ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Δηλωτική σύνθεση διεπαφών για γεωγραφικά δεδομένα για το πλαίσιο JSF 2.0

Διπλωματική Εργασία

Δημήτριος Ι. Βολιώτης

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Επ. Καθ. Σαμολαδάς Βασίλειος (επιβλέπων)
Επ. Καθ. Δεληγιαννάκης Αντώνιος
Καθ. Χριστοδουλάκης Σταύρος

Εργασία που κατατέθηκε στο Πολυτεχνείο Κρήτης για την εκπλήρωση των απαιτήσεων απόκτησης του Διπλώματος στο τμήμα Ηλεκτρονικών Μηχανικών & Μηχανικών Υπολογιστών

Χανιά, Οκτώβριος 2012

Στην οικογένεια μου και στους φίλους μου.

ΠΕΡΙΛΗΨΗ

Στα πλαίσια της παρούσας διπλωματικής κατασκευάστηκε μία σουίτα 15 προσαρμοσμένων συστατικών (custom component suite) για το πλαίσιο JavaServer Faces (JSF) με το όνομα JVFACES. Τα συστατικά αυτά υπακούν πλήρως τις προδιαγραφές του Java Enterprise Edition (J2EE) και πιο ειδικά στο JSF 2.0 specification. Εκμεταλλεύονται την τεχνολογία AJAX (Asynchronous Javascript and XML) και αφορούν αποκλειστικά την διαχείριση γεωγραφικών δεδομένων. Για την κατασκευή τους χρησιμοποιήθηκε σε μεγάλο βαθμό η javascript βιβλιοθήκη OpenLayers καθώς επίσης και η σουίτα Java Topology Suite (JTS).

Η σουίτα JVFACES είναι σχεδιασμένη για το πλαίσιο JSF 2.0 ώστε να λειτουργεί χωρίς καμία εξάρτηση από άλλες σουίτες του και έχει ελάχιστες απαιτήσεις, εκτός από το ίδιο το πλαίσιο. Αποτελείται από ένα και μοναδικό αρχείο jar, το οποίο εγκαθίσταται εύκολα και δεν χρειάζεται παραμετροποίηση. Η χρήση της ορίζεται από αναλυτικό εγχειρίδιο και διαφαίνεται μέσα από παραδείγματα. Απευθύνεται κυρίως σε web developers, εξοικειωμένους με το περιβάλλον JSF και με ενδιαφέρον για το ανάπτυξη WebGIS εφαρμογών.

Στόχος της σουίτας JVFACES είναι να αποτελέσει ένα ισχυρό και πάνω από όλα εύχρηστο εργαλείο ανάπτυξης WebGIS εφαρμογών, αποδεσμεύοντας τους developers από την εκτενή/άμεση χρήση javascript και κάνοντας ευκολότερη την δημιουργία, συντήρηση και παραμετροποίηση του πηγαίου κώδικα των εφαρμογών τους.

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα ήθελα να ευχαριστήσω ορισμένα άτομα που συνέβαλαν αποφασιστικά στην ανάπτυξη και συγγραφή της παρούσας εργασίας.

Καταρχήν θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Βασίλειο Σαμολαδά για την ανάθεση της διπλωματικής εργασίας, καθώς επίσης για την συνεχή επίβλεψη και την καθοδήγησή του σε όλα τα στάδια εκπόνησης αυτής της διπλωματικής εργασίας. Να ευχαριστήσω προκαταβολικά τον κ. Αντώνιο Δεληγιαννάκη και τον κ. Σταύρο Χριστοδουλάκη για την ανάγνωση και αξιολόγηση της εργασίας αυτής.

Ιδιαίτερες ευχαριστίες στον φίλο και συνάδελφο Μιχάλη Αργυρίου για την πολύτιμη βοήθειά του επί τεχνικών θεμάτων και τέλος ένα μεγάλο ευχαριστώ στην οικογένεια μου και στους φίλους μου που με στήριξαν σε όλη τη διάρκεια των σπουδών και που συνεχίζουν ακόμα.

Δημήτριος Ι. Βολιώτης
Πολυτεχνείο Κρήτης
Χανιά, 2012

Περιεχόμενα

Ευρετήριο πινάκων

Κατάλογος εικόνων

1 Εισαγωγή

Στο Κεφάλαιο αυτό θα αναφερθούμε σε εισαγωγικές έννοιες, όπως η περιοχή ενδιαφέροντός μας, το "πρόβλημα" που καλούμαστε να αντιμετωπίσουμε και την λύση που προτείνουμε μέσω της εργασίας μας. Τέλος, θα περιγράψουμε την δομή των επόμενων κεφαλαίων.

1.1 Η περιοχή ενδιαφέροντος

1.1.1 Γενικά

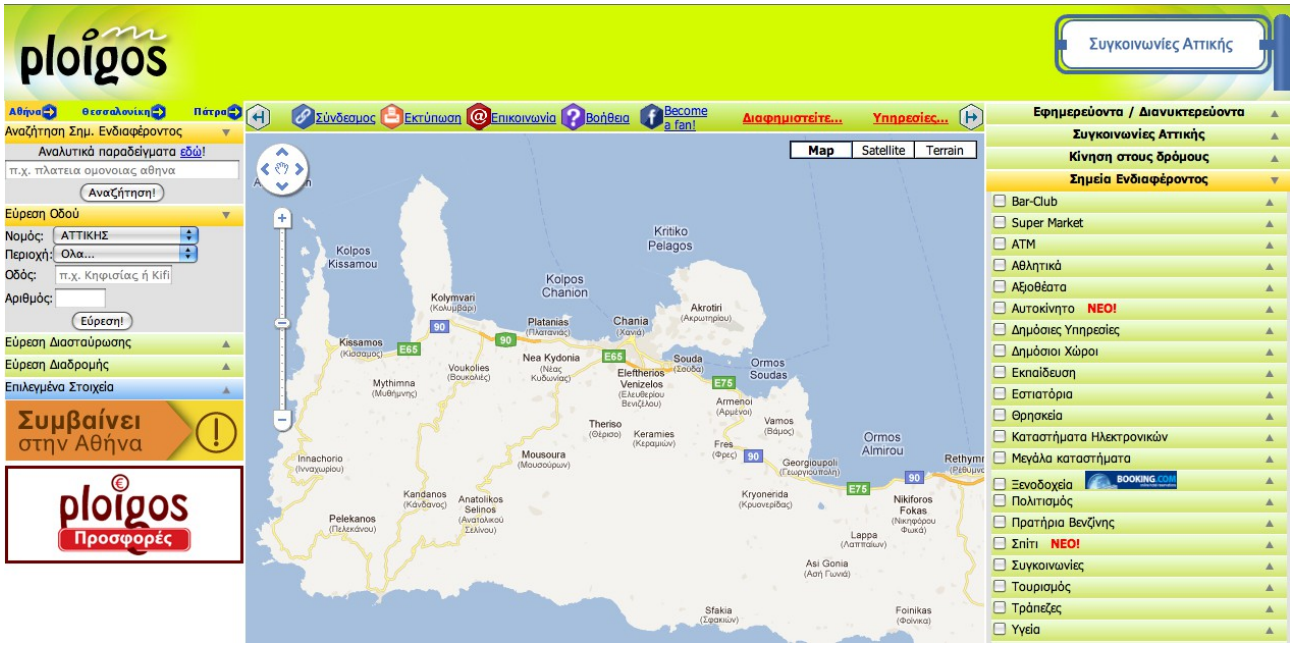
Στη σημερινή εποχή το διαδίκτυο έχει γίνει βασικό αγαθό-υπηρεσία στον ανεπτυγμένο αλλά και στον αναπτυσσόμενο κόσμο. Η διείσδυση του διαδικτύου παγκοσμίως τον Ιούνιο του 2010 ήταν στο 28% του πληθυσμού, καταγράφοντας αύξηση κατά 44% τα τελευταία 10 χρόνια, σύμφωνα με το Internet World Stats. Η αλλαγή που έχει επιφέρει σε πάρα πολλούς τομείς της ζωής μας είναι τεράστια.

Η ευρεία πλέον διαθεσιμότητα των υπηρεσιών του διαδικτύου έχει μεταβάλλει τον τρόπο που χρησιμοποιούμε τους υπολογιστές. Το μεγαλύτερο ποσοστό των χρηστών σήμερα χρησιμοποιεί τον υπολογιστή σαν ένα τερματικό του διαδικτύου. Για πολλούς ανθρώπους ένας υπολογιστής χωρίς πρόσβαση στο διαδίκτυο είναι εν μέρει άχρηστος. Αυτό οφείλεται στην μεγάλη πλέον διαθεσιμότητα εφαρμογών που δεν τρέχουν στον υπολογιστή μας, αλλά σε κάποιον απομακρυσμένο υπολογιστή του δικτύου. Η πρόσβαση σε αυτές τις εφαρμογές γίνεται μέσω του παγκόσμιου ιστού (world wide web), μέσω του φυλλομετρητή (browser) του υπολογιστή μας. Τέτοιες εφαρμογές είναι και οι εξής πολύ δημοφιλείς εφαρμογές (χωρισμένες σε κατηγορίες):

Κατηγορία	**Εφαρμογές**
Σουίτες γραφείου	Google Docs[1]
Υπηρεσίες ηλ. ταχυδρομείου	Gmail[2], Hotmail[3], Yahoo Mail[4]
Υπηρεσίες αναπαραγωγής βίντεο	YouTube[5], GreekTube[6]
Υπηρεσίες κοινωνικής δικτύωσης	Facebook[7], Twitter[8]
Υπηρεσίες instant messaging	Google Talk[9]
Υπηρεσίες χαρτογράφησης μέσω Web	Google Maps[10], Yahoo Maps[11], Bing Maps[12], ploigos[13]

Πίνακας 1: Δημοφιλείς Web Εφαρμογές

Η κατηγορία στην οποία θα εστιάσουμε στη συγκεκριμένη εργασία είναι οι υπηρεσίες χαρτογράφησης μέσω διαδικτύου (web mapping[14]).



Εικόνα 1: Υπηρεσία χαρτογράφησης μέσω web με χρήση Google Maps: www.ploigos.gr

1.1.2 Web Mapping / WebGIS

Η άνθιση λοιπόν του διαδικτύου και των τεχνολογιών πληροφορικής οδήγησε στην προσφορά διαδικτυακών υπηρεσιών που πριν ελάχιστα χρόνια ήταν διαθέσιμες μόνο σε κυβερνήσεις και ερευνητικούς οργανισμούς. Μια τέτοια υπηρεσία είναι και η υπηρεσία χαρτογράφησης μέσω διαδικτύου (web mapping).

Με τον όρο Web Mapping αναφερόμαστε στη διαδικασία του σχεδιασμού, υλοποίησης, δημιουργίας και διανομής χαρτών μέσω του παγκόσμιου ιστού (World Wide Web) και των παραγώγων του. Ενώ το web mapping ασχολείται κυρίως με τεχνολογικά θέματα, η χαρτογραφία μέσω web μελετά επίσης θεωρητικές πτυχές: την χρήση των web χαρτών, την αξιολόγηση και τη βελτιστοποίηση των τεχνικών και των ροών εργασίας, η χρηστικότητα των χαρτών web, τις κοινωνικές πτυχές, και πολλά άλλα.

Το Web GIS είναι παρόμοια έννοια με το Web mapping αλλά με έμφαση στην ανάλυση, επεξεργασία συγκεκριμένων γεωγραφικών δεδομένων και σε διερευνητικές πτυχές. Συχνά οι όροι web GIS και web mapping θεωρούνται συνώνυμοι παρόλο που δεν έχουν ακριβώς την ίδια σημασία.

Η χρήση του διαδικτύου ως μέσο διάθεσης των χαρτών μπορεί να θεωρηθεί σαν μία τεράστια εξέλιξη στην χαρτογραφία, στην οποία δημιουργεί πολλές νέες δυνατότητες όπως:

- *Οι χάρτες πραγματικού χρόνου:* Αν ο χάρτης δημιουργείται αυτόματα από βάσεις δεδομένων, τότε μπορούν να εμφανίζονται πληροφορίες σχεδόν σε πραγματικό χρόνο. Μόλις τα δεδομένα γίνουν διαθέσιμα στη βάση, αυτόματα εμφανίζονται και στο χάρτη που τραβάει πληροφορίες από αυτή.

- *Η φθηνότερη διανομή των χαρτών μέσω του διαδικτύου:* Παλαιότερα έπρεπε να αγοράσει κανείς τον χάρτη τυπωμένο, ή να αγοράσει και να εγκαταστήσει το απαραίτητο λογισμικό. Σήμερα αρκεί ένα απλό τερματικό που να παρέχει υπηρεσίες διαδικτύου.

- *Συχνή ενημέρωση των δεδομένων, η οποία γίνεται φθηνότερη και ευκολότερη:* Εφόσον τα δεδομένα αντλούνται κάθε φορά από κάποια κεντρική βάση, αρκεί απλά να ενημερωθεί αυτή. Δεν χρειάζεται να γίνει επέμβαση στο τερματικό του χρήστη.

- *Δυνατότητα προσωποποιημένου περιεχομένου στον χάρτη:* Κάθε χρήστης μπορεί να επιλέγει τις πληροφορίες που θα εμφανίζονται στον χάρτη, όπως και να προσθέσει δικές του προσωπικές πληροφορίες.

- *Πολλαπλές πηγές δεδομένων:* Μπορούν να συνδυαστούν δεδομένα από διάφορες πηγές, ενώ πολλές φορές δίνεται και η δυνατότητα στο κοινό να προσθέτει δεδομένα.

- *Ευκολότερος διαμοιρασμός γεωγραφικών πληροφοριών:* Κάθε χρήστης μπορεί να διανείμει προσωπικές ή μη γεωγραφικές πληροφορίες εύκολα λόγω της ευρείας διαθεσιμότητας των χαρτών σε όλους.

1.2 *Το πρόβλημα*

Εκτός βέβαια από τις πολλές νέες δυνατότητες, η ευρεία χρήση των υπηρεσιών χαρτογράφησης μέσω διαδικτύου αντιμετωπίζει και διάφορα τεχνικά ζητήματα:

- ➔ *Ζητήματα αξιοπιστίας:* Η αξιοπιστία του διαδικτύου και των web servers δεν είναι ακόμα αρκετά καλή. Ειδικά αν ο χάρτης βασίζεται σε εξωτερικές, κατανεμημένες πηγές δεδομένων, πολλές φορές δεν είναι εγγυημένη η διαθεσιμότητα των πληροφοριών.

- ➔ *Ζήτημα ταχύτητας του δικτύου:* Οι χάρτες μέσω διαδικτύου χρειάζονται αρκετά υψηλές

ταχύτητες μεταφοράς δεδομένων.

➔ *Απαιτούν μεγάλη πολυπλοκότητα για να αναπτυχθούν και να συντηρηθούν.*

➔ *Υψηλό κόστος γεωγραφικών δεδομένων:* Αν εξαιρεθούν οι ΗΠΑ όπου τα δεδομένα είναι διαθέσιμα σχετικά φθηνά από την κυβέρνηση, σε άλλες περιοχές όπως η Ευρώπη είναι ακριβά, καθώς διατίθενται από ιδιωτικές εταιρίες. Επίσης σε άλλες αναπτυσσόμενες χώρες δεν έχουν συλλεχθεί τέτοια δεδομένα.

Τα Συστήματα GIS[15] και WEB GIS /WEB MAPPING έχουν επιπλέον ιδιαιτερότητες καθώς απαιτούν εξειδικευμένες γνώσεις σε πολλά και διαφορετικά αντικείμενα όπως χαρτογραφία, κλίμακες απεικόνισης, προβολικά συστήματα, μετατροπή συντεταγμένων, γεω-αναφορά, δόμηση χωρικών βάσεων δεδομένων, τεχνολογίες 3D απεικόνισης και επιπλέον απαιτούν εξειδικευμένες γνώσεις του συγκεκριμένου αντικειμένου / τομέα στον οποίο προορίζονται να εφαρμοστούν.

*1.3 **Η λύση που προτείνουμε***

Το μερίδιο των GIS μέσα στην αγορά των έργων Πληροφορικής είναι σημαντικό αλλά λόγω των ιδιαίτερων απαιτήσεων τεχνογνωσίας που προϋποθέτουν απαιτείται απόλυτα επαγγελματική διαχείριση και μεγάλη εμπειρία για να αποφέρουν θετικό οικονομικό αποτέλεσμα στο σύνολο. Για το λόγω αυτό, η συγκεκριμένη εργασία έχει ως **στόχο** τη σύνθεση κατάλληλων "εργαλείων", τα οποία απευθύνονται στους developers Web GIS συστημάτων, και θα τους παρέχουν έναν δηλωτικό περιβάλλον σύνθεσης διεπαφών για γεωγραφικά δεδομένα και κατ' επέκταση, την αντιμετώπιση των "ζητημάτων" που προκύπτουν από τα Web GIS συστήματα, όπως αυτά αναφέρθηκαν στην παραπάνω ενότητα.

Ουσιαστικά αναφερόμαστε σε μία σουίτα προσαρμοσμένων συστατικών (custom components), που θα λειτουργούν αποκλειστικά πάνω στο Java-based πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών JavaServer Faces[16][46][47][48][49][50][51][52]. Η σουίτα αυτή θα εστιάζει στην μείωση της πολυπλοκότητας σύνθεσης των εν λόγω συστημάτων, στην ευκολία διαχείρισης, συντήρησης και επέκτασής τους, καθώς επίσης και στην μείωση του χρόνου και του κόστους που απαιτεί η υλοποίηση τους. Για την επίτευξη αυτών των στόχων, η σύνθεση της σουίτας θα βασιστεί σε ορισμένα εξαιρετικά open source εργαλεία και βιβλιοθήκες, όπως:

✔ *η javascript βιβλιοθήκη OpenLayers[17]*, που ειδικεύεται στην σύνθεση γεωγραφικών εφαρμογών

✔ *η Java βιβλιοθήκη Java Topology Suite[18]*, για την σύνθεση γεωγραφικών δεδομένων

✔ *το πλαίσιο JavaServer Faces*, που με δηλωτικό τρόπο, απλοποιεί τη σύνθεση προσαρμοσμένων συστατικών/ διεπαφών σε μορφή XML

- ✔ *η τεχνολογία AJAX (Asynchronous javascript and XML)[19]*, για γρήγορη client-side δημιουργία διαδραστικών εφαρμογών

- ✔ *τα standards του Open Geospatial Consortium (OGC)[20]*

1.4 Η δομή των επόμενων κεφαλαίων

Το υπόλοιπο της παρούσας διπλωματικής εργασίας δομείται ως εξής:

- ➔ Στο Κεφάλαιο 2 περιγράφεται η σχετική έρευνα και τεχνολογίες για τα συστήματα WebGIS, που έγινε για της ανάγκης της εργασίας μας, καθώς και τα εργαλεία και βιβλιοθήκες που συντέλεσαν στην ανάπτυξη της σουίτας JVFACES.

- ➔ Στο Κεφάλαιο 3 αναλύεται η σουίτα JVFACES ως προς τα συστατικά της. Συγκεκριμένα περιγράφεται η δομή τους, η χρήση τους και οι απαιτήσεις τους.

- ➔ Τέλος, στο Κεφάλαιο 4 συνοψίζεται η συνεισφορά της εργασίας μας και αναφέρονται μελλοντικές επεκτάσεις.

2 Σχετική Έρευνα & Τεχνολογίες

Στο κεφάλαιο αυτό θα γίνει αναλυτική περιγραφή στις τεχνολογίες που χρησιμοποιήθηκαν καθώς επίσης και στα εργαλεία που συντέλεσαν στην δημιουργία της σουίτας JVFACES. Συγκεκριμένα, θα αναφερθούμε στο (α) πλαίσιο JavaServer Faces (JSF), στην (β) Javascript βιβλιοθήκη OpenLayers, στην (γ) κοινοπραξία Open Geospatial Consortium(OGC) και τα πρωτόκολλα WFS[21] και WMS[22], και τέλος, στις δημοφιλείς (δ) Web Mapping τεχνολογίες των Google και Yahoo.

2.1 Το πλαίσιο JavaServer Faces (JSF)

2.1.1 Γιατί επιλέξαμε το JSF;

Σήμερα μπορεί κανείς να επιλέξει μεταξύ πολλών πλαισίων (frameworks) για την ανάπτυξη διεπαφών χρήστη μιας εφαρμογής web. Το JavaServer Faces (JSF) είναι ένα πλαίσιο που βασίζεται σε συστατικά (components). Για παράδειγμα, αν θέλει να εμφανίσει έναν πίνακα με γραμμές και στήλες, δεν χρειάζεται να παράγει HTML ετικέτες για τις γραμμές και τα κελιά σε βρόχο, αλλά να προσθέσει ένα συστατικό-πίνακα σε μια σελίδα. Χρησιμοποιώντας συστατικά, μπορεί να σκεφτεί τις διεπαφές χρήστη (user interfaces) ως ένα υψηλότερο επίπεδο προγραμματισμού από ότι τον σκέτο HTML κώδικα. Μπορεί να χρησιμοποιήσει ξανά τα δικά του συστατικά αλλά και τρίτων (third-party). Επίσης έχει τη δυνατότητα να χρησιμοποιήσει ένα οπτικό (visual) περιβάλλον ανάπτυξης, στο οποίο μπορεί να μεταφέρει και να αποθέσεί (drag and drop) συστατικά σε μια φόρμα.

Το πλαίσιο JSF τα εξής τμήματα:

- Ένα σύνολο από προκατασκευασμένες διεπαφές χρήστης (User Interface Components).
- Ένα μοντέλο προγραμματισμού οδηγούμενο από συμβάντα (event-driven programming model).
- Ένα μοντέλο συστατικών που επιτρέπει σε τρίτους προγραμματιστές να παρέχουν πρόσθετα συστατικά.

Ορισμένα συστατικά JSF είναι απλά, όπως στους πεδία εισαγωγής (input fields) και κουμπιά (buttons). Άλλα είναι αρκετά εξελιγμένα, όπως για παράδειγμα, οι πίνακες δεδομένων (data tables) και τα δέντρα (trees).

Το JSF περιέχει όλο τον απαραίτητο κώδικα για το χειρισμό συμβάντων και την οργάνωση συστατικών. Οι προγραμματιστές εφαρμογών δεν χρειάζεται να νοιάζονται για λεπτομέρειες και έτσι συγκεντρώνουν την προσπάθειά τους στην λογική της εφαρμογής (application logic). Το JSF περιλαμβάνεται σε κάθε διακομιστή Java EE[23] εφαρμογής, και μπορεί εύκολα να προστεθεί σε ένα αυτόνομο web container όπως ο Tomcat[24]. Ένα ακόμα

πλεονέκτημα είναι ότι έχει δοθεί ιδιαίτερη προσοχή στον σχεδιασμό του πλαισίου, ώστε το JSF να βελτιώνεται και να ενημερώνονται συνεχώς.

Εμείς θα εστιάσουμε στην έκδοση 2.0 του JSF (JSF 2.0), η οποία είναι πολύ πιο εύκολη στην χρήση από ότι η 1.x, καθώς παρέχει νέα και ισχυρά χαρακτηριστικά, όπως η εύκολη ενσωμάτωση Ajax και η αναβαθμισμένος τρόπος υλοποίησης προσαρμοσμένων συστατικών.

2.1.2 Ένα απλό παράδειγμα

Ας δούμε ένα απλό παράδειγμα μιας JSF εφαρμογής. Το παράδειγμα μας ξεκινά με μια σελίδα εισαγωγής (login screen).

Εικόνα 2: A login screen

Το αρχείο που παράγει την σελίδα της Εικόνας 2 είναι ουσιαστικά ένα XHTML αρχείο με μερικές πρόσθετες ετικέτες (JSF tags), όπως φαίνεται στην Εικόνα 3.

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4. <html xmlns="http://www.w3.org/1999/xhtml"
5.       xmlns:h="http://java.sun.com/jsf/html">
6.    <h:head>
7.       <title>Welcome</title>
8.    </h:head>
9.    <h:body>
10.      <h:form>
11.         <h3>Please enter your name and password.</h3>
12.         <table>
13.            <tr>
14.               <td>Name:</td>
15.               <td><h:inputText value="#{user.name}"/></td>
16.            </tr>
17.            <tr>
18.               <td>Password:</td>
19.               <td><h:inputSecret value="#{user.password}"/></td>
20.            </tr>
21.         </table>
22.         <p><h:commandButton value="Login" action="welcome"/></p>
23.      </h:form>
24.   </h:body>
25. </html>
```

Εικόνα 3: login/web/index.xhtml

Από την Εικόνα 3, παρατηρούμε τα εξής σημεία:

- Ένα πλήθος από τυποποιημένες HTML ετικέτες, όπως: <p>, <table>, κλπ..
- Ορισμένες JSF ετικέτες με πρόθεμα (prefix), όπως: <h:head>, <h:inputText>, κλπ.. Το χαρακτηριστικό (attribute) xmlns δηλώνει τον JSF χώρο ονομάτων (namespace).
- Οι ετικέτες <h:inputText>, <h:inputSecret>, <commandButton> αφορούν την περιοχή κειμένου (text field).
- Οι περιοχές κειμένου είναι συνδεδεμένες με αντικείμενα. Για παράδειγμα, το χαρακτηριστικό value=”#{user.name}” δηλώνει τη σύνδεση την περιοχής κειμένου με το πεδίο “name” του αντικειμένου “user”. Η σύνδεση αυτή γίνεται με την τεχνολογία Managed Beans[25] (δες Εικόνα 4).

