# Oggetti del Framework (concepts)

GeoWeb si compone di oggetti di base, ai quali si farà spesso riferimento in questa documentazione. In particolare vedremo che in una istanza di GeoWeb potremo configurare applicazioni composte da Progetti, i quali potranno essere organizzati in Categorie.
Ogni Progetto conterrà i riferimenti a Temi e Classi definiti in GeoWeb, a cui sono riferibili automatismi, maschere di editing dei dati, mappe e planimetrie in cui visualizzare la parte geometrica delle anagrafiche gestite, e report di restituzione dei dati.

## Istanza

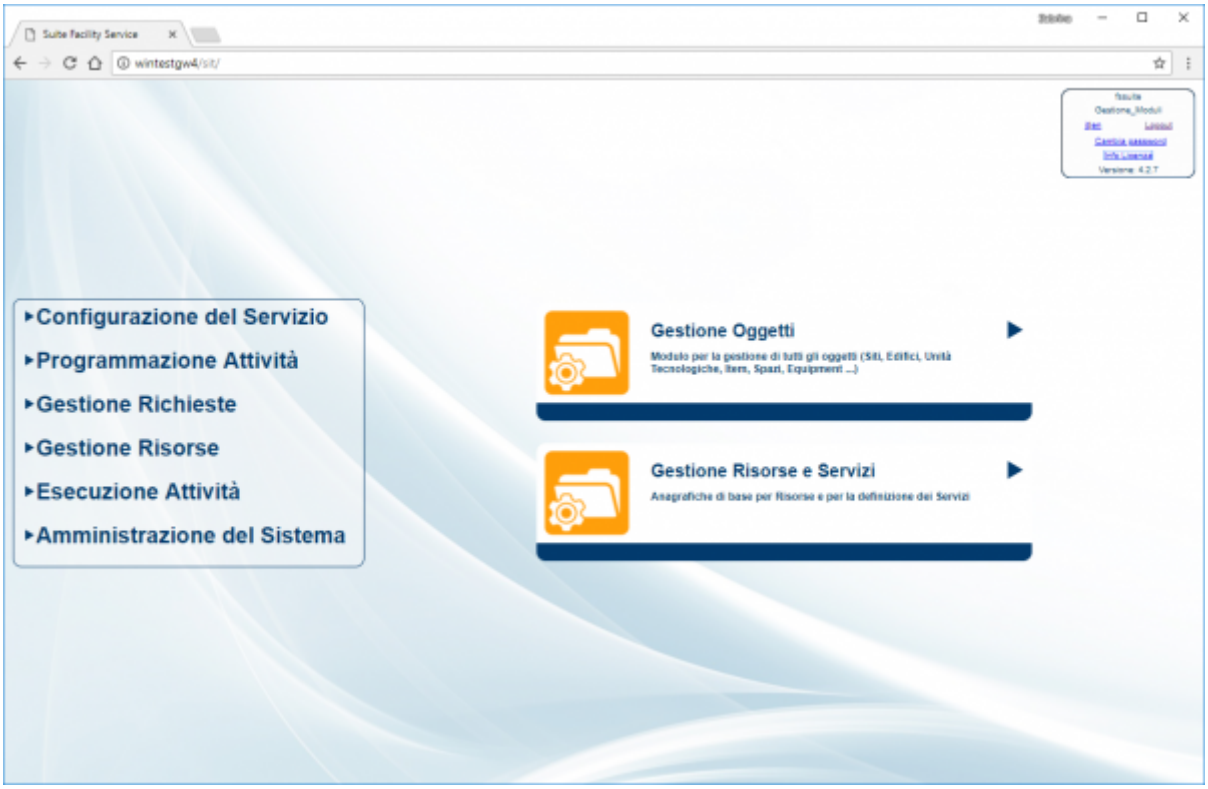
Per istanza di Geoweb si intende una singola installazione del framework, raggiungibile via web tramite un URL. A tale istanza si potrà accedere tramite autenticazione, fornita attraverso le Pagine di Accesso.

Una istanza viene configurata all'interno di un Application Server Tomcat, attraverso l'associazione a uno schema DB per i Dati e uno Schema DB per i Metadati, a un libreria MapGuide, a una cartella di Risorse Web Statiche (immagini, templates, documenti, ecc.) ecc.

Tale configurazione viene attuata attraverso l'utilizzo dei file di Configurazione.

## Progetti

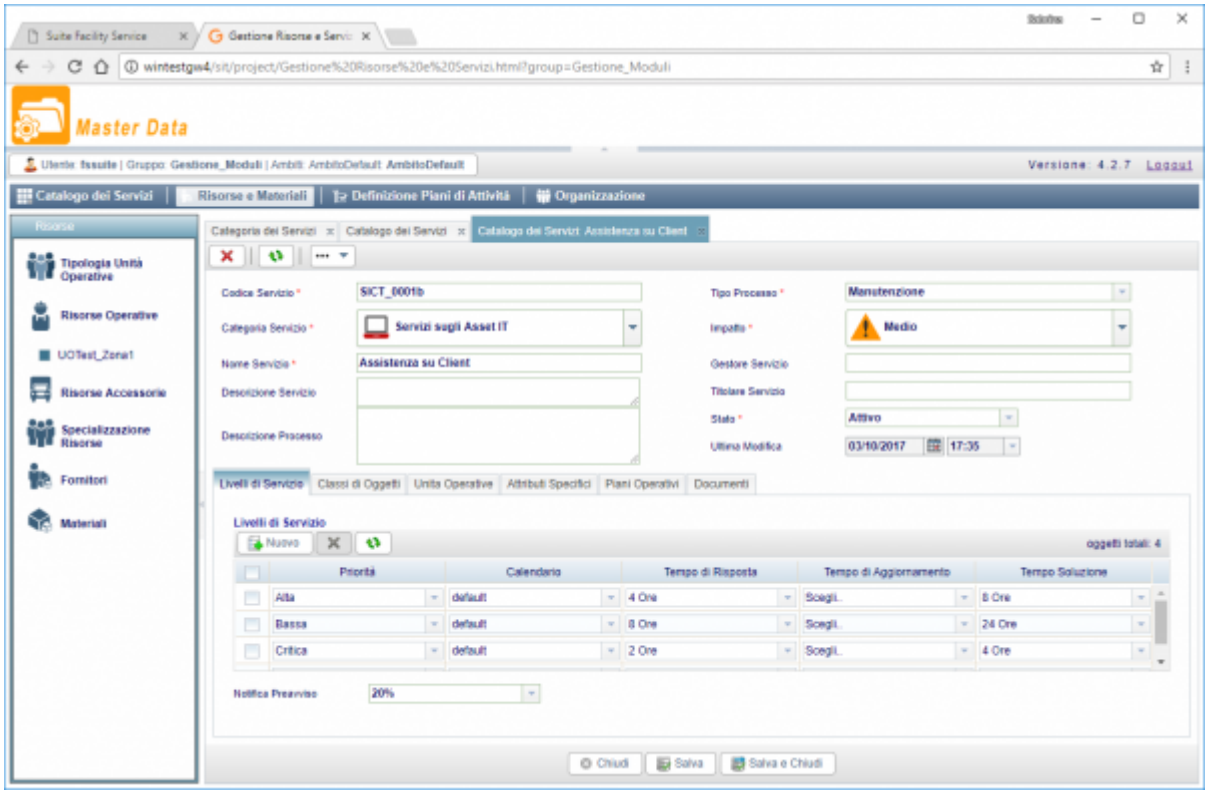
Una istanza può ospitare più Progetti, ciascuno dei quali da accesso a una struttura organizzata (container) in menù e sottomenù, da cui accedere alle Classi, alle Mappe, e ad altri componenti disponibili nel framework. Subito dopo la login il sistema mostra un elenco dei Progetti a cui ha accesso l'utente autenticato.



