Creazione di Report

Geoweb si avvale di Jasper Report come motore di Reportistica, il quale dispone di uno strumento di authoring potente ed efficace quale Jaspersoft Studio (che sostituisce iReport).

Le report costruite con **Jaspersoft Studio/iReport** possono essere prodotte e utilizzate da Geoweb, a seconda del componente specifico, nei seguenti formati:

- **PDF** [in: Report Esportabili, Report di Classe, Report di Cruscotto]
- **XLSX** [in: Report Esportabili]
- **DOCX** [in: Report Esportabili]
- **XLS** [in: Report di Classe, Report di Cruscotto, Report di analisi]
- **RTF** [in: Report di Classe]
- **HTML** [in: Report HTML]

E' possibile utilizzare le funzionalità di Jasper Report per ottenere in Geoweb le features seguenti:

- Report Esportabili:
  - deprecano le Report di Classe e le Report di Cruscotto
  - comprendono sia le report di prodotto, sia report custom, entrambe configurabili dal Solution Manager
  - sono rese fruibili in apposite schede cruscotto specifiche per l'utente finale
  - la loro configurazione ed abilitazione è operata dal Solution Manager
  - le informazioni relative alle report sono sullo schema dati
- Report di Classe @Deprecated from 4.7.1:
  - realizzazione di formati di stampa personalizzati per stampare elenchi di dati dalle anagrafiche gestite (lista dei record selezionati)
  - realizzazione di formati di stampa personalizzati per stampare una scheda del record corrente (dettaglio)
- Report di Cruscotto @Deprecated from 4.7.1: pannello di presentazione ('scheda') di reportistiche predefinite e studiate per esporre i dati della tematica gestita. Le report possono essere costruite in XLS o in PDF, e possono essere eseguite con la richiesta a run time, all'utente che le esegue, di parametri utente (ad esempio intervalli di date entro cui eseguire l'estrazione dati). Tali pannelli possono essere integrati direttamente da Client, nel caso in cui un utente molto esperto, con opportune credenziali, voglia contribuire alla definizione di proprie report o modificare/aggiornare le report esistenti.
- Report di analisi: particolari tipologie di report excel che prevedono un foglio 'dati', nel quale vengono riversati a run-time da Geoweb, i dati del recordset corrente (lista dei record selezionati a livello di classe), e uno o più fogli templates nelle quali sono già preparati grafici e tabelle Pivot personalizzati e disegnati per analizzare i dati del recordset stesso.
- Interfacce personalizzate HTML (Report HTML): si tratta di pagine disegnate con iReport e renderizzate in formato HTML, che Geoweb è in grado di aprire come 'scheda' e a cui è possibile agganciare come hyperlink delle API Javascript, che permettono di eseguire azioni in Geoweb. In pratica, a tutti gli effetti, queste pagine diventano una interfaccia personalizzata che integra i casi d'uso standard applicativi resi disponibili da Geoweb per tutte le classi definite (apertura record in lista, filtro, clic su lista per apertura del dettaglio, ecc.). Di seguito un esempio delle azioni che possono essere eseguite da una report HTML:

- apertura di schede di dettaglio o di lista
- apertura di report
- esecuzione di procedure
- creazione di nuovi record
- ecc.

Report Standard (STAMPA SCHEDA)

La report “standard” di dettaglio o (“STAMPA SCHEDA”) è abilitata in automatico per tutti i dettagli delle classi di geoweb. I file jasper sono integrati direttamente nel framework.

Disabilitare la Report Standard (STAMPA SCHEDA)

Per **disabilitare** la report automatica “Stampa scheda”, aggiungere al configuration.properties:

```
hideReportSingleView=true
```

Con questa impostazione, per **abilitare** lo stampa scheda **solo** per una o più classi, è sufficiente impostare la seguente azione (di dettaglio):

```
openGenericReport({'href':
gwContextPath+'/rest/'+project_name+'/'+data.className+'/report/record/'+dat
a.itemDB[data.keyColumn]+'/ReportSingleView.pdf'});
```

Meccanismi di integrazione con Geoweb

Parametri delle sub report

Nel caso che vengano utilizzate delle sottoreport, o caricate delle immagini da filesystem, è necessario definire in jasper report dei parametri che vengono riconosciuti e passati a runtime da Geoweb

Parametro	**tipo**	**utilizzo**
SUBREPORT_URL	java.lang.String	usato per indicare il percorso delle subreport
SUBREPORT_CONN	Object	usato per passare la connessione alle subreport (o immagine)

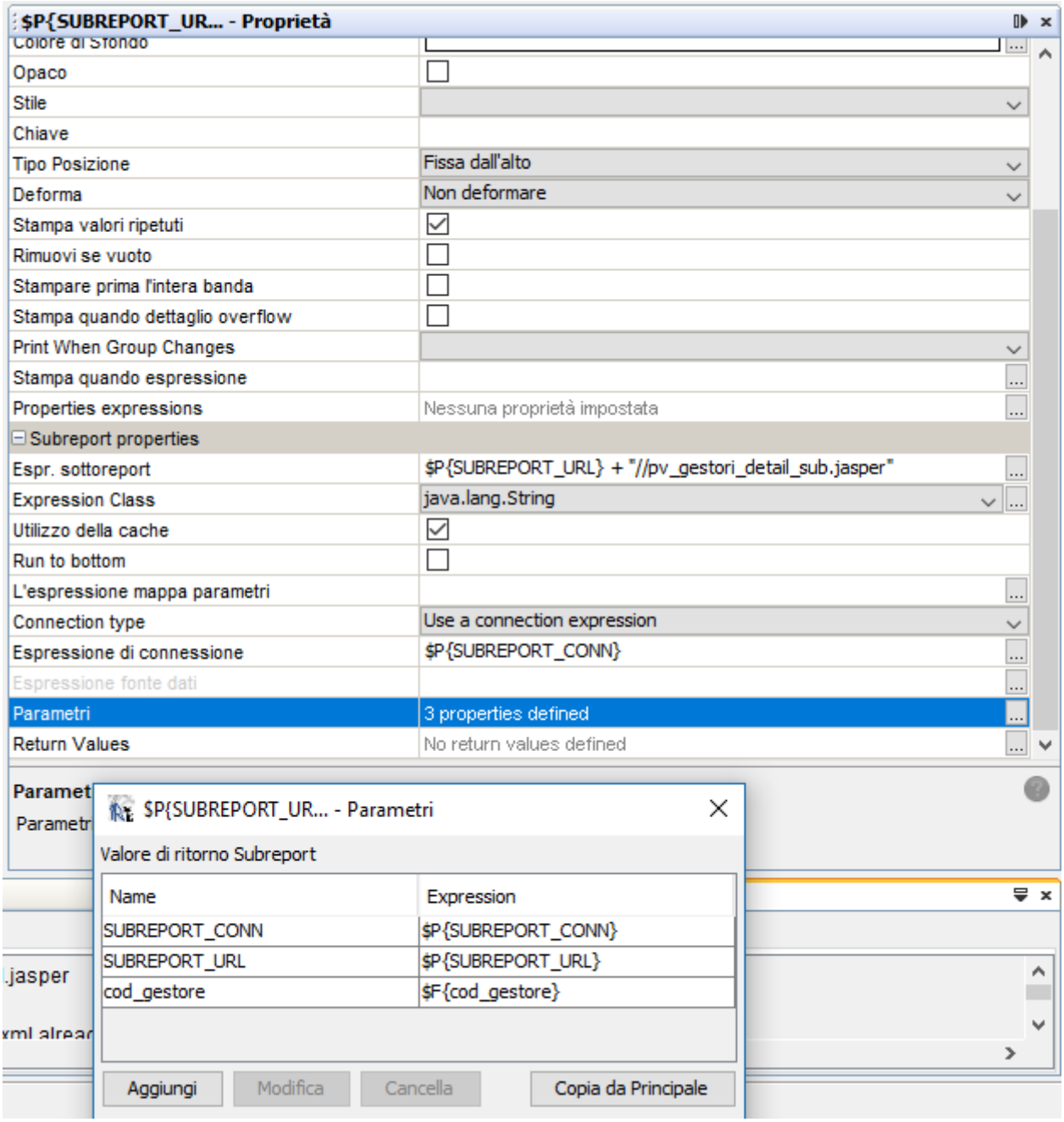
Questi parametri vanno:

- definiti nella Report principale
- ~~definiti nelle singole subReport~~ (non necessario)
- passati come proprietà nel controllo SubReport (vedi esempio sotto):
  - in chiaro nella voce 'Parametri'
  - nell'espressione di ricerca del percorso su filesystem della subreport (espressione

subreport)
    - nell'espressione da passare alla connessione (espressione connessione. Attenzione a specificare in 'Connection Type' il valore: '**use a connection expression**'.
- nel caso di immagini caricate da filesystem va utilizzata la SUBREPORT_URL nell'espressione di ricerca del file immagine, analogamente a come visto sopra per la subreport

ATTENZIONE: evitare di utilizzare l'anteprima di i*REPORT che in locale funziona non sempre correttamente e fa perdere tempo.
Mettere le Report direttamente nel server e testare tramite GeoWeb.
EVITARE anche lo Wizard di iReport.

Esempio:



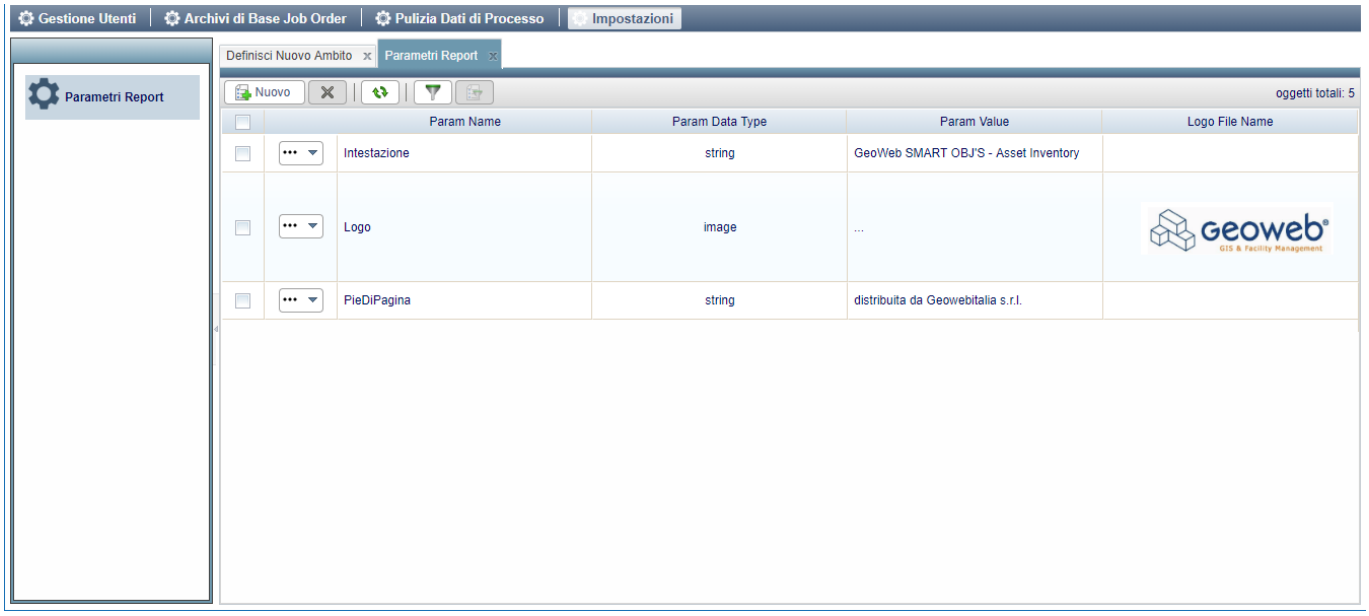
Elementi parametrici gestibili da client

Geoweb implementa nello schema Dati una particolare classe, denominata ***gw_global_report_param***, che permette di definire da Client elementi del report, quali ad esempio il LOGO stampato nelle report, il testo da utilizzare come Intestazione o Piè di Pagina nella report.

I vantaggi di questa funzionalità sono i seguenti:

- una immagine aziendale definita come logo può essere modificata direttamente dall'utente finale, così come e un cambio di indirizzo o ragione sociale inserito nella intestazione o piè di pagine delle report.
- la possibilità di utilizzare in maniera diffusa tali parametri da tutte le report definite a sistema, eviteranno di modificare queste informazioni riaprendo i templates delle report una ad una.

