

Il Documento Digitale

[Torna indietro](#)

Il Documento Digitale, di seguito abbreviato con DDOC, è un componente strutturato appositamente per gestire gli attributi di una classe come se fossero contenuti in un Documento formale, che viene compilato e validato in vari passaggi e da utenti diversi.

Il controllo permette quindi di suddividere l'anagrafica in sezioni, ognuna delle quali può essere controllata (visibile, editabile, messa in evidenza all'apertura) in maniera programmatica, in funzione dell'utente correntemente connesso, dello stato e di tutte le altre informazioni verificabili nella sessione corrente.

Dal punto di vista pratico, inserendo gli attributi in determinate sezioni, si ha la possibilità di definire delle logiche autorizzative a livello di attributo, superando il limite di Geoweb che organizza le autorizzazioni a livello di classe.

Organizzazione del DDOC

La struttura in sezioni è realizzata come un TabContainer, con le intestazioni delle sezioni poste a sinistra, in verticale. Al click su ogni sezione viene mostrato a destra lo specifico layout della sezione. La visibilità e l'editabilità di ogni singola sezione possono essere configurate tramite l'implementazione di una class java in un file .groovy (vedi sotto).

Le sezioni del documento possono essere combinate in maniera differente formando specifici **template**.

All'apertura di una scheda documento, esso viene aperto con il *template* specificato (o con uno di default: il primo trovato, per nome in ordine alfabetico).

La scheda DDOC, presenta una intestazione fissa nella parte superiore, sopra le *section*, contenente uno specifico detailLayout. Questa è di fatto una *section* speciale, marcata con lo specifico flag **is_header** (vedi sotto) e ne può esistere solo una per template.

Questa è concepita per mostrare gli attributi che sono trasversali a tutte le section, e che devono essere sempre in primo piano (per esempio codici, data creazione, stato, etc..).

Essa è sempre visibile a prescindere da quale section viene selezionata sotto.

A destra della sezione di intestazione può opzionalmente essere presente un pulsante **stampa report** pdf. Il nome del report, che deve essere definito a livello di classe, è specificato nei dati del Template. Se omesso non viene visualizzato alcun pulsante.

