Report Esportabili - Messa in esercizio

Installare gli script

Questo passaggio è necessario solo per eventuali porting alla 4.7.1 da una versione precedente:

La presenza della tabella gwa_func è un prerequisito.

postgres.sql

```
-- issue #1171

-------------------------------------------------------------------------
-----------
--- Script GW 4.6

--- REPLACE xxx_metadata with the metadata schema name
--- REPLACE xxx_data with the data schema name
--- GSC, 2023.12
-------------------------------------------------------------------------
-----------



-------------------------------------------------------------------------
-----------
-- I M P O R T A N T
-- gwa_function relation should already exist
-------------------------------------------------------------------------
-----------



-- Table: xxx_data.gwr_report

-- DROP TABLE xxx_data.gwr_report;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_report
(
    id_report INTEGER NOT NULL,
    cod_report CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT
NULL,
    name_report CHARACTER VARYING(200) COLLATE pg_catalog."default" NOT
NULL,
    descr_report CHARACTER VARYING(500) COLLATE pg_catalog."default",
    author_report CHARACTER VARYING(200) COLLATE pg_catalog."default"
NOT NULL,
    date_report TIMESTAMP WITHOUT TIME zone NOT NULL,
    pdf_output INTEGER,
    xlsx_output INTEGER,
```

```
    docx_output INTEGER,
    is_active INTEGER,
    type_generation CHARACTER VARYING(50) COLLATE pg_catalog."default",
    main_report_name CHARACTER VARYING(200) COLLATE
pg_catalog."default" NOT NULL,
    report_resources bytea,
    is_custom INTEGER,
    cod_report_category CHARACTER VARYING(50) COLLATE
pg_catalog."default",
    report_resources_name CHARACTER VARYING(100) COLLATE
pg_catalog."default",
    report_resources_ctype CHARACTER VARYING(100) COLLATE
pg_catalog."default",
    has_parameters_to_prompt INTEGER DEFAULT 0,
    is_filterable INTEGER DEFAULT 0,
    filterable_gw_class_names CHARACTER VARYING COLLATE
pg_catalog."default",
    is_filterable_from_detail INTEGER DEFAULT 0,
    is_in_toolbar INTEGER,
    is_in_toolbar_detail INTEGER,
    CONSTRAINT pk_gwr_report PRIMARY KEY (id_report),
    CONSTRAINT cod_report_unique UNIQUE (cod_report)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_report
    OWNER TO test_gw46data;

COMMENT ON COLUMN xxx_data.gwr_report.filterable_gw_class_names
    IS 'List of comma separated gwClassName
Ex:
gwclassname1,gwclassname2

As result the report would to be executable from the gwClassList with
the current applied filter';

-----------------------------------------------------------------
-- Table: xxx_data.gwr_report_category

-- DROP TABLE xxx_data.gwr_report_category;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_report_category
(
    id_report_category INTEGER NOT NULL,
    cod_report_category CHARACTER VARYING(50) COLLATE
pg_catalog."default",
    name_report_category CHARACTER VARYING(200) COLLATE
```

```
pg_catalog."default",
    descr_report_category CHARACTER VARYING(500) COLLATE
pg_catalog."default",
    ord_report_category INTEGER,
    CONSTRAINT pk_gwr_report_category PRIMARY KEY (id_report_category),
    CONSTRAINT cod_report_category_unique UNIQUE (cod_report_category)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_report_category
    OWNER TO test_gw46data;

-------------------------------------------------------------
-- Table: xxx_data.gwr_model

-- DROP TABLE xxx_data.gwr_model;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_model
(
    id_model INTEGER NOT NULL,
    cod_model CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT
NULL,
    name_model CHARACTER VARYING(200) COLLATE pg_catalog."default" NOT
NULL,
    descr_model CHARACTER VARYING(500) COLLATE pg_catalog."default",
    author_model CHARACTER VARYING(200) COLLATE pg_catalog."default"
NOT NULL,
    date_model TIMESTAMP WITHOUT TIME zone NOT NULL,
    type_header_model CHARACTER VARYING(100) COLLATE
pg_catalog."default" NOT NULL,
    header_model_string CHARACTER VARYING COLLATE pg_catalog."default",
    header_model_image bytea,
    type_footer_model CHARACTER VARYING(100) COLLATE
pg_catalog."default" NOT NULL,
    footer_model_string CHARACTER VARYING COLLATE pg_catalog."default",
    footer_model_image bytea,
    is_custom INTEGER,
    header_model_image_name CHARACTER VARYING COLLATE
pg_catalog."default",
    header_model_image_ctype CHARACTER VARYING COLLATE
pg_catalog."default",
    footer_model_image_name CHARACTER VARYING COLLATE
pg_catalog."default",
    footer_model_image_ctype CHARACTER VARYING COLLATE
pg_catalog."default",
    header_h_alignment CHARACTER VARYING COLLATE pg_catalog."default",
    footer_h_alignment CHARACTER VARYING COLLATE pg_catalog."default",
    header_model_html CHARACTER VARYING COLLATE pg_catalog."default",
```

```
    footer_model_html CHARACTER VARYING COLLATE pg_catalog."default",
    CONSTRAINT pk_gwr_model PRIMARY KEY (id_model),
    CONSTRAINT cod_model_unique UNIQUE (cod_model)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_model
    OWNER TO test_gw46data;

COMMENT ON COLUMN xxx_data.gwr_model.type_header_model
    IS 'allowed values: [''image'', ''string'', ''html'']';

COMMENT ON COLUMN xxx_data.gwr_model.type_footer_model
    IS 'allowed values: [''image'', ''string'', ''html'']';

COMMENT ON COLUMN xxx_data.gwr_model.header_h_alignment
