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 vidimato 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.

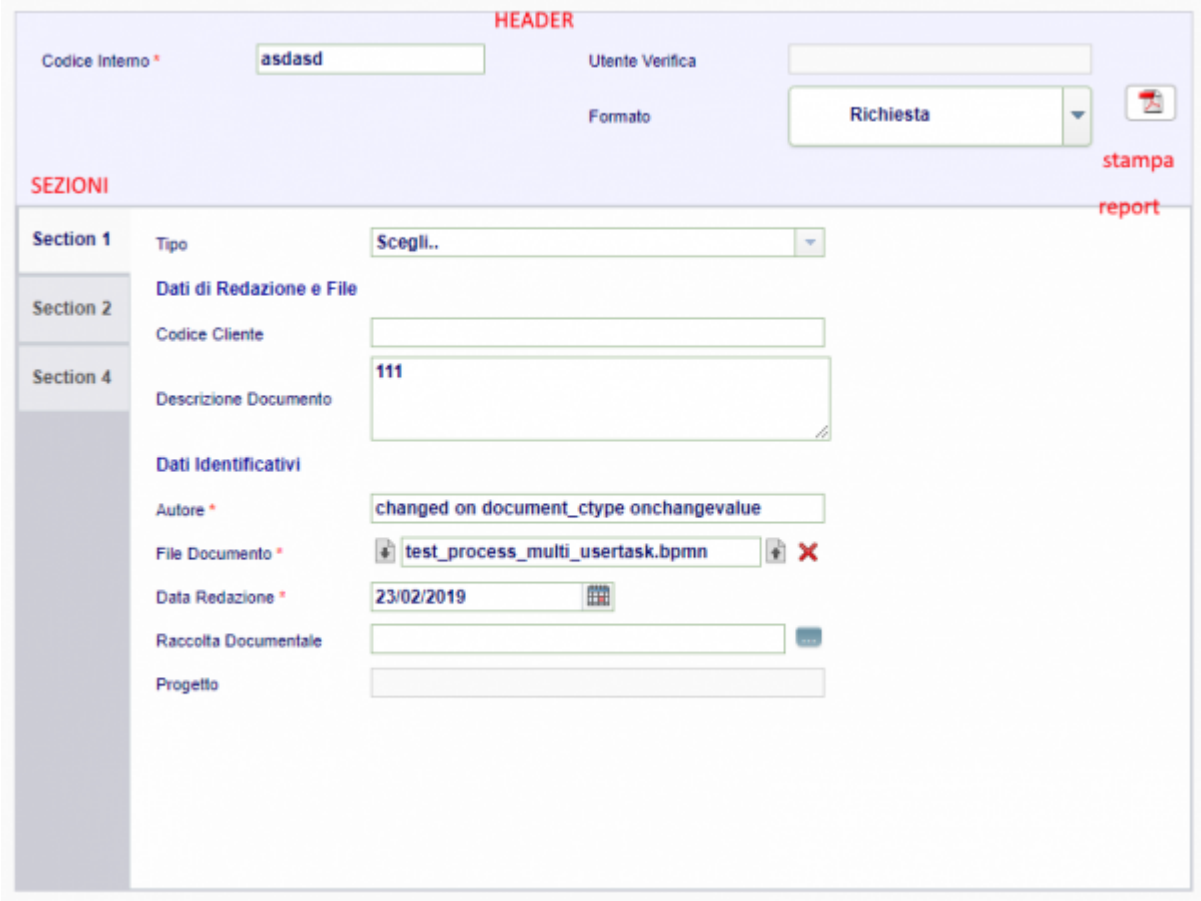
Dal punto di vista pratico, inserendo gli attributi in determinate sezioni, si ha la possibilità di definire delle logiche autorizzative a livello di attributo, superando il limite di Geoweb che organizza le autorizzazioni a livello di classe.

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.