HEADER

Codice Interno * Utente Verifica
Formato **stampa report**

SEZIONI

Section 1 Tipo

Section 2 **Dati di Redazione e File**

Codice Cliente

Section 4 **Dati Identificativi**

Descrizione Documento

Autore *

File Documento *

Data Redazione *

Raccolta Documentale

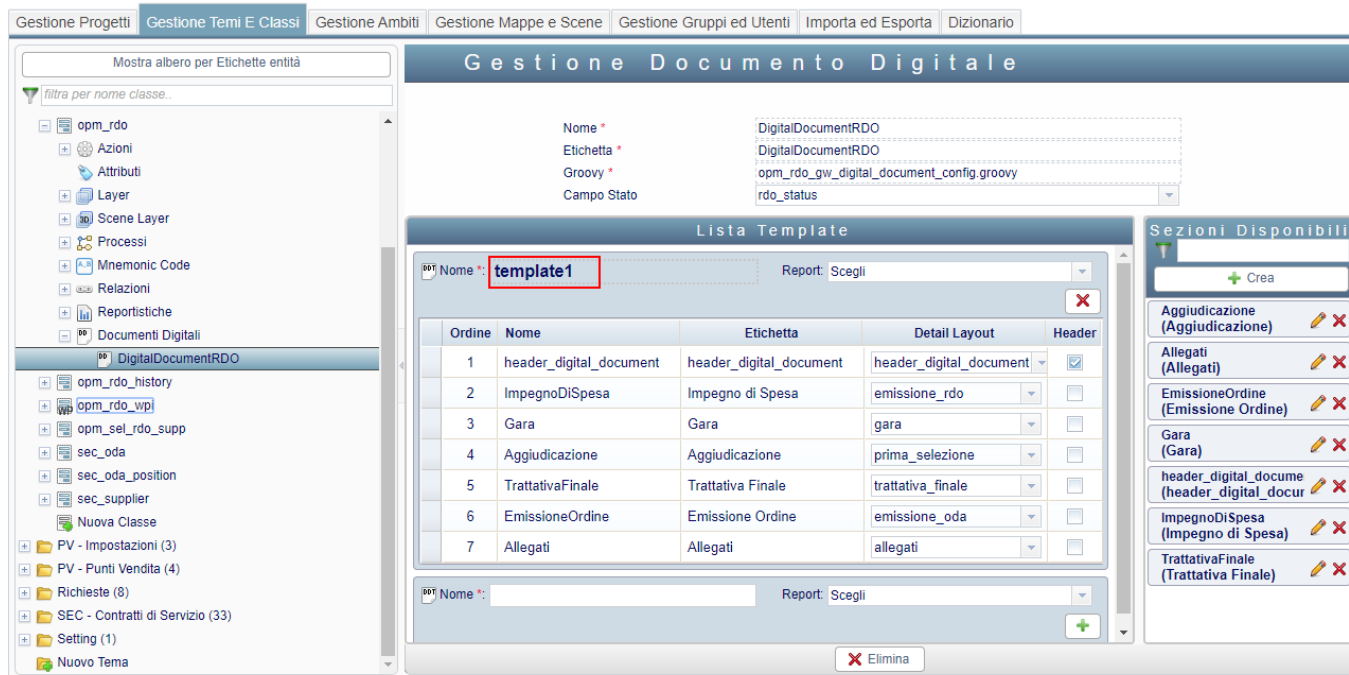
Progetto

Configurazione

I passaggi base per configurare il DDOC sono i seguenti:

1. **progettare** il documento, definendo quali attributi vanno in quali sezioni, e quale deve essere la sequenza corretta
2. definire per ciascuna **sezione** uno o più **Gruppi Attributi**, nei quali incapsulare gli attributi da gestire
3. inserire i Gruppi Attributi in un **Detail Layout specifico** per ogni sezione
4. definire un Report di Classe (opzionale)
5. definire il **Documento Digitale** a livello di classe, attribuendo:
 - Nome: denominazione
 - Etichetta Etichetta attribuita
 - Groovy nome del file groovy, comprensivo di estensione, che verrà eseguito all'apertura del DDOC sul client
 - Campo Stato (opz.) nome del campo in cui viene salvato lo Stato dell'anagrafica
6. configurare almeno un **Template del DDOC**, definendo il numero e l'ordine delle sezioni, e attribuendo un Detail Layout a ciascuna di esse. Designare una delle sezioni come header, ovvero la sezione che sarà visualizzata in modo fisso, nella parte superiore in alto del controllo. Nella

intestazione del template selezionare, se previsto, il Report definito al punto 4

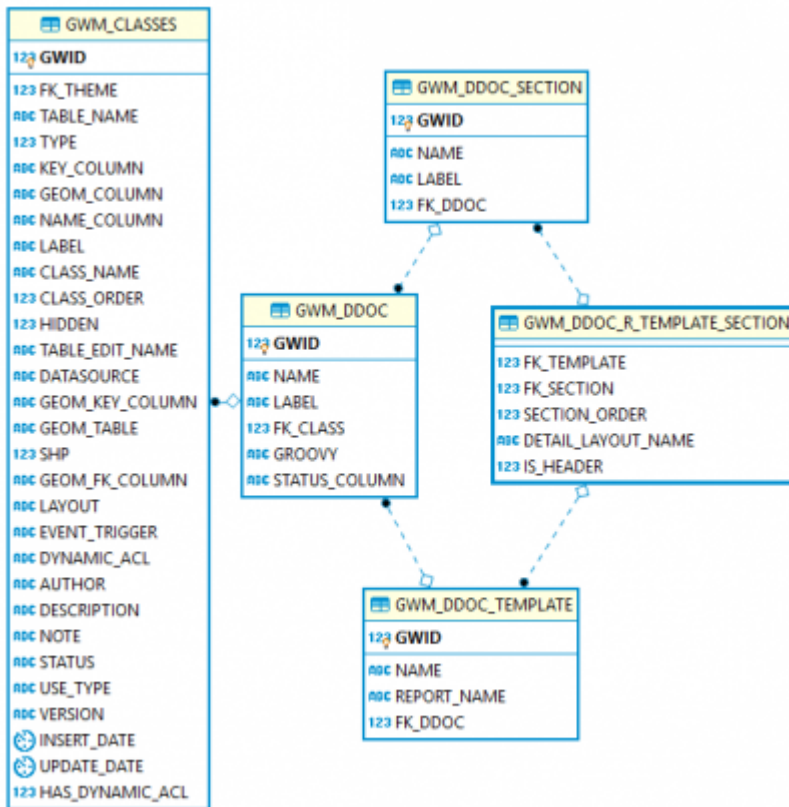


7. configurare tra gli attributi il **widget GW_DIGITAL_DOCUMENT_WIDGET**, configurando tra i parametri il nome del Template