    IS 'horizontal alignment. possible values ["left", "center",
"right"], default "left"';

COMMENT ON COLUMN xxx_data.gwr_model.footer_h_alignment
    IS 'horizontal alignment. possible values ["left", "center",
"right"], default "left"';

COMMENT ON COLUMN xxx_data.gwr_model.header_model_html
    IS 'it contains complex html, including svg elements';

COMMENT ON COLUMN xxx_data.gwr_model.footer_model_html
    IS 'it contains complex html, including svg elements';

-----------------------------------------------------------------
-- Table: xxx_data.gwr_param

-- DROP TABLE xxx_data.gwr_param;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_param
(
    id_report_param INTEGER NOT NULL,
    cod_report CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT
NULL,
    cod_report_param CHARACTER VARYING(50) COLLATE pg_catalog."default"
NOT NULL,
    name_report_param CHARACTER VARYING(200) COLLATE
pg_catalog."default" NOT NULL,
    type_report_param CHARACTER VARYING(100) COLLATE
pg_catalog."default" NOT NULL,
    string_value CHARACTER VARYING(4000) COLLATE pg_catalog."default",
    image_value bytea,
```

```
    image_value_name CHARACTER VARYING COLLATE pg_catalog."default",
    image_value_ctype CHARACTER VARYING COLLATE pg_catalog."default",
    CONSTRAINT pk_gwr_param PRIMARY KEY (id_report_param),
    CONSTRAINT fk_gwr_report FOREIGN KEY (cod_report)
        REFERENCES xxx_data.gwr_report (cod_report) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_param
    OWNER TO test_gw46data;

-------------------------------------------------------------
-- Table: xxx_data.gwr_r_report_model

-- DROP TABLE xxx_data.gwr_r_report_model;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_r_report_model
(
    id_r_report_model INTEGER NOT NULL,
    cod_report CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    cod_model CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT
NULL,
    is_default INTEGER NOT NULL,
    CONSTRAINT id_r_report_model_pk PRIMARY KEY (id_r_report_model),
    CONSTRAINT fk_gwr_model FOREIGN KEY (cod_model)
        REFERENCES xxx_data.gwr_model (cod_model) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT fk_gwr_report FOREIGN KEY (cod_report)
        REFERENCES xxx_data.gwr_report (cod_report) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_r_report_model
    OWNER TO test_gw46data;

-------------------------------------------------------------
-- Table: xxx_data.gwr_r_report_function
```

```
-- DROP TABLE xxx_data.gwr_r_report_function;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_r_report_function
(
    id_r_report_function INTEGER NOT NULL,
    cod_report CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT
NULL,
    cod_function CHARACTER VARYING(50) COLLATE pg_catalog."default" NOT
NULL,
    CONSTRAINT pk_gwr_report_r_gwa_func PRIMARY KEY
(id_r_report_function),
    CONSTRAINT fk_gwr_function FOREIGN KEY (cod_function)
        REFERENCES xxx_data.gwa_func (cod_func) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT fk_gwr_report FOREIGN KEY (cod_report)
        REFERENCES xxx_data.gwr_report (cod_report) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_r_report_function
    OWNER TO test_gw46data;

-------------------------------------------------------------------
-- Table: xxx_data.gwr_r_rep_cat_func

-- DROP TABLE xxx_data.gwr_r_rep_cat_func;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_r_rep_cat_func
(
    id_r_rep_cat_func INTEGER NOT NULL,
    cod_report CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    cod_report_category CHARACTER VARYING COLLATE pg_catalog."default"
NOT NULL,
    ord_report INTEGER,
    cod_func CHARACTER VARYING COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT gwr_r_report_rep_cat_pkey PRIMARY KEY
(id_r_rep_cat_func),
    CONSTRAINT "unique" UNIQUE (cod_report, cod_report_category,
cod_func),
    CONSTRAINT fk_gwa_func FOREIGN KEY (cod_func)
        REFERENCES xxx_data.gwa_func (cod_func) MATCH SIMPLE
```

```
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT fk_gwr_report FOREIGN KEY (cod_report)
        REFERENCES xxx_data.gwr_report (cod_report) MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT fk_gwr_report_category FOREIGN KEY (cod_report_category)
        REFERENCES xxx_data.gwr_report_category (cod_report_category)
MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_r_rep_cat_func
    OWNER TO test_gw46data;

-----------------------------------------------------------------
-- Table: xxx_data.gwr_filterable_classes

-- DROP TABLE xxx_data.gwr_filterable_classes;

CREATE TABLE IF NOT EXISTS xxx_data.gwr_filterable_classes
(
    id_gwr_filterable_classes INTEGER NOT NULL,
    gw_class_name CHARACTER VARYING COLLATE pg_catalog."default" NOT
NULL,
    gw_class_label CHARACTER VARYING COLLATE pg_catalog."default",
    CONSTRAINT gwr_filterable_classes_pkey PRIMARY KEY
(id_gwr_filterable_classes)
)
WITH (
    OIDS = FALSE
)
TABLESPACE pg_default;

ALTER TABLE xxx_data.gwr_filterable_classes
    OWNER TO test_gw46data;

COMMENT ON TABLE xxx_data.gwr_filterable_classes
    IS 'tabella usata per esplicitare su quali classi le report
esportabili possono essere collegate (e quindi comparire nei rispettivi
menu), al fine di usare la gwClassList od il gwClassDetail per lanciare
le report esportabili con specifici filtri';
```

```
-------------------------------------------------------------------
```

