Elenco generale Widget

Widget Base

Nelle maschere utente vengono utilizzati dei controlli (tecnicamente widget) che sono associati agli attributi delle classi. Ognuno di essi è configurabile attraverso una serie di parametri, che vengono poi salvati e mantenuti in formato xml.
Ogni widget è inoltre codificato con un numero intero, denominato CTRL_TYPE, che viene utilizzato nelle tabelle dei metadati (GWM_ATTRIBUTES).
Le interfacce di amministrazione degli attributi in alcuni casi guidano alla configurazione, ma nella maggior parte dei casi presentano il contenuto dell'XML direttamente modificabile in forma testuale.
In questo paragrafo saranno elencati tutti i widget esistenti nella versione corrente, e descritti i parametri di configurazione attesi.
Esistono dei parametri comuni a tutti i tipi di controllo, che saranno trattati in un'unica soluzione all'inizio del paragrafo.

tipologia	Nome	CTRL_TYPE	Descrizione breve
	ACTION_WIDGET	30	Permette di inserire un pulsante Azione all'interno della form
	ACTIVE_USER_WIDGET	40	Salva nel campo associato il valore dell'Utente attivo nella sessione corrente
	ACTIVE_GROUP_WIDGET	44	Salva nel campo associato il valore del Gruppo attivo nella sessione corrente
	INSERT_USER_WIDGET	48	Salva nel campo associato il valore dell'Utente attivo in fase di Inserimento Record
	EXTERNALLINK	21	Controllo per inserire un link che punta ad un url esterno al sistema
	GW_DIGITAL_DOCUMENT_WIDGET	52	
Gestore Documentale	ATTACHMENTS	26	Controllo per gestire i documenti salvati in una cartella del gestore documentale configurato (es. alfresco). Per utilizzare il widget è necessario che il gestore documentale sia raggiungibile, e che sia stata dedicata una cartella documenti all'applicazione. Il contenuto del campo associato sarà il riferimento alla sottocartella documenti
	CMIS_DOCUMENT_WIDGET	43	Questo controllo punta a un singolo documento contenuto nel gestore documentale configurato (es. alfresco). Il nome del documento può essere definito come concatenazione di stringhe e/o contenuti di campi del DB

tipologia	Nome	CTRL_TYPE	Descrizione breve
Data/Ora	DATE	5	Inserimento di una data attraverso uso del calendario con inserimento opzionale dell'ora
	TIME	45	Inserimento solo dell'ora con widget tipo Spinner

tipologia	Nome	CTRL_TYPE	Descrizione breve
Input guidato	CHECKBOX	6	Casella con segno di spunta. Può assumere 3 valori: True, False, Null
	COLORPICKER	39	Controllo che permette di scegliere un colore, che viene salvato nel campo associato sotto forma di codice Esadecimale (string) tramite la presentazione di una palette grafica
	COMBOBOX	2	Lista di scelta mono-selezione in cui i valori sono definiti internamente al widget
	DBCOMBOBOX	3	Lista di scelta mono-selezione in cui i valori sono recuperati dal database tramite query da configurare
	DBIMAGECOMBOBOX	33	Lista di scelta mono-selezione che presenta, oltre al testo, anche una immagine. I valori sono recuperati dal database tramite query da configurare. Nel database dovranno già essere salvate le immagini da mostrare.
	DBSUGGESTBOX	24	Casella di inserimento testo con funzioni di suggerimento da selezione dei valori presenti in una lista definita
	DBWINDOWLIST	4	Lista di scelta mono-selezione in cui i valori sono recuperati dal database tramite query da configurare. Questo controllo permette di mostrare all'utente più colonne dei record da selezionare. Non richiede che sia definita una classe sulla tabella o vista utilizzata nella query di selezione Viene mostrata con una finestra modale che si apre in pop-up
	DOCUMENT	7	Controllo per gestire l'inserimento un singolo documento, che viene salvato nel DB. Il documento viene selezionato tramite una finestra standard windows
	IMAGE	8	Controllo per gestire l'inserimento un file immagine che viene salvata nel DB Il controllo mostrerà direttamente l'immagine in fase di consultazione o modifica del record corrente nel form L'immagine viene selezionata tramite una finestra standard windows
	IMG_PATH_GALLERY	47	Consente di inserire in una scheda un widget per la visualizzazione di una galleria di immagini
	DOMAIN	28	Controllo per la gestione di tabelle di dominio condivise

tipologia	Nome	CTRL_TYPE	Descrizione breve
Etichette, comunicazione	LABEL	20	Inserisce un testo descrittivo non collegato ad alcun campo del database
	HTML	46	Inserisce un testo descrittivo, collegato a un campo del DB, e che viene renderizzato come HTML
	HTMLTEMPLATE	nn	Inserisce un testo formattato non collegato ad alcun campo del database
Relazioni tra classi	LINKLIST	18	Presentazione di un elenco di record relazionati 1 a N rispetto alla classe corrente Il click sull'elenco darà accesso al record visualizzato nel form di dettaglio La navigazione si sposta alla classe collegata Il record di dettaglio si aprirà come scheda separata, in pop-up o docked, secondo la configurazione già definita per la classe collegata
	LINKLISTNAM	19	Presentazione di un elenco di record relazionati N a M rispetto alla classe corrente Il click sull'elenco darà accesso al record visualizzato nel form di dettaglio La navigazione si sposta alla classe collegata Il record di dettaglio si aprirà come scheda separata, in pop-up o docked, secondo la configurazione già definita per la classe collegata
	EXTERNALTABLE	29	Lista di scelta mono-selezione che permette di selezionare un record presente in una classe definita in GW Viene mostrata con una finestra modale che si apre in pop-up Permette di aggiungere un nuovo record nella classe utilizzata dalla lista
	CHILDLIST	17	Presentazione di una anagrafica relazionata 1 a molti rispetto a quella corrente, inserita nel form come elenco di valori direttamente editabile riga per riga (inserimento, modifica cancellazione)

tipologia	Nome	CTRL_TYPE	Descrizione breve
	MAIN_ACTION_WIDGET	62	Analogamente al widget Action, permette di inserire un pulsante Azione all'interno della form/lista. Questo ha una stilizzazione dinamica che dipende dai valori di alcuni campi dei dati, che determinano label, tooltip, icon, color. I campi che contengono le stilizzazioni del widget, devono essere presenti in lista (tuttalpiù nascosti nell'xml del widget con la regola css display: none; nello style di cella e header) per poter funzionare con il widget in lista
Data input	NORMAL	0	Casella di testo semplice
	NUMBERBOX	25	Casella di testo con controlli dell'input specifici per campi numerici
	TEXTAREA	16	Casella di testo estesa per gestione di campi descrittivi disposti su più righe di testo (xml o note)
	TEXTBOX	1	Casella di testo a cui sono applicabili funzioni di controllo dell'input, come pattern (es. indirizzo email, o codice fiscale) oppure accettare o meno determinati tipi di caratteri
	HTMLEDITOR	55	Editor utile alla scrittura di testo formattato, pensato per il corpo delle email
Geometrie	POINT	13	Inserimento, visualizzazione, zoom su mappa di una geometria di tipo puntuale
	POLYGON	15	Inserimento, visualizzazione, zoom su mappa di una geometria di tipo polilinea
	POLYLINE	14	Inserimento, visualizzazione, zoom su mappa di una geometria di tipo poligonale
Posizione nella mappa o nella planimetria	POSITION_WIDGET	35	
	POSITION2_WIDGET	42	Il widget Position2 permette di memorizzare l'informazione della posizione di un record di una classe. La posizione è rappresentata dal codice di un site o di un building o di una room o di una workstation o di una generic position

tipologia	Nome	CTRL_TYPE	Descrizione breve
JSON	LABEL_VALUE_JSON_WIDGET	66	Il widget permette di visualizzare nel dettaglio di classe le informazioni (come sequenza coppie chiave-valore), presenti in un campo JSON (solo le key del primo livello) **(dalla 4.7.4, issue #1484)**
	PROPERTY_SET_WIDGET	67	Il widget permette di visualizzare nel dettaglio di classe le informazioni (come sequenza coppie chiave-valore), presenti in un campo JSON (solo le key del primo livello). A differenza del LABEL_VALUE_JSON_WIDGET i metadati non stanno nel json stesso, ma vengono recuperati in base alla Definizione della Famiglia Attributi attiva presente nel dettaglio in quel momento.

Widget Enterprise

I widget sotto elencati sono disponibili nella versione Enterprise Edition, e sono compresi in appositi plugin, eventualmente forniti con tale versione.

Plugin	Nome	CTRL_TYPE	Descrizione breve
gw-advanced	FINAL_BALANCE_JOBS_WIDGET	104	Controllo che presenta, in forma di matrice, le Attività legate agli Oggetti. E' utilizzato per spuntare quello che viene effettivamente fatto durante un intervento di manutenzione Utilizza istanze di classi specifiche
	SELECTION_LIST_WIDGET	102	Meccanismo di Selezione che permette di selezionare gli Oggetti presentandoli in base alle tipologie e permettendone il filtro nel rispetto delle gerarchie fissate nel modello dati SmartObjs Utilizza istanze di classi specifiche
	SELECTION_LIST2_WIDGET	105	Versione 2 dello widget SELECTION_LIST_WIDGET. Permette di configurare in maniera libera le gerarchie tra gli oggetti, e mostra, oltre all'elenco degli oggetti da selezionare, anche l'albero gerarchico da scorrere
	LABEL_VALUES_XML_WIDGET	103	Il controllo renderizza in un piccolo form che viene aperto in pop-up il contenuto XML del campo associato
gw-workflow	ACTION_WORKFLOW_WIDGET	32	Inserisce un pulsante Azione in una form di processo
	GW_RELATED_PROCESSES_WIDGET	302	Utilizzato per avviare le istanze di processo e visualizzare qualle in corso e quelle concluse
	GW_ARCHIVED_PROCESSES_WIDGET	301	Utilizzato per mostrare una griglia contente tutte le istanze di processo che si sono concluse e che hanno riguardato l'entità collegata
	GW_TASKS_WIDGET	300	Utilizzato per la gestione dei task

Plugin	Nome	CTRL_TYPE	Descrizione breve
gw-classification	CLASSIFICATION	27	Utilizzato per le classi 'CLASSIFICATE', permette di selezionare e/o visualizzare la famiglie di appartenenza con una visualizzazione che ne presenta la gerarchia definita
	VARIABLEATTRIBUTES	36	Utilizzato come sopra, renderizza gli attributi variabili della classe all'interno del form
	RELATION_WIDGET	31	Mostra i record correlati e stabiliti sugli attributi variabili delle classi 'CLASSIFICATE'
gw-3d-visualizer	GRAPHIC_LAYOUT_WIDGET	50	Utilizzato per le classi che contengono riferimenti ai valori layout_code o pk_layout. Permette l'apertura di una mappa a partire da essi, semplicemente cliccando su di esso.
	INDIRECT_LOCALIZATION	201	Definisce un pulsante che permette di selezionare e fare lo zoom sull'oggetto al quale è associato.
	3D_ZOOM_WIDGET		- deprecato -
gw-mnemonic-code	MNEMONIC_CODE_WIDGET	1000	Utilizzato per la creazione di codici parlanti
	MNEMONIC_CODE_SELECTION_WIDGET	1001	Utilizzato per la selezione di un record da un albero gerarchico di codici parlanti
gw-cde	CONTENT_HANDLER	600	Widget per la gestione dei contenuti: utilizzato per poter consultare e/o manipolare le informazioni relative a contenuti documentali informativi, raccolti e catalogati
gw-cms	JSON_FILE_REST	656	Scheda in sola lettura dal contenuto di un file .json del documentale

Matrice applicabilità tipo widget per tipo di attributo

(pagina da aggiornare)

Parametri degli Widget

Elenco dei Parametri comuni alla maggior parte degli widget

(torna a elenco widget)

Nome	Descrizione breve
width	Larghezza in pixel del controllo visualizzato nel form. Per alcuni controlli, come la ChildList, la LinkList, LinkListNaM, ecc, che richiedono uno spazio importante nel form, questo valore può essere definito, come '-1'. In questo caso il controllo occupa tutto lo spazio disponibile nella porzione di layout in cui è inserito il gruppo attributi in cui si trova il controllo stesso. Per utilizzare questo valore è importante tuttavia che il gruppo attributi non contenga altri controlli
height	Altezza in pixel del controllo visualizzato nel form. Vale quanto detto sopra per i controlli di tipo ChildList, LinklList, LinkListNaM, ecc.
defaultValue	Valore di default che compare solo in fase di inserimento
maxLength	Lunghezza massima, in caratteri, della stringa da inserire/modificare
required	Impostabile a True o False per definire se l'attributo sia Obbligatorio nel form corrente. Nel caso sia impostato a True, non è possibile chiudere la maschera finché l'attributo non viene valorizzato
readonly	Impostabile a True o False per definire se l'attributo debba essere visualizzato in modalità di sola lettura
disabled	Disabilita lo widget nella form
hideRelations	Nasconde l'icona della Relazione, quando lo widget è usato nelle relazioni. Per default è impostato a 'false'
intermediateChanges	Applicabile a Normal, TextBox, TextArea, NumberBox. Definisce se gli eventi associati al widget (Event Handler) debbano scattare alla perdita del focus (false) oppure alla modifica di ogni singolo carattere all'interno della stringa o del numero (true). Per default è impostato a 'false'
listWidth	Larghezza del widget in lista (px)
isDefaultOrderBy	Impostabile a True o False per definire se l'attributo debba essere utilizzato per ordinare la lista dei record presenti nella classe Per default è impostato a True, che significa che viene utilizzato il criterio standard di GW, applicato a tutti gli attributi. Occorre quindi impostare il parametro a False per 'sganciarlo' da tale criterio (1)
isDefaultOrderByAscending	Impostabile a True o False per definire se l'attributo debba essere utilizzato per ordinare la lista dei record in ordine crescente (Ascending) Per default è impostato a True, che significa che viene utilizzato il criterio standard di GW, applicato a tutti gli attributi. Occorre quindi impostare il parametro a False per 'sganciarlo' da tale criterio (1)

Nome	Descrizione breve
isDefaultOrderByOnFieldToShow	Impostabile a True o False per definire se l'attributo debba essere utilizzato per ordinare la lista dei record in base al parametro FieldToShow definito. Per default è impostato a True, che significa che viene utilizzato il criterio standard di GW, applicato a tutti gli attributi. Occorre quindi impostare il parametro a False per 'sganciarlo' da tale criterio (1) Applicabile a controlli di tipo Elenco (DBCombobox, DbWindowList, ecc.)
listCellTextAlign	Imposta in modo rapido l'allineamento del valore del widget dentro la cella. Sovrascrive le impostazioni di base che avrebbe il widget (es: 'center' è il default degli widget 'Date' in lista) (2) Valori ammessi: 'center', 'left', 'right'
listCellStyleRules	Le regole css vengono applicate alle celle che ospitano il widget (2) Valori ammessi: qualsiasi regola css
listCellHeaderStyleRules	Le regole di stile vengono applicate all'header della colonna del widget (2) Valori ammessi: qualsiasi regola css
listCellClass	Le regole css definite dalla classe vengono applicate alle celle. La regola css deve essere già esistente fra quelle base (o essere stata in qualche modo aggiunta tramite i meccanismi dei plugin) (2) Valori ammessi: qualsiasi nome di classe css
importCSVWithoutDecoding	Utilizzabile negli widget di Input guidato, ovvero per i quali il valore mostrato è diverso da quello salvato su DB, come nel caso delle DBComboBox, nelle ExternalTable, ecc., stabilisce se, nel caso di importazione via csv configurata su Geoweb, deve essere interpretato il valore della colonna cosi come mostrato o cosi come realmente salvato su DB. Il default a 'false' significa che normalmente il valore viene interpretato, nel CSV, come decodificato.
toTranslate	usato nelle Liste di Selezione valori. Se impostato a true, le Etichette impostate come <values> potranno essere tradotte dal Dizionario, che ovviamente dovrà contenere le etichette e le relative traduzioni

(1) Nella modalità di visualizzazione Lista della classe, GW ordina normalmente in modo crescente (Ascending) utilizzando i primi 3 attributi presenti in lista. Usando i parametri isDefaultOrderBy... è possibile agire su questo criterio, definendo in autonomia per quali attributi debba essere ordinabile la lista, se crescente o decrescente, ecc.
(2) Gli attributi vengono presentati in Lista secondo dei criteri di stilizzazione standard GW. In particolari esigenze può essere utile renderizzare una colonna in maniera diversa dalle altre. Usando i parametri listCell... è possibile agire in tale senso. Ulteriori spiegazioni sono disponibili al paragrafo Personalizzazione stile dei controlli in Lista

Espressioni nei Widget (sintassi)

Alcuni parametri dei Widget di tipo Selezione da una lista, possono utilizzare espressioni lato client, ovvero prevedere che il contenuto delle liste presentate dipenda, ad esempio, da un settaggio di un altro attributo.

La sintassi da utilizzare per riferirsi ad un altro attributo, che deve essere necessariamente definito

nella classe e inserito in un gruppo attributi, anche se in modalità *hidden*, è la seguente:

cod_service_category=#{cod_service_category}

Configurazione Widget Base

ACTION_WIDGET

(torna a elenco widget)

Definisce un pulsante, collocabile all'interno del form layout, al quale associare una Action definita a livello di classe.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
actionName	nome della Action definita a livello di classe
useColSpan	Impostato a True, il pulsante compare senza l'etichetta definita a livello di Attributo, e allineato con le altre etichette del form. Impostato a False, l'etichetta viene stampata, e il pulsante viene allineato in corrispondenza della colonna delle altre caselle dei controlli. default=*true*
alignment	Definisce l'allineamento del pulsante rispetto alla colonna sopra definita. default=*left*

ACTIVE_GROUP_WIDGET

(torna a elenco widget)

Il controllo permette di salvare nel campo associato il valore del Gruppo attivo nella sessione corrente.

Non ci sono parametri oltre ai parametri comuni.

E' importante tenere presente che lo widget funziona correttamente SOLO se la classe viene movimentata attraverso l'interfaccia Utente di Geoweb. Non funziona, ad esempio, con il metodo API 'UpdateClassRecord'.

ACTIVE_USER_WIDGET

(torna a elenco widget)

Il controllo permette di salvare nel campo associato il valore dell'Utente attivo nella sessione corrente.

Non ci sono parametri oltre ai parametri comuni.

E' importante tenere presente che lo widget funziona correttamente SOLO se la classe viene movimentata attraverso l'interfaccia Utente di Geoweb. Non funziona, ad esempio, con il metodo API 'UpdateClassRecord'.

INSERT_USER_WIDGET

(torna a elenco widget)

Il controllo permette di salvare nel campo associato il valore dell'Utente che esegue la transazione di Inserimento del record.
Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
type	tipologia dell'attributo. Può assumere due valori: *insert*: salva lo username dell'utente che esegue l'inserimento *update*: salva lo username dell'utente che esegue la modifica, a ogni modifica che viene effettuata

E' importante tenere presente che lo widget funziona correttamente SOLO se la classe viene movimentata attraverso l'interfaccia Utente di Geoweb. Non funziona, ad esempio, con il metodo API 'InsertClassRecord'.

EXTERNALLINK

(torna a elenco widget)

Controllo per inserire un link che punta ad un url esterno al sistema. L'URL sarà aperto in una pagina nuova.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
url	URL valido da raggiungere
label	etichetta utilizzata nel form come hyperlink

GW_DIGITAL_DOCUMENT_WIDGET

(torna a elenco widget)

L'attributo gwDigitalDocumentWidget è un componente di Geoweb complesso, che consente di gestire gli attributi di una classe in modalità Documento.

La definizione di Documento sottende alla gestione di informazioni che appartengono ad una singola entità, ma che possono essere inserite in momenti diversi e da utenti diversi.

Il componente permette di organizzare i dati di una singola anagrafica separandoli in sezioni, in modo che ogni sezione possa essere gestita in maniera autonoma, sia come modalità di rappresentazione che come livelli autorizzativi, che come comportamenti, legando questi aspetti allo Stato dell'entità e all'Utente che vi accede.