8. inserire il widget in un singolo Gruppo Attributi

9. definire un Detail Layout semplice in cui inserire il Gruppo Attributi con il DDOC, e se opportuno flaggare quest'ultimo come *default*

Modello Dati



Dettaglio delle Tabelle

gwm_ddoc:

- **gwid**: Integer, required, chiave primaria;
- **name**: String, required. Identificativo mnemonico per il DDOC (usato per la GUI nel gwAdmin);
- **label**: String, required. (usato per la GUI nel gwAdmin);
- **fk_class**: Integer, required. Chiave esterna con gwm_classes;
- **status_column**: String, optional. Nome della colonna della classe 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 che ospiterà una class java, la quale dovrà implementare i seguenti metodi dell'interfaccia *com.geowebframework.transfer.objects.digitaldocument.GwDigitalDocumentConfigAbs* oppure estendere la class *com.geowebframework.transfer.objects.digitaldocument.GwDigitalDocumentConfigImpl* (che ne è l'implementazione di default):

```
public Boolean isVisible(String ddocName, String templateName, String  
sectionName, String entityId, String entityName, String entityStatus,  
String gwUser, String gwGroup, Map<String, Object> gwActiveScopesMap)
```

```
public Boolean isEditable(String ddocName, String templateName, String  
sectionName, String entityId, String entityName, String entityStatus,  
String gwUser, String gwGroup, Map<String, Object> gwActiveScopesMap)
```

```
public Boolean isDefaultSelected(String ddocName, String templateName,
String sectionName, String entityId, String entityName, String
entityStatus, String gwUser, String gwGroup, Map<String, Object>
gwActiveScopesMap)
```

L'utente configuratore, di volta in volta, avrà a disposizione tutti i parametri in ingresso per decidere, ritornando *true* o *false*, se abilitare o negare lo specifico permesso. All'interno dei corpi dei metodi da implementare saranno disponibili anche tutti i servizi normalmente disponibili nei groovy di classe (accessibili con notazione *services.identificativo_servizio*).

Se il groovy non venisse dichiarato verrà usata un'implementazione di default che abiliterà tutti i permessi. Da ricordare comunque che continuano ad operare sempre e comunque tutti i meccanismi di ACL, statica e dinamica, che sono propri del dettaglio di classe, e che verranno applicati a monte: solo in caso di esito positivo si procederà anche a valutare il responso della classe nel groovy.

Inoltre, per evitare di riscrivere nel groovy controlli doppi, se per un dato set di input *isVisible()* dovesse ritornare *false* e, per gli stessi input *isEditable()* dovesse invece ritornare *true*, la sezione non verrebbe comunque visualizzata.

Quindi va tenuto presente che il valore ritornato da *isEditable()* sarà valutato solo per le section per cui l'esito di *isVisible()* è stato positivo. Con *isDefaultSelected* si decide quale section il DDOC presenterà appena aperto. Anche *isDefaultSelected()* verrà valutato solo se *isVisible()* e *isEditable()* abbiano precedentemente ritornato esito positivo. Di default viene presentata la prima section editabile, o la prima in assoluto se sono tutte in sola visualizzazione.

gwm_ddoc_section:

- **gwid**: Integer, required, chiave primaria;
- **name**: String, required. Identificativo mnemonico per la section (usato per la GUI nel gwAdmin);
- **label**: String, required. (usato per la GUI lato client, sarà l'intestazione della sezione di sinistra);
- **fk_ddoc**: Integer, required. Chiave esterna con *gwm_ddoc*.

gwm_ddoc_template:

- **gwid**: Integer, required, chiave primaria;
- **name**: String, required. Identificativo mnemonico per il template (usato per la GUI nel gwAdmin) ;
- **fk_ddoc**: Integer, required. Chiave esterna con *gwm_ddoc*.