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 slezionare, 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

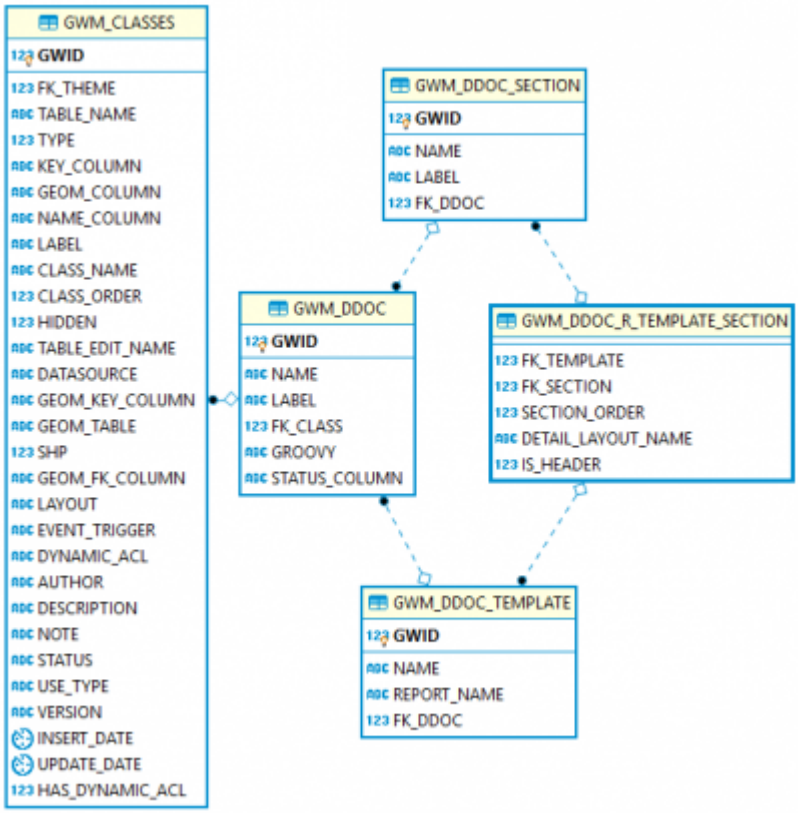


TABELLE IN DETTAGLIO

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 (accedibili 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.

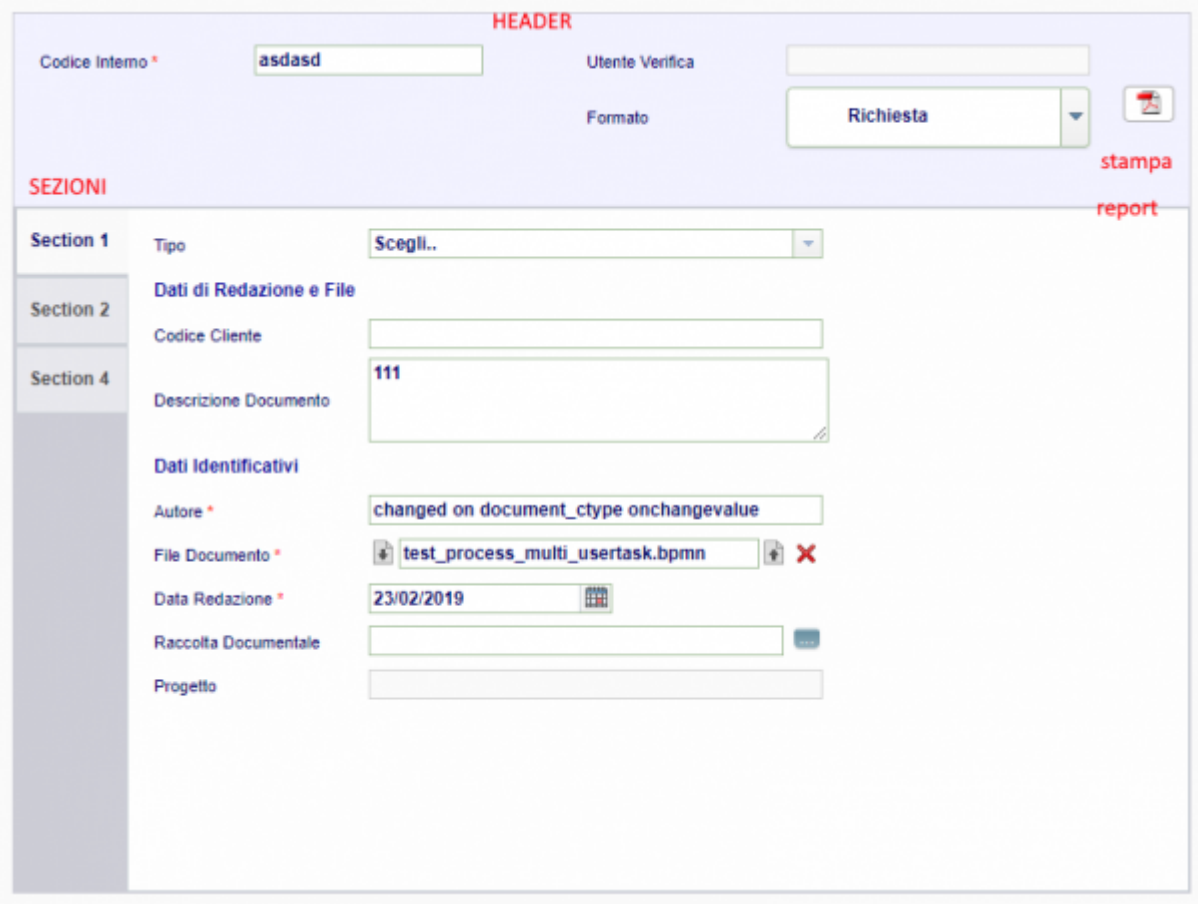
ESEMPIO DI CONFIGURAZIONE GROOVY

```
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';
    }
}
```

