Base Language - Feature #3342

implement a facility to easily run a JasperReports report definition from 4GL code, passing data queried/calculated in that 4GL code

09/28/2017 09:47 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Stanislav Lomany	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Feature #3261: enhanced browse that can optionall...	**Closed**
Related to Base Language - Feature #3869: enhance JasperReports support with ...	**New**

History

#1 - 09/28/2017 10:05 AM - Greg Shah

This is supposed to be a generic facility that can be used by any customer. We did something similar to this in #1389 for a specific customer.

That original version (if I recall correctly) had dummy 4GL functions and customer-specific conversion rules to map those into calls to hand-coded Java.

The JasperReports library will be integrated with our server as a standard dependency. It will always be there. The runtime integration and helpers will always be there. A customer would do the following to use it:

1. Create the JasperReports report template/definition that is used by the JasperReports library/engine.
2. Write 4GL code to do queries, calculate values.
3. Write 4GL code to use FWD runtime helpers to generate reports, passing in whatever 4GL data is needed.

The idea is to provide a pure Java, standard mechanism in FWD for doing reporting using 4GL code and data.

I think the new generic version should not require dummy functions. If we already had OO 4GL support that could directly call Java classes, I would want to use that approach.

For now, the OO 4GL facilities are not ready. So we will take an alternate approach. My preference is to create a 4GL Syntax Extension. This would involve creating a new handle-based resource type and the minimum methods/attributes needed to control it. This is closest to the OO 4GL approach and could be easily migrated to it. It will require changes to the parser to add keywords, a CREATE-style language statement and the methods/attributes needed. It will also require the runtime version of this "legacy-like" handle resource.

As part of this work, we must create a sample report and some sample 4GL code to use it. This will be featured in the documentation and it must be complex enough to do real reports.

#2 - 10/27/2017 05:59 AM - Stanislav Lomany

- File AddressReport.pdf added

My suggestion on how 4GL code for a jasper report should look like:

```
def query q for address.

def var rpt as handle.
create report rpt.
rpt:report-query = query q:handle.
rpt:report-design = "AddressReport.jasper".
rpt:set-report-param("ReportTitle", "Address Report").
rpt:export-report("pdf", "AddressReport.pdf").
```

The attached report is the one I suggest to implement. Is it complex enough for Monday?

#3 - 10/27/2017 06:10 AM - Greg Shah

Overall, I like the approach. It is different from the callback approach we took previously. Some questions:

1. Do I understand correctly that the runtime will figure out the mapping between the columns returned by the query and the report definition? How does that work?

2. It seems to me that some reports might take more than one query to provide the needed details. Is that possible?

3. How do we handle things like grouping (e.g. sub-totals)?

4. How do we handle calculated columns?

5. Can we have arbitrary per-page or per-report data items that can be provided as substitutions for the corresponding entries in the report definition? These would be values that would not necessarily come from a query.

I really like the register a query idea. It will handle a good range of common reporting cases. But I do think we may still need a call-back model to implement the full range of possibilities. But before we go there, I need to understand your proposed solution better (with the questions above).

> Is it complex enough for Monday?

Probably so.

#4 - 10/27/2017 07:24 AM - Stanislav Lomany

- File DataSourceReport.jrxml added

> 1. Do I understand correctly that the runtime will figure out the mapping between the columns returned by the query and the report definition? How does that work?

There is JRDataSource interface which has two functions: next() and getFieldValue(). We implement a custom data source which in getFieldValue() checks field name and finds appropriate field among query buffers.

> 2. It seems to me that some reports might take more than one query to provide the needed details. Is that possible?

The only (AFAIK) way do that is subreports. Every subreport has its own dataset. To implement subreports, we can combine 4GL wrappers, e.g. rpt:add-subreport(subreport).

> 3. How do we handle things like grouping (e.g. sub-totals)?

It is all defined in report design file.

> 4. How do we handle calculated columns?

In report design file too.