Il widget è illustrato in maniera più dettagliata a questa pagina

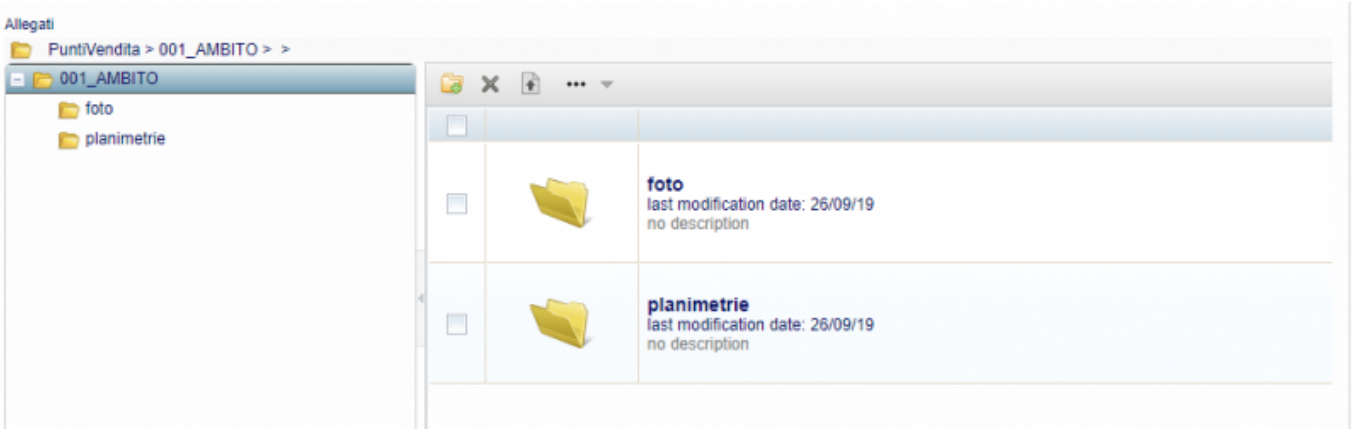
Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
templateName	nome del template definito a livello di componente Documento Digitale (vedi descrizione)

ATTACHMENTS

(torna a elenco widget)

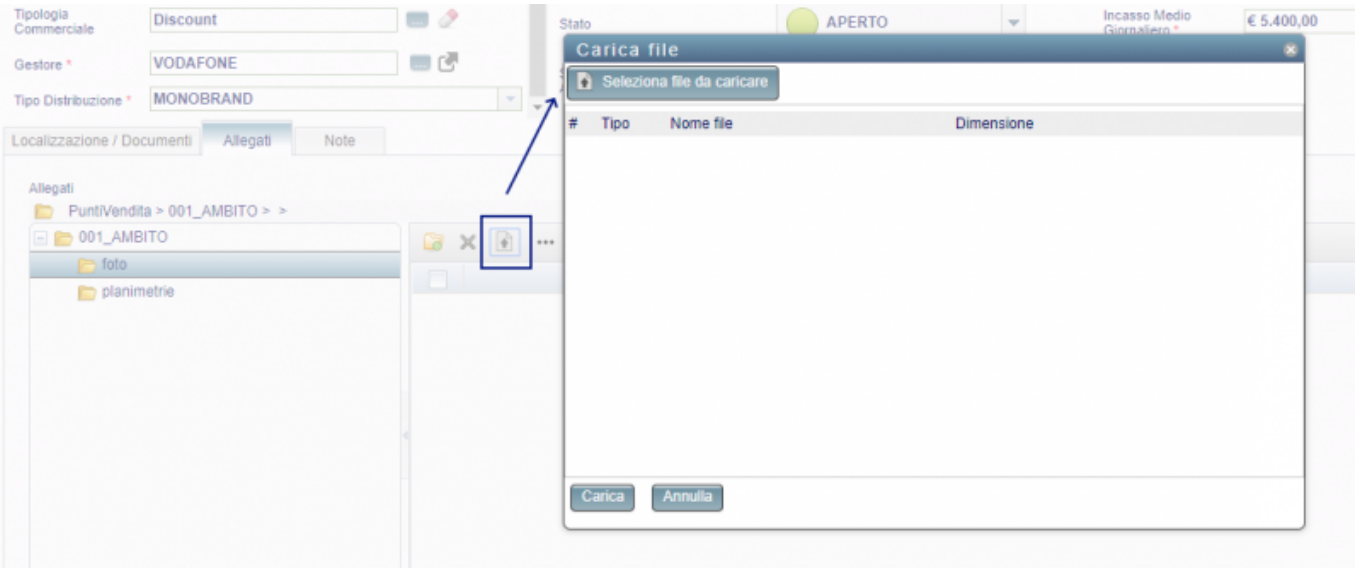
Controllo per allegare al record un insieme di documenti, che vengono salvati in una cartella del gestore documentale configurato (es. alfresco).



Per utilizzare il widget è necessario che il gestore documentale sia raggiungibile, e che sia stata dedicata una cartella documenti all'applicazione.
Il widget non è associato ad alcun campo della anagrafica collegata alla classe. Tuttavia i documenti rimarranno associati al record corrente.
Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
path	Percorso relativo alla cartella root che viene definita nel 'configuration properties'. Il percorso può essere parametrico, e può contenere espressioni compatibili con geoweb. Es. /PuntiVendita/${cod_punto_vendita}/ Se non esiste, il percorso viene creato al momento in cui si accede alle funzionalità dello widget
folderName	eventuale nome della cartella da creare sotto il percorso definito in path
createRootFolder	se impostato a true il widget ricrea una struttura di cartelle definite nel gestore documentale (vedi parametro sotto)

Parametro	Descrizione
templateFolderPath	percorso di una struttura di cartelle definite nel gestore documentale e da utilizzare come template per la generazione automatica della struttura di cartelle relativa al record corrente Es. /templatePUNTOVENDITA
allowNewFolder	se impostato a true permette la creazione, da client, di nuove cartelle
pattern	stabilisce un pattern attraverso cui filtrare i files da caricare
multipleUpload	se impostato a true, permette il caricamento di files multipli



CMIS_DOCUMENT_WIDGET

(torna a elenco widget)

Questo controllo punta a un singolo documento contenuto nel gestore documentale configurato (es. alfresco). Il percorso del documento sarà sempre relativo al percorso definito nel configuration.properties, alla voce “cmisBasePath”. Tale percorso relativo può essere definito da una espressione, come una concatenazione di stringhe e/o contenuti di campi del DB `(es. /Punti Vendita/${cod_punto_vendita}/Allegati/).`
Possono inoltre essere definiti due comportamenti:

1. Il documento caricato viene salvato in CMIS con il suo **nome originale**. Tale *nome* viene inoltre salvato nell'attributo a cui viene agganciato il widget.
2. Il documento caricato viene **rinominato**, al momento dell'upload, **con un nome definito in un campo, configurato come attributo**, che deve essere precompilato, o come default value, o nel trigger della classe, nella sezione 'beforeInsert'. In questo caso non devo agganciare il widget a un campo del DB, ma devo configurare nel parametro nameField il nome del campo (FISICO, non calcolato sulla Vista) da cui ottenere il nuovo nome. Il documento prenderà questo nome anche su CMIS.

Posso inoltre configurare attributi su cui salvare content_type e autore.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
path	percorso, relativo al cmisBasePath definito nel configuration.properties, su cui vengono salvati i documenti all'interno del CMIS. Tale percorso può essere definito da una espressione, anche basata su altri attributi della classe. (es. /Punti Vendita/${cod_punto_vendita}/Allegati/)
nameField	nel comportamento di tipo 2 (vedi sopra), ovvero con Rinomina del file che viene caricato, questo parametro va configurato con il nome dell'attributo che contiene il nuovo nome che il documento assumerà. Se il comportamento è di tipo 1, questo parametro va lasciato vuoto
contentTypeField	opzionale, nome dell'attributo su cui può essere salvato il content type del documento
authorField	opzionale, nome dell'attributo su cui può essere salvato l'autore del documento, letto dai metadati del documento

DATE

(torna a elenco widget)

Inserimento di una data attraverso uso del calendario con inserimento opzionale dell'ora.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
format	Formato della data, secondo le specifiche seguenti https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html
formatTime	Formato dell'ora, secondo le stesse specifiche di cui sopra
handleTime	Stabilisce se mostrare e gestire anche l'ora

TIME

(torna a elenco widget)

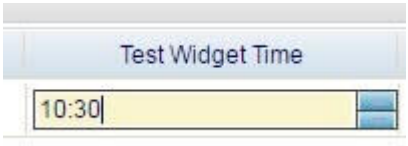
Questo tipo di controllo permette di gestire (visualizzare, inserire, modificare) un tipo di dato ora. Sul DataBase tale dato verrà comunque sempre persistito come Date (sempre comprensiva di ora, importante).

Il widget è implementato sia sul dettaglio della classe



, che su gwClassList/childList in visualizzazione





che su childList in editazione

Quando il record è in visualizzazione o readonly verrà semplicemente mostrata una stringa rappresentante la data formattata. Il widget è implementato con una casella di testo, della tipologia Spinner, che permesse la gestione del dato. Questo widget supporta multiple modalità di inserimento. La prima è la modalità libera da tastiera: l'utente può inserire una stringa che rispetti il pattern impostato in timePattern. La seconda modalità è agire sui tastini dello Spinner feccia su/freccia giu, che rispettivamente incrementato/decrementato l'ora corrente di un valore definito dalla coppia smallDelta/deltaUM del widget. Per esempio con smallDelta=15 e deltaUM='minute', ad ogni click su freccia su aumenterò la data corrente di 15 minuti. Una terza modalità è, quando il widget ha il focus, usare i tasti freccia su/freccia giu, che hanno effetti del tutto uguali a quelli dei tastini dello Spinner. La quarta modalità è agire, quando il mouse staziona sopra il widget, con la rotellina del mouse: ad ogni scatto della rotellina equivale un incremento/decremento di entità smallDelta/deltaUM. Inoltre è possibile incrementare/decrementare l'ora di un delta di entità largeDelta/deltaUM, quando il widget ha il focus, agendo rispettivamente sui tasti Page Up/Page Down della tastiera.

Il widget deriva direttamente dal widget WidgetBase, e ne eredita tutti i parametri:

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

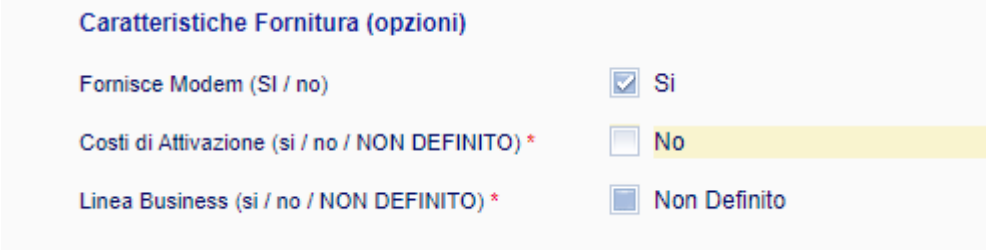
Parametro	**Descrizione**
timePattern	String, default 'HH:mm', pattern usato per mostrare l'ora
smallDelta	Integer (>0), default 15, delta minimo per il quale l'ora del widget può essere incrementato/decrementato. Lavora in congiunzione con il parametro deltaUM. E' comandato, quando il widget ha il focus, indifferentemente dai tastini dello Spinner, dai stati freccia su/freccia giu della tastiera e dalla rotella del mouse.
largeDelta	Integer (>0), default 15, delta minimo per il quale l'ora del widget può essere incrementato/decrementato. Lavora in congiunzione con il parametro deltaUM. E' comandato, quando il widget ha il focus, indifferentemente dai tastini dello Spinner, dai stati freccia su/freccia giu della tastiera e dalla rotella del mouse.
deltaUM	Integer (>0), default 60, delta massimo per il quale l'ora del widget può essere incrementato/decrementato. Lavora in congiunzione con il parametro deltaUM. E' comandato, quando il widget ha il focus, dai stati page su/page giu della tastiera. possibili valori:hour, minute, second, millisecond
promptMessage	String, default 'format: '+timePattern, messaggio che viene mostrato all'utente quando si trova, a widget senza ora, ad inserire un input tramite tastiera

---

CHECKBOX

(torna a elenco widget)

Casella con segno di spunta. Può assumere i significati: True, False e con la configurazione 'Gestisci 3 Stati' anche il Null.
Tali valori possono essere espressi attraverso stringhe (SI/NO, VERO/FALSO, ecc.) o numeri (0/1) e rappresentati in ogni caso attraverso etichette a scelta del configuratore.



Gestire il terzo stato significa definire che l'utente possa scegliere di NON EFFETTUARE una scelta, perché magari non conosce la risposta. Se non viene utilizzato bisogna necessariamente definire un valore di Default, per non confondere il FALSE con il NULL.

IMPORTANTE: per applicare un valore di default è necessario intervenire, oltre che sulla configurazione dell'attributo, anche nella **definizione del campo su DB**. Ci deve essere sempre allineamento, altrimenti il widget assume comportamenti non corretti.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
handleThreeState	Gestisce i 3 Stati, permette quindi di inserire il valore NULL e di abbinargli una etichetta
typeValue	Tipo di valore (di solito true/false)
labels	Etichette delle opzioni
defaultValue	sono ammessi i seguenti valori: checked, notChecked, mixed (solo se handleThreeState=true). Come detto sopra, il valore di default deve essere impostato ANCHE nella definizione del campo su DB, e va mantenuto sempre allineato

---

COLORPICKER

(torna a elenco widget)

Controllo che permette di scegliere un colore, che viene salvato nel campo associato sotto forma di codice Esadecimale (string) tramite la presentazione di una palette grafica.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
handleOpacity	Mostra lo strumento per opacizzare il colore, se impostato a true
saveAsHexStringOnDB	Salva il colore con il codice esadecimale nel database se impostatoa a true

---

COMBOBOX

(torna a elenco widget)

Lista di scelta mono-selezione in cui i valori sono definiti internamente al widget.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
values	Elenco dei valori (etichette mostrate)
keys	Elenco delle chiavi
initSelValue	Valore di default

DBCOMBOBOX

(torna a elenco widget)

Lista di scelta mono-selezione in cui i valori sono recuperati dal database tramite query da configurare.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

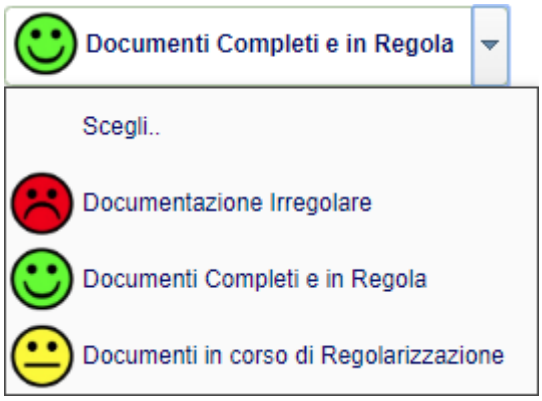
Parametro	Descrizione
labelWidth	Larghezza in px dell'etichetta del widget
strQuery	Istruzione sql di select, scritta in chiaro. La select deve comprendere il campo che sarà usato nel parametro <fieldToStore> e nel parametro <fieldToShow>. L'istruzione non dovrà comprendere la clausola where, che potrà opzionalmente essere inserita nel parametro <queryClausole> (vedi sotto)
queryClausole	Clausola where che può essere utilizzata come filtro. La parola 'where' va omessa. Può accettare delle espressioni
fieldToShow	Campo da mostrare
fieldToStore	Campo da salvare
initSelValue	Valore di default

DBIMAGECOMBOBOX

(torna a elenco widget)

Lista di scelta mono-selezione che presenta, oltre al testo, anche una immagine. I valori sono recuperati dal database tramite query da configurare. Le immagini da mostrare nel controllo dovranno già essere salvate nel database, nella stessa tabella da cui leggere i valori da mettere in lista.

Sarà buona norma utilizzare immagini con stessa larghezza e altezza (es. 50×50 px), e che abbiano le stesse dimensioni tra loro.



Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
strQuery	Istruzione sql di select, scritta in chiaro. La select deve comprendere il campo che sarà usato nel parametro <fieldToStore>,il campo che sarà utilizzato nel parametro <fieldToShow> e il campo che sarà utilizzato nel parametro <blobField>, in quest'ordine. L'istruzione non dovrà comprendere la clausola where, che potrà opzionalmente essere inserita nel parametro <queryClausole> (vedi sotto)
queryClausole	Clausola where che può essere utilizzata come filtro. La parola 'where' va omessa. Può accettare delle espressioni
fieldToShow	Campo da mostrare
fieldToStore	Campo da salvare
initSelValue	Valore di default
blobField	Campo binario in cui è salvata l'immagine
contentTypeField	Campo nel quale viene salvato il content type dell'immagine (dal widget Image, per esempio). Quando configurato, permette di far gestire anche immagini di tipo **.SVG (solo dalle versioni: 4.7.2, 4.6.17; issue #1378)**
cssClassField	Campo nel quale è salvato il set di regole css usato per l'immagine. Quando configurato, ha priorità su blobField+contentTypeField. Sono supportate: regole css FontAwesome, regole css Geoweb, regole css di altre librerie importate, B64 generabili dal tool del webadmin (comprese icone personalizzate nello stile, stacked, etc..) **(solo dalle versioni: 4.7.2, 4.6.17; issue #1379)**
showValueToShowInList	se impostato a *true*, mostra l'immagine e il valore contenuto nel campo <fieldToShow>, se impostato a *false* visualizza solo l'immagine

DBSUGGESTBOX

(torna a elenco widget)

Casella di inserimento testo con funzioni di suggerimento da selezione dei valori presenti in una lista definita. La selezione può avvenire sulla stessa classe e sullo stesso campo che si sta configurando.

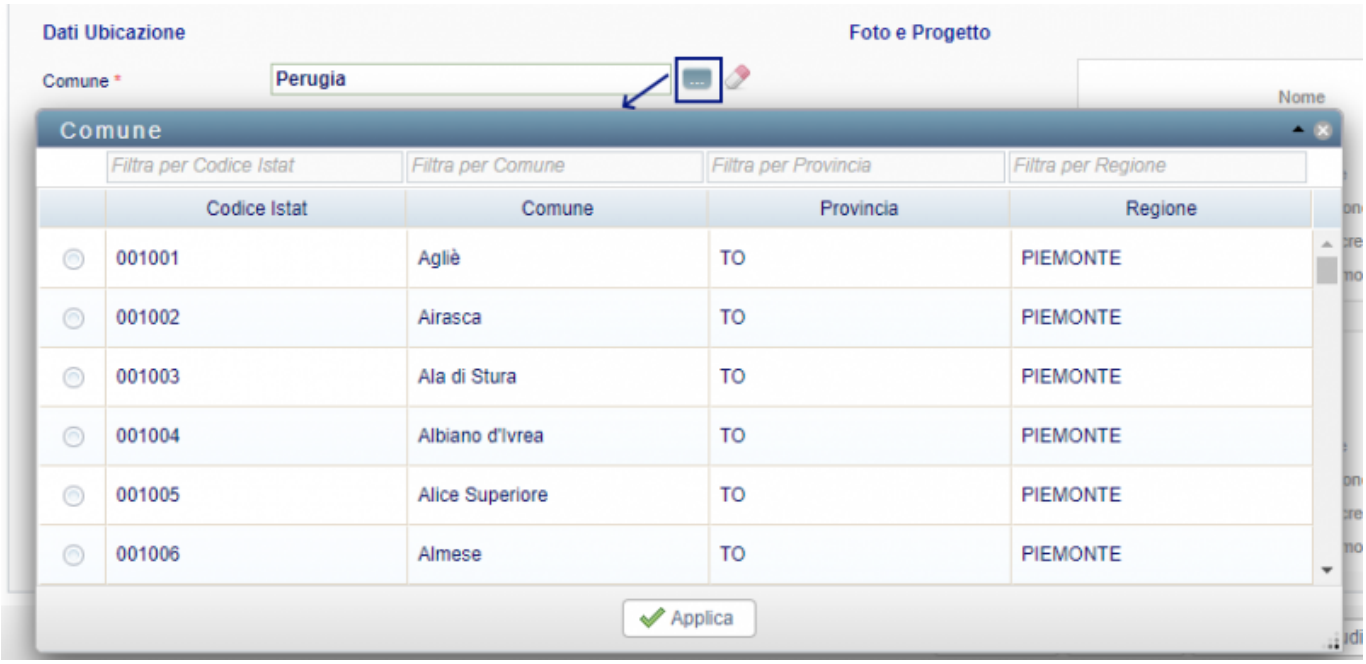
Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
strQuery	Istruzione sql di select, scritta in chiaro. La select deve comprendere il campo che sarà usato nel parametro <fieldToShow>. L'istruzione non dovrà comprendere la clausola where, che potrà opzionalmente essere inserita nel parametro <queryClausole> (vedi sotto). Se la selezione opera nello stesso campo che si sta configurando, sarà utile utilizzare lo statement 'distinct' nella istruzione di select (Es. *select distinct marca from prodotti*)
queryClausole	Clausola where che può essere utilizzata come filtro. La parola 'where' va omessa. Può accettare delle espressioni
fieldToShow	Campo da mostrare
initSelValue	Valore di default

DBWINDOWLIST

(torna a elenco widget)

Elenco di scelta mono/multi selezione in cui i valori sono recuperati dal database tramite query da configurare. Questo controllo permette di mostrare all'utente più colonne dei record da selezionare. Viene mostrata con una finestra modale che si apre in pop-up.