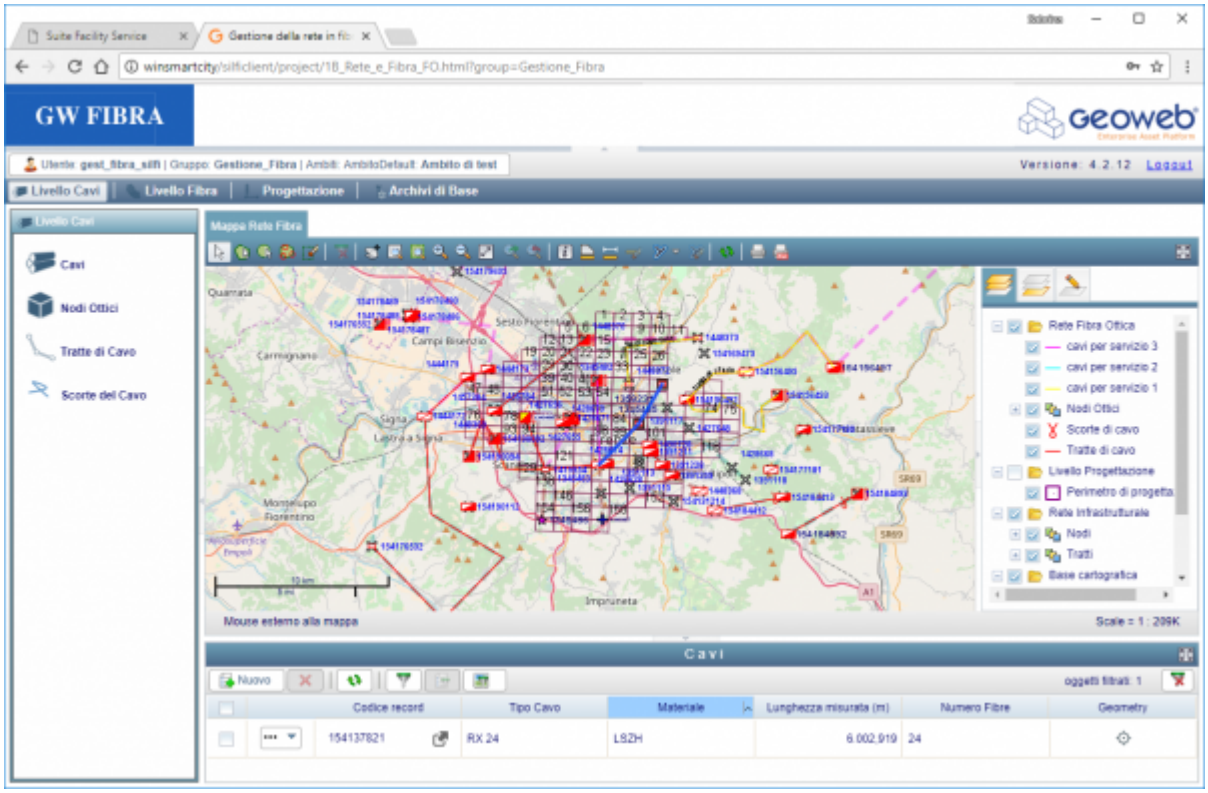

Come è possibile vedere dall'immagine, i Progetti, rappresentati dai due banner a dx, possono essere categorizzati, con un menù che compare sulla sx in maniera da organizzare meglio l'interfaccia utente in caso di applicazioni molto complesse.

I Progetti possono essere di due tipologie: GeoManager e GeoExplorer. I Progetti di tipo GeoManager danno accesso ad un contenitore strutturato principalmente per mostrare dati, mentre i progetti di tipo GeoExplorer danno accesso ad applicazioni dove il contenuto centrale è una (o più di una) mappa GIS o una (o più di una) planimetria costruttiva (es. la rappresentazione di un piano di un edificio). Cliccando su un banner, il Progetto viene aperto in una nuova finestra del browser, in maniera che sia possibile aprire più di un Progetto contemporaneamente.

