servBIRD 3.18

Handbuch für Berichtsentwickler

Dokumentversion vom 20. 11. 2023

Inhalt

1 Über diese Dokumentation

1.1 Änderungshistorie

Datum	Version	Änderungsbeschreibung	Änderung von
19.08.2016	v0.1	Initialer Stand der servBIRD 3.6 Dokumentation	Team TRADUI
24.02.2017	v0.2	Beschreibung neuer Bereiche	Team TRADUI
11.12.2017	v0.3	Stand der servBIRD 3.8 Dokumentation	Team TRADUI
16.01.2019	v0.4	Stand der servBIRD 3.10 Dokumentation	Team TRADUI
12.03.2019	v0.5	• Kleinere Änderung an der Struktur • Übersetzungen nachgezogen	Busch, F. (TRADUI)
31.07.2019	v0.6	Stand der servBIRD 3.12 Dokumentation	Team TRADUI
24.06.2020	v0.7	Initialer Stand der servBIRD 3.14 Dokumentation	Team TRADUI
07.12.2020	v0.8	Initialer Stand der servBIRD 3.16 Dokumentation	Team TRADUI
10.06.2021	v0.9	Initialer Stand der servBIRD 3.18 Dokumentation	Team TRADUI

1.2 Zielgruppe und Voraussetzungen

Die vollständige **servBIRD** Dokumentation ist in vier Handbücher aufgeteilt:

1. Das Benutzerhandbuch ...ist für den Endanwender gedacht.
2. Das Administrationshandbuch ...befasst sich mit der Installation, der Einrichtung, der Wartung und dem Betrieb des servBIRD. Dieses Handbuch ist mit seinen technischen Details für den Administrator bzw. erfahrenen Anwender gedacht.
3. Das Handbuch für Berichtsentwickler ...beleuchtet spezielle Features und Möglichkeiten von servBIRD, die im BIRT-Bericht eingestellt werden müssen.
4. Das Handbuch für Anwendungsentwickler ...ist primär für Anwendungsentwickler gedacht, die den servBIRD erweitern oder in ihre Anwendung integrieren möchten.

1.3 Hinweise und Konventionen

Tipp

Dieser Block hebt Informationen hervor, die bspw. zeit- oder ressourcenschonende Verfahren oder Best Practices erläutern.

Information

In diesem Block werden Informationen von besonderer Bedeutung oder besonderem Interesse hervorgehoben.

Hinweis

Dieser Block weist auf zu beachtende Informationen hin oder warnt vor Stolperfallen. Bitte lesen Sie diese Blöcke aufmerksam!

Achtung

Diese Hinweise deuten auf Informationen hin, die bei Missachtung oder falscher Nutzung zu Fehlfunktion der Software oder Löschung von relevanten Daten führen können.

