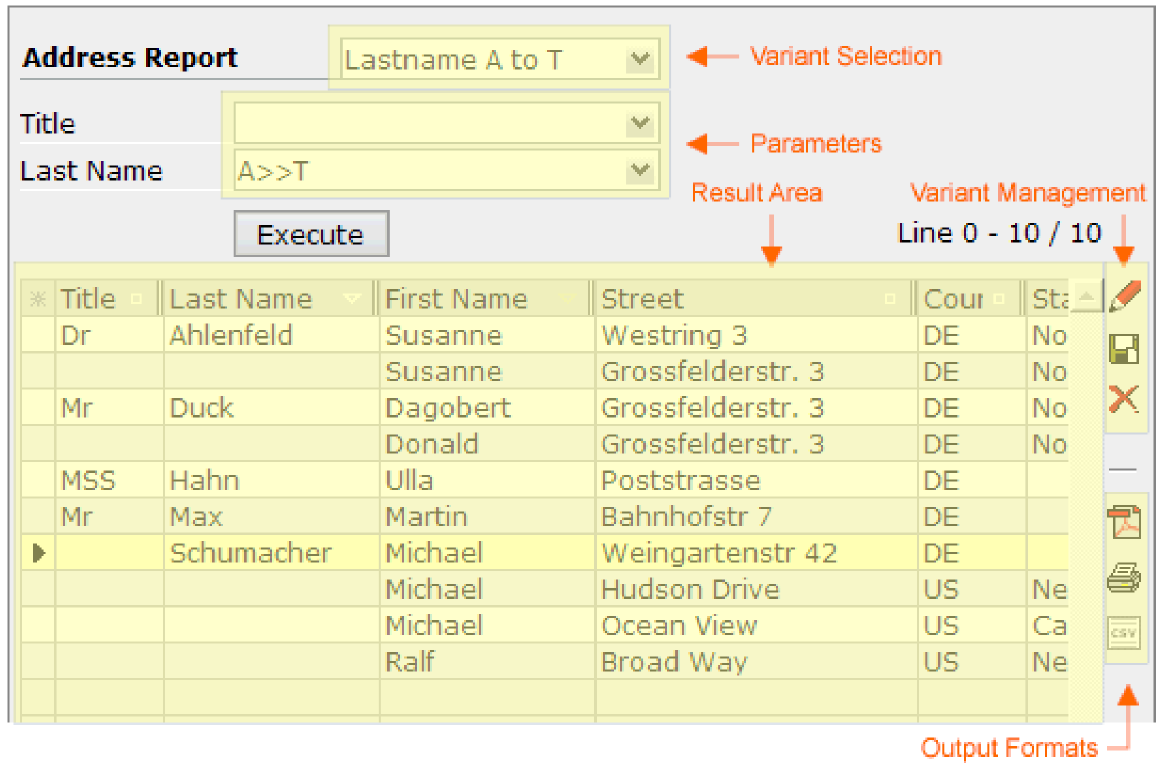
DBQUERY

The DBQUERY control is designed to significantly reduce the effort for developing queries against relational databases. In the control definition, you specify the SELECT string, the filter criteria and the output columns - the rest is done automatically. The sequence, sorting and grouping of the output grid can be defined in a very flexible way. You even can store so-called variants: if you have certain filter criteria you always want to use in a query, then you can store them under a name in order to have quick access to often-used queries. The following typical aspects of a query are covered:

- Filter criteria (ad hoc input, saved in query variants).
- Result output (server-side scrolling, context menu).
- Report variants (column visibility and column order, sorting, grouping, etc.).
- (Default) PDF generation.

This chapter covers the following topics:

Example



Address Report Lastname A to T

Title:

Last Name: A>>T

Line 0 - 10 / 10

Title	Last Name	First Name	Street	Cour	Sta
Dr	Ahlenfeld	Susanne	Westring 3	DE	No
		Susanne	Grossfelderstr. 3	DE	No
Mr	Duck	Dagobert	Grossfelderstr. 3	DE	No
		Donald	Grossfelderstr. 3	DE	No
MSS	Hahn	Ulla	Poststrasse	DE	
Mr	Max	Martin	Bahnhofstr 7	DE	
	Schumacher	Michael	Weingartenstr 42	DE	
		Michael	Hudson Drive	US	Ne
		Michael	Ocean View	US	Ca
		Ralf	Broad Way	US	Ne