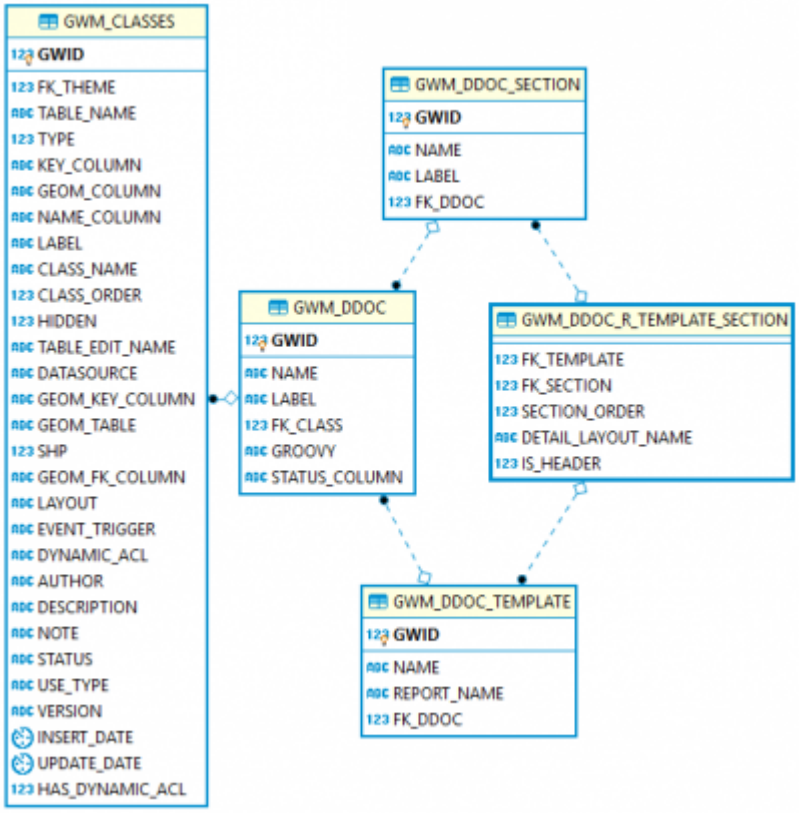
OLD

Questo nuovo componente di Geoweb nasce con l'obbiettivo di dare maggiore evidenza alla rappresentazione digitale del documento, rispetto a come viene attualmente declinata all'interno di Geoweb. Al momento la rappresentazione del documento è strettamente legata alle logiche di funzionamento e di visualizzazione della scheda dettaglio di classe e talvolta della scheda di processo (procedura), che sono funzionalità di base del framework. Queste tendono, in maniera variabile rispetto alla specifica configurazione, a nascondere informazioni base del documento, o comunque sia a non darle il giusto risalto. Il problema è che l'utente non capisce chiaramente la differenza quando sta visualizzando la scheda di dettaglio di una classe per la gestione di un documento digitale, rispetto a quando sta visualizzando, per esempio, la scheda di un asset.



Lato GUI, si procederà quindi a creare una precisa struttura standard del **DDOC**, sempre riconoscibile dall'utente, che sarà divisa in sezioni (**section**). La struttura in sezioni è realizzata come un simil 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 con controllo assoluto, tramite l'implementazione di una class

java in un file .groovy (vedi sotto). Le sezioni del documento possono essere combinate in maniere differenti 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 name in ordine alfabetico). La scheda DDOC, nella parte superiore, sopra le *section*, presenta una intestazione fissa, contenente uno specifico detailLayout. Questa è di fatto una *section* speciale, marchiata con lo specifico flag **is_header** (vedi sotto) e ne può esistere solo una per template. Questa è concepita per essere usata per mostrare i widget che sono in qualche modo trasversali a tutte le section, e che devono essere sempre in primo piano (per esempio widget codici, codici parlanti, stati vari, etc..). Quindi essa è sempre visibile a prescindere da quale section viene selezionata sotto. A destra della section di intestazione può opzionalmente essere presente un bottone **stampa report** pdf (il name della report è specificato nella tabella *gwm_ddoc_template*. Se omesso non viene visualizzato alcun bottone).



MODELLO DATI

TABELLE IN DETTAGLIO

