# Sviluppo

## Generare la documentazione delle API JS

Per generare la documentazione lanciare E:\progetti\[CartoWebNet2](http://code.google.com/p/cartowebnet2/wiki/CartoWebNet2)\tools\yuidoc\bin\yuidoc.bat

Oppure utilizzare il comando in APIdocs in Aptana

Documentazione viene creata in E:\progetti\[CartoWebNet2](http://code.google.com/p/cartowebnet2/wiki/CartoWebNet2)\lib\Cwn2\apidocs\html

Url documentazione: <http://parodis-dts-pc.datasiel.net/CartoWebNet2/lib/Cwn2/apidocs/html/module_CWN2.html>

## Custom ExtJS Themes

**spket**

<http://www.sencha.com/blog/spket-ide-1611-released-includes-new-ext-theme-builder/>

<http://www.spket.com/>

<http://www.spket.com/ext-theme-builder.html>

Installato plugin su Aptana 3.0 di parodi.datasiel.net

Creato progetto “extjs themes” in aptana

1. Selezionare Perspective Spket II (in alto a destra)
2. From the menu bar, select **File > New > File**
3. In the **File name** field, type **ext.theme** as the name for the new file. Then click **Finish**.
4. Click the **Browse...** button to specify the path to the **Ext Resources** folder which contains css and images:

**E:\progetti\CartoWebNet2\lib\ExtJs\resources**

1. If you check the **Full** option, it will generate "ext-all-**name**.css", otherwise, it will generate "xtheme-**name**.css".
2. In the effect page, adjust the scale to set the color.
3. Click the **Export...** button and specify a folder to save the new resources.

## Creare nuovi bottoni

### Codice del Bottone

Creare il codice javascript del bottone in un file "nomebottone.js" nella dir "Cwn2/src/buttons/". Il codice js deve contenere un oggetto con 2 proprietà:

* **options**: hash table delle opzioni del bottone, tra cui:
  + **id**: identificativo del bottone. E' quello da mettere nel "name" della configurazione
  + **tooltip**: tooltip associato al bottone
  + **iconCls**: classe css dell'icona del bottone
  + **handler**: eventuale funzione da richiamare alla selezione del bottone
  + **control**: eventuale controllo ol associato al bottone
* **create**: funzione di creazione del bottone.

Per i bottoni con interazione su mappa bisogna impostare le seguenti proprietà:

* enableToggle: true
* toggleGroup: "mapInteractToggleGroup"

Deve esserci inoltre il caricamento dell'array itemDefs della toolbar.

Nel caso di bottone associato a controllo OL deve esserci nella funzione create l'aggiunta del controllo alla mappa

### Stile

L'attributo "iconCls" delle options del bottone fa riferimento ad una classe che deve essere contenuta nel base.css dello stile gestito (p.es: stili/default/base.css). Nella classe css viene referenziata l'immagine del bottone, che in genere è contenuta nella dir "icons" dello stile (p.es: stili/default/icons).

Esempio:

.fit {  
  background-image:url("icons/fit\_all.png") !important;  
  height:20px !important;  
  width:20px !important;  
}

### Configurazione

I bottoni sono caricati come elementi della toolbar. Sono raggruppati in gruppi di elementi.

Ogni elemento ha i seguenti attributi:

* type: tipo dell'elemento: "button"
* name: identificativo del bottone (corrisponde all'attributo id delle options di creazione del bottone)

Esempio:

        "toolbar" : {  
                "itemGroups": [  
                        {  
                                "name": "navigazione",  
                                "items": [  
                                        ...  
                                        {"type": "button", "name": "fitall"},  
                                        ...  
                                ]  
                                ...

### Esempi

**Esempio di bottone associato a controllo OL.**

// definizione del bottone  
var button = {  
        options : {  
                id: "fitall",  
                tooltip: "Fit All - Visualizza alla massima estensione",  
                iconCls: "fit",  
                pressed: false,  
                control : new OpenLayers.Control.ZoomToMaxExtent()  
        },  
        create : function(options) {  
                // aggiungo il controllo alla mappa  
                CWN2.Base.map.addControl(options.control);  
                // ritorno la action  
                return new GeoExt.Action(options);  
        }  
};  
  
// caricamento array itemDefs  
CWN2.ToolbarBuilder.itemDefs[button.options.id] = button;

**Esempio di bottone che apre una finestra.**

Notare la funzione "handler" che mostra la finestra che viene create nella funzione "create" del bottone.

var button = {  
        options : {  
                tooltip: "Caricamento Livelli",  
                iconCls: "loadlayers",  
                pressed: false,  
                id: "loadlayers",  
                handler: function(){  
                        // rendo visibile la finestra dell'help  
                Ext.getCmp("load-layer-win").show();  
            }  
        },  
        create : function(options) {  
                // costruisco la finestra di help  
                var loadLayerWin = new Ext.Window({  
                title: "Caricamento Livelli",  
                        id: "load-layer-win",  
                height: 400,  
                width: 600,  
                layout: "fit",  
                closeAction: "hide",  
                items: []  
            });  
                  
                return new GeoExt.Action(options);  
        }  
};  
  
CWN2.ToolbarBuilder.itemDefs[button.options.id] = button;

### Configurazione dinamica

Alcuni bottoni hanno bisogno di opzioni di configurazione dinamica.