Non richiede che sia definita una classe sulla tabella o vista utilizzata nella query di selezione. Per questo motivo occorre configurare nei parametri le intestazioni di colonna.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
strQuery	Istruzione sql di select, scritta in chiaro. La select deve comprendere tutti i campo che saranno visualizzati nell'elenco, e anche i campi usati nei parametri <fieldToStore> e <fieldToShow>. L'istruzione non dovrà comprendere la clausola where, che potrà opzionalmente essere inserita nel parametro <queryClausole> (vedi sotto)
queryClausole	Clausola where che può essere utilizzata come filtro. La parola 'where' va omessa. Può accettare delle espressioni
fieldToShow	Campo da mostrare
fieldToStore	Campo da salvare
initSelValue	Valore di default
columnsLabel	Elenco delle etichette delle colonne in elenco
columnsView	Elenco delle colonne visualizzate in elenco
columnsType	Elenco dei tipi delle colonne in elenco
allowMultipleSelection	Permette la selezione multipla, se impostato a true
multipleSelectionSeparator	Separatore delle opzioni della selezione multipla

DOCUMENT

(torna a elenco widget)

Controllo per gestire l'inserimento un singolo documento, che viene selezionato tramite una finestra standard Windows. Il documento selezionato può essere memorizzato su un campo del DB oppure su File System.

Il controllo va configurato sul campo stringa destinato a contenere il NOME del documento.
In particolare, se si vuole salvare il documento su DB, dovranno essere definiti tre campi:

- un campo stringa che conterrà il nome del documento, e sul quale viene agganciato e configurato il widget Document
- un campo binario, che conterrà il documento nel suo formato originale;
- un campo stringa, che conterrà il content-type del documento, per informare Geoweb riguardo a quale applicazione potrà essere utilizzata per riaprirlo da client.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
blobField	nome del campo binario su cui salvare il documento
contentTypeField	nome del campo stringa su cui viene salvato il content-type
mode	modalità di salvataggio del documento. 1=salvataggio su DB (default), 0=salvataggio su File System

fsPath	se mode=0, percorso di salvataggio su FS.
	Se lasciato vuoto, il file viene salvato nella cartella dei contenuti statici 'documents', con la sintassi seguente:
	`"documents/" + classname + "/" + filename`
	. Se inserito, è trattato come percorso assoluto. In questo caso la sintassi è la seguente:
	`fsPath + "/" + filename`
	. Il parametro può contenere anche espressioni `${}`.

Parametro	Descrizione
blobTable	*deprecato*
valueCtrAct	*deprecato*
labelDet	*deprecato*
labelEdit	*deprecato*
labelDel	*deprecato*
downloadURL	*deprecato*

Note

1. Nel field del document, ci deve essere solo il nome, comprensivo di estensione. Non ci devono essere sub-path espressi con **/**, che verrebbero mal interpretati dal Controller Spring. Il path, e la sua eventuale costruzione dinamica, deve essere spostato in **fsPath**, anche ricorrendo ad EL.
2. Nel caso di path assoluti iniziare il *fsPath* con

```
file:///
esempi:
file:///C:/...
file:///opt/...
```

3. In caso di folder sul file system, ci devono essere i permessi per le folder interessate.
4. In caso di fsPath costruiti con `${}`, con widget in lista, bisogna assicurarsi che i campi utilizzati per la risoluzione siano presenti in lista. Al limite anche nascosti settando

```
display: none;
```

   nei tag dell'xml del widget <listCellCssRules> <listHeaderCssRules>

IMAGE

(torna a elenco widget)

Controllo per gestire l’inserimento un file immagine che viene salvata nel DB Il controllo mostrerà direttamente l’immagine in fase di consultazione o modifica del record corrente nel form L’immagine viene selezionata tramite una finestra standard windows

Controllo per gestire l’inserimento un file immagine, che viene selezionato tramite una finestra standard Windows. Il file immagine selezionato può essere memorizzato su un campo del DB oppure su File System.

Il controllo va configurato sul campo stringa destinato a contenere il NOME dell'immagine.
In particolare, se si vuole salvare il file immagine su DB, dovranno essere definiti tre campi:

- un campo stringa che conterrà il nome dell'immagine, e sul quale viene agganciato e configurato il widget Document
- un campo binario, che conterrà l'immagine nel suo formato originale;
- un campo stringa, che conterrà il content-type dell'immagine, per informare Geoweb riguardo a

quale applicazione potrà essere utilizzata per riaprirla da client.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
blobField	nome del campo binario su cui salvare l'immagine
contentTypeField	nome del campo stringa su cui viene salvato il content-type
mode	modalità di salvataggio del file immagine. 1=salvataggio su DB (default), 0=salvataggio su File System
fsPath	se mode=0, percorso di salvataggio su FS. Se lasciato vuoto, il file viene salvato nella cartella dei contenuti statici 'documents', con la sintassi seguente: “documents/” + classname + “/” + filename Se inserito, è trattato come percorso assoluto. In questo caso la sintassi è la seguente: fsPath + “/” + filename Il parametro può contenere anche espressioni.
blobTable	*deprecato*
valueCtrAct	*deprecato*
labelDet	*deprecato*
labelEdit	*deprecato*
labelDel	*deprecato*
showLabelInDetail	stabilisce se deve essere visualizzata o meno l'etichetta dell'attributo; default: false
disableExpandToFullSize	se impostato a *true* disabilita la proprietà dell'immagine di essere visualizzata nelle sue dimensioni originali, in pop-up, qualora venga cliccata su client. Per default il parametro è impostato a *false*

IMG_PATH_GALLERY

(torna a elenco widget)

Il widget ImgPathGallery consente di inserire in una scheda un visualizzatore di una galleria di immagini. Le immagini sono recuperate tramite un percorso assoluto nel File System.

Il widget è illustrato in maniera più dettagliata a questa pagina.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Tipo	Required	Valori Ammessi	Default	Descrizione
dataSource	String	true	['FS', 'REST', 'CMIS', 'DB']	'FS'	Determina la tipologia di sorgente per le immagini: File System, Documentale CMS, Documentale CMIS (@Deprecated), o Database (@NotImplemented)

Parametro	Tipo	Required	Valori Ammessi	Default	Descrizione
fsPath	String	false	*		Path dove verranno salvate le immagini. E' possibile usare una o più notazioni *${columnName}* in ogni punto della stringa. Ognuna di esse verrà valutata e sostituita con il valore corrente del record a cui fa riferimento l'attributo
regEx	String	false	*		Espressione regolare. Permette di filtrare il nome delle immagini che si possono visualizzare
orderBy	String	false	['name', 'date']	'name'	Permette di ordinare la galleria e visualizzare le immagini in base al nome oppure in base alla data di creazione dell'immagine
asc	Boolean	false	[true, false]	true	Permette di scegliere se effettuare l'ordinamento (tramite il parametro orderBy) in modo ascendente (*true*) o decrescente (*false*)
imgPerPage	Integer	false	*	7	Permette di selezione quante anteprime visualizzare (thumbnail)
timeInterval	Integer	false	*	3000	Espresso in ms. Permette di configurare l'intervallo di tempo nel cambio di immagine nella modalità presentazione, parametro definito in millisecondi
themeDark	Boolean	false	[true, false]	false	Permette di scegliere se utilizzare il tema chiaro (false) oppure se utilizzare il tema scuro(true)
zipNamePattern	String	false	*	'archivio.zip'	Permette di configurare il nome del file .zip quando si effettua il download dell'intera galleria. Anche qui è possibile usare una o più notazioni *${columnName}* in ogni punto della stringa. Ognuna di esse verrà valutata e sostituita con il valore corrente del record a cui fa riferimento l'attributo. L'estensione .zip verrà aggiunta in automatico
showDescriptionArea	Boolean	false	[true, false]	true	Viene valutato solo se dataSource vale 'CMIS'. Quando è posto a true viene visualizzata una specifica area per mostrare (e, se readonly vale true, editare), la property 'description' del documentale

DOMAIN

(torna a elenco widget)

Controllo per la gestione di tabelle di dominio condivise tra più anagrafiche.

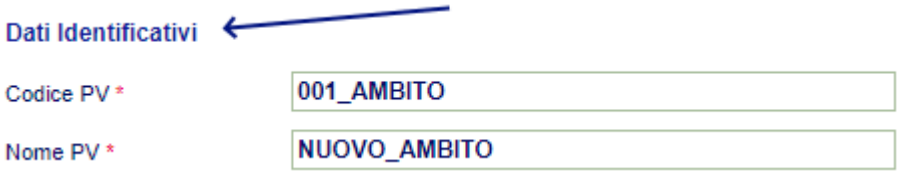
Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
idAttributeDomain	
domainTableName	**tbl_attribute_domain_value**
strQuery	istruzione sql di select, scritta in chiaro. La select deve comprendere i campi usati nei parametri <fieldToStore> e <fieldToShow>
fieldToShow	campo con il valore da mostrare
fieldToStore	campo con il valore da salvare in tabella
initSelValue	valore iniziale di default

LABEL

(torna a elenco widget)

Inserisce un testo descrittivo non collegato ad alcun campo del database. Viene utilizzato generalmente per qualificare il gruppo di attributi che lo seguono.



Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
text	Testo dell'etichetta

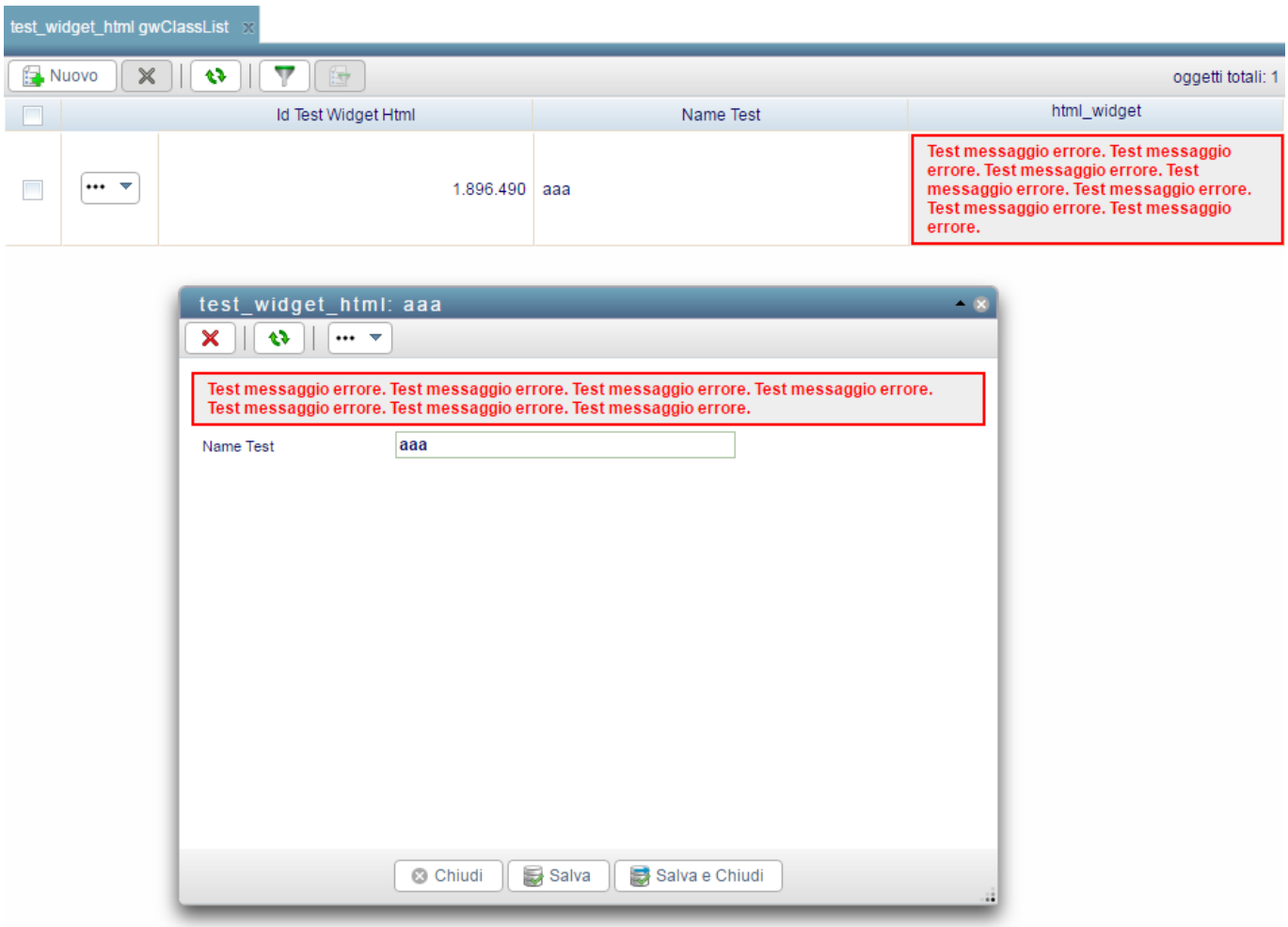
HTML

(torna a elenco widget)

Questo widget serve per renderizzare codice html contenuto in un campo di una tabella di classe.
In pratica rende possible inserire nel dettaglio di classe un elemento di spicco, popolato a runtime o derivato da altre informazioni o calcoli della vista associata alla classe.

Ad esempio, per mostrare eventuali errori risultanti dalla validazione di un serviceTask di processo, a questa stringa sul db corrisponderà:

Test messaggio errore. Test messaggio errore. Test messaggio errore. Test messaggio errore. Test messaggio errore. Test messaggio errore. Test messaggio errore.



E' supportata anche la visualizzazione in lista.

E' possibile utilizzare un set di stili creati ad hoc. Questi possono essere definiti in apposite regole in un apposito foglio di stile, situato nel war del client:

```
META-INF/static-resources/css/configuratorRules.css
```

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
useColSpan	se a true elimina lo spazio destinato alla label dell'attributo (che di norma rende allineati verticalmente gli input della form); default: true
showLabelInDetail	stabilisce se deve essere visualizzata o meno l'etichetta dell'attributo; default: false

---

HTMLTEMPLATE

(torna a elenco widget)

Il widget htmlTemplate permette di definire un testo fisso formattato in html definito in maniera fissa, sia nel contenuto che nella formattazione, in fase di configurazione. Può essere utilizzato, ad esempio,

per mostrare all'utente delle istruzioni per la compilazione dei campi della classe.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
textHtml	String. Codice html che verrà formattato e visualizzato
behaviorType	Integer. Default 1. Possibili valori [1,0]. Di norma vengono sostituite dinamicamente tutte le espressioni del tipo *${field_name}* e del tipo *${session_var_name}* in base al contesto. Per la lista vengono passati i valori dei soli widget aggiunti in lista. Analogamente per il dettaglio di classe. Se behaviorType è impostato a 0 vengono valutati i valori di tutti gli attributi anche quelli non presenti nella lista o nel dettaglio. Questo comporta una query sul dettaglio aggiuntiva, che può causare dei rallentamenti. **Nota: behaviorType a 0 è tassativamente da evitare se il widget va in lista.** Come best practise vanno nel caso messi tutti i field di riferimento in lista, al limite nascondendoli agendo nell'**xml del widget**, cosi: **<listCellStyleRules>display:none;</listCellStyleRules><listCellHeaderStyleRules>display:none;</listCellHeaderStyleRules>**. Sul dettaglio di classe, invece, si può valutare l'utilizzo di behaviorType a 0 nel caso in cui nell'htmlTemplate si richiamino molti attributi attualmente non presenti nel dettaglio, e non si voglia aggiungerli manualmente per poi nasconderli

---

MAIN_ACTION_WIDGET

(torna a elenco widget)

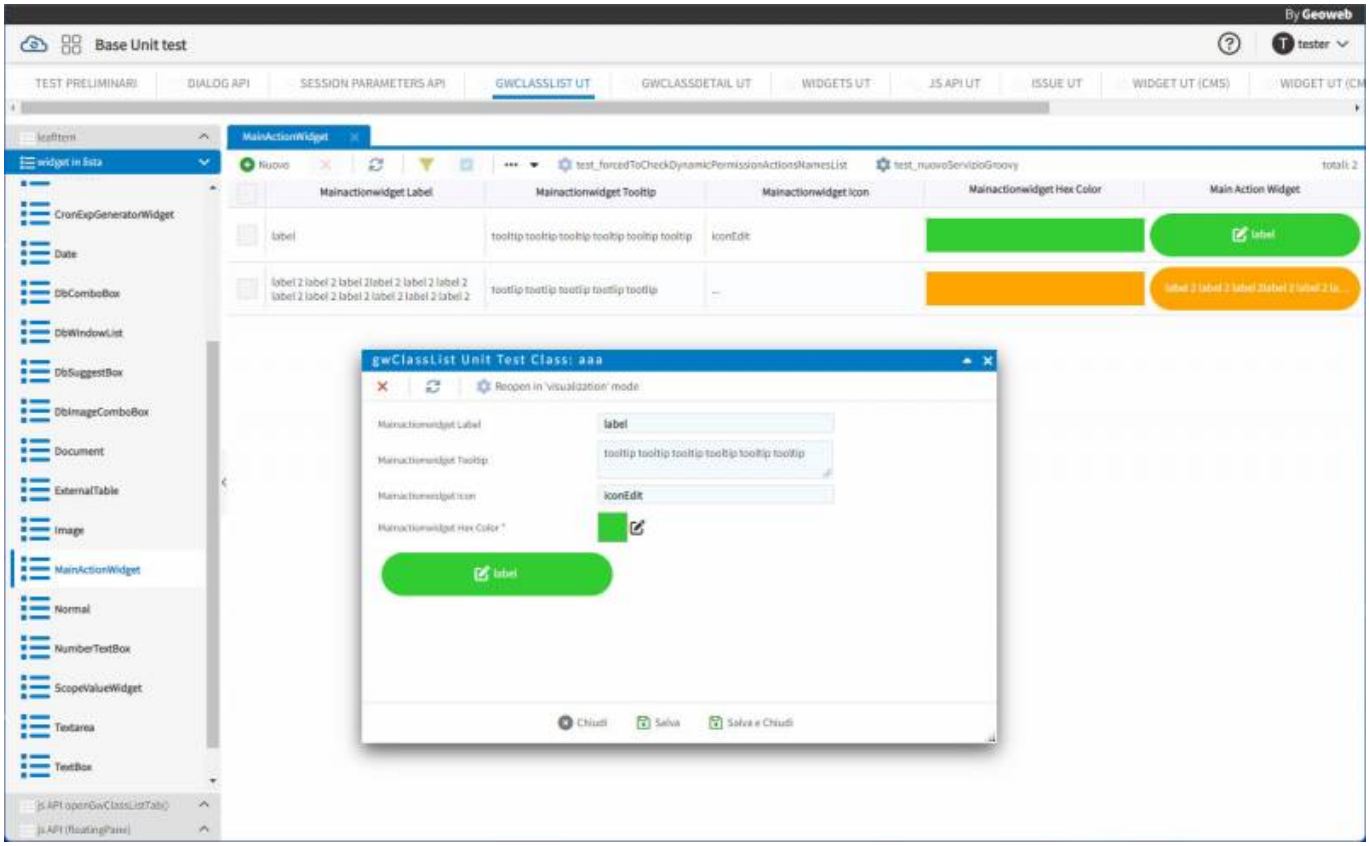
Analogamente al widget Action, permette di inserire un pulsante all'interno della dettaglio/lista, al quale associare una Action definita a livello di classe. Questo pulsante ha un alto impatto visivo. Ha inoltre una stilizzazione dinamica che dipende dai dati, che determinano: label, tooltip, icon, color. Vengono eseguiti i check statici e dinamici sull'azione.

