PRINT-MAP-LEGEND

Original file

Allegati

PRINT MAP LEGEND

Introduzione:

Print Map Legend, è una funzionalità del framework che consente di stampare in una report pdf una serie di Layout (insieme di disegni) tutti con una specifica tematizzazione e legenda. La report può mostrare i layout per intero o suddivisi in aree di stampa più piccole.

Nella tabella gwd_print_area sono definite le aree di stampa legate al layout attraverso il campo drawing.

La funzione che si preoccupa di creare il byte array contente la report è printLayout così definita:

byte[] printLayout(Integer projectGwid,Integer dpi,String IdDrawingSet,**int** jasperImageWidth,**int** jasperImageHeight,String format,String orientation,Boolean adapt,Double scale,String nameLegendLayer,String nameJasper,HashMap<String, Boolean> layersOnOff,String mgMapName,String title)

Parametri in input:

- projectGwId: Id del progetto aperto
- dpi: punti per pollici, intero per la definizione dell'immagine ottenuta
- idDrawingSet Stringa contenente una lista ordinata di id_drawing_set separati da #
- jasperImageWidth: Larghezza dello spazio dedicato all'immagine della mappa nel file jasper
- jasperImageHieght Altezza dello spazio dedicato all'immagine della mappa nel file jasper
- Format(A4,A3,A2,A1)
- orientation (vertical-horizonatal)
- Adapt(boolean):se true prova ad allargare la width e la height affinché contenga tutta l'area da stampare
- scale: Double,valore della scala fisso, viene preso il centro dell'ingombro e impostata la scala richiesta; se -1 l 'immagine viene adattata
- nameLegendLayer: Stringa contente il nome del layer in legenda (la legenda puo' contenere un layer unico)
- nameJasper: nome della report nei contenuti statici con estensione .jasper
- layersOnOff: HashMap contente tutti i nomi dei layers di cui si vuole controllare la visibilità rispetto al default impostato nella mappa. Se 1 lo vogliamo acceso se 0 spento. Se il layer non compare in questo hashmap sarà visibile o meno come definito nel default
- mgMapName (opzionale): Il nome della mappa da stampare; se non presente viene presa quella associata al drawing set
- title: Titolo della report

La report avrà un titolo principale e coinciderà con quello passato del parametro title.

Ogni pagina della report visualizzerà una mappa con un titolo ed una legenda

Il titolo della mappa, coinciderà con il nome del layout o con il nome dell'area di stampa. L'imgombro visualizzato sarà l'ingombro di tutti i disegni del layout o l'ingombro dell'area di stampa, indicato nel campo geometry della tabella gwd_print_area.

E' possibile utilizzare questa funzionalità attraverso una action agganciata ad una classe di Geoweb che richiama un groovy specifico.

Per la configurazione è necessario creare:

- AZIONE SULLA CLASSE
- GROOVY
- JAPSER REPORT

Configurazione:

ACTION:

Per quanto riguarda la action, essa deve essere definita in una classe che abbia una relazione con la classe drawingset. Puo' essere una action di lista che di dettaglio. Essa deve richiamare il groovy, passandogli il progetto corrente e la chiave o le chiavi del record(dettaglio) o dei records(lista). Nel caso di lista, nell'esempio, ho passato genericamente il queryParameter contenete la clausola where per recuperare successivamente le chiavi degli oggetti selezionati. Al ritorno del groovy, la action, deve preoccuparsi di mostrare il pdf generato e rimandato dal groovy come byte array. Per la stampa del pdf è predisposta già la funzione che stampa il pdf a partire dal bytearray tornato dal groovy.

```
debugger;
    function saveAs(blob, fileName) {
                                var url = window.URL.createObjectURL(blob);

                                var anchorElem = document.createElement("a");
                                anchorElem.style = "display: none";
                                anchorElem.href = url;
                                anchorElem.download = fileName;

                                document.body.appendChild(anchorElem);
                                anchorElem.click();

                                document.body.removeChild(anchorElem);

                                // On Edge, revokeObjectURL should be called only after
                                // a.click() has completed, atleast on EdgeHTML 15.15048
                                setTimeout(function() {
                                    window.URL.revokeObjectURL(url);
                                }, 1000);
                            }

    function openAs(blob) {
                                var url = window.URL.createObjectURL(blob);

                               window.open(url);
                            }
```