gwm_ddoc_r_template_section:

- **gwid**: Integer, required, chiave primaria;
- **name**: String, required. Identificativo mnemonico per il template (usato per la GUI nel gwAdmin) ;
- **fk_template**: Integer, required. Chiave esterna con *gwm_ddoc_template*;
- **fk_section**: Integer, required. Chiave esterna con *gwm_ddoc_section*;
- **section_order**: Integer, required. Determinerà lato client l'ordine con le quale verranno presentate le sezioni;
- **detail_layout_name**: String, required. Determinerà quale detailLayout, fra quelli configurati per la classe, verrà usato per rappresentare la sezione nello specifico template;

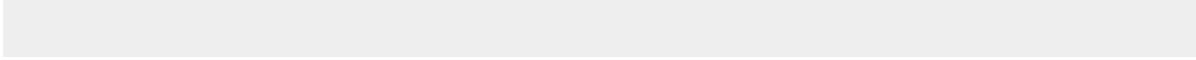
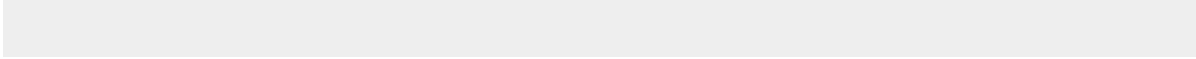
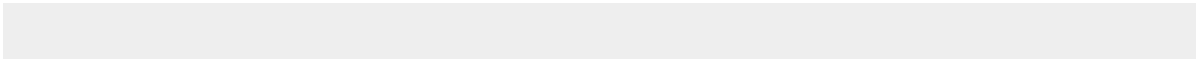
- **is_header**: Integer, optional. Determina quale delle section all'interno del template sarà usato come intestazione del DDOC. Ce ne può essere solo uno per template.

ESEMPIO DI CONFIGURAZIONE GROOVY

```
public class TestGwDigitalDocumentConfigImpl extends
com.geowebframework.transfer.objects.digitaldocument.GwDigitalDocumentConfig
Impl {
    public Boolean isVisible(
        String ddocName,
        String templateName,
        String sectionName,
        String entityId,
        String entityName,
        String entityStatus,
        String gwUser,
        String gwGroup,
        Map<String, Object> gwActiveScopesMap
    ){
        return sectionName!='test_section_5';
    }
    public Boolean isEditable(
        String ddocName,
        String templateName,
        String sectionName,
        String entityId,
        String entityName,
        String entityStatus,
        String gwUser,
        String gwGroup,
        Map<String, Object> gwActiveScopesMap
    ){
        return sectionName!='test_section_2';
    }
    public Boolean isDefaultSelected(
        String ddocName,
        String templateName,
        String sectionName,
        String entityId,
        String entityName,
        String entityStatus,
        String gwUser,
        String gwGroup,
        Map<String, Object> gwActiveScopesMap
    ){
        return sectionName=='test_section_4';
    }
}
```

API js

Avendo disponibile il *ddocWidget*, è possibile fa riferimento alle seguenti proprietà e function espote:

- property:
 - **ddocSectionHeader**, di tipo Object.

 - **ddocSections**, di tipo Object[].

- function:
 - **getDDOCSection(name)**, function che, dal parametro *name* (*String*), ritorna un Object.

 - **isDDOCSectionVisible(name)**, boolean
 - **isDDOCSectionEditable(name)**, boolean
 - **isDDOCSectionDefaultSelected(name)**, boolean
 - **isDDOCSectionHeader(name)**, boolean

Si potrebbe volere, per esempio, in un'azione di dettaglio pronto, eseguire delle operazioni solo se una certa *ddocSection* risultasse *editabile* in quel momento (in base alla valutazione dinamica del *groovy*).

```
var name = 'ddoc_section_name'; //known
var isEditable = ddocWidget.isDDOCSectionEditable(name);
if(isEditable){
    //do stuff
}
```

From:
<https://wiki.geowebframework.com/> - GeowebFramework

Permanent link:
https://wiki.geowebframework.com/doku.php?id=gwusermanual:interface:widget:gw_digital_document_widget&rev=1642605494

Last update: 2022/01/19 16:18