```
import java.io.Serializable;
import javax.inject.Named;
   // or import javax.faces.bean.ManagedBean;
import javax.enterprise.context.SessionScoped;
   // or import javax.faces.bean.SessionScoped;

@Named("user") // or @ManagedBean(name="user")
@SessionScoped
public class UserBean implements Serializable {
   private String name;
```

Εικόνα 4: login/src/java/UserBean.java

Όταν ο χρήστης εισάγει το όνομα και τον κωδικό και κάνει κλικ στον κουμπί “Login”, τότε εμφανίζεται το αρχείο welcome.xhtml, όπως ορίζεται στο χαρακτηριστικό “action” της ετικέτας <h:commandButton>.

Εικόνα 5: The welcome page

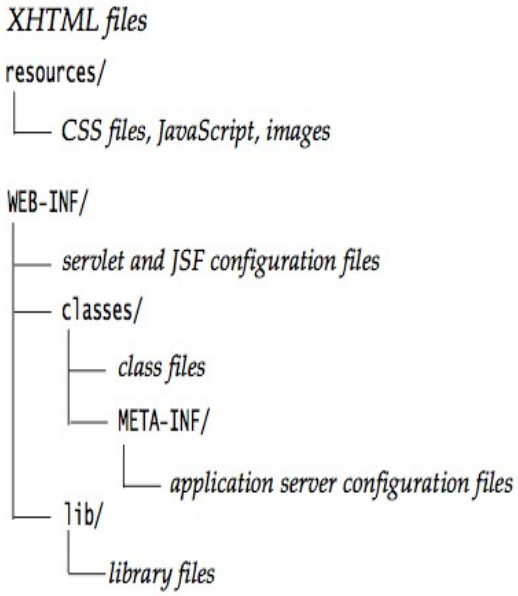
Και ο κώδικας για την “welcome page” είναι ακόμα πιο απλός, στην ίδια λογική όμως:

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3. "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4. <html xmlns="http://www.w3.org/1999/xhtml"
5.       xmlns:h="http://java.sun.com/jsf/html">
6.    <h:head>
7.       <title>Welcome</title>
8.    </h:head>
9.    <h:body>
10.      <h3>Welcome to JavaServer Faces, #{user.name}!</h3>
11.   </h:body>
12. </html>
```

Εικόνα 6: login/web/welcome.xhtml

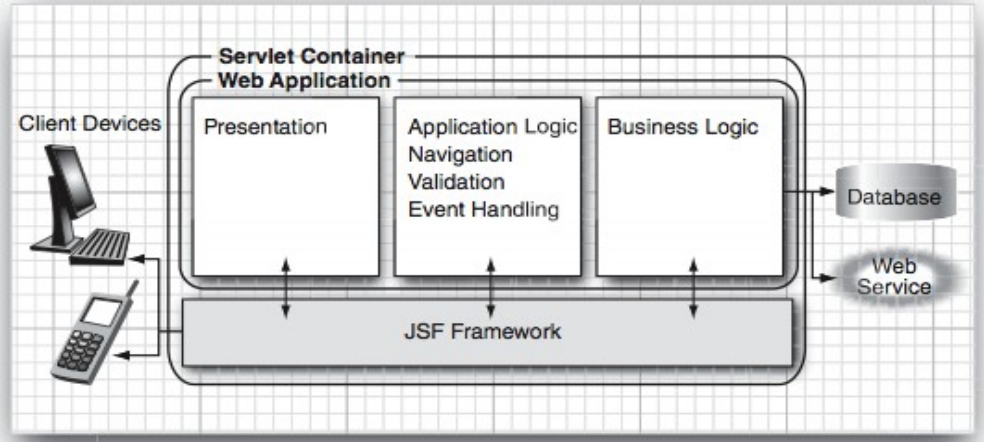
Το παραπάνω παράδειγμα παρόλο που δεν έχει τόσο σπουδαία λειτουργικότητα, παρουσιάζει ορισμένα βασικά αλλά και απαραίτητα συστατικά για την υλοποίηση μιας JSF εφαρμογής.

Οι εφαρμογές JSF γίνονται deployed ως αρχεία WAR[26] (συμπιεσμένα αρχεία με κατάληξη ".war"). Συνεπώς, η **δομή** του καταλόγου αρχείων μιας JSF εφαρμογής γενικά είναι η εξής:



Εικόνα 7: Δομή συμπιεσμένου αρχείου .war

Για να εξηγήσουμε καλύτερα τις υπηρεσίες που προσφέρει το πλαίσιο JSF στους προγραμματιστές, παρουσιάζουμε μια υψηλού επιπέδου επισκόπηση της αρχιτεκτονικής του στην Εικόνα 8.



Εικόνα 8: Επισκόπηση υψηλού επιπέδου του πλαισίου JSF

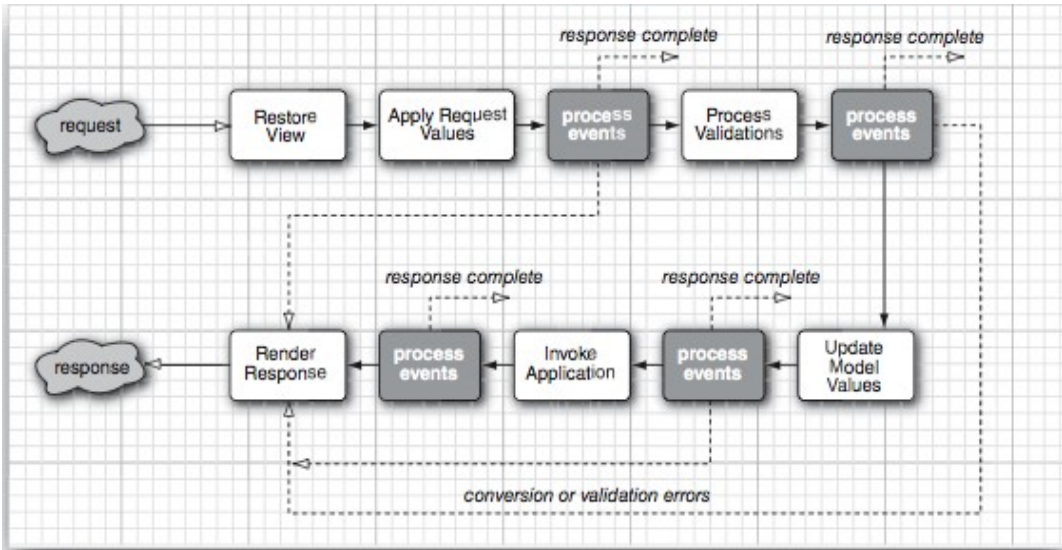
Όπως μπορούμε να δούμε, στην παραπάνω εικόνα, το πλαίσιο JSF είναι υπεύθυνο για την αλληλεπίδραση με συσκευές-πελάτες ενώ παρέχει τα εργαλεία για να "δέσει" την οπτική παρουσίαση (visual presentation), τη λογική της εφαρμογής (application logic) και την επιχειρηματική λογική (business logic) μιας Web εφαρμογής. Ωστόσο, η διατήρηση μιας βάσης δεδομένων και άλλες web-υπηρεσίες είναι έξω από το πεδίο του JSF.

2.1.3 Ο κύκλος ζωής του JSF και τα Events

Συχνά οι εφαρμογές Web πρέπει να αποκρίνονται σε εκδηλώσεις/συμβάντα χρήστη (user events), όπως για παράδειγμα, στην επιλογή ενός στοιχείου από ένα μενού, ή στο κλικ ενός κουμπιού (button) κλπ. Για το σκοπό αυτό, το πλαίσιο JSF, υποστηρίζει 4 κατηγορίες συμβάντων (events):

- *Value change events* (συμβάν για την αλλαγή τιμής ενός πεδίου ή μενού).
- *Action events* (συμβάν για την νέας σελίδας).
- *Phase events* (συμβάν για την μετάβαση σε άλλη φάση του κύκλου ζωής του JSF).
- *System events* (συμβάντα που λαμβάνουν χώρα πριν ή μετά από ειδικές ενέργειες του JSF, όπως δημιουργία/καταστροφή εφαρμογής, επικύρωση/προσθήκη/αποβολή συστατικών, κλπ).

Οι αιτήσεις (requests) υπόκεινται σε επεξεργασία από την υλοποίηση του JSF με έναν ελεγκτή Servlet[27] (controller Servlet), ο οποίος με την σειρά του εκτελεί τον κύκλο ζωής του JSF. Ο χειρισμός των συμβάντων φαίνεται στην Εικόνα 9:



Εικόνα 9: Χειρισμός συμβάντων κατά τον κύκλο ζωής του JSF

2.1.4 JSF και AJAX

Το AJAX (Asynchronous Javascript and XML) είναι μία τεχνολογία για την ενημέρωση μίας ιστοσελίδας στο πρόγραμμα περιήγησης (browser) χωρίς την υποβολή φόρμας και την φόρτωση (rendering) της απόκρισης (response). Η ιστοσελίδα περιέχει κατάλληλο κώδικα Javascript που επικοινωνεί με τον διακομιστή και κάνει αλλαγές στη δομή της σελίδας. Το αποτέλεσμα για τον χρήστη είναι μια πιο ομαλή εμπειρία περιήγησης από ότι η αλλαγή ολόκληρων σελίδων.

Το AJAX, που παλαιότερα θεωρούνταν "πολυτέλεια", αποτελεί πλέον απαραίτητο για την υλοποίηση σύγχρονων, γρήγορων και ανταγωνιστικών Web-εφαρμογών. Για το λόγω αυτό, το πλαίσιο JSF 2.0 έχει ενσωματωμένη υποστήριξη AJAX για τις standard ετικέτες του. Για παράδειγμα, στην Εικόνα 10 παρουσιάζουμε τον τρόπο ενσωμάτωσης Ajax συμπεριφοράς σε ένα πεδίο εισαγωγής κειμένου:

```
<h:inputText value="#{someBean.someProperty}">
   <f:ajax event="keyup" render="someOtherComponentId"/>
</h:inputText>
```

Εικόνα 10: Πεδίο εισαγωγής κειμένου με συμπεριφορά Ajax

Στο παράδειγμα της Εικόνας 10, το πεδίο εισαγωγής κειμένου παίρνει τιμή από κάποιο Java Bean. Κάθε φορά όμως που ο χρήστης αλλάζει την τιμή του πεδίου αυτού (πιο συγκεκριμένα, την στιγμή που επανέρχεται το πλήκτρο που πατήθηκε), ενεργοποιείται το event "keyup", και η ετικέτα f:ajax κάνει μία κλήση Ajax (Ajax call) προς τον Server, ο οποίος επεξεργάζεται την νέα τιμή. Όταν η

κλήση Ajax επιστρέψει, το JSF θα φορτώσει/εμφανίσει ένα συστατικό (component) με αναγνωριστικό κλειδί (id) ίσο με “someOtherComponentId”.

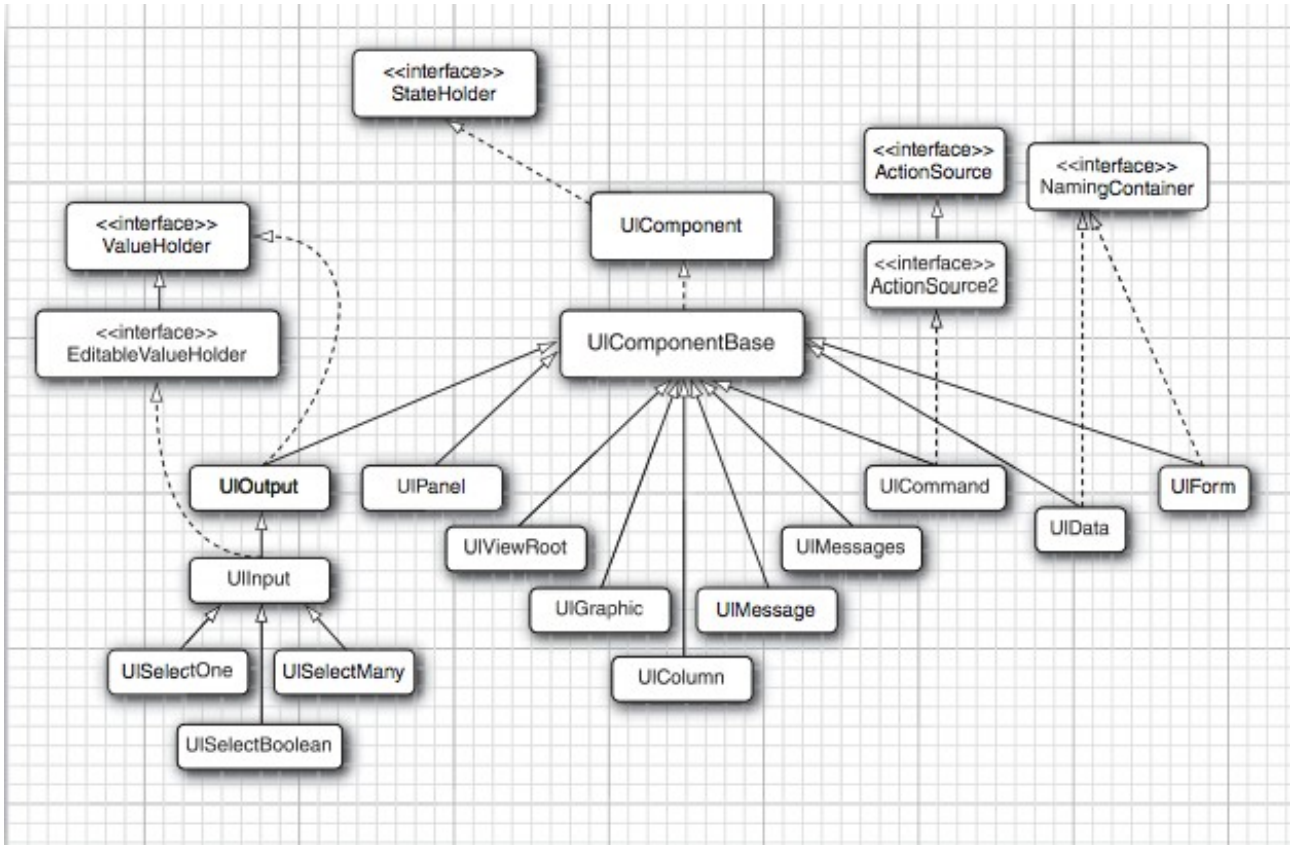
2.1.5 Προσαρμοσμένα συστατικά JSF (Custom JSF components)

Το πλαίσιο JSF παρέχει ένα βασικό σύνολο συστατικών (components) για την κατασκευή Web εφαρμογών βασισμένων σε HTML, όπως πεδία κειμένου (text fields), πλαίσια ελέγχου (checkboxes), κουμπιά (buttons), κλπ, αλλά και συνδυασμούς αυτών. Το JSF API όμως δεν σταματά εκεί. Παρέχει στους προγραμματιστές τη δυνατότητα σύνθεσης προσαρμοσμένων συστατικών (custom components) για πιο εξειδικευμένες/σύνθετες εργασίες. Αυτά σε συνδυασμό με αντίστοιχες προσαρμοσμένες ετικέτες συμβάλλουν στην επέκταση του JSF, έτσι ώστε να το καθιστούν ένα συνεχώς εξελισσόμενο ισχυρό εργαλείο ανάπτυξης Web εφαρμογών.

Κατά την υλοποίηση ενός προσαρμοσμένου συστατικού, ο προγραμματιστής επιλέγει ένα από τα βασικά συστατικά του JSF και το επεκτείνει. Κάθε προσαρμοσμένο συστατικό όμως θα πρέπει να πληρεί τις ακόλουθες προδιαγραφές:

1. *Να διατηρεί την κατάστασή του*, όπως οι τρέχουσες κάθε φορά τιμές των χαρακτηριστικών (attributes) του.
2. *Να κωδικοποιεί (encode) την διεπαφή χρήστη*, παράγοντας κώδικα (markup), συνήθως HTML ή/και Javascript.
3. *Να αποκωδικοποιεί αιτήσεις HTTP*, όπως για παράδειγμα, κλικ σε ένα κουμπί.

Επίσης, η κλάση ενός νέου συστατικού πρέπει να είναι υποκλάση της κλάσης UIComponent. Αυτή η υπερκλάση ορίζει πάνω από 40 αφηρημένες (abstract) μεθόδους. Στην Εικόνα 11 φαίνονται οι κλάσεις του JSF.



Εικόνα 11: Ιεραρχία κλάσεων συστατικών JSF

2.1.6 Η ιδέα για τη σουίτα JVFACES

Δουλεύοντας με το JSF, παρατηρήσαμε ότι δεν παρέχει υποστήριξη για γεωγραφικά δεδομένα. Εκεί βασίσαμε την ιδέα για τη δημιουργία της σουίτας JVFACES. Αυτή θα είναι ουσιαστικά ένα σύνολο από ετικέτες-συστατικά, που θα παρέχουν τη δυνατότητα υλοποίησης διεπαφών για δημιουργία και επεξεργασία χαρτών. Για την υλοποίηση όμως τέτοιων προσαρμοσμένων JSF συστατικών θα χρειαστούμε κι άλλα, ειδικά εργαλεία που αναλύονται στις επόμενες ενότητες.

2.2 *Η βιβλιοθήκη OpenLayers*

Η OpenLayers είναι μία βιβλιοθήκη αποκλειστικά σε Javascript για την απεικόνιση δεδομένων χάρτη στους περισσότερους σύγχρονους περιηγητές διαδικτύου, χωρίς εξαρτήσεις από την πλευρά του διακομιστή.

Εικόνα 12: Παράδειγμα χρήσης της OpenLayers από την επίσημη σελίδα της στο διαδίκτυο (www.openlayers.org)

Πηγαίος Κώδικας Javascript:

```
var map = new OpenLayers.Map("map");

var ol_wms = new OpenLayers.Layer.WMS(
    "OpenLayers WMS",
    "http://vmap0.tiles.osgeo.org/wms/vmap0",
    {layers: "basic"}
);

var dm_wms = new OpenLayers.Layer.WMS(
    "Canadian Data",
    "http://www2.dmsolutions.ca/cgi-bin/mswms_gmap",
    {
        layers: "bathymetry,land_fn,park,drain_fn,drainage," +
                "prov_bound,fedlimit,rail,road,popplace",
        transparent: "true",
        format: "image/png"
    },
    {isBaseLayer: false, visibility: false}
);

map.addLayers([ol_wms, dm_wms]);
map.addControl(new OpenLayers.Control.LayerSwitcher());
map.zoomToMaxExtent();
```

Η βιβλιοθήκη OpenLayers υλοποιεί μία συνεχώς αναπτυσσόμενη διεπαφή προγραμματισμού εφαρμογών (API) σε γλώσσα Javascript για την κατασκευή πλούσιων γεωγραφικών εφαρμογών βασισμένων στον διαδίκτυο, παρόμοια με το API του Google Maps και του MSN Virtual Earth, με μία σημαντική διαφορά – Η βιβλιοθήκη OpenLayers είναι ένα εντελώς δωρεάν λογισμικό, το οποίο αναπτύχθηκε από την Κοινότητα λογισμικού ανοικτού κώδικα. Από τον Νοέμβριο του 2007, η OpenLayers αποτελεί ένα από τα projects του Open Source Geospatial Foundation, ενός μη-κερδοσκοπικού, μη-κυβερνητικού οργανισμού, αποστολή του οποίου είναι να στηρίξει και να προωθήσει τη συνεργατική ανάπτυξη ανοικτών γεωχωρικών τεχνολογιών και δεδομένων.

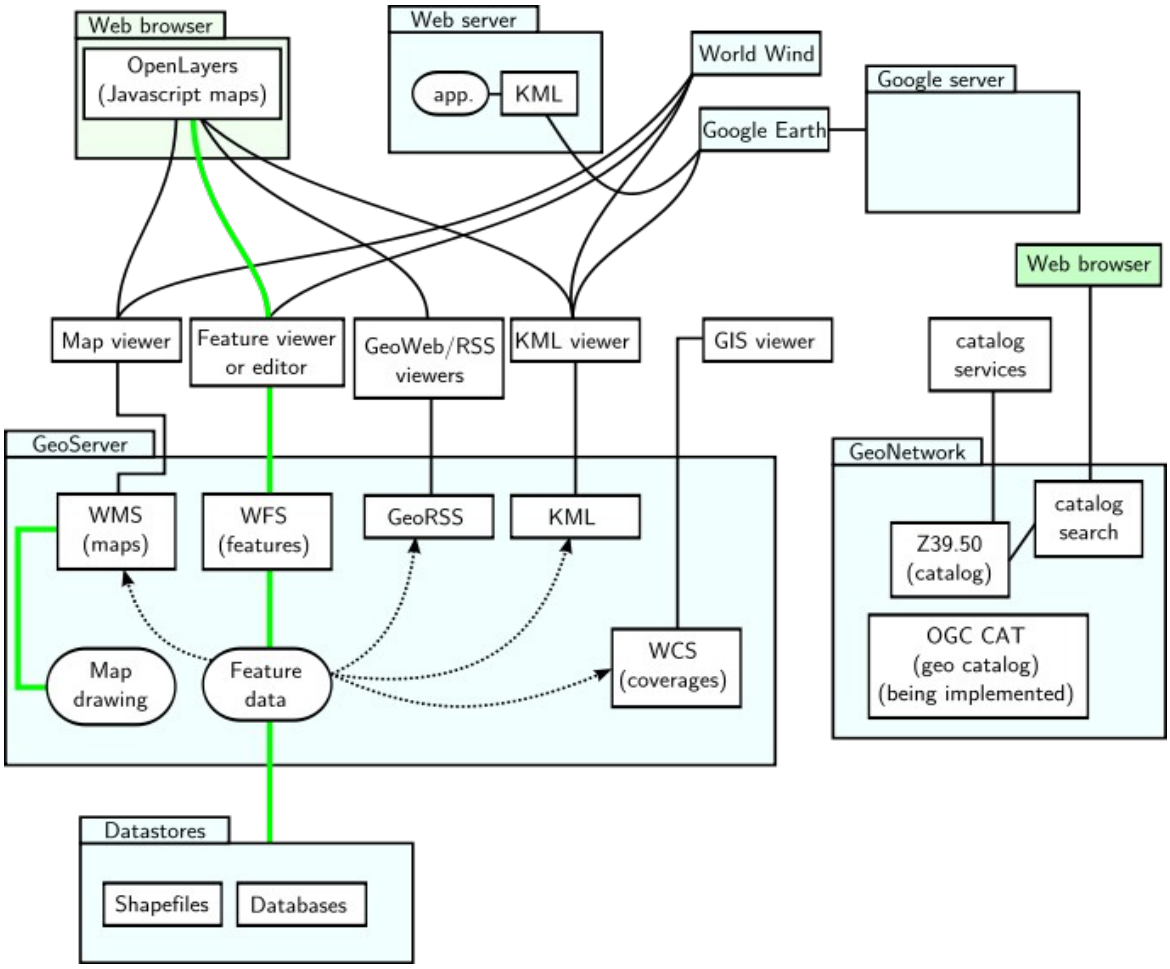
Στόχος της OpenLayers είναι να διαχωρίσει τα εργαλεία χάρτη (map tools) από τα δεδομένα χάρτη, έτσι ώστε να όλα τα εργαλεία να λειτουργήσουν σε όλες τις πηγές δεδομένων. Κάνει εύκολη την εισαγωγή ενός δυναμικού χάρτη σε οποιαδήποτε σελίδα του διαδικτύου. Επίσης μπορεί να εμφανίσει map tiles και markers, φορτώνοντας τα από οποιαδήποτε πηγή. Γενικά η βιβλιοθήκη OpenLayers έχει αναπτυχθεί για να αυξήσει τη χρήση γεωγραφικών πληροφοριών κάθε είδους.

Αναλυτικά δίνεται η δυνατότητα αν φορτωθούν χωρικά δεδομένα από τις εξής πηγές:

- *Web Map Service*
- *Web Feature Service*
- *Google Maps*
- *OpenStreetMap[28]*
- *Virtual Earth (Bing Maps)*
- *Yahoo! Maps*
- *UMN MapServer[29]*
- *MapGuide Open Source[30]*
- *GeoServer[31]*
- *ka-Map[32]*
- *World Wind servers[33]*
- *ArcGIS Server[34]*

και παρέχει υποστήριξη επίσης σε:

- GeoRSS[35]
- KML[36]
- GML[37]
- και GeoJSON[38]



Εικόνα 13: Η OpenLayers μπορεί να επικοινωνήσει μέσω διαφόρων πρωτοκόλλων.

Οι συναρτήσεις της βιβλιοθήκης OpenLayers, όπως αυτές περιγράφονται και στο class documentation (*http://dev.openlayers.org/releases/OpenLayers-2.11/doc/apidocs/files/OpenLayers-js.html*), θα αποδειχθούν εξαιρετικά χρήσιμες στην κατασκευή των JSF συστατικών.