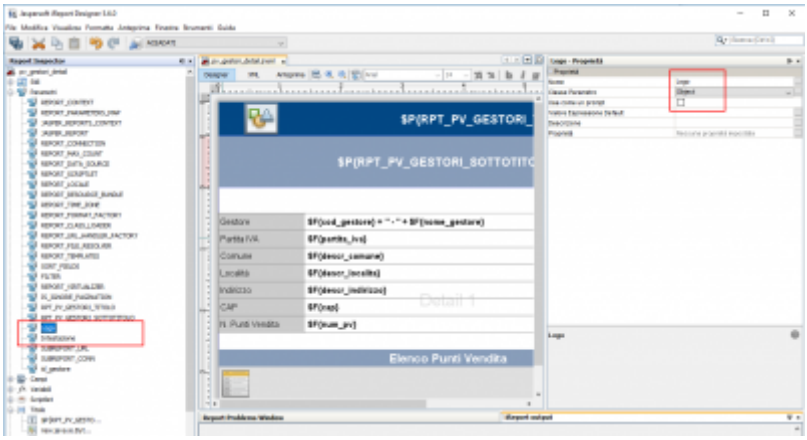
La classe presenta dei record definiti a titolo di esempio, che l'utente può andare a modificare, o integrare.



I dati messi a disposizione sono i seguenti:

- ***param_name***: nome del parametro, è l'identificativo univoco del parametro che va ridefinito e riutilizzato nella Report Jasper. E' importante annotare che il param_name è case sensitive, quindi va rispettata la sintassi utilizzata, spazi, maiuscole, minuscole;
- ***param_data_type***: accetta solo due valori: 'string'/'image'. rispettivamente corrispondono, in JasperReport, a parametri di tipo *java.lang.String* / *Object* ;
- ***param_value***: valore del parametro, utilizzato solo se param_data_type='string', è il testo dei parametri definiti come stringhe (es. Intestazione);
- ***logo_file_name***: file immagine da utilizzare, ad esempio come logo, utilizzato solo se param_data_type='image';

In iReport i parametri vanno definiti con le regole qui sopra enunciate, e quindi inseriti nella report come un normale parametro.



Per quanto riguarda l'immagine del Logo, viene riportata la sintassi da utilizzare nel campo 'Espressione immagine':

```
new java.io.ByteArrayInputStream((byte[])$P{Logo})
```

IMPORTANTE: il parametro del Logo deve essere definito come *Object*.

Opzioni e Utilità varie

Stampa in assenza di dati da stampare

Opzione **Quando non ci sono dati Stampa**: é importante impostare questa opzione per tutte le report, in maniera tale che queste non diano errori se la report viene chiamata e non ci sono dati da stampare. L'impostazione corretta è **Tutte le sezioni senza Dettaglio**.

Concatenazione di campi di cui almeno uno potrebbe essere Null

Nel caso in cui si voglia concatenare all'interno di un campo di testo più campi di cui almeno uno potrebbe assumere valore nullo, va utilizzata la seguenti sintassi di concatenazione:

```
($F{campo1}!=null?$F{campo1}:"")+" "+($F{campo2}!=null?$F{campo2}:"") +"
"+($F{campo3}!=null?$F{campo3}:"")
```

Data corrente e Numero di Pagina

Nel caso in cui si voglia visualizzare all'interno di una report la data corrente ed il numero di pagina: inserire nella sezione Piè di Pagina:

1. data corrente (menù Palette –> Strumenti)
2. numero di pagina o pagina X di Y o Pagina totale
3. per la data, configurare nella finestra delle proprietà il formato (pattern)

Concatenazione campo String con campo Date

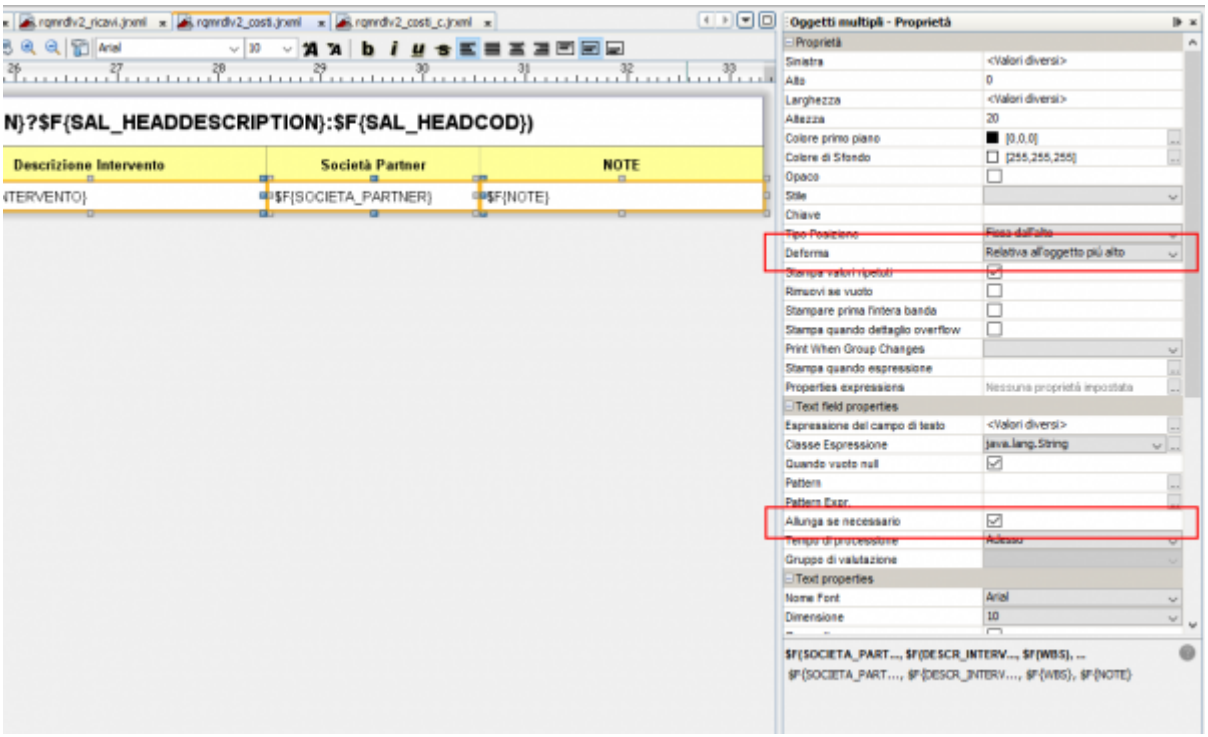
Nel caso in cui si voglia concatenare all'interno di un campo di testo un campo (field) di tipo stringa ed un campo (field) di tipo data è necessario effettuare la conversione di formato secondo la seguente sintassi di esempio:

```
($F{field_string}!=null?$F{field_string}:"") + " " +
($F{field_date}!=null?(new
SimpleDateFormat("dd/MM/yyyy")).format($F{field_date}):"")
```

Aumentare altezza delle righe per comprendere stringhe di lunghezza variabile (es. descrizioni)

Nel caso che si debbano stampare record in tabella, e una o più colonne contengono valori di lunghezza variabile, può essere necessario aumentare l'altezza delle righe perché ne sia visualizzato il contenuto per intero. In questo caso occorre:

- selezionare tutti i campi del dettaglio
- impostare a true il parametro 'Allunga se necessario'
- Impostare il parametro 'Deforma' a 'Relativa all'oggetto più alto'.

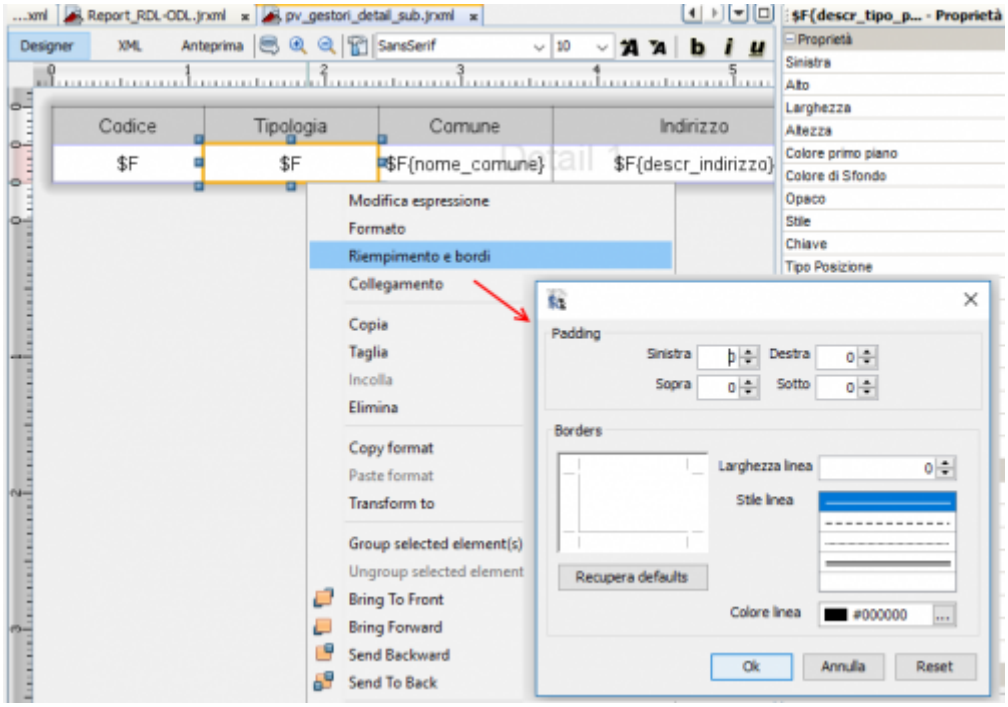


Report con output Excel

Nelle Report che devono produrre un file Excel, è opportuno tenere presente alcune regole di formattazione che permettono di ottenere un file di output ben fatto:

1. settare nelle proprietà del report “Ignora Paginazione” = SI
2. eliminare i margini di pagina

3. nel posizionare i campi uno a fianco all'altro fare attenzione che siano perfettaemnte adiacenti tra loro
4. adottare regole di formattazione dei bordi tali che non ci siano bordi sovrapposti tra due campi adiacenti, sia in verticale che in orizzontale.



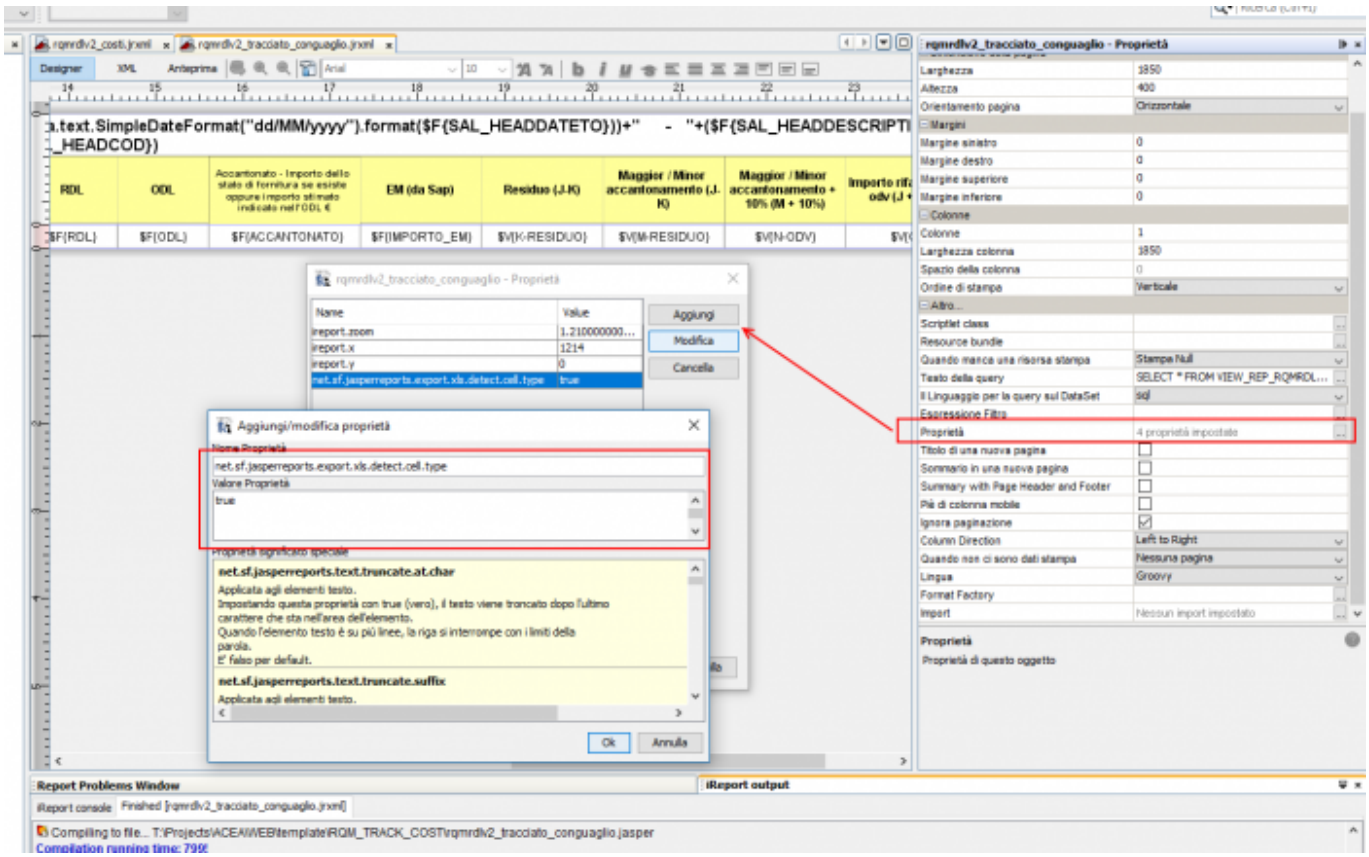
Riconoscimento automatico formato Celle Numeriche

Per far si che i campi Numerici vengano riconosciuti da Excel come celle Numeriche, visualizzare l'XML del report e nella sezione property aggiungere la seguente riga:

```
<property name="net.sf.jasperreports.export.xls.detect.cell.type"
value="true"/>
```

Oppure impostare la proprietà da interfaccia com nella immagine visualizzata sotto.