Queste configurazioni dinamiche vengono gestite nell’oggetto di configurazione buttonsConfig della configurazione della toolbar (configuration 🡪 application 🡪 layout 🡪 toolbar 🡪 buttonsConfig).

Questo oggetto è una hash table avente come key il nome della configurazione (es: “canaliTematici”) e come value l’oggetto configurazione

I bottoni multitab (es: loadLayer) utilizzano un array (panels) contenente gli oggetti di configurazione dei pannelli. Questi oggetti dipendono dal tipo di bottone e dal tipo di pannello.

## Esecuzione di funzioni all’avvio

### Funzioni definite nel codice JavaScript

Definire la funzione nel codice JS e indicarla nel parametro callback delle opzioni di inizializzazione

### Funzioni definite in configurazione

Definire oggetto onLoadCallback in Configuration.Application che contiene il nome della funzione da eseguire all’avvio e i parametri richiesti dalla funzione.

La funzione deve essere definita nell’oggetto (namespace) CWN2.OnLoadCallbacksRegistry

Vedi gestione ALFA\_GIS con find per coordinate (funzione CWN2.OnLoadCallbacksRegistry.loadMarker)

# Sviluppo Applicazioni

Sviluppo di applicazioni basate su CWN2

## Regole generali

Le applicazioni sono sotto la dir \CartoWebNet2\pages\apps, preferibilmente in una sottocartella dedicata.

Eventuali configurazioni javascript differenziate negli ambienti di sviluppo e esercizio vanno messi in un file config.js sotto la directory della applicazione

Il codice js può essere incluso nel file html o in un file js apposito.

**Configurazione applicazione**

La configurazione può essere definita come varibile direttamente nel codice oppure essere ritornata da un servizio.

**Creazione della applicazione CWN2**

L’applicazione CWN2 viene creata passando un oggetto di inizializzazione contenente:

* appConfig: configurazione della applicazione
* divID: nome del div contenente la applicazione mappa
* callBack: eventuale funzione di callback chiamata dopo la inizializzazione
* proxy: proxy da utilizzare per chiamate xml/json fuori dominio
* language: linguaggio (“it”/”en”)
* debug: flag di debug

Esempio

cwn2\_app = new CWN2.Application({

appConfig: config,

divID: "mapdiv",

callBack: loadMap,

proxy: "/CartoWebNet2/services/proxy/proxy.asp?url=",

language:"it",

debug:true

});

**Callback**

La funzione di callback deve essere definita prima della creazione della applicazione e verrà richiamata dopo la inizializzazione passando come parametro l’oggetto applicazione.

## Ingresso con evidenziazione su uno o più oggetti vettoriali

Vedi EsempioPanel.html.

Deve essere definita una funzione di callback che effettua la evidenziazione degli oggetti. Questa funzione deve essere indicata nell’oggetto di inizializzazione della applicazione cwn2 e in genere contiene la definizione di un oggetto di configurazione e il richiamo della funzione “hiliteFeatures” a cui viene passato l’oggetto di configurazione

## Applicazione che carica una mappa del Catalogo SIT

Esempio di utilizzo **EsempioMappaCatalogo.html**:

* carica una mappa in base al codice\_catalogo

La applicazione effettua le seguenti operazioni

* Definisce l’oggetto configurazione della applicazione
* Legge il parametro ID\_MAP (codice catalogo della mappa) in querystring
* Controlla che il parametro sia impostato
* Crea la applicazione CWN2, indicando il parametro idMap

## Applicazione con find WMS

La find può essere fatta utilizzando ALFA\_GIS oppure passando parametro di inizializzazione

Esempio di utilizzo **EsempioFindWMS.html**:

* carica una mappa in base al codice\_catalogo (vedi EsempioMappaCatalogo.html)
* effettua la find