> 5. Can we have arbitrary per-page or per-report data items that can be provided as substitutions for the corresponding entries in the report definition? These would be values that would not necessarily come from a query.

Yes, e.g. rpt:set-report-param("ReportTitle", "Address Report") above sets report title which is in-report parameter:

```
<textFieldExpression><![CDATA[$P{ReportTitle}]]></textFieldExpression>
```

For Jasper reports everything is defined in a report design file, in Java code you only have to connect it with a data set. I've attached sample report design file in case you want to look at it.

#5 - 10/27/2017 07:32 AM - Greg Shah

> 2. It seems to me that some reports might take more than one query to provide the needed details. Is that possible?

> The only (AFAIK) way do that is subreports. Every subreport has its own dataset. To implement subreports, we can combine 4GL wrappers, e.g. rpt:add-subreport(subreport).

OK, this seems to be a reasonable approach.

> 4. How do we handle calculated columns?

> In report design file too.

The idea here is that there would be 4GL code to calculate the value of a column, possibly with access to state (vars, screen-buffer...) from the business logic. How does this get handled?

Also: how does the proposed facility's capabilities compare with the approach we integrated for the pilot customer (many years ago)? In particular, are there things that could be done with that customer's approach that cannot be done with this proposed plan?

#6 - 10/27/2017 07:33 AM - Greg Shah

Can you show an example of embedding an image in the report?

#7 - 10/27/2017 08:38 AM - Stanislav Lomany

> The idea here is that there would be 4GL code to calculate the value of a column, possibly with access to state (vars, screen-buffer...) from the business logic. How does this get handled?

We have our own data source, so we can add arbitrary 4GL function as a calculated parameter: rpt:add-calc-param("param-name", "function-name").

> are there things that could be done with that customer's approach that cannot be done with this proposed plan?

The answer is "I don't think so". Customer uses 1. HashMaps as parameters 2. some filtering logic in Java code. Japer has in-report filtering capabilities as well. And eventually a customer can create a temp-table and filter and modify data in it in any way he likes.

> Can you show an example of embedding an image in the report?

BufferedImages can be passed as report parameters. So it has to be declared before using report:

```
rpt:set-report-image("logo", "logo.png").
```

Sure you can use an external image in report by just specifying its file name.

#8 - 10/27/2017 08:41 AM - Greg Shah

OK, go with this plan.

#9 - 10/28/2017 07:15 AM - Stanislav Lomany

Created task branch 3342a from P2J trunk revision 11183.

#10 - 10/29/2017 02:54 PM - Stanislav Lomany

Quick question: we have <customer_package_redacted>.BufferedOutputStreamWrapper class. Can I copy-paste it under com.goldencode package?

#11 - 10/29/2017 11:06 PM - Greg Shah

> we have <customer_package_redacted>.BufferedOutputStreamWrapper class. Can I copy-paste it under com.goldencode package?

No, we don't own that. But we have com.goldencode.p2j.util.OutputStreamWrapper which extends OutputStream. Can you use that and wrap it in a BufferedOutputStream?

#12 - 10/30/2017 09:57 AM - Stanislav Lomany

Into which package should I put all classes related to (Jasper) reports? We have com.goldencode.p2j.report, but it is for AST reporting.

#13 - 10/30/2017 11:31 AM - Greg Shah

Use com.goldencode.p2j.reporting. The overlap is confusing but we will deal with it for now.

#14 - 10/30/2017 06:06 PM - Stanislav Lomany

Rebased task branch 3342a from P2J trunk revision 11186.

#15 - 10/30/2017 06:41 PM - Stanislav Lomany

Rebased task branch 3342a from P2J trunk revision 11187.

#16 - 10/30/2017 06:48 PM - Stanislav Lomany

Rebased task branch 3342a from P2J trunk revision 11188.

#17 - 10/30/2017 07:27 PM - Greg Shah

Code Review Task Branch 3342a Revision 11195

The changes overall are really good. Some of the conversion changes were not quite right. I have made edits in rev 11196 to make it consistent.