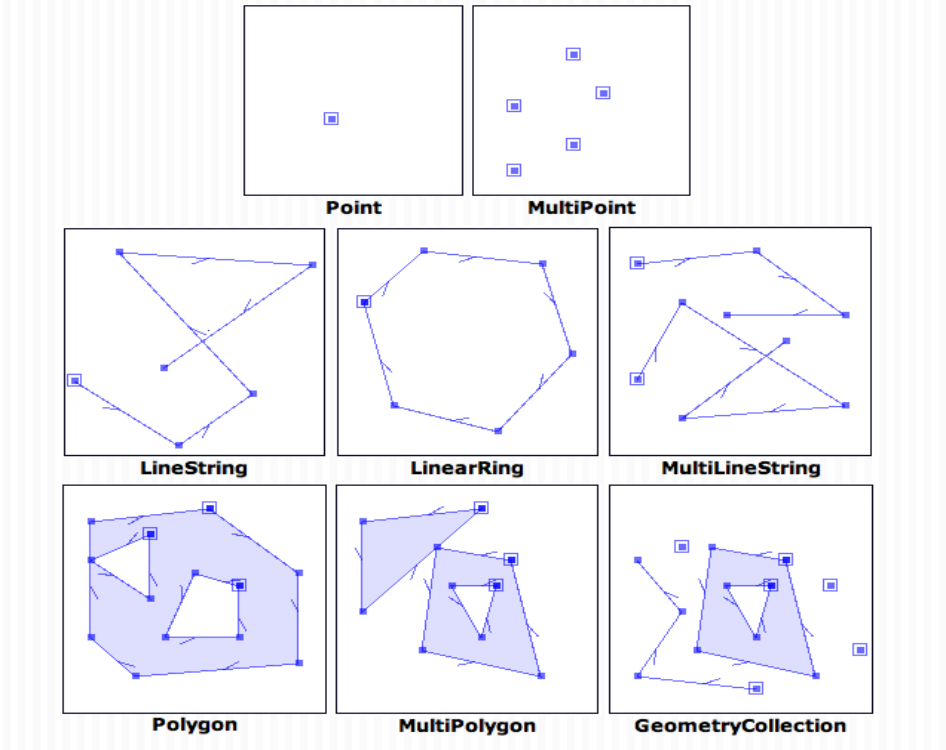
2.3 *Η βιβλιοθήκη Java Topology Suite (JTS)*

Η βιβλιοθήκη **JTS Topology Suite (JTS)** είναι μία διεπαφή προγραμματισμού εφαρμογών (API) για 2-διάστατα χωρικά κατηγορήματα (predicates) και συναρτήσεις.

Οι σχεδιαστικοί του στόχοι είναι οι εξής:

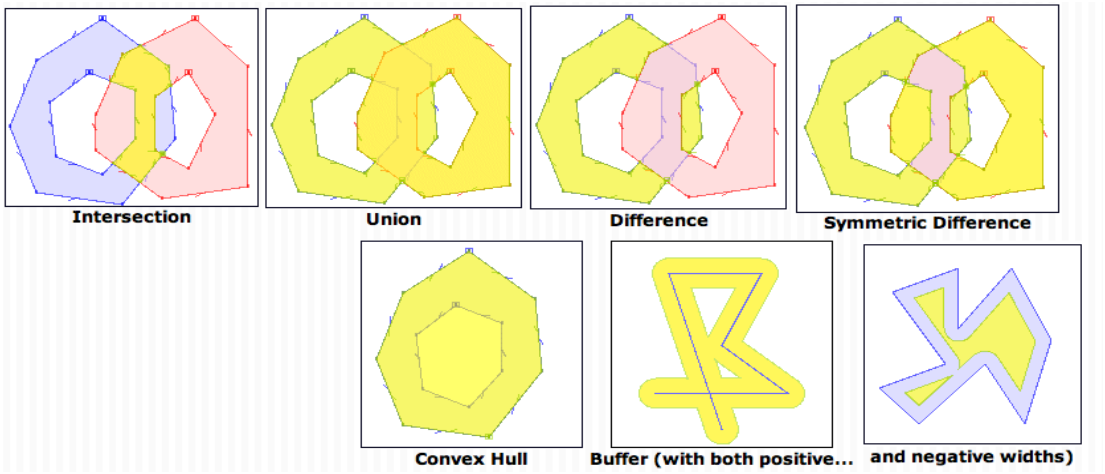
- Το JTS συμμορφώνεται στις προδιαγραφές του *Simple Features για SQL* (SFS)[39] που εκδίδονται από το Open GIS Consortium (OGC).
- Το JTS παρέχει μια πλήρη, περιεκτική και ισχυρή υλοποίηση 2-διάστατων χωρικών αλγορίθμών.
- Το JTS είναι αρκετά γρήγορο για χρήση στην παραγωγή.
- Το JTS είναι γραμμένο 100% σε καθαρή Java.
- Το JTS είναι ανοιχτός/ελεύθερος κώδικας υπό την άδεια του GNU Lesser General Public License (LGPL).

Το JTS παρέχει τους ακόλουθους τύπους χωρικών δεδομένων:



Εικόνα 14: Τύποι χωρικών δεδομένων που υποστηρίζονται από το JTS.

Επίσης το JTS υποστηρίζει βασικές μεθόδους χωρικής ανάλυσης. Αυτές δέχονται μία ή δύο Γεωμετρίες (Geometries) ως ορίσματα και επιστρέφουν ως αποτέλεσμα μία νέα Γεωμετρία. Οι εν λόγω μέθοδοι φαίνονται παρακάτω:



Εικόνα 15: Μέθοδοι χωρικής ανάλυσης από το JTS.

Η ανάπτυξη του JTS χρηματοδοτήθηκε από ένα κοινό σχέδιο των ακόλουθων οργανισμών:

- GeoConnections[40]
- British Columbia Ministry of Sustainable Resource Management (MSRM)[41]
- Centre for Topographic Information - Sherbrooke (CTI-S)[42]
- Στην Vivid Solutions, Inc. (VSI)[43] ανατέθηκε η σύμβαση για την ανάπτυξη του λογισμικού.

2.4 *Τεχνολογίες GIS και Υπηρεσίες WebGIS*

Ένα σύστημα γεωγραφικών πληροφοριών (GIS) είναι ένα σύστημα σχεδιασμένο για να λαμβάνει να αποθηκεύει, να (δια)χειρίζεται, να αναλύει και να παρουσιάζει όλα τα είδη γεωγραφικών δεδομένων.

Πολλές ειδικότητες μπορούν να επωφεληθούν από την τεχνολογία GIS. Η ενεργός αγορά GIS έχει οδηγήσει σε μείωση του κόστους και της συνεχούς βελτίωσης των στοιχείων υλικού και λογισμικού του GIS. Οι εξελίξεις αυτές θα οδηγήσουν, με τη σειρά τους, σε μια πολύ ευρύτερη χρήση της τεχνολογίας σε όλη την επιστήμη, την κυβέρνηση, τις επιχειρήσεις και τη βιομηχανία, με

εφαρμογές συμπεριλαμβανομένης της ακίνητης περιουσίας, της δημόσιας υγείας, τη χαρτογράφηση του εγκλήματος, την εθνική άμυνα, την αειφόρο ανάπτυξη, τους φυσικούς πόρους, την αρχιτεκτονική τοπίου, αρχαιολογία, τον περιφερειακό και κοινοτικό σχεδιασμό και τις μεταφορές.

2.4.1 Open Geospatial Consortium (OGC)

Το Open Geospatial consortium (OGC) είναι μία διεθνής, εθελοντική κοινοπραξία από πάνω από 400 εταιρείες, κυβερνητικούς οργανισμούς, πανεπιστήμια, και άτομα που συμμετέχουν σε μια διαδικασία συναίνεσης για την ανάπτυξη δημοσίων διαθέσιμων προδιαγραφών γεω-επεξεργασίας (geoprocessing).

Τα πρότυπα και τα πρωτόκολλα που καθορίζονται από τις προδιαγραφές OpenGIS[44] υποστηρίζουν διαλειτουργικές λύσεις, ώστε να δώσεις στους προγραμματιστές την κατάλληλη τεχνολογία για να δημιουργήσουν πολυσύνθετα χωρικά δεδομένα και υπηρεσίες, χρήσιμες για όλα τα είδη εφαρμογών.

Το OGC έχει υλοποιήσει πάνω από 30 πρότυπα (standards), στα οποία συμπεριλαμβάνονται και τα παρακάτω, που θα μας απασχολήσουν στην παρούσα εργασία:

- *WMS - (Web Map Service)*
- *WFS - (Web Feature Service)*
- *SFS - (Simple Feature for SQL)*
- *GML - (Geography Markup Language)*
- *SLD - (Styled Layer Descriptor)[45]*
- *KML - (Keyhole Markup Language)*

2.4.2 WMS

Το Web Map Service (WMS) είναι ένα πρωτόκολλο για την εξυπηρέτηση εικόνων χάρτη με γεω-αναφορά (georeferenced map images) μέσω του διαδικτύου που δημιουργούνται από ένα διακομιστή χάρτη χρησιμοποιώντας δεδομένα από μια GIS βάση δεδομένων. Οι προδιαγραφές του αναπτύχθηκαν για πρώτη φορά από το Open Geospatial Consortium (OGC) το 1999.

Το OGC κυκλοφόρησε την έκδοση 1.0.0 του WMS τον Απρίλιο του 2007, ακολούθησε η έκδοση 1.1.0 τον Ιούνιο του 2001 και η έκδοση 1.1.1 τον Ιανουάριο του 2002. Από τον Ιανουάριο του 2004, το OGC κυκλοφορεί η έκδοση 1.3.0 του WMS.

Το WMS ορίζει έναν πλήθος αιτημάτων διαφορετικών τύπων, δύο εκ των οποίων απαιτούνται από κάθε διακομιστή WMS:

- *GetCapabilities* – Επιστρέφει τις παραμέτρους σχετικά με το WMS και τα διαθέσιμα στρώματα (layers).
- *GetMap* – Με τις παραμέτρους που παρέχονται, επιστρέφει μία εικόνα τύπου χάρτη (map image).

Ορισμένοι WMS πάροχοι υποστηρίζουν προαιρετικά και τους εξής τύπους αιτημάτων:

- GetFeatureInfo
- DescribeLayer
- GetLegendGraphic

2.4.3 WFS

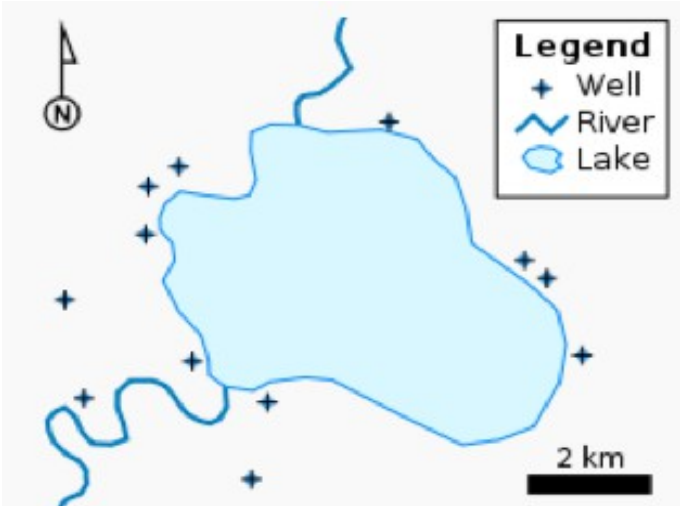
Το Web Feature Service (WFS), που αναπτύχθηκε από το OGC, παρέχει μία διεπαφή που επιτρέπει τις αιτήσεις για γεωγραφικά χαρακτηριστικά σε όλον τον παγκόσμιο ιστό χρησιμοποιώντας κλήσεις ανεξάρτητα από πλατφόρμα. Τα γεωγραφικά χαρακτηριστικά μπορεί κάποιος να τα φανταστεί σαν τον "πηγαίο κώδικα" πίσω από ένα χάρτη, σε αντίθεση με τη διεπαφή WMS ή τις online πύλες χαρτογράφησης όπως το Google Maps τα οποία επιστρέφουν μόνο μία εικόνα, την οποία οι τελικοί χρήστες δεν μπορούν να επεξεργαστούν ή να την αναλύσουν χωρικά.

Οι προδιαγραφές του WFS καθορίζουν διεπαφές για την περιγραφή εργασιών χειρισμού δεδομένων των γεωγραφικών χαρακτηριστικών. Οι συγκεκριμένες εργασίες χειρισμού δεδομένων περιλαμβάνουν δυνατότητες όπως:

- Να αναζητηθούν και να ληφθούν χαρακτηριστικά με βάση χωρικούς ή μη χωρικούς περιορισμούς.
- Να δημιουργηθούν νέα στιγμιότυπα χαρακτηριστικών.
- Να διαγραφούν στιγμιότυπα χαρακτηριστικών.
- Να ανανεωθούν στιγμιότυπα χαρακτηριστικών.

Γενικά, το WFS περιγράφει διαδικασίες εύρεσης, ανάκτησης ή μετασχηματισμού. Ο πελάτης (client) δημιουργεί μία αίτηση (request) και την στέλνει σε έναν *web feature server* μέσω HTTP. Ο Server αυτός στη συνέχεια επεξεργάζεται και εκτελεί την αίτηση που έλαβε.

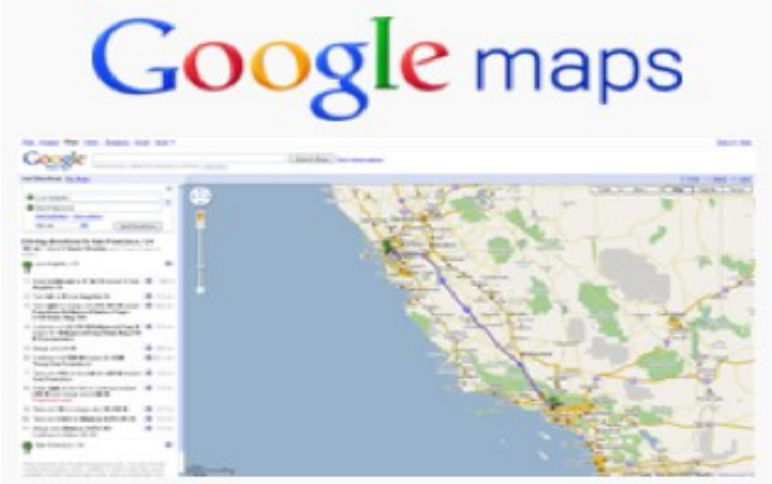
Για την ανταλλαγή δεδομένων ανάμεσα σε ένα Web Feature Server και σε έναν πελάτη (client) χρησιμοποιείται η γλώσσα GML (Geography Markup Language). Η GML είναι μία XML γραμματική που ορίστηκε από το OGC για τη περιγραφή κυρίως “vector” γεωγραφικών χαρακτηριστικών, όπως σημεία, γραμμές και πολύγωνα.



Εικόνα 16: Ένας vector χάρτης, με σημεία, γραμμές και πολύγωνα.

2.4.4 Google Maps API

Το Google Maps (πρώην Google Local) είναι μία εφαρμογή που προσφέρει υπηρεσίες web mapping, η οποία παρέχεται από την Google δωρεάν (για μη εμπορική χρήση).



Εικόνα 17: Screenshot of Google Maps showing a route from San Francisco to Los Angeles.

Τα τελευταία χρόνια με την εκρηκτική άνθιση των τεχνολογίας χαρτογράφησης και την επιτυχία του Google Maps, αναπτύχθηκαν κι άλλα αντίστοιχα API όπως το Yahoo! Maps API, Bing Maps Platform και το OpenLayers και άλλα. Ορισμένα από αυτά, όπως το Google Maps και το OpenLayers εκθέτουν το API τους ώστε να δώσουν τη δυνατότητα σε προγραμματιστές να δημιουργήσουν νέες προσαρμοσμένες εφαρμογές. Με αυτόν τον τρόπο, θα μπορέσουμε κι εμείς να αξιοποιήσουμε τα δύο αυτά API για να υλοποιήσουμε JSF συστατικά για τη σουίτα JVFACES.

3 Σχεδίαση του JVFACES

3.1 Τι είναι το JVFACES;

Το JVFACES είναι μία σουίτα προσαρμοσμένων συστατικών ανοικτού κώδικα για το JavaServer Faces. Περιλαμβάνει ένα σύνολο 15 JSF συστατικών που αφορούν γεωγραφικά δεδομένα και εκμεταλλεύονται την τεχνολογία AJAX. Κύριος στόχος της είναι να αποτελέσει ένα ισχυρό εργαλείο που θα επεκτείνει τις δυνατότητες του JSF και θα επιτρέπει στους JSF developers την δημιουργία διεπαφών γεωγραφικής πληροφορίας, με δηλωτικό τρόπο.

15 Components	**AJAX support**
Lightweight	**Easy to Use**

Πίνακας 2: Τι είναι το JVFACES;

- Σύνολο 15 συστατικών (map, layerWMS, layerGoogle, popup, control και άλλα).
- Ενσωματωμένη υποστήριξη AJAX.
- Συμβατότητα με όλους τους σύγχρονους browsers:
  - Mozilla Firefox
  - Microsoft Internet Explorer
  - Safari
  - Google Chrome
  - Opera
- Ένα και μοναδικό αρχείο jar, χωρίς πολλές απαιτήσεις και εξαρτήσεις από άλλες βιβλιοθήκες.
- Εκτενής τεκμηρίωση.

3.2 Εγκατάσταση

Το JVFACES αποτελείται από ένα και μοναδικό αρχείο jar, με όνομα: **jvfaces-1.0.jar**, το οποίο μπορείτε να το κατεβάσετε από την σελίδα: www.vincubator.softnet.tuc.gr/resources/jvfaces-1.0.jar.

Μοναδικές απαιτήσεις είναι ένα περιβάλλον με εγκατεστημένη την JAVA 5+ και μια υλοποίηση του JSF 2.0.

Αφού κατεβάσετε το αρχείο jar, η μόνη ρύθμιση που χρειάζεται είναι να ορίσετε το κατάλληλο classpath στην σελίδα σας ώστε να μπορείτε να χρησιμοποιείτε τα συστατικά (components) του JVFACES.

```
<html xmlns=”http://www.w3c.org/1999/xhtml”
        xmlns:h=”http://java.sun.com/jsf/html”
        xmlns:f="http://java.sun.com/jsf/core"
        xmlns:jv="http://gr.tuc.softnet.jsfgis">

<h:head>
</h:head>

<h:body>

      <jv:map   jsVar=”map1” div=”map1” width=”512px” height=”256px” projection=”4326”>

            <jv:layerWMS    jsVar=”basicLayer” name=”wms_layer”
                            url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                            layers="basic"
                            isBaseLayer="true">
            </jv:layerWMS>

             ....

      </jv:map>

</h:body>

</html>
```

Πίνακας 3: Ένα απλό παράδειγμα.

3.3 Συστατικά της σουίτας JVFACES

Τα συστατικά της σουίτας JVFACES είναι Java κλάσεις που επεκτείνουν άμεσα ή έμμεσα την κλάση “UIComponentBase” του JSF framework παράγοντας κώδικα HTML και javascript, σύμφωνα με τις προδιαγραφές του JSF 2.0 για την υλοποίηση “custom UI components” και εκμεταλλευόμενοι το javascript API για JSF, αλλά και το AJAX integration για JSF. Επίσης χρησιμοποιήθηκαν εκτενώς οι javascript συναρτήσεις της βιβλιοθήκης OpenLayers, σύμφωνα με τις προδιαγραφές του OpenLayers API, που βασίζουν τη λειτουργία τους στην τεχνολογία AJAX.

3.3.1 Map

Το συστατικό “Map” χρησιμοποιείται με την ετικέτα “map” μέσα σε ένα αρχείο xhtml για να δημιουργηθεί ένα div, με δοθείσες διαστάσεις, στο οποίο προορίζουμε να εμφανίσουμε κάποιο χάρτη. Για την ακρίβεια, κάποια στρώματα (layers). Επίσης καθορίζουμε το κέντρο συντεταγμένων του χάρτη, την αρχική εστίαση (zoom) και τη μορφή προβολής (projection).

Η ετικέτα “map” κάνει, εκτός των άλλων, χρήση της συνάρτησης OpenLayers.Map, παραμετροποιώντας την κατάλληλα, και δεν έχει κάποιο οπτικό αποτέλεσμα. Επομένως δεν θα πρέπει να χρησιμοποιείται “μόνη της”, αλλά σε συνδυασμό με μία τουλάχιστον ετικέτα-παιδί, που εισάγει κάποιο στρώμα (layer) στο χάρτη. Αυτές οι ετικέτες θα αναλυθούν σε επόμενες ενότητες.

Περιορισμοί

Η ετικέτα “map” σχεδιάστηκε για να είναι ετικέτα-γονέας, άμεσος ή έμμεσος, για τις άλλες ετικέτες της σουίτας.

Πληροφορίες

Όνομα Ετικέτας	**map**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLMap**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIComponentBase**

Πίνακας 4: Πληροφορίες για το συστατικό "Map"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
jsVar	String	Ότι ισχύει για τα ονόματα μεταβλητών της javascript.	Το όνομα της javascript μεταβλητής που θα δημιουργηθεί για το χάρτη.
div	String	οποιαδήποτε	Το id του div που θα δημιουργηθεί για το χάρτη.
width	Integer+String	Μορφή: <integer> + ”px” π.χ.: 512px	Το πλάτος του χάρτη.
height	Integer+String	Μορφή: <integer> + ”px” π.χ.: 256px	Το ύψος του χάρτη.
projection	Integer	Επιτρεπτές τιμές: “4326” ή “900913”	Ο τύπος προβολής του χάρτη. Αν χρησιμοποιήσουμε Google ή Yahoo layer τότε θέτουμε απαραιτήτως την τιμή “900913”.
center	Float+String+ Float	Μορφή: <float> + “,” + <float> π.χ.: 5.0,13.6	Καθορίζει το κέντρο του χάρτη.
zoom	Integer	Επιτρεπτές τιμές: 1 έως 16 (αναλόγως ποια layers χρησιμοποιούμε)	Καθορίζει την αρχική εστίαση του χάρτη.

Πίνακας 5: Χαρακτηριστικά της ετικέτας "map"

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” height=”256px” projection=”4326”
           center=”0.0,0.0” zoom=”1”>

      (ετικέτα/ες για εισαγωγή layer)
           ....

