Export Plugin - Reference Documentation

Grails Plugin Collective

Version 2.0.0

Table of Contents

Chapter 1. Introduction

Welcome to the Export plugin for Grails.

This plugin offers export functionality supporting different formats e.g. CSV, Excel, Open Document Spreadsheet, PDF and XML and can be extended to add additional formats.

1.1. License

This plugin is released under the Apache License, Version 2.0

Please refer to https://github.com/gpc/export/blob/master/LICENSE.txt for more information about licensing scheme of Grails Export

1.2. Current version

Current version is **2.0.0** (for Grails 2.x the latest version is **1.6**)

Plugin release	Supported Grails Version
1.6	1.3 > 3.0
2.0.0	> 3.0

This version is developed with **Grails 3.0.3.BUILD-SNAPSHOT**

1.3. Version History

- July, 18, 2012 (version 1.5)
  - more hidden internal domain attributes excluded
- July, 18, 2012 (version 1.4)
  - Missing shadow.jpg file added
- May, 18, 2012 (version 1.3)
  - Option to disable header output added (header.enabled)
- May, 2, 2012 (version 1.2)
  - Exporter caching fields issue fixed
- February, 11, 2012 (version 1.1)
  - Support for resources plugin added
  - ConfigurationHolder removed
- July, 10, 2011 (version 1.0)
  - bug fixes

    - Excel now converts URLs to hyperlinks

- March, 13, 2011 (version 0.9)
    - bug fixes
    - Excel now supports title and column widths parameters

- January, 16, 2011 (version 0.8)
    - bug fixes
    - PDF support for multiple header rows and page orientation
    - DSL dependency management

- April, 10, 2010 (version 0.7)
    - multiple bug fixes
    - enhanced formatter closure with access to the current domain object
    - CSV
        - encoding (text encoding)
    - PDF
        - font.family (global font family setting)
        - title.encoding (encoding for title font)
        - header.encoding (encoding for header font)
        - text.encoding (encoding for text font)
        - pdf.encoding (global font encoding)
        - column.widths (allows to set different column widths)
    - RTF
        - font.family (global font family setting)
        - title.encoding (encoding for title font)
        - header.encoding (encoding for header font)
        - text.encoding (encoding for text font)
        - rtf.encoding (global font encoding)
        - column.widths (allows to set different column widths)
    - XML
        - encoding (text encoding)
        - xml.root (specify root element name)
        - depth (depth for building tree affects how collections and relationships are exported)

- November, 7, 2009 (version 0.6)

  - Better support for inherited domain class attributes

- September 28, 2009 (version 0.5)
  - ODS export issues fixed

- August 15, 2009 (version 0.4)
  - CSV
    - quoteChar fixed
    - lineEnd added
  - PDF
    - border.color added
    - separator.color added

- May 31, 2009 (version 0.3)
  - Excel file extension is now xls
  - RTF export added
  - Null values are now treated gracefully
  - Specify nonexistant fields and create output for them with closures e.g. for static content like today’s date

- Apr. 16, 2009 (version 0.2)
  - Issue with missing xercesImpl.jar fixed
  - Additional output parameters to PDF export added e.g. encoding, font size

- Feb. 15, 2009 (version 0.1)
  - Initial version released
  - Supported formats
    - CSV
    - Excel
    - ODS
    - PDF
    - XML

1.4. Acknowledgments

Following are the plugin contributors:

- Graeme Rocher
- Puneet Behl

- Nathan Wells
- Ruben
- Arturo Ojeda López
- Fabio Issamu Oshiro
- Dean Del Ponte
- Cristiano Limiti
- Mirko Weber
- Joás Garcia
- Fran García
- Dustin Clark

1.5. Source code

The full source code for this plugin can be found on GitHub.

The master branch is for the Grails 3.x version. The code for the Grails 2.x version is on branch: `grails-2.x`.

For issues, improvements or new features go to the plugin's go to GitHub issues.

To contact me directly my email address is puneet DOT behl007 AT gmail DOT com

Feel free to send me any correction about this document.

Chapter 2. Getting started

2.1. Installation

To install the plugin add a dependency to BuildConfig.groovy:

```
plugins {
    compile ":export:1.6"
}
```

To install the plugin with Grails 3.x:

```
dependencies {
    compile "org.grails.plugins:export:2.0.0"
}
```

You can find the dependency declaration for the latest version on the plugin portal page.