Please do a quick test of your example report with this version to make sure I didn't break it. If it passed conversion testing, you can merge to trunk.

#18 - 10/30/2017 09:23 PM - Stanislav Lomany

There seems to be conversion mismatch: order of widgets changed in frame definition:

```
diff -r generated/20171029b/src/aero/timco/majic/ui/train2/EmpTrain01Femptrain.java generated/20171030a/src/ae
ro/timco/majic/ui/train2/EmpTrain01Femptrain.java
294,299d293
<          effectiveDate.setDbname("majic");
<          effectiveDate.setTable("crew");
<          effectiveDate.setHelpStatic("Enter date when this crew takes effect");
<          crewCode.setDbname("majic");
<          crewCode.setTable("crew");
<          crewCode.setHelpStatic("Enter the code used to identify the crew");
302a297,299
>          crewCode.setDbname("majic");
>          crewCode.setTable("crew");
>          crewCode.setHelpStatic("Enter the code used to identify the crew");
305a303,305
>          effectiveDate.setDbname("majic");
>          effectiveDate.setTable("crew");
>          effectiveDate.setHelpStatic("Enter date when this crew takes effect");
```

Note that I've compared conversion results against 11185. What do you think?

#19 - 10/30/2017 09:24 PM - Greg Shah

I saw the same thing in my run. I don't think it will make a difference. I think you are OK.

#20 - 10/30/2017 09:38 PM - Stanislav Lomany

3342a has been merged into the trunk as bzr revision 11189.

#21 - 10/31/2017 08:07 AM - Stanislav Lomany

Created task branch 3342b from P2J trunk revision 11190.

#22 - 10/31/2017 09:07 AM - Stanislav Lomany

Rebased task branch 3342b from P2J trunk revision 11191.

#23 - 10/31/2017 09:13 AM - Greg Shah

Code Review Task Branch 3342b Revision 11192

I'm fine with the change. Please merge 3342b to trunk.

#24 - 10/31/2017 09:21 AM - Stanislav Lomany

3342b has been merged into the trunk as bzr revision 11192.

#25 - 10/31/2017 09:26 AM - Stanislav Lomany

Created task branch 3342c from P2J trunk revision 11192.

#26 - 11/01/2017 09:35 PM - Stanislav Lomany

- File AddressReport.jrxml added

- File jasper.p added

- File home.png added

Rebased task branch 3342c from P2J trunk revision 11193. Please review. Contains final fixes for Jasper. Images were working "out of the box", but I made changes so image names are specified relatively to the reports base directory rather than current directory (otherwise specifying image names in jrxml may be confusing).

I've added a testcase for the case if somebody takes this task.

#27 - 11/02/2017 06:11 AM - Greg Shah

Code Review Task Branch 3342c Revision 11194

I'm fine with the changes.

One question: why not default the reports directory to "." instead of "reports"? It seems to me that using "reports" will require extra work to setup the user's directory structure for this code to work OR the override will have to be put into the directory.xml. By using ".", things will work naturally without extra setup. If the customer wants to do something different they can configure that.

Please upload the .pdf output for your sample.

#28 - 11/02/2017 08:29 AM - Stanislav Lomany

- File AddressReport.pdf added

> One question: why not default the reports directory to "." instead of "reports"?

On my taste the best place for reports is application_root_directory/reports (application_root_directory is where p2j, deploy etc reside). But unfortunately the current directory is the P2J server directory. Do we have a way to find application root directory?

OK, I'll change the directory to "." (i.e. P2J server directory). Should I merge to trunk after that?

Output PDF is attached.

#29 - 11/02/2017 09:54 AM - Greg Shah

> But unfortunately the current directory is the P2J server directory.

Sorry, I completely missed the part about files coming from the server.

That present different issues:

1. We do expect that application-specific resources would come from the application's jar file. This would be on the server and would be something we want to support. The .jrxml file and any predefined image resources would fall into this category.

