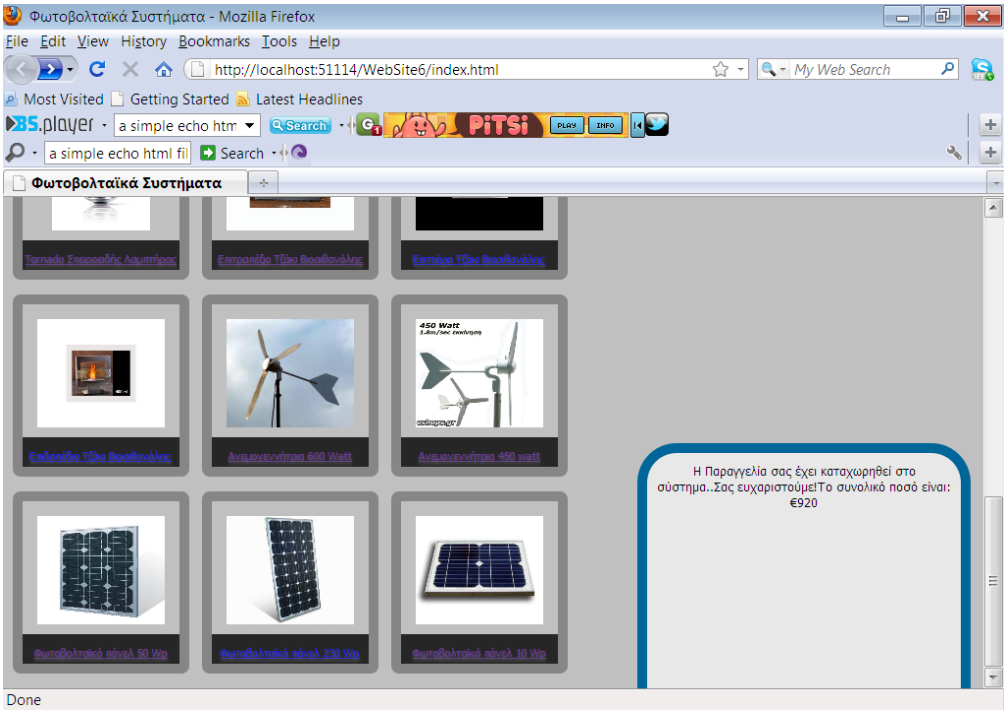
Εικόνα 4.6 Η αρχική σελίδα του e-shop. Παρουσίαση των προϊόντων με εικόνες.

Εάν τώρα ο πελάτης επιθυμεί να αγοράσει κάποιο προϊόν μπορεί πολύ απλά να πατήσει πάνω στην Εικόνα του προϊόντος και τότε αυτό αυτόματα μπαίνει στη φόρμα που είναι δεξιά στη σελίδα. Εμφανίζονται στη φόρμα αυτή πληροφορίες σχετικά με το προϊόν που επέλεξε ο χρήστης, το όνομά του, η τιμή του και το πλήθος των τεμαχίων ανά προϊόν. Με τον ίδιο τρόπο μπορεί ο χρήστης να επιλέξει όσα διαφορετικά προϊόντα επιθυμεί και στο κάτω μέρος της φόρμας εμφανίζεται το συνολικό ποσό που καλείται να πληρώσει ο πελάτης.

Παρέχεται επίσης η δυνατότητα διαγραφής ενός ή περισσότερων προϊόντων από τη λίστα παραγγελίας, εφόσον δεν ο πελάτης δεν επιθυμεί τελικά την αγορά του. Εάν κάποιος θέλει να αγοράσει περισσότερα από ένα τεμάχια από το ίδιο προϊόν μπορεί πολύ απλά να κλικάρει επάνω στην εικόνα του προϊόντος τόσες φορές όσα τεμάχια επιθυμεί να παραγγείλει. Το άθροισμα στο συνολικό ποσό γίνεται αυτόματα. Η Εικόνα 4.7 δείχνει μία παραγγελία έτοιμη προς υποβολή.



Εικόνα 4.7 Μία παραγγελία προτού υποβληθεί από τον πελάτη. Ο πελάτης σε αυτό το σημείο μπορεί εφόσον το επιθυμεί να διαγράψει κάποιο προϊόν από τη λίστα παραγγελίας ή να προχωρήσει στην υποβολή της παραγγελίας.



Εικόνα 4.8 Μήνυμα επιτυχημένης υποβολής παραγγελίας

Για να υποβάλλει τέλος ο πελάτης την παραγγελία του αρκεί να πατήσει το button «Υποβολή Παραγγελίας». Τότε εμφανίζεται μήνυμα από το κατάστημα που τον ενημερώνει για την επιτυχή υποβολή της παραγγελίας του και για το συνολικό ποσό (Εικόνα 4.8).

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Jesse Garrett, "Ajax: A New Approach to Web Applications", www.adaptivepath.com/publications/essays/archives/000385.php , February 2005

[2] Nicholas C. Zakas, Jeremy McPeak, Joe Fawcett, "Professional Ajax", 2007, 2nd edition, Wiley Publishing

[3] http://techcityinc.com/2008/12/01/best-15-ajax-based-start-pages/

[4] Robert Cooper, Charles Collins, "GWT in Practice", Manning Publications Co, 2008

[5] Ed Burnette, "The Pragmatic Bookshelf", Raleigh, North Carolina Dallas, The Pragmatic Programmers LLC., Texas 2006, http://www.pragmaticprogrammer.com

[6] Robert Hanson, Adam Tacy, "Manning GWT in Action,Easy Ajax with the Google Web Toolkit", Copyright 2007, Manning Publications, www.manning.com

[7] Rawld Gill, Craig Riecke, Alex Russell, "Mastering Dojo JavaScript and Ajax Tools for Great Web Experiences", The Pragmatic Bookshelf Raleigh, North Carolina Dallas, Texas, 2008

[8] Μωυσιάδης Βασίλειος, "Διαδικτυακή Εφαρμογή Διεξαγωγής Εξετάσεων", Πανεπιστήμιο Μακεδονίας, Κοζάνη, Μάρτιος 2008

[9] Chris Ullman, Lucinda Dykes, "Wrox – Beginning Ajax", by Wiley Publishing, Inc., Indianapolis, Indiana, 2007

[10] Joe Lennon, Product Manager,Core International, "Build an Ajax application with the Dojo Toolkit", Skill Level: Intermediate, 01 Mar 2011

[11] Bear Bibeault and Yehuda Katz, "jQuery in Action", Manning Publications Co, 2008

[12] Jonathan Chaffer& Karl Swedberg, "jQuery Reference Guide", Packt Publishing Ltd. 2007

[13] Jonathan Chaffer& Karl Swedberg, "Learning jQuery", Packt Publishing Ltd. 2009

[14] Bear Bibeault and Yehuda Katz, “jQuery in Action”, Manning Publications Co, 2nd Edition

[15] http://www.microsoft.com/visualstudio/en-us/products/2010-editions/visual-web-developer-express

[16] http://en.wikipedia.org/wiki/Ajax_(programming)

[17] http://en.wikipedia.org/wiki/Ajax_framework

[18] https://alexbosworth.backpackit.com/pub/67688,