</jv:map>
```

Πίνακας 6: Χρήση της ετικέτας "map"

Επεξήγηση

Στον πίνακα 6, παρουσιάζεται η ετικέτα “map” παραμετροποιημένη ώστε να δημιουργεί ένα div, σε μια σελίδα xhtml, διαστάσεων 512 x 256 pixels, και το προετοιμάζει ώστε να παρουσιάσει στρώματα χάρτη, με προβολή τύπου “4326”, κεντραρισμένα στις συντεταγμένες “5.0”, “13.6” και με αρχική εστίαση επιπέδου 1 (δηλαδή χωρίς zoom).

3.3.2 LayerWMS

Το συστατικό “LayerWMS” χρησιμοποιείται με την ετικέτα “layerWMS” μέσα σε ένα αρχείο xhtml για να δημιουργηθεί ένα στρώμα (layer) WMS, δηλαδή δεδομένα από το OGC Web Map Service, πάνω σε ένα χάρτη.

Η ετικέτα “layerWMS” κάνει, εκτός των άλλων, χρήση της συνάρτησης OpenLayers.Layer.WMS, παραμετροποιώντας την ως προς το όνομα του στρώματος, τη διεύθυνση που βρίσκεται το στρώμα, τα υποστρώματα από τα οποία αποτελείται και επιθυμούμε να προβάλλουμε και το φορμά του αρχείου που αιτούμαστε. Επίσης καθορίζουμε αν θα είναι αυτό το στρώμα, το βασικό στρώμα του χάρτη μας και/ή αν το στρώμα θα είναι διαφανές (transparent).

Περιορισμοί

- Η ετικέτα “layerWMS” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “map”.
- Σε μια ετικέτα “map” πρέπει να υπάρχει τουλάχιστον μία ετικέτα “layerWMS”.
- Σε μια ετικέτα “map” μπορούν να υπάρχουν πάνω από μία ετικέτες “layerWMS”.
- Σε μια ετικέτα “map” θα πρέπει να υπάρχει ακριβώς μία ετικέτα “layerWMS” που θα ορίζει το βασικό στρώμα.

Πληροφορίες

Όνομα Ετικέτας	**layerWMS**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLLayerWMS**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

Πίνακας 7: Πληροφορίες για το συστατικό "LayerWMS"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
jsVar	String	Ότι ισχύει για τα ονόματα μεταβλητών της javascript.	Το όνομα της javascript μεταβλητής που θα δημιουργηθεί για το στρώμα.
name	String	οποιαδήποτε	Το όνομα του στρώματος που θα δημιουργηθεί στο χάρτη.
url	String	π.χ.: http://vmap0.tiles.osgeo.org/wms/vmap0?	Η διεύθυνση όπου είναι αποθηκευμένο το στρώμα του αιτούμαστε.
layers	String,String,...	Μορφή: <String> + ”,”+ <String> +... π.χ.: bathymetry,park	Τα ονόματα των υποστρωμάτων του WMS στρώματος που αιτούμαστε.
format	String	Ονόματα γνωστών φορμά. Συνήθως: “image/jpeg”	Το φορμά του αρχείου που αποτελεί το στρώμα που αιτούμαστε.
isBaseLayer	Boolean	Επιτρεπτές τιμές: “true” ή “false”	Καθορίζει το βασικό στρώμα του χάρτη.
transparent	Boolean	Επιτρεπτές τιμές: “true” ή “false”	Καθορίζει την “διαφάνεια” του στρώματος. (Από ένα διαφανές στρώμα, διακρίνουμε και τα άλλα στρώματα που βρίσκονται πίσω του.

Πίνακας 8: Χαρακτηριστικά της ετικέτας "layerWMS"

Χρήση

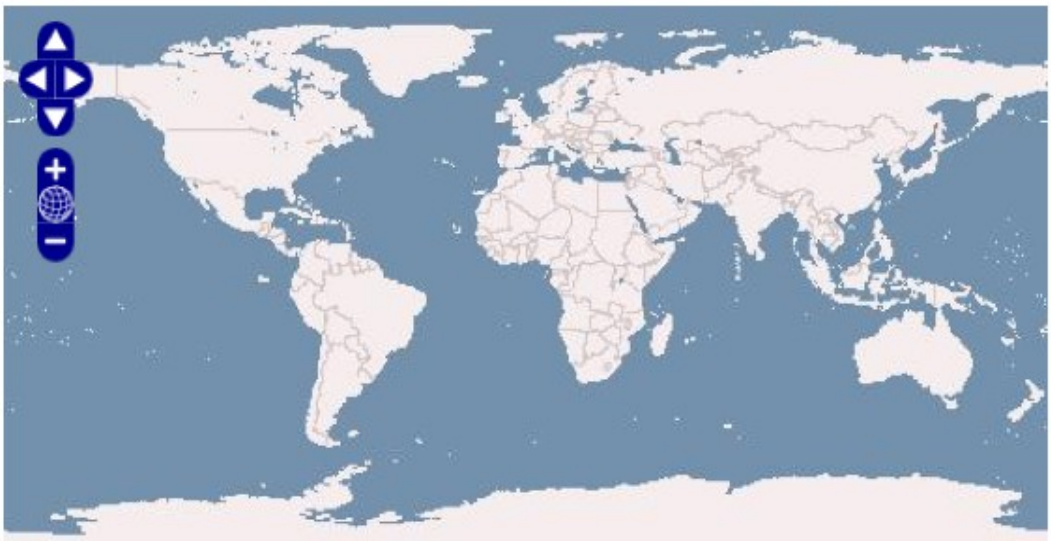
```
<jv:map    jsVar=”map1” div=”map1” width=”512px” height=”256px” projection=”4326”
           center=”0.0,0.0” zoom=”1”>

    <jv:layerWMS    jsVar=”basicLayer” name=”OpenLayers WMS” layers=”basic”
                    url=”http://vmap0.tiles.osgeo.org/wms/vmap0?” isBaseLayer=”true”>
    </jv:layerWMS>

</jv:map>
```

Πίνακας 9: Χρήση της ετικέτας "layerWMS"

Επεξήγηση

Στον πίνακα 9, παρουσιάζεται η ετικέτα “layerWMS” παραμετροποιημένη ώστε να δημιουργεί ένα στρώμα με όνομα “OpenLayers WMS”, το οποίο βρίσκεται στην διεύθυνση “http://vmap0.tiles.osgeo.org/wms/vmap0?”. Ζητήσαμε μόνο ένα υπόστρωμα, αυτό με όνομα “basic”. Εισάγεται στον χάρτη μας, που προηγουμένως με την ετικέτα “map” δημιουργήθηκε και ορίσαμε το στρώμα αυτό ως βασικό στρώμα του.

Αποτέλεσμα



Εικόνα 18: Παράδειγμα χρήσης της ετικέτας "layerWMS"

3.3.3 LayerGoogle

Το συστατικό “LayerGoogle” χρησιμοποιείται με την ετικέτα “layerGoogle” μέσα σε ένα αρχείο xhtml για να δημιουργηθεί ένα στρώμα (layer) της Google, από το Google Maps API Version 2, πάνω σε ένα χάρτη, ορισμένο ως sphericalMercator (δηλαδή το στρώμα αυτό είναι μπορεί να αλληλεπιδρά με τον υπόλοιπο χάρτη, να μοιράζεται κοινό τύπο προβολής με άλλα layers του ίδιου χάρτη και να επιτρέπει την σχεδίαση διανυσματικών δεδομένων πάνω του).

Η ετικέτα “layerGoogle” κάνει, εκτός των άλλων, χρήση της συνάρτησης OpenLayers.Layer.Google, παραμετροποιώντας την ως προς το όνομα του στρώματος και τον τύπο του Google στρώματος που αιτούμαστε. Επίσης καθορίζουμε, ανάλογα με την σειρά που θα το δηλώσουμε, αν το στρώμα αυτό θα είναι το βασικό στρώμα του χάρτη μας.

Περιορισμοί

- Η ετικέτα “layerGoogle” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “map” αλλά η δεύτερη θα πρέπει να έχει attribute “projection” ίσο με “900913.
- Σε μια ετικέτα “map” μπορούν να υπάρχουν πάνω από μία ετικέτες “layerGoogle” με διαφορετικό attribute “type”.
- Τα στρώματα Google είναι αδιαφανή και μπορούν να χρησιμοποιηθούν μαζί με στρώματα WMS, αλλά και με άλλα στρώματα που θα αναλύσουμε σε επόμενες ενότητες, στον ίδιο χάρτη.
- Για να αποκτήσουμε άδεια χρήσης των στρωμάτων της Google, θα πρέπει το αρχείο xhtml που θέλουμε να εμφανιστούν να περιέχει στο “h:head” του το script:

```
<script type="text/javascript" src="http://maps.google.com/maps?file=api&v=2&key=”To key που μας δόθηκε από την Google”/>
```

Πίνακας 10: Απαραίτητο script για χρήση Google layer.

Πληροφορίες

Όνομα Ετικέτας	**layerGoogle**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLLayerGoogle**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

Πίνακας 11: Πληροφορίες για το συστατικό "LayerGoogle"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
jsVar	String	Ότι ισχύει για τα ονόματα μεταβλητών της javascript.	Το όνομα της javascript μεταβλητής που θα δημιουργηθεί για το στρώμα.
name	String	οποιαδήποτε	Το όνομα του στρώματος που θα δημιουργηθεί στο χάρτη.
type	String	Επιτρεπτές τιμές: 1) G_PHYSICAL_MAP	Δημιουργεί το G_PHYSICAL_MAP layer της Google.
		2) G_HYBRID_MAP	Δημιουργεί το G_HYBRID_MAP layer της Google.
		3) G_SATELLITE_MAP	Δημιουργεί το G_SATELLITE_MAP layer της Google.
		4) Καμία δήλωση του attribute “type”	Δημιουργεί το Google Street layer της Google.

Πίνακας 12: Χαρακτηριστικά της ετικέτας "layerWMS"

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” center=”0.0,0.0” zoom=”1”
           height=”512px” projection=”900913”>

      <jv:layerGoogle    jsVar=”gmap1” name=”Google hybrid” type=”G_HYBRID_MAP”/>

</jv:map>
```

Πίνακας 13: Χρήση της ετικέτας "layerGoogle"

Επεξήγηση

Στον πίνακα 13, παρουσιάζεται η ετικέτα “layerGoogle” παραμετροποιημένη ώστε να δημιουργεί ένα στρώμα με όνομα “Google hybrid” και είναι τύπου “G_HYBRID_MAP”. Εισάγεται στον χάρτη μας, που προηγουμένως με την ετικέτα “map” δημιουργήθηκε. Προσέξτε ότι σε σχέση με τα προηγούμενα παραδείγματα, έχουμε αλλάξει την τιμή του attribute “projection” και του “height”, της ετικέτας “map”, κατάλληλα ώστε να εμφανίζεται σωστά το στρώμα της Google.

Αποτέλεσμα

Εικόνα 19: Παράδειγμα χρήσης της ετικέτας layerGoogle.

3.3.4 LayerYahoo

Το συστατικό “LayerYahoo” χρησιμοποιείται με την ετικέτα “layerYahoo” μέσα σε ένα αρχείο xhtml για να δημιουργηθεί ένα στρώμα (layer) της Yahoo, από το Yahoo Maps API Version 3, πάνω σε ένα χάρτη, ορισμένο ως sphericalMercator (δηλαδή το στρώμα αυτό είναι μπορεί να αλληλεπιδρά με τον υπόλοιπο χάρτη, να μοιράζεται κοινό τύπο προβολής με άλλα layers του ίδιου χάρτη και να επιτρέπει την σχεδίαση διανυσματικών δεδομένων πάνω του).

Η ετικέτα “layerYahoo” κάνει, εκτός των άλλων, χρήση της συνάρτησης OpenLayers.Layer.Yahoo, παραμετροποιώντας την ως προς το όνομα του στρώματος και τον τύπο του Yahoo στρώματος που αιτούμαστε. Επίσης καθορίζουμε, ανάλογα με την σειρά που θα το δηλώσουμε, αν το στρώμα αυτό θα είναι το βασικό στρώμα του χάρτη μας.

Περιορισμοί

- Η ετικέτα “layerYahoo” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “map” αλλά η δεύτερη θα πρέπει να έχει attribute “projection” ίσο με “900913.
- Σε μια ετικέτα “map” μπορούν να υπάρχουν πάνω από μία ετικέτες “layerYahoo” με διαφορετικό attribute “type”.
- Τα στρώματα Yahoo είναι αδιαφανή και μπορούν να χρησιμοποιηθούν μαζί με στρώματα WMS, Google αλλά και άλλα στρώματα που θα αναλύσουμε σε επόμενες ενότητες στον ίδιο χάρτη.
- Για να αποκτήσουμε άδεια χρήσης των στρωμάτων της Yahoo, θα πρέπει το αρχείο xhtml που θέλουμε να εμφανιστούν να περιέχει στο “h:head” του το script:

```
<script type="text/javascript" src="http://api.maps.yahoo.com/ajaxymap?v=3.0&appid=”To key που μας δόθηκε από την Yahoo”/>
```

Πίνακας 14: Απαραίτητο script για χρήση Yahoo layer.

Πληροφορίες

Όνομα Ετικέτας	**layerYahoo**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLLayerYahoo**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

Πίνακας 15: Πληροφορίες για το συστατικό "LayerYahoo"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
jsVar	String	Ότι ισχύει για τα ονόματα μεταβλητών της javascript.	Το όνομα της javascript μεταβλητής που θα δημιουργηθεί για το στρώμα.
name	String	οποιαδήποτε	Το όνομα του στρώματος που θα δημιουργηθεί στο χάρτη.
type	String	Επιτρεπτές τιμές:	
		1) YAHOO_MAP_HYB	Δημιουργεί το Yahoo Hybrid layer της Yahoo.
		2) YAHOO_MAP_SAT	Δημιουργεί το Yahoo Satellite layer της Yahoo.
		3) Καμία δήλωση του attribute “type”	Δημιουργεί το Yahoo Street layer της Yahoo.

Πίνακας 16: Χαρακτηριστικά της ετικέτας "layerYahoo"

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” center=”0.0,0.0” zoom=”1”
           height=”512px” projection=”900913”>

      <jv:layerYahoo    jsVar=”ymap1” name=”Yahoo satellite” type=”YAHOO_MAP_SAT”/>

</jv:map>
```

Πίνακας 17: Χρήση της ετικέτας "layerYahoo"

Επεξήγηση

Στον πίνακα 17, παρουσιάζεται η ετικέτα “layerYahoo” παραμετροποιημένη ώστε να δημιουργεί ένα στρώμα με όνομα “Yahoo satellite” και είναι τύπου “YAHOO_MAP_SAT”. Εισάγεται στον χάρτη μας, που προηγουμένως με την ετικέτα “map” δημιουργήθηκε. Προσέξτε ότι και εδώ έχουμε θέσει την τιμή του attribute “projection” και του “height”, της ετικέτας “map”, κατάλληλα ώστε να εμφανίζεται σωστά το στρώμα της Yahoo.

Αποτέλεσμα

Εικόνα 20: Παράδειγμα χρήσης της ετικέτας layerYahoo.

3.3.5 Control

Το συστατικό “Control” χρησιμοποιείται με την ετικέτα “control” μέσα σε ένα αρχείο xhtml για να αλληλεπιδρά με το χάρτη, δίνοντας στον χρήστη τη δυνατότητα να αλλάζει το περιεχόμενο που προβάλλει κάθε στιγμή ο χάρτης.

Με την ετικέτα “control” μπορούμε να δώσουμε σε ένα χάρτη, 4 είδη “control”:

1. Δυνατότητα εναλλαγής στρώματος.
2. Εμφάνιση της τρέχουσας θέσης του δείκτη του ποντικιού.
3. Δυνατότητα διαχείρισης του ιστορικού πλοήγησης στο χάρτη.
4. Εμφάνιση μικρογραφίας της πλαισίου που παρουσιάζει ο χάρτης, π.χ. σε περίπτωση που έχουμε εστιάσει κάπου στο χάρτη.
5. Δυνατότητα εναλλαγής της λειτουργίας του αριστερού πλήκτρου του ποντικιού πάνω στο χάρτη:
   a) Προκαθορισμένη λειτουργία αριστερού πλήκτρου ποντικιού. (*κρατώντας πατημένο το αριστερό πλήκτρο και κίνηση του ποντικιού κάνουμε drag το χάρτη*)
   b) Λειτουργία “zoomBox”. (*κρατώντας πατημένο το αριστερό πλήκτρο και κίνηση του ποντικιού σχηματίζουμε ένα ορθογώνιο, στο οποίο θα εστιάσει ο χάρτης, με το που θα αφήσουμε το πλήκτρο να επανέλθει στην αρχική του θέση*)

Περιορισμοί

- Η ετικέτα “control” σχεδιάστηκε για να είναι έμμεση ετικέτα-παιδί της ετικέτας “map”. Πιο συγκεκριμένα, πρέπει να περικλείεται από μία ετικέτα “h:form”, η οποία είναι άμεση ετικέτα-παιδί της ετικέτας “map”.
- Σε μια ετικέτα “map” μπορούν να υπάρχουν πάνω από μία ετικέτες “control” με διαφορετικό attribute “type”. Συγκεκριμένα, μέχρι 5, όσα δηλαδή και τα διαφορετικά είδη control.
- Μία ετικέτα “h:form” μπορεί να περιλαμβάνει 1 έως 5 ετικέτες “control”, ή κάθε ετικέτα “control” να εμπεριέχεται σε ξεχωριστή ετικέτα “form” (ανάλογα με τις απαιτήσεις της εφαρμογής που υλοποιούμε).
- Με το attribute “value” της ετικέτας “control” μπορούμε θέσουμε ως ενεργό ή ανενεργό ένα “control” ή να το συνδέσουμε με κάποιο managed bean.

Πληροφορίες

Όνομα Ετικέτας	**control**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLControl**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

Πίνακας 18: Πληροφορίες για το συστατικό "Control"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
type	String	Επιτρεπτές τιμές:	
		1) “LayerSwitcher”	Δυνατότητα εναλλαγής στρωμάτων.
		2) “MousePosition”	Εμφάνιση τρέχουσας θέσης του δείκτη του ποντικιού.
		3) “OverviewMap”	Εμφάνιση μικρογραφίας του χάρτη.
		4) “NavToolbar”	Δυνατότητα διαχείρισης του ιστορικού πλοήγησης στο χάρτη.
		5) “Panel”	Εναλλαγή λειτουργίας αριστερού πλήκτρου ποντικιού.
value	Boolean ή MethodExpression	Επιτρεπτές τιμές: “true”, “false”, #{someBean.someProperty}	Καθορίζει αν το “control” είναι ενεργό ή μη ενεργό.

Πίνακας 19: Χαρακτηριστικά της ετικέτας "control"

Χρήση

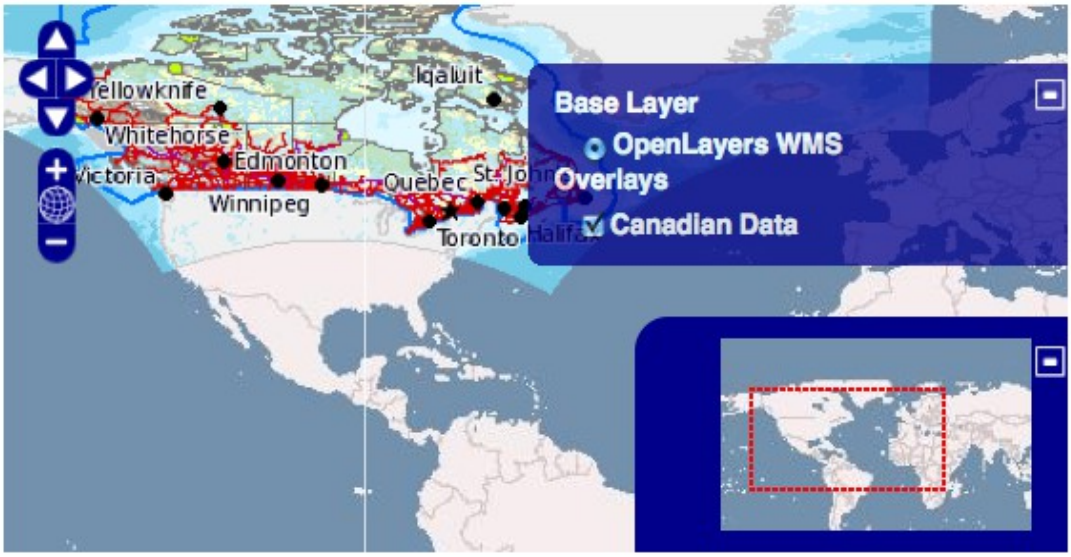
```
<jv:map    jsVar=”map1” div=”map1” width=”512px” height=”512px” projection=”900913”>
      <jv:layerWMS  jsVar="basicLayer" name="OpenLayers WMS"
                          url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                          layers="basic" isBaseLayer="true">
      </jv:layerWMS>
      <jv:layerWMS jsVar="dm_wms" name="Canadian Data"
                         url="http://www2.dmsolutions.ca/cgi-bin/mswms_gmap"
                         layers="bathymetry,land_fn,park,drain_fn,drainage,prov_bound,fedlimit,rail,road,popplace"
                         format="image/jpeg" isBaseLayer="false" transparent="true">
      </jv:layerWMS>
      <h:form>
            <control   type=”LayerSwitcher” value=”true”/>
            <control   type=”OverviewMap” value=”true”>
      </h:form>

</jv:map>
```

Πίνακας 20: Χρήση της ετικέτας "control"

Επεξήγηση

Στον πίνακα 20, παρουσιάζεται η ετικέτα “control”. Χρησιμοποιούμε την ετικέτα 2 φορές για να εισάγουμε στο χάρτη μας 2 διαφορετικά είδη control. Την πρώτη φορά, παραμετροποιήσαμε την ετικέτα “control” έτσι ώστε να δημιουργεί ένα control τύπου “LayerSwitcher” ενώ την δεύτερη ένα control τύπου “OverviewMap”. Θέσαμε τα 2 αυτά controls ενεργά με την τιμή “true” στο attribute “value” και τα εισάγαμε στο χάρτη, τοποθετώντας τα μέσα σε φόρμα, η οποία με τη σειρά της εμπεριέχεται στην ετικέτα “map”. Στον συγκεκριμένο χάρτη έχουμε δημιουργήσει προηγουμένως 2 στρώματα WMS για να φανεί καλύτερα η λειτουργία του “LayerSwitcher control”.

Αποτέλεσμα



Εικόνα 21: Παράδειγμα χρήσης της ετικέτας "control".

3.3.6 LayerVector

Το συστατικό “LayerVector” χρησιμοποιείται με την ετικέτα “layerVector” μέσα σε ένα αρχείο xhtml για να δημιουργηθεί ένα στρώμα διανυσμάτων (vector layer), δηλαδή ένα στρώμα που παρουσιάζει διανυσματικά δεδομένα από διάφορες πηγές.

Τα διανυσματικά δεδομένα είναι σχήματα (σημεία, γραμμές και πολύγωνα) και μπορούν να περιλαμβάνουν επίσης πεδία με θεματικά δεδομένα τύπου String, Integer, Float κλπ., για παράδειγμα ένα πολύγωνο που περικλείει μία την Ελλάδα, που έχει ένα πεδίο “όνομα” με τιμή “Ελλάδα”, ένα πεδίο “πληθυσμός με τιμή “10.000.000”.

Οι πηγές από τις οποίες μπορούμε να αντλήσουμε διανυσματικά δεδομένα, μπορεί να είναι 3 τύπων:

1. Ένα managed bean.
2. Κάποιο Web Feature Service (WFS).
3. Ένα αρχείο xml (σε gml format) που βρίσκεται οπουδήποτε στο Web.

Και οι τρεις παραπάνω πηγές υποστηρίζονται από την ετικέτα “layerVector”. Σε αυτή την ενότητα όμως θα παρουσιάσουμε την πρώτη. Οι άλλες 2 θα αναλυθούν σε επόμενες ενότητες διότι απαιτούν και την χρήση δύο ακόμα ετικετών (“protocolHTTP” και “protocolWFS”) για τις οποίες θα μιλήσουμε αργότερα.

Δημιουργία Vector Layer από managed bean

Η ετικέτα “layerVector κάνει, εκτός των άλλων, χρήση των συναρτήσεων OpenLayers.Layer.Vector και OpenLayers.Feature.Vector, παραμετροποιώντας τες κατάλληλα ως προς το όνομα του στρώματος και τον πηγή (εδώ: managed bean) από όπου θα αντλήσουμε τα διανυσματικά δεδομένα. Επίσης καθορίζουμε, με την τιμή του attribute “editingToolbar”, αν στο στρώμα αυτό θα είναι διαθέσιμη στο χρήστη η δυνατότητα δημιουργίας νέων δεδομένων, σχεδιάζοντας τα με το ποντίκι πάνω στο στρώμα.

Περιορισμοί

- Η ετικέτα “layerVector” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “map”.
- Σε μια ετικέτα “map” μπορούν να υπάρχουν πάνω από μία ετικέτες “layerVector” με διαφορετικό/μοναδικό attribute “jsVar” η κάθε μία.
- Για να γίνει χρήση της ετικέτας “layerVector” είναι απαρέτητο να έχει προηγουμένως εισαχθεί στο χάρτη μία τουλάχιστον από τις ετικέτες: “layerWMS”, “layerGoogle”, ”layerYahoo”.

Πληροφορίες

Όνομα Ετικέτας	**layerVector**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLLayerVector**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

Πίνακας 21: Πληροφορίες για το συστατικό "LayerVector"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
jsVar	String	Ότι ισχύει για τα ονόματα μεταβλητών της javascript.	Το όνομα της javascript μεταβλητής που θα δημιουργηθεί για το στρώμα.
name	String	οποιαδήποτε	Το όνομα του στρώματος που θα δημιουργηθεί στο χάρτη.
editingToolbar	Boolean	Επιτρεπτές τιμές: true ή false	Η τιμή “true” θέτει το layer editable και εμφανίζει το κατάλληλο βοηθητικό panel.
value	MethodExpression	π.χ.: #{someBean.someProperty}	Το “someBean.someProperty” είναι τύπου Java List από Java κλάσεις. Οι κλάσεις αυτές περιέχουν ένα αντικείμενο τύπου Geometry* και δύο άλλα τύπου String,Float,Int κλπ.

Πίνακας 22: Χαρακτηριστικά της ετικέτας “layerVector”.

- Τα αντικείμενα τύπου Geometry μπορεί να είναι: σημεία, γραμμές ή πολύγωνα και δημιουργούνται με τη βοήθεια της βιβλιοθήκης JTS που αναφέραμε στο Κεφάλαιο 2 (Ενότητα 2.3).

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” zoom=”2”
           height=”512px” projection=”900913”>
      <jv:layerGoogle    jsVar=”gmap1” name=”google hybrid” type=”G_HYBRID_MAP”/>

      <jv:layerVector   jsVar="vec" name="Vector Layer" editingToolbar="true"
                        value="#{countryLayer.countries}">
      </jv:layerVector>
</jv:map>
```

Πίνακας 23: Χρήση της ετικέτας "layerVector"

Επεξήγηση

Στον πίνακα 23, παρουσιάζεται η ετικέτα “layerVector” παραμετροποιημένη ώστε να δημιουργεί ένα στρώμα με όνομα “Vector Layer” και να είναι editable. Εισάγεται στον χάρτη μας, που προηγουμένως με την ετικέτα “map” δημιουργήθηκε και έχει ως βασικό στρώμα ένα στρώμα Google. Προσέξτε ότι και εδώ έχουμε θέσει την τιμή του attribute “value” ίσο με "#{countryLayer.countries}", πράγμα που σημαίνει ότι το JSF θα ψάξει για ένα managed bean με όνομα “countryLayer”, με ένα property “countries”, που θα πρέπει να πληρεί τις κατάλληλες προδιαγραφές που αναλύθηκαν παραπάνω, και θα εισάγει/εμφανίσει την διανυσματική πληροφορία στο χάρτη “map1”. Στην Εικόνα 22, παρατηρούμε ότι το “countries” αποτελείται από:

(a) ένα πολύγωνο που περικλείει το Ηνωμένο Βασίλειο,

(b) μία γραμμή στην περιοχή της Γαλλίας,

(c) ένα σημείο κάπου στην Ελλάδα

(d) και ένα σημείο κάπου στην Αργεντινή.

Αποτέλεσμα

Εικόνα 22: Παράδειγμα χρήσης της ετικέτας "layerVector".