There is a DBQUERY control with its inner components. Look at the corresponding layout definition:

```
<dbquery valueprop="dBQueryInfo" query="SELECT * FROM ADDRESS AS A, BUSINESSPARTNER AS B
         WHERE A.BUSINESSPARTNERID = B.ID" datasource="addressdb" title="Adress Report"
         rowareaname="Report Demo" executebuttonname="Execute" maxrows="200"
         image="images/addresses.gif" height="100%" rowcount="40" titletext="Address Report">
    <dbfilter labelname="Title" labelwidth="150" querycolumn="title" valuehelptable="title"
              valuehelpcolumn="id" fieldwidth="200" hideout="true">
    </dbfilter>
    <dbfilter labelname="First Name" labelwidth="150" querycolumn="firstname" fieldwidth="200">
    </dbfilter>
    <dbfilter labelname="Lat Name" labelwidth="150" querycolumn="lastname" fieldwidth="200">
    </dbfilter>
    <dbcolumn name="Title" column="TITLE" width="50" widthpdf="2cm" groupby="true">
    </dbcolumn>
    <dbcolumn name="Last Name" column="LASTNAME" width="100" widthpdf="3cm" sortorder="1"
              sortascending="true" groupby="true">
    </dbcolumn>
    <dbcolumn name="First Name" column="FIRSTNAME" width="100" widthpdf="3cm" sortorder="2"
              sortascending="true">
    </dbcolumn>
    <dbcolumn name="Street" column="STREET" width="150" widthpdf="5cm">
    </dbcolumn>
    <dbcolumn name="Country" column="COUNTRY" width="50" widthpdf="2cm">
    </dbcolumn>
    <dbcolumn name="State" column="STATE" width="100%" widthpdf="5cm">
    </dbcolumn>
</dbquery>
```

Look at the following items:

- There is a DBQUERY definition with the property `valueprop "dBQueryInfo"`. The property `query` contains the SELECT string.

- There are DBFILTER definitions for each filter criterion.
- There are DBCOLUMN definitions for each grid column of the result area.

The adapter code is the following:

```
// This class is a generated one.

import java.sql.Connection;
import java.sql.DriverManager;

import com.softwareag.cis.server.Adapter;
import com.softwareag.cis.server.IDynamicAccess;
import com.softwareag.cis.server.util.DBQUERYDataObject;
import com.softwareag.cis.server.util.DBQUERYInfo;
import com.softwareag.cis.server.util.DBTEXTGRIDCollection;
import com.softwareag.cis.server.util.DBTEXTGRIDLine;
import com.softwareag.cis.server.util.DelegateError;
import com.softwareag.cis.server.util.IDBQUERYConnectionProvider;
import com.softwareag.cis.server.util.IDBQUERYContextMenuRequestListener;
import com.softwareag.cis.server.util.IDBQUERYGeneratePDFRequestListener;
import com.softwareag.cis.server.util.IDBQUERYOptimizer;
import com.softwareag.cis.server.util.MENUNODEInfo;
import com.softwareag.cis.server.util.TREECollection;
import com.softwareag.cis.server.util.ValidValueLine;

public class DBQUERYAdapter
    extends Adapter
    implements IDBDemoAdapter
{
    // ------------------------------------------------------------------------
    // inner classes
    // ------------------------------------------------------------------------

    /** class used for a simple connection management. */
    public class ConnectionProvider
        implements IDBQUERYConnectionProvider
    {
        public Connection getDBConnection(String datasource)
       {
           try
           {
              Class.forName("org.hsqldb.jdbcDriver");
              return DriverManager.getConnection("jdbc:hsqldb:hsql://localhost", "sa", "");
           }
           catch (Exception exc)
           {
           throw new DelegateError(exc);
           }
       }
    }
    /** class used for SQL optimization. */
    public class SQLOptimizer
    implements IDBQUERYOptimizer
    {
        public String optimizeQuery(String query, String[] gridColumns)
        {
            // do checks + optimization here
            return query;
        }
    }
    /** class used for pop-up menu. */
    public class MyMenuNodeInfo
    extends MENUNODEInfo
    {
        IDynamicAccess m_row; // = row for that the context menu is requested
        public MyMenuNodeInfo(IDynamicAccess row, String text, String image)
        {
            super(text, image);
            m_row = row;
        }
        public void reactOnSelect()
        {
            outputMessage("S", getText() + " selected");
        }
    }
    /** class used to create the context menu within the result grid. */
    public class ResultAreaContextMenu
    implements IDBQUERYContextMenuRequestListener
    {
        public void reactOnContextMenuRequestFor(IDynamicAccess row)
        {
            // opens a pop-up menu with two entries
           TREECollection cm = new TREECollection();
           cm.addTopNode(new MyMenuNodeInfo(row, "Edit", "images/edit.gif"),true);
           cm.addTopNode(new MyMenuNodeInfo(row,"Remove", "images/remove.gif"),true);
           showPopupMenu(cm);
        }
    }
    /** class used for PDF conversion. */
    public class GeneratePDFRequestListener
    implements IDBQUERYGeneratePDFRequestListener
    {
        public void generatePDFAndDisplayDocument(DBQUERYDataObject dataObject)
        {
        // create PDF document here
        }
    }

      DBQUERYInfo m_dBQueryInfo = new DBQUERYInfo(this,new ConnectionProvider(),
                                  new ResultAreaContextMenu(),
                                  new SQLOptimizer(),
                                  new GeneratePDFRequestListener());
      public DBQUERYInfo getDBQueryInfo() { return m_dBQueryInfo; }
}
```