A seguire un esempio di implementazione completo della action:

```
var callback2 = function(
                    responseHashMap //Object
                    ){
                       var result = responseHashMap.result;
                        if(result.success==true){
                            debugger;
                            var pdfAsB64String=result.data;
                            var byteCharacters =atob(pdfAsB64String);
                            var byteNumbers = new Array(byteCharacters.length);
                            for (var i = 0; i < byteCharacters.length; i++) {
                                byteNumbers[i] = byteCharacters.charCodeAt(i);
                            }
                            var byteArray = new Uint8Array(byteNumbers);
                            var blob1 = new Blob([byteArray], {type: "application/pdf"});

                            var fileName1 = "cool.pdf";
                            openAs(blob1);
                        }else{
                            var title = 'Print Map legend'; //optional
                            var message = result.message; //optional
                            var yesCallback = function(){ //optional
                            };
                            var params2 = { title: title, message: message, yesCallback: yesCallback};
                            showOkDialog(params2);
                        }
                    };

    var parameterMapGroovy = {
    queryParameter: queryParameter,
    project_gwid:project_gwid
};

groovyActionGeneric('act_gwd_storey_printmaplegend2', parameterMapGroovy, callback2);
```

GROOVY:

Per quanto riguarda il groovy, esso deve recuperare il drawingset o la lista dei drawingset se non gli vengono passati in chiaro dall'azione. A seguire un esempio, in cui vengono recuperati la lista dei drawing set a partire da una lista di storey. Devono essere preparate tutte le variabili da passare alla funzione printLayout; Le variabili guideranno la configurazione della report di output come la si vuole ottenere; Fare attenzione ai valori delle variabili format,orientation,width,height che dovranno essere coerenti con il file jasper creato(formato e quindi dimensione del foglio, dimensione dell'area dedicata all' immagine e orientamento). La funzione è disponibile nel service 'mapGuidePrintService' visibile in tutti i groovy. Il groovy deve tornare un hashmap contenente il byte array il nome della file se gli si vuole dare un nome, l'esito dell'operazione ed un eventuale messaggio.

```
log.debug("INIZIO PRINT MAP LEGEND------------------------------");
//PREDISPONGO L HASH MAP DI RITORNO
def map = [:];
map.success = false;
map.message="";
// RECUPERO I PARAMETRI IN INPUT IL PROJECT ID E IL FILTRO ESEGUITO NELLA LISTA PER GLI OGGETTI SELEZIONATI
def projectGwid=parameterMap.project_gwid;
def filters =(java.util.List) parameterMap.queryParameter.filters; //array
//RECUPERO GLI OGGETTI SELEZIONATI NEL MIO CASO SONO I PIANI
def records=classService.selectClassRecords('gwd_storey_read', filters);

def idDrawingSet=null;
//FISSO LA MAPPA CHE VOGLIO STAMPARE

def mgMapName = "FloorPlanSpaceInv";

try {
    //CICLO I MIEI PIANI E RECUPERO IL RELATIVO LAYOUT
    for(i=0;i<records.size();i++){
        def fkLayout=records[i].fk_layout;
        //RECUPERO TUTTI I DRAWING SET DEL LAYOUT
        ArrayList<GwdDrawingSet> list=layout2DService.getDrawingSetsByLayout(fkLayout);
        //CICLO I DRAWING SET E PRENDO LE CHIAVI DI TUTTI QUELLI CHE HANNO LA MAPPA CHE HO SCELTO E LI MEMORIZZO IN UNA STRINGA SEPARATI DA CANCELLETTO
        for(y=0;y<list.size();y++){
            def gwdDrawingSet=list.get(y);
            def nomeMapDrawingSet=gwdDrawingSet.getMap();
            def idDrawingSetCurr=gwdDrawingSet.getPkDrawingSet();
                    if(nomeMapDrawingSet!=null && mgMapName.equals(nomeMapDrawingSet)){
                if(idDrawingSet!=null){
                    idDrawingSet+="#"+idDrawingSetCurr;
                }else{
                    idDrawingSet=""+idDrawingSetCurr;
                }
        }
        }
    }


    //DEFINISCO LA VISIBILITA DEI LAYERS
    HashMap<String, Boolean> layersOnOffHashMap=new HashMap<String, Boolean>();
    layersOnOffHashMap.put("SPI_ROOM_URB_DEST_USE",true);
    def layersOnOff=layersOnOffHashMap;

     //DEFINISCO I PARAMETRI COERENTI CON LA REPORT
     def format="A4";
     def orientation="horizontal";
     def dpi=150;
     def printScale =true;
     def printDate = true;
     def adapt=true;
     def scale=-1;
     def displayWidth=656;//DIMENSIONI IN LARGHEZZA DEL RETTANGOLO IN CUI SARA VISIBILE L IMMAGINE
     def displayHeight=480; //DIMENSIONI IN ALTEZZA DEL RETTANGOLO IN CUI SARA VISIBILE L IMMAGINE
     def title = "TEST";



        //DEFINISCO IL NOME DEL LAYER IN LEGENDA
        def nameLegendLayer="SPI_ROOM_URB_DEST_USE";

        //DEFINISCO IL NOME DEL JASPER
        def nameJasper="reportTestPrintLegend.jasper";
        //DEFINISCO IL NOME DEL FILE CHE ANDRO AD APRIRE
        def reportName="testsilvia.pdf";

        //RICHIAMO LA FUNZIONE CON I PARAMETRI
        byte[] byteArray =mapGuidePrintService.printLayout(projectGwid,dpi,idDrawingSet,displayWidth,displayHeight,format,orientation,adapt,scale,nameLegendLayer,nameJasper,layersOnOff,mgMapName,title);

        //PREPARO IL RITORNO IN CASO DI SUCCESSO
        map.message= 'OK!';
        map.success = true;
        map.data=byteArray;
        map.reportName=reportName;
    }catch(Exception e) {
    //PREPARO IL RITORNO IN CASO DI ECCEZIONE
        map.success = false;
        map.message =e.message;
    }

return map;
```