3.3.7 StyleMap

Το συστατικό “StyleMap” χρησιμοποιείται με την ετικέτα “styleMap” μέσα σε ένα αρχείο xhtml για να δημιουργηθεί ένα νέα (διαφορετικά από τα προκαθορισμένα) στυλ (style) εμφάνισης των features πάνω σε ένα vector layer.

Τα στυλ αυτά περιγράφονται από κάποιο αρχείο sld, το οποίο είναι ουσιαστικά ένα αρχείο xml, που τίθεται ως τιμή στο attribute “sld” της ετικέτας “styleMap”. Το αρχείο sld μπορεί να περιλαμβάνει πολλά styles με διαφορετικά ονόματα (namedLayer). Το ίδιο “namedLayer” επίσης μπορεί να περιγράφει διαφορετικά τα features που είναι επιλεγμένα ή όχι. Γι' αυτό, η ετικέτα “styleMap” περιλαμβάνει attributes όπως το “namedLayer”, “default” και “select”.

Η ετικέτα “styleMap” είναι σχεδιασμένη για να χρησιμοποιείται ως άμεση ετικέτα-παιδί της ετικέτας “layerVector”, ώστε να περιγράφει το στυλ των features της ετικέτας-γονέα της και μόνο.

Περιορισμοί

- Η ετικέτα “styleMap” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “layerVector”.
- Μία ετικέτα “layerVector” μπορεί να έχει το πολύ μία ετικέτα-παιδί “styleMap”.
- Μία ετικέτα “map” μπορεί να έχει το πολύ τόσες ετικέτες “styleMap” όσες και “layerVector”.

Πληροφορίες

Όνομα Ετικέτας	**styleMap**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLStyleMap**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

Πίνακας 24: Πληροφορίες για το συστατικό "StyleMap"

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
sld	String	π.χ.: somePath/something/someSLD.xml	Το PATH για κάποιο αρχείο sld.
namedLayer	String	Ότι ισχύει για τα ονόματα ετικετών ενός xml αρχείου.	Το όνομα ενός “namedLayer” του αρχείου sld.
default	Integer	Επιτρεπτές τιμές: “0” έως {πλήθος στυλ του αρχείου sld} - 1	Ο αριθμός (ανάλογα με τη θέση του) ενός “userStyle” του “namedLayer” του αρχείου sld. Περιγράφει τα features όταν αυτά δεν είναι επιλεγμένα.
select	Integer	Επιτρεπτές τιμές: “0” έως {πλήθος στυλ του αρχείου sld} - 1	Ο αριθμός (ανάλογα με τη θέση του) ενός “userStyle” του “namedLayer” του αρχείου sld. Περιγράφει τα features όταν αυτά είναι επιλεγμένα*.

Πίνακας 25: Χαρακτηριστικά της ετικέτας “styleMap”.

* Το πως επιλέγουμε κάποιο ή κάποια feature θα εξηγήσουμε σε επόμενη ενότητα, που θα αναλύσουμε την ετικέτα “selectFeature”.

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” zoom=”2”
           height=”512px” projection=”900913”>
      <jv:layerGoogle    jsVar=”gmap1” name=”google hybrid” type=”G_HYBRID_MAP”/>

      <jv:layerVector   jsVar="vec" name="Vector Layer" editingToolbar="true"
                        value="#{countryLayer.countries}">
            <jv:styleMap   sld="resources/mySLD.xml" namedLayer="MyNamedLayer"
                           default="0"  select="1">
            </jv:styleMap>
      </jv:layerVector>
</jv:map>
```

Πίνακας 26: Χρήση της ετικέτας "styleMap"

Επεξήγηση

Στον πίνακα 26, παρουσιάζεται η ετικέτα “styleMap” ως ετικέτα-παιδί μιας ετικέτας “layerVector, ώστε να δώσει νέα στυλ στα features του γονέα της. Η ετικέτα “styleMap” είναι παραμετροποιημένη κατάλληλα ώστε να διαβάζεται το αρχείο “mySLD.xml”, από το PATH “resources/mySLD.xml”. Το αρχείο αυτό θα πρέπει να περιέχει ένα tag με όνομα “MyNamedLayer” με τουλάχιστον δύο tags-παιδιά “UserStyle”. Αυτά ορίζονται να δώσουν στυλ στα features του Vector-layer-γονέα. Προσοχή: στο παράδειγμα αυτό δεν διακρίνουμε τα στυλ για τα επιλεγμένα features διότι δεν έχουν δώσει ακόμα στο χάρτη μας τη δυνατότητα επιλογής feature.

Στην Εικόνα 23, παρατηρούμε τα features της Εικόνας 22, με τα νέα στυλ εμφάνισης που ορίσαμε.

Αποτέλεσμα

Εικόνα 23: Παράδειγμα χρήσης της ετικέτας "styleMap".

3.3.8 ProtocolHTTP

Όπως αναφέραμε στην Ενότητα 3.3.6, η πηγή των διανυσματικών δεδομένων ενός Vector-layer μπορεί, εκτός του managed-bean, μπορεί να είναι και ένα αρχείο xml (σε gml format) που βρίσκεται οπουδήποτε στο Web. Το αρχείο αυτό μπορούμε να λάβουμε με το πρωτόκολλο HTTP χάρη στο συστατικό "ProtocolHTTP", η χρήση του οποίου γίνεται με τη βοήθεια της ετικέτας "protocolHTTP".

Η ετικέτα "protocolHTTP" είναι σχεδιασμένη για να χρησιμοποιείται ως άμεση ετικέτα-παιδί της ετικέτας "layerVector", με ένα και μοναδικό attribute "url", που δέχεται σαν τιμή την διεύθυνση που βρίσκεται το αρχείο xml που επιθυμούμε.

Περιορισμοί

- Η ετικέτα "protocolHTTP" σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας "layerVector".
- Μία ετικέτα "layerVector" μπορεί να έχει το πολύ μία ετικέτα-παιδί "protocolHTTP".
- Μία ετικέτα "protocolHTTP" αναφέρεται και επηρεάζει τα δεδομένα της ετικέτας-γονέας της και μόνο.
- Για τα αρχεία gml απαιτείται δήλωση του attribute κατάλληλου "projection", της έμμεσης ετικέτας-γονέα "map".

Πληροφορίες

Όνομα Ετικέτας	**protocolHTTP**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLProtocolHTTP**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

*Πίνακας 27: Πληροφορίες για το συστατικό "**ProtocolHTTP**"*

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
url	String	π.χ.: somePath/something/someXML.gml	Το PATH για κάποιο αρχείο gml.

*Πίνακας 28: Χαρακτηριστικά της ετικέτας "**protocolHTTP**".*

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” zoom=”1”
           height=”512px” projection=”900913”>
      <jv:layerGoogle    jsVar=”gmap1” name=”google hybrid” type=”G_HYBRID_MAP”/>

      <jv:layerVector   jsVar="vec" name="Vector Layer" editingToolbar="false">
                ...“ strategy ??? ”...
                <jv:protocolHTTP   url="somePath/someXML.gml"/>
      </jv:layerVector>
</jv:map>
```

*Πίνακας 29: Χρήση της ετικέτας **"protocolHTTP"***

Επεξήγηση

Στον πίνακα 29, παρουσιάζεται:

- Με πράσινο φόντο, η ετικέτα “protocolHTTP”, παραμετροποιημένη ώστε να βρίσκει το αρχείο “someXML.gml” και να εισάγει τα δεδομένα του στο “Vector layer” και κατ' επέκταση στο χάρτη, ο οποίος και θα τα παρουσιάσει.
- Με πορτοκαλί φόντο, 3 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Και με κόκκινο φόντο, μαρκάραμε την περιοχή που θα χρειαστεί η δήλωση μίας ετικέτας που θα δούμε σε επόμενη ενότητα, αλλά είναι απαραίτητη ώστε να ολοκληρώσουμε το παράδειγμα που αναδεικνύει τη λειτουργία της ετικέτας “protocolHTTP”.

Αποτέλεσμα

Το αποτέλεσμα θα παρουσιαστεί στην ενότητα 3.3.10, δηλαδή όταν θα έχει ολοκληρωθεί το παραπάνω παράδειγμα του Πίνακα 29.

3.3.9 ProtocolWFS

Όπως αναφέραμε σε προηγούμενες Ενότητες, η πηγή των διανυσματικών δεδομένων ενός Vector-layer μπορεί, εκτός του managed-bean και αρχεία gml, μπορεί να είναι και ένα Web Feature Service WFS. Τη χρήση κάποιου WFS αναλαμβάνει το συστατικό “ProtocolWFS”, το οποίο χρησιμοποιείται με τη βοήθεια της ετικέτας “protocolWFS”.

Η ετικέτα “protocolWFS” είναι σχεδιασμένη για να χρησιμοποιείται ως άμεση ετικέτα-παιδί της ετικέτας “layerVector”, με attributes το url του service, το τοπικό όνομα του feature που επιθυμούμε και το feature namespace.

Περιορισμοί

- Η ετικέτα “protocolHTTP” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “layerVector”.
- Μία ετικέτα “layerVector” μπορεί να έχει το πολύ μία ετικέτα-παιδί “protocolWFS”.
- Μία ετικέτα “protocolWFS” αναφέρεται και επηρεάζει τα δεδομένα της ετικέτας-γονέας της και μόνο.

Πληροφορίες

Όνομα Ετικέτας	**protocolWFS**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLProtocolWFS**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

*Πίνακας 30: Πληροφορίες για το συστατικό "**ProtocolWFS**"*

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
url	String	π.χ.: “somePath/something/wfs”	Το url για κάποιο WFS.
featureType	String	οτιδήποτε	Το όνομα ενός feature.
featureNS	String	π.χ.: “somePath/something/someName”	Το namespace κάποιου συνόλου από features.

*Πίνακας 31: Χαρακτηριστικά της ετικέτας “**protocolWFS**”.*

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” zoom=”1”
           height=”256px” projection=”4326”>
      <jv:layerWMS    jsVar="basicLayer"    name="OpenLayers WMS"
                      url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                      layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"    name=”Vector Layer” editingToolbar=”false”>
                ... “ strategy ??? ”  ...
                <jv:protocolWFS   url="http://demo.opengeo.org/geoserver/wfs"
                                  featureType="states"
                                  featureNS="http://www.openplans.org/topp">
                </jv:protocolWFS>
      </jv:layerVector>
</jv:map>
```

*Πίνακας 32: Χρήση της ετικέτας **"protocolWFS"***

Επεξήγηση

Στον πίνακα 32, παρουσιάζεται:

- Με πράσινο φόντο, η ετικέτα “protocolWFS”, παραμετροποιημένη ώστε μέσω του WFS από την διεύθυνση "http://demo.opengeo.org/geoserver/wfs", να διαβάζει το feature με όνομα “states”, με namespace "http://www.openplans.org/topp" και να το εισάγει στο “Vector layer” και κατ' επέκταση στο χάρτη, ο οποίος και θα τα παρουσιάσει.
- Με πορτοκαλί φόντο, 3 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Και με κόκκινο φόντο, μαρκάραμε την περιοχή που θα χρειαστεί η δήλωση μίας ετικέτας που θα δούμε σε επόμενη ενότητα, αλλά είναι απαραίτητη ώστε να ολοκληρώσουμε το παράδειγμα που αναδεικνύει τη λειτουργία της ετικέτας “protocolWFS”.

Αποτέλεσμα

Το αποτέλεσμα θα παρουσιαστεί στην ενότητα 3.3.11, δηλαδή όταν θα έχει ολοκληρωθεί το παραπάνω παράδειγμα του Πίνακα 32.

3.3.10 StrategyFixed

Όπως αναφέραμε στην Ενότητα 3.3.8, η πηγή των διανυσματικών δεδομένων ενός Vector-layer μπορεί μπορεί να είναι και ένα αρχείο xml (σε gml format) που βρίσκεται οπουδήποτε στο Web. Το αρχείο αυτό μπορούμε να λάβουμε με το πρωτόκολλο HTTP χάρη στο συστατικό “ProtocolHTTP”, η χρήση του οποίου γίνεται με τη βοήθεια της ετικέτας “protocolHTTP”. Το συστατικό αυτό όμως πρέπει να συνδυαστεί με “κάτι” ακόμα που θα ορίσει τον τρόπο που αιτούμαστε features. Τον τρόπο αυτό καλούμε “στρατηγική” και γίνεται μέσω κάποιων components που θα αναλύσουμε αυτήν την ενότητα αλλά και στις επόμενες τρεις.

Σε αυτή την ενότητα θα αναλύσουμε την στρατηγική που ορίζει ότι, για ένα vector layer, αιτούμαστε features μία φορά και ποτέ ξανά. Αυτό γίνεται με τη βοήθεια του συστατικού “StrategyFixed”, που χρησιμοποιείται με την ετικέτα “strategyFixed”. Η ετικέτα “strategyFixed” είναι σχεδιασμένη για να χρησιμοποιείται ως άμεση ετικέτα-παιδί της ετικέτας “layerVector”.

Περιορισμοί

- Η ετικέτα “strategyFixed” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “layerVector”.
- Μία ετικέτα “layerVector” μπορεί να έχει το πολύ μία ετικέτα-παιδί “strategyFixed”.
- Μία ετικέτα “strategyFixed” αναφέρεται και επηρεάζει τα δεδομένα της ετικέτας-γονέας της και μόνο.
- Μία ετικέτα “strategyFixed” απαιτεί την ύπαρξη ακριβώς μίας ετικέτας που ορίζει το πρωτόκολλο αίτησης διανυσματικών δεδομένων (protocolHTTP ή protocolWFS), ως ετικέτα-παιδί του ίδιου vector layer.

Πληροφορίες

Όνομα Ετικέτας	**strategyFixed**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLStrategyFixed**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

*Πίνακας 33: Πληροφορίες για το συστατικό **"StrategyFixed"***

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
preload	Boolean	Επιτρεπτές τιμές: “true” ή “false”	Με την τιμή “true” προγραμματίζουμε την φόρτωση (load) των δεδομένων πριν γίνει ορατό το στρώμα.

*Πίνακας 34: Χαρακτηριστικά της ετικέτας “**strategyFixed**”.*

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” zoom=”1”
           height=”512px” projection=”900913”>
      <jv:layerWMS   jsVar="basicLayer"   name="OpenLayers WMS"
                     url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                     layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"   name=”Vector Layer” editingToolbar=”false”>
            <jv:strategyFixed   preload=”true”/>
            <jv:protocolWFS   url="http://demo.opengeo.org/geoserver/wfs"
                              featureType="states"
                              featureNS="http://www.openplans.org/topp">
            </jv:protocolWFS>
      </jv:layerVector>
</jv:map>
```

*Πίνακας 35: Χρήση της ετικέτας "**strategyFixed**".*

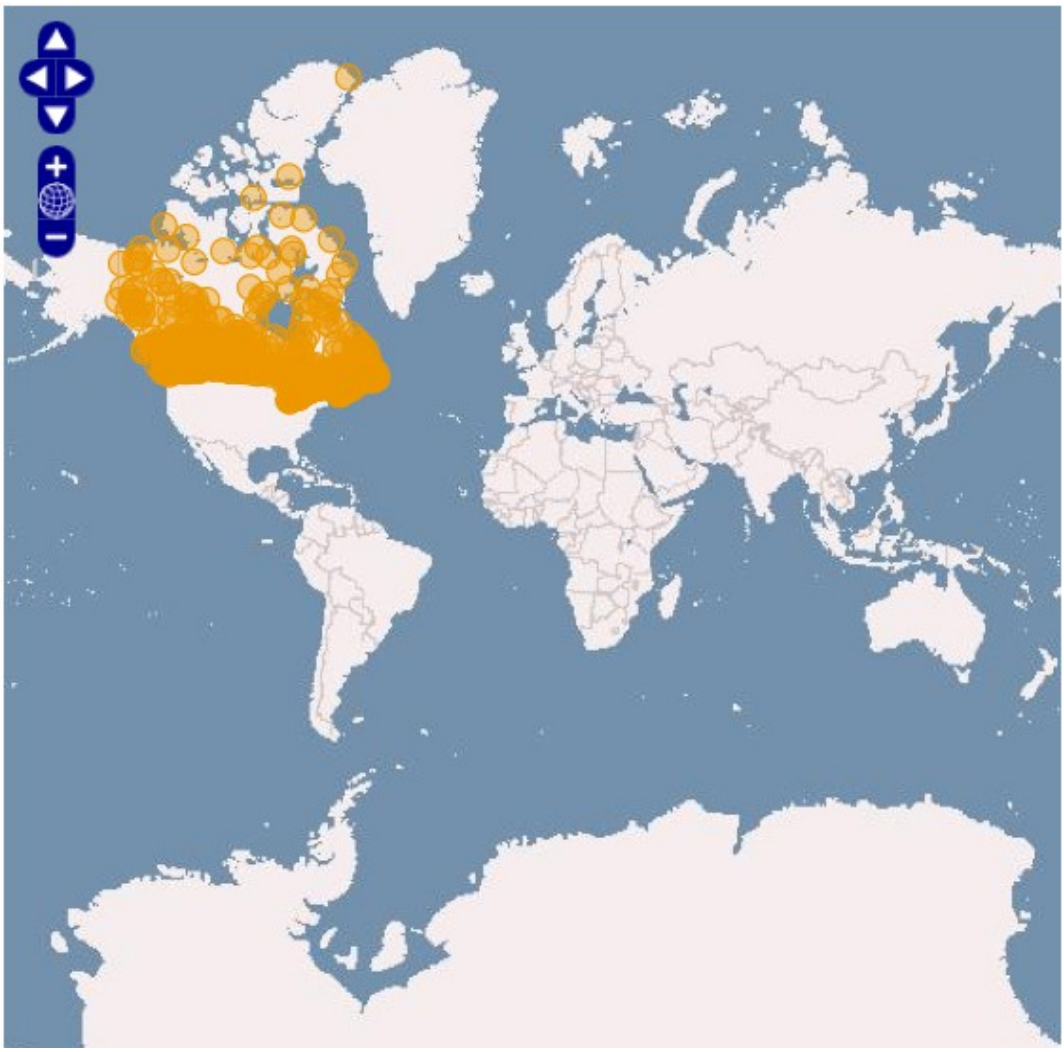
Επεξήγηση

Στον Πίνακα 35, παρουσιάζεται:

- Με πορτοκαλί φόντο, 3 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Με πράσινο φόντο, η ετικέτα “protocolHTTP”, παραμετροποιημένη ώστε να βρίσκει το αρχείο “someXML.gml” και να εισάγει τα δεδομένα του στο “Vector layer” και κατ' επέκταση στο χάρτη, ο οποίος και θα τα παρουσιάσει. (Όπως ακριβώς είδαμε και στο παράδειγμα του Πίνακα 29).
- Και με κόκκινο φόντο, την ετικέτα “strategyFixed”, με preload ίσο με “true” ώστε να φορτώνει τα δεδομένα που ζητήσαμε με την “protocolHTTP” πριν γίνει ορατό το vector layer και όπως ορίζει η συγκεκριμένη στρατηγική, δεν μπορούμε να κάνουμε άλλη αίτηση για νέα δεδομένα σε αυτό το στρώμα.

Αποτέλεσμα

Το παράδειγμα της ενότητας αυτής συμπληρώνει το παράδειγμα της ενότητας 3.3.8, επομένως μπορούμε να παρουσιάσουμε το αποτέλεσμα της χρήσης των σε συνεργασία ετικετών: “protocolHTTP” και “strategyFixed” στην Εικόνα 24.



Εικόνα 24: Παράδειγμα χρήσης των ετικετών "protocolHTTP" και "strategyFixed".

3.3.11 StrategyBBox

Στην προηγούμενη ενότητα είδαμε πως να ορίζουμε τη στρατηγική “strategyFixed”, κατά την οποία φορτώνουμε όλα τα διανυσματικά δεδομένα που ζητήσαμε, μία φορά και δεν μπορούμε να κάνουμε αίτηση για νέα δεδομένα.

Σε αυτή την ενότητα θα αναλύσουμε την στρατηγική που ορίζει ότι, για ένα vector layer, αιτούμαστε features που περικλείονται σε ένα bounded box δηλαδή στο ορθογώνιο τμήμα του χάρτη που βλέπουμε μία δεδομένη στιγμή, ενώ μετακινώντας (κάνοντας drag) το χάρτη σε άλλο σημείο γίνεται νέα αίτηση για τα διανυσματικά δεδομένα της νέας περιοχής. Αυτό γίνεται με τη βοήθεια του συστατικού “StrategyBBox”, που χρησιμοποιείται με την ετικέτα “strategyBBox”. Η ετικέτα “strategyBBox” είναι σχεδιασμένη για να χρησιμοποιείται ως άμεση ετικέτα-παιδί της ετικέτας “layerVector” και να συνδυάζεται με την ετικέτα “protocolWFS”.

Περιορισμοί

- Η ετικέτα “strategyBBox” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “layerVector”.
- Μία ετικέτα “layerVector” μπορεί να έχει το πολύ μία ετικέτα-παιδί “strategyBBox”.
- Μία ετικέτα “strategyBBox” αναφέρεται και επηρεάζει τα δεδομένα της ετικέτας-γονέας της και μόνο.
- Μία ετικέτα “strategyBBox” απαιτεί την ύπαρξη ακριβώς μίας ετικέτας που ορίζει το πρωτόκολλο αίτησης διανυσματικών δεδομένων (protocolWFS ή protocolHTTP), ως ετικέτα-παιδί του ίδιου vector layer.

Πληροφορίες

Όνομα Ετικέτας	**strategyBBox**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLStrategyBBox**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

*Πίνακας 36: Πληροφορίες για το συστατικό "**StrategyBBox**"*

Χαρακτηριστικά

Η ετικέτα “strategyBBox” δεν έχει κανένα attribute. (δεν παραμετροποιείται)

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px” zoom=”3” center=”-70.927,53.014”
           height=”256px” projection=”4326”>
      <jv:layerWMS    jsVar="basicLayer"   name="OpenLayers WMS"
                      url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                      layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"   name=”Vector Layer” editingToolbar=”false”>
                <jv:strategyBBox/>
                <jv:protocolWFS   url="http://demo.opengeo.org/geoserver/wfs"
                                  featureType="states"
                                  featureNS="http://www.openplans.org/topp">
                </jv:protocolWFS>
      </jv:layerVector>
</jv:map>
```

*Πίνακας 37: Χρήση της ετικέτας **"strategyBBox"***

Επεξήγηση

Στον Πίνακα 36, παρουσιάζεται:

- Με πορτοκαλί φόντο, 3 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Με πράσινο φόντο, η ετικέτα “protocolWFS”, όπως ακριβώς είδαμε και στο παράδειγμα του Πίνακα 32).
- Και με κόκκινο φόντο, την ετικέτα “strategyBBox” που λόγω του zoom=”3” στην ετικέτα “map”, τα δεδομένα που θα λάβουμε με τη χρήση της ετικέτας “protocolWFS” θα φορτώνονται στον χάρτη μόνο όταν και αν αυτά βρεθούν στο ορθογώνιο ορατό σε μας μέρος του χάρτη που βλέπουμε. Κάνοντας drag τον χάρτη φορτώνονται και εμφανίζονται και άλλα features, αν υπάρχουν.

Αποτέλεσμα

Το παράδειγμα της ενότητας αυτής συμπληρώνει το παράδειγμα της ενότητας 3.3.9, όμως δεν θα γίνει κατανοητή η λειτουργία του με κάποιο screenshot του χάρτη διότι δεν διαφαίνεται το dragging στο χάρτη. Γι' αυτό η χρήση της ετικέτας: "strategyBBox" θα φανεί στα online παραδείγματα στην διεύθυνση: http://vincubator.softnet.tuc.gr.

3.3.12 StrategyCluster

Στην προηγούμενες ενότητες είδαμε πως να ορίζουμε τη στρατηγική "strategyFixed", κατά την οποία φορτώνουμε εξαρχής όλα τα διανυσματικά δεδομένα που ζητήσαμε αλλά και τη στρατηγική "strategyBBox", κατά την οποία φορτώνονται τα δεδομένα του τρέχοντος κάθε στιγμή ορθογώνιου πλαισίου που είναι ορατό.

Σε αυτή την ενότητα θα αναλύσουμε την στρατηγική που ορίζει ότι, για ένα vector layer, θα φορτώνονται τα διανυσματικά δεδομένα του κάνοντας ταυτόχρονα ομαδοποίηση (clustering). Αυτό επιτυγχάνεται με το συστατικό "StrategyCluster", οποίο χρησιμοποιείται με την ετικέτα "strategyCluster", παραμετροποιώντας την κατάλληλα ο προς τον τρόπο ομαδοποίησης, χάρη στα attributes "distance" και "threshold", που υποστηρίζει.

Περιορισμοί

- Η ετικέτα "strategyCluster" σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας "layerVector".
- Η ετικέτα "strategyCluster" μπορεί και πρέπει να συνδυάσει τη λειτουργία της με μία εκ των δύο ετικετών: "strategyFixed" ή "strategyBBox", που αναλύθηκαν σε παραπάνω ενότητες.
- Μία ετικέτα "layerVector" μπορεί να έχει το πολύ μία ετικέτα-παιδί "strategyCluster".
- Μία ετικέτα "layerVector" μπορεί να έχει, εκτός από μία ετικέτα-παιδί "strategyCluster", και μία ετικέτα-παιδί "strategyFixed" ή "strategyBBox".
- Μία ετικέτα "strategyCluster" αναφέρεται και επηρεάζει τα δεδομένα της ετικέτας-γονέας της και μόνο.
- Μία ετικέτα "strategyCluster" απαιτεί την ύπαρξη ακριβώς μίας ετικέτας που ορίζει το πρωτόκολλο αίτησης διανυσματικών δεδομένων (protocolWFS ή protocolHTTP), ως ετικέτα-παιδί του ίδιου vector layer.

Πληροφορίες

Όνομα Ετικέτας	**strategyCluster**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLStrategyCluster**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

*Πίνακας 38: Πληροφορίες για το συστατικό "**StrategyCluster**"*

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
distance	Integer	Επιτρεπτές τιμές: θετικοί ακέραιοι	Η ακτίνα σε pixels που ορίζει το νοητό κύκλο, μέσα στον οποίο τα features, θεωρούνται ένα cluster.
threshold	Integer	Επιτρεπτές τιμές: 1 ή μεγαλύτερο	Ο μικρότερος αριθμός από features που μπορούν να σχηματίζουν ένα cluster.

*Πίνακας 39: Χαρακτηριστικά της ετικέτας "**strategyCluster**".*

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px”  zoom=”1”
           height=”512px” projection=”900913”>
      <jv:layerWMS    jsVar="basicLayer"   name="OpenLayers WMS"
                      url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                      layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"   name=”Vector Layer” editingToolbar=”false”>
                <jv:strategyFixed/>
                <jv:strategyCluster distance="20" threshold="1"/>
                <jv:protocolHTTP   url="somePath/someXML.gml"/>
      </jv:layerVector>
</jv:map>
```