***Progetto di tipo GeoManager***



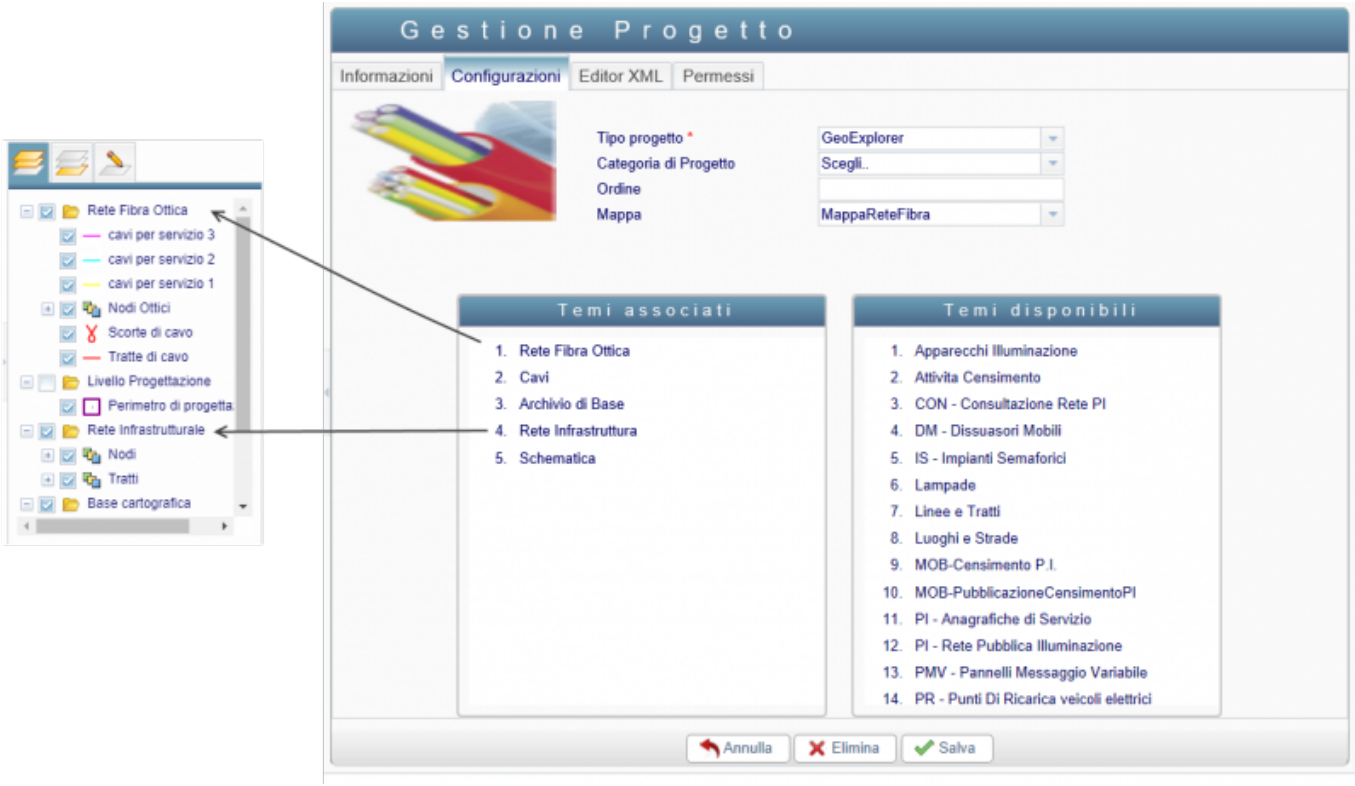

***Progetto di tipo GeoExplorer***




# Temi

Per Tema, in Geoweb, si intende un raggruppamento di Classi, messe insieme secondo logiche puramente applicative, che il Configuratore può organizzare a proprio piacimento, tenendo presente che:

- ogni Classe appartiene unicamente ad un solo Tema
- con Progetti di tipo GeoExplorer (vedi paragrafo precedente) il Tema va associato al Progetto in maniera che sia possibile la visualizzazione in Mappa delle Classi con componente geometrica nella legenda interattiva dei layers.




# Classi

La Classe rappresenta la definizione dell'oggetto centrale di GeoWeb. E' attraverso la Classe infatti che vengono legati i meccanismi principali di funzionamento delle applicazioni. E' necessario creare classi per:

- definire Form di inserimento, Modifica e Consultazione dei dati
- definire e attuare Processi
- visualizzare Layer in Mappa
- importare ed esportare dati in excel o shape file
- produrre report da stampare

Una Classe è riferita generalmente a una anagrafica, ovvero a una Entità del Database, la quale può essere una semplice tabella oppure una vista logica.

Le Classi hanno una configurazione minima in cui va indicato:

1. un **nome** univoco all'interno dell'Istanza, un etichetta e una descrizione
2. la **tabella** o vista da cui attingere i dati per consultare l'anagrafica, e il nome della tabella (in questo caso la tabella principale) in cui salvare le transazioni effettuate tramite GeoWeb
3. il nome del campo che diventerà la **chiave univoca** della Classe e del nome del campo che rappresenterà il **nome dell'oggetto** per l'utente che utilizza l'applicazione

4. la definizione degli **Attributi**, ovvero i campi dell'anagrafica renderizzati attraverso specifici controlli da mostrare nella form
5. un **Layout** grafico in cui posizionare gli Attributi

Possono inoltre essere riferiti alla classe:

1. **Azioni**, ovvero automatismi che si attuano lato client e che eseguono operazioni lato client e/o script lato server
2. **Trigger**, ovvero procedure lato server che vengono eseguite in maniera automatica al verificarsi di transazioni come l'inserimento, la modifica e la cancellazione di un record dell'anagrafica
3. **Report**, in PDF, in Excel, di dettaglio e/o di lista, ovvero che stampano il singolo record visualizzato oppure il recordset corrente, ottenuto da filtri o da selezioni effettuate dall'utente
4. **Relazioni**, definizione esplicita di relazioni tra tabelle, nelle quali si definiscono i campi di collegamento e le cardinalità
5. **Processi** (Business Process Management), in sostanza sequenze operative più o meno complesse in cui concorrono anche più attori, diversi task utente e task di sistema per determinare l'avanzamento e la conclusione della sequenza
6. **Layer**, cioè strati informativi in cui l'informazione geometrica/geografica contenuta nell'anagrafica viene rappresentata in forma planimetrica per essere rappresentati in una mappa 2D
7. **Scene**, ovvero contenitori della scena 3D ... (da completare)

Ogni classe può essere richiamata direttamente da menù nel Progetto a cui fa riferimento, oppure essere richiamata attraverso meccanismi di presentazione più strutturati, come i menù gerarchici, task di Processo, report di cruscotto html, ecc. Una classe può anche utilizzare altre classi per presentare contenuti relazionati.