N.B. settare la proprietà sia sul report principale che sui sottoreport



Guida per ottenere Report Multi Pagina / Multi Foglio in xls

Di seguito una piccola guida scritta specificatamente per produrre report in excel su più Fogli di lavoro:

guida_per_ottenere_report_multipagina-multifoglio_in_excel.pdf

Cambiare LOCALE (localizzazione) su Ireport

Un oggetto Locale in java rappresenta una regione geografica, politica o culturale specifica.

Il Locale decodifica i valori delle date, valute, separatori numerici, ecc.

Ogni Locale decodifica in modo diverso, ad esempio nel locale Americano *(en_US)*, il valore € sarà sostituito da $, la virgola viene utilizzata come separatore delle migliaia ed il punto come separatore dei decimali.

Il problema si presenta se Ireport o Java sono installati in un server con lingua o localizzazione diversa da quella Italiana.

Occorre cambiare il parametro **REPORT_LOCALE** (ovvero, il parametro del locale), come segue:

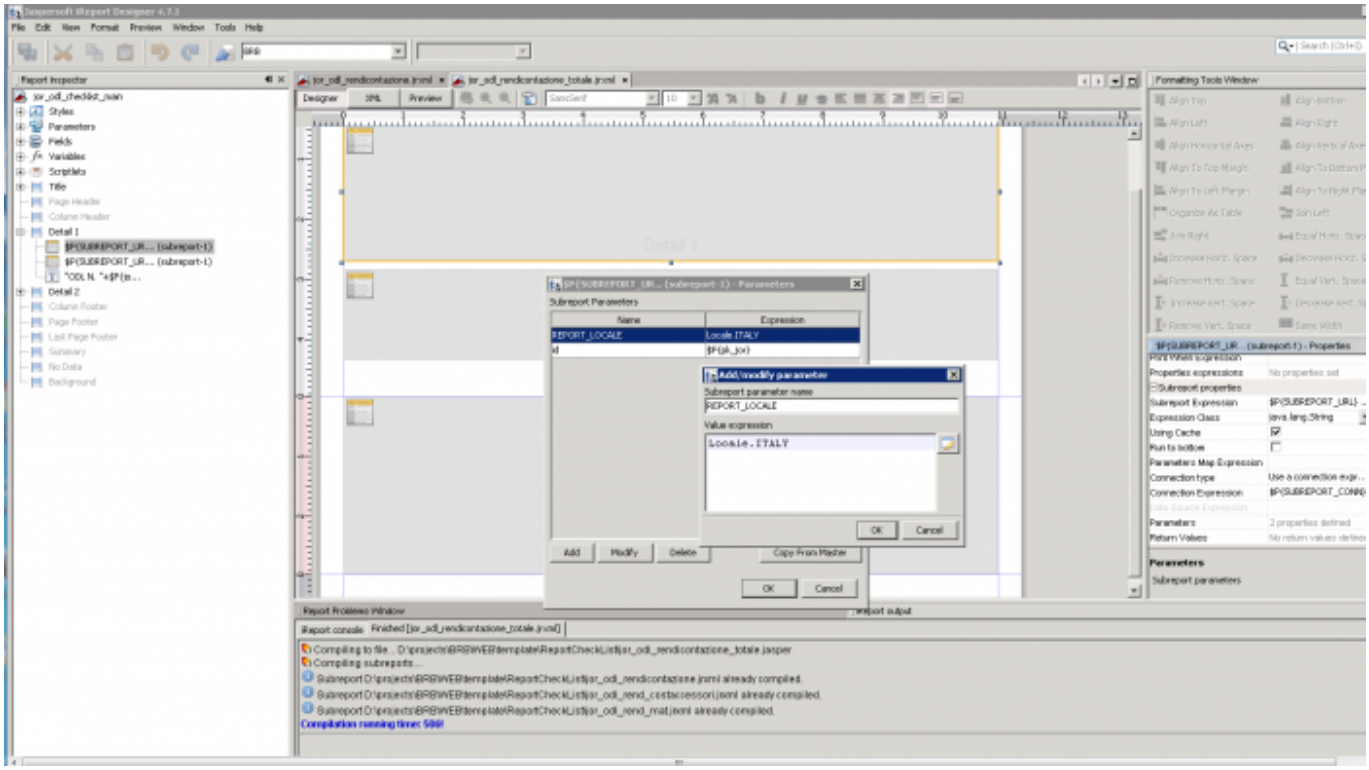
- Creare Main Report;

- Creare Subreport (sottoreport);
- Entrare nelle *proprietà* della sottoreport e selezionare *Parametri*.
- Aggiungere il parametro (nelle proprietà del sottoreport) **REPORT_LOCALE** con il valore **Locale.ITALY** (vedi figura).

EDIT: Soluzione 2, lavorando nella Report,si può forzare nella Text Field Espression direttamente la definizione come da esempio:

new java.text.DecimalFormat("€#,##0.00", new java.text.DecimalFormatSymbols(java.util.Locale.ITALY)).format($V{Variable}) + " €"

od anche (in caso di possibili valori null): $F{Field}!=null?(new java.text.DecimalFormat("€ #,##0.00", new java.text.DecimalFormatSymbols(java.util.Locale.ITALY)).format($F{Field})):""



Inserimento di immagini tipo SVG su report

Di seguito i passaggi per poter inserire un'immagine di tipo svg all'interno di una report con ireport.

1. Inserire un oggetto di tipo immagine nel dettaglio della report;
2. Aprire l'xml editor e modificare la riga di riferimento dell'immagine con la seguente:

<image scaleImage="RetainShape"><reportElement x="700" y="0" width="100" height="20" uuid="d037faeb-f4c2-4f81-904b-9d1020597ed3"/><imageExpression><![CDATA[net.sf.jasperreports.renderers.BatikRenderer.getInstance(new java.io.File("\\\\wingeoweb4\\c$\\projects\\ACEA\\WEB\\template\\svg\\" + $F{COD_CAB} +".svg"))]]></imageExpression></image>

Nell’esempio sopra riportato sono stati creati files di immagini di tipo svg che hanno come nome il codice di un singolo elemento presente in tabella, in questo modo ad ogni codice cabina viene visualizzato il relativo file di immagine. — *Matteo De Salvo 2019/01/14 15:03*

Inserimento di immagini salvate in un campo del DB

Di seguito i passaggi per poter inserire nel report una immagine salvata su DB (es. la foto dell'edificio) in una report. IMPORTANTE: quanto descritto funziona correttamente con DB Postgres, e con tipo di campo *bytea*, ma non è garantito con Oracle e SQL Server.

1. Inserire un oggetto di tipo immagine nel dettaglio della report;
2. Verificare che il campo sia definito come *java.lang.Object*.
3. modificare la proprietà 'Espressione immagine' come segue:

```
new java.io.ByteArrayInputStream((byte[])$F{<nomeCampoBytea>})
```

Prevenzione problematiche immagine

Esiste un bug, che si manifesta con una mancata erogazione da parte del server delle immagini di una o più report html, dopo che si esegue una qualche operazione che ne forza il refresh (direttamente o indirettamente).

Anni fa era, per sanare la problematica che i grafici non si aggiornavano refreshando la report dopo aver modificato il dataset sottostante, era stato introdotto un parametro che eludeva sempre il meccanismo di caching del browser.

Quindi tutte le immagini di report html, che fossero immagini fisse o dinamiche (di grafici costruiti sul dataset), venivano risolte con URL sempre diversi, anche per la stessa immagine della stessa report aperta in due momenti diversi.

Ci sono situazioni in cui le immagini con nuovo URL vengono erogate con ritardo o non proprio erogate. Non è ben chiara la problematica, ma sembra essere condizionata negativamente dalla scarsità di risorse del server e/o dal contemporaneo caricamento/refresh di più report nello stesso momento.

Per sanare tale problematica sono stati introdotti due meccanismi alternativi:

1. UTILIZZO URL ASSOLUTI CHE PUNTANO AI CONTENUTI STATICI
2. UTILIZZO PARAMETRO REPORT enableCache

Quale soluzione usare?

La soluzione 1 è preferibile quando la stessa immagine è usata nella stessa report, solo html, dove si susseguono più righe, oppure dove è presente in molte report html. E' preferibile perché l'immagine viene servita una sola volta e poi sempre recuperata dalla cache del browser.

La soluzione 2 è più rapida come configurazione, ma andrebbe usata solo in report non particolarmente complesse dove le immagini non sono ripetute in molti posti, e l'effetto è che ogni immagine verrebbe richiesta ogni volta con una nuova chiamata.

Nei casi di report dove sono configurati grafici, non è necessaria alcuna modifica.

Maggiori dettagli sulla problematica, anche di natura tecnica, disponibili qui:

https://gitlab.com/geowebframework/geowebframework/-/issues/931

1 UTILIZZO URL ASSOLUTI CHE PUNTANO AI CONTENUTI STATICI

Soluzione forse più radicale, ma più corretta e parca di risorse, è quella di impostare l'immagine specificando l'URL completo. In questa maniera l'immagine non verrà gestita da jasper Report, che semplicemente la riverserà cosi come impostata in una regola css background-image, che verrà risolta dal browser tramite i soliti meccanismi di caching.

Nota bene: La seguente soluzione è applicabile solo se la report dovrà essere erogata esclusivamente tramite browser, e non generata in pdf (in quanto le immagini non vengono servite).

L'URL viene costruito automaticamente rispettando i pattern di percorso standard per recuperare immagini dai contenuti statici:

```
${STATIC_CONTENTS_URL}+"images/image.png"
```

oppure

```
${IMAGES_URL}+"image.png"
```

A tale scopo vengono introdotti due nuovi parametri passati alla report:

STATIC_CONTENTS_URL: path fino alla folder dei contenuti statici 'WEB/' **IMAGES_URL**: path fino alla folder images dei contenuti statici 'WEB/images/'

NOTA BENE

Assicurarsi che la spunta **Posticipato**/**Caricamento ritardato** sia abilitata in JasperStudio.

L'impostazione Usa Cache non è più rilevante, e può essere lasciata a true.

Trick per vecchie implementazioni: pre 4.6.3

In vecchie installazioni pre 4.6.3, tale meccanismo può essere riprodotto tramite sola configurazione, aggiungendo dei parametri di report costruiti in maniera prestabilita come da esempio:

1. STATIC_CONTENTS_URL
2. IMAGES_URL

codice js esempio:

```
var reportUrl = 'path/To/file_name.jasper';
var parametersMap = {
    reportParameters: {
        STATIC_CONTENTS_URL: gwContextPath+'/resources/'+gwRevision +'/',
        IMAGES_URL: gwContextPath+'/resources/'+gwRevision +'/images/'
    }
};

var title = 'title';
openGwHtmlReportTab(reportUrl, parametersMap, title);
```

utilizzo parametri per l'url in JasperStudio:

```
$P{IMAGES_URL}+"my_image.png"
```

che verrà sostituito a runtime in:

```
"/webclient/resources/8c1572f9fffd9862c1d10c6c76544930ec166d5a/images/my_ima
ge.png"
```

2 UTILIZZO PARAMETRO REPORT enableCache

Nuovo parametro enableCache, Boolean, default false.

Quando è messo a true permette l'utilizzo del meccanismo di caching del browser, ma per tutte le immagini.

Nota bene: Quindi non va usato quando ci sono I grafici

Esempio:

```
var reportUrl = 'path/To/file_name.jasper';

var parametersMap = {

       enableCache: true

};

var title = 'title';

openGwHtmlReportTab(reportUrl, parametersMap, title);
```

Leggere valori di campi JSON

Per le Report di classe, Geoweb come esegue preventivamente la query sui campi di una classe (vale sempre la convenzione degli attributi aggiunti ad almeno un gruppo attributo). Nel record-set che arriva alla report le key di un particolare campo JSON hanno sempre l'alias nella forma descritta precedentemente nei vari flussi import export:

```
[json_field_name]+"|"+[key]
```

Come sempre si ricorda che in fase di design è bene definire una query sulla tabella/vista collegata alla classe, altrimenti non si riesce neanche ad eseguire una anteprima. Tuttavia occorre ricordare che tale query non verrà utilizzata a run-time, perché come già detto sopra il record-set viene passato direttamente da Geoweb.