Programming the DBQUERY is quite simple. Define an instance of the class `DBQUERYInfo`. This instance is referenced by the `valueprop` definition inside the DBQUERY tag. The `DBQUERYInfo` offers a set of constructors.

Mandatory Parameters

Same to all constructors are the (two) mandatory parameters. First you have to pass the adapter object that defines the `DBQUERYInfo` instance. This object is used, for example, to open a pop-up inside the DBQUERY control. With the second, you have to pass an implementation of interface `IDBQUERYConnectionProvider`. As the `DBQUERYInfo` class does not open a database connection on its own, it uses this object to obtain a connection. The connection is only used to read data from the database. There are no updates (insert/update/delete) done with this connection. Internally, the provided connection is buffered and used each time the query is executed. As the `DBQUERYInfo` does not open the connection, it does not care about closing the connection.

Optional Parameters

The constructors vary in the list of their optional parameters. By construction, you may pass an implementation of interface `IDBQUERYContextMenuRequestListener`. This object is called (with method `reactOnContextMenuRequestFor`) if the user clicks into a line inside of the report result grid with right mouse button. See the Javadoc documentation of interface `IDBQUERYContextMenuRequestListener` for details.

If you want to check/optimize the SQL statement prior its execution, you may pass an implementation of interface `IDBQUERYOptimizer`. The object is called with method `optimizeQuery` each time the report is executed. See the Javadoc documentation of interface `IDBQUERYOptimizer` for details.

If you do not want to use the default PDF conversion of the DBQUERY control, you may pass an implementation of interface `IDBQUERYGeneratePDFRequestListener`. This object is called (with method `generateAndDisplayPDF`) if the user clicks the "PDF" icon within the DBQUERY control. See the Javadoc documentation of interface `IDBQUERYGeneratePDFRequestListener` for details.

DBQUERY Properties

Java Binding			
valueprop	Property VALUEPROP points to a property of type DBQUERYInfo (package com.softwareag.cis.server.util). This class encapsulates the reports execution, the variant management and the PDF and CSV output.	Obligatory	
directselectmethod	The DIRECTSELECTMETHOD property is used to point to a method of your adapter class, which is called when a selection event occurs within the result grid.	Optional	
directselectevent	The DIRECTSELECTEVENT property is used to define whether the direct select method is called by a single or a double click. Typically you use a single click ("onclick") if you want to select something in the grid and to display simultaneously details of what was selected in the same page. Use a double click ("ondblclick") to navigate to the next page.	Optional	ondblclick onclick
DB Binding			
datasource	Logical identifier of the database on that the report is executed. This value is passed in the method "IDBQUERYConnectionProvider. GetDBConnection".	Obligatory	