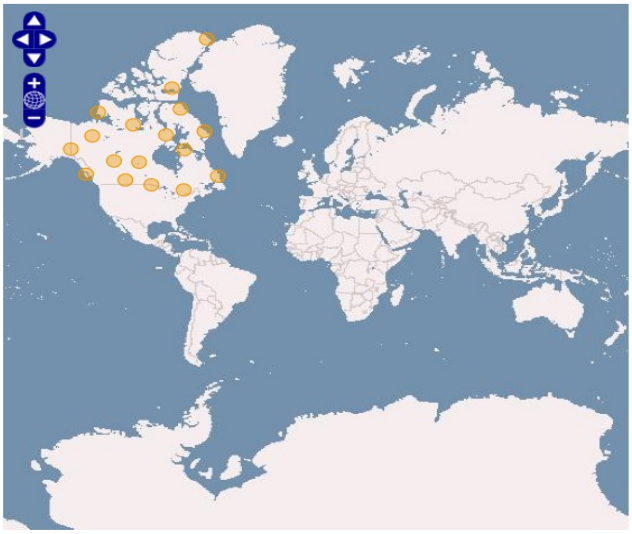
*Πίνακας 40: Χρήση της ετικέτας "**strategyBBox**"*

Επεξήγηση

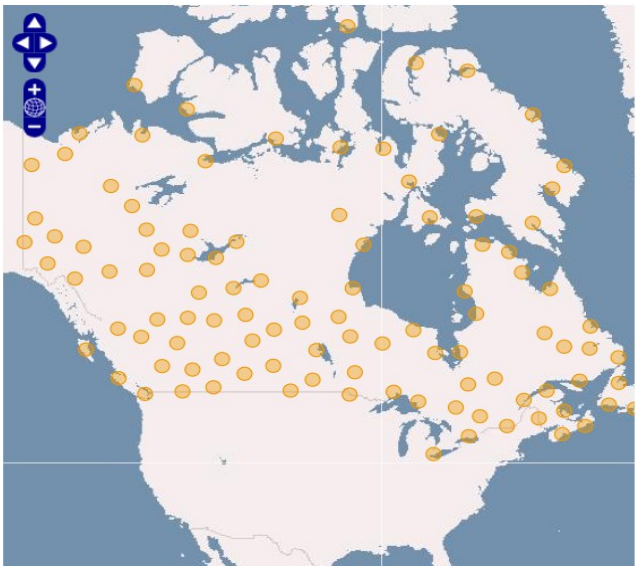
Στον Πίνακα 40, παρουσιάζεται:

- Με πορτοκαλί φόντο, 4 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Με πράσινο φόντο, η ετικέτα “strategyFixed”, που έχουμε αναλύσει στην Ενότητα 3.3.10.
- Και με κόκκινο φόντο, την ετικέτα “strategyCluster” που λόγω του distance=”20” pixels και threshold=”1”, σχηματίζει clusters με τα διανυσματικά δεδομένα που ζήτησε η ετικέτα “protocolHTTP”. Κάθε cluster επιτρέπεται να έχουν το λιγότερο ένα feature. Από κάθε feature, σε ακτίνα 20 pixels σχηματίζεται ένα cluster. Προσέξτε ότι κάθε φορά που κάνουμε zoom στο χάρτη, το πλήθος των clusters αλλάζει διότι μεταβάλλεται και η απόσταση τους (η απόσταση σε pixels που βλέπουμε, όχι η πραγματική απόστασή τους).

Αποτέλεσμα



*Εικόνα 25: Παράδειγμα χρήσης της ετικέτας "strategyCluster" (αρχικά με **zoom="1"**).*



*Εικόνα 26: Παράδειγμα χρήσης της ετικέτας "strategyCluster" με **zoom="3"**.*

3.3.13 StrategyPaging

Στην προηγούμενες τρεις ενότητες είδαμε πως να ορίζουμε τις στρατηγικές “strategyFixed”, κατά την οποία φορτώνουμε εξαρχής όλα τα διανυσματικά δεδομένα που ζητήσαμε, τη στρατηγική “strategyBBox”, κατά την οποία φορτώνονται τα δεδομένα του τρέχοντος κάθε στιγμή ορθογώνιου πλαισίου που είναι ορατό, αλλά και τη στρατηγική “strategyCluster”, κατά την οποία ομαδοποιούμε τα διανυσματικά δεδομένα που εμφανίζονται στο vector layer του χάρτη μας.

Σε αυτή την ενότητα θα αναλύσουμε την στρατηγική που ορίζει ότι, για ένα vector layer, θα φορτώνονται τα διανυσματικά δεδομένα του κάνοντας ταυτόχρονα σελιδοποίηση (paging). Αυτό επιτυγχάνεται με το συστατικό “StrategyPaging”, οποίο χρησιμοποιείται με την ετικέτα “strategyPaging”, η οποία δημιουργεί ταυτόχρονα και το ένα προκαθορισμένο panel εναλλαγής σελίδων (από features).

Περιορισμοί

- Η ετικέτα “StrategyPaging” σχεδιάστηκε για να είναι άμεση ετικέτα-παιδί της ετικέτας “layerVector”.
- Η ετικέτα “StrategyPaging” μπορεί και πρέπει να συνδυάσει τη λειτουργία της με την ετικέτα “strategyFixed”, που αναλύθηκε σε παραπάνω ενότητα.
- Μία ετικέτα “layerVector” μπορεί να έχει το πολύ μία ετικέτα-παιδί “StrategyPaging”.
- Μία ετικέτα “layerVector” μπορεί να έχει, εκτός από μία ετικέτα-παιδί “StrategyPaging”, και μία ετικέτα-παιδί “strategyFixed”.
- Μία ετικέτα “StrategyPaging” αναφέρεται και επηρεάζει τα δεδομένα της ετικέτας-γονέας της και μόνο.
- Μία ετικέτα “StrategyPaging” απαιτεί την ύπαρξη ακριβώς μίας ετικέτας που ορίζει το πρωτόκολλο αίτησης διανυσματικών δεδομένων (protocolWFS ή protocolHTTP), ως ετικέτα-παιδί του ίδιου vector layer.
- Μία ετικέτα “StrategyPaging” ομαδοποιεί τα διανυσματικά δεδομένα σε σελίδες των 10 features το πολύ, η κάθε μία.

Πληροφορίες

Όνομα Ετικέτας	**strategyPaging**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLStrategyPaging**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIInput**

*Πίνακας 41: Πληροφορίες για το συστατικό "**StrategyPaging**"*

Χαρακτηριστικά

Η ετικέτα "StrategyPaging" δεν έχει κανένα attribute. (δεν παραμετροποιείται)

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px”  zoom=”1”
           height=”512px” projection=”900913”>
      <jv:layerWMS    jsVar="basicLayer"   name="OpenLayers WMS"
                      url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                      layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"   name=”Vector Layer” editingToolbar=”false”>
                <jv:strategyFixed/>
                <jv:strategyCluster/>
                <jv:protocolHTTP   url="somePath/someXML.gml"/>
      </jv:layerVector>
</jv:map>
```

*Πίνακας 42: Χρήση της ετικέτας "**strategyPaging**"*

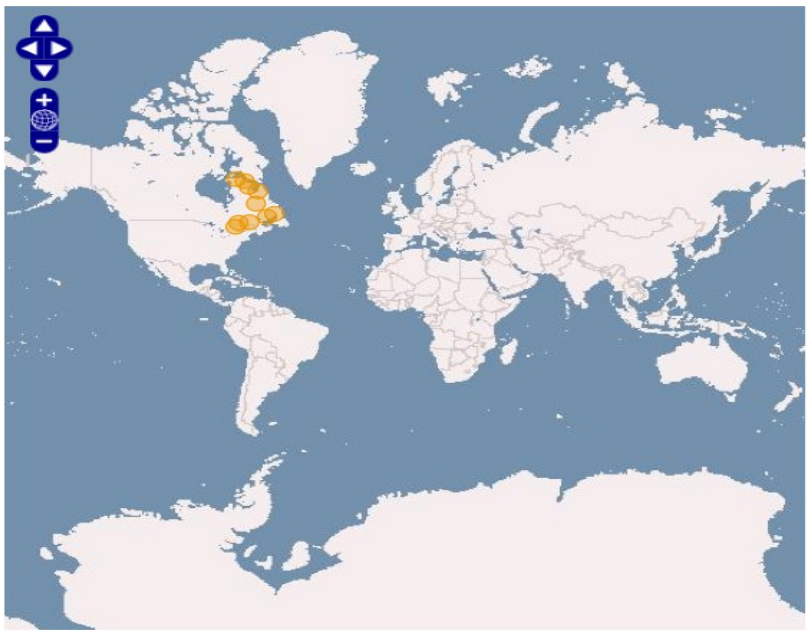
Επεξήγηση

Στον Πίνακα 42, παρουσιάζεται:

- Με πορτοκαλί φόντο, 4 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Με πράσινο φόντο, η ετικέτα "strategyFixed", που έχουμε αναλύσει στην Ενότητα 3.3.10.
- Και με κόκκινο φόντο, την ετικέτα "strategyPaging",που ομαδοποιεί τα διανυσματικά δεδομένα που ζήτησε η ετικέτα "protocolHTTP" σε σελίδες των 10 fearures η κάθε μία.

Αποτέλεσμα

Εικόνα 27: Παράδειγμα χρήσης της ετικέτας "strategyPaging". (αρχικά εμφανίζεται η σελίδα 1). Προσοχή: όλα τα feature είναι εξαρχής φορτωμένα στο χάρτη, λόγω της ετικέτας "strategyFixed".



Εικόνα 28: Μετά το πάτημα του button "next" εμφανίζεται η σελίδα με τα δέκα χαρακτηριστικά, νούμερο 2. (Χωρίς ανανέωση της web page).

3.3.14 FeatureSelect

Στην ενότητα αυτή θα δούμε πως μπορούμε να παρέχουμε, σε ένα χάρτη με τουλάχιστον ένα vector-layer, τη δυνατότητα να μπορεί ο χρήστης να επιλέξει features από αυτόν, να στείλει δεδομένα σχετικά με το feature που επέλεξε στον Server για επεξεργασία και επίσης να αξιοποιήσει τα επεξεργασμένα δεδομένα για να ανανεώσει μερικώς το περιεχόμενο της σελίδας ασύγχρονα, χωρίς μετάβαση σε νέα σελίδα ή ανανέωση ολόκληρης της τρέχουσας σελίδας.

Την παραπάνω λειτουργία αναλαμβάνει να διεκπεραιώσει το συστατικό “FeatureSelect”, που χρησιμοποιείται με βοήθεια της ετικέτας “featureSelect”. Αυτή παραμετροποιείται, με κατάλληλα attributes, ως προς:

i. το vector-layer από όπου θα μπορεί ο χρήστης να επιλέξει features του,
ii. την αποστολή πληροφορίας, σχετικής με το επιλεγμένο/α feature/s,
iii. και με την μερική ανανέωση της τρέχουσας σελίδας, σύμφωνα με τα επεξεργασμένα δεδομένα.

Περιορισμοί

- Η ετικέτα “featureSelect” σχεδιάστηκε έτσι ώστε να πρέπει να εισάγεται σε ετικέτα “h:form”, η οποία με τη σειρά της να είναι ετικέτα-παιδί της ετικέτας “map”.
- Η ετικέτα “featureSelect” μπορεί και πρέπει να συνδυάσει τη λειτουργία της με ακριβώς μία ετικέτα “layerVector”, οι οποίες να βρίσκεται στην ίδια ετικέτα “map”.
- Μία ετικέτα “featureSelect” συνεργάζεται με μία μόνο ετικέτα “layerVector” στην οποία αναφέρεται, με το attribute της “forVector”.
- Μία ετικέτα “featureSelect” μπορεί να ανανεώσει μερικώς την τρέχουσα σελίδα αλλά μόνο ότι περικλείεται ετικέτα-γονέα της “h:form”.
- Η ετικέτα “featureSelect” μπορεί και πρέπει να συνεργαστεί με μία ετικέτα “h:inputText”, οι οποίες να έχουν την ίδια ετικέτα-γονέα “h:form”.

Πληροφορίες

Όνομα Ετικέτας	**featureSelect**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLFeatureSelect**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIComponentBase**

*Πίνακας 43: Πληροφορίες για το συστατικό "**FeatureSelect**"*

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
forVector	String	Επιτρεπτές τιμές: ότι ισχύει για τα ονόματα των javascript μεταβλητών	Δίνουμε την τιμή του attribute “jsVar” της ετικέτας “layerVector” στην οποία αναφέρεται η συγκεκριμένη ετικέτα “featureSelect”.
execute	String	οποιοδήποτε String	Δίνουμε την τιμή του attribute “id” της ετικέτας “h:inputText” με την οποία συνεργάζεται η συγκεκριμένη ετικέτα “featureSelect”.
render	String ή Annotation	Επιτρεπτές τιμές: οποιαδήποτε String ή @this ή @form	Δίνουμε την τιμή του attribute “id” της ετικέτας “h:outputText” που θέλουμε να ανανεώσουμε ή annotation για να ανανεώσουμε όλα τα περιεχόμενα της ετικέτας-γονέα “h:form”.
dataToSend	String	Επιτρεπτές τιμές: ένα String (χωρίς “_”) ή Strings ενωμένα με underscores(“_”)	Τα ονόματα των properties των επιλεγμένων features που θέλουμε να στείλουμε στον Server για επεξεργασία.

Πίνακας 44: Χαρακτηριστικά της ετικέτας "featureSelect".

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px”  zoom=”1”
           height=”256px” projection=”4326”>
      <jv:layerWMS    jsVar="basicLayer"   name="OpenLayers WMS"
                      url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                      layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"  name="vec"  editingToolbar="false"
                      value="#{someBean.aProperty}">
      </jv:layerVector>
      <h:form>
        You have selected:   <h:inputText    id="in"    value="#{country.selected}"/>
        <jv:featureSelect     forVector="vec"    execute="in"
                              render="out"   dataToSend="name_geometry">
        </jv:featureSelect>
        <h:outputText    id="out" value="#{country.helloString}"  style=" color: red"/>
      <h:form>
</jv:map>
```

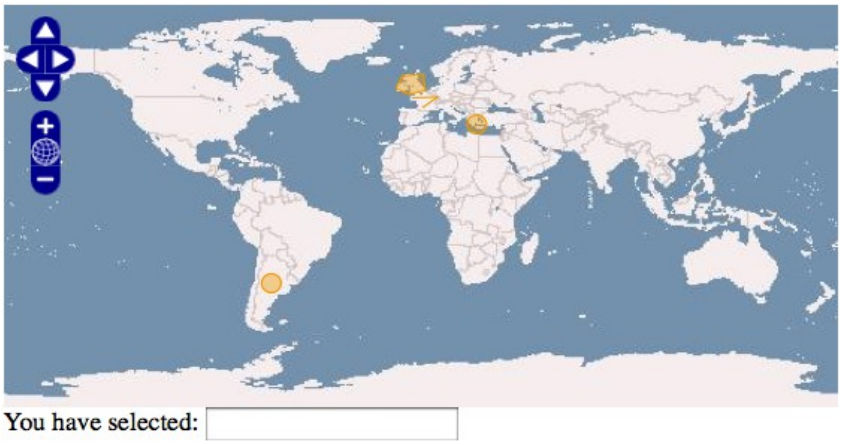
*Πίνακας 45: Χρήση της ετικέτας **"featureSelect"***

Επεξήγηση

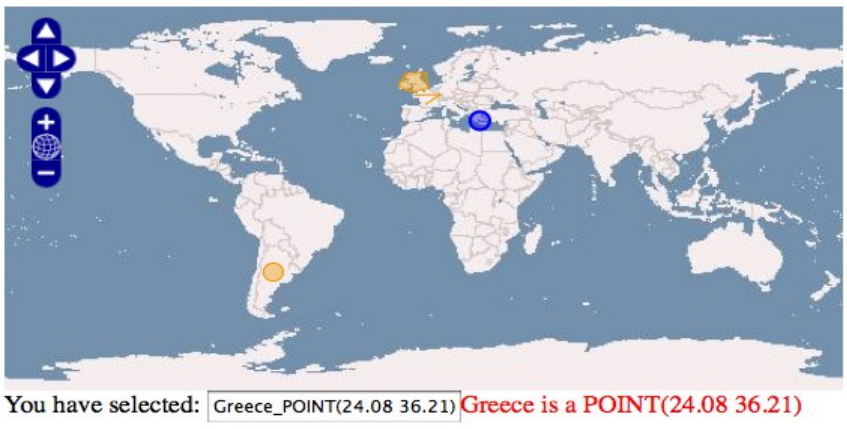
Στον Πίνακα 45, παρουσιάζεται:

- Με πορτοκαλί φόντο, 3 ετικέτες που έχουν ήδη αναλύσει σε προηγούμενες ενότητες.
- Και με κόκκινο φόντο, την ετικέτα “featureSelect”, που δίνει στο χρήστη τη δυνατότητα να επιλέγει τα διανυσματικά δεδομένα που θέλει από το vector layer ”vec” και να στείλει τα properties “name” και “geometry”.
- Με πράσινο φόντο, 2 βοηθητικές standard ετικέτες του JSF:
  - Η ετικέτα “h:inputText” δείχνει τα επιλεγμένα προς αποστολή δεδομένα και τα αποθηκεύει σε Java Bean.
  - Η ετικέτα “h:outputText” εμφανίζει ένα μήνυμα που φορτώνεται/ανανεώνεται ανάλογα με τα δεδομένα που επιλέγουμε.

Αποτέλεσμα



Εικόνα 29: Πριν επιλέξουμε κάποιο feature.



Εικόνα 30: Αφού επιλέξαμε την Ελλάδα.

3.3.15 Popup

Στην ενότητα αυτή θα δούμε πως μπορούμε να παρέχουμε, σε ένα χάρτη με τουλάχιστον ένα vector-layer, τη δυνατότητα να μπορεί ο χρήστης επιλέγοντας features από αυτόν, να δημιουργείται ένα νέο μικρό div με τη μορφή ενός παραθύρου/φούσκας, που ονομάζουμε “popup”.

Την παραπάνω λειτουργία αναλαμβάνει να διεκπεραιώσει το συστατικό “Popup”, που χρησιμοποιείται με βοήθεια της ετικέτας “popup”. Αυτή παραμετροποιείται ως προς περιεχόμενό που θα παρουσιάσει αυτό το div-φούσκα.

Περιορισμοί

- Η ετικέτα “popup” σχεδιάστηκε έτσι ώστε να πρέπει να εισάγεται σε ετικέτα “featureSelect”, η οποία να είναι ετικέτα-παιδί μιας ετικέτας “h:form”, με τη σειρά της να είναι ετικέτα-παιδί της ετικέτας “layerVector”.
- Η ετικέτα “popup” μπορεί και πρέπει να συνδυάσει τη λειτουργία της με ακριβώς μία ετικέτα “featureSelect”, οι οποίες να βρίσκεται στην ίδια ετικέτα “layerVector”.
- Μία ετικέτα “popup” συνεργάζεται με μία μόνο ετικέτα “featureSelect”, της οποίας είναι άμεση ετικέτα-παιδί.
- Η ετικέτα “popup” δημιουργεί ή καταστρέφει το μικρό div της ασύγχρονα, χωρίς ανανέωση της web page.
- Η τιμή του attribute “contentHTML” της ετικέτας “popup”, είναι κώδικας HTML. (οι ειδικοί χαρακτήρες πρέπει να κωδικοποιούνται. Π.χ.: Το σύμβολο “<” γράφεται “<”)

Πληροφορίες

Όνομα Ετικέτας	**popup**
Κλάση Συστατικού	**gr.tuc.softnet.jsfgis.OLPopup**
Renderer	**Το ίδιο το συστατικό λειτουργεί ως renderer του εαυτού του.**
Οικογένεια συστατικών	**UIComponentBase**

*Πίνακας 46: Πληροφορίες για το συστατικό **"Popup"***

Χαρακτηριστικά

Όνομα	Τύπος	Τιμή	Περιγραφή
jsVar	String	Επιτρεπτές τιμές: ότι ισχύει για τα ονόματα των javascript μεταβλητών	Η όνομα της javascript μεταβλητής που είναι αντικείμενο τύπου: OpenLayers.Popup
contentHTML	String	οποιοδήποτε String	Το HTML περιεχόμενο του μικρού div-φούσκας.

*Πίνακας 47: Χαρακτηριστικά της ετικέτας **"popup"**.*

Χρήση

```
<jv:map    jsVar=”map1” div=”map1” width=”512px”  zoom=”1”
           height=”256px” projection=”4326”>
      <jv:layerWMS   jsVar="basicLayer"   name="OpenLayers WMS"
                     url="http://vmap0.tiles.osgeo.org/wms/vmap0?"
                     layers="basic"  isBaseLayer="true">
      <jv:layerVector  jsVar="vec"  name="vec"  editingToolbar="false"
                     value="#{someBean.aProperty}">
      </jv:layerVector>
      <h:form>
         You have selected:  <h:inputText   id="in"    value="#{country.selected}"/>
         <jv:featureSelect    forVector="vec"    execute="in"
                              render="out"   dataToSend="name_geometry">
            <jv:popup   jsVar="popup1"
                  contentHTML="&lt;h3&gt;Hello from  &quot;
                                          +feature.attributes.name+&quot;&lt;/h3&gt;">
            </jv:popup>
         </jv:featureSelect>
         <h:outputText   id="out" value="#{country.helloString}"  style=" color: red"/>
      </h:form>
</jv:map>
```

*Πίνακας 48: Χρήση της ετικέτας **"popup"***

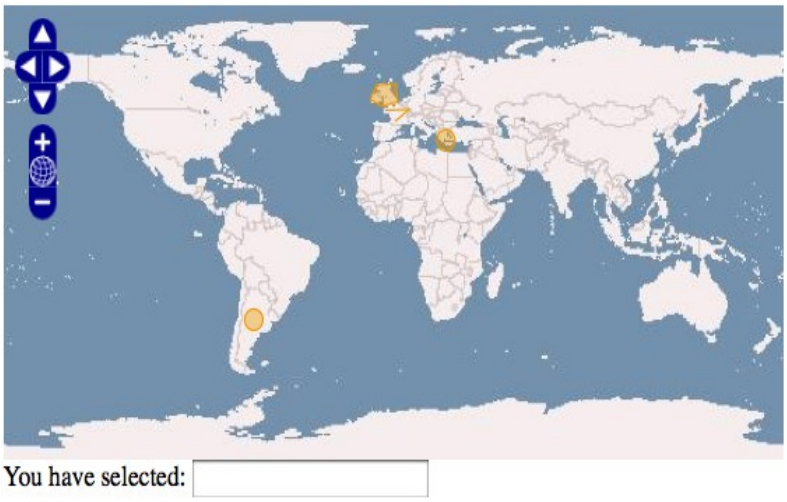
Επεξήγηση

Στον Πίνακα 48, παρουσιάζεται:

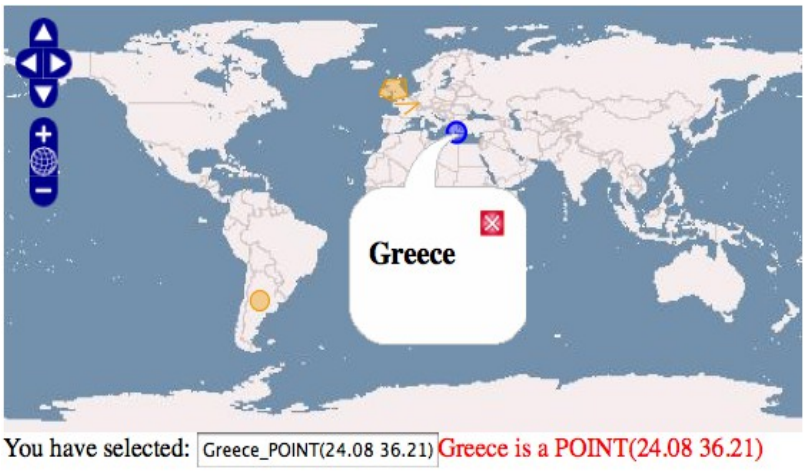
- Με πορτοκαλί φόντο, 7 ετικέτες που έχουν ήδη αναλύσει στον Πίνακα 45.
- Και με πράσινο φόντο, την ετικέτα “popup”, που εισάγει στον χάρτη (και στην web page) ένα μικρό div-φούσκα με περιεχόμενο το κείμενο “hello from <όνομα επιλεγμένης χώρας>”.