Con i dati JSON le cose si complicano un po' perché ogni database ha il proprio modo ci costruire la query su dati di tipo JSON. Geoweb esegue dei cast su alcuni tipi di valori. Non li esegue per tutti i tipi di dato, al fine di evitare conflitti per valori non omogeni fra record diversi dove la medesima key ospita tipi di dato differenti. A seconda delle casistiche può essere opportuno o meno replicare nella query i medesimi cast. Va tenuto presente che in assenza di cast il valore è generalmente ritornato come stringa. In generale il cast va fatto uguale se quel dato è usato per qualche logica e non semplicemente visualizzato nella report. Le modalità di selezione usate da Geoweb sono riepilogate negli esempi sottostanti. E per finire Geoweb costruisce una alias standard secondo la convenzione sopra esposta. Le regole sotto esposte naturalmente valgono anche per qualsiasi report Jasper (destinata a report HTML, di cruscotto, od esportazione custom) che coinvolga informazioni campi JSON.

Esempi:

Se il campo JSON vale 'json_field' e la key vale 'jkey', la formula di SELECT varia in base al database ed al tipo di dato selezionato. Da notare che se 'jkey' contiene un carattere apice singolo (

```
'
```

), esso va sottoposto ad escape con doppio apice singolo (

```
''
```

).

Postgres

FORMULA	ALIAS	TIPI
`(json_field ->> 'jkey') AS "json_field|jkey"`	`json_field|jkey`	`STRING, BOOLEAN`
`(json_field ->> 'jkey')::numeric AS "json_field|jkey"`	`json_field|jkey`	`INTEGER, NUMBER`
`(json_field ->> 'jkey')::timestamp AS "json_field|jkey"`	`json_field|jkey`	`DATE`

Oracle

FORMULA	ALIAS	TIPI
`json_value(json_field, '$.jkey') AS "json_field|jkey"`	`json_field|jkey`	`STRING`
`CAST(json_value(json_field, '$.jkey') AS NUMBER) AS "json_field|jkey"`	`json_field|jkey`	`INTEGER, NUMBER`
`CAST(json_value(json_field, '$.jkey') AS TIMESTAMP) AS "json_field|jkey"`	`json_field|jkey`	`DATE`
`CASE json_value(json_field, '$.jkey') WHEN 'true' THEN 'true' ELSE 'false' END AS "json_field|jkey"`	`json_field|jkey`	`BOOLEAN`

MSSQL Server

FORMULA	ALIAS	TIPI
`json_value(json_field, '$.jkey') AS "json_field|jkey"`	`json_field|jkey`	`STRING`
`CAST(JSON_VALUE(json_field, '$.jkey') AS FLOAT) AS "json_field|jkey"`	`json_field|jkey`	`INTEGER, NUMBER`
`CAST(JSON_VALUE(json_field, '$.jkey') AS DATETIME) AS "json_field|jkey"`	`json_field|jkey`	`DATE`
`CASE WHEN CAST(JSON_VALUE(json_field, '$.jkey') AS BIT) = 1 THEN 'true' ELSE 'false' END AS "json_field|jkey"`	`json_field|jkey`	`BOOLEAN`

Tipologie di Report

Report di Classe

@Deprecated from 4.7.1

Sono le report di lista o di dettaglio che vengono agganciate alla classe, e che seguono eventuali filtri applicati alla classe stessa al momento in cui viene chiamata la report.

Il recordset su cui lavora la report verrà passato a run-time direttamente da Geoweb.

Queste report hanno il vincolo di poter includere come campi solo gli attributi definiti nella classe e presenti almeno in un gruppo attributi.

Tale vincolo può essere aggirato con uno stratagemma, che è quello di definire una report di classe contenente il campo ID, e richiamare una sub report costruita liberamente senza i vincoli del set di attributi usato nella classe, e passandogli l'ID come parametro.

Questo sistema, tuttavia, porta con se problemi prestazionali, e va usato quindi solo se strettamente necessario.

Regole di definizione del Template Jasper

Non ci sono particolari regole da seguire per realizzare una Report di Classe, tranne il tenere presente che:

- in fase di design è bene definire una semplice query sulla tabella/vista collegata alla classe, altrimenti non si riesce neanche ad eseguire una anteprima. Tuttavia occorre ricordare che tale query non verrà utilizzata a run-time, perché come già detto sopra il recordset viene passato direttamente da Geoweb.
- non possono essere inclusi campi (neanche calcolati) che non siano già definiti nella classe come attributi, nonché inseriti in almeno un gruppo attributi (vedi sopra)
- se vengono costruiti dei Raggruppamenti, nelle report di Lista, è fondamentale che i campi per i quali ordinare siano indicati nell'apposito campo in fase di configurazione della classe.

Configurare una Report di Classe

Nell'Admin, aprire il sotto menù della Classe, aprire la voce *Reportistica* e selezionare *Nuova Reportistica*.

Definire i seguenti parametri:

- `Nome Report`: denominazione (rispettare regole di nomenclatura)
- `Descrizione`: Etichetta che comparirà nel menù delle Azioni
- URL: il campo URL individua la posizione fisica (comprensivo di nome del file) del report all'interno dei contenuti statici a partire dalla cartella template, ad esempio l'URL "/Inventario/Edifici/DettaglioEdificio.jasper" individua il file "DettaglioEdificio.jasper" che si trova in "…/WEB/templates/Inventario/Edifici"
- `tipo Report`: formato in output. Può assumere i valori: PDF, RTF, XLS, HTML, Analisi Excel
- `Ordina per`: è necessario specificare i campi per i quali ordinare i record, nel caso di report di Lista, anche se i campi sono già definiti nella query della report. NB: questo è fondamentale se nella Report vengono usati i Raggruppamenti!!!
- `Numero di righe max`: ????
- `Uso`: indica se la report è utilizzabile sulla Lista, sul Dettaglio, o è adatta per entrambe le modalità.
- `Titolo della Report`: nome che sarà assunto dal file di output
- `Nome Attributo`: nel caso di report di Dettaglio, al nome del file (sopra) sarà aggiunto il valore dell'attributo qui indicato. Es. Ordine di Lavoro n. <CODICE_ODL>

Problematiche che possono verificarsi - non risolte

Nel caso che la classe sia molto complessa e abbia parecchi Widget definiti come LinkList, ExternalTable, DBComboBox, ecc, quindi parecchi campi che vengono decodificati internamente da Geoweb tramite join con altre Tabelle/Classi potrebbero verificarsi problemi a recuperare nomi di campi definiti nella classe come attributi, in quanto Geoweb va a rinominare con degli alias i campi se si verificano ambiguità con i nomi.

Nel caso in cui non vengano quindi visualizzati i contenuti di alcuni campi, bisogna andare a consultare i LOG e vedere la query che viene effettuata sulla classe, in maniera da ottenere i corretti riferimenti. ATTENZIONE però, se viene aggiunto un nuovo widget che punta a un altra classe/tabella,

occorre verificare la nuova query.

Report di Cruscotto

@Deprecated from 4.7.1

Le report di Cruscotto sono report costruite liberamente sul dataset inerente alla tematica applicativa gestita.

L'applicazione fornisce un pannello di presentazione ('scheda') di tali report, le quali possono essere costruite in XLS o in PDF, e possono essere eseguite con la richiesta a run-time, all'utente che le esegue, di parametri utente (ad esempio intervalli di date entro cui eseguire l'estrazione dati).

Tali pannelli vengono amministrati direttamente da Client. Questo torna utile nel caso in cui un utente molto esperto, con opportune credenziali, voglia contribuire alla definizione di proprie report o modificare/aggiornare le report esistenti.

Non esistono vincoli nella Definizione dei Template Jasper.

Occorre invece predisporre Geoweb alla realizzazione di tale pannello, e alla funzionalità Client di poter aggiungere, eliminare o modificare le Report già presenti.

Configurare un Cruscotto

Per fare questo occorre configurare un leafItem di tipo *gwReportsSheet*, come mostrato sotto:

```
            <leafItem name="NOME_LEAFITEM" label="LABEL_LEAFITEM"
image="FILE_ICONA.png" type="gwReportsSheet">
                <parameter name="NAME"
value="NOME_CARTELLA_REPORT"></parameter>
                <parameter name="VISUALIZATION_GROUPS"
value="GRUPPO1,GRUPPO2,.."></parameter>
                <parameter name="MANAGER_GROUPS"
value="GRUPPO1,.."></parameter>
            </leafItem>
```

NOME_LEAFITEM –> nome del Leafitem (attenzione il nome del leafItem deve essere sempre univoco a livello di Progetto)

LABEL_LEAFITEM –> Etichetta del leafItem, che sarà visualizzata a Menù

FILE_ICONA –> Nome dell'immagine png che sarà visualizzata a fianco della Etichetta sul menù

NOME_CARTELLA_REPORT –> Denominazione univoca (a livello di istanza) che sarà utilizzata da Geoweb per creare una cartella sotto ../WEB/Templates, la quale ospiterà tutte le report contenute nel cruscotto e che saranno caricate (upload) e configurate da Client. ATTENZIONE a una regola, ovvero **NOME_CARTELLA_REPORT** deve essere UGUALE a **NOME_LEAFITEM**

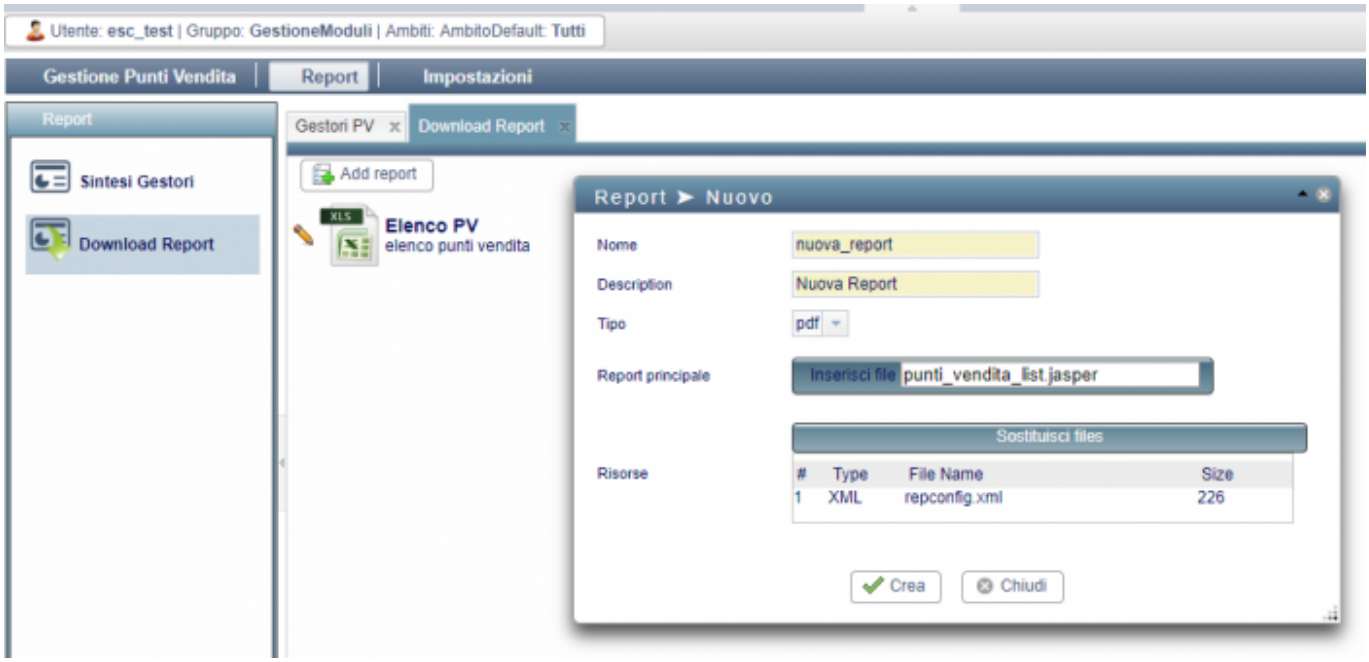
GRUPPO1,GRUPPO2,.. –> nomi dei Gruppi di Utenti (uno o più di uno, separati da virgola) che avranno diritto di Visualizzare i Report (**VISUALIZATION_GROUPS**), o diritto di Aggiungere, Eliminare, Modificare Report (**MANAGER_GROUPS**)

Amministrare le Report di Cruscotto

Una volta definito il leafItem, al salvataggio dell'XML, Geoweb crea una cartella sotto ../WEB/Templates, denominata come definito nel parametro NAME, e destinata ad accogliere i templates delle report.

Un utente che appartiene a uno dei gruppi definiti come 'MANAGER_GROUPS' vedrà comparire, all'accesso da Client, un pulsante con scritto 'Add Report'.