## Attributi e Controlli (Widget)

Gli Attributi sono il modo in cui la classe riconosce il singolo campo dell'anagrafica da gestire, e lo presenta all'utente finale attraverso un controllo (o Widget) utile a guidarne e finalizzarne la compilazione.

La definizione minima di un Attributo consiste in:

- **Nome**, che deve essere presente e univoco all'interno della classe; la cosa migliore è definire il nome dell'attributo uguale al nome del campo
- **Etichetta**, ovvero la descrizione che viene mostrata nel form relativamente al controllo
- **Descrizione**, ovvero una descrizione più estesa che viene mostrata all'utente sotto forma di tooltip
- **Campo**, Campo dell'anagrafica associata alla classe
- **Tipo Dati**, ovvero tipologia dati del campo di origine. GeoWeb si astrae dall'ambiente DB (Oracle, Postgres, SQL Server) e accetta i seguenti tipi di dato:
    - STRING
    - INTEGER
    - NUMBER
    - DATE
    - BOOLEAN
    - DOCUMENT

    - IMAGE
    - POINT
    - POLYLINE
    - POLYGON
    - Nessuno (per definire attributi legati all'anagrafica in maniera diversa che attraverso un campo, es. Tabella collegata, Etichetta o Tag HTML statici, Componenti particolari, ecc.)

- **Tipo Widget**, ovvero con quale controllo, più o meno strutturato, si desidera presentare il campo all'utente finale per la compilazione; non tutti i Tipi di Widget sono applicabili agli stessi Tipi di Dati.
- **Parametri Widget**, cioè le configurazioni specifiche di ogni singolo widget, che variano in funzione del Tipo Widget scelto. Possono quindi essere indicate, ad esempio: la larghezza in pixel del controllo, il pattern specifico in caso di campo Numerico o Data, il comportamento (sola lettura, obbligatorio, ecc.) all'interno della form, un valore di default impostato all'apertura, la tabella da cui selezionare in caso di presentazione di un elenco di valori, ecc.
  I parametri degli widget sono definiti attraverso delle strutture xml specifiche per Tipo Widget.

Una volta definiti i singoli Attributi occorre indicare quale di questi dovrà comparire nella modalità di visualizzazione dell'anagrafica come **Lista**, quale di questi dovrà essere utilizzato per fare un **Filtro** sui record, e come organizzarli in **Gruppi Attributi** per presentarli nel **Layout** delle maschere di dettaglio.

## Layout e Gruppi Attributi

Definire un layout significa 'disegnare' il form attraverso cui l'anagrafica potrà essere movimentata dall'utente finale.
Geoweb mette a disposizione delle strutture standard semplici che possono essere assemblate e annidate tra loro per ottenere la maggiore elasticità possibile, pur senza appesantire il framework da complessi meccanismi di form design.

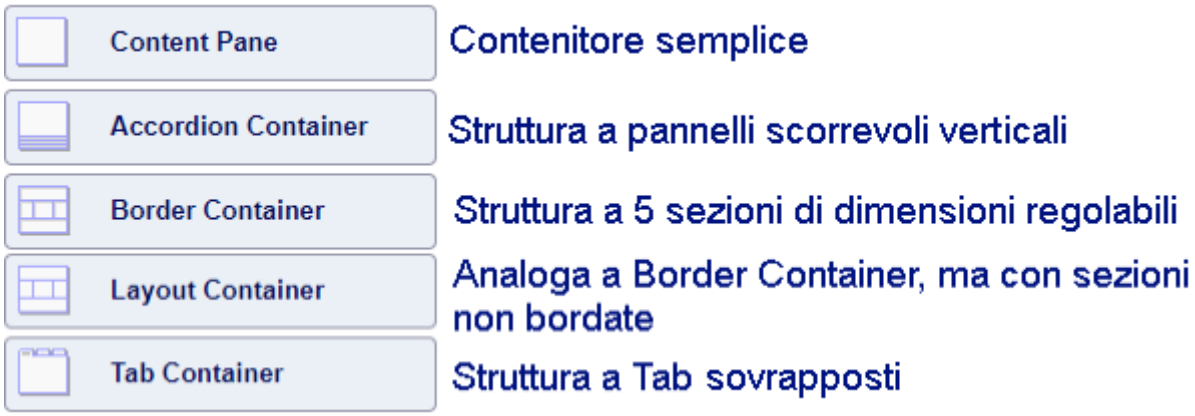
La struttura layout di fatto è un 'Contenitore' XML, che può essere frazionato in strutture più piccole secondo schemi preordinati. Ciascuna struttura può annidarne altre, allo scopo di aumentare le possibilità di graphic design proposte all'utente implementatore.

Definizione XML di un layout

```
<?xml version='1.0' encoding='UTF-8'?>
<ContainerNode type="TabContainer" width="800" widthUOM="px" height="0"
heightUOM="px" title="Tab Container"
                     tabPosition="top" startSelectedInParent="false">
    <ContainerNode type="LayoutContainer" title="Dati Contratto"
startSelectedInParent="false">
        <ContainerNode type="ContentPane" region="center"
title="Administrative Form"
                               startSelectedInParent="false"
                               attributeGroup="Administrative Form"/>
        <ContainerNode type="ContentPane" height="50" heightUOM="%"
region="bottom" title="List Assets
                               Form"
```

```
                                startSelectedInParent="false"
attributeGroup="List Assets Form"/>
        <ContainerNode type="ContentPane" width="50" widthUOM="%"
region="left" title="Main Contract Form"
                                startSelectedInParent="false"
attributeGroup="Main Contract Form"/>
    </ContainerNode>
    <ContainerNode type="LayoutContainer" title="Scadenziario"
startSelectedInParent="false">
        <ContainerNode type="LayoutContainer" region="center" title="Layout
Container" startSelectedInParent="false">
            <ContainerNode type="ContentPane" region="center" title="Event
Alert" startSelectedInParent="false"
                                                attributeGroup="Event Alert"/>
            <ContainerNode type="ContentPane" height="65" heightUOM="px"
region="top" title="Event Main"
                                                startSelectedInParent="false"
attributeGroup="Event Main"/>
            <ContainerNode type="ContentPane" height="230" heightUOM="px"
region="bottom" title="Event Recipient list"
                                                startSelectedInParent="false"
attributeGroup="Event Recipient list"/>
            <ContainerNode type="ContentPane" width="50" widthUOM="%"
region="left" title="Event Event data"
                                                startSelectedInParent="false"
attributeGroup="Event Event data"/>
        </ContainerNode>
    </ContainerNode>
</ContainerNode>
```

I layout possono avere i seguenti schemi:




**Gruppi di Attributi**

Gli attributi vengono posizionati nelle sezioni del Layout incapsulati in strutture denominate 'Gruppi di Attributi'.