REPORT:

La report potrà essere personalizzata a proprio piacimento, l'unica cosa fissa saranno i parametri e i dati che riceverà in input.

Parametri:

1. TITLE:String
2. DATE:String
3. SUBREPORT_URL String

Per quanto riguarda I parametri, abbiamo semplicemente la data attuale, il titolo della report e la stringa contenente il path ai contenuti statici da utilizzare per richiamare eventuali sottoreport.

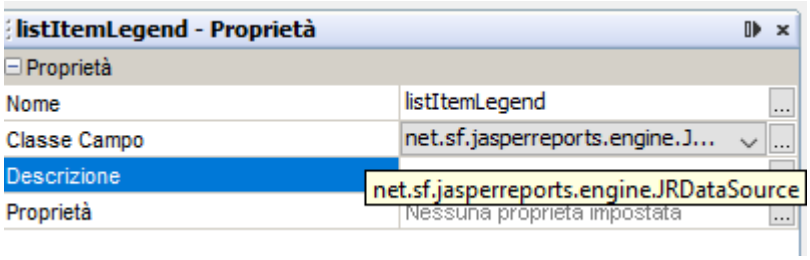
Campi:

1. idLayout:String
2. nameLayout:String
3. namePrintArea:String

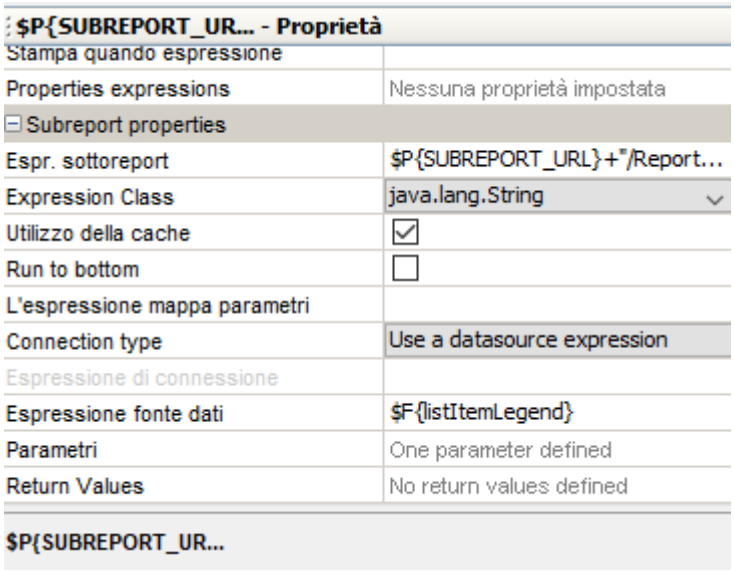
4. imageByteArray:InputStream
5. listItemLegend: net.sf.jasperreports.engine.JRDataSource