Αποτέλεσμα

Παράδειγμα χρήσης της ετικέτας “popup”:



Εικόνα 31: Πριν επιλέξουμε κάποιο feature.



Εικόνα 32: Αφού επιλέξαμε την Ελλάδα.

3.4 Παραδείγματα

Στις προηγούμενες Ενότητες αναλύσαμε τη σουίτα JVFACES και τα συστατικά από τα οποία αποτελείται. Για να δείτε περισσότερα παραδείγματα απλών εφαρμογών, μαζί με τον πηγαίο τους κώδικα JSF, επισκεφθείτε το **Online Demo** στη διεύθυνση: http://vincubator.softnet.tuc.gr.

4 Υλοποίηση του JVFACES

4.1 Γενικά

Έχοντας περιγράψει την σχεδίαση της σουίτας JVFACES στο Κεφάλαιο 3, είδαμε ότι αυτή αποτελείται από 15 προσαρμοσμένα συστατικά για το πλαίσιο JavaServer Faces (JSF), τα οποία υπακούν πλήρως τις προδιαγραφές του Java Enterprise Edition (J2EE) και πιο ειδικά στο JSF 2.0 specification. Για την κατασκευή τους χρησιμοποιήθηκαν: α) σε μεγάλο βαθμό η javascript βιβλιοθήκη OpenLayers που εκμεταλλεύεται την τεχνολογία AJAX (Asynchronous Javascript and XML), καθώς επίσης β) η σουίτα Java Topology Suite (JTS) που αφορούν την δημιουργία/ διαχείριση γεωγραφικών δεδομένων.

4.2 Προσαρμοσμένα συστατικά σε JSF 2.0

Το πλαίσιο JSF 2.0 πάνω στο οποίο εργαστήκαμε προσφέρει δύο μηχανισμούς ανάπτυξης προσαρμοσμένων συστατικών: α) τα composite components και τα β) τα custom components, τους οποίους θα αναλύσουμε στις παρακάτω ενότητες.

Composite components

Το πλαίσιο JSF στη δεύτερη έκδοσή του παρουσιάζει ένα νέο μηχανισμό με τον οποίο μπορεί ο προγραμματιστής να συνδυάσει άλλα, ήδη υπάρχοντα, standard components του JSF ώστε να δημιουργήσει δικά του, σύνθετα components.

Σε ένα νέο αρχείο *xhtml*, αφού δηλώσουμε την βιβλιοθήκη ετικετών που ορίζει το JSF για τα composite components, η **δομή** ενός composite component η οποία ακολουθεί είναι πολύ απλή και αποτελείται από δύο τμήματα:

1) το *interface* (διεπαφή) που είναι ουσιαστικά ο ορισμός της ετικέτας και των attributes του νέου component.
2) Και το *implementation* (υλοποίηση), που δεν είναι άλλο από τα standard JSF components που θέλουμε να συνδυάσουμε, όπως components για input/output, buttons, forms, και πολλά άλλα components που παράγουν markup, ακόμα και javascript.

Composite Component

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:composite="http://java.sun.com/jsf/composite">
  <composite:interface>
    <composite:attribute name="image"/>
    <composite:attribute name="actionMethod"
      method-signature="java.lang.String action()" />
  </composite:interface>

  <composite:implementation>
    <h:form>
      <h:commandLink action="#{cc.attrs.actionMethod}">
      <h:graphicImage url="#{cc.attrs.image}" styleClass="icon" />
      </h:commandLink>
    </h:form>
  </composite:implementation>
</html>
```

Αφού δημιουργήσουμε το νέο component, μπορούμε να το χρησιμοποιήσουμε μέσα σε άλλα αρχεία xhtml που δημιουργούμε δηλώνοντας αρχικά το prefix του και στη συνέχεια σε όποιο σημείο του κώδικά μας καλώντας το με την ετικέτα του που δημιουργήσαμε με το interface.

Το αποτέλεσμα που είναι διπλό: 1) ένας κώδικας μικρότερος, που παρουσιάζει το τι κάνει με πιο δηλωτικό τρόπο που δεν περιπλέκεται από τις λεπτομέρειες υλοποίησης και 2) ένα νέο σύνθετο - επαναχρησιμοποιούμενο συστατικό.

Μειονέκτημα - Συμπέρασμα

Έχοντας αναλύσει τα composite components, συμπεραίνουμε ότι μας επιτρέπουν να συνθέσουμε τα σχετικά απλά components που θέλουμε, δηλαδή αυτά αποτελούνται αποκλειστικά από standard components του JSF, αλλά δυστυχώς αδυνατούν να υποστηρίξουν καινούριους τύπους δεδομένων που θα δημιουργήσουμε (π.χ. Κλάσεις για γεωγραφικά δεδομένα) ή δεν δίνουν στον προγραμματιστή απόλυτη ελευθερία να υλοποιήσει ένα component με εξ' ολοκλήρου νέα συμπεριφορά.

Στην εργασία αυτή, χρειάζεται να υλοποιηθούν και ορισμένα components όπως π.χ. το layerVector που θα διαβάζει γεωγραφικά δεδομένα από κλάσεις ειδικά υλοποιημένες γι αυτά, ώστε να δημιουργεί vector layers για του χάρτες πράγμα που απαιτεί ειδικά υλοποιημένη συμπεριφορά του component αυτού (δηλαδή που δεν γίνεται από κάποιο από τα standard components του JSF). Επομένως θα πρέπει να “κατέβουμε” σε επίπεδο υλοποίησης του component σε γλώσσα **Java**. Σε αυτό το εγχείρημα θα βοηθήσει ο άλλος μηχανισμός υλοποίησης components του JSF, τα custom components που θα αναλύσουμε στις επόμενες ενότητες.

Custom Components

Με το μηχανισμό custom components, το πλαίσιο JSF μας προσφέρει τη δυνατότητα να υλοποιήσουμε σε Java, σύνθετα components, που δεν περιορίζονται μόνα στα standard components του JSF, αλλά με συμπεριφορά που θα ορίσουμε εμείς στο 100%. Αυτό σημαίνει ότι εξαρτάται από εμάς να ακολουθήσουμε όλες τις απαραίτητες διαδικασίες που πρέπει να πληρεί ένα custom component σύμφωνα με τις προδιαγραφές του JSF.

Το μηχανισμό αυτό ακολουθήσαμε για την υλοποίηση των components του JVFACES. Στις παρακάτω ενότητες θα τον παρουσιάσουμε σε βήματα και θα αναλύσουμε λεπτομέρειες υλοποίησης.

4.3 Υλοποίηση κλάσεων των συστατικών σε JAVA

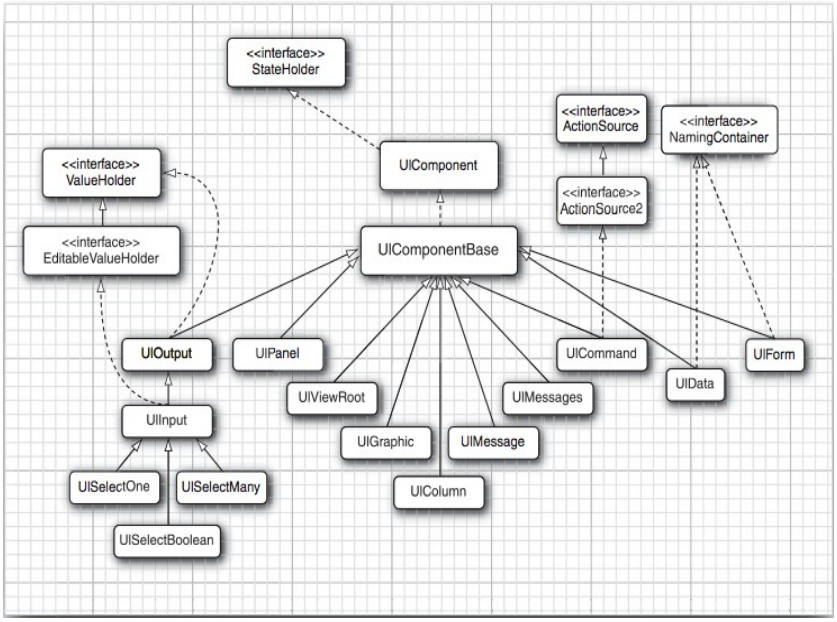
Ως πρώτο βήμα για την δημιουργία ενός νέου custom components σε JSF είναι η υλοποίηση της Java κλάσης του component. Μια τέτοια κλάση θα πρέπει να πληρεί απαραιτήτως τρεις προϋποθέσεις:

1. Να διατηρεί πάντα την κατάστασή του (δηλαδή είναι υπεύθυνο για τις τρέχουσες κάθε φορά τιμές των διαφόρων πεδίων που αυτό ορίζει).
2. Να κωδικοποιεί τη διεπαφή χρήστη, παράγοντας την απαραίτητη γλώσσα markup (HTML).
3. Να αποκωδικοποιεί τα αιτήματα HTTP που προκαλεί ο χρήστης (π.χ. κλικ σε κουμπιά).

Στην περίπτωση των components του JVFACES όμως, θα δούμε επίσης components με δύο ακόμα δυνατότητες καθώς τα αντικείμενα τύπου χάρτη το απαιτούν:

1. Να κωδικοποιούν javascript για δημιουργία του δυναμικού περιεχομένου στην διεπαφή χρήστη.
2. Να αποκωδικοποιούν αιτήματα AJAX.

Οι κλάσεις των custom components του JSF, σύμφωνα με το προδιαγραφές του, θα πρέπει να επεκτείνουν κάποια από από τις abstract κλάσεις που ορίζει το JSF για αυτό το σκοπό. Η ιεραρχία των abstract κλάσεων φαίνεται στην παρακάτω Εικόνα:



Η επιλογή της κατάλληλης κλάσεις γίνεται ανάλογα με τον είδος ή την λειτουργία του custom component που θέλουμε κάθε φορά να υλοποιήσουμε. Ο ρόλος τους είναι καθαρά βοηθητικός (καθώς περιέχουν κατάλληλες προϋλοποιημένες βοηθητικές μεθόδους) γι' αυτό και θα δούμε ότι για την υλοποίηση των components του JVFACES χρησιμοποιήθηκε κυρίως η UIInput και η γενική superclass αυτής, UIComponentBase.

Παραδείγματα

```
public class OLMap extends UIInput {}
public class OLLayerWMS extends UIInput {}
public class OLControl extends UIInput {}
...
        :
public class OLPopup extends UIComponentBase {
public class OLFeatureSelect extends UIComponentBase {}
```

4.4 Κωδικοποίηση: παράγοντας γλώσσα σήμανσης (markup)

Τα components του JVFACES από μόνα τους να παράγουν HTML δηλαδή χωρίς εξωτερικό renderer. Αυτό γίνεται με τρεις συγκεκριμένες μεθόδους, όπως ορίζει το JSF specification:

- με την **EncodeBegin()**
- με την **EncodeChildren()**
- και με την **EncodeEnd()**

Από τα χαρακτηριστικά ονόματά τους αντιλαμβανόμαστε ότι η μόνη διαφορά των τριών παραπάνω μεθόδων είναι ότι το πλαίσιο JSF καλεί στο τέλος του life cycle πρώτα την μέθοδο encodeBegin, στη συνέχεια την encodeChildren και τελευταία την μέθοδο encodeEnd. Επίσης να σημειώσουμε ότι η μέθοδος encodeChildren καλείται εφόσον το εκάστοτε component έχει component(s)-παιδιά. Διαφορετικά δεν υπάρχει λόγος να χρησιμοποιήσουμε και τις δύο μεθόδους (encodeBegin, encodeEnd) αλλά μία από τις δύο χωρίς καμία διαφορά.

Παράδειγμα

Για παράδειγμα, το component OLMap, θέλουμε να δημιουργεί, μεταξύ των άλλων, ένα div στο οποίο προορίζουμε να εμφανίσουμε κάποιο χάρτη.

Επομένως θέλουμε να παράξουμε ένα κώδικα HTML σαν αυτόν:

```
<div id="map1" class="smallmap olMap" style="width:512px;height:256px;"/>
```

Για να το πετύχουμε λοιπόν γράφουμε στη μέθοδο encodeBegin του συγκεκριμένου component τα εξής:

```
public void encodeBegin(FacesContext context) throws IOException {
        ResponseWriter writer = context.getResponseWriter();
                ...
        writer.startElement("div",this);
        writer.writeAttribute("style","width:512px;height:256px;", null);
        writer.writeAttribute("class","smallmap olMap",null);
        writer.writeAttribute("id",div,null);
        writer.endElement("div");
                ...
}
```

Στο παραπάνω τμήμα κώδικα διακρίνουμε την κλάση ResponseWriter η οποία μας βοηθά να παράξουμε markup (εδώ HTML tags) χάρη στις μεθόδους τις. Με τη μέθοδο startElement() και endElement() δημιουργήσαμε την αρχή και το τέλος της ετικέτας που επιθυμούσαμε (εδώ div), ενώ

με τη μέθοδο writeAttribute δημιουργήσαμε τα τρία ζεύγη attribute-τιμής που θέλαμε.

Αντίστοιχα, στο component OLStrategyPaging, θέλουμε μεταξύ άλλων να δημιουργήσουμε ένα στοιχείο τύπου button, για εναλλαγή σε άλλο page. Επομένως θέλουμε το συγκεκριμένο component να παράξει κώδικα HTML σαν αυτόν:

```
<input type=”button” id=”...” value=”...” onclick=”...” />
```

Για να το πετύχουμε λοιπόν γράφουμε στη μέθοδο encodeBegin του συγκεκριμένου component τα εξής:

```
writer.startElement("input", this);
writer.writeAttribute("type", "button", null);
writer.writeAttribute("id",this.getParent().getId()+"_next", null);
writer.writeAttribute("value", "next", null);
writer.writeAttribute("onclick", this.getParent().getId()+"_strategies2.pageNext();", null);
writer.endElement("input");
```

Κατ' αυτόν λοιπόν τον τρόπο εργαστήκαμε σε όλα τα components του JVFACES ώστε να παράγει καθένα την επιθυμητή HTML.

4.5 Περιγραφή βιβλιοθήκης ετικετών

Εκτός από την υλοποίηση της κλάσης για τα components, θα πρέπει να παρέχουμε και ένα αρχείο περιγραφής (file descriptor) που θα καθορίζει το τρόπο που θα χρησιμοποιείται η ετικέτα του custom component μέσα σε στις σελίδες JSF.

Ειδικότερα, το *αρχείο περιγραφής* (file descriptor) ορίζει τα εξής:

- Ένα πεδίο ονομάτων (namespace). [π.χ. ***http://gr.tuc.softnet.jsfgis***]
- Ένα όνομα [π.χ. ***map***, ***layerWMS***] και τον τύπο του component (component type) [π.χ. ***gr.tuc.softnet.jsfgis.OLMap***] για την κάθε ετικέτα (tag).

Παράδειγμα

```
<facelet-taglib...>
<namespace> http://gr.tuc.softnet.jsfgis </namespace>
<tag>
    <tag-name> map </tag-name>
    <component>
      <component-type> gr.tuc.softnet.jsfgis.OLMap </component-type>
    </component>
</tag>
</facelet-taglib>
```

Table 49: Μέρος του αρχείου: jsfgis.taglib.xml

Ο τύπος ενός component είναι ένα αναγνωριστικό στοιχείο της κλάσης του component αυτού, επομένως θα πρέπει να αντιστοιχίζεται με τη κλάση αυτή καθεαυτή. Κάτι τέτοιο επιτυγχάνεται θέτοντας ένα annotation στην συγκεκριμένη κλάση:

Παράδειγμα

```
@FacesComponent("gr.tuc.softnet.jsfgis.OLMap")
public class OLMap extends UIInput
```

4.6 Επεξεργασία των χαρακτηριστικών των ετικετών

Όταν θέλουμε να χρησιμοποιήσουμε ένα οποιοδήποτε component σε μια σελίδα JSF χρειαζόμαστε μόνο την ετικέτα (tag) του. Στον Πίνακα 49, ορίσαμε ότι για το component “OLMap” η ετικέτα του θα λέγεται <map>, η οποία με πρόθεμα (prefix) “jv” για το **JV**FACES γίνεται: <jv:map>.

Μια ετικέτα όμως μπορεί να έχει και attribute έτσι ώστε να “περνά” ορίσματα στο component κάνοντας το παραμετροποιήσημο. Έτσι και στο παράδειγμα μας, θέλοντας να δώσουμε χαρακτηριστικά στον χάρτη που θα δημιουργήσει το component “OLMap”, δίνουμε στην ετικέτα του attributes, όπως αυτά περιγράφηκαν στην Ενότητα 3.3.1.

Παράδειγμα

```
<jv:map jsVar="map1" div="map1" width="512px" height="256px" projection="4326">
...
</jv:map>
```

Επεξήγηση

Πλέον είναι εμφανές ότι η ετικέτα “map” θα καλέσει το component να δημιουργήσει ένα div με τιμή “map1”, για ένα χάρτη δεδομένου ύψους / πλάτους και τύπου προβολής, με όνομα “map1”. Ο τρόπος που θα το κάνει αυτό όμως δεν απασχολεί τον developer της σελίδας καθώς ο κώδικας είναι υλοποιημένος στον component OLMap το οποίο καλεί.

Από τη στιγμή που χρησιμοποιήθηκε η ετικέτα map, το JSF θα καλέσει το component OLMap, το οποίο ξεκινά με την μέθοδο encodeBegin. Η τελευταία αρχικά θα αποθηκεύσει σε Java μεταβλητές τις τιμές των attributes που δόθηκαν, εφόσον ελέγξει την καταλληλότητα τους (validation).

Στην περίπτωση του OLMap, τα attributes μπορούν να θεωρηθούν απλού τύπου (String, Integer, κλπ). Παρόλαυτά, ένα attribute μπορεί να δεχθεί ως τιμή ένα “Value Expression”.

Το Value Expression χρησιμοποιείται σε μια σελίδα JSF με την εξής σύνταξη:

#{someBean.someProperty}

και με αυτόν τον τρόπο “ζητάμε” από το JSF ουσιαστικά ένα αντικείμενο Java που είναι αποθηκευμένο σε ένα Java Managed Bean.

Στην περίπτωση του component “OLLayerVector”, όπως αυτό περιγράφηκε στην Ενότητα 3.3.6, ορίσαμε ένα attribute με όνομα “value” που δέχεται τιμές τύπου value expression.

Παράδειγμα

```
<jv:layerVector jsVar="vec" name="vec" editingToolbar="true" value="#{jvbean.countries}">
...
</jv:layerVector>
```

Επεξήγηση

Όπως βλέπουμε στον παραπάνω κώδικα, τιμή του attribute “value” είναι το property “countries” του managed-bean “jvbean”.

Το managed-bean “jvbean” είναι μια κλάση που περιέχει ένα Java “ArrayList” ονόματι “countries” από αντικείμενα τύπου “JvbeanBase” που υλοποιήσαμε εμείς.

Η κλάση “**JvbeanBase**” ως managed-bean

Η κλάση “JvbeanBase” είναι ουσιαστικά μια δομή που υλοποιήσαμε για να περιγράφει ένα αντικείμενο τύπου “χώρας”. Συνεπώς, περιέχει properties τύπου String ή Integer για τα θεματικά δεδομένα και ένα property τύπου “Geometry” (βλέπε **JTS**) για τα διανυσματικά δεδομένα.

Υλοποιεί σαφώς public μεθόδους **setters** και **getters** όπως απαιτείται στα managed-beans, καθώς επίσης και μια μέθοδο για το διανυσματικό γεωμετρικό υπολογισμό της τομής (βλέπε **Example 10** στο demo) και τέλος μία ακόμη μέθοδο για την ανανέωση του control "layerSwitcher" που ενεργοποιείται με "AjaxBehaviorEvent" (βλέπε **Example 11** στο demo).

4.7 Κωδικοποίηση javascript

Οι χάρτες σε μια σελίδα δεν είναι ποτέ στατικοί, όπως πχ μια εικόνα ή μια φωτογραφία, αλλά δυναμικοί. Επομένως η javascript είναι αναγκαία για τη δημιουργία τους. Προσοχή όμως! Σκοπός τις εργασίας μας είναι η δηλωτική σύνθεση διεπαφών για γεωγραφικά δεδομένα, άρα ο web developer δεν χρειάζεται να ασχοληθεί με λεπτομέρειες υλοποίησης χαρτών (όπως: javascript APIs) αλλά μόνο με τις ετικέτες και τα χαρακτηριστικά τους (όπως είδαμε στην προηγούμενη ενότητα).

Συνεπώς, τα components που υλοποιήσαμε θα παράγουν την απαραίτητη javascript, παραμετροποιώντας ανάλογα με τις τιμές των χαρακτηριστικών των ετικετών του JVFACES.

Το πλαίσιο JSF 2.0 μας δίνει αυτή τη δυνατότητα καθώς επιτρέπει στα custom components να κωδικοποιούν javascript (όπως ακριβώς δείξαμε σε προηγούμενη ενότητα με την κωδικοποίηση HTML).

Παράδειγμα Κώδικα

```
writer.startElement("script",this);
writer.writeAttribute("type", "text/javascript", null);

 if(type.equals("MousePosition")||type.equals("Panel")){
            writer.write("   document.getElementById(\""+type+"\").onchange();\n");
            writer.write("function toggleC(){\n");
            writer.write("   if(document.getElementById(\""+type+"\").value == 'true'){\n");
            writer.write("      c_"+type+".activate()\n");
            writer.write("}\n");
            writer.write("   else{\n");
            writer.write("      c_"+type+".deactivate()\n");
            writer.write("   }\n");
            writer.write("}\n");
}
writer.endElement("script");
```

Επεξήγηση

Στον παραπάνω Πίνακα βλέπουμε ένα τμήμα του κώδικα του component "OLControl". Ο κώδικας αυτός εμπεριέχεται στη μέθοδο encodeBegin του εν λόγω component και με τη χρήση των Java συναρτήσεων writer.startElement(), writer.writeAttribute() και writer.endElement() υλοποιείται η HTML

ετικέτα <script>, στο σώμα της οποίας γράφουμε κώδικα javascript, ο οποίος στην περίπτωσή μας παράγεται με τη βοήθεια της Java μεθόδου writer.write() που παίρνει τον κώδικα javascript ως όρισμα.

OpenLayers

Εκτός όμως από δικές μας απλές συναρτήσεις javascript, για την υλοποίηση του JVFACES χρησιμοποιήθηκε κατά κόρον η πανίσχυρη javascript βιβλιοθήκη OpenLayers (όπως αυτή περιγράφθηκε στην Ενότητα 2.2). Όλα τα components του JVFACES κάνουν χρήση μεθόδων της OpenLayers αλλά μία ήταν η κύρια μέθοδος, η οποία έδωσε και το όνομά της στο αντίστοιχο component.

Παράδειγμα Κώδικα

```
// load OpenLayers.js
writer.startElement("script",this);
writer.writeAttribute("type", "text/javascript", null);
writer.writeAttribute("src", "http://openlayers.org/api/OpenLayers.js", null);
writer.endElement("script");
```

Επεξήγηση

Στον παραπάνω Πίνακα βλέπουμε τμήμα του component “OLMap” χάρη στο οποίο παράγεται ετικέτα “<script>” στο σώμα της οποίας δηλώνουμε ότι θα χρησιμοποιήσουμε το αρχείο “OpenLayers.js”, που περιλαμβάνει όλες τις μεθόδους της συγκεκριμένης βιβλιοθήκης που σκοπεύουμε να χρησιμοποιήσουμε στο ίδιο το component “OLMap” αλλά και στα όλα τα υπόλοιπα.

Ο λόγος που επιλέγουμε το component “OLMap” για τη δήλωση του αρχείου OpenLayers.js είναι ότι το συγκεκριμένο component είναι το βασικό (root) component της σουίτας JVFACES. Με άλλα λόγια για να υλοποιηθεί οτιδήποτε με τη σουίτα JVFACES, θα πρέπει να υπάρχει το component “OLMap”, επομένως και όλα τα child-component της σουίτας δεν πρόκειται να αντιμετωπίσουν πρόβλημα μη εντοπισμού του αρχείου “OpenLayers.js”.

Πλέον το component “OLMap” μπορεί να κάνει χρήση της συνάρτησης “OpenLayers.Map()” και να δημιοργήσει ένα div όπου προορίζεται να απεικονίσει ένα χάρτη. Στη συνέχεια ένα child-component του “OLMap” όπως το “OLLayerWMS” θα μπορεί με χρήση της συνάρτησης “OpenLayers.WMS()” να δημιουργήσει ένα layer στον χάρτη αυτόν. Ομοίως, και τα υπόλοιπα components του JVFACES θα μπορούν να χρησιμοποιηθούν για την προσθήκη άλλων γεωγραφικών δεδομένων στον χάρτη αυτόν, παράγοντας τον απαραίτητο κώδικα javascript.

Τμήμα Κώδικα του “OLMap”