query	SQL statement with a complete SELECT and FROM clause and with an optional WHERE clause. With the SELECT clause you define the result set of the report (each DBCOLUMN control refers to one element/column name of this result set via the property COLUMN). Prior the reports execution the values of the DBFILTER controls are added to this query.	Obligatory	
maxrows	Specifies the maximum number of rows fetched from database. The value "0" represents unlimited. Default is "200".	Optional	20 50 100 200 500 0
executeonload	Flag which indicates if the report is to be executed on page load. Default is "false".	Optional	true false
filterlinkoperator	The values of the DBFILTER controls are added dynamically to QUERY prior the reports execution. With this property you can specify the operator to be used to add the DBFILTER values. Default is "AND".	Optional	and or
Height			
height	The height of the DBQUERY control in pixels or as percentage value.	Obligatory	100 150 200 250 300 250 400 50% 100%
Title			
title	Name of the database report.	Optional	
titletextid	Text ID (report title) for the multi language management.	Optional	

titlelabelwidth	Width of the title in pixels or as percentage value.	Optional	100 120 140 160 180 200 50% 100%
titlestyle	Direct manipulation of title style.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Row Area			
rowareaname	Name of the surrounding row area.	Optional	
rowareatextid	Text ID (row area) for the multi language management.	Optional	
foldable	The surrounding row area can be shrinked by clicking on its title. This standard behaviour can be disabled by setting FOLDABLE to "false".	Optional	true false
rowareastyle	Inline style for the surrounding row area.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
image	URL of the image that is shown at the right hand of the filters. The URL can be relative or absolute.	Optional	
executebuttonname	Name that is displayed on the execute button.	Optional	
executebuttontextid	Text ID (execute button) for the multi language management.	Optional	
outputmesstostatusbar	Flag that indicates if messages that are generated by the report (DBQUERYInfo) are displayed within the status bar (default) or inside the DBQUERY control above the result grid ("false").	Optional	true false

showprintversion	If switched to "true" then a small print icon will appear right from the grid. The print icon opens up a modal popup from which the HTML produced inside the report can be directly sent to the printer. Pay attention: if specifying "true" then the adapter property holding the REPORTInfo object must create the REPORTInfo instance with passing "this" in the constructor.	Optional	true false
showpdf	If set to "true" then a PDF icon is rendered in the right top corner of the control. When the user clicks on the icon then the report is automatically rendered as pdf - and the result will show up in a popup window. Pay attention: if setting this property to "true" then you also have to choose a special constructor when creating the REPORTInfo instance on server side, in which the instance of the model is passed as argument.	Optional	true false
personalizable	If defined to "false" then no re-arranging of columns is offered to the user. Default is "true". This means: if using COLUMN controls inside the grid definition then the user can re-arrange the sequence of columns by dragging and dropping them within the top title row.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	
Result Grid			
rowcount	The property ROWCOUNT defines the number of rows that are fetched from the server.	Optional	1 2 3 int-value
withselectioncolumn	Flag which indicates if the result grid does have a selection column. Default is "true".	Optional	true false
withtitlerow	Flag indicates if the result grid does have a title row (default) or not ("false").	Optional	true false

hscroll	Indicator if the result grid shows a horizontal scroll bar.	Optional	true false
fixlayout	When switching the FIXLAYOUT property to value "true" then internally the result grid is arranged in a way that the area always determines its size out of the width specification of the DBCOLUMN controls. The browser does not look into the column contents in order to try to optimise the size of the area - but always follows the width and height that you define.	Optional	true false
backgroundstyle	Direct style manipulation of the table style which surrounds the table cells inside the result grid.	Optional	

DBFILTER Properties

The DBFILTER tag is the typical tag that is placed inside a DBQUERY definition. The DBFILTER defines one filter criterion with its binding to a table column (property `querycolumn`). The parameter values are added dynamically to the WHERE clause of the SQL query prior to report execution. The SQL statement is defined within the `query` property of the DBQUERY control.