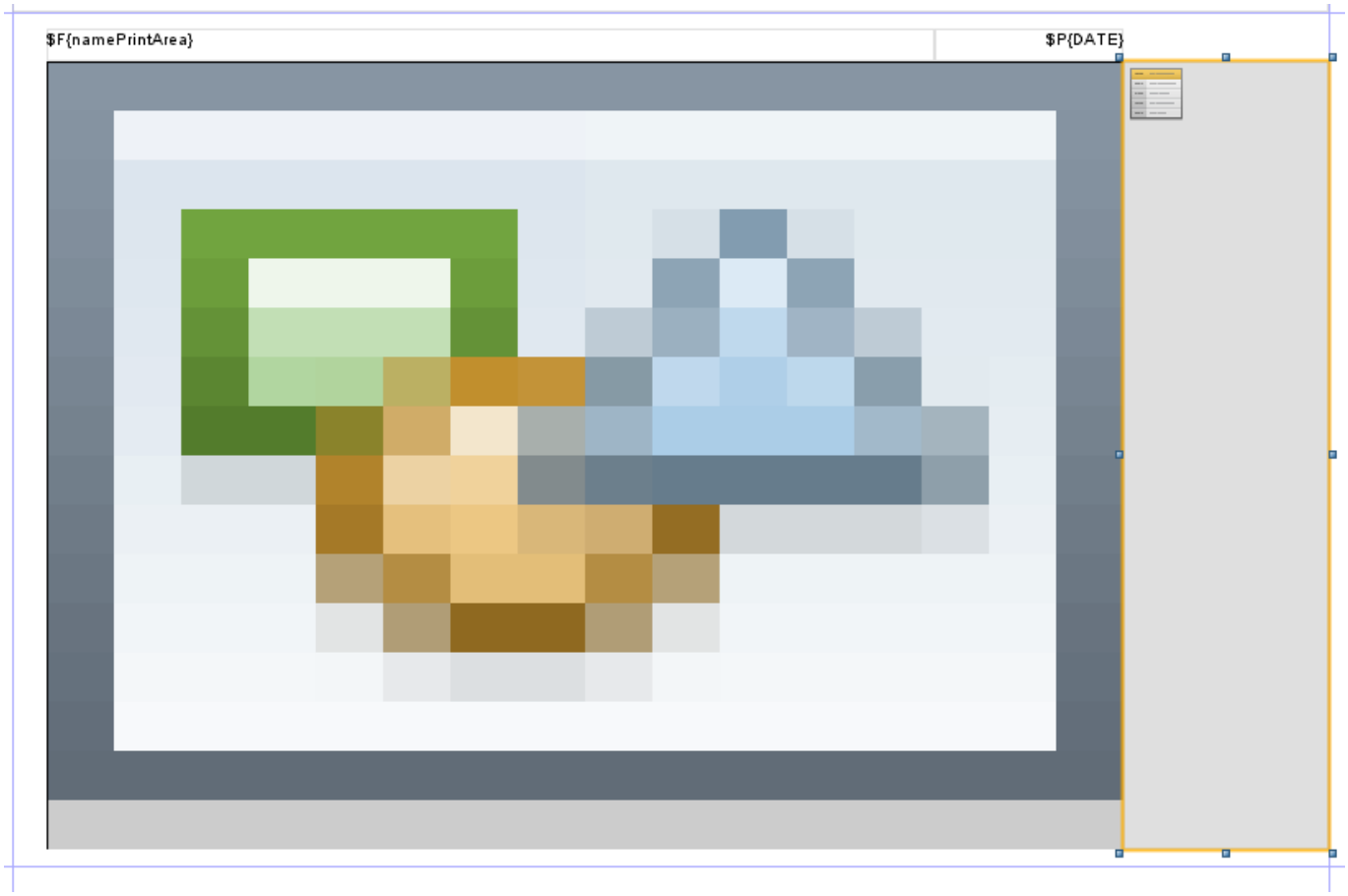
Quest'ultimo, conterrà la lista delle voci del menu legenda(testo,icona), è un oggetto di tipo fonte dati(una lista di record) da utilizzare come input in un eventuale sottoreport;