Cliccando il pulsante compare un pannello in pop-up che permette di definire la nuova Report, e fare l'upload del templates realizzato. Sarà Geoweb che provvederà a salvarlo all'interno della cartella di File System sopra citata.



Oltre al templates principale, l'utente da Client avrà la possibilità, tramite tale pannello, di uploadare anche eventuali file di risorsa funzionali alla report principale, quali subreport, immagini, ecc.

Report di Analisi

La report di tipo “Analisi Excel” è una tipologia di Reportistica di Geoweb, definita a livello di Classe nell'Admin, che scrive i dati in un Documento Excel (.xlsx) costruito a mò di template con una serie di regole.

NB: per questo tipo di Report, viene utilizzato direttamente Excel come strumento di Authoring e non occorre fare nulla con iReport.

Il Documento conterrà dei fogli di lavoro di sintesi (Tabelle Pivot, Grafici) basati sul foglio Data, riempito a run-time da Geoweb.

Regole di definizione del Documento Excel

- la posizione del Documento Template deve trovarsi nel path definito nell'Admin e relativo al percorso standard di tutte le altre report (comportamento analogo alle altre report di classe) es. <context_templates_path>/Analysis/template_analysis_aec_room.xlsx
- il nome del foglio dati, ovvero il foglio di lavoro del Documento destinato a ospitare i dati deve chiamarsi *Data*
- il contenuto del foglio *Data*, in cui l'elenco delle colonne debbono essere riferite al nome dei CAMPI degli attributi della classe (se non trova l'attributo nella classe, questo viene segnalato nel log). Le colonne possono essere un sottoinsieme degli attributi della classe.

Configurazione come Report di Classe:

Va configurata nell'Admin, analogamente a quanto visto per le Report di Classe, come report di Lista di tipo “Analisi Excel”.

Dopo aver scelto il tipo “Analisi Excel” viene presentata una funzione per generare un template di Analisi Excel già contenente un foglio Data con i nomi delle colonne riferite al nome degli attributi della classe.

Esecuzione della Report: La report compare nel pulsante *Azioni* incluso nella toolbar della Lista della Classe.

Se viene lanciata dalla lista senza elementi selezionati, riempirà il foglio Data con tutti i record della lista (considerando eventuali ACL e Ambiti).

Se viene lanciata dalla lista con elementi selezionati, riempirà il foglio Data solo con quest'ultimi.

Configurazione come Report di Cruscotto:

La report va aperta da azione javascript, passando dei parametri di Filtro:

`openListReportWithFilter(className,reportName,type,attachment,grid,queryParameter)`

- `className`: nome della classe
- `reportName`: nome della report
- `type`: 4 (corrisponde a Template Analisi Excel)
- `attachment`: se aprire la report come allegato (true) o in pop-up (false)
- `grid` (opzionale): se la funzione javascript è lanciata da una lista il valore è 'grid', altrimenti 'null'
- `queryParameter` (opzionale): filtri

Esecuzione della Report:

Come report lanciata da un cruscotto o da un pannello di lancio in HTML (Report HTML)

Interfacce personalizzate HTML (Report HTML)

Si tratta di pagine disegnate con iReport e renderizzate in formato HTML, che Geoweb è in grado di aprire come 'scheda' e a cui è possibile agganciare come hyperlink delle API Javascript, che permettono di eseguire azioni in Geoweb. In pratica, a tutti gli effetti, queste pagine diventano una interfaccia disegnata ad hoc per l'esigenza contingente e che integra i casi d'uso standard applicativi resi disponibili da Geoweb per tutte le classi definite (apertura record in lista, filtro, clic su lista per apertura del dettaglio, ecc.).



Chiamata di una Interfaccia Report HTML da Menù (leafitem)

La configurazione di un report HTML viene effettuata attraverso gli strumenti di amministrazione, definendo un apposito leafItem di tipo *gwHtmlReport* nell'XML del progetto da cui il report deve essere chiamato.

Il LeafItem è fatto nel modo seguente:

```
<leafItem name="NOME_VOCE_MENU" label="ETICHETTA_VOCE_MENU"
image="FILE_ICONA.png" type="gwHtmlReport">
    <parameter name="reportUrl" value="PERCORSO_NOME_FILE "
hideToClient="false"></parameter>
</leafItem>
```

NOME_VOCE_MENU –> indicare il nome della voce di menù

ETICHETTA_VOCE_MENU –> indicare l'etichetta mostrata all'utente in corrispondenza della voce di

menù che richiama la visualizzazione del report html

FILE_ICONA.png –> nome del file dell'icona visualizzata in corrispondenza della voce di menù (file posizionato all'interno della cartella images dei contenuti statici)

PERCORSO_NOME_FILE –> percorso e nome del file che richiama la visualizzazione del report HTML (a partire dalla cartella templates dei contenuti statici), ad esempio "Inventario/Sintesi/DatiInventatioEdifici.jasper" individua il file "DatiInventatioEdifici.jasper" che si trova in "…/WEB/templates/Inventario/Sintesi".

Dalla versione 4.6.16

A partire dalla Versione 4.6.16 sono disponibili **nuovi parametri** di configurazione:

- premessa: la reportHtml, una volta renderizzata, crea una suddivisione dello spazio in tre colonne. La colonna centrale è quella principale e che contiene i componenti della report, le altre due colonne laterali sono vuote e vengono create in automatico per adattare lo spazio rimanente (in parti uguale a sinistra e a destra).

- **jrPageWidth**: larghezza (in percentuale) della colonna centrale
- **sideColumnWidth**: larghezza (in percentuale) delle colonne laterali (default 50%)

Esempio Configurazione Report HTML 100% in orizzontale:

```
var reportParameters =
{
...
'jrPageWidth':'100%',
'sideColumnWidth':'0%',
...
}
```

Attenzione: Configurare la report html 100% in orizzontale può deformare le immagini o altri componenti della report.

Regole di definizione del Template Jasper

Visto che la pagina sarà aperta a tutta grandezza nell'area 'scheda' dell'interfaccia Client, occorre tenere presenti delle regole di formattazione delle report html che saranno preparate:

Dimensioni minime della pagina

- Larghezza = 700
- Altezza = 100
- Numero minimo di bande previste 3, di seguito le caratteristiche delle bande:
- Titolo – h banda = 50
- Testata di colonna – h banda = 30

- Dettaglio – h banda = 20

Esempio di struttura di report con Titolo – Testata di colonna - Dettaglio



RILIEVO RETE ELETTRICA BT - CONTEGGIO POD PER TIPO FORNITURA E ZONA

Zona	Cabine rilevate	Centralizzata	Singola	Colonna Montante	Totale POD
$F{ZONA}	$F{NUM_TOT_CAB}	$F{POD_TOT_CENTR}	$F{POD_TOT_SINGOLO}	$F{POD_TOT_CM}	**$F{POD_TOT}**

TITOLO La banda dovrà contenere il titolo della report e opzionalmente il logo del cliente / prodotto ed un eventuale grafico. Il titolo della report dovrà essere inserito utilizzando l'oggetto 'testo statico' di iReport e dovrà avere le seguenti caratteristiche:

- Font = Arial
- Dimensione = 14
- Grassetto = Si
- Colore in primo piano = 2,72,133 (scala rgb)
- Colore di sfondo = Nessuno
- Allineamento orizzontale = Centro
- Allineamento verticale = Medio
- Larghezza dell'elemento = Pari alla larghezza della banda

TESTATA DI COLONNA

Contiene le etichette dei campi che vengono richiamati nella report. Le etichette dovranno essere inserite utilizzando l'oggetto 'testo statico' di iReport e dovranno avere le seguenti caratteristiche:

- Font = SansSerif (ininfluente nel caso in cui l'output del font rimanga quello di default)
- Dimensione = 10
- Grassetto = Si
- Colore in primo piano = 255,255,255 (scala rgb)
- Colore di sfondo = 2,72,133 (scala rgb)
- Opaco = Si
- Allineamento orizzontale = Centro
- Allineamento verticale = Medio
- Altezza elemento = Pari all'altezza della banda

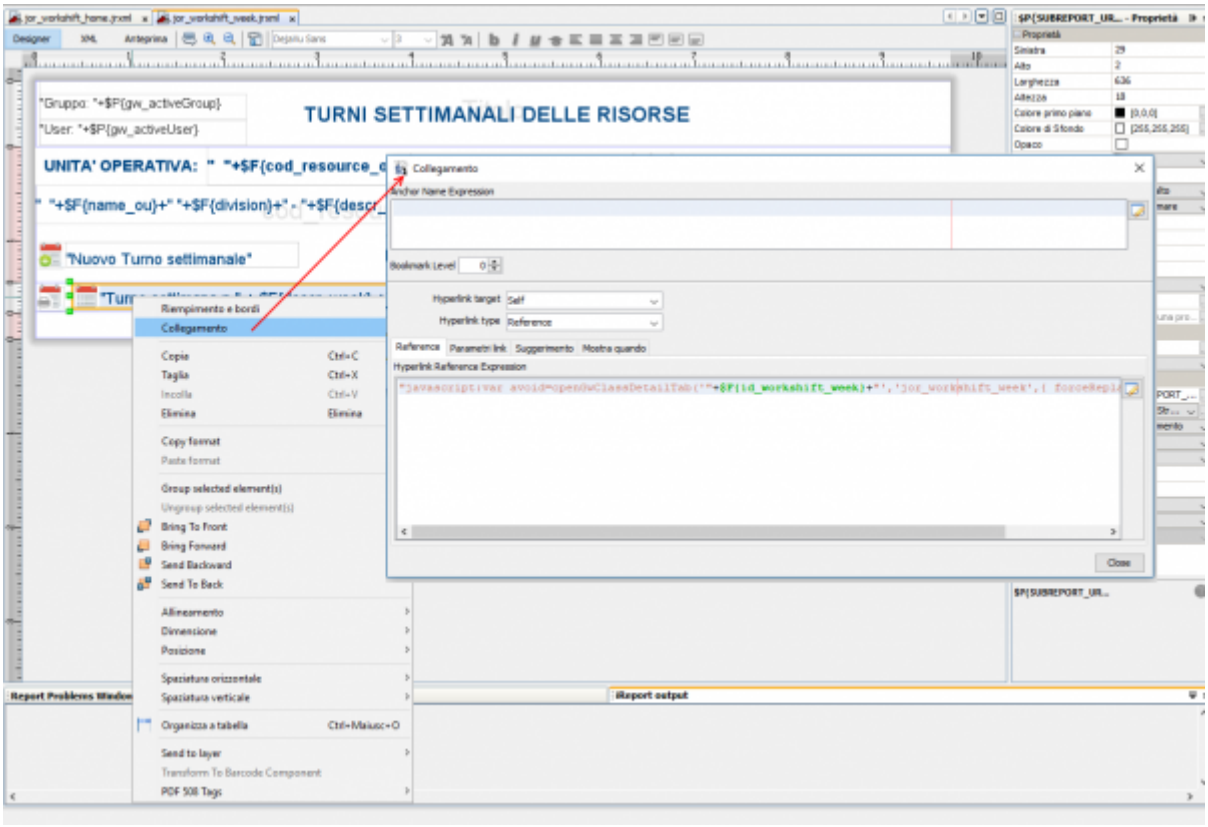
Definizione degli HyperLink per chiamare le API Javascript di Geoweb

Creare un oggetto con hyperlink

Inserire l'oggetto destinato ad accogliere il collegamento. Questo può essere un oggetto di tipo 'Testo Statico', oppure un 'Campo di Testo', oppure una Immagine. Dal menù contestuale (tasto dx), selezionare Collegamento o Hyperlink.

1. Scegliere il tipo di Hyperlink Target = Self
2. Scegliere il tipo di Hyperlink Type = Reference
3. Inserire nel box Hyperlink Reference Expression l'istruzione Javascript con tutti gli opportuni

parametri, facendo attenzione a scriverla su un'unica riga, altrimenti iReport non la riconosce correttamente.