```
writer.startElement("script",this);
writer.writeAttribute("type", "text/javascript", null);
                                ...
if(projection.equals("900913")){
    writer.write("var maxExtent = new OpenLayers.Bounds(-20037508, -20037508, 20037508,
20037508),\n"
                        +"restrictedExtent = maxExtent.clone(),\n"
                        +"maxResolution = 156543.0339;\n"
                        +"\n"
                        +"var options = {\n"
                        +"projection: new OpenLayers.Projection(\"EPSG:900913\"),\n"
                        +"displayProjection: new OpenLayers.Projection(\"EPSG:4326\"),\n"
                        +"units: \"m\",\n"
                        +"numZoomLevels: 18,\n"
                        +"maxResolution: maxResolution,\n"
                        +"maxExtent: maxExtent,\n"
                        +"restrictedExtent: restrictedExtent\n"
                        +"};\n");

    writer.write("var " + jsVar.trim() + " = new OpenLayers.Map(\'" + div + "\', options);\n");
}
else if(projection.equals("4326"))
    writer.write("var " + jsVar.trim() + " = new OpenLayers.Map(\'" + div + "\');\n");
                                ...
writer.endElement("script");
```

Τμήμα κώδικα του “OLLayerWMS”

```
writer.startElement("script",this);
writer.writeAttribute("type", "text/javascript", null);
                ...
if(isBaseLayer.equalsIgnoreCase("true")){
    writer.write("var "+ jsVar +" = new OpenLayers.Layer.WMS(\'"+name+"\',\'"+url+"\',
     {layers: \'"+layers+"\'});\n");
}
else{
    writer.write("var "+jsVar+" = new OpenLayers.Layer.WMS(\'"+name+"\',\'"+url+"\',
     {layers: \'"+layers+"\',transparent: \'"+transparent+"\'},{isBaseLayer: "
     +isBaseLayer+ ",          visibility: " + visibility +"});\n");
}
writer.write(pjsVar + ".addLayers(["+jsVar+"]);\n");
                ...
writer.endElement("script");
```

4.8 Υλοποίηση AJAX συστατικών

Τα πλεονεκτήματα της τεχνολογίας AJAX είναι πολλά και γνωστά πλέον όπως αναφέραμε και στην Ενότητα 2.1.4. Γι' αυτό θα δείξουμε ότι τα components του JVFACES εκμεταλλεύονται αυτήν την τεχνολογία στο σύνολο τους.

Αρχικά να πούμε ότι οι συναρτήσεις της βιβλιοθήκης OpenLayers που χρησιμοποιήσαμε για την υλοποίηση των components μας είναι έτσι γραμμένες ώστε να όλα τα requests που προκαλούν να είναι AJAX calls.

Επίσης, το ίδιο το JSF 2.0 υποστηρίζει ήδη την ετικέτα “<f:ajax>”, η οποία συνεργάζεται άψογα με τα components του JVFACES, όπως συμβαίνει και στα “Example 10” και “Example 11” του demo που δημιουργήσαμε (http://vincubator.softnet.tuc.gr) για την επίδειξη των δυνατοτήτων του JVFACES.

Τέλος, τα ίδια τα custom components του JSF μπορούν να παράξουν κατάλληλη javascript που εκτελεί AJAX call. Για παράδειγμα, το component “OLFeatureSelect”, που είναι σχεδιασμένο όπως αναλύσαμε στην Ενότητα 3.3.14, παράγει javascript κώδικα που εκτελεί AJAX call ώστε να αποστείλει τα χαρακτηριστικά του επιλεγμένου feature ώστε αυτά να τεθούν σε πιθανή επιθυμητή επεξεργασία όπου το αποτέλεσμα αυτής να ανανεώσει ασύγχρονα τη σελίδα μας.

Τμήμα κώδικα του “OLFeatureSelect”

```
public void encodeEnd(FacesContext context) throws IOException {
        ResponseWriter writer = context.getResponseWriter();
        ...
        writer.startElement("script",this);
        writer.writeAttribute("type", "text/javascript", null);
        ...
        writer.write("function onFeatureSelect(feature) {\n");
        ...
        writer.write("   jsf.ajax.request(\""+formId+":"+execute+"\",null,   {execute:\""+formId+":"+execute+"\",
                render:\""+formId+":"+render+"\"})\n"
        ...
        writer.write("var selectControl = new OpenLayers.Control.SelectFeature("+forVector+",
        {onSelect:onFeatureSelect, onUnselect: onFeatureUnselect,
        multipleKey: \"shiftKey\" })\n"
            + pjsVar + ".addControl(selectControl);\n"
            + "selectControl.activate();\n");
        ...
        writer.endElement("script");
}
```

4.9 Συστατικά τύπου: γονέας-παιδί

Η σουίτα JVFACES έχει σχεδιαστεί για τη σύνθεση διεπαφών για γεωγραφικά δεδομένα για το πλαίσιο JSF 2.0. Το τελευταίο επιτρέπει σε ένα component να διαχειρίζεται και να αλληλεπιδρά με άλλα components έτσι ώστε να δημιουργούμε σύνθετες διεπαφές. Ο καλύτερος τρόπος για να επιτύχουμε κάτι τέτοιο είναι να συνδέσουμε ορισμένα components με τη σχέση γονέα-παιδιού ή αλλιώς με εμφωλευμένα components.

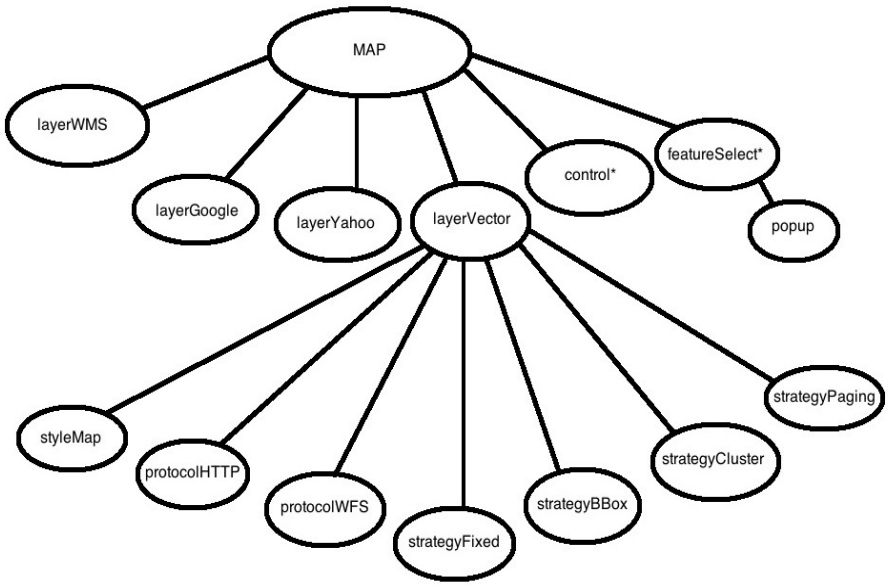
Για παράδειγμα, το σύνθετο JSF component "<h:dataTable>" χρησιμοποιείται πάντα σε συνδυασμό με εμφωλευμένο/α component/s "<h:column>" για την δημιουργία πινάκων από συλλογές (table of collections) κατ' αυτόν τον τρόπο:

```
<h:dataTable  var="someVar"  value="#{someCollection}" border="…">
    <h:column>  #{someVar.property1}  </h:column>
    <h:column>  #{someVar.property2}  </h:column>
    ...
</h:dataTable>
```

Από το παράδειγμα αντιλαμβανόμαστε ότι το component "h:dataTable" είναι το component-γονέας του "h:column". Αυτή είναι η λογική/σωστή σύνταξη για τη δημιουργία αυτού του σύνθετου component και επομένως η χρήση του <h:column> "έξω" από ένα <h:dataTable> απαγορεύεται.

Σύμφωνα με αυτή τη λογική κατασκευάστηκαν και τα components του JVFACES. Έτσι λοιπόν ορίζεται το component "OLMap" ως root/parent component όλων των υπολοίπων της σουίτας (που χωρίς αυτό δεν έχει νόημα η χρήση των υπολοίπων γι' αυτό και απαγορεύεται με τρόπο που θα εξηγήσουμε στη συνέχεια).

Η σχέση γονέα-παιδιού για κάθε component της σουίτας JVFACES έχει περιγραφεί στο Κεφάλαιο 3 για κάθε ένα component ξεχωριστά. Η παρακάτω Εικόνα δείχνει και σχηματικά τη συνολική σύνδεση τους.



Να σημειώσουμε ότι τα δύο components “control” και “featureSelect” είναι σημαδεμένα με αστερίσκο για να δείξουμε ότι είναι μεν component-παιδιά του “map” αλλά όχι άμεσα. Ως άμεσο γονέα-component είναι το component “h:form”, άμεσος γονέας του οποίου είναι το “map”.

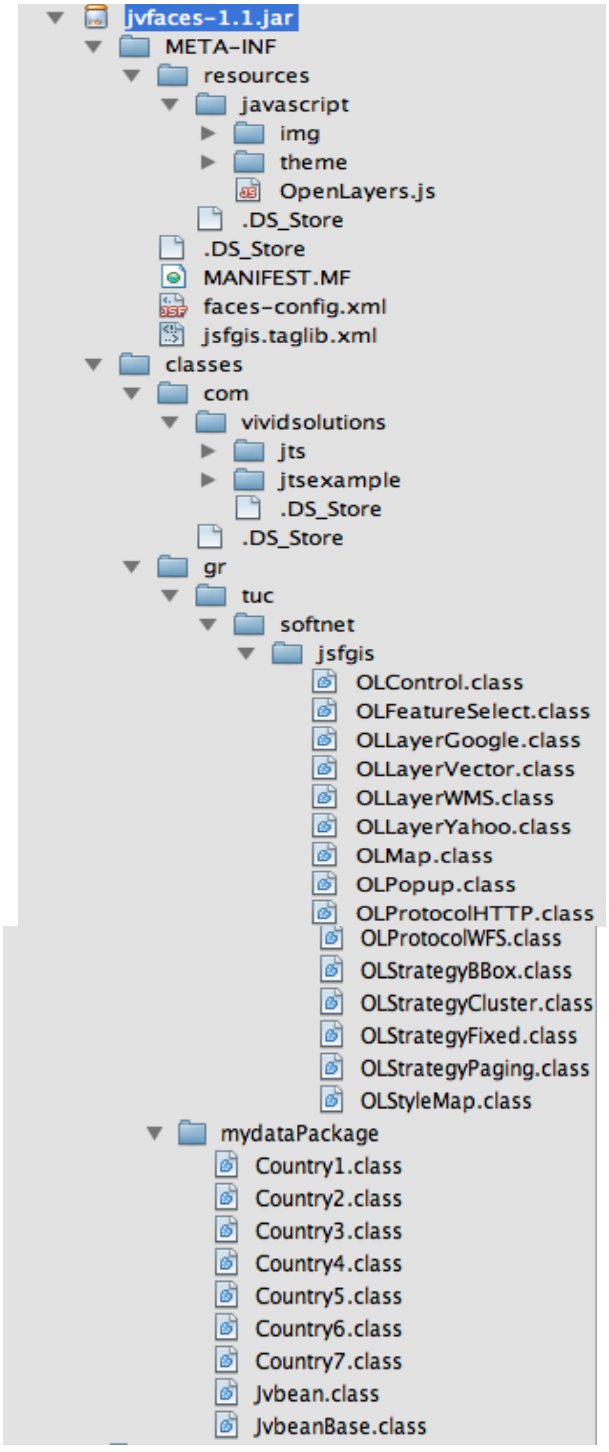
Για να αποφύγουμε τη λάθος χρήση/σύνταξη συνδυάζοντας τα components του JVFACES, κάθε component εκτός του “map” έχει ενσωματωμένο κώδικα Java στη μέθοδο encodeBegin() για τον έλεγχο του γονέα του, ο οποίος σε περίπτωση λάθους προκαλεί ειδικό TagException που εξηγεί το λάθος.

Τμήμα κώδικα του “OLLayerWMS”

```
Boolean test = this.getParent().getClass().getName().contains("OLMap");
if(test.equals(false)) throw new TagException(Tag, “Tag \”layerWMS\” must be inside of tag \”map\”.”);
```

4.10 Ενοποίηση συστατικών και δημιουργία βιβλιοθήκης JVFACES

Έχοντας ολοκληρώσει την υλοποίηση των 15 components της βιβλιοθήκης δεν μένει παρά να τα πακετάρουμε όλα, μαζί με κάποια αρχεία περιγραφής τους, καθώς και ορισμένα recourses, σε ένα και μοναδικό αρχείο .jar όπως ορίζουν οι προδιαγραφές του JSF 2.0. Στην παρακάτω Εικόνα δείχνουμε την δομή του αρχείου αυτού που αποτελεί ουσιαστικά την σουίτα JVFACES.


jvfaces-1.1.jar
META-INF
resources
javascript
img
theme
OpenLayers.js
.DS_Store
.DS_Store
MANIFEST.MF
faces-config.xml
jsfgis.taglib.xml
classes
com
vividsolutions
jts
jtsexample
.DS_Store
.DS_Store
gr
tuc
softnet
jsfgis
OLControl.class
OLFeatureSelect.class
OLLayerGoogle.class
OLLayerVector.class
OLLayerWMS.class
OLLayerYahoo.class
OLMap.class
OLPopup.class
OLProtocolHTTP.class
OLProtocolWFS.class
OLStrategyBBox.class
OLStrategyCluster.class
OLStrategyFixed.class
OLStrategyPaging.class
OLStyleMap.class
mydataPackage
Country1.class
Country2.class
Country3.class
Country4.class
Country5.class
Country6.class
Country7.class
Jvbean.class
JvbeanBase.class

5 Συμπεράσματα

5.1 Συνεισφορά της εργασίας

Οι δυνατότητες της χρήσης των υπηρεσιών χαρτογράφησης μέσω διαδικτύου είναι πλέον γνωστές, όπως και οι τεχνικές δυσκολίες/ζητήματα που αντιμετωπίζουν:

- *Ζητήματα αξιοπιστίας:* Η αξιοπιστία του διαδικτύου και των web servers δεν είναι ακόμα αρκετά καλή. Ειδικά αν ο χάρτης βασίζεται σε εξωτερικές, κατανεμημένες πηγές δεδομένων, πολλές φορές δεν είναι εγγυημένη η διαθεσιμότητα των πληροφοριών.

- *Ζήτημα ταχύτητας του δικτύου:* Οι χάρτες μέσω διαδικτύου χρειάζονται αρκετά υψηλές ταχύτητες μεταφοράς δεδομένων.

- *Απαιτούν μεγάλη πολυπλοκότητα για να αναπτυχθούν και να συντηρηθούν.*

- *Υψηλό κόστος γεωγραφικών δεδομένων:* Αν εξαιρεθούν οι ΗΠΑ όπου τα δεδομένα είναι διαθέσιμα σχετικά φθηνά από την κυβέρνηση, σε άλλες περιοχές όπως η Ευρώπη είναι ακριβά, καθώς διατίθενται από ιδιωτικές εταιρίες. Επίσης σε άλλες αναπτυσσόμενες χώρες δεν έχουν συλλεχθεί τέτοια δεδομένα.

Τα Συστήματα GIS και WEB GIS /WEB MAPPING έχουν ιδιαιτερότητες καθώς απαιτούν εξειδικευμένες γνώσεις σε πολλά και διαφορετικά αντικείμενα όπως χαρτογραφία, κλίμακες απεικόνισης, προβολικά συστήματα, μετατροπή συντεταγμένων, γεω-αναφορά, δόμηση χωρικών βάσεων δεδομένων, τεχνολογίες 3D απεικόνισης και επιπλέον απαιτούν εξειδικευμένες γνώσεις του συγκεκριμένου αντικειμένου / τομέα στον οποίο προορίζονται να εφαρμοστούν.

Το μερίδιο των GIS μέσα στην αγορά των έργων Πληροφορικής είναι σημαντικό, αλλά λόγω των ιδιαίτερων απαιτήσεων τεχνογνωσίας που προϋποθέτουν απαιτείται απόλυτα επαγγελματική διαχείριση και μεγάλη εμπειρία για να αποφέρουν θετικό οικονομικό αποτέλεσμα στο σύνολο. Για το λόγω αυτό, η συγκεκριμένη εργασία έθεσε ως **στόχο** τη σύνθεση κατάλληλων "εργαλείων", τα οποία απευθύνονται στους developers Web GIS συστημάτων, και θα τους παρέχουν έναν δηλωτικό περιβάλλον σύνθεσης διεπαφών για γεωγραφικά δεδομένα και κατ' επέκταση, την αντιμετώπιση των "ζητημάτων" που προκύπτουν από τα Web GIS συστήματα

Ουσιαστικά υλοποιήσαμε μία σουίτα προσαρμοσμένων συστατικών (custom components), που λειτουργούν αποκλειστικά πάνω στο Java-based πλαίσιο ανάπτυξης διαδικτυακών εφαρμογών JavaServer Faces framework (JSF). Η **σουίτα JVFACES** εστίασε στην μείωση της πολυπλοκότητας σύνθεσης των εν λόγω συστημάτων, στην ευκολία διαχείρισης, συντήρησης και επέκτασής τους, καθώς επίσης και στην μείωση του χρόνου και του κόστους που απαιτεί η υλοποίηση τους. Για την επίτευξη αυτών των στόχων, η σύνθεση της σουίτας βασίστηκε σε

ορισμένα εξαιρετικά open source εργαλεία και βιβλιοθήκες, όπως:

- ✔ *η javascript βιβλιοθήκη OpenLayers*, που ειδικεύεται στην σύνθεση γεωγραφικών εφαρμογών, παρέχοντας δυνατότητα άντλησης δεδομένων από πολλές αξιόπιστες πηγές (WMS, WFS, Google Maps, Yahoo Maps, Bing Maps, OpenStreetMap, GeoServer, ArcGIS Server, και άλλα), εκμεταλλευόμενη την τεχνολογία AJAX για γρηγορότερη/αποδοτικότερη επικοινωνία client - server.

- ✔ *η Java βιβλιοθήκη Java Topology Suite*, για την σύνθεση γεωγραφικών δεδομένων και την διευκόλυνση γεωμετρικών υπολογισμών για διανυσματικά δεδομένα.

- ✔ *το πλαίσιο JavaServer Faces*, που με δηλωτικό τρόπο, απλοποιεί τη σύνθεση προσαρμοσμένων συστατικών/ διεπαφών σε μορφή XML.

- ✔ *η τεχνολογία AJAX (Asynchronous javascript and XML)*, για γρήγορη client-side δημιουργία διαδραστικών εφαρμογών

- ✔ *τα standards του Open Geospatial Consortium (OGC),* που έχουν στόχο την υποστήριξη και προώθηση της από κοινού ανάπτυξης των ανοικτών τεχνολογιών και γεωχωρικών δεδομένων.

Συνοψίζοντας, το JVFACES είναι μία σουίτα προσαρμοσμένων συστατικών ανοικτού κώδικα για το JavaServer Faces. Περιλαμβάνει ένα σύνολο 15 JSF συστατικών που αφορούν γεωγραφικά δεδομένα και εκμεταλλεύονται την τεχνολογία AJAX. Κύριος στόχος της είναι να αποτελέσει ένα ισχυρό εργαλείο που θα επεκτείνει τις δυνατότητες του JSF και θα επιτρέπει στους JSF developers την δημιουργία διεπαφών γεωγραφικής πληροφορίας, με δηλωτικό τρόπο.

5.2 *Μελλοντική Επέκταση*

Η σουίτα JVFACES βασίστηκε σε open source εργαλεία και σε τεχνολογίες, σύγχρονες και συνεχώς εξελισσόμενες. Επίσης είναι και η ίδια ανοικτού κώδικα και υλοποιημένη έτσι ώστε να επιτρέπει την μελλοντική επέκταση και εξέλιξή της.

Κάποιες προτάσεις για μελλοντική επέκταση που είμαστε σε θέση να κάνουμε είναι:

- Η εκμετάλλευση του μελλοντικού OpenLayers API version 3.0 (αντί της τρέχουσας έκδοσης 2.10 που χρησιμοποιήθηκε)
- Η υποστήριξη περισσότερων τύπων layer με εκμετάλλευση των:
  - Google Maps API version 3 (αντί της έκδοσης 2)
  - BING Maps API και άλλων
- Η εκμετάλλευση των νέων δυνατοτήτων που προσφέρει το JSF 2.1
- Η υλοποίηση αναλυτικότερων συστατικών/ετικετών αλλά και attributes για περισσότερες δυνατότητες παραμετροποίησης των λειτουργιών πάνω σε διαδραστικούς χάρτες
- Δημιουργία περισσότερων JSF events προσανατολισμένα στην περιγραφή γεωγραφικής πληροφορίας για αποδοτικότερη χρήση των Java Managed Beans

Παραπομπές

[1] Google Docs
[online] https://docs.google.com/

[2] Gmail
[online] https://mail.google.com/mail/

[3] Hotmail [online] https://login.live.com/

[4] Yahoo Mail [online] https://login.yahoo.com

[5] YouTube [online] http://www.youtube.com/

[6] GreekTube [online] http://www.greektube.org/

[7] Facebook [online] http://www.facebook.com/

[8] Twitter [online] http://twitter.com/

[9] Google Talk [online] http://www.google.com/talk/

[10] Google Maps [online] http://maps.google.com/

[11] Yahoo Maps [online] http://maps.yahoo.com/

[12] Bing Maps [online] http://www.bing.com/maps/

[13] ploigos [online] http://www.ploigos.gr/

[14] Web Mapping [online] http://en.wikipedia.org/wiki/Web_mapping

[15] GIS [online] http://en.wikipedia.org/wiki/Geographic_information_system

[16] JavaServer Faces [online]
http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[17] OpenLayers [online] http://openlayers.org/

[18] Java Topology Suite [online] http://www.vividsolutions.com/jts/JTSHome.htm

[19] Asynchronous javascript and XML
[online] http://www.oracle.com/technetwork/articles/javaee/ajax-135201.html

[20] Open Geospatial Consortium [online] http://www.opengeospatial.org/

[21] Web Feature Service [online] http://www.opengeospatial.org/standards/wfs

[22] Web Map Service [online] http://www.opengeospatial.org/standards/wms

[23] Java EE
[online] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[24] Apache Tomcat [online] http://tomcat.apache.org/

[25] Managed-beans
[online] http://download.oracle.com/javaee/5/tutorial/doc/bnawq.html

[26] Web Application Archives
[online] http://java.sun.com/j2ee/tutorial/1_3-fcs/doc/WCC3.html

[27] Java Servlet Technology
[online] http://www.oracle.com/technetwork/java/javaee/servlet/index.html

[28] OpenStreetMap [online] http://www.openstreetmap.org/

[29] UMN MapServer [online] http://mapserver.org/

[30] Mapguide open source [online] http://mapguide.osgeo.org/

[31] GeoServer [online] http://geoserver.org/display/GEOS/Welcome

[32] ka-Map [online] http://ka-map.maptools.org/

[33] World Wind servers [online] http://worldwind.arc.nasa.gov/java/server/readme.html

[34] ArcGIS Server [online] http://www.esri.com/software/arcgis/arcgisserver/index.html

[35] GeoRSS [online] http://georss.org/Main_Page

[36] KML [online] http://code.google.com/apis/kml/documentation/

[37] GML [online] http://en.wikipedia.org/wiki/Geography_Markup_Language

[38] GeoJSON [online] http://geojson.org/

[39] Simple Features – SQL [online] http://www.opengeospatial.org/standards/sfs

[40] GeoConnections [online] http://www.geoconnections.org

[41] British Columbia Ministry of Sustainable Resource Management [online] http://www2.gov.bc.ca/en/themes/families_and_residents/environment_ecosystems_energy/index.page

[42] Centre for Topographic Information – Sherbrooke [online] http://www.cits.rncan.gc.ca/site/eng/index.html

[43] Vivid Solitions, Inc [online] http://www.vividsolutions.com/

[44] OpenGIS [online] http://www.opengis.com/

[45] Styled Layer Descriptor [online] http://www.opengeospatial.org/standards/sld

[46] **Ed Burns, Roger Kitain** [paper] "JavaServer Faces Specification", version 2.0, June 2009, Sun Microsystems, Inc, 4150 Network Circle, Santa Clara, CA 95054 U.S.A

[47] **David Geary, Cay Horstmann** [book] "Core JavaServer Faces", third edition, May 2010, Prentice Hall

[48] **Anghel Leonard** [book] "JSF 2.0 Cookbook", June 2010, Published by Packt Publishing Ltd, 32 Lincoln Road, Olton, Birmingham, B27 6PA, UK

[49] **Ed Burns, Chris Schalk, Neil Griffin** [book] "JavaServer Faces 2.0: The Complete Reference", 2010, McGraw Hill Companies

[50] **Oracle Mojarra JavaServer Faces** [online] http://javaserverfaces.java.net

[51] **JSF 2.0 Tutorials** [online] http://www.coreservlets.com/JSF-Tutorial/jsf2/

[52] **Jim Driscoll**, "driscoll's blog" [online] http://www.java.net/blogs/driscoll/