L'oggetto net.sf.jasperreports.engine.JRDataSource deve essere così dichiarato nel jasper:

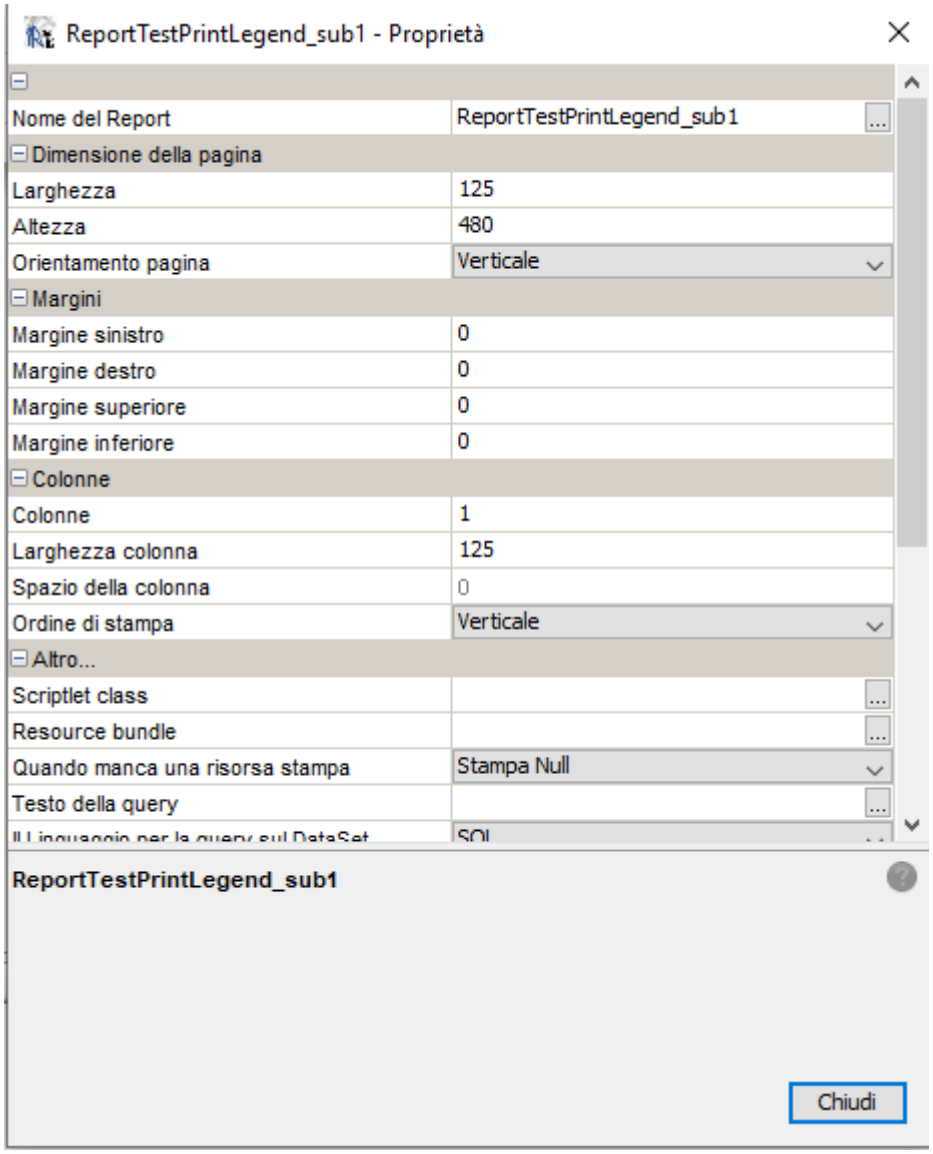


L'oggetto listItemLegend deve essere passato alla sottoreport della legenda, come fonte dati affinchè la lista di dati che contiene, possa essere considerata la lista dei dati da stampare nel detail di quest'ultima.