2. If there is an opportunity for user-defined input to define the filename, then there is a security danger because the user can specify anything on the server side including private key keystores or the directory.xml. I think anything to or from the filesystem needs to be on the client. Any image resource not found in the jar should be found on the client system. The output .pdf should be written to the client system.

I believe we already have much of the backing support for both of these behaviors in the FWD runtime. But the question is how to hook it in to the jasper environment?

- Eugenie wrote image loading code that first checks the application's jar and then searches the propath on the client. The only problem with this is that the images are loaded in the FWD client because the actual byte stream is not used on the FWD server. Yet, I think that the support can be leveraged in some way.
- We have the InputStreamWrapper and OutputStreamWrapper that allow us to read/write on the server but have those operations redirected from/to the client.
- Can you change the JR file resolver to operate from an InputStream/OutputStream that we provide?

#30 - 11/02/2017 12:26 PM - Stanislav Lomany

> 1. We do expect that application-specific resources would come from the application's jar file. This would be on the server and would be something we want to support. The .jrxml file and any predefined image resources would fall into this category.

I'll add support for in-jar files.

> 2. If there is an opportunity for user-defined input to define the filename, then there is a security danger because the user can specify anything on the server side including private key keystores or the directory.xml.

Yes, they can point an arbitrary file, but it can be a compilable jasper report only. Do you want me to remove capability to read non-in-jar files or limit access to the reports directory only?

> I think anything to or from the filesystem needs to be on the client. Any image resource not found in the jar should be found on the client system.

Why do we need loading images from client? Report templates are shared among all users, why should images be client-specific?

#31 - 11/02/2017 12:49 PM - Greg Shah

> Do you want me to remove capability to read non-in-jar files or limit access to the reports directory only?

From (unmodified) converted code there is no way to write or read from the server filesystem. That means that the application's 4GL code cannot dynamically generate anything in the reports dir. For this reason, there is little need for that server-side directory because anything static can be easily put into the jar.

I can see a need for being able to access dynamically generated content. Is it a problem to use the InputStreamWrapper to read these resources from the client?

> Why do we need loading images from client? Report templates are shared among all users, why should images be client-specific?

I think it would be the case where the application generates some output as an image. Are images the only kind of resource that can be read from a file?

#32 - 11/02/2017 01:21 PM - Stanislav Lomany

> From (unmodified) converted code there is no way to write or read from the server filesystem. That means that the application's 4GL code cannot dynamically generate anything in the reports dir. For this reason, there is little need for that server-side directory because anything static can be easily put into the jar.

Yes, but I have to note that having files outside of jar is very convenient at report development stage. So, limit access to jars only?

> I can see a need for being able to access dynamically generated content. Is it a problem to use the InputStreamWrapper to read these resources from the client?

No, it's not a problem.

> I think it would be the case where the application generates some output as an image.

Images can be 1. referenced inside a jar; 2. use binary data from database; 3. yes, there may be data from client, but no real world example that doesn't fall into two previous categories comes to my mind.

> Are images the only kind of resource that can be read from a file?

Images, subreports, style templates (not sure it is the full list).

#33 - 11/02/2017 01:27 PM - Constantin Asofiei

Stanislav Lomany wrote:

> From (unmodified) converted code there is no way to write or read from the server filesystem. That means that the application's 4GL code cannot dynamically generate anything in the reports dir. For this reason, there is little need for that server-side directory because anything static can be easily put into the jar.

> Yes, but I have to note that having files outside of jar is very convenient at report development stage. So, limit access to jars only?

Why not treat any file needed by the jasper report generation as a generic resource? For .jrxml case, we can search in the server's jars first and, if not found, search it on the client-side (same way we resolve images and other stuff).

The problem we want to avoid is to let the user have access to the server OS filesystem explicitly, correct? We can leverage the search-path or propath too, when we want to lookup resources on client-side.