Ciascun gruppo di attributi, definito da un nome, può contenere un numero teoricamente (dipende dai

limiti fisici di interfaccia, dimensioni del monitor, risoluzione grafica minima, ecc.)) illimitato di attributi disposti tra di loro in ordine verticale.

Fino alla versione 4.4, un attributo poteva essere contenuto da un SOLO gruppo Attributi.

A partire dalla versione 4.4 invece esiste una relazione molti a molti tra Attributi e relativi Gruppi. Questa nuova struttura dati fornisce una libertà maggiore di disegno delle interfacce di una singola classe, permettendo di definire tutti i layout necessari per le varie tipologie di utilizzo (processi, document, ecc.) senza necessità di duplicare la classe.

## Azioni

Le **Azioni** rappresentano la possibilità di eseguire routines più o meno complesse e attivate da una scelta dell'utente (click sul pulsante o selezione dal menù a tendina) o da eventi specifici (es. apertura della form).

Tendenzialmente le **Azioni** contengono codice eseguibile lato client, e scritto in linguaggio JavaScript. Sono naturalmente accessibili tutte le API Javascript. Tra le API Javascript sono inoltre disponibili funzioni pronte per chiamare delle routine server-side, scritte in linguaggio Groovy, e definite come API Java.

Geoweb mette a disposizione in questa wiki una [**[gwusermanual:interface:action#libreria_di_azioni|libreria di azioni]]** da definire e utilizzare dove opportuno.

### Tipologie di Azioni

Di seguito sono elencate sinteticamente le Tipologie di azioni, ciascuna delle quali è meglio dettagliata nella pagina Azioni.

- **Dettaglio**: l'azione compare sul form di Dettaglio della classe, posizionata nel menù a tendina delle Azioni . L'azione agisce sul record corrente, che è disponibile attraverso l'utilizzo della variabile: *data.itemDB.<nome del campo>*
- **Dettaglio (posto nella Toolbar)**: il comportamento dell'azione è analogo a quello precedente, solo che l'azione compare come un pulsante nella toolbar della form
- **Selezionati in Lista**: l'azione compare sulla toolbar della lista, posizionata nel menù a tendina delle Azioni, e viene eseguita sul recordset correntemente visualizzato e/o filtrato.L'oggetto *grid* rende disponibile l'accesso al recordset corrente.
- **Selezionati in Lista (posto nella Toolbar)**: il comportamento dell'azione è analogo a quello precedente, solo che l'azione compare come un pulsante nella toolbar della lista
- **Al caricamento del dettaglio (eseguita in automatico)**: l'azione viene eseguita senza input utente, una volta che il form di dettaglio viene aperto e si è completato il caricamento di tutti gli elementi che lo compongono
- **Ad inserimento record concluso (eseguita in automatico)**: l'azione viene eseguita senza input utente, una volta che viene completato l'inserimento di un nuovo record
- **A salvataggio record concluso (eseguita in automatico)**: l'azione viene eseguita senza input utente, una volta che viene completato il salvataggio di un record aperto per la modifica

- **Alla chiusura del dettaglio (eseguita in automatico)**: l'azione viene eseguita senza input utente, quando l'utente esegue la chiusura del form di Dettaglio tramite la selezione del tasto Chiudi
- **Alla apertura del progetto**: l'azione viene eseguita senza input utente, quando viene aperto un progetto
- **Alla apertura del progetto, controllo ACL (eseguita in automatico)**: l'azione viene eseguita senza input utente, quando viene aperto un progetto se l'utente ha i permessi di esecuzione dell'azione
- **Classe**: si definisce in questo modo un'azione che deve agire su tutti i record della classe, indipendentemente dal recordset selezionato. Ne è un esempio l'azione 'Aggiorna Cache'. L'azione compare sul menù delle azioni, nella toolbar di lista.
- **Libera**: una azione Libera non compare automaticamente da nessuna parte, occorre quindi definire un meccanismo che la richiami, come un ACTION_WIDGET, un leafItem di tipo leafItemAction, un hyperlink posizionato in una Report HTML.
- **Posta in sostituzione del bottone Nuovo**: quando verrà utilizzato questo tipo di azione, il classico bottone 'Nuovo' (per la creazione di un nuovo record), verrà sostituito con il codice contenuto nell'azione. In questo modo potrà essere personalizzato a seconda delle esigenze.

## Relazioni

Le classi di Geoweb possono essere esplicitamente relazionate tra di loro. Questo è fattibile se esistono campi, in genere codificati, da un lato e dall'altro, che permettano di 'agganciare' i record di una classe ai record di un altra.

E' necessario inoltre che i campi utilizzati per stabilire la relazione siano dello stesso tipo tra loro (entrambi stringa oppure entrambi interi).