DB Binding			
querycolumn	Name of the column within the reports query the DBFILTER control is bound to. With this name user input will be added to the SQL statement on report execution. The SQL statement is defined in the property QUERY of the DBQUERY control. Example: If you define the SQL statement like "SELECT STREET, ZIPCODE, TOWN FROM ADDRESS" in property QUERY of the DBQUERY control you can specify any column name of table "ADDRESS" here.	Obligatory	
valuehelptable	You may have a table in that the valid values for this filter are kept. In that case you can provide for a filter value help by using the properties VALUEHELPTABLE and VALUEHELPCOLUMN. Input the name of the table here.	Optional	
valuehelpcolumn	You may have a table in that the valid values for this filter are kept. In that case you can provide for a filter value help by using the properties VALUEHELPTABLE and VALUEHELPCOLUMN. Input the name of the column here.	Optional	
usequeryforvaluehelp	Flag that indicates that the value help is coming from the query result set.	Optional	true false

columnalias	Alias of the database column	Optional	
Filter Name			
labelname	Name of the filter.	Optional	
labeltextid	Text ID (filter name) for the multi language management.	Optional	
labelwidth	Width of the filter name in pixels or as percentage value.	Optional	100 120 140 160 180 200 50% 100%
Filter Input			
fieldwidth	Width of the filter input field in pixels or as percentage value.	Optional	100 120 140 160 180 200 50% 100%
fieldlength	Width of the filter input field in amount of characters. FIELDWIDTH and FIELDLENGTH should not be used together.	Optional	5 10 15 20 int-value

fielddatatype	Specifies the data type of the filter. As a consequence the fields inside the grid of the value help popup are checking the data during input (e.g. if the DATATYPE is "int", it is not allowed to enter alphabetic characters) and adds a logic to transfer the data into various output formats (e.g. if the DATATYPE is "date", the date is formatted into the right date format). In addition these fields have a standard "value help" popup dialog for some data types (e.g. if the DATATYPE is "date" then automatically a date input dialog pops up if invoking "value help").	Optional	int float date
fielduppercase	Flag which indicates if the alphabetic characters input should be converted to upper case if necessary. Default is "false".	Optional	true false
fieldstyle	Explicit style information for the input field. Example: if you want the text to be right aligned, define "text-align: right".	Optional	
fieldhelpid	Name that is used to identify the online help page to be opened if the user presses the F1-key inside the FIELD control. Please refer to chapter "Online Help Management" for details.	Optional	
Comment			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	

DBCOLUMN Properties

The DBCOLUMN tag is the typical tag that is placed inside a DBQUERY definition. The DBCOLUMN definition defines one column within the result grid. It is bound to a name (column name or column alias) in the result set of the report. This result set is defined by the `query` property inside the DBQUERY definition.

DB Binding			
column	Name of the table column the DBCOLUMN control refers to. Please pay attention: If you use column aliases in the SQL query (refer to property QUERY of the DBQUERY control) you must specify the column alias here. Example: If you use the query "Select CUSTNAME as CN FROM CUSTOMER" you have to use column alias "CN" here. "	Obligatory	
Appearance			
name	Name of the title cell of the grid column.	Optional	
textid	Text ID (name) for the multi language management.	Optional	

width	Width of the column in pixels or as percentage value.	Optional	100 120 140 160 180 200 50% 100%
widthpdf	Width in centimetres the column should occupy inside the PDF document.	Optional	0.5 0.75 1 2 5 10
align	Horizontal alignment of the text within the column. Default is "left", other values are "center" or "right".	Optional	left center right
straighttext	Flag which indicates whether the text displayed inside the column if formatted as HTML text or as straight text. Default is "false".	Optional	true false
convertspaces	Flag which indicates if spaces inside the text of a cell should be converted in "non-breakable-spaces". In general HTML converts several appearances of space-characters ("blanks") into one space-character. If you set CONVERTSPACES to "true", this default behaviour is switched off.	Optional	true false
cuttextline	If a text does not fit into a cell then it is cut off. If you set CUTTEXTLINE to "false", it will be broken - following HTML rules for breaking text. Therefore the cell will contain more than one text line.	Optional	true false

datatype	Data type of the content of the column. Therefore certain rendering rules are applied (e.g. in case of "date", a YYYYMMDD date is converted into a proper date format).	Optional	date float int long time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time ------------------------ N n.n P n.n string n xs:byte xs:short
Comment			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	

DBPARAMSINGLEVALUE Properties

The DBPARAMSINGLEVALUE tag is the typical tag that is placed inside a DBQUERY definition. The DBPARAMSINGLEVALUE defines a filter criterion. In contrast to the DBFILTER tag, it is not visualized within the DBQUERY control and is therefore not accessible/changeable by the user.