2.2. Usage

Export plugin uses resource plugin to provide resources (since version 1.1). Add the following markup to your GSP header:

```
<head>
...
    <r:require module="export"/>
...
</head>
```

To use the export features in plugin versions up to 1.0 add the following markup to your GSP header:

```
<export:resource />
```

Please refer to the [configuration] section to understand how to configure the file manager.

2.3. Export Bar

Add the export bar to your GSP page e.g. below the paginate buttons:

```
...
<div class="paginateButtons">
    <g:paginate total="${Book.count()}" />
</div>
<export:formats />

or

<export:formats formats="['csv', 'excel', 'ods', 'pdf', 'rtf', 'xml']" />

</div>
...
```

The formats tag supports the following attributes and allows you to pass through HTML attributes: * formats (Formats which should be displayed, List of Strings, e.g. ['csv', 'excel', 'ods', 'pdf', 'rtf', 'xml']) * params (Additional request parameters, Map, e.g. [sort: params?.sort, order: params?.order]) * action (Action which should be called, defaults to current action) * controller (Controller which should be called, defaults to current controller)

To make the plugin work you need to add some code to your controller:

```
class BookController {
    // Export service provided by Export plugin
    def exportService
    def grailsApplication  //inject GrailsApplication
...

    def list = {
        if(!params.max) params.max = 10

        if(params?.format && params.format != "html"){
            response.contentType =
grailsApplication.config.grails.mime.types[params.format]
            response.setHeader("Content-disposition", "attachment;
filename=books.${params.extension}")

exportService.export(params.format, response.outputStream,Book.list(params), [:], [:])
        }

        [ bookInstanceList: Book.list( params ) ]
    }

}
...
```

This will export all book objects with all attributes except version. You can also configure which attributes should be exported. The following example will rely on a simple domain class:

```
class Book {
    String title
    String author
}
```

2.4. Customizing labels on export bar

You can customize the labels displayed on the export bar by adding the following lines to grails-app/i18n/messages.properties:

```
default.csv=CSV
default.excel=Excel
default.pdf=PDF
default.rtf=RTF
default.xml=XML
default.ods=ODS
```

2.5. Customize output of the fields

You can see how the fields are now set explicitly. You can use labels to customize the output of the fields e.g. for i18n. Using formatters it is possible to customize how the resulting values are generated which can be used to specify a particular date format etc. by just mapping an attribute name to a closure. It is also possible to specify non-existant fields and add formatters for those fields to produce static content like today’s date.

```
class BookController {
    def exportService
    def grailsApplication  //inject GrailsApplication
...

    def list = {
        if(!params.max) params.max = 10

        if(params?.format && params.format != "html"){
            response.contentType =
grailsApplication.config.grails.mime.types[params.format]
            response.setHeader("Content-disposition", "attachment;
filename=books.${params.extension}")


            List fields = ["author", "title"]
            Map labels = ["author": "Author", "title": "Title"]

            /* Formatter closure in previous releases
            def upperCase = { value ->
                return value.toUpperCase()
            }
            */

            // Formatter closure
            def upperCase = { domain, value ->
                return value.toUpperCase()
            }

            Map formatters = [author: upperCase]
            Map parameters = [title: "Cool books", "column.widths": [0.2, 0.3, 0.5]]

            exportService.export(params.format, response.outputStream,
Book.list(params), fields, labels, formatters, parameters)
        }

        [ bookInstanceList: Book.list( params ) ]
    }

}
...
```

Chapter 3. Additional Parameters

With the new formatter closure you can combine multiple attributes:

```
// Formatter closure
      def title = { domain, value ->
    return domain?.author + ": " + domain?.title
}
```

⚠ Version 0.7 changes formatter closures. To upgrade from a previous release you need to add the domain argument to your closures as shown in the code sample above.