Importare metadati geoweb

E' messa a disposizione la configurazione dei metadati del progetto di test dedicato alle Report Esportabili.

La presenza della classe gwa_func è un prerequisito.

L'utente configuratore deve inizialmente importare questo progetto nell'ambiente di sviluppo del prodotto.

I permessi di azioni e progetti sono impostati al gruppo 'SOLUTION_MANAGER'. Fare gli eventuali aggiustamenti del caso.

Raffinare la configurazione metadati

Successivamente dovrà prendere i vari blocchi dell'xml di progetto ed applicarli puntualmente agli xml delle Funzioni già in essere. Quindi tipicamente:

- le Funzioni appropriate dovranno essere integrate, nel loro menu di navigazione, dalla nuova scheda gwExportableReport. [primo menubarItem dei metadati di progetto]
- all'utente Solution Manager, dovrà essere abilitata una nuova funzione dì gestione delle reportistiche [secondo menubarItem dei metadati di progetto]
  - di questo vanno integrate il leafItem di apertura gwExportableReportSM, e le anagrafiche di gestione:
    - Report
    - Modelli Report (opzionalmente, in quanto gestibili anche da dettaglio Report)
    - Parametri Report (opzionalmente, in quanto gestibili anche da dettaglio Report)
    - Categorie Report (opzionalmente, in quanto gestibili solo dalla scheda gwExportableReportSM)
  - gli accordion delle 'Anagrafiche di Supporto' e delle 'Funzioni utente' possono essere ignorati
- accordati tutti i permessi alle varie classi

Tool Necessari

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).

Creazione Report di prodotto

Le report attualmente presenti nei prodotti: di classe, statiche, HTML (improprie) vanno deprecate e riscritte ex novo.

A tale scopo si rimanda alle convenzioni sugli stili ed alla guida di configurazione delle Report precedente in quanto continuano a valere tutti i meccanismi di integrazione con Geoweb.

L'unica cosa rilevante che viene aggiunta nel nuovo meccanismo di reportistica è un nuovo parametro che viene dinamicamente passato alla report **GW_BUILT_CLAUSE**. Questo parametro, **va utilizzato nella clausola WHERE della query della report (o della sottoreport)**, cosi:

```
SELECT *
FROM gwr_report
WHERE $P!{GW_BUILT_CLAUSE}
ORDER BY gwr_report.cod_report ASC
```

Normalmente GW_BUILT_CLAUSE vale **1=1** (quindi una condizione neutra).

Invece nei casi dove la Report è configurata per essere esportabile anche da Lista/Dettaglio di talune anagrafiche, questo parametro viene passato alla Report come una composizione della **PRIMARY_KEY** della classe con il filtro esatto cosi come composto dalla lista(o dettaglio) di Geoweb. Esempio GW_BUILT_CLAUSE lista sulla classe gwr_report stessa:

```
SELECT *
FROM gwr_report
WHERE (id_report IN (SELECT a0.id_report
FROM gwr_report a0
WHERE ((((a0.id_report IN (15465637,15465638,15465639,15465643,15465645))))
AND ((('SOLUTION_MANAGER' = 'SOLUTION_MANAGER'))))))
ORDER BY gwr_report.cod_report ASC
```

Creazione Report Custom

La creazione di Report Custom è in capo al Solution Manager, che sebbene teoricamente possa agire in autonomia, più realisticamente, si avvarrà della collaborazione di personale con competenze tecniche specifiche negli ambiti:

- SQL
- Jasper Report
- Geoweb

Questo ruolo potrebbe essere svolto proprio da Geoweb Italia su richiesta del cliente