Example: a table "ADDRESS" may have a column "COUNTRY". You want to restrict the user to see German addresses only. In this case, use a DBPARAMSINGLEVALUE with the property `querycolumn` set to "COUNTRY", `operator` set to "=" (equal), and `value` set to "DE".

Basic			
querycolumn	Name of the column the DBPARAMSINGLEVALUE control refers to. This name is used to add the parameters value to the SQL statement prior the reports execution. The SQL statement is defined in the property QUERY of the DBQUERY control.	Obligatory	
operator	Name of the operator to use to append the value to WHERE clause of the SQL statement.	Obligatory	= != ~ !~ > >= < <= NULL NOT NULL
value	The parameter value. You can either enter a fixed value or you can specify the name with that a value can be looked up from the session context when executing the report.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	

DBPARAMDOUBLEVALUE Properties

The DBPARAMDOUBLEVALUE tag is very similar to DBPARAMSINGLEVALUE. The only difference is that you can specify operators that work on two operands (e.g. "is between").

Basic			
querycolumn	Name of the column the DBPARAMSINGLEVALUE control refers to. This name is used to add the parameters value to the SQL statement prior the reports execution. The SQL statement is defined in the property QUERY of the DBQUERY control. Example: If you define the SQL statement like "SELECT STREET, ZIPCODE, TOWN FROM ADDRESS" in property QUERY of the DBQUERY control you can specify any column name of table "ADDRESS" here.	Obligatory	
operator	Name of the operator to use to append the value to WHERE clause of the SQL statement. The operator specified here must work on two operands. At the moment "between" and "not between" are supported.	Obligatory	>> !gt;>
value1	The first parameter value. You can either enter a fixed value or you can specify the name with that a value can be looked up from the session context when executing the report. Example: If you enter "$FROMDATE$" here, the DBPARAMSINGLEVALUE will try to lookup a value bound to the name "FROMDATE" from the session context. An exception will be raised if nothing is found.	Obligatory	
value2	The second parameter value.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor’s tree view.	Optional	

Variant Management

The DBQUERY control provides for so-called query variants. Within a variant, you can save e.g. an often-used filter criteria combination. If you do not want to use variants at all, you just set the property `personalizable` to false.

Persistence

The variant data is stored in the local file system.

Key

The key of a report variant is consists of:

- the name of the web application,
- the name of the project that contains the HTML page,
- the name of the HTML page that contains the DBQUERY control,

- the actual user logged on (read from the session context on save),
- the name of the variant (specified in the save pop-up).

As the name of the page (that embeds the DBQUERY control) is part of the variant key, you can have multiple pages within one Application Designer project - each with its own variant set. But: as a consequence, variants do not work inside the Layout Painter. Reason: if you preview the XML layout definition, the editor first removes the temporary HTML page (name starts with "`ZZZZZZZZ`") of a former preview. In a second step, it creates a new page - with a different name (the name contains a timestamp). This means: inside the Layout Painter, you never view the same HTML page twice. If you work with the published HTML page, variants will work properly.

Create/Change/Remove

To create a variant, you just save your current input (filter criteria or settings inside the variant properties pop-up). To save a variant, choose the save icon to the right of the result grid. To change an existing variant, open the variant (with input field to the right of the report title label), apply your changes (either by changing the filter criteria or within the variant properties pop-up - icon above the save icon) and save them. To remove a variant, choose the remove icon (icon below save icon), select one or more variants and choose ok.

PDF Generation

The DBQUERY control provides for a generation of a PDF document. The generation is invoked when choosing the "PDF" icon to the right of the result grid. The document is displayed inside a pop-up. It contains the following data:

- Report title (header line).
- Timestamp of the reports execution (footer).
- Table with used filter criteria (body).
- Table with result of the report (body).

How to do the PDF generation on your own?

The `DBQUERYInfo` class provides for a constructor where you can pass an implementation of interface `IDBQUERYGeneratePDFRequestListener`. This implementation is called with the method `generatePDFAndDisplayDocument` if the "PDF" icon is chosen. The current data of the DBQUERY is passed within that call. For details, see the JavaDoc documentation of interface `IDBQUERYGeneratePDFRequestListener` and class `DBQUERYInfo` (both from package `com.softwareag.cis.server.util`).