#34 - 11/02/2017 01:40 PM - Stanislav Lomany

> we can search in the server's jars first and, if not found, search it on the client-side

OK, I'll go this way. Should we have context-specific cache for compiled reports? For other files?

#35 - 11/02/2017 01:43 PM - Greg Shah

> Yes, but I have to note that having files outside of jar is very convenient at report development stage.

If you feel that the client-side support doesn't provide the same capability, then it is OK to do a 3 level process:

1. jar file
2. server-side report dir
3. client side via propath

> Why not treat any file needed by the jasper report generation as a generic resource? For .jrxml case, we can search in the server's jars first and, if not found, search it on the client-side (same way we resolve images and other stuff).

Agreed. This is the right way to go.

> Should we have context-specific cache for compiled reports? For other files?

Unless it helps you in some way, I don't think we need to worry about caching at this time.

#36 - 11/02/2017 02:20 PM - Stanislav Lomany

> If you feel that the client-side support doesn't provide the same capability, then it is OK to do a 3 level process:
>
> 1. jar file
> 2. server-side report dir
> 3. client side via propath

Actually, client-side provides the same capability, so I'm inclined to exclude level 2.

> Unless it helps you in some way, I don't think we need to worry about caching at this time.

Compiling reports is a big performance hit. Transferring image files may be pricey too.

#37 - 11/02/2017 02:27 PM - Constantin Asofiei

Stanislav Lomany wrote:

> Compiling reports is a big performance hit.

Isn't compiling the jasper file done only once, after the design is complete? Or do you want the 4GL business logic to design the report from scratch, too, and compile it at runtime?

> Transferring image files may be pricey too.

Report generation is done on server-side, right? I think it would be beneficial to cache the report's resources in each context; but we should be careful for not allowing too large files to be cached (i.e. the user selects a very big file which will OOME the server).

#38 - 11/02/2017 03:01 PM - Stanislav Lomany

> Isn't compiling the jasper file done only once, after the design is complete?

Right. If we are going to put reports into jar, they can be compiled before that. But having access to report xml is beneficial at design time.

> Or do you want the 4GL business logic to design the report from scratch, too, and compile it at runtime?

That may be the case for enhanced browse (export it to cvs/xls/pdf).

So I think I'll make support for .jrxml as well as .jasper (compiled) reports.

> Report generation is done on server-side, right? I think it would be beneficial to cache the report's resources in each context; but we should be careful for not allowing too large files to be cached (i.e. the user selects a very big file which will OOME the server).

Right, report generation is server-side. One note: IMO the *only* way to use client-side Jasper resources is report design. And in this case caching better to be off.

#39 - 11/02/2017 03:07 PM - Constantin Asofiei

Stanislav Lomany wrote:

> > Or do you want the 4GL business logic to design the report from scratch, too, and compile it at runtime?
>
> That may be the case for enhanced browse (export it to cvs/xls/pdf).
>
> So I think I'll make support for .jrxml as well as .jasper (compiled) reports.

In this case, compiling the .jrxml will be done 'on-the-fly', don't know how easy is to cache it (especially build a key to identify two equal report designs).

> One note: IMO the *only* way to use client-side Jasper resources is report design. And in this case caching better to be off.

Something else: someone can design a report which refers to a client-side resource (i.e. an image local to the client). Another way can be to specify an image path as a parameter which is set by the business logic - which again, can be anything. How will we handle this?

BTW, please add the annotations to the reporting.Report interface, to mark the methods, attributes and class (see any existing interface which handles a legacy resource for an example).

#40 - 11/02/2017 03:08 PM - Constantin Asofiei

And also add the reporting.Report interface to the HandleCommon interface; any extension resource in FWD must be registered and configured the same way as a legacy one.

#41 - 11/02/2017 03:46 PM - Stanislav Lomany

> Something else: someone can design a report which refers to a client-side resource (i.e. an image local to the client).

Constantin, could you please give me an example why user may want that? Are you suggesting that there may be reports that customer do not want to keep at server side? I just work better if I understand that I'm doing a useful feature.
Suggest that we are talking about PDF output.