Nota

nella gwClassList, i campi che contengono le stilizzazioni del widget, devono essere presenti in lista (tuttalpiù nascosti nell'xml del widget con la regola css *display: none;* nello style della cella e dell'header) per poter funzionare con il widget in lista.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
actionName	String, required, nome della Action definita a livello di classe
useColSpan	Impostato a True, il pulsante compare senza l'etichetta definita a livello di Attributo, e allineato con le altre etichette del form. Impostato a False, l'etichetta viene stampata, e il pulsante viene allineato in corrispondenza della colonna delle altre caselle dei controlli. default=*true* (ha senso solo nel gwClassDetail)
alignment	Definisce l'allineamento del pulsante rispetto alla colonna sopra definita. default=*left* (ha senso solo nel gwClassDetail)
labelField	String, required, campo che contiene la il valore della label da utilizzare
tooltipField	String, optional, in caso di assenza viene utilizzato il valore recuperato tramite labelField
iconField	String, optional, configura l'icona da utilizzare per il button. Ammesse notazioni: Geoweb, Fontawesome e css class
hexColorField	String, optional, esadecimale nel formato FFFFFF (senza #)



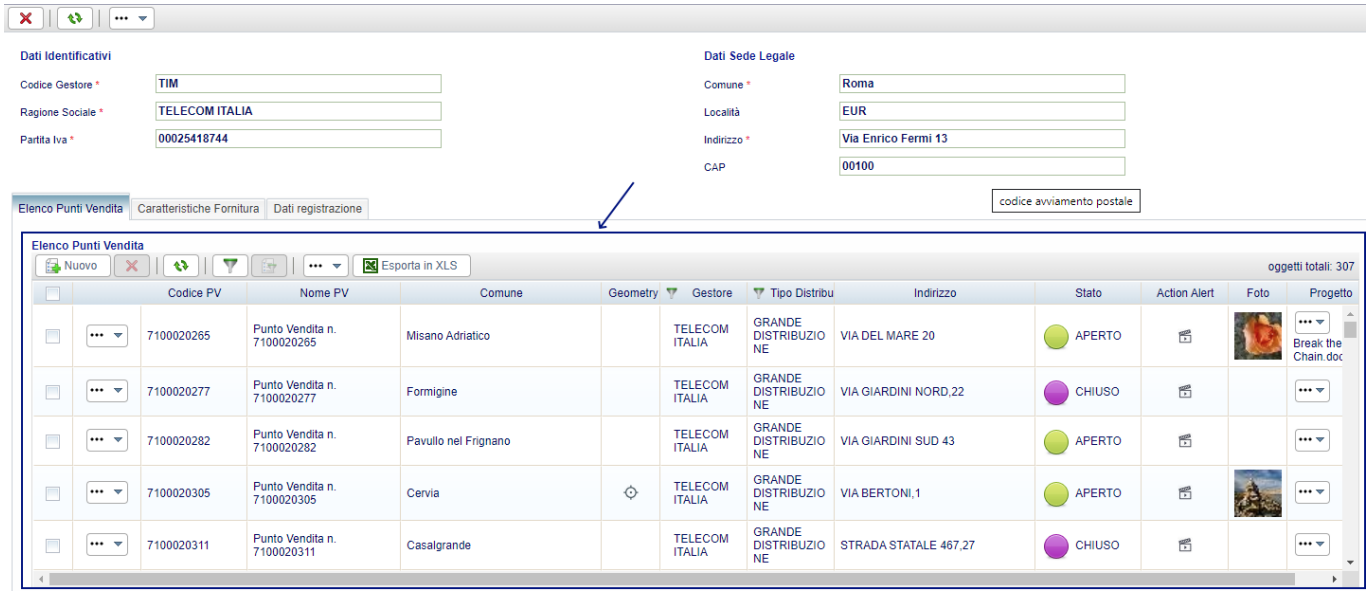
LINKLIST

(torna a elenco widget)

Il controllo permette di presentare il contenuto di una classe relazionata, all'interno della classe corrente. I record che vengono visualizzati sono unicamente quelli collegati 1 a N rispetto al record attivo. La classe collegata deve essere configurata attraverso la definizione di una relazione Geoweb esplicita.

La link list viene normalmente configurata per conto suo dentro ad un Gruppo Attributi, che viene a sua volta posizionato dentro ad una sezione del layout, adatta a contenere un elenco di record.

Le colonne visualizzate sono quelle definite in Lista nella classe relazionata. Viene inoltre riportata la toolbar di lista, completa di tutte le azioni e operazioni che possono essere eseguite nella classe relazionata, da parte dell'utente connesso.



Il click sull'elenco darà accesso al record visualizzato nel form di dettaglio e la navigazione si sposterà alla classe collegata, aprendo il record di dettaglio in pop-up o tab, secondo la configurazione già definita nel progetto per la classe collegata.
Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
relationName	nome della relazione da utilizzare; la relazione porta in sé la definizione dei campi padre e figlio delle classi collegate, che quindi non occorre ripetere.
columnNamesToHide	colonne da nascondere rispetto a quelle definite in Lista per la classe relazionata
onRowClickGwActionName	azione che può essere eseguita al click sulla riga. Usata normalmente per disabilitare l'apertura della scheda di dettaglio
onRowClickEnabled	**(da 4.8.4, issue #1476)** se impostato a false, il click sulla riga non esegue alcuna operazione (né l'apertura del dettaglio, né l'azione sul click); se non impostato vale l'impostazione generale gwClassListSettings dell'xml di progetto, altrimenti default=true
onInsertButtonClickGwActionName	azione che può essere eseguita quando si clicca sul pulsante 'Nuovo'
onDeleteButtonClickGwActionName	azione che può essere eseguita quando si clicca sul pulsante 'Nuovo'
insertCallbackGwActionName	azione che può essere inserita come callback della transazione di Inserimento nuovo record
updateCallbackGwActionName	azione che può essere inserita come callback della transazione di Aggiornamento di un record
deleteCallbackGwActionName	azione che può essere inserita come callback della transazione di Eliminazione di un record
refreshDetailOnChange	se impostato a true, esegue il refresh del dettaglio della classe corrente (padre) ad ogni modifica della classe relazionata; default=false
inheritClassPermission	se impostato a true, il controllo eredita i permessi della classe corrente (padre), sovrascrivendo i permessi della classe relazionata; default=false
hideActionsDropDownButtonColumn	se impostato a true, nasconde il tasto 'More Options' (con i tre puntini), indipendentemente dal contenuto; default=false

Parametro	Descrizione
hideIndirectSelectionColumn	se impostato a true, nasconde la colonna di selezione i record in lista, indipendentemente dai permessi; default=false
hideNewIgnorePermissions	se impostato a true, nasconde il tasto 'Nuovo', indipendentemente dai permessi; default=false
hideDeleteIgnorePermissions	se impostato a true, nasconde il tasto 'Elimina', indipendentemente dai permessi; default=false

LINKLISTNAM

(torna a elenco widget)

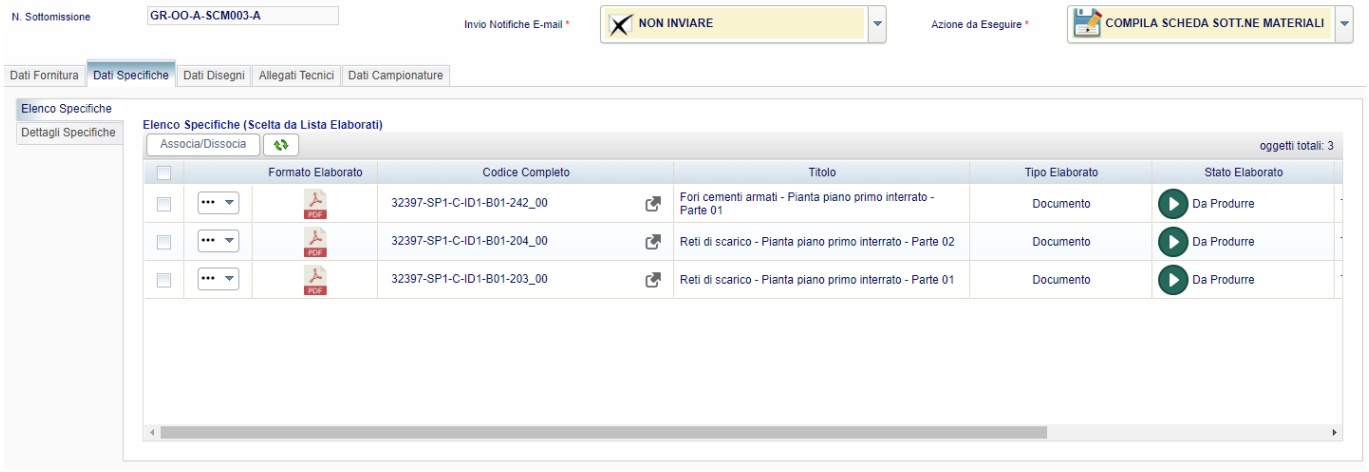
Il controllo permette di presentare il contenuto di una classe relazionata con collegamento N a M (molti a molti), all'interno della classe corrente. I record che vengono visualizzati sono unicamente quelli collegati rispetto al record attivo. Il tipo di collegamento N a M presuppone l'esistenza di una tabella/classe intermedia che mette in collegamento la classe padre con quella di destinazione, e nella quale sono presenti sia le chiave della classe padre che le chiavi della tabella/classe collegata. C La classe collegata deve essere configurata attraverso la definizione di due relazioni Geoweb esplicite:

- la prima relazione è tra la classe padre e la tabella intermedia, ed è definita nella classe corrente.
- la seconda relazione è definita nella tabella intermedia, e deve puntare alla tabella\classe collegata.

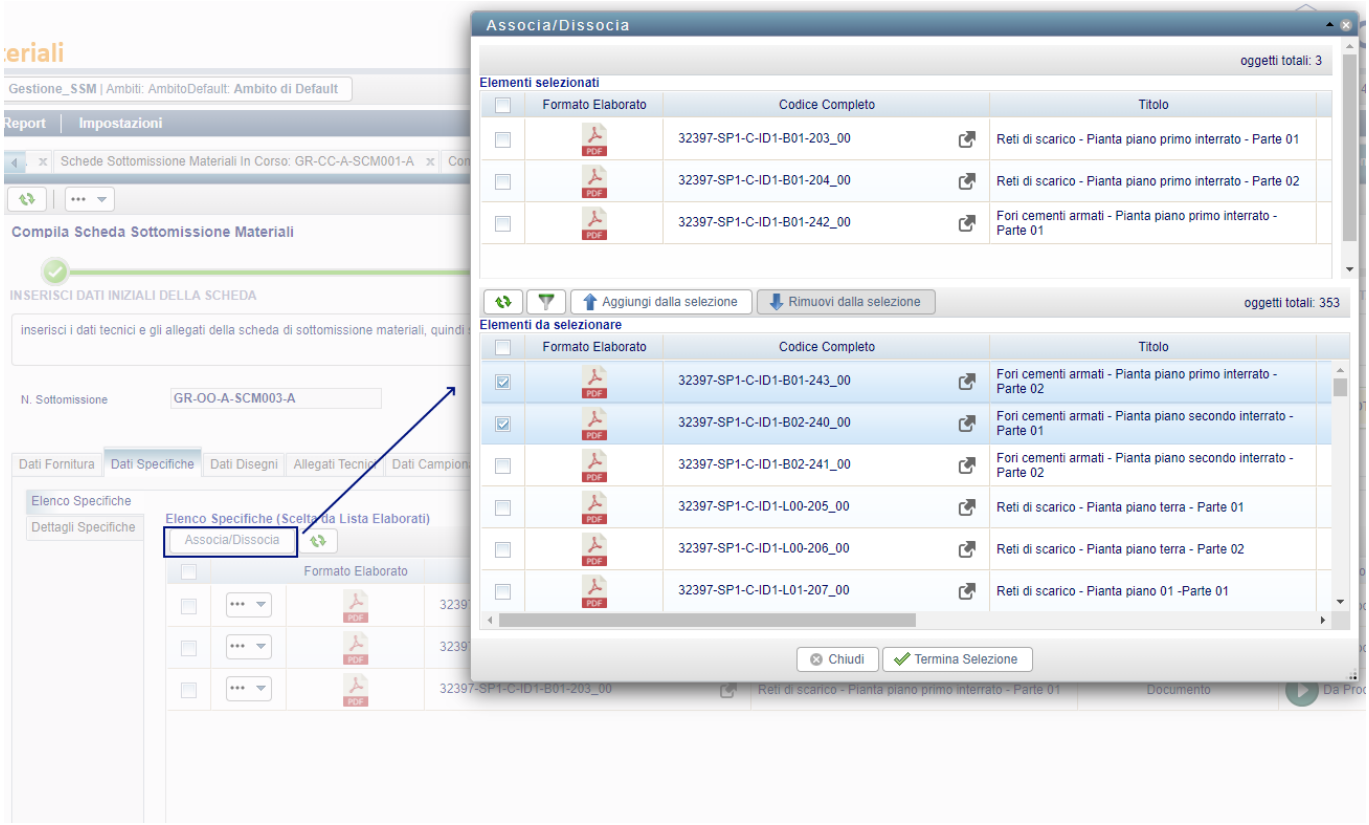
Un esempio di relazione N a M è data tra clienti e prodotti: i clienti possono acquistare diversi prodotti e i prodotti possono essere acquistati da diversi clienti. La tabella intermedia può essere quella degli acquisti, in cui vengono annotati gli acquisti dei prodotti effettuati dai diversi clienti.

Configurazione e utilizzo

La link list viene normalmente configurata per conto suo dentro ad un Gruppo Attributi, che viene a sua volta posizionato dentro ad una sezione del layout, adatta a contenere un elenco di record.
Le colonne visualizzate sono quelle definite in Lista nella classe relazionata. Viene inoltre riportata la toolbar di lista, completa di tutte le azioni e operazioni che possono essere eseguite nella classe relazionata, da parte dell'utente connesso.



La selezione dei record associati avviene tramite la presentazione di un form che contiene: in basso i record che possono essere selezionati, in alto i record che vengono scelti per essere associati.



Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
relationAssociationName	nome della relazione definita nella classe padre e che punta alla classe intermedia
relationTargetName	nome della relazione definita nella classe intermedia e che punta alla classe relazionata
columnNamesToHide	nome delle colonne in Lista, definite nella classe relazionata, da nascondere
queryClausole	eventuale filtro da applicare all'elenco dei record da selezionare mostrati nel form di Associa/Dissocia
inheritClassPermission	se impostato a true, il controllo eredita i permessi della classe padre, sovrascrivendo i permessi della classe relazionata; default=false
hideActionsDropDownButtonColumn	se impostato a true, nasconde il tasto 'More Options' (con i tre puntini), indipendentemente dal contenuto; default=false
hideIndirectSelectionColumn	se impostato a true, nasconde la colonna di selezione i record in lista, indipendentemente dai permessi; default=false
onRowClickEnabled	**(da 4.8.4, issue #1476)** se impostato a false, il click sulla riga non esegue alcuna operazione (né l'apertura del dettaglio, né l'azione sul click); se non impostato vale l'impostazione generale gwClassListSettings dell'xml di progetto, altrimenti default=true

Parametro	Descrizione
afterSelectionScriptName	**(da 4.7.4)** String, opzional. Nome dello script da eseguire alla fine della selezione. Il groovy viene eseguito nella medesima transazione delle associazioni/disassociazioni (rollback in caso di exception). Variabili disponibili: **services, all session variables, scriptName, log, parameters/parameterMap** (con dentro: **attributeGwid, projectName, associationClassKeyColumnValue, associatedList, removedList, startClass, associationClass, targetClass, associationRelation, targetRelation**)

EXTERNALTABLE

(torna a elenco widget)

Elenco di scelta mono/multi selezione che permette di selezionare uno o più valori presenti in una classe definita in Geoweb. Viene mostrata con una finestra modale che si apre in pop-up.

Il widget presenta le colonne definite in Lista nella classe di riferimento.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
classReference	Classe di riferimento
queryClausole	Clausola where che può essere utilizzata come filtro. La parola 'where' va omessa. Può accettare delle espressioni
fieldToShow	Campo da mostrare (ATTENZIONE, il campo scelto deve essere uno degli attributi in LISTA della classe di riferimento)
fieldToStore	Campo da salvare
allowMultipleSelection	Accetta una selezione multipla, se impostato a true
multipleSelectionSeparator	Separatore dei valori in caso di selezione multipla
hideSelectInMapButton	Nasconde la selezione in mappa
tableName	
initSelValueToShow	Valore di default del campo da mostrare
isVariableAttribute	Identifica il widget come un attributo variabile
isGeometricGwClassType	
dialogWidth	Larghezza in pixel della finestra di dialogo che sarà aperta
dialogHeight	Altezza in pixel della finestra di dialogo che sarà aperta
attributeNamesToHide	Permette di selezionare una o più colonne da nascondere
openDialogExpanded	Se impostato a true, apre la finestra di dialogo espansa in modo che copra le dimensioni massime
onRowClickEnabled	**(da 4.8.4, issue #1476)** se impostato a false, il click sulla riga non esegue alcuna operazione (né l'apertura del dettaglio, né l'azione sul click); se non impostato vale l'impostazione generale gwClassListSettings dell'xml di progetto, altrimenti default=true
initSelValue	Valore di default

CHILDLIST

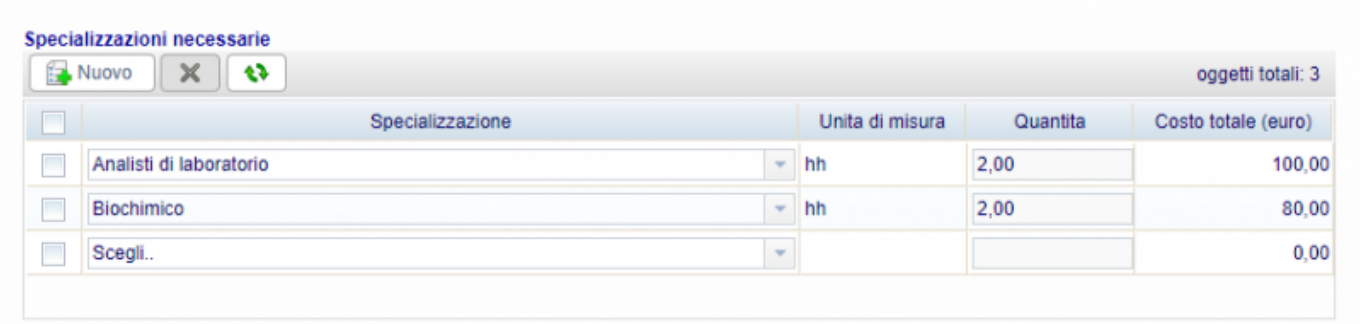
(torna a elenco widget)

Questo controllo consente di gestire una sottotabella della anagrafica principale, permettendo l'editing dei record direttamente dalla lista visualizzata.

La relazione che c'è tra le anagrafiche è la seguente:

TabellaPartenza.CampoPartenza = TabellaDestinazione.CampoDestinazione

La sottotabella deve essere definita essa stessa come classe, in quanto lo widget fa riferimento alla sua definizione. Le colonne della lista sono le colonne della lista sono infatti quelle configurate nell'anagrafica di destinazione.



Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
fkClasse	Campo della classe padre (classe di partenza) collegato al campo della classe figlia (classe di destinazione)
fkAttributo	Campo della classe figlia (classe di destinazione) collegato al campo della classe padre (classe di partenza)
className	Nome della classe di destinazione
columnsToHide	Se si vuole ottenere la lista della classe di destinazione privata di alcune colonne reputate inutili nel contesto corrente

---

NORMAL

(torna a elenco widget)

Questo tipo di controllo è una semplice casella di testo in cui viene editata e mostrata l'informazione. Il widget utilizza solamente i parametri comuni

---

NUMBERBOX

(torna a elenco widget)

Il numberbox è uno widget studiato per i valori di tipo numerico. E' possibile stabilire un pattern per le cifre, il numero di cifre dopo la virgola, e un formato di presentazione (es. Valuta, Percentuale, ecc.). Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
numberFormatType	Formato del Numero per default NUMBER. Valori ammessi (scrivere in maiuscolo): NUMBER, CURRENCY, PERCENT
pattern	Pattern da utilizzare sia per la visualizzazione che per l'input del numero. Per la documentazione fare riferimento alla guida in linea della Unicode CLDR
min	Numero minimo accettabile in fase di input
max	Numero massimo accettabile in fase di input
currencyCode	Codice della valuta utilizzabile in caso che il Formato del Numero sia stato impostato a CURRENCY. Fare riferimento alla codifica attiva Unicode CLDR
placeHolder	Testo che compare all'interno della casella di testo, come un suggerimento
promptMessage	Messaggio che compare come tooltip quando il focus viene spostato sullo widget
invalidMessage	Messaggio che compare se l'input utente non è conforme alle regole configurate
missingMessage	Messaggio che compare se il campo è in modalità 'required' e non è stato compilato

---

TEXTAREA

(torna a elenco widget)

Casella di testo pensata per inserire testi piuttosto lunghi, strutturati anche su più righe. La caseela di testo si può allungare o allargare anche a run-time, ovvero quando la maschera è in visualizzazione. Inoltre la casella di testo può essere renderizzata sia come sermplice boh html, che come un Editor di testo struttturato con gli strumenti base necessari ad una fromattazione del testo da inserire. Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione	Parametro xml
numRows	Numero di righe della casella di testo	
numCols	Numero di colonne della casella di testo	
type	Tipo di controllo, numerico, utilizzato nella form. I valori possibili sono: 0 (HTML), 1 (Editor Semplice), 2 (Editor Avanzato)	
maxWidth	Larghezza massima a cui si può portare, a run-time, la casella di testo	
maxHeight	Altezza massima a cui si può portare, a run-time, la casella di testo	

HTMLEDITOR

(torna a elenco widget)

L'html editor è un widget utile alla scrittura di testo formattato, pensato per il corpo delle email. In edit è possibile modificare e formattare il testo, mentre in visualize viene mostrato il testo formattato.

Il widget utilizza i parametri comuni

TEXTBOX

(torna a elenco widget)

Analogo allo widget NORMAL, ma strutturato per il controllo raffinato dell'input. Permette di verificare il testo inserito tramite Espressioni regolari, e di modificare il testo appena inserito con trasformazioni in maiuscolo, minuscolo, ecc. Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
regExp	Stringa di espressione regolare che verrà applicata per validare il testo inserito. E' disponibile on line parecchia documentazione utile a definire una espressione regolare corretta. Sono suggerite una guida completa in inglese e un articolo in italiano ben fatto. Esempi: Espressione regolare per evitare l'inserimento di caratteri diversi da lettere, numeri e tratti bassi: ^[\w\d_]*$ Stessa Espressione regolare vista sopra, che include nel set ammesso anche gli spazi vuoti: ^[\w\d\s_]*$ indirizzo email: [a-zA-Z0-9._%-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4} codice fiscale: [a-zA-Z]{6}\d\d[a-zA-Z]\d\d[a-zA-Z]\d\d\d[a-zA-Z] numero telefonico (solo numeri): [0-9]+\-[0-9]+ partita iva: ^[0-9]{11}$
regExpGen	Funzione javascript che effettua la validazione. Deve tornare il corpo dell'espressione regolare senza marcatori di inizio: **^[** e di fine **]*$** Esempio:

```
function(constraints){
    if(true) return "\\d{5}";
    return "\\d{10}";
}
```

trim	Toglie i caratteri SPAZIO a inizio e a fine del testo inserito
lowercase	Trasforma il testo inserito in minuscolo
uppercase	Trasforma il testo inserito in maiuscolo
propercase	Trasforma il testo mettendo in maiuscolo le iniziali di parola
placeHolder	Testo che compare all'interno della casella di testo, come un suggerimento
promptMessage	Messaggio che compare come tooltip quando il focus viene spostato sullo widget
invalidMessage	Messaggio che compare se l'input utente non è conforme alle regole configurate
missingMessage	Messaggio che compare se il campo è in modalità 'required' e non è stato compilato

POINT

(torna a elenco widget)

Il widget si applica ai campi di tipo geometry, Point. Permette di gestire la visualizzazione, l'inserimento e la modifica, e di effettuare lo zoom su mappa o su planimetria, utilizzando il visualizzatore 2D.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
type	Tipo di geometria, in questo caso deve essere impostato, in maniera fissa, a *point*
drawing	se impostato a *true* indica che la geometria viene gestita in un disegno CAD, e quindi viene caricata e tenuta aggiornata tramite tale disegno (drawing)

Parametro	Descrizione
drawingType	qualora *drawing* sia impostato a true, compilare con il NAME della definizione di disegno contenuto nella classe DRAWING_TYPE. Tale classe contiene le regole xml di configurazione ed interpretazione del disegno
coordinateSystem	codice del Sistema di Coordinate. Formati ammessi: ['EPSG:3857', '3857']. Se specificato viene usato per convertire il sistema di coordinate di origine della geometria (qui indicato) in maniera da essere correttamente visualizzata e gestita nella mappa, se questa utilizza una proiezione in un sistema di coordinate differente. Es. se il sistema di origine è LatLon, specificare il valore '4326'
persistSridToDB	Valutato solo se coordinateSystem non è nullo. Di norma l'SRID della geometria non viene persistito sul DB, perchè viene impostato nella definizione della fonte dati in fase di configurazione della mappa in Mapguide. Ponendo questo flag a true, lo srid viene aggiunto esplicitamente all'oggetto geometry del DB
startingScale	valore intero, per default -1. Indica a quale scala verrà visualizzato in mappa l'oggetto di tipo puntuale al momento dell'esecuzione dell'azione di 'Zoom' (button con icona a forma di 'mirino'). Verrà usata la scala supportata dalla mappa uguale o immediatamente più grande del valore impostato. Se vale -1, essa viene aperta al massimo zoom possibile (*)
multipleEditing	deprecato

(*) Il valore dello zoom minimo raggiungibile dalla mappa dipende SEMPRE dalla base cartografica utilizzata come sfondo. Ad esempio, GoogleMaps, per ogni zona del globo, ha diversi livelli di zoom differenziati sia in base alla tipologia della mappa che in base al dettaglio disponibile per le varie zone del mondo (le città hanno in genere dettaglio più grande rispetto agli oceani..).

POLYGON

(torna a elenco widget)

Il widget si applica ai campi di tipo geometry, Poligono. Permette di gestire la visualizzazione, l'inserimento e la modifica, e di effettuare lo zoom su mappa o su planimetria, utilizzando il visualizzatore 2D.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
type	Tipo di geometria, in questo caso deve essere impostato, in maniera fissa, a *polygon*
drawing	se impostato a *true* indica che la geometria viene gestita in un disegno CAD, e quindi viene caricata e tenuta aggiornata tramite tale disegno (drawing)
drawingType	qualora *drawing* sia impostato a true, compilare con il NAME della definizione di disegno contenuto nella classe DRAWING_TYPE. Tale classe contiene le regole xml di configurazione ed interpretazione del disegno
coordinateSystem	codice del Sistema di Coordinate. Formati ammessi: ['EPSG:3857', '3857']. Se specificato viene usato per convertire il sistema di coordinate di origine della geometria (qui indicato) in maniera da essere correttamente visualizzata e gestita nella mappa, se questa utilizza una proiezione in un sistema di coordinate differente. Es. se il sistema di origine è LatLon, specificare il valore '4326'

Parametro	Descrizione
persistSridToDB	Valutato solo se coordinateSystem non è nullo. Di norma l'SRID della geometria non viene persistito sul DB, perchè viene impostato nella definizione della fonte dati in fase di configurazione della mappa in Mapguide. Ponendo questo flag a true, lo srid viene aggiunto esplicitamente all'oggetto geometry del DB
startingScale	valore intero, per default -1. Indica a quale scala verrà visualizzato in mappa l'oggetto di tipo puntuale al momento dell'esecuzione dell'azione di 'Zoom' (button con icona a forma di 'mirino'). Verrà usata la scala supportata dalla mappa uguale o immediatamente più grande del valore impostato. Se vale -1, essa viene aperta al massimo zoom possibile (*)
multipleEditing	deprecato
editAsRectangle	Se impostato a *true* abilita l'interfaccia di editing all'editazione dei poligoni esclusivamente come rettangoli allineati secondo gli assi cartesiani

(*) Il valore dello zoom minimo raggiungibile dalla mappa dipende SEMPRE dalla base cartografica utilizzata come sfondo. Ad esempio, GoogleMaps, per ogni zona del globo, ha diversi livelli di zoom differenziati sia in base alla tipologia della mappa che in base al dettaglio disponibile per le varie zone del mondo (le città hanno in genere dettaglio più grande rispetto agli oceani..).

POLYLINE

(torna a elenco widget)

Il widget si applica ai campi di tipo geometry, Polilinea. Permette di gestire la visualizzazione, l'inserimento e la modifica, e di effettuare lo zoom su mappa o su planimetria, utilizzando il visualizzatore 2D.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
type	Tipo di geometria, in questo caso deve essere impostato, in maniera fissa, a *linestring*
drawing	se impostato a *true* indica che la geometria viene gestita in un disegno CAD, e quindi viene caricata e tenuta aggiornata tramite tale disegno (drawing)
drawingType	qualora *drawing* sia impostato a true, compilare con il NAME della definizione di disegno contenuto nella classe DRAWING_TYPE. Tale classe contiene le regole xml di configurazione ed interpretazione del disegno
coordinateSystem	codice del Sistema di Coordinate. Formati ammessi: ['EPSG:3857', '3857']. Se specificato viene usato per convertire il sistema di coordinate di origine della geometria (qui indicato) in maniera da essere correttamente visualizzata e gestita nella mappa, se questa utilizza una proiezione in un sistema di coordinate differente. Es. se il sistema di origine è LatLon, specificare il valore '4326'
persistSridToDB	Valutato solo se coordinateSystem non è nullo. Di norma l'SRID della geometria non viene persistito sul DB, perchè viene impostato nella definizione della fonte dati in fase di configurazione della mappa in Mapguide. Ponendo questo flag a true, lo srid viene aggiunto esplicitamente all'oggetto geometry del DB

Parametro	Descrizione
startingScale	valore intero, per default -1. Indica a quale scala verrà visualizzato in mappa l'oggetto di tipo puntuale al momento dell'esecuzione dell'azione di 'Zoom' (button con icona a forma di 'mirino'). Verrà usata la scala supportata dalla mappa uguale o immediatamente più grande del valore impostato. Se vale -1, essa viene aperta al massimo zoom possibile (*)
multipleEditing	deprecato

(*) Il valore dello zoom minimo raggiungibile dalla mappa dipende SEMPRE dalla base cartografica utilizzata come sfondo. Ad esempio, GoogleMaps, per ogni zona del globo, ha diversi livelli di zoom differenziati sia in base alla tipologia della mappa che in base al dettaglio disponibile per le varie zone del mondo (le città hanno in genere dettaglio più grande rispetto agli oceani..).

POSITION_WIDGET

(da finire) Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
strQuery	select code_position,descr_position_type,descr_position,pk_plan from view_gwd_position
queryClausole	pk_plan=#{id_plan}
fieldToShow	code_position
fieldToStore	code_position
initSelValue	
toTranslate	false
columnsLabel	Codice,Tipo,Descrizione
columnsView	code_position,descr_position_type,descr_position
allowMultipleSelection	false
multipleSelectionSeparator	
geometryClassReference	

POSITION2_WIDGET

(da finire)

Il widget Position2 permette di memorizzare l'informazione della posizione di un record di una classe. La posizione è rappresentata dal codice di un site o di un building o di una room o di una workstation o di una generic position.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
classToHide	

Parametro	Descrizione
classList	<string>gwd_site</string> <string>gwd_building</string> <string>gwd_room</string> <string>gwd_workstation</string> <string>gwd_generic_position</string>
codeColumnList	<string>site_code</string> <string>cod_building</string> <string>cod_room</string> <string>cod_workstation</string> <string>cod_position</string>
nameColumnList	<string>name_site</string> <string>name_building</string> <string>description</string> <string>name</string> <string>cod_position</string>
descriptionColumnList	<string>description</string> <string>descr_building</string> <string>description</string> <string>name</string> <string>cod_position</string>
filterColumnNameList	<string>site_code</string> <string>type_building</string> <string>room_number</string> <string>type_workstation</string> <string>cod_position</string>
showMapSelectionButtonList	<boolean>true</boolean> <boolean>true</boolean> <boolean>true</boolean> <boolean>true</boolean> <boolean>true</boolean>
showZoomButtonList	<boolean>true</boolean> <boolean>true</boolean> <boolean>true</boolean> <boolean>true</boolean> <boolean>true</boolean>

LABEL_VALUE_JSON_WIDGET

(dalla 4.7.4, issue #1484)

Il widget LabelValueJsonWidget permette di visualizzare le informazioni contenute in un campo JSON. Rappresenta il corrispettivo del labelValuesXmlWidget, che invece lavora con tipo di dato testuale in formato xml. Caratteristiche:

- In particolare vengono presentate le coppie chiave valore esclusivamente del primo livello.
- Eventuali valori con JSON annidati vengono trasformati in String JSON e visualizzati come tali
- Le coppie chiave valore sono rese tramite una elenco di coppie label-widget.
- Le coppie label-widget possono essere raggruppati in sezioni.
- Se dall'elaborazione dei dati dovesse risultare un solo raggruppamento (configurato o automatico), la relativa intestazione non verrà visualizzata
- I widget senza raggruppamento, vengono raccolti in un gruppo creato automaticamente '*Altri*' (localizzato)

- Di base i widget vengono resi mediante un semplice campo di testo in sola lettura.
- Nel JSON possono essere configurati dei metadati (in "_metadata", key riservata e pertanto non oggetto di rendering), che sono utilizzati per una rappresentazione più rispondente alla natura del dato.
- I dati di tipo *image* e *file* sono sempre su documentale (supportati sia CMIS che CMS), ed il value è sempre il nome dell'immagine o del file, comprensivo di path relativo (che viene aggiunto in automatico al root path configurato nel configuration.properties)

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
labelWidth	string (html width), optional, default "250px" usata per il dimensionamento delle label
valueWidth	string (html width), optional, default "400px" usata per il dimensionamento dei widget dei valori
multiColumnViewEnabled	boolean, optional, default true Quando posto a true abilita la presentazione dei gruppi della Famiglia Attributi su un layout a più colonne. Il numero di colonne dipende dall'effettivo spazio orizzontale disponibile

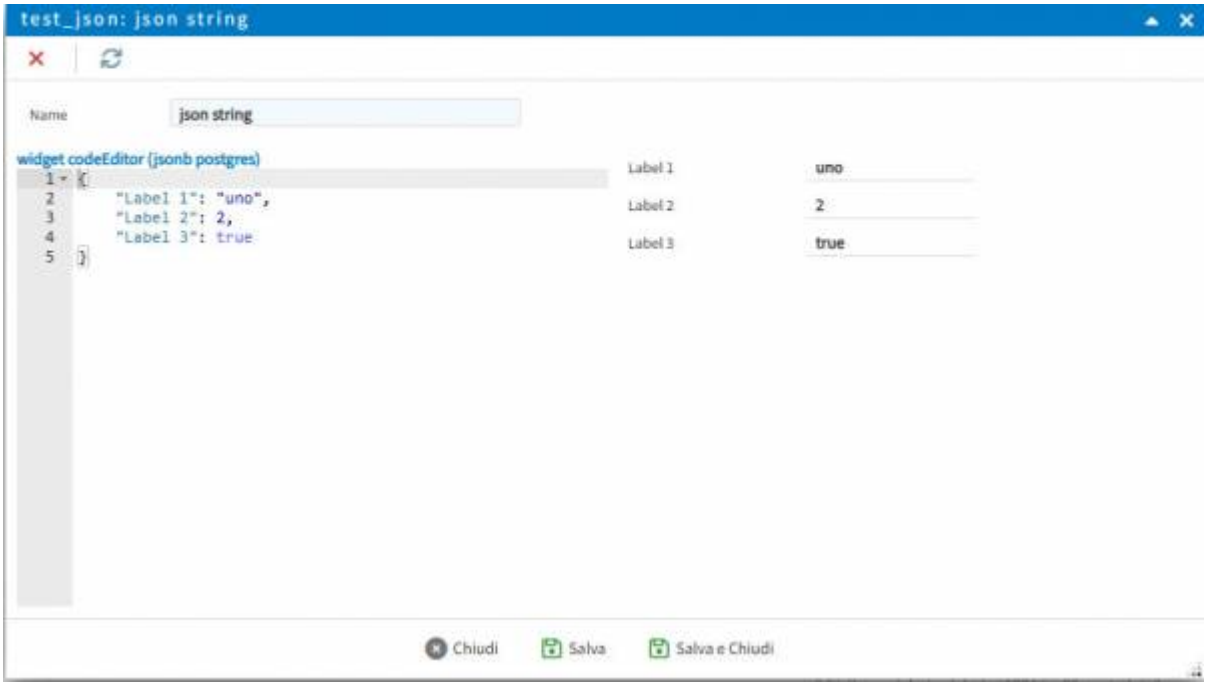
Il widget si costruisce sopra un attributo con **dataType JSON** (costruito a sua volta su un campo appropriato nel database: *json* o *jsonb* per Postgres).

Nella versione più semplice il JSON viene rappresentato automaticamente:

- usando la key per la label
- usando il valore cosi come è dentro un TextBox in sola lettura

```
{
    "Label 1": "uno",
    "Label 2": 2,
    "Label 3": true
}
```

A sinistra il JSON di origine in un widget *codeEditor*, e a destra la relativa rappresentazione *labelValueJsonWidget*.



Opzionalmente, è possibile configurare nel valore JSON anche la key **_metadata**. Dentro sono presenti due sezioni:

- **groups**, contiene informazioni relative ad eventuali raggruppamenti, ogni *key* rappresenta il *codice del gruppo* (in rapporto 1:1 con i vari “group_code” in *parameters*), e per *value* un oggetto con:
  - **label** string, usata come intestazione del raggruppamento
  - **order** integer, usato per l'ordinamento (gli elementi senza ordine hanno precedenza, in base all'ordine di comparsa nel JSON)
- **parameters**, contiene i metadati relativi alle singole coppie chiave valore, organizzate per chiave. **Ogni key è in rapporto 1:1 con le key di primo livello del JSON**, e nel relativo oggetto *value* sono presenti:
  - **type** string, optional, default “string”, valori possibili **string**, **numeric**, **date**, **boolean**, **image**, **html**
  - **label** string, optional, label da usare per la rappresentazione. In sua assenza viene usata la medesima *key* di primo livello del JSON
  - **order**, integer, optional, determina l'ordine all'interno del gruppo (gli elementi senza ordine hanno precedenza, in base all'ordine di comparsa nel JSON)
  - **group_code**, string, optional, se configurato determina i quale gruppo verrà posizionata la coppia chiave-valore. In sua assenza verrà usato un raggruppamento predefinito con label localizzata '*Altri*'
  - **params**, object, optional, se configurato specializza la rappresentazione del valore. I parametri variano in base al *type*
    - *type string*
      - **useTextArea**, boolean, optional, default false. Quando a true viene usata una TextArea per la rappresentazione
      - **height**, string (HTML height), optional, default “40px”. Da usare solo con useTextArea true, determina l'altezza iniziale della TextArea
      - **maxHeight**, string (HTML height), optional, default “120px”. Da usare solo con useTextArea true, determina l'altezza massima della TextArea
    - *type numeric*
      - **constraints**, object, optional, default null. Con dentro:
        - **places** integer, optional, default null. Numero di cifre da visualizzare

dopo la virgola nei numeri decimali (ha priorità sul pattern)
        - **pattern** string, optional, default null. Stesso funzionamento del *widget NumberBox*. Esempi: “#.”, “#.00”, “#.## %”, “#.00 €”, “#. €”
    - *type date* (nel *value* del primo livello del JSON ammessi valori nel formato stringa e intero (data in millisecondi))
        - **constraints**, object, optional, default null. Con dentro:
            - **datePattern** string, optional, default “dd/MM/yyyy”. Formati ammessi: *“dd-MM-yyyy”, “yyyy-MM-dd”, “yyyy/MM/dd”, “dd/MM/yyyy”, “dd MMMM yyyy”, “yyyy MMMM dd”, “yyyy/MM/dd HH:mm:ss”, “yyyy/MM/dd HH:mm”, “yyyy-MM-dd HH:mm:ss”, “yyyy-MM-dd HH:mm”, “dd-MM-yyyy HH:mm:ss”, “dd-MM-yyyy HH:mm”, “dd/MM/yyyy HH:mm:ss”, “dd/MM/yyyy HH:mm”, “HH:mm:ss”: “HH:mm:ss”, “EEEE, dd MMMM yyyy”*
        - **format** @Deprecated from 4.7.4 (usare constraints.datePattern), string, optional default “dd/MM/yyyy”. Formati ammessi: *“dd-MM-yyyy”, “yyyy-MM-dd”, “yyyy/MM/dd”, “dd/MM/yyyy”, “dd MMMM yyyy”, “yyyy MMMM dd”, “yyyy/MM/dd HH:mm:ss”, “yyyy/MM/dd HH:mm”, “yyyy-MM-dd HH:mm:ss”, “yyyy-MM-dd HH:mm”, “dd-MM-yyyy HH:mm:ss”, “dd-MM-yyyy HH:mm”, “dd/MM/yyyy HH:mm:ss”, “dd/MM/yyyy HH:mm”, “HH:mm:ss”: “HH:mm:ss”, “EEEE, dd MMMM yyyy”*
    - *type image* (nel *value* del primo livello del JSON ammessi solo valori stringa (nome del file, con estensione, con eventuale path relativo))
        - **height**, string (HTML height), optional, default “100px”. Quando usato imposta un'altezza specifica per l'immagine
    - *type file* (nel *value* del primo livello del JSON ammessi solo valori stringa (nome del file, con estensione, con eventuale path relativo))
    - *type boolean* (nel *value* del primo livello del JSON ammessi valori nel formato boolean true/false)
    - *type html* (nel *value* del primo livello del JSON ammessi solo valori stringa, contenenti codice html)

Esempi

Esempio semplificato Configurazione con metadati minimale per un tipo di dato stringa.