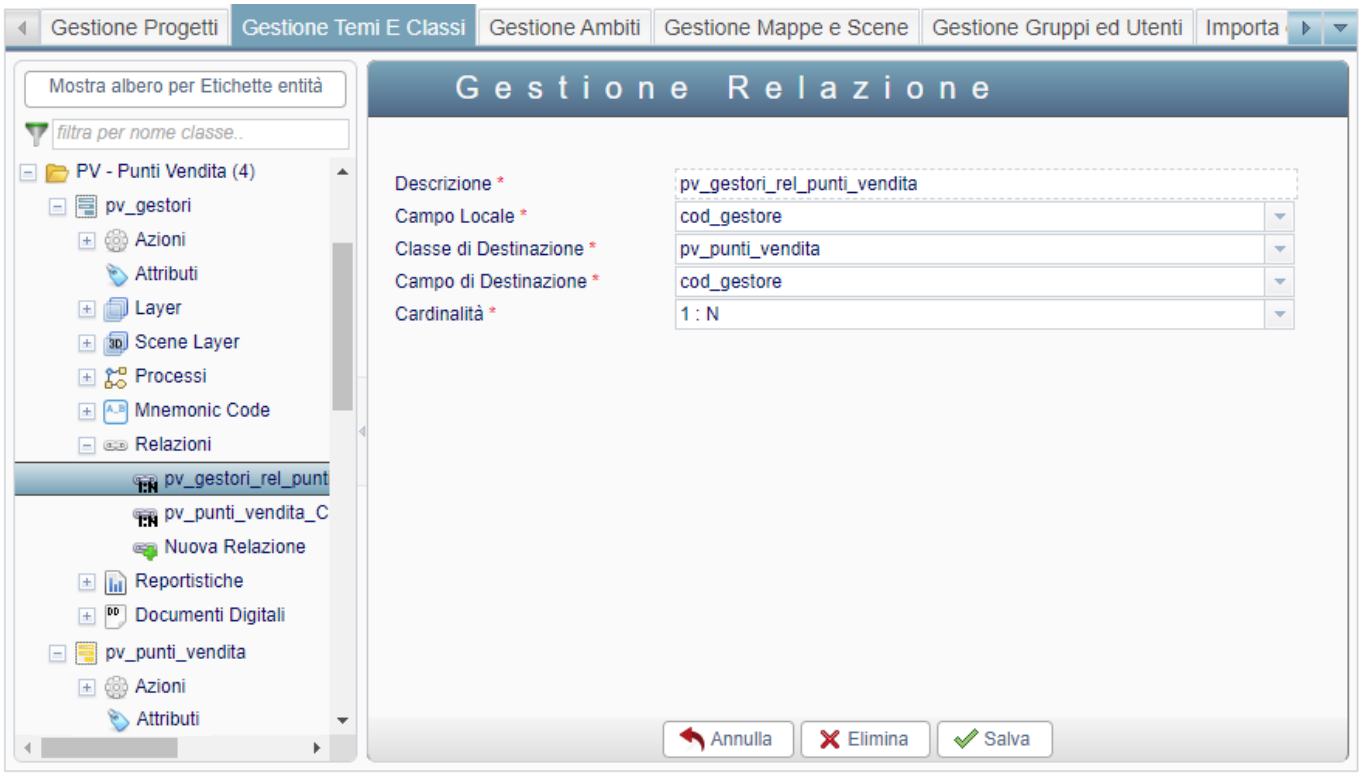
Il tutto avviene in maniera analoga a quando si definisce una Foreign Key sul DB.

Stabilire una relazione serve per la configurazione di particolari Widget, come la LinkList e la LinkListNaM.

I parametri da configurare sono i seguenti:

- nome della Relazione (usare una denominazione più chiara possibile)
- campo locale (da usare nella relazione)
- classe da relazionare
- campo esterno (da comparare al 'campo locale')
- cardinalità della relazione (1:N, N:1)

La configurazione di una relazione in una classe comporta che Geoweb, in automatico, definisca la relazione opposta nella classe relazionata.




## Report di classe

In Geoweb ogni classe viene corredata di una funzione di stampa di Dettaglio standard, denominata Stampa Scheda, che produce un PDF stampabile di tutti gli attributi presenti nella classe e riportati organizzati per Gruppi Attributi, ordinati in ordine alfabetico.

Questa funzione, pur essendo utile perché risponde alla esigenza primaria di avere la stampa dei contenuti di un singolo record, è generalmente insufficiente a soddisfare la normale esigenza di dotare le anagrafiche gestite di una report di stampa formalmente ed esteticamente efficace.

Geoweb quindi permette di configurare **Report di Classe** costruite per stampare uno (dettaglio) o più record (lista), che possono essere realizzate in maniera completamente personalizzata e attinente alla tematica corrente.
Tali report possono essere prodotte in formato stampabile (PDF/RTF) oppure in formato Tabellare (XLS).

## Codici Parlanti

Nelle classi di Geoweb è possibile configurare degli elementi chiamati **MnemonicCode**, o *Codici Parlanti*, i quali vengono usati come riferimento di configurazione dai widget del plugin gwMnemonicCode.

Questo componente viene utilizzato per la definizione di codici gerarchici strutturati a livelli variabili, come ad esempio le WBS.

« Con il termine “codice parlante” si intende un particolare metodo di codifica dei prodotti che consente di individuare direttamente dal codice alcune delle caratteristiche del prodotto. Il codice parlante è strutturato, ovvero presenta uno schema ben definito dove ogni carattere, o gruppo di caratteri, indica una specifica caratteristica dell'articolo.»

[spostare la parte sotto su gwObject] La configurazione di questi elementi è variabile e dipende dal tipo di oggetto a cui il codice stesso si deve riferire, perciò non tutti i parametri seguenti saranno necessari.

**ATTENZIONE!!** per ogni MnemonicCode possono essere configurate varie *McPart*, ognuna relativa ad uno specifico tipo di oggetto. Ogni *McPart* aggiuntiva va posta direttamente sotto la precedente.