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 od implementare i seguenti metodi dell'interfaccia *com.geowebframework.transfer.objects.digitaldocument.GwDigitalDocumentConfigAbs* od 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 (accedibili con notazione *services.identificativo_servizio*). Se il groovy non venisse dichiarato verrà usata un'implementazione di default che abilità 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.

NOTE Per visualizzare il documento digitale si è preferito sviluppare un nuovo widget e non un nuovo tipo di scheda, per specifiche ragioni. Essendoci l'esigenza, dettata dalle nuove linee di sviluppo, di mostrare assieme al documento anche tutti le istanze di processi che sono in qualche modo collegate ad esso, il documento digitale è inglobato in un widget. Tale widget può quindi essere disposto con la massima flessibilità, facendo leva sul meccanismo di layout della classe di geoweb, accanto ad altri specifici widget che si occuperanno di mostrare e gestire i processi collegati al documento (oggetto di sviluppi paralleli). Questo nuovo widget ha sostanzialmente il solo parametro *template*.

ALTRE CONSIDERAZIONI DI SVILUPPO Il solo parametro template è sufficiente in quanto il DDOC è recuperato in automatico facendo leva sul fatto che ne può esistere solo uno per classe. Dal DDOC è recuperato il file .groovy che codifica la configurazione del widget

Come sviluppo futuro si potrebbero aggiungere i seguenti parametri:

- *ddoc_class_name*;
- *local_field*;
- *ddoc_field* con lo scopo di visualizzare dentro un dettaglio di classe il documento i cui dati sono però ospitati in un'altra classe. In particolare, una specifica fase di una scheda workflow di tipo procedure, potrebbe ospitare un widget gwDigitalDocument, che faccia riferimento a dati ospitati, come giusto, su un'apposita classe.

ESEMPIO DI CONFIGURAZIONE XML WIDGET

```
<gwDigitalDocumentWidget>
    …
    <template></template>
</gwDigitalDocumentWidget>
```

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';
    }
}
```

Template

Il template è la definizione della tipologia di rappresentazione delle informazioni del Documento Digitale. Nel Template sono definibili delle Sezioni che sono associate a specifici Detail Layout della classe che include il Document. Sono configurabili più template all'interno della stessa classe.