Utilizzo dei fogli di stile predefiniti

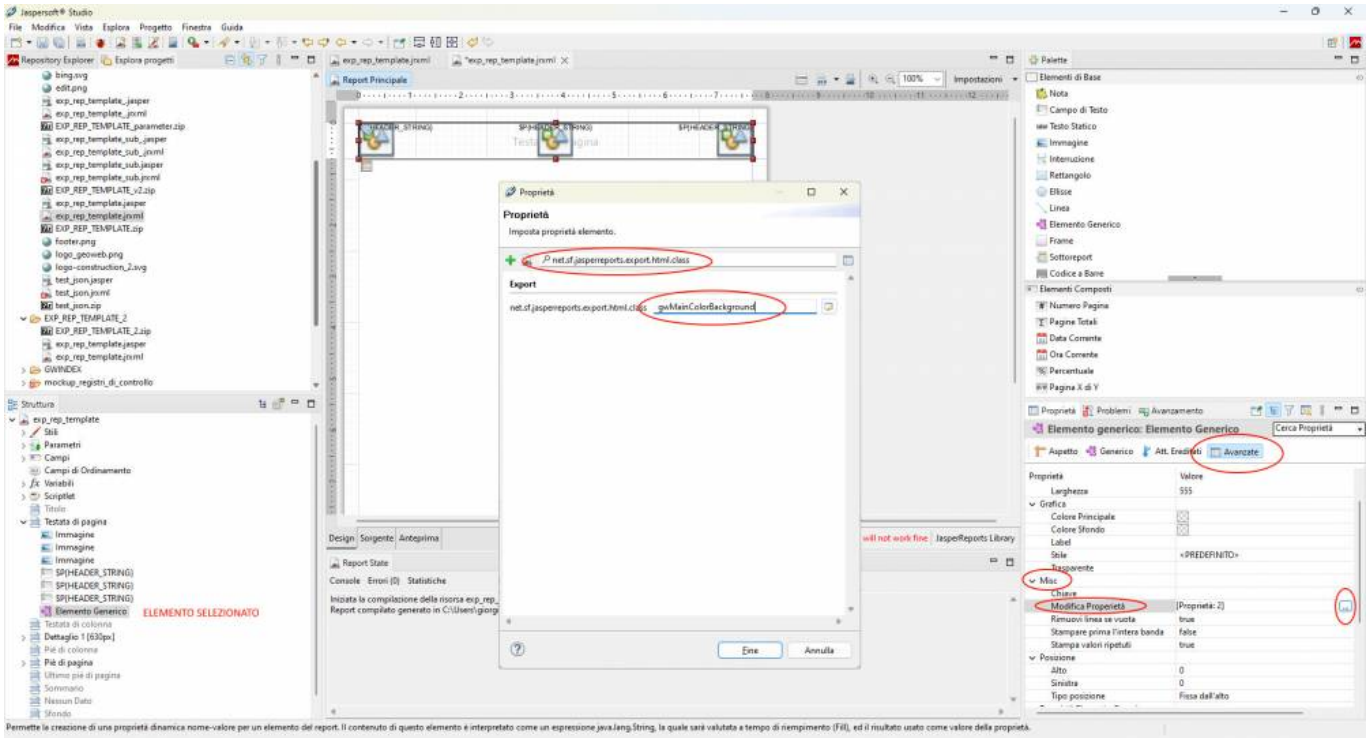
Geoweb rende disponibile alla webapp un foglio di stile predefinito nel quale sono definite svariate regole css, con selettore basato sulle classi CSS.

In generale, per applicare lo stile derivante dalla regola css dentro la report è necessario:

1. identificare nell'xml della report il nodo di interesse (oppure da UI nelle ultime versioni di JasperStudio)
2. aggiungere manualmente la property **net.sf.jasperreports.export.html.class** dentro il tag **reportElement** dentro il tag dell'elemento target
3. definire il valore della property: **[classe/i css]** (per esempio *gwReportButton*, vedi sotto per tutte le opzioni)

La property in Jaspersoft Studio 6 si imposta seguendo questi passaggi:

1. selezionare l'elemento
2. tab **Avanzate**
3. sezione **Misc**
4. voce **Modifica Proprietà** ⇒ click su '...'
5. filtrare per **net.sf.jasperreports.export.html.class**
6. impostare il valore (per esempio *gwMainColorBackground*)



La property può essere impostata anche a mano editando l'xml sorgente della report (è necessario in IReport 5.6, che potrebbe non avere una UI dedicata)

```
<genericElement>
    <reportElement stretchType="ContainerHeight" x="0" y="0" width="555"
height="50" isRemoveLineWhenBlank="true" uuid="0cd7b66a-
ae3b-4cb4-9fd9-14a1f363d518">
        <property name="net.sf.jasperreports.export.html.class"
value="gwMainColorBackground"/>
        ...
    </reportElement>
    ...
</genericElement>
```

Usare gli stili predefiniti offre i seguenti vantaggi:

- uniformità dello stile fra le varie report html
- garantire l'accessibilità
- in generale facilitazioni nel modificare in maniera centralizzata tutti gli stili, compresi quelli delle report, senza dover rimettere mano alle report html stesse

Generalmente, per rendere l'effetto di una pagina web dentro la report html, viene utilizzato questo stratagemma (esempio del classico link/button):

- viene posizionata una *Immagine* rappresentativa dell'azione da compiere con a fianco un **Testo Statico** che illustra in maniera esplicita l'azione (l'immagine dovrebbe essere .svg, con un colore standard o Fontawesome)
- **sopra ad entrambi gli oggetti** viene posta una **immagine trasparente**, che ricopre perfettamente i due elementi sopra descritti, e alla quale viene effettivamente applicato il selettore css voluto.
- il collegamento, o hyperlink, viene effettivamente agganciato all'immagine trasparente, che deve trovarsi posizionata 'on top', ovvero sopra agli altri elementi, in maniera che sia la prima

che viene 'trovata' dal mouse.



Selettori CSS predefiniti

Elenco dei selettori css predefiniti:

- **reportlink** (**@Deprecated per gwReportButton**, vedi sotto) sull'hover del mouse viene applicato uno stile scuro, ma semitrasparente (che permette comunque la lettura del testo sottostante) (reportlink:hover{ rgba(51, 74, 102, 0.15); }) (ne viene scoraggiato l'uso per le nuove report)

Dalla versione 4.6.11.30 (issue #1251)

Dato che in 4.6 lo stile hover dei button è differente da quello delle *row* delle *grid*, sono state aggiunti contestualmente i seguenti selettori css (integrabili nelle report):

- **gwReportButton**: da usare per i button. Sull'hover del mouse viene applicato uno stile scuro, ma semitrasparente (che permette comunque la lettura del testo sottostante)
- **gwReportRow**: da usare per le row degli elenchi. Sull'hover del mouse viene applicato uno stile simile (al momento) allo stile di gwReportButton

---

Dalla versione 4.6.12.4 (issue #1276)

Introdotti nuovi selettori css per stilizzare le report HTML:

- **gwReportRowOddStyled**: usato per la banda della riga della report, da usare per stilizzare le righe pari
- **gwReportRowEvenStyled**: usato per la banda della riga della report, da usare per stilizzare le righe dispari
- **gwReportRowGrouper**: usato per le intestazioni (o anche raggruppamenti) delle riga della report (grigio chiaro #CCCCCC)
- **gwReportRowGrouperGrouper**: usato come ulteriore livello di gearachizzazione, superiore a gwReportRowGrouper (grigio #EAEAEA)

Dalla versione 4.7.3.4 (issue #1347)

Introdotti nuovi selettori css per stilizzare le report HTML:

- **gwReportToolbar**: usato per le toolbar delle report (stesso colore di gwReportRowGrouperGrouper) (grigio #EAEAEA)

Elenco dei selettori css applicabili per modificare lo sfondo (css background-color). Sono usabili nelle report html nella maniera descritta sopra.

Nota: solo per pura conoscenza le proprietà con *var(*)* sono modificabili dinamicamente dal framework tramite api javascript.

```
/* G W   B A C K G R O U N D */
/*---------------------------------------*/
.gwMainColorBackground { background-color: var(--gwUIColor) !important; }
/*issue #1347*/
.whiteBackground, /*@Deprecated*/
.gwWhiteBackground { background-color: var(--gwBGColor2) !important;}
.gwWhiteHTMLBackground { background-color: #FFFFFF !important;}
.gwBlackBackground { background-color: var(--gwFontColor1) !important; }
/*issue #1347*/
.gwBlackHTMLBackground { background-color: #000000 !important; }
.gwLightGrayBackground { background-color: var(--gwBGColor3) !important; }
/*issue #1347*/
.gwLightGrayHTMLBackground { background-color: #D3D3D3 !important; }
.gwGrayBackground { background-color: #A9A9A9 !important; }
.gwGrayHTMLBackground { background-color: #A9A9A9 !important; }
.gwDarkGrayBackground { background-color: #696969 !important; }
.gwDarkGrayHTMLBackground { background-color: #808080 !important; }
.gwPageHeaderInfoSectionBackground { background-color: var(--gwFontColor3)
!important; } /*issue #1347*/
.gwMacroFunctionGrouperAreaBackground {background-color: #EAEAEA
!important;}
/*---------------------------------------*/
```

Elenco dei selettori css utilizzati nel framework per la tematizzazione del **colore dei testi** (css color).

```
/* G W   C O L O R */
/*---------------------------------------*/
/*@@@_ICON_COLOR_MARKER_@@@*/
/*when modifying a color, diffrente from HTML variant,
search in code the marker for check if @Deprecated icon[X]Color
in code need to be updated*/
.gwGreenColor { color: var(--gwUICommandGreenColor) !important; /*issue
#1347*/ }
.gwGreenMapSelectionColor { color: #B0C782 !important; }
```

```
.gwRedColor { color: var(--gwUICommandRedColor) !important; /*issue #1347*/
}
.gwPinkColor { color: #ffc0cb !important; }
.gwYellowColor { color: var(--gwUICommandYellowColor) !important; /*issue
#1347*/ }
.gwYellowMapMeasureColor { color: #FFAA32 !important; } /*#FFAA32 FF9900
DD8800*/
.gwBlueColor { color: #6699CC !important; }
.gwFolderColor { color: #FFD800 !important; }
.gwGreenExcelColor { color: #008000 !important; }
.gwGreenCSVColor { color: #008000 !important; }
.gwRedPDFColor { color: #CB0606 !important; }
.gwBlueRTFColor { color: #295293 !important; }
.gwBlueDOCXColor { color: #295293 !important; } /*issue #1171*/
.gwMainColor { color: var(--gwUIColor) !important; } /*issue #1347*/
.gwLightGrayColor { color: #F6F6F6 !important; }
.gwGrayColor { color: var(--gwFontColor4) !important; } /*issue #1347*/
.gwDarkGrayColor { color: #696969 !important; }
.gwUIBorderColor { color: #EFEFEF !important; }
.gwUICommandColor { color: var(--gwUICommandColor) !important; } /*base UI
command, like refresh*/ /*var(--gwUICommandColor2) var(--gwUICommandColor)
AAAAAA*/ /*issue #1347*/
.gwUISelectedColor { color: var(--gwColorSelected) !important; } /*selected
leafItem, treeNode*/ /*issue #1347*/
.gwUINotSelectedColor { color: var(--gwBGColorNotSelected) !important; }
/*not selected dijitAccordionContainer title*/ /*issue #1347*/
.gwWhiteColor { color: var(--gwFontColor2) !important; /*issue #1347*/ }
.gwBlackColor { color: var(--gwFontColor1) !important; } /*issue #1347*/
.gwWhiteHTMLColor { color: #FFFFFF !important; }
.gwBlackHTMLColor { color: #000000 !important; }
.gwLightGrayHTMLColor { color: #D3D3D3 !important; }
.gwGrayHTMLColor { color: #A9A9A9 !important; }
.gwDarkGrayHTMLColor { color: #808080 !important; }
.gwPageHeaderInfoSectionColor { color: var(--gwFontColor3) !important; }
/*issue #1347*/
.gwContextFunctionIconColor { color: #9D9D9D !important; }
.gwMacroFunctionGrouperLabelColor { color: #696969 !important; }
.gwMacroFunctionLabelColor { color: #696969 !important; }
```

Elenco dei selettori css utilizzati nelle report per la tematizzazione del **colore dei bordi** (css border-color).

```
/* G W   B O R D E R   C O L O R */
/*--------------------------------------*/
.gwMainColorBorder { border-color: var(--gwUIColor) !important; } /*issue
#1347*/
.gwMainColorBorderTop { border-top-color: var(--gwUIColor) !important; }
/*issue #1347*/
.gwMainColorBorderBottom { border-bottom-color: var(--gwUIColor) !important;
} /*issue #1347*/
```

```
.gwMainColorBorderLeft { border-left-color: var(--gwUIColor) !important; }
/*issue #1347*/
.gwMainColorBorderRight { border-right-color: var(--gwUIColor) !important; }
/*issue #1347*/
.gwWhiteBorder { border-color: var(--gwBGColor2) !important;} /*issue
#1347*/
.gwWhiteHTMLBorder { border-color: #FFFFFF !important;} /*issue #1347*/
.gwBlackBorder { border-color: var(--gwFontColor1) !important; } /*issue
#1347*/
.gwBlackBorderTop { border-top-color: var(--gwFontColor1) !important; }
/*issue #1347*/
.gwBlackBorderBottom { border-bottom-color: var(--gwFontColor1) !important;
} /*issue #1347*/
.gwBlackBorderLeft { border-left-color: var(--gwFontColor1) !important; }
/*issue #1347*/
.gwBlackBorderRight { border-right--color: var(--gwFontColor1) !important; }
/*issue #1347*/
.gwBlackHTMLBorder { border-color: #000000 !important; } /*issue #1347*/
.gwLightGrayBorder { border-color: var(--gwBGColor3) !important; } /*issue
#1347*/
.gwLightGrayHTMLBorder { border-color: #D3D3D3 !important; } /*issue #1347*/
.gwGrayBorder { border-color: #A9A9A9 !important; } /*issue #1347*/
.gwGrayHTMLBorder { border-color: #A9A9A9 !important; } /*issue #1347*/
.gwDarkGrayBorder { border-color: #696969 !important; } /*issue #1347*/
.gwDarkGrayHTMLBorder { border-color: #808080 !important; } /*issue #1347*/
```

Linee Guida stilizzazione report HTML

In generale sulle report si impostano i colori (a fuoco, in formato esadecimale, come #007AC2, od altri) prendendoli fra quelli standard.