#42 - 11/02/2017 03:51 PM - Stanislav Lomany

> In this case, compiling the .jrxml will be done 'on-the-fly', don't know how easy is to cache it (especially build a key to identify two equal report designs).

I supposed we can cache reports by file name: I thought we have a fixed set of reports. Client-side reports can be cached by file name as well (if we need caching for client side).

> Something else: someone can design a report which refers to a client-side resource (i.e. an image local to the client). Another way can be to specify an image path as a parameter which is set by the business logic - which again, can be anything. How will we handle this?

We just get an image (or anything) from server-side jar or upload from client.

#43 - 11/02/2017 04:00 PM - Constantin Asofiei

Stanislav Lomany wrote:

> Something else: someone can design a report which refers to a client-side resource (i.e. an image local to the client).

Constantin, could you please give me an example why user may want that? Are you suggesting that there may be reports that customer do not want to keep at server side?

Some department header images which are local to each client installation, and the report is designed with a file name and not an image stream... but thinking about it, my reasoning above was more in line with an already built ABL logic. This new feature is ours, and we can decide how it works and what limitations it has.

#44 - 11/02/2017 04:02 PM - Constantin Asofiei

Stanislav Lomany wrote:

> In this case, compiling the .jrxml will be done 'on-the-fly', don't know how easy is to cache it (especially build a key to identify two equal report designs).

I supposed we can cache reports by file name: I thought we have a fixed set of reports. Client-side reports can be cached by file name as well (if we need caching for client side).

The generated report doesn't necessarily has to keep the name on different executions (an example are reports ran from BROWSE widget): think about running customer reports, the report layout is the same, but the filename can be set to be customer's name.

#45 - 11/02/2017 04:05 PM - Greg Shah

If I understand correctly, the use of jasperContext.setFileResolver(fileName -> new File(ReportFactory.getFilePath(fileName))); will allow us to intercept file access and control resolution to the two level approach we've agreed.

#46 - 11/02/2017 04:11 PM - Stanislav Lomany

> If I understand correctly, the use of jasperContext.setFileResolver(fileName -> new File(ReportFactory.getFilePath(fileName))); will allow us to intercept file access and control resolution to the two level approach we've agreed.

Right.

#47 - 11/02/2017 04:27 PM - Stanislav Lomany

> The generated report doesn't necessarily has to keep the name on different executions (an example are reports ran from BROWSE widget): think about running customer reports, the report layout is the same, but the filename can be set to be customer's name.

There are two file names involved: input file which is report design file (it should be compiled before running report), compiled report should be cached. And output file (exported report with data), which can change from run to run, it shouldn't be cached in any way.

#48 - 11/03/2017 09:35 AM - Stanislav Lomany

> Eugenie wrote image loading code that first checks the application's jar and then searches the propath on the client.

Where I can find it?

#49 - 11/03/2017 10:56 AM - Eugenie Lyzenko

Stanislav Lomany wrote:

> > Eugenie wrote image loading code that first checks the application's jar and then searches the propath on the client.
>
> Where I can find it?

LogicalTerminal.getImageStreamFromApplication(). Also ImageHelper class to see pending images processing for client side.

#50 - 11/03/2017 11:03 AM - Greg Shah

You can also look at LogicalTerminal.setImage() to see how it is a 2 level search.

#51 - 11/06/2017 01:50 PM - Stanislav Lomany

I suppose that MultiClassLoader can get resource from any jar (see MultiClassLoader.findResource). Is there any difference with LT.getImageStreamFromApplication?

#52 - 11/08/2017 05:32 PM - Stanislav Lomany

Rebased task branch 3342c from P2J trunk revision 11197. Please review.

#53 - 11/08/2017 06:38 PM - Greg Shah

Code Review Task Branch 3342c Revision 11199

I really like it.

The only request: the getFileFromClient() should probably be in FileSystemOps/FileSystemDaemon instead of in ThinClient.