I parametri a disposizione sono i seguenti:

```
<MnemonicCode>
  <parts>
    <McPart>
      <partName>
        [nome di questa McPart]
      </partName>
      <partLabel>
        [etichetta da mostrare nel dettaglio del MnemonicCodeWidget]
      </partLabel>
      <lastCharacter>
        [opzionale: carattere separatore tra questa McPart e la seguente]
      </lastCharacter>
      <characterNumber>
        [opzionale: massimo numero di caratteri in questa McPart]
      </characterNumber>
      <required>
        [boolean, indica se questa McPart debba avere obbligatoriamente un
valore o meno]
      </required>
      <includePreviousParts>
        [boolean, indica se il valore di questa McPart include anche quello
delle precedenti]
      </includePreviousParts>
      <initSelValue>
        [opzionale: valore iniziale impostato se questa McPart non ha valore
all'apertura del dettaglio]
      </initSelValue>
      <fillingStrategy>
        [0:classe, 1:progressivo numerico, 2:immissione libera,
3:classificazione]
      </fillingStrategy>
      <regExp>
        [opzionale: espressione regolare utilizzata per la validazione del
valore di questa parte]
      </regExp>
      <regExpGen>
        [opzionale: funzione js che restituisce una stringa rappresentante
```

```
il corpo di un'espressione regolare]
      </regExpGen>
      <placeHolder>
        [opzionale: definisce un consiglio che aiuti l'utente a compilare
questo campo]
      </placeHolder>
      <promptMessage>
        [opzionale: tooltip che appare quando il campo è vuoto ed ha il
focus]
      </promptMessage>
      <invalidMessage>
        [opzionale: tooltip che appare quando il contenuto del campo non è
valido]
      </invalidMessage>
      <missingMessage>
        [opzionale: tooltip che appare quando il campo è vuoto ed il campo è
'required']
      </missingMessage>
      <trim>
        [opzionale: boolean, indica se rimuovere tutti gli spazi vuoti
all'inizio ed alla fine del valore inserito]
      </trim>
      <lowercase>
        [opzionale: boolean, indica se convertire tutti i caratteri del
valore in minuscolo]
      </lowercase>
      <uppercase>
        [opzionale: boolean, indica se convertire tutti i caratteri del
valore in maiuscolo]
      </uppercase>
      <propercase>
        [opzionale: boolean, indica se convertire il primo carattere di ogni
parola in maiuscolo]
      </propercase>
      <readonly>
        [boolean, indica se questa parte può essere modificata o meno]
      </readonly>

      <!-- new 2018.09.12 [TESTARE FUNZIONAMENTO] -->
      <attributeNameReference>
        [opzionale: nome dell'attributo di riferimento, che verrà modificato
ad ogni modifica di questo campo]
      </attributeNameReference>

      <!-- new 2019.08.09 -->
      <filler>
        [opzionale: singolo carattere usato per riempire gli spazi vuoti.
Funziona solo insieme a 'characterNumber']
      </filler>
```

```
      <orderby>
        [opzionale: i valori relativi a questa McPart saranno ordinati in
base al nome dell'attributo in questo parametro]
      </orderby>

      <!-- fillingStrategy 0 or 3 -->
      <targetClassName>
        [nome della classe di riferimento per questa McPart]
      </targetClassName>
      <targetFieldToStore>
        [nome dell'attributo in cui viene salvato il valore del codice di
questa parte]
      </targetFieldToStore>
      <targetFieldToShow>
        [nome dell'attributo in cui viene salvata la descrizione del codice
di questa parte]
      </targetFieldToShow>
      <queryClausole>
        [opzionale: clausola SQL che definisce la condizione con cui
filtrare i valori di questa McPart]
      </queryClausole>
      <!-- fillingStrategy 1 -->
      <zeroPadding>
        [numero di cifre di questa McPart, i cui spazi vuoti saranno
riempiti da zeri]
      </zeroPadding>
      <progRule>
        [espressione della regola usata per calcolare il progressivo
numerico]
      </progRule>
    </McPart>
  </parts>
</MnemonicCode>
```

# Mappe, Planimetrie, Modelli BIM

GeoWeb è in grado di integrare visualizzatori di dati spaziali (mappe), geometrici (planimetrie), e modelli BIM 3D.

## Mappe e Planimetrie

I dati sorgenti degli oggetti geometrici e/o spaziali risiedono nello stesso database in cui sono contenuti tutti gli altri dati dell'applicazione. Le geometrie sono contenuti in campi di tipo geometry, che geoweb tratta in maniera appropriata, disponendo di appositi WIDGET per crearli, modificarli, e visualizzarli.

Naturalmente la visualizzazione strutturata dei dati geometrico/spaziale avviene attraverso l'utilizzo di mappe o disegni planimetrici.

Geoweb di avvale del server di Mappe MapGuide OpenSource per la distribuzione su client sia delle mappe che delle planimetrie 2D. Entrambe possono essere costruite con un apposito strumento di Authoring, quale MapGuide Maestro.

L'interazione con le mappe è garantita da apposite API sviluppate ad hoc e da 'contenitori' di mappe che provvedono tutte le funzionalità per la navigazione 2D e l'editing speditivo degli oggetti geometrici.

Geoweb consente inoltre di aggiungere, a supporto della comprensione degli oggetti gestiti, eventuali sfondi cartografici disponibili su WEB (Google Maps, OpenStreet View, Bing, WMS proprietari erogati da enti pubblici o privati), oppure sfondi in formati vettoriali SDF, generalmente utilizzati per gli oggetti planimetrici (es. planimetrie strutturali).

### Modelli BIM 3D

Un modello BIM 3D consiste in un modello geometrico tridimensionale che permette di raccogliere, combinare e collegare digitalmente, tramite esso, tutti i dati rilevanti di una costruzione.

Geoweb permette la visualizzazione di questo genere di modelli attraverso il modulo BIM Explorer. I dati geometrici dei modelli 3D risiedono nel database MongoDB.

L'interazione con i modelli è garantita da apposite API sviluppate ad hoc e dal modulo BIM Explorer che provvede a tutte le funzionalità per la navigazione 3D e la modifica dei singoli oggetti geometrici.

Inoltre, Geoweb, attraverso una serie di funzionalità che permettono di facilitare la comprensione e la visualizzazione dei modelli gestiti, permette di aggiungere, importare od esportare eventuali annotazioni collegabili sia al modello intero che ad una parte di esso.

## Ambiti

In Geoweb definire un Ambito significa creare una direttiva attraverso la quale i dati provenienti da più fonti di competenza, ma presenti nella stessa istanza applicativa, ovvero nella stessa installazione, possono essere partizionati per essere consultati e gestiti solo da soggetti appartenenti alla singola competenza (Ambito).

Tipicamente (ma non solo) l'ambito viene utilizzato per distinguere competenze territoriali. Ad esempio, se creo una installazione per gestire i clienti di tutta italia, e voglio però affidare i clienti ad agenti commerciali di ZONA, dovrò utilizzare un codice di ZONA da abbinare sia ai singoli clienti che all'agente di competenza.

Per qualificare un Ambito e per riconoscere i dati da selezionare, occorre quindi.

- definire un nome e descrizione per l’Ambito e un valore che lo identifichi;
- inserire tale valore in tutte entità inserite nel sistema che si vogliono far gestire, consultare, in

maniera separata da diversi utenti;
- associare all'Ambito i gruppi di utenti che vi possono accedere;

# Permessi Statici e Dinamici

## Permessi Statici

I permessi di accesso alle Applicazioni consentono di stabilire quali utenti hanno diritto di accedere ai moduli applicativi creati con Geoweb ed a quale ambito fanno riferimento.