La applicazione effettua le seguenti operazioni

* Definisce l’oggetto configurazione della applicazione
* Legge i seguenti parametri in querystring:
  + ID\_MAP (codice catalogo della mappa)
  + FIND\_LAYER (nome del layer su cui effettuare la find)
  + FIND\_FIELD (nome del campo)
  + FIND\_VALUES (elenco dei valori da ricercare separati da ‘|’): viene separato in un array di valori
* Controlla che i parametri siano impostati
* Crea la applicazione CWN2, indicando i parametro idMap e findOptions

Per gestire chiavi/valori multipli si utilizza il separatore ","

Es:

fields: "CT24\_COD\_COM,CT24\_SEZ,CT24\_FOGLIO,CT24\_NUMERO"

values: ["D969,L,8,147"]

# Configurazione

## Livelli di sfondo

I livelli di sfondo hanno la proprietà isBaseLayer impostata a “true”

I livelli si sfondo disponibili sono:

* Google Satellite
* Google Stradario
* Google Ibrida
* Google Terreno
* OpenStreetMap

Per impostare il livello di sfondo attivo all’avvio è necessario impostare la proprietà visible a true per il livello e impostarla a false per gli altri livelli di sfondo

## Gestire lo stile dei livelli vettoriali

Documentazione OL:

<http://docs.openlayers.org/library/feature_styling.html>

Per i livelli vettoriali puntuali è necessario utilizzare una “ExternalGraphics” (immagine png/gif/jpg) in modo da avere la visualizzazione anche sulla legenda.

Per configurare lo stile è necessario configurare l’elemento styleMap del layer.

Esempio di configurazione:

**"styleMap"**: **{**

**"default"** : **{**

**"cursor"**: "pointer",

**"pointRadius"**: 10,

**"externalGraphic"**: "http://parodis-dts-pc.datasiel.net/CartoWebNet2/icons/events/alien.png",

**"graphicTitle"**: "${label}",

**"graphicWidth"**: 32,

**"graphicHeight"**: 37,

**"graphicYOffset"**: -37

**}**,

**"select"** : **{**

**"cursor"**: "pointer",

**"pointRadius"**: 10,

**"externalGraphic"**: "http://parodis-dts-pc.datasiel.net/CartoWebNet2/icons/events/alien\_s.png",

**"graphicTitle"**: "${label}",

**"graphicWidth"**: 32,

**"graphicHeight"**: 37,

**"graphicYOffset"**: -37

**}**

**}**

“default” determina lo stile normale del layer

“select” determina lo stile degli oggetti selezionati

Ho installato una libreria di icone su \CartoWebNet2\icons, le icone sono suddivise per area tematica.

NOTA: è necessario creare uno stile per la selezione a partire dallo stile di default

**Punto di applicazione delle icone “ExternalGraphics”**

Le icone vengono applicate sul punto medio del lato superiore se l’icona richiede l’applicazione su un altro punto bisogna spostarla opportunamente utilizzanndo le proprietà graphicXOffset e graphicYOffset

Esempio: l’icona  deve essere configurata in modo che il punto di applicazione sia sul punto mediano del lato inferiore. Quindi se l’icona ha graphicHeight=37 deve essere impostato graphicYOffset=-37

## Gestire la info sui livelli vettoriali

….

## Gestire l’etichetta sui livelli vettoriali

….

## Gestire la legenda per livelli multiclasse

Impostare nella configurazione del livello l’attributo classes contenente un array di elementi con attributi

* label: etichetta
* icon: url della icona

Configuration.Layers[i].legend.classes

Vedi E:\progetti\CartoWebNet2\config\ProvaCaricamentoLayerMulticlasse.json

## Gestire la legenda in un documento (pdf,html,png,jpg)

Impostare nella configurazione del livello l’attributo popupUrl, impostato con la url del documento da visualizzare

Configuration.Layers[i].legend.popupUrl

## Gestione bottone selezionato all’avvio

Impostato parametro "pressed" in oggetto “toolbar” deve contenere il nome del bottone da selezionare all’avvio.

Esempio (selezioneterritoriale.js di DownloadDatiRepertorio)

"pressed": "drawRegularPolygon",

## Gestione Geocode: configurazione

I parametri di configurazione del bottone find sono:

* "zoomToViewport": true/false. Indica se effettuare lo zoom sulla viewport passata dal servizio google (valido per la ricerca per indirizzi con google)
* "externalGraphics": url della icona da utilizzare come segnaposto es: "http://geoportale.regione.liguria.it/geoviewer/stili/default/icons/marker\_yellow.png",
* "graphicWidth": dimensione in pixel della larghezza della icona segnaposto,
* "graphicHeight": dimensione in pixel della altezza della icona segnaposto,
* "graphicYOffset": dimensione in pixel dell’offset lungo la y della icona segnaposto