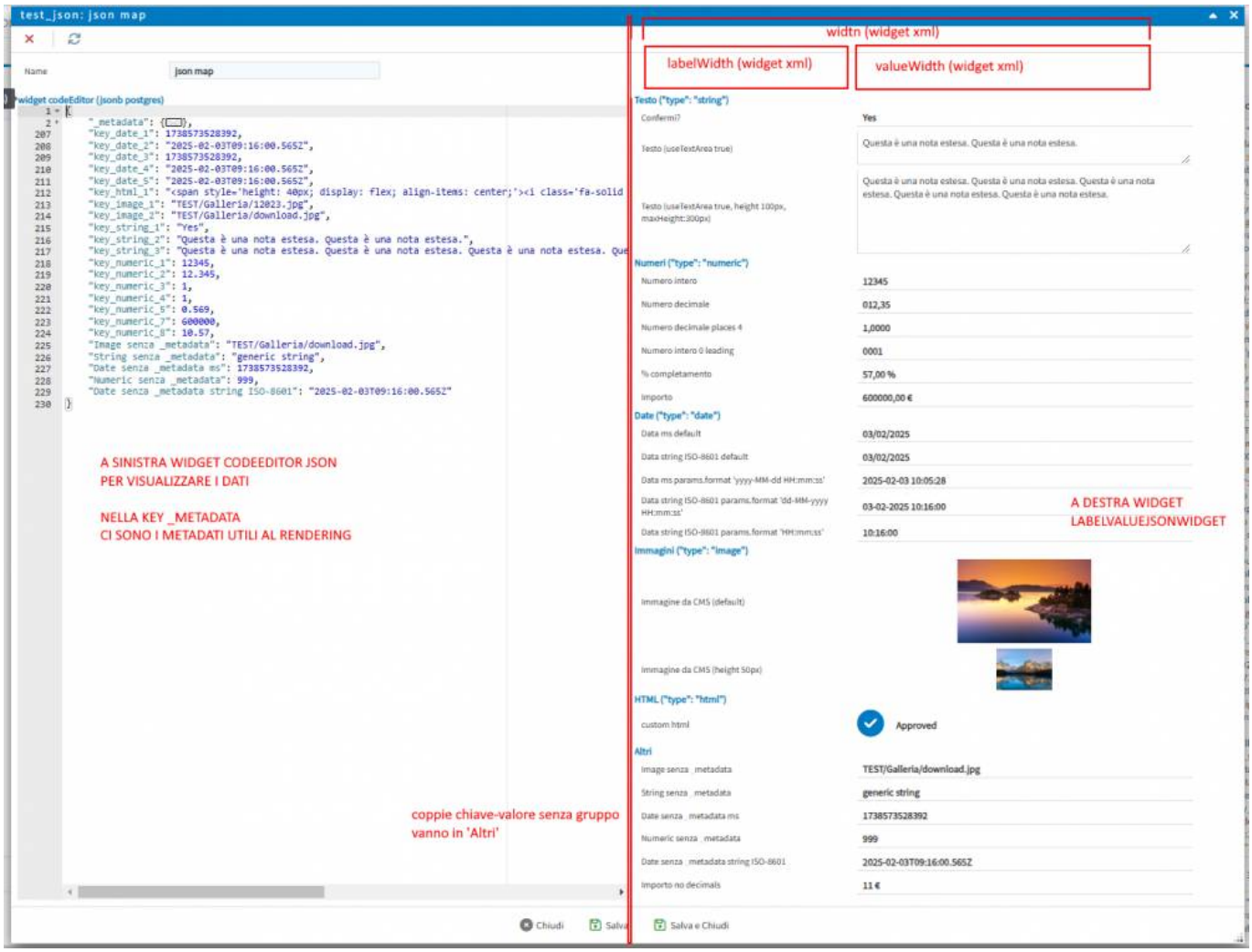
```
{
    "key_date_1": "valore 1",

    "_metadata": {
        "groups": {
            "group_1": {
                "label": "Testo (\"type\": \"string\")",
                "order": 1
            }
        },
        "parameters": {
            "key_date_1": {
                "type": "string",
                "label": "Label visualizzata",
```

```
                "order": 1,
                "group_code": "group_1"
            }
        }
    }

}
```

Esempio esteso In questo esempio sono presenti tutte le tipologie di dato, predisposte con molte varianti di presentazione. A sinistra del gwClassDetail è presente il dato JSON in un widget codeEditor. A destra il relativo risultato finale.



Relativo dato JSON comprensivo di metadati

```
{
    "key_date_1": 1738573528392,
    "key_date_2": "2025-02-03T09:16:00.565Z",
    "key_date_3": 1738573528392,
    "key_date_4": "2025-02-03T09:16:00.565Z",
    "key_date_5": "2025-02-03T09:16:00.565Z",
    "key_html_1": "<span style='height: 40px; display: flex; align-items:
center;'><i class='fa-solid fa-circle-check gwMainColor icon32x32'></i><span
style='margin-left: 15px; font-weight: bold;' >Approved<span/><span/>",
    "key_image_1": "TEST/Galleria/12023.jpg",
```

```
    "key_image_2": "TEST/Galleria/download.jpg",
    "key_string_1": "Yes",
    "key_string_2": "Questa è una nota estesa. Questa è una nota estesa.",
    "key_string_3": "Questa è una nota estesa. Questa è una nota estesa.
Questa è una nota estesa. Questa è una nota estesa. Questa è una nota
estesa.",
    "key_numeric_1": 12345,
    "key_numeric_2": 12.345,
    "key_numeric_3": 1,
    "key_numeric_4": 1,
    "key_numeric_5": 0.569,
    "key_numeric_7": 600000,
    "key_numeric_8": 10.57,
    "Image senza _metadata": "TEST/Galleria/download.jpg",
    "String senza _metadata": "generic string",
    "Date senza _metadata ms": 1738573528392,
    "Numeric senza _metadata": 999,
    "Date senza _metadata string ISO-8601": "2025-02-03T09:16:00.565Z",

    "_metadata": {
        "groups": {
            "group_1": {
                "label": "Testo (\"type\": \"string\")",
                "order": 1
            },
            "group_2": {
                "label": "Numeri (\"type\": \"numeric\")",
                "order": 2
            },
            "group_3": {
                "label": "Date (\"type\": \"date\")",
                "order": 3
            },
            "group_4": {
                "label": "Immagini (\"type\": \"image\")",
                "order": 4
            },
            "group_5": {
                "label": "HTML (\"type\": \"html\")",
                "order": 5
            }
        },
        "parameters": {
            "key_date_1": {
                "type": "date",
                "label": "Data ms default",
                "order": 1,
                "group_code": "group_3"
            },
            "key_date_2": {
```

```
                "type": "date",
                "label": "Data string ISO-8601 default",
                "order": 2,
                "group_code": "group_3"
            },
            "key_date_3": {
                "type": "date",
                "label": "Data ms params.format 'yyyy-MM-dd HH:mm:ss'",
                "order": 3,
                "params": {
                    "constraints": {"datePattern": "yyyy-MM-dd HH:mm:ss"}
                },
                "group_code": "group_3"
            },
            "key_date_4": {
                "type": "date",
                "label": "Data string ISO-8601 params.format 'dd-MM-yyyy
HH:mm:ss'",
                "order": 4,
                "params": {
                    "constraints": {"datePattern": "dd-MM-yyyy HH:mm:ss"}
                },
                "group_code": "group_3"
            },
            "key_date_5": {
                "type": "date",
                "label": "Data string ISO-8601 params.format 'HH:mm:ss'",
                "order": 5,
                "params": {
                    "constraints": {"datePattern": "HH:mm:ss"}
                },
                "group_code": "group_3"
            },
            "key_html_1": {
                "type": "html",
                "label": "custom html",
                "order": 1,
                "params": {},
                "group_code": "group_5"
            },
            "key_image_1": {
                "type": "image",
                "label": "Immagine da CMS (default)",
                "order": 1,
                "group_code": "group_4"
            },
            "key_image_2": {
                "type": "image",
                "label": "Immagine da CMS (height 50px)",
                "order": 2,
                "params": {
```

```
                    "height": "50px"
                },
                "group_code": "group_4"
            },
            "key_string_1": {
                "type": "string",
                "label": "Confermi?",
                "order": 1,
                "group_code": "group_1"
            },
            "key_string_2": {
                "type": "string",
                "label": "Testo (useTextArea true)",
                "order": 2,
                "params": {
                    "useTextArea": true
                },
                "group_code": "group_1"
            },
            "key_string_3": {
                "type": "string",
                "label": "Testo (useTextArea true, height 100px,
maxHeight:300px)",
                "order": 3,
                "params": {
                    "height": "100px",
                    "maxHeight": "300px",
                    "useTextArea": true
                },
                "group_code": "group_1"
            },
            "key_numeric_1": {
                "type": "numeric",
                "label": "Numero intero",
                "order": 1,
                "params": {
                    "constraints": {
                        "places": 0,
                        "pattern": "#."
                    }
                },
                "group_code": "group_2"
            },
            "key_numeric_2": {
                "type": "numeric",
                "label": "Numero decimale",
                "order": 2,
                "params": {
                    "constraints": {
                        "places": 2,
```

```
                    "pattern": "00#."
                }
            },
            "group_code": "group_2"
        },
        "key_numeric_3": {
            "type": "numeric",
            "label": "Numero decimale places 4",
            "order": 3,
            "params": {
                "constraints": {
                    "places": 4,
                    "pattern": "#."
                }
            },
            "group_code": "group_2"
        },
        "key_numeric_4": {
            "type": "numeric",
            "label": "Numero intero 0 leading",
            "order": 4,
            "params": {
                "constraints": {
                    "places": 0,
                    "pattern": "000#."
                }
            },
            "group_code": "group_2"
        },
        "key_numeric_5": {
            "type": "numeric",
            "label": "% completamento",
            "order": 5,
            "params": {
                "constraints": {
                    "places": 2,
                    "pattern": "#.## %"
                }
            },
            "group_code": "group_2"
        },
        "key_numeric_6": {
            "type": "numeric",
            "label": "% completamento places 2",
            "order": 6,
            "params": {
                "constraints": {
                    "pattern": "#. %"
                }
            },
            "group_code": "group_2"
```

```
            },
            "key_numeric_7": {
                "type": "numeric",
                "label": "Importo",
                "order": 7,
                "params": {
                    "constraints": {
                        "pattern": "#.00 €"
                    }
                },
                "group_code": "group_2"
            },
            "key_numeric_8": {
                "type": "numeric",
                "label": "Importo no decimals",
                "order": 8,
                "params": {
                    "constraints": {
                        "pattern": "#. €"
                    }
                }
            }
        }
    }
}
```

Popolamento JSON tramite groovy

Il widget nasce con finalità di presentazione del dato. Nelle varie piattaforme il dato tipicamente deriva da attributi variabili, ospitati in classi/tabelle differenti dalla classe padre dove tipicamente dov'à essere usato questo widget.

Nella classe padre si avrà quindi un campo JSON, che verrà popolato con solo una sintesi di tutta la mole di dati presente negli attributi variabili.

Il popolamento del dato avviene tipicamente tramite groovy.

Sotto un groovy di esempio per la creazione del JSON con relativi metadati.

Si tenga presente che quando nel GwAdmin è impostato il tipo di dato JSON per un attributo, nella mappa destinata all'insert/update, si può usare indistintamente sia una string in formato JSON, che una mappa, con coppie chiave valore. Per praticità conviene sempre usare una mappa. Questo anche perchè nei anche nei vari metodi di seleizone il valoredi un field json viene restituito come mappa.

```
String gwClassName = "gw_class_name";

def map = [:];

def jsonMap = [:];
```

```
//popolamento key-value primo livello json
//TODO rinominare le key e recuperare i valori da altre classi/tabelle
jsonMap.key_date_1 = 1738573528392;
jsonMap.key_date_2 = "2025-02-03T09:16:00.565Z";
jsonMap.key_date_3 = 1738573528392;
jsonMap.key_date_4 = "2025-02-03T09:16:00.565Z";
jsonMap.key_date_5 = "2025-02-03T09:16:00.565Z";
jsonMap.key_html_1 = "<span style='height: 40px; display: flex; align-items:
center;'><i class='fa-solid fa-circle-check gwMainColor icon32x32'></i><span
style='margin-left: 15px; font-weight: bold;' >Approved<span/><span/>";
jsonMap.key_image_1 = "TEST/Galleria/12023.jpg";
jsonMap.key_image_2 = "TEST/Galleria/download.jpg";
jsonMap.key_string_1 = "Yes";
jsonMap.key_string_2 = "Questa è una nota estesa. Questa è una nota
estesa.";
jsonMap.key_string_3 = "Questa è una nota estesa. Questa è una nota estesa.
Questa è una nota estesa. Questa è una nota estesa. Questa è una nota
estesa.";
jsonMap.key_numeric_1 = 12345;
jsonMap.key_numeric_2 = 12.345;
jsonMap.key_numeric_3 = 1;
jsonMap.key_numeric_4 = 1;
jsonMap.key_numeric_5 = 0.569;
jsonMap.key_numeric_7 = 600000;
jsonMap.key_numeric_8 = 10.57;

//popolamento key-value primo livello json senza _metadata (e con key
complessa che viene usata come label)
jsonMap["Image senza _metadata"] = "TEST/Galleria/download.jpg";
jsonMap["String senza _metadata"] = "generic string";
jsonMap["Date senza _metadata ms"] = 1738573528392;
jsonMap["Numeric senza _metadata"] = 999;
jsonMap["Date senza _metadata string ISO-8601"] =
"2025-02-03T09:16:00.565Z";



//popolamento key _metadata

//definizione dei gruppi
def groupsMap = [
    group_1: [
      label: "Testo (\"type\": \"string\")",
      order: 1
    ],
    group_2: [
      label: "Testo (\"type\": \"numeric\")",
      order: 2
    ],
    group_3: [
```

```
      label: "Testo (\"type\": \"date\")",
      order: 3
    ],
    group_4: [
      label: "Testo (\"type\": \"image\")",
      order: 4
    ],
    group_5: [
      label: "Testo (\"type\": \"html\")",
      order: 5
    ]
];

//definizione dei parametri dei metadata
def parametersMap = [
    key_string_1: [
        type: "string",
        label: "Confermi?",
        order: 1,
        group_code: "group_1"
    ],
    key_string_2: [
        type: "string",
        label: "Testo (useTextArea true)",
        order: 2,
        params: [
            useTextArea: true
        ],
        group_code: "group_1"
    ],
    key_string_3: [
        type: "string",
        label: "Testo (useTextArea true, height 100px, maxHeight:300px)",
        order: 3,
        params: [
            useTextArea: true,
            height: "100px",
            maxHeight: "300px"
        ],
        group_code: "group_1"
    ],

    key_numeric_1: [
        type: "numeric",
        label: "Numero intero",
        order: 1,
        params: [
            constraints: [
                places: 0,
                pattern: "#."
```

```
            ]
        ],
        group_code: "group_2"
    ],
    key_numeric_2: [
        type: "numeric",
        label: "Numero decimale",
        order: 2,
        params: [
            constraints: [
                places: 2,
                pattern: "00#."
            ]
        ],
        group_code: "group_2"
    ],
    key_numeric_3: [
        type: "numeric",
        label: "Numero decimale places 4",
        order: 3,
        params: [
            constraints: [
                places: 4,
                pattern: "#."
            ]
        ],
        group_code: "group_2"
    ],
    key_numeric_4: [
        type: "numeric",
        label: "Numero intero 0 leading",
        order: 4,
        params: [
            constraints: [
                places: 0,
                pattern: "000#."
            ]
        ],
        group_code: "group_2"
    ],
    key_numeric_5: [
        type: "numeric",
        label: "% completamento",
        order: 5,
        params: [
            constraints: [
                places: 2,
                pattern: "#.## %"
            ]
        ],
        group_code: "group_2"
```

```
    ],
    key_numeric_6: [
        type: "numeric",
        label: "% completamento places 2",
        order: 6,
        params: [
            constraints: [
                pattern: "#. %"
            ]
        ],
        group_code: "group_2"
    ],
    key_numeric_7: [
        type: "numeric",
        label: "Importo",
        order: 7,
        params: [
            constraints: [
                pattern: "#.00 €"
            ]
        ],
        group_code: "group_2"
    ],
    key_numeric_8: [
        type: "numeric",
        label: "Importo no decimals",
        order: 8,
        params: [
            constraints: [
                pattern: "#. €"
            ]
        ]
    ],

    key_date_1: [
        type: "date",
        label: "Data ms default",
        order: 1,
        group_code: "group_3"
    ],
    key_date_2: [
        type: "date",
        label: "Data string ISO-8601 default",
        order: 2,
        group_code: "group_3"
    ],
    key_date_3: [
        type: "date",
        label: "Data ms params.format 'yyyy-MM-dd HH:mm:ss'",
        order: 3,
```

```
        params: [
            format: "yyyy-MM-dd HH:mm:ss"
        ],
        group_code: "group_3"
    ],
    key_date_4: [
        type: "date",
        label: "Data string ISO-8601 params.format 'dd-MM-yyyy HH:mm:ss'",
        order: 4,
        params: [
            format: "dd-MM-yyyy HH:mm:ss"
        ],
        group_code: "group_3"
    ],
    key_date_5: [
        type: "date",
        label: "Data string ISO-8601 params.format 'HH:mm:ss'",
        order: 5,
        params: [
            format: "HH:mm:ss"
        ],
        group_code: "group_3"
    ],

    key_image_1: [
        type: "image",
        label: "Immagine da CMS (default)",
        order: 1,
        group_code: "group_4"
    ],
    key_image_2: [
        type: "image",
        label: "Immagine da CMS (height 50px)",
        order: 2,
        params: [
            height: "50px"
        ],
        group_code: "group_4"
    ],

    key_html_1: [
        type: "html",
        label: "custom html",
        order: 1,
        params: [],
        group_code: "group_5"
    ]
];


jsonMap._metadata = [
```

```
  groups: groupsMap,
  parameters: parametersMap
];

map.id_field = 123;
//...
map.jsonb_field = jsonMap;

services.classService.updateClassRecord(gwClassName, map);
```

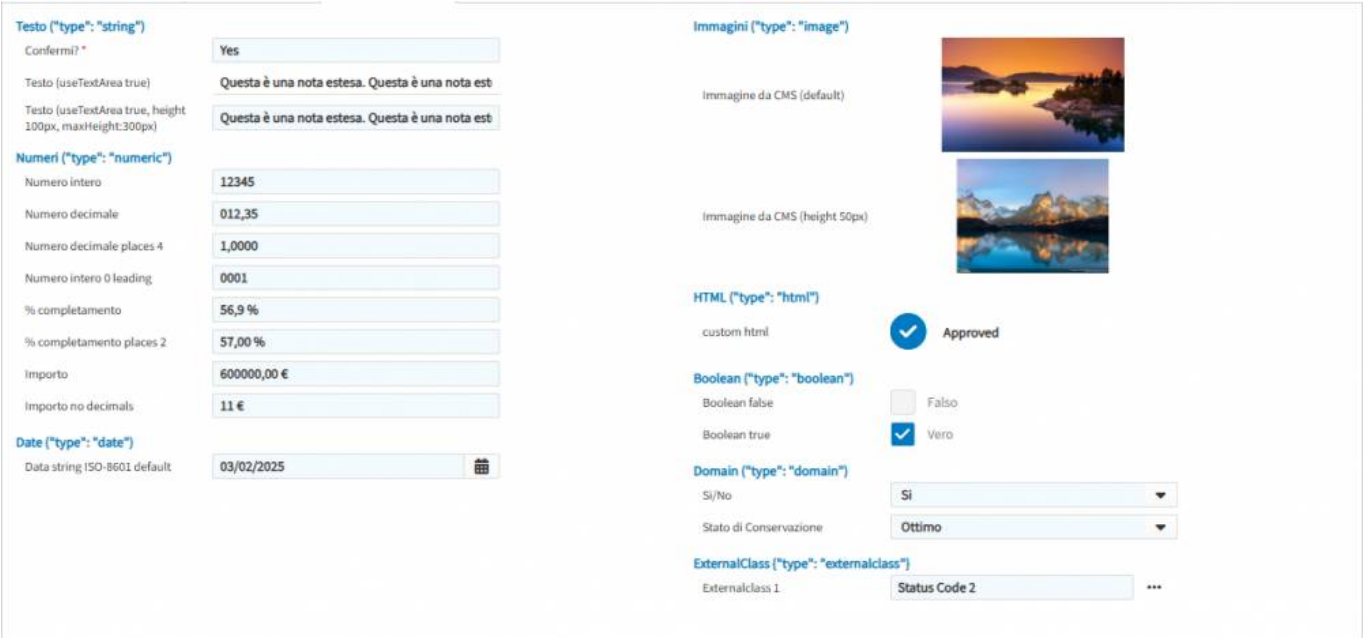
Note implementative per le date

Sebbene il visualizzatore JSON supporto formalmente una grande varietà di formati supportati per il type 'date', nelle implementazioni va usato la versione stringa nel formato '**dd-MM-yyyy HH:mm:ss**' (uguale al *timestamp* del database). Quindi NON la versione ISO-8601, con o senza TIMEZONE.