Questi colori, oltre che essere scelti fra quelli disponibili, vanno combinati in maniera da non creare situazioni a basso contrasto fra colore del testo e colore di sfondo.

Queste settings, oltre che per la preview dentro JasperSoft Studio, sono determinanti ed esaustive per l'esportazione in pdf ed in xlsx.

Per le report HTML si devono invece prendere ulteriori accorgimenti.

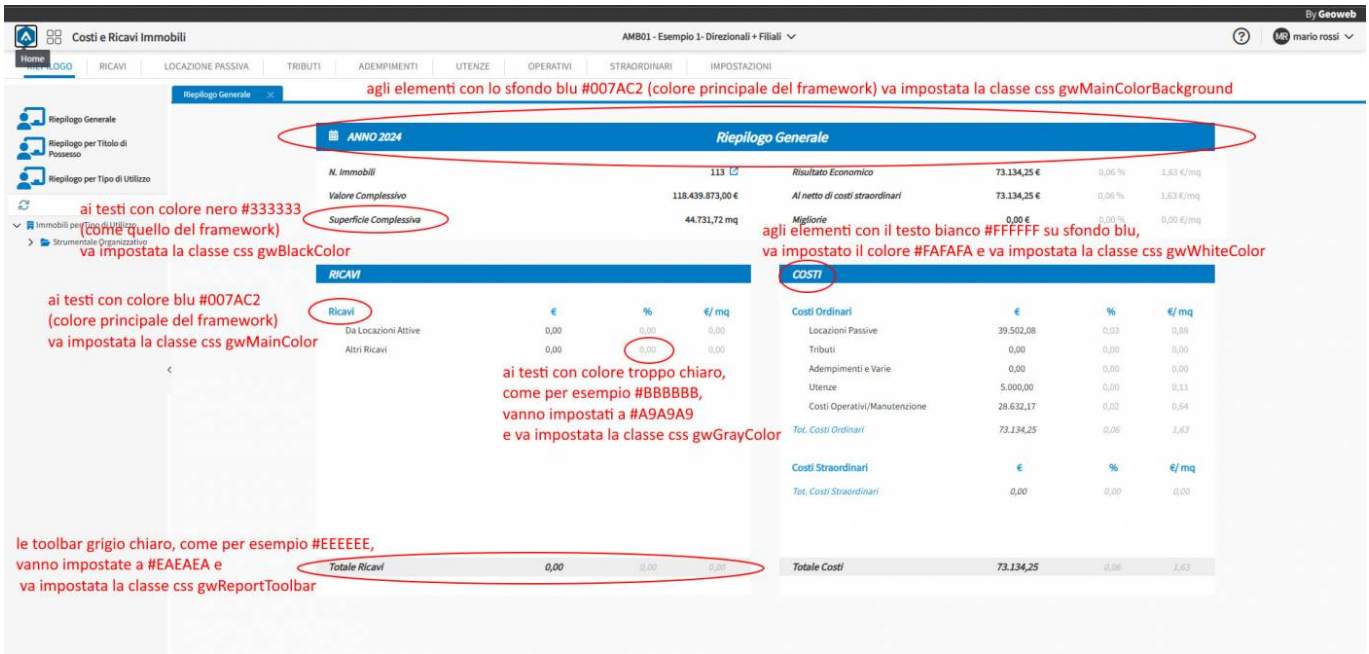
Tutti gli elementi che hanno colore su testo, sfondo o bordo devono applicare una specifica classe css, in maniera tale che vengano applicati gli stili predefiniti del framework. Questo ha risvolti sia nell'ambito dell'accessibilità (per la modalità a contrasto elevato), sia in caso di futuri cambiamenti di stile.

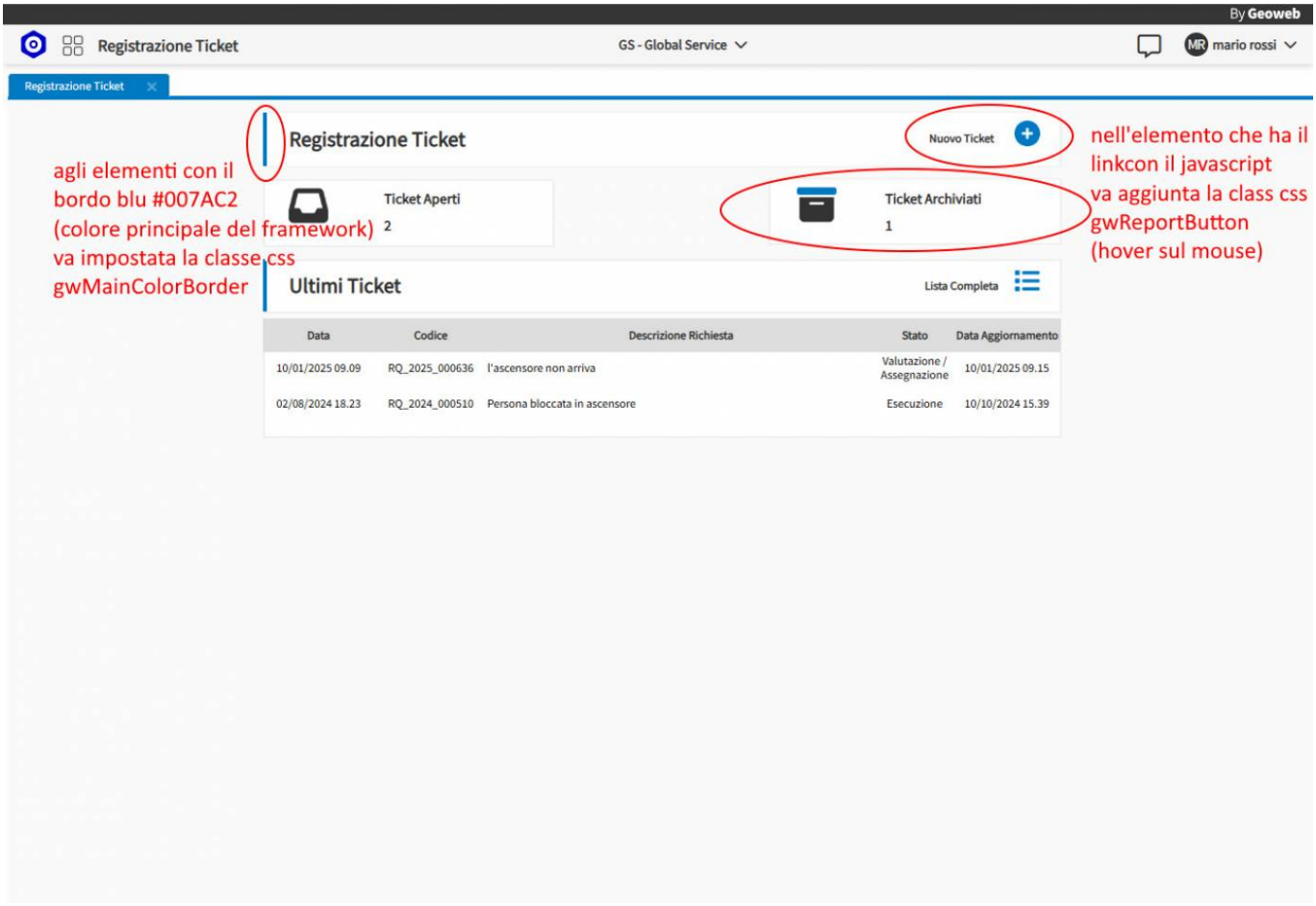
Alla luce di quanto visto sopra, si possono delineare le seguenti linee guida per le report HTML:

- applicare dove necessario le classi css **gwReportButton** (per l'hover del mouse) sugli elemtni dove si applicano i link
- applicare dove necessario le classi css **gwReportRowGrouper** e **gwReportRowGrouperGrouper**
- agli elementi con lo sfondo blu #007AC2 (colore principale del framework) va impostata la

classe css **gwMainColorBackground**

- agli elementi con il testo bianco #FFFFFF su sfondo blu, va impostato il colore #FAFAFA e va impostata la classe css **gwWhiteColor**
- ai testi con colore blu #007AC2 (colore principale del framework) va impostata la classe css **gwMainColor**
- ai testi con colore nero #333333 (come quello del framework) va impostata la classe css **gwBlackColor**
- ai testi con colore troppo chiaro, come per esempio #BBBBBB, vanno impostati a #A9A9A9 e va impostata la classe css **gwGrayColor**
- le toolbar che raggruppano record di lista, vanno impostate a #EAEAEA e con la classe css **gwReportRowGrouper**
- le toolbar che raggruppano, come ulteriore livello, i gruppi di record di lista, vanno impostate a #CCCCCC e va impostata la classe css **gwReportRowGrouperGrouper**
- le toolbar grigio chiaro, come per esempio #EEEEEE, vanno impostate a #EAEAEA e va impostata la classe css **gwReportToolbar**
- agli elementi con il bordo blu #007AC2 (colore principale del framework) va impostata la classe css scegliendo la più corretta fra **gwMainColorBorder**, **gwMainColorBorderTop**, **gwMainColorBorderBottom**, **gwMainColorBorderLeft**, **gwMainColorBorderRight**





Accessibilità Navigazione da tastiera

Il framework in fase di apertura della report html, provvede in automatico ad eseguire tutta una serie di operazioni volte ad ottimizzare l'esperienza utente, a garantire l'uniformità degli stili e soddisfare i requisiti sull'accessibilità. Operazioni svolte:

- a tutti i link (tag <a>) aggiunge un attributo **tabindex="0"** per far prendere il focus navigando da tastiera tramite tab
- per tutti i link (tag <a>) fa si che, quando il link ha il focus, al click sul tasto ENTER venga eseguita l'azione del link.
- per tutti i parentNode dei nodi link (tag <a>), applica in automatico la class **gwReportButton**
- vengono uniformati i colori impostati a mano nella report (senza specificare la class css). le varianti di colore note vengono ricondotte a quelle della palette condivisa. Ogni difformità viene sanata e vengono aggiunte in automatico le class css corrette. **Rimangono esclusi da questa automazione i colori totalmente fuori standard**.