With that change, it seems ready to go. Some limited testing to ensure that the FWD client works at all, should be enough. You can test that with Hotel GUI or ChUI.

#54 - 11/09/2017 06:15 AM - Stanislav Lomany

Please take a look at 3342c rev 11200 and I'll merge to trunk.

#55 - 11/09/2017 06:25 AM - Greg Shah

Code Review Task Branch 3342c Revision 11200

I'm good with the code.

Please merge to trunk.

#56 - 11/09/2017 06:49 AM - Stanislav Lomany

- Status changed from New to WIP

3342c has been merged into the trunk as bzr revision 11198.

#57 - 11/09/2017 06:50 AM - Stanislav Lomany

- Status changed from WIP to Review

#58 - 11/09/2017 07:06 AM - Greg Shah

- Status changed from Review to Closed

- % Done changed from 0 to 100

#59 - 11/10/2017 02:17 PM - Greg Shah

Stanislav:

Please write up the documentation for this feature. It should be placed in:

JasperReports Integration

Please pattern it after Email Send.

#60 - 01/08/2018 04:16 PM - Greg Shah

What is the effort to enable output to XLS?

#61 - 01/08/2018 04:22 PM - Stanislav Lomany

1-8 hours depending on if there will be issues with Jasper.

#64 - 01/17/2018 06:41 PM - Greg Shah

Please go ahead with adding support for output to XLS. Make sure that the following are all tested:

- control over column widths (different columns can have different widths)
- control over row height (different rows can have different heights)
- fonts including size and attributes
- colors (background and foreground)
- cell borders
- formulas

#65 - 01/19/2018 08:25 AM - Greg Shah

I'm trying to do some planning based on your results with jasper XLS support. Are you having any difficulties with the features in #3342-64? When can you have something working checked in?

I think you can put the changes in 3435a, they should be pretty safe.

#66 - 01/19/2018 09:06 AM - Stanislav Lomany

> Are you having any difficulties with the features in #3342-64?

That depends if color and font pickers work. Not sure what do you mean by "formulas".

> When can you have something working checked in?

In the middle or end of the next week.

#67 - 01/19/2018 09:06 AM - Greg Shah

- Related to Feature #3261: enhanced browse that can optionally selected as a replacement for the default ABL browse added

#68 - 01/19/2018 09:32 AM - Greg Shah

Spreadsheets commonly have formulas in cells, instead of just data values. A formula might accumulate the sum of all values in a range of cells A1 through A4 using a formula that looks like this =SUM(A1:A4). Of course, much more complex logic is possible. But ultimately, a formula is just a very specific text-based input that needs to be in a cell. It is such a common requirement that it would be very surprising if it was not possible.

Formulas won't be used for the enhanced browse output but they are definitely needed for the direct jasper usage that generates an XLS.

#69 - 01/19/2018 09:35 AM - Greg Shah

> In the middle or end of the next week.

Will this include the previous work that was done to export browse (to PDF/CSV/XLS)? I think the popup menu issues are gone as well as the other issues that were causing you some trouble.

#70 - 04/17/2018 01:06 AM - Stanislav Lomany

- File brws-jasper.p added

Here is the testcase that allows to run all kinds of browse Jasper reports. Note that the word cutting problem is still in place.

#71 - 04/17/2018 05:28 AM - Greg Shah

This doesn't seem to be a complete sample. There are no .jrxml report definitions. There are no queries being attached. There are no parameters being set.

Please provide a complete sample that shows how to output XLS.

#72 - 04/17/2018 05:31 AM - Greg Shah

To be clear: the example you posted is about the "hidden" browse export to various formats. That is not what we are looking for.

We need a standalone program that demonstrates how to create an XLS file using the REPORT handle-based object and no UI code.

#73 - 04/17/2018 06:04 AM - Greg Shah

Please make sure the testcase shows the features listed in #3342-64.