Questo per le logiche dei filtri del queryBuilder di geoweb.

PROPERTY_SET_WIDGET

(dalla 4.7.4, issue #1327)



Il widget PropertySetWidget permette di visualizzare le informazioni contenute in un campo JSON. I metadati usati per la presentazione dei valori del JSON sono presi dalla definizione della Famiglia Attributi correntemente impostata in un altro widget dello stesso dettaglio (Questo lo differenzia dal LABEL_VALUE_JSON_VISUALIZER, dove i metadati, se presenti, sono contenuti direttamente nel JSON stesso).

Caratteristiche:

- In particolare vengono presentate le coppie chiave valore esclusivamente del primo livello.

- Eventuali valori con JSON annidati vengono trasformati in String JSON e visualizzati come tali
- Le coppie chiave valore sono rese tramite una elenco di coppie label-widget.
- Le coppie label-widget possono essere raggruppati in gruppi.
- Se dall'elaborazione dei metadati dovesse risultare un solo raggruppamento, la relativa intestazione non verrà visualizzata
- I dati di tipo *image* e *file* sono sempre su documentale (supportati sia CMIS che CMS), ed il value è sempre il nome dell'immagine o del file, comprensivo di path relativo (che viene aggiunto in automatico al root path configurato nel configuration.properties)

xml di esempio

```
<PropertySetWidget>
  <width>-1</width>
  <height>-1</height>
  <readonly>false</readonly>
  <disabled>false</disabled>
  <intermediateChanges>false</intermediateChanges>
  <isDefaultOrderBy>true</isDefaultOrderBy>
  <isDefaultOrderByAscending>true</isDefaultOrderByAscending>
  <isDefaultOrderByOnFieldToShow>true</isDefaultOrderByOnFieldToShow>
  <listCellTextAlign></listCellTextAlign>
  <listCellStyleRules></listCellStyleRules>
  <listCellClass></listCellClass>
  <listCellHeaderStyleRules></listCellHeaderStyleRules>
  <importCSVWithoutDecoding>true</importCSVWithoutDecoding>
  <labelWidth>150</labelWidth>
  <valueWidth>250</valueWidth>
  <propertySetCodeField>cod_attribute_family</propertySetCodeField>
  <multiColumnViewEnabled>true</multiColumnViewEnabled>
  <checkRequiredServerSide>true</checkRequiredServerSide>
  <checkPropertiesServerSide>true</checkPropertiesServerSide>
<avoidExceptionOnFailedPropertyCheckServerSide>true</avoidExceptionOnFailedP
ropertyCheckServerSide>
</PropertySetWidget>
```

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
labelWidth	string (html width), optional, default “250px” usata per il dimensionamento delle label
valueWidth	string (html width), optional, default “400px” usata per il dimensionamento dei widget dei valori
propertySetCodeField	string, required, è il nome di un altro widget presente nel gwClassDetail che detiene il valore del codice della Famiglia Attributi. Questo widget sarà il punto di riferimento del PropertySetWidget, che ne monitorerà i cambiamenti di valore al fine di presentare i valori del JSON secondo la definizione della Famiglia Attributi corrente

Parametro	Descrizione
multiColumnViewEnabled	boolean, optional, default true Quando posto a true abilita la presentazione dei gruppi della Famiglia Attributi su un layout a più colonne. Il numero di colonne dipende dall'effettivo spazio orizzontale disponibile
checkRequiredServerSide	boolean, optional, default true. Quando posto a true abilita il controllo quando si utilizzano i metodi insertClassRecord()/updateClassRecord() sui valori delle key dichiarate required da definizione PropertySet. Nel caso il cui valore sia nullviene sollevata un'eccezione (issue #1649) **dalla v4.9.0**
checkMaxLengthServerSide	boolean, optional, default true. Quando posto a true abilita il controllo quando si utilizzano i metodi insertClassRecord()/updateClassRecord() sui valori delle key di tipo string nella cui definizione PropertySet è impostato un maxLength. Nel caso il cui valore sia null viene sollevata un'eccezione (issue #1657) **dalla v4.9.0**
checkUnknownPropertiesServerSide	boolean, optional, default true. Quando posto a true abilita il controllo quando si utilizzano i metodi insertClassRecord()/updateClassRecord() sulla coerenza delle key con la definizione del PropertySet, e lancia eccezione in caso di discrepanze (vedi anche avoidExceptionOnFailedPropertyCheckServerSide sotto). Non viene fatto un check sulla coerenza del tipo di valore (issue #1658) **dalla v4.9.0**
avoidExceptionOnUnknownPropertiesServerSide	boolean, optional, default true. Valutato solo quando checkUnknownPropertiesServerSideè true. Quando posto a true, nell'utilizzo dei metodi insertClassRecord()/updateClassRecord(), evita di sollevare una eccezione in caso di mancata aderenza alla definizione del PropertySet, ma si limita a rimuovere le key non riconosciute (issue #1658) **dalla v4.9.0**

Il widget si costruisce sopra un attributo con **dataType JSON** (costruito a sua volta su un campo appropriato nel database: *json* o *jsonb* per Postgres).

Configurazione Widget Enterprise

Plugin Classification

Questo plugin nasce per definire in Geoweb delle classi 'gerarchizzate' o 'classificate'. In particolare tali classi servono per descrivere degli oggetti che vanno inventariati e gestiti allo stesso modo tra loro, ma possono essere di diversa natura, e quindi avere una definizione di attributi diversa una dall'altra.

Tipicamente, in una azienda, classi di questo genere sono utili a descrivere l'elenco degli Asset che costituiscono l'inventario aziendale.

Ad esempio fanno parte degli Asset aziendali le apparecchiature informatiche, gli arredi, le auto aziendali ecc. Pur essendo quindi un insieme di oggetti che costituiscono nel loro insieme il patrimonio aziendale, questi oggetti hanno natura diversa e quindi caratteristiche diverse tra loro che occorre annotare e gestire.

Quindi la struttura della 'Classificazione' permette di raggruppare tali oggetti in Famiglie e SottoFamiglie (gerarchie di oggetti), e ognuna delle sottofamiglie può avere una serie di attributi che la caratterizzano per la sua particolare Natura (RAM del computer, piuttosto che cavalli fiscali dell'auto aziendale, ecc.).

Il plugin quindi fornisce una serie di componenti, quali un componente di Menu (leafItemClassificationMenu) che permette di sfogliare la gerarchia delle famiglie, di definire nuove famiglie e definire/modificare il set di attributi variabili. Il plugin mette a disposizione inoltre dei Widget che permettono di classificare un nuovo oggetto, modificarne uno esistente, visionare e inserire dati negli Attributi Variabili.

Di seguito il collegamento alla prima analisi di modello dati su cui è stato basato il plugin.

classificazione_datamodel_vintage.pdf

CLASSIFICATION

(torna a elenco widget enterprise)

Questo widget permette di definire la famiglia di appartenenza dell'Oggetto che si sta inserendo o modificando. Permette di scegliere la Famiglia grazie a un componente che fa sfogliare il tree su una pop-up.
In visualizzazione che ne presenta la gerarchia definita

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

classReference	fa riferimento alla classe gerarchizzata
mapPath	percorso delle mappe, utile per la tematizzazione automatica, in mappa, delle famiglie che vengono aggiunte vedi riferimenti

VARIABLEATTRIBUTES

(torna a elenco widget enterprise)

(to do)

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

par1
par2

RELATION_WIDGET

(torna a elenco widget enterprise)

(to do)

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

par1
par2

Plugin Workflow

(torna a elenco widget Enterprise)

Il plugin Workflow contiene le funzionalità di interfacciamento al motore Activiti. In questa sezione vengono descritti i soli widget, ma tutto l'argomento è trattato diffusamente nel capitolo workflow.

GW_RELATED_PROCESSES_WIDGET

(torna a elenco widget Enterprise)

Questi nuovi widget nascono dall’esigenza di rendere evidente all’utente tutti i processi in qualche maniera collegati ad una specifica entità in Geoweb (siano essi lanciabili, in corso o già terminati). Se questa entità potrà essere del tutto arbitraria, di fatto lo sviluppo dei nuovi componenti è stato dettato dalla necessità di esplicitare tutti i processi in qualche modo collegati a specifiche istanze dell’entità **digital document** (oggetto di uno sviluppo parallelo). Date le specifiche funzionali richieste, si è deciso di racchiudere il loro soddisfacimento nel widget **gwRelatedProcessesWidget**. Questo widget va a riunire e potenziare le funzionalità già implementate nei widget **gwTasksWidget** e **gwArchivedProcessesWidget**, che vengono ancora supportati, ma di fatto ne viene scoraggiato l’uso.

Questo widget rende fruibili per l’utente, in maniera semplice e concentrata in un unico cruscotto, le seguenti funzionalità relative all’oggetto collegato:

- Possibilità di avviare istanze di processo;
- Eventuale visualizzazione delle istanze di processo in corso;
- Eventuale visualizzazione delle istanze di processo concluse.

Il widget è renderizzato con un contenitore esterno simile all'AccordionContainer dojo, che ospita due sottosezioni:

- Sezione **processi avviabili/in corso**. Questa ospiterà una lista di oggetti eterogenei. Il numero, la tipologia e l’ordinamento di tali oggetti è configurabile tramite in un apposito file .groovy (vedi sotto), che viene valutato a *runtime* e permette di cambiare dinamicamente l’interfaccia, in base a tutta una serie di parametri. Questi oggetti possono essere di due tipologie:
  - Button per avviare nuove istanze di processo (eventualmente personalizzabile da file .groovy in icona e label);
  - Blocco con informazioni sull’istanza di processo in corso (label ed icona personalizzazbili in groovy), e il(i) task corrente(i). Al click su(i) task in corso verrà aperta la corrispettiva

scheda gwTaskDetail con, a seconda dei casi, la possibilità di completare o richiedere l'esecuzione del task o di poter solo visualizzarlo.
Il blocco presenta la possibilità di essere espanso e mostrare ulteriori informazioni:
- Eventuale button per cancellare l'istanza di processo in corso (la cui visibilità icona e label è configurabile da file .groovy). Da notare che a monte verrà comunque verificato l'apposito permesso statico 'Cancella istanza di processo', legato a gruppo ed ambito, configurabile nel gwAdmin). L'annullamento dell'istanza di processo comporterà la ricomparsa del button per avviare il processo;
- Lista delle attività (*task*) passate già eseguite, in ordine di data completamento decrescente. Vengono visualizzate le seguenti informazioni:
  - *nome attività*;
  - *data di completamento*;

- Sezione dei **processi conclusi** ospiterà i processi conclusi.

Il widget permette di avviare più tipologie di processo alla volta, ma vincola l'esecuzione di una sola istanza di processo alla volta per una data tipologia (*processDefinitionKey*). Sta all'utente configuratore specificare nell'implementazione del file .groovy se un'eventuale istanza di processo in corso di una certa tipologia, debba inibire anche la possibilità di avviare un'istanza di processo di un'altra tipologia, o meno. A tale scopo viene fornita come parametro nel metodo del .groovy anche la lista delle istanze di processo:

```
List<ProcessInstance> processInstanceList
```

Il widget tiene sincronizzate nelle istanze di processo generate in Activiti le seguenti variabili di processo, con nome fisso:

- **gw_related_entity_id**, String, il contenuto del campo dichiarato *keyColumn* per la classe;
- **gw_related_entity_name**, String, il contenuto del campo con name *entityCodColumnName* specificato nel file .groovy, oppure, in sua assenza, il contenuto del campo configurato come *nameColumn* per la classe.

E' obbligatorio per l'utente configuratore la creazione, nella tabella e relativa gwClass di processo, dei campi *gw_related_entity_id* e *gw_related_entity_name* e relativi gwAttribute in modo da averli sincronizzati in automatico e poterli eventualmente usare in ogni momento per eseguire delle query. Volutamente non verranno riversate fra le variabili di processo tutte le variabili della entità collegata (es: *digital document*). Sia per evitare un inutile sovraffollamento, che per evitare possibili collisioni dei nomi usati nella classe collegata con quelli propri delle variabili di processo (tipo generici *start_date*, etc..). Se necessario, nei groovy di processo si potranno recuperare tutte le informazioni della classe collegata tramite *gw_related_entity_id* o *gw_related_entity_name*.

```
  Es. (postgres)
  ALTER TABLE test_gw44data.aim_wpi_asset_assignto ADD COLUMN
gw_related_entity_id character varying(250);
  ALTER TABLE test_gw44data.aim_wpi_asset_assignto ADD COLUMN
gw_related_entity_name character varying(250);
```

PARAMETRI

Parametro	Descrizione
entityCodColumnName	String, optional. Nome della colonna della classe del widget che verrà usata per recuperare il valore con cui eseguire la query in Activiti. Se omessa, o valorizzata come stringa vuota, di default verrà utilizzata la *name_column* della gwClass. Il parametro può essere utile in quei casi dove per la stessa entità in geoweb si prevede che possano essere collegati in maniera indipendente diversi processi in Activiti, che però fanno riferimento a contesti diversi. Per esempio, questo campo potrebbe essere generato da una vista, concatenando qualche informazione che funga da ambito/contesto (es: cod_commessa) al contenuto del columnName base della classe, cosicché il widget possa mostrare in maniera certa solo i processi che fanno riferimento alla determinata entità nello specifico contesto
entityStatusColumnName	String, optional. Nome della colonna della classe del widget che verrà usata per recuperare lo stato del record corrente. Il valore dello stato verrà passato come parametro (*entityStatus*) nell'invocazione del metodo del file groovy (vedi sotto). Se omesso, verrà passato *null*
groovy	String, required. Nome del file groovy (comprensivo di estensione .groovy) che ospiterà una class java. In alternativa si potrà impostare direttamente una java class disponibile nel *classPath*, indicandone il nome preceduto dal path completo. Questa java class dovrà estendere i metodi della classe *GwRelatedProcessesConfigImpl* (che a sua volta implementa i metodi dell'interfaccia *GwRelatedProcessesConfigAbs*). Se omesso verrà utilizzata una implementazione base dell'interfaccia che ritorna una lista di definizione di comandi vuota

Esempio di configurazione xml widget:

```
<gwRelatedProcessesWidget>
    …
    <entityCodColumnName></entityCodColumnName>
    <entityStatusColumnName></entityStatusColumnName >
    <groovy>TestGwRelatedProcessesWidgetConfigImpl </groovy>
</gwRelatedProcessesWidget >
```

Esempio configurazione groovy:

```
import org.activiti.engine.runtime.ProcessInstance;
import
com.geowebframework.workflowplugin.model.widget.GwRelatedProcessesWidgetConf
igImpl;
public class TestGwRelatedProcessesWidgetConfigImpl extends
GwRelatedProcessesWidgetConfigImpl {
    public List< Map<String,Object>> getRelatedProcessDefinitionList (
        String entityId,
        String entityName,
        String entityStatus,
        String gwUser,
        String gwGroup,
        Map<String, Object> gwActiveScopesMap,
        List<ProcessInstance> processInstanceList
    ){
```

```
    return [
        [
            processDefinitionKey: "",
            message: "", //required if necessary
            isStartable: true, //default true
            startLabel: "",    //optional
            startIconClass: "",    //optional
            startIconImage: "",    //optional
            runningIsVisible: true,    //optional, default true
            runningLabel: "",    //optional
            runningIconClass: "", //optional
            runningIconImage: "", //optional
            isCancelable: true, //optional
            cancelLabel: "", //optional
            cancelIconClass: "", //optional
            cancelIconImage: "", //optional
            archivedIsVisible: true, //optional, default true
            processCodRelatedEntityVariableName: "" //optional, default
provided
        ]
    ]
    }
}
```

GW_ARCHIVED_PROCESSES_WIDGET

(torna a elenco widget Enterprise)

Questi nuovi widget nascono dall'esigenza di rendere evidente all'utente tutti i processi in qualche maniera collegati ad una specifica entità in Geoweb (siano essi lanciabili, in corso o già terminati). Se questa entità potrà essere del tutto arbitraria, di fatto lo sviluppo dei nuovi componenti è stato dettato dalla necessità di esplicitare tutti i processi in qualche modo collegati a specifiche istanze dell'entità **digital document** (oggetto di uno sviluppo parallelo).

Questo widget si preoccuperà di mostrare all'utente una griglia contente tutte le istanze di processo che si sono concluse e che hanno riguardato l'entità collegata.