I permessi sui dati determinano per ciascun gruppo i diritti sulle classi di dati presenti nel sistema. Per ogni classe si può stabilire sia il tipo di permesso concesso, sia l'ambito di competenza (unità locale, tipo di impianto, reparto, ecc.).

I permessi per ogni classe ed ambito possono essere di 4 tipi tra loro combinabili:

- Inserimento
- Lettura
- Modifica
- Cancellazione

## Permessi Dinamici

I Permessi dinamici aggiungono ulteriori modalità di autorizzazione in Lettura, Modifica e Cancellazione sui dati esistenti, basate sul controllo sul valore di uno o più campi (ad esempio dello stato di un singolo record).

# Eventi

Dalla versione 4.6.x del framework è stato introdotto il concetto di evento.

L'evento (chiamato GeowEvent) è un oggetto interno a Geoweb che ha lo scopo di registrare un'azione svolta dall'utente oppure eseguita automaticamente dal sistema. Questa azione, ovviamente, deve essere in qualche modo importante da registrare: questo vuol dire che potrebbe essere visualizzata dall'utente in uno storico degli eventi e potrebbe essere notificata all'utente o a più utenti tramite un sistema di notifiche.

L'oggetto GeowEvent contiene diversi campi necessari alla registrazione dell'azione che è stata svolta, memorizzando al contempo le informazioni riguardanti l'oggetto sul quale è stata effettuata l'azione. Di fondamentale importanza sono i campi che categorizzano l'evento, cioè **eventCategory** ed **eventSubcategory**. Questi campi vanno compilati in modo coerente (eventi con categoria uguale devono avere il campo eventCategory compilato allo stesso modo). Di seguito viene riportato il codice groovy necessario alla creazione dell'oggetto evento e alla sua registrazione tramite il servizio eventManagementService. Nell'esempio sono riportati, con il loro commento, tutti i campi che compongono l'evento. Si ricorda che **codSolution** e **codModule** sono obbligatori!

GeowEvent.groovy

```
import com.geowebframework.transfer.objects.event.GeowEvent;



GeowEvent event = new GeowEvent();
event.setEventOrigin("user");    //STRINGA - Origine dell'evento: può
assumere solo 2 valori 'user' oppure 'system'
event.setEventUser("esc");     //STRINGA - Utente che crea l'evento.
Deve essere valorizzato solo se eventOrigin=user
event.setEventNotes("Questa e' una nota dell'evento che sto
registrando. "); //STRINGA - Note a corredo dell'evento (max 4000)


event.setEventClassnameObject("cde_deliverable");       //STRINGA -
Nome della classe dell'oggetto su cui si è verificato l'evento
event.setEventPkObject(3619269);            //INTERO - Chiave
dell'oggetto (cioè del record) su cui si è verificato
                                                        //l'evento
event.setEventCodObjectKey("cod_deliverable_full");     //STRINGA -
Nome del campo/attributo della classe dell'oggetto, (campo) che
                                                        //contiene il
codice dell'oggetto su cui si è verificato l'evento
event.setEventCodObjectValue("PRE-001.01_000008_00");   //STRINGA -
Valore del campo/attributo codice dell'oggetto
event.setEventDescrObject("Questa è una descrizione."); //STRINGA -
Descrizione dell'oggetto su cui si è verificato l'evento
event.setEventTypeObjectKey(null);          /*STRINGA - Nome del
campo/attributo della classe dell'oggetto, (campo) che
                                                        contiene il
tipo dell'oggetto. Il tipo dell'oggetto è utile solo
                            nel caso in cui oggetti della stessa classe
ma di tipo diverso debbano avere
                                                        diverse
notifiche o diversi destinatari di notifica*/
event.setEventTypeObjectValue(null);            //STRINGA - Valore del
campo/attributo tipo dell'oggetto


event.setCodSolution("S01");  //STRINGA - Campo obbligatorio!! Codice
della soluzione, recuperabile dalla tabella gwa_solution
event.setCodModule("LLPP_PPM");   //STRINGA - Campo obbligatorio!!
Codice del modulo, recuperabile dalla tabella gwa_module
event.setScopeObject(""); //STRINGA - Variabile di progetto in sessione
event.setEventCategory("CAMBIO STATO");        //STRINGA - Categoria
dell'evento. Va sempre valorizzato!!
event.setEventSubcategory("PUBBLICAZIONE");    //STRINGA -
```

```
Sottocategoria dell'evento. Potrebbe essere nullo se l'evento può
essere
                                                    //descritto solo dalle
categorie.
event.setEventTitle("Cambio stato - pubblicazione");  //STRINGA -
Titolo dell'evento. Se è nullo appare come concatenazione di
                                                        //categoria e
sottocategoria.
event.setEventDescription("Questo elaborato ha cambiato stato: ora e'
pubblicato. ");  //STRINGA - Eventuale descrizione dell'evento
event.setEventMessage("L'elaborato con nome {{nome}} e' stato
pubblicato dall'utente {{user}}."); /*STRINGA - Messaggio legato
                                                    all'evento e alla sua
(futura) notifica. Se e' valorizzato, in fase di
                                                    notifica, l'utente
vedrà il messaggio salvato su questo campo.
                                                    Eventuali variabili
dinamiche possono essere inserite all'interno della
                                                    stringa con la
notazione doppia graffa: {{nome_variabile}}. */

def map = [:];
map.user = "esc";
map.nome = "Elaborato di test PRE-001"

event.setEventFields(map);  /*MAP(STRING,OBJECT) - Mappa che contiene
le variabili dinamiche che sono state inserite
                           nel messaggio.
                           Il messaggio può contenere uno dei campi
dell'evento compilati sopra oppure altri campi aggiuntivi.
                           In questo ultimo caso i campi aggiuntivi
vanno inseriti in questa mappa (con nomi diversi rispetto
                           ai campi sopraelencati) */


services.eventManagementService.recordEvent(event);
```

From:
https://wiki.geowebframework.com/ - **GeowebFramework**

Permanent link:
**https://wiki.geowebframework.com/doku.php?id=gwusermanual:gwobjects:start&rev=1673881728**

Last update: **2023/01/16 16:08**