Codice dell'azione che esegue il replace al volo dello stile, per valutare le property effettivamente gestite:

```
//issue #1347
function handleGwHtmlReportContent(parentDomNode){
    require(['dijit/registry', 'dojo/query', 'dojo/dom', 'dojo/dom-class',
'dojo/topic'], function(registry, query, dom, domClass, topic){
        //text color main blue
        const color007AC2 = 'rgb(0, 122, 194)'; //#007AC2
        const color0033CC = 'rgb(0, 51, 204)'; //#0033CC
        const color0E96D6 = 'rgb(14, 150, 214)'; //#0E96D6
```

```
        const colorFFFFFF = 'rgb(255, 255, 255)'; //FFFFFF
        const color000000 = 'rgb(0, 0, 0)'; //#000000
        const colorFAFAFA = 'rgb(250, 250, 250)'; //#FAFAFA
        const color333333 = 'rgb(51, 51, 51)'; //#333333

        //background color gray
        const colorEEEEEE = 'rgb(238, 238, 238)'; //#EEEEEE
        const colorEAEAEA = 'rgb(234, 234, 234)'; //#EAEAEA
        const colorEDF4FB = 'rgb(237, 244, 251)'; //#EDF4FB

        //text color gray
        const colorBBBBBB = 'rgb(187, 187, 187)'; //#BBBBBB
        const colorA9A9A9 = 'rgb(169, 169, 169)'; //#A9A9A9
        const color666666 = 'rgb(102, 102, 102)'; //#666666
        const color696969 = 'rgb(105, 105, 105)'; //#696969
        const color838383 = 'rgb(131, 131, 131)'; //#838383

        const allElements = parentDomNode.querySelectorAll('*');
        allElements.forEach(function(element){
            const style = window.getComputedStyle(element);

            //BACKGROUND COLOR
            if(style.backgroundColor === color007AC2){
                element.classList.add('gwMainColorBackground');
            } else if(style.backgroundColor === colorEEEEEE ||
style.backgroundColor === colorEAEAEA || style.backgroundColor ===
colorEDF4FB){
                element.style.backgroundColor = colorEAEAEA;
                element.classList.add('gwReportToolbar');
            } else if(style.backgroundColor === colorFAFAFA){
                element.classList.add('gwWhiteBackground');
            }

            //COLOR
            if(style.color === color007AC2 || style.color === color0033CC ||
style.color === color0E96D6){
                element.style.color = color007AC2;
                element.classList.add('gwMainColor');
            } else if(style.color === color000000 || style.color ===
color333333){
                element.style.color = color333333;
                element.classList.add('gwFontColor');
            } else if(style.color === colorFFFFFF || style.color ===
colorFAFAFA){
                element.style.color = colorFAFAFA;
                element.classList.add('gwWhiteColor');
            } else if(style.color === colorBBBBBB || style.color ===
colorA9A9A9){
                element.style.color = colorA9A9A9;
```

```
                element.classList.add('gwGrayColor');
            } else if(style.color === color666666 || style.color ===
color696969 || style.color === color838383){
                element.style.color = color696969;
                element.classList.add('gwDarkGrayColor');
            }

            //BORDER COLOR
            if(style.borderColor === color007AC2)
element.classList.add('gwMainColorBorder');
            else if(style.borderTopColor === color007AC2)
element.classList.add('gwMainColorBorderTop');
            else if(style.borderBottomColor === color007AC2)
element.classList.add('gwMainColorBorderBottom');
            else if(style.borderLeftColor === color007AC2)
element.classList.add('gwMainColorBorderLeft');
            else if(style.borderRightColor === color007AC2)
element.classList.add('gwMainColorBorderRight');

            else if(style.borderColor === color000000 || style.borderColor
=== color333333) { element.style.borderColor = 'rgb(51, 51, 51)';
element.classList.add('gwBlackBorder'); }
            else if(style.borderTopColor === color000000 ||
style.borderTopColor === color333333) { element.style.borderTopColor =
'rgb(51, 51, 51)'; element.classList.add('gwBlackBorderTop'); }
            else if(style.borderBottomColor === color000000 ||
style.borderBottomColor === color333333) { element.style.borderBottomColor =
'rgb(51, 51, 51)'; element.classList.add('gwBlackBorderBottom'); }
            else if(style.borderLeftColor === color000000 ||
style.borderLeftColor === color333333) { element.style.borderLeftColor =
'rgb(51, 51, 51)'; element.classList.add('gwBlackBorderLeft'); }
            else if(style.borderRightColor === color000000 ||
style.borderRightColor === color333333) { element.style.borderRightColor =
'rgb(51, 51, 51)'; element.classList.add('gwBlackBorderRight'); }
        });

        query('a', parentDomNode).forEach(function(domNode){
            domNode.setAttribute('tabindex', '0'); //make it focussable
            domNode.parentNode.classList.add('gwReportButton'); //style it
when hovered by mouse

            //enable keyboard usability
            domNode.addEventListener('keydown', (event) => {
                if (event.key === 'Enter') {
                    const jsCode = event.target.getAttribute('href');
                    if (jsCode && jsCode.startsWith('javascript:')) {
                        eval(jsCode.replace('javascript:', '').trim());
                    }
                }
            });
        });
```

```
    });
}
```

Esempi di API richiamabili da HyperLink situati in Report HTML

Tutte le API Javascript esposte da Geoweb sono richiamabili attraverso la definizione corretta di un hyperlink nelle Report HTML.

Di seguito sono riportati alcuni esempi di chiamate semplici. Le espressioni vanno inserite nell' Hyperlink Reference Expression e definite come una stringa unica, disposta su una unica riga.

1. Inserimento di un nuovo record su classe definita
2. Apertura scheda di dettaglio(pop-up)
3. Apertura scheda di dettaglio(tab)
4. Apertura lista con filtro su classe definita
5. Apertura nuova Report HTML
6. Apertura di una Report qualsiasi, stampabile o excel
7. Apertura di mappa
8. Apertura di planimetria

Inserimento di un nuovo record su classe definita

torna su

```
“javascript:var avoidReturn = openGwClassDetailFloatingPane('newItem',
'<className>');”
```

'**newItem**' = parola chiave che indica al metodo che si è in fase di inserimento di un Nuovo Record

className = nome della classe a cui si fa riferimento per apertura della scheda di dettaglio in pop up

```
Esempio:
"javascript:var avoidReturn = openGwClassDetailFloatingPane('newItem',
'ril_cabina');"
```

Apertura scheda di dettaglio(pop-up)

torna su

```
“javascript:var avoidReturn =
```

openGwClassDetailFloatingPane(‘<itemId>’,’<className>’);”

itemID = Inserire la chiave primaria della classe tra apici singoli (anche se il valore è numerico).

className = nome della classe a cui si fa riferimento per apertura della scheda di dettaglio in pop up

```
Esempio:
Nell'esempio l'itemID viene recuperato dal valore del campo corrente
"javascript:var avoidReturn = openGwClassDetailFloatingPane('" + $F{OGC_FID}
+"', 'ril_cabina');"
```

Apertura scheda di dettaglio (tab corrente)

torna su

“javascript:var avoidReturn = openGwClassDetailTab(‘<itemId>’,’<className>’,{ forceReplaceTab: true });”

itemID = Inserire la chiave primaria della classe tra apici singoli (anche se il valore è numerico).

className = nome della classe a cui si fa riferimento per apertura della scheda di dettaglio in pop up

{ forceReplaceTab: true } = parametro che forza la scheda corrente ad essere rimpiazzata con il nuovo contenuto

```
Esempio:
Nell'esempio l'itemID viene recuperato dal valore del campo corrente
"javascript:var avoidReturn = openGwClassDetailTab('" + $F{OGC_FID} +"',
'ril_cabina',{ forceReplaceTab: true });"
```

Apertura lista con filtro su classe definita

torna su

“javascript:var tabWidgetType = 'gwClassList'; var className = '<className>'; var tabName = className; var tabWidgetId =tabWidgetType+'_'+tabName+'_tab'; var tabWidgetTitle = '<TitoloLista>'; var parametersMap = {className: className,staticFilters: { condition: '<condition>', columnName: '<columnName>', operator: '<operatorType>', filterType: 'STRING', value: ['<valueToFilter>'] } };var avoidReturn = openTab(tabWidgetId, tabWidgetType, tabWidgetTitle, parametersMap);”

className: nome della classe di riferimento sulla quale si vuole aprire la lista filtrata

tabWidgetTitle: Titolo della Lista, ovvero dell'elenco filtrato che verrà aperto su un nuovo Tab

parameterMap: in questa variabile si deve definire la condizione del filtro che si vuole applicare e quindi:

- **condition**: operatore condizionale, AND, OR, NOT; la prima condizione ha sempre impostato questo valore a 'AND'
- **columnName**: nome della colonna sulla quale si effettua il filtro;
- **operatorType**: tipo operatore che verrà usato per la query SQL; valori possibili: '=','<>','>','<','like'
- **filterType**: stringa o numero, dipende dal tipo di campo che si filtra: valori possibili: STRING, NUMBER
- **valueToFilter**: valore per il quale filtrare

E' possibile aggiungere ulteriori condizioni di filtraggio, aggiungendo una ulteriore lista racchiusa tra apici. Per utilizzzare la condizione IS NULL oppure NOT IS NULL utilizzare la seguente sintassi ..... (to do)

```
Esempio:
"javascript:var tabWidgetType = 'gwClassList'; var className = 'Ril_Pod';
var tabName = className; var tabWidgetId =tabWidgetType+'_'+tabName+'_tab';
var tabWidgetTitle = 'Elenco POD fornitura Centralizzata'; var parametersMap
= {className: className,staticFilters: { condition: 'AND', columnName:
'tipo_fornitura', operator: '=', filterType: 'STRING', value:
['Centralizzata'] } };var avoidReturn = openTab(tabWidgetId, tabWidgetType,
tabWidgetTitle, parametersMap);"
```

Apertura nuova Report HTML

torna su

Apertura di una nuova interfaccia personalizzata realizzata come Report HTML

```
“javascript:var tabWidgetType = 'gwHtmlReport'; var reportUrl='<reportUrl>';
var tabName = className; var tabWidgetId = tabWidgetType+'_'+tabName+'_tab';
var tabWidgetTitle = '<tabTitle>'; var parametersMap =
{reportUrl:reportUrl};var avoidReturn = openTab(tabWidgetId, tabWidgetType,
tabWidgetTitle, parametersMap);”
```

Definire la seguente variabile:

reportUrl: inserire il percorso dove è salvato il template di report da aprire. Il percorso è da intendersi sempre relativo alla posizione di ..\WEB\template

tabTitle: Titolo della scheda che ospiterà la nuova Report HTML

```
Esempio:
"javascript:var tabWidgetType = 'gwHtmlReport'; var
reportUrl='Cruscotto/Conteggi_POD_Cabine.jasper'; var tabName = className;
var tabWidgetId = tabWidgetType+'_'+tabName+'_tab'; var tabWidgetTitle =
'titolo scheda'; var parametersMap = {reportUrl:reportUrl};var avoidReturn =
openTab(tabWidgetId, tabWidgetType, tabWidgetTitle, parametersMap);"
```

Apertura di una Report qualsiasi, stampabile o excel

torna su

Questa API apre una report qualsiasi, eseguendo la query che impostata nel template. Questo significa che posso eseguire anche una Report di Classe, fermo restando che NON sarà Geoweb a passare il recordset ma questo dovrà essere calcolabile a run-time direttamente dalla report. Geoweb accetta nella chiamata i parametri **className reportParameters**, in maniera opzionale. Questi vengono utilizzati per:

- per recuperare dalla sessione eventuali parametri di ambito
- per essere utilizzati come parametri dalla Report che viene eseguita

```
“javascript:var reportName='<outputReportName>'; var itemId =” <itemId> “;
var className = '<className>'; var reportUrl='<reportURL>'; var attachment =
<attachment>; var reportParameters =
{className:className,attachment:attachment}; var avoidReturn =
openGwReport(reportUrl,reportName,reportParameters,className,attachment);”
```

className = nome della classe in cui è configurata la Report

reportUrl = inserire il percorso dove è salvato il template di report da aprire. Il percorso è da intendersi sempre relativo alla posizione di ..\WEB\template

attachment = definisce la presenza o meno di allegati alla report. Valori possibili: true/false

tabTitle = Titolo della scheda che ospiterà la nuova Report HTML

```
Esempio:
'"javascript:var reportName='TurniSettimanaliRisorse.pdf';var className =
'jor_workshift_week';var
reportUrl='JOR_Cruscotto_WSP/TurniSettimanaliRisorse.jasper'; var attachment
= true; var reportParameters =
{className:className,id_workshift_week:"+$F{id_workshift_week}+"}; var
avoid=openGwReport(reportUrl,reportName,reportParameters,className,attachmen
t);"'
```

Apertura di mappa

torna su

“javascript:var gwMapName = '<gwMapName >';var mapId = gwMapName; var mapUri = <mapUri>;var mapLabel = '<mapLabel>';var avoidReturn = openMapTab(mapId, mapUri, mapLabel);”

I parametri che devono essere impostati:

gwMapName = nome della mappa che si intende aprire, la mappa deve essere stata definita negli strumenti di amministrazione di Geoweb.

mapUri = riferimento alla mappa originale, definita come concatenazione dei parametri che seguono, concatenati come path:

- project_gwid (nome del progetto) +
- '/mapLayout/' +
- gwMapName +
- '.html'
    - esempio: project_gwid+'*/mapLayout/*'+gwMapName+'*.html*'

mapLabel = etichetta che deve essere visualizzata in apertura del tab

```
Esempio:
"javascript:var gwMapName = 'nome della mappa da aprire';var mapId =
gwMapName;var mapUri = project_gwid+'/mapLayout/'+gwMapName+'.html';var
mapLabel = 'Mappa cruscotto';var avoidReturn = openMapTab(mapId, mapUri,
mapLabel);"
```

Apertura di planimetria da report HTML

torna su

“javascript:var gwMapName = '<gwMapName >'; var planId = <planId>; var mapId = gwMapName+'_'+planId; var mapUri = <mapUri>; ;var mapLabel = <mapLabel>;var avoidReturn = openMapTab(mapId, mapUri, mapLabel);”

gwMapName = nome della mappa che contiene la planimetria che si intende aprire, la mappa deve essere stata definita negli strumenti di amministrazione di Geoweb.

planId = chiave primaria della planimetria che si intende aprire

mapUri = riferimento alla mappa originale, definita come concatenazione dei parametri che seguono, concatenati come path:

- project_gwid (nome del progetto) +
- '/mapLayout/' +

- gwMapName + '/' +
- planId +
- '.html'
    - esempio: project_gwid+'/mapLayout/'+gwMapName+'/'+planId+'.html'

mapLabel = etichetta che deve essere visualizzata in apertura del tab

```
Esempio:
"javascript:var gwMapName = 'nome_mappa_in_geoweb';var planId = 1234;var
mapId = gwMapName+'_'+planId ;var mapUri =
project_gwid+'/mapLayout/'+gwMapName+'/'+planId+'.html'; var mapLabel =
'Titolo scheda planimetria'; var avoidReturn = openMapTab(mapId, mapUri,
mapLabel);"
```

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

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

Last update: **2025/07/23 15:46**