PARAMETRI

Parametro	Descrizione
entityCodColumnName	String, optional. Nome della colonna della classe del widget che verrà usata per recuperare il valore con cui eseguire la query in Activiti. Se omessa, o valorizzata come stringa vuota, di default verrà utilizzata la *name_column* della gwClass. Il parametro può essere utile in quei casi dove per la stessa entità in geoweb si prevede che possano essere collegati in maniera indipendente diversi processi in Activiti, che però fanno riferimento a contesti diversi. Per esempio, questo campo potrebbe essere generato da una vista, concatenando qualche informazione che funga da ambito/contesto (es: cod_commessa) al contenuto del columnName base della classe, cosicché il widget possa mostrare in maniera certa solo i processi che fanno riferimento alla determinata entità nello specifico contesto
entityStatusColumnName	String, optional. Nome della colonna della classe del widget che verrà usata per recuperare lo stato del record corrente. Il valore dello stato verrà passato come parametro (*entityStatus*) nell'invocazione dei vari metodi groovy (vedi sotto). Se omesso, verrà passato *null*
groovy	String, optional. Nome del file .groovy (comprensivo di estensione .groovy) che ospiterà una class java. In alternativa si potrà impostare direttamente una java class disponibile nel *classPath*, indicandone il nome preceduto dal path completo. Questa java class dovrà estendere la classe *GwArchivedProcessesWidgetConfigImpl* (che implementa l'interfaccia *GwArchivedProcessesWidgetConfigAbs*). Se omesso verrà utilizzata una implementazione base dell'interfaccia che ritorna una lista di definizione di comandi vuota

Esempio configurazione xml widget:

```
<gwArchivedProcessesWidget>
    …
    <entityCodColumnName></entityCodColumnName>
    <entityStatusColumnName></entityStatusColumnName >
    <groovy>generic_document_gwArchivedProcessesWidgetConfigImpl</groovy>
</gwArchivedProcessesWidget>
```

Esempio configurazione groovy:

```
import
com.geowebframework.workflowplugin.model.widget.GwArchivedProcessesWidgetCon
figImpl;

public class TestGwArchivedProcessesWidgetConfigImpl extends
GwArchivedProcessesWidgetConfigImpl {

    public List<Map<String,Object>> getArchivedProcessDefinitionList(String
entityId, String entityName, String entityStatus, String gwUser, String
gwGroup, Map<String, Object> gwActiveScopesMap){
        return [
            [
                processDefinitionKey: 'utc_gwprocess'
            ],
            [
```

```
                processDefinitionKey: 'utc_gwprocess_proc'
            ]
        ]
    }

}
```

GW_TASKS_WIDGET

(torna a elenco widget Enterprise)

Questo nuovo widget nasce dall'esigenza di rendere evidente all'utente tutti i processi in qualche maniera collegati ad una specifica entità in Geoweb (siano essi lanciabili, in corso o già terminati). Se questa entità potrà essere del tutto arbitraria, di fatto lo sviluppo dei nuovi componenti è stato dettato dalla necessità di esplicitare tutti i processi in qualche modo collegati a specifiche istanze dell'entità **digital document** (oggetto di uno sviluppo parallelo).

Questo widget si preoccuperà di:

- Rendere usufruibili per l'utente, in maniera semplice e concentrata in un'unica interfaccia, tutti i *comandi* che permettano di innescare i vari processi necessari:
  - I comandi saranno disponibili come button contenuti nel widget, sotto la *label* configurata nel *gwAttribute*, disposti in verticale, uno sotto l'altro. Essi occuperanno in orizzontale tutto lo spazio disponibile;
  - Le definizioni di questi comandi in grado di avviare processi (startableProcessDefinition) saranno recuperate dinamicamente invocando l'esecuzione di un metodo

    ```
    List<Map<String,Object> getStartableProcessDefinitionList()
    ```

    della class java (che implementa l'interfaccia *GwTasksWidgetConfigAbs*) contenuta nel file configurato nel parametro **groovy** (vedi sotto). Queste definizioni sono state volutamente fatte ritornare da un metodo groovy (da implementare ed invocato dinamicamente), per riuscire un domani a rendere configurabili da interfaccia lato client i processi che l'utente può lanciare (Creando apposite classi di gestione in geoweb, per poi eseguire dal groovy delle query sulle tabelle delle classi al fine di recuperare le informazioni sui processi avviabili);

  - Tipicamente questi processi avranno come effetto ultimo sul documento, la variazione dello stato: il valore contenuto nella colonna **status_column**;
- Vincolare l'esecuzione di una sola istanza di processo (*processInstance*) alla volta (e quindi ovviamente una sola *processDefinition*):
  - Quindi all'avvio di una data istanza di processo, verrà inibita la possibilità di avviarne altre di una qualsiasi tipologia. La UI presentata non sarà più quella iniziale, ma i button per avviare i processi saranno sostituiti da un layout toolbar+griglia (sotto maggiori dettagli). Inoltre la sotto label del widget in alto, comparirà un ulteriore label indicante la tipologia di istanza di processo in corso, con il pattern 'In Corso: '+<gwProcessLabel>;
  - Per tutta la durata dell'esecuzione della istanza di processo dovrà essere possibile, in qualunque momento, annullare l'esecuzione dell'istanza stessa (l'annullamento potrà essere lanciato da un apposito button che comparirà sulla toolbar);

    - L’annullamento dell’istanza di processo potrà essere abilitato/disabilitato mediante implementazione del metodo

      ```
      Map<String,Object> getCancelableProcessDefinition()
      ```

      della classe .groovy (da notare che a monte verrà comunque verificato l’apposito permesso statico ‘*Cancella istanza di processo*’, legato a gruppo ed ambito, configurabile nel gwAdmin);

    - L’annullamento dell’istanza di processo comporterà la ricomparsa del layout del widget iniziale, con tutti i comandi dichiarati compatibili (tramite groovy) con lo stato corrente dell’entità collegata (es: *digital document*);
- Rendere visibili i task utente in corso e conclusi riguardanti l’eventuale istanza di processo in corso:
    - Questi verranno mostrati come record di simil griglia, disposta sotto la toolbar, dove compariranno le seguenti colonne:
        - label task;
        - data creazione task;
        - data completamento task;
    - o I record verranno ordinati mettendo in cima i task in corso seguiti da quelli completati. Come secondo criterio di ordinamento verrà usata la data di creazione dei task (desc);
    - I task in corso rispetteranno le convenzioni cromatiche tipiche delle altre liste dei processi (verde=task propri, giallo=task richiedibili, bianco= task assegnati ad altri);
    - Solo al click sui task in corso verrà aperta la corrispettiva scheda gwTaskDetail con, a seconda dei casi, anche la possibilità di completare o richiedere l’esecuzione del task;
- Tenere sincronizzate nelle istanze di processo generate in Activiti le variabili di processo **gw_related_entity_id** (il contenuto del campo configurato come *keyColumn*) e **gw_related_entity_name** (il contenuto del campo configurato come *nameColumn*) con nome fisso.
  E’ mandatorio per l’utente configuratore la creazione, nella tabella e relativa gwClass di processo, dei campi e relativi gwAttribute in modo da averli sincronizzati in automatico e poterli eventualmente usare in ogni momento per eseguire query. Volutamente non verranno riversate fra le variabili di processo tutte le variabili della entità collegata (es: digital document). Sia per evitare un inutile sovraffollamento, che per evitare possibili collisioni dei nomi usati nella classe collegata con quelli propri delle variabili di processo (tipo generici start_date, etc..). Se necessario, nei groovy di processo si potranno recuperare tutte le informazioni della classe collegata tramite *gw_related_entity_id* o *gw_related_entity_name*.

PARAMETRI

Parametro	Descrizione
entityCodColumnName	String, optional. Nome della colonna della classe del widget che verrà usata per recuperare il valore con cui eseguire la query in Activiti. Se omessa, o valorizzata come stringa vuota, di default verrà utilizzata la *name_column* della gwClass. Il parametro può essere utile in quei casi dove per la stessa entità in geoweb si prevede che possano essere collegati in maniera indipendente diversi processi in Activiti, che però fanno riferimento a contesti diversi. Per esempio, questo campo potrebbe essere generato da una vista, concatenando qualche informazione che funga da ambito/contesto (es: *cod_commessa*) al contenuto del *columnName* base della classe, cosicché il widget possa mostrare in maniera certa solo i processi che fanno riferimento alla determinata entità nello specifico contesto
entityStatusColumnName	String, optional. Nome della colonna della classe del widget che verrà usata per recuperare lo stato del record corrente. Il valore dello stato verrà passato come parametro (*entityStatus*) nell'invocazione dei vari metodi groovy (vedi sotto). Se omesso, verrà passato *null*
groovy	String, optional. Nome del file groovy (comprensivo di estensione .groovy) che ospiterà una class java. In alternativa si potrà impostare direttamente una java class disponibile nel *classPath*, indicandone il nome preceduto dal path completo. Questa java class dovrà implementare i metodi dell'interfaccia *GwTasksWidgetConfigAbs*. Se omesso verrà utilizzata una implementazione base dell'interfaccia che ritorna una lista di definizione di comandi vuota

Esempio configurazione xml widget:

```
<gwTasksWidget>
    …
    <entityCodColumnName></entityCodColumnName>
    <entityStatusColumnName></entityStatusColumnName>
    <groovy>generic_document_gwTasksWidgetConfigImpl</groovy>
</gwTasksWidget>
```

Esempio configurazione groovy:

```
import org.activiti.engine.runtime.ProcessInstance;
import
com.geowebframework.workflowplugin.model.widget.GwTasksWidgetConfigImpl;

public class TestGwTasksWidgetConfigImpl extends
com.geowebframework.workflowplugin.model.widget.GwTasksWidgetConfigImpl {

    public List< Map<String,Object>> getStartableProcessDefinitionList
(String entityId, String entityName, String entityStatus, String gwUser,
String gwGroup, Map<String, Object> gwActiveScopesMap){
        def res  = [] ;

        def definition1 = [ processDefinitionKey: "utc_gwprocess_wiz" ];

        def definition2 = [
            processDefinitionKey: "utc_gwprocess_proc",
```

```
            message: "utc_gwprocess_proc_start_message",
            label: "Start Process utc_gwprocess_proc (c)",
            iconClass: ""
        ];
        res.add(definition1);
        if(entityStatus!="cannotValidate") res.add(definition2);

        return res;
    }

    public Map<String,Object> getCancelableProcessDefinition (String
processDefinitionKey, String processInstanceId, String entityId, String
entityName, String entityStatus, String gwUser, String gwGroup, Map<String,
Object> gwActiveScopesMap){
        return [
             isCancelable: true
        ];
    }
}
```

Plugin ThreeDVisualizer

(torna a elenco widget Enterprise)

Questo plugin racchiude tutti i widget che fanno riferimento al modulo relativo al Bim Explorer. La definizione e il funzionamento generale di un singolo widget è del tutto analoga agli altri.

GRAPHIC_LAYOUT_WIDGET

(torna a elenco widget Enterprise)

Definisce un pulsante, collocabile sia all'interno della scheda di dettaglio che in lista, che permette di aprire un nuovo tab dove sarà visualizzata la mappa, associata al widget in questione. Al click sul pulsante viene recuperata nel database la mappa associata al valore del widget tramite una query sulla tabella gwd_layout e questa viene mostrata in un nuovo tab. Inoltre all'interno della scheda di dettaglio è stato creato un altro pulsante tramite il quale si può modificare il layout, e di conseguenza pure la mappa che viene recuperata, scegliendo fra tutte quelli disponibili nel database.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	Descrizione
FieldToStore	Corrisponde alla colonna della tabella gwd_layout il cui valore viene salvato nel database. Può essere scelto fra layout_code o pk_layout.
FieldToShow	Corrisponde alla colonna della tabella gwd_layout il cui valore viene mostrato all'utente. Può essere scelto fra layout_code o layout_label. I due valori possono essere differenti.
isCode	Flag booleano attraverso il quale va indicato se l'attributo corrisponde ad un Integer oppure a String. Se il valore è True allora l'attributo corrisponde a String altrimenti se il valore è false allora l'attributo corrisponde ad Integer

INDIRECT_LOCALIZATION

(torna a elenco widget Enterprise)

Definisce un pulsante, collocabile sia all'interno della scheda di dettaglio che in lista, che permette di selezionare e fare lo zoom sull'oggetto al quale è associato. Nel momento in cui si clicca sul pulsante, viene aperto un nuovo tab contenente una scena del Bim Explorer sulla quale sono caricati i modelli associati all'oggetto, ottenuti tramite la relazione fra il *codeColumnName3D* e il *codeColumnRelationTableName3D*. La camera sarà inizialmente impostata in modo tale da avere in primo piano l'oggetto su cui viene fatto lo zoom.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

Parametro	**Descrizione**
codeColumnName3D	Corrisponde al nome della colonna della tabella, corrispondente alla classe sulla quale viene definito il widget, che è in relazione con la colonna della *relationTableName3D* definita in *codeColumnRelationTableName3D*.
relationTableName3D	Corrisponde al nome della tabella sulla quale vengono ricercati gli oggetti sui quali fare lo zoom.
codeColumnRelationTableName3D	Corrisponde al nome della colonna della *relationTableName3D* che è in relazione con la colonna *codeColumnName3D*.
defaultGwmScene	Questo valore indica la scena di default dalla quale deve essere ricavato l'insieme dei modelli ad essa associata.
openInNewTab	Questo flag deve essere popolato tramite un booleano. Se TRUE allora il widget mantiene il solito comportamento e viene aperto un tab con il Bim Explorer nella stessa scheda, mentre se FALSE viene aperto un nuovo progetto (projectName) in un'altra scheda. In tutti gli altri casi viene eseguito di default il comportamento che ha il widget se il flag è FALSE
projectName	Indica il nome del progetto da aprire in caso di widget openInNewTab è FALSE. Di default si cercherà di aprire il progetto BIM_DATA
sessionCodeField	Indica il nome della variabile in sessione in cui andare a mettere il codice del gwdBimProject che deve essere aperto. Viene inserito fra i parametri in sessione tramite gwp solamente se è popolato questo campo

AZIONE DI ZOOM INDIRETTO MULTIPLO
Parallelamente all'introduzione di questi due nuovi flag in fase di configurazione è stata sviluppata un'azione che permette di eseguire lo zoom selezionando più di un elemento della lista. In fase di configurazione deve essere creata una azione da porre in lista sulla toolbar, il codice dell'azione deve essere il seguente:

```
multipleIndirectLocalization(grid,attributeName);
```

I parametri in ingresso per questa funzione devono essere grid e attributeName (ovvero il nome dell'attributo su cui è configurato l'indirectLocalizationWidget)

Plugin gwMnemonicCode

(torna a elenco widget Enterprise)

Questo plugin nasce per definire in Geoweb dei **codici parlanti**, ovvero codici composti di varie 'sezioni', ognuna delle quali corrisponde ad uno specifico oggetto, il cui ordine definisce una gerarchia tra gli oggetti che queste rappresentano.

Quindi la struttura del 'MnemonicCode' permette di raggruppare tali oggetti in gerarchie che ne permettono la facile organizzazione e gestione grazie agli strumenti messi a disposizione da questo plugin, quali un componente di Menu (gwMnemonicCodeSheet), che permette di visualizzare e gestire in un'unica scheda l'intera gerarchia di oggetti relativi ad un codice parlante definito in una classe, uno widget (MnemonicCodeWidget) di creazione di codici parlanti ed uno widget (MnemonicCodeSelectionWidget) che permette di effettuare una selezione da un albero gerarchico di codici parlanti. Il plugin mette a disposizione inoltre una sezione, nella configurazione (lato admin) di una classe, di creazione di un MnemonicCode al quale i vari componenti possono riferirsi. Ogni classe può avere più MnemonicCode configurati ed i singoli componenti possono fare riferimento ad uno di essi.

MNEMONIC_CODE_WIDGET

(torna a elenco widget enterprise)

Questo widget permette di definire un codice parlante per il record in cui viene inserito. La struttura di tale codice parlante dipende dalla configurazione del MnemonicCode relativo a questo attributo.

Oltre ai parametri comuni, i parametri specifici per questo widget sono i seguenti:

mnemonicCodeName	il nome del MnemonicCode su cui questo widget viene configurato
widgetIsEditable	boolean, indica se questo widget può o meno essere modificato anche dopo essere stato definito alla creazione. Se 'false', una volta definito, il codice di questo widget potrà essere visualizzato ma non modificato, anche se si hanno i permessi di modifica
progressivePopulation	boolean, indica se questo widget debba essere compilato seguendo l'ordine delle varie parti, e quindi senza lasciare vuote delle parti intermedie

MNEMONIC_CODE_SELECTION_WIDGET

(torna a elenco widget enterprise)

Questo widget permette di selezionare un codice parlante per il record, scegliendolo da un albero gerarchico relativo al MnemonicCode a cui esso fa riferimento.

Oltre ai parametri comuni, i parametri specifici sono i seguenti:

rootLabel	il titolo della finestra in cui verrà visualizzato l'albero
initSelValue	opzionale: valore iniziale impostato se questo widget non ha valore all'apertura della finestra

classReference	il nome della classe in cui è configurato il MnemonicCode su cui questo widget viene configurato
queryClausole	opzionale: clausola SQL che definisce la condizione con cui filtrare i valori di questo widget
initialFilterRoot	opzionale: codice iniziale che funge da filtro. Se impostato, nell'albero saranno visibili solo i 'figli' diretti del codice inserito
selectabilityDepth	opzionale: livello di profondità minimo entro cui è possibile selezionare i record in questo albero. Ad esempio, se impostato a 2, permetterà di selezionare solo record presenti negli ultimi due livelli di profondità. Se impostato a 0, o lasciato vuoto, permetterà di selezionare qualunque livello di profondità
mnemonicCodeName	il nome del MnemonicCode su cui questo widget viene configurato
selectionModeMultiple	DEPRECATO
nodeLabelType	DEPRECATO

JSON_FILE_REST

Torna all'elenco dei widget

Disponibile dalla versione **4.8.4** (issue #1961), plugin **gw-cms**.

Mostra in sola lettura, come scheda, il contenuto di un file `.json` salvato sul documentale, con la stessa resa delle schede di gw-cde-navigator.

A differenza di LABEL_VALUE_JSON_WIDGET, non legge un campo json/jsonb dal database e non richiede il blocco `_metadata`: layout e struttura si ricavano dal contenuto del file. Si possono quindi visualizzare anche json prodotti da terze parti, senza conoscenza del framework.

Il widget non crea cartelle e non modifica il file. La lettura verifica il permesso di visualizzazione sulla classe e l'ACL dinamica del record.

Individuazione del file

Come per il RestDocumentWidget:

- **cartella**: dal parametro `path`, sotto il `rootPath` del documentale; sono ammesse le espressioni `${}` sui campi del record e sulle variabili di sessione;
- **nome del file**: dal valore della colonna dell'attributo oppure, se l'attributo non ha colonna, dal campo indicato in `nameField`. Se il nome non ha estensione, si cerca nella cartella il file con quel nome.

Se il file non esiste viene mostrato un messaggio; un file che non contiene json valido produce un messaggio d'errore.

Parametri

Parametri comuni

Parametro	Descrizione
path	Cartella del documentale in cui si trova il file, relativa al rootPath. Espressioni `${}` ammesse (es. `${codice_fascicolo}/SCHEDE`)
nameField	Campo da cui leggere il nome del file quando l'attributo non ha colonna
width	Larghezza; default -1 (100% del pannello)
height	Altezza; default -1 (100% del pannello). In un gruppo insieme ad altri attributi il default è 400px

Layout consigliato

Configurare l'attributo **da solo in un tab o in un accordion pane**, con `height -1`: la scheda occupa tutto il pannello e scorre al suo interno.

Regole di rendering del json

Contenuto json	Resa
`{ “label”: …, “value”: … }`	campo con etichetta e valore
array	sezione, titolo dalla chiave, campi a griglia
`{ “campi”: […], …altre chiavi }`	sezione con i campi a griglia, seguita dalle altre chiavi
`{ “base64”, “formato”, “nome”, “descrizione” }` o stringa `data:image…`	immagine, ingrandibile con zoom
altri oggetti	gruppo con titolo e contenuto rientrato
valori semplici	campo con etichetta ricavata dalla chiave (_ e camelCase diventano spazi)
chiavi che contengono `separator`	ignorate

I testi lunghi (oltre 50 caratteri o con un a capo) sono mostrati in un'area di testo a tutta larghezza. Il contenuto del file non viene mai interpretato come html.

Esempio

```
{
  "Dati generali": [
    { "label": "Codice", "value": "FB-001" },
    { "label": "Indirizzo", "value": "Via Roma 1, Milano" }
  ],
  "Strutture": {
    "campi": [
      { "label": "Tipologia", "value": "Cemento armato" },
      { "label": "Anno", "value": "1975" }
    ],
    "foto": { "base64": "...", "formato": "jpeg", "nome": "prospetto.jpg",
"descrizione": "Prospetto principale" }
  },
  "note": "Testo libero"
```

```
}
```

Generazione dei file

I file json visualizzabili si possono generare con l'API `buildLabelValueJson()` di `gwClassListService` (issue #1843).

Per mostrare in un tab l'albero di una cartella di schede vedi il leafItem gwJsonFileRestTree.

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

Permanent link:
https://wiki.geowebframework.com/doku.php?id=gwusermanual:interface:widget

Last update: **2026/11/05 12:28**