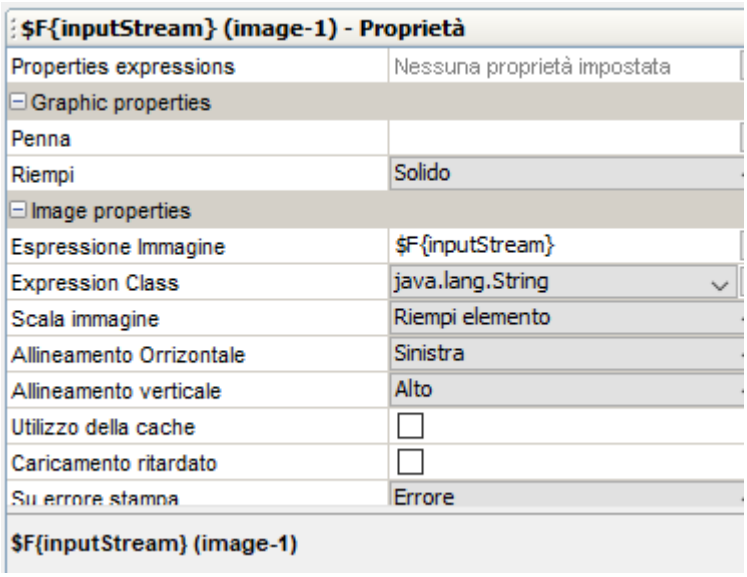
La sottoreport dovrà avere le stesse dimensioni riservategli nella report principale, e non dovrà avere bordi.



I campi della sottoreport coincidono con quelli del bean della lista dei dati passati nel JRDatasource:

1. text:String
2. inputStream:inputStream
3. icon:byte[]

L’input stream puo’ essere utilizzato in oggetto image in questa maniera:



Il byte[] può essere utilizzato in un oggetto image in questa maniera:

new java.io.ByteArrayInputStream((byte[]) $F{icon})

PRINT MAP LEGEND DA MENU DI MAPPA

E' stata implementata un nuovo command map, un bottone del menu di mappa, che consente di lanciare attraverso un menu a tendina, delle stampe pre-configurate. Tale funzione è specifica solo per le mappe di tipo planimetrico, quindi, se configurata per una mappa cartografica, non avrà alcun effetto.

Per la configurazione è necessario, inserire due groovy nella cartella groovy dei contenuti statici. Uno fisso printmaplegend_menu.groovy che forniamo in allegato, l'altro dovrà essere creato da voi basato sull'esempio fornito, e dovrà essere personalizzato in base alle vostre esigenze e per ciascuna mappa; Esso dovrà essere denominato ***"printMapLegend_"+nomedellamappa+".groovy"*** . Il Groovy dovrà restituire un oggetto ArrayList<HashMap<String, Object», ogni elemento della lista rappresenta una voce di menu del menu a tendina che verrà aperto al click sul bottone di mappa. Se la lista è vuota non verrà inserito il bottone nel menu di mappa. Ogni elemento della lista deve essere un oggetto HashMap contenente queste chiavi.

- projectGwId
- mgMapName
- orientation
- title
- printScale
- printDate
- adapt
- scale
- displayWidth
- displayHeight
- listIdDrawingSet
- dpi
- nameJasper
- ***reportName***
- ***layersOnOff***

- ***nameLegendLayer***
- ***label***
- ***id***

Mentre tutte le proprietà elencate possono essere identiche per tutte le voci di menu, quest'ultimi 5 punti in grassetto, dovrebbero distinguersi, Id (stringa o numero univoco nello stesso groovy) obbligatoriamente diverso da voce di menu a voce di menu.

Questa l'interfaccia che verrà mostrata:

Al click sulla voce, verrà eseguito il groovy printmaplegend_menu.groovy che eseguirà il metodo sopra descritto mapGuidePrintService.printLayout() con i parametri che gli verranno passati dalle proprietà impostate sulla voce di menu cliccata; Esso creerà il byteArray della report e richiamerà una funzione javascript che lo aprirà all'utente a video su una nuova pagina.