3.1. PDF Export

PDF export supports some additional parameters which can be used just like the title attribute in the code sample above. The following parameters are supported:

Parameter	Default Value	Allowed Values	Description
pdf.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman") see http://itextdocs.lowagie.com/tutorial/fonts/index.php for further information about encodings	Specifies font encoding
title.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman")	Title font encoding
header.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman")	Header font encoding
text.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman")	Text font encoding
title.font.size	"10"	a number as String	Title font size
header.font.size	"10"	a number as String	Header font size
text.font.size	"10"	a number as String	Text font size

font.family	com.lowagie.text.FontFactory.HELVETICA	constants defined in http://www.1t3xt.info/api/com/lowagie/text/FontFactory.html	Global font family
title.font.family	com.lowagie.text.FontFactory.HELVETICA	constants defined in http://www.1t3xt.info/api/com/lowagie/text/FontFactory.html	Title font family
header.font.family	com.lowagie.text.FontFactory.HELVETICA	constants defined in http://www.1t3xt.info/api/com/lowagie/text/FontFactory.html	Header font family
text.font.family	com.lowagie.text.FontFactory.HELVETICA	constants defined in http://www.1t3xt.info/api/com/lowagie/text/FontFactory.html	Text font family
title.font.style	"bold"	"bold", "italic", "normal", "bolditalic"	Title font style
header.font.style	"bold"	"bold", "italic", "normal", "bolditalic"	Header font style
text.font.style	"bold"	"bold", "italic", "normal", "bolditalic"	Text font style
border.color	new Color(163, 163, 163)	a java.awt.Color object e.g. Color.RED)	border color
separator.color	new Color(238, 238, 238)	a java.awt.Color object e.g. new Color(100, 100, 100))	rowÏ separator color
column.widths	equal size for all columns	List of floats e.g. [0.2, 0.3, 0.5] for three columns	Column widths in percent'
header.rows	"1"	A number as string	Number of header rows
header	defaults to fields	list of list of strings e.g. [["Intended times"], ["Actual times", "Duration"]]	Header fields
header.parameters	no colspan	list of maps e.g. [["colspan0": 3], ["colspan0": 1, "colspan1": 2]]	Header parameters currently supported colspan
pdf.orientation	landscape	portrait or landscape	Page orientation
header.enabled	true	boolean true or false	Enable/Disable header output

For chinese characters in PDF use the following parameters:

```
...

Map parameters = ["pdf.encoding":"UniGB-UCS2-H", "font.family": "STSong-Light"]

...
```

3.2. CSV Export

CSV export also supports some additional parameters:

Parameter	Default Value	Allowed Value	Description
encoding	JVM default encoding	http://java.sun.com/javase/6/docs/api/java/nio/charset/Charset.html e.g. "UTF-8"	encodings
separator	,	a single charater	field separator
quoteCharacter	"	a single character	use 'u0000' for no quote character and put a slash before the four 0s, quote character
lineEnd	default platform line ending	String e.g. "rn"	line ending
header.enabled	true	true/false	enable/disable header output

3.3. Excel Export

Parameter	Default Value	Allowed Values	Description
column.widths	equal size for all columns	List of floats e.g. [0.2, 0.3, 0.5]	column widths in percent
header.enabled	true	true/false	enable/disable header output

3.4. RTF Export

ℹ Supports mainly the same parameters as PDF

Parameters	Default Value	Allowed Values	Description

rtf.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman") see http://itextdocs.lowagie.com/tutorial/fonts/index.php for further information about encodings	Font encoding
title.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman") see http://itextdocs.lowagie.com/tutorial/fonts/index.php for further information about encodings	Title font encoding
header.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman") see http://itextdocs.lowagie.com/tutorial/fonts/index.php for further information about encodings	Header font encoding
text.encoding	"Cp1252" (=latin 1)	"Cp1250", "Cp1252" (=latin 2), "Cp1257", "Identity-H", "Identity-V", "MacRoman") see http://itextdocs.lowagie.com/tutorial/fonts/index.php for further information about encodings	Text font encoding
header.enabled	true	true/false	enable/disable header output

3.5. XML Export

Parameters	Default Value	Allowed Values	Description
encoding	JVM default encoding	http://java.sun.com/javase/6/docs/api/java/nio/charset/Charset.html e.g. "UTF-8")	encoding
xml.root	object class name	a String	root element name

depth	1	1 means only direct domain attributes 2 or more collection attributes as well	depth (as integer) for building tree affects how collections and relationships are exported