#74 - 04/17/2018 09:12 PM - Stanislav Lomany

> control over row height (different rows can have different heights)

You can control height of all rows at once. And you can control overflow rules which may end up in automatic increase of the height of a specific row. Not sure why a user may want to control row height in other cases.

> formulas

That may be tricky. You can use in-jasper formulas and, probably, you should, if you can. You can use excel formulas either, but the problem is that formulas contain references to specific cells, like A2 or D5, but you never know how is report going to be mapped to the actual cells.

#75 - 04/17/2018 10:20 PM - Stanislav Lomany

- File ReportFormatting.jrxml added

- File jasper-xls-formatting.p added

This is the testcase for xls output. .jrxml files (or .jasper files, which are compiled .jrxml s) are loaded from the client directory first (they are cached, so you have to restart server if you want to change .jrxml), then from server jars.
Note that there may be vertical text overflow for stretched cells. This overflow looks differently on different devices, so I'm not yet sure how can we possible handle it.

#76 - 04/18/2018 07:18 AM - Greg Shah

> Not sure why a user may want to control row height in other cases.

One reason may be to implement some find of fixed formatting/spacing to achieve a specific output result when printed. For example, the program creates a spreadsheet for a sales quote. The user is allowed to make arbitrary edits to the quote using Excel/Libreoffice/Google Sheets. Then the user prints the quote and expects a certain look and feel. Such a thing might require very specific sizing for individual rows.

> You can use excel formulas either, but the problem is that formulas contain references to specific cells, like A2 or D5, but you never know how is report going to be mapped to the actual cells.

There will be cases where the program really does need to provide spreadsheet formulas so that the result can be manipulated in Excel/Libreoffice/Google Sheets. In other words, we need to support the idea that the user will have a real functional spreadsheet as a result of this output.

Eugenie: you will need to investigate how the JasperReports output can be forced to map to specific cells. This seems important to resolve.

#77 - 06/26/2018 05:26 PM - Greg Shah

What is the development effort needed to add the ability to support multiple queries for a single report? We have customer requirements to support reports that have sub-reports or sub-groupings that would map to separate queries. If I understand correctly, we can only provide a single query today.

Can we come up with a solution that gives a wide range of flexibility in this area?

#78 - 07/12/2018 09:51 AM - Greg Shah

Greg Shah wrote:

> What is the development effort needed to add the ability to support multiple queries for a single report? We have customer requirements to support reports that have sub-reports or sub-groupings that would map to separate queries. If I understand correctly, we can only provide a single query today.
>
> Can we come up with a solution that gives a wide range of flexibility in this area?

Stanislav: Do you have thoughts on this?

#79 - 07/12/2018 10:34 AM - Stanislav Lomany

> What is the development effort needed to add the ability to support multiple queries for a single report? We have customer requirements to support reports that have sub-reports or sub-groupings that would map to separate queries. If I understand correctly, we can only provide a single query today.
> Can we come up with a solution that gives a wide range of flexibility in this area?

I think we can wrap Jasper sub-reports with FwdReport instances and combine FwdReport objects to create a parent report with multiple sub-reports. Not sure about time estimates for that.

#80 - 01/11/2019 11:58 AM - Greg Shah

- Related to Feature #3869: enhance JasperReports support with sub-reports and ProDataSets added

Files

AddressReport.pdf	13.4 KB	10/27/2017	Stanislav Lomany
DataSourceReport.jrxml	7.72 KB	10/27/2017	Stanislav Lomany

home.png	6.55 KB	11/02/2017	Stanislav Lomany
AddressReport.jrxml	7.74 KB	11/02/2017	Stanislav Lomany
jasper.p	3.82 KB	11/02/2017	Stanislav Lomany
AddressReport.pdf	20.1 KB	11/02/2017	Stanislav Lomany
brws-jasper.p	1.28 KB	04/17/2018	Stanislav Lomany
jasper-xls-formatting.p	3.56 KB	04/18/2018	Stanislav Lomany
ReportFormatting.jrxml	9.82 KB	04/18/2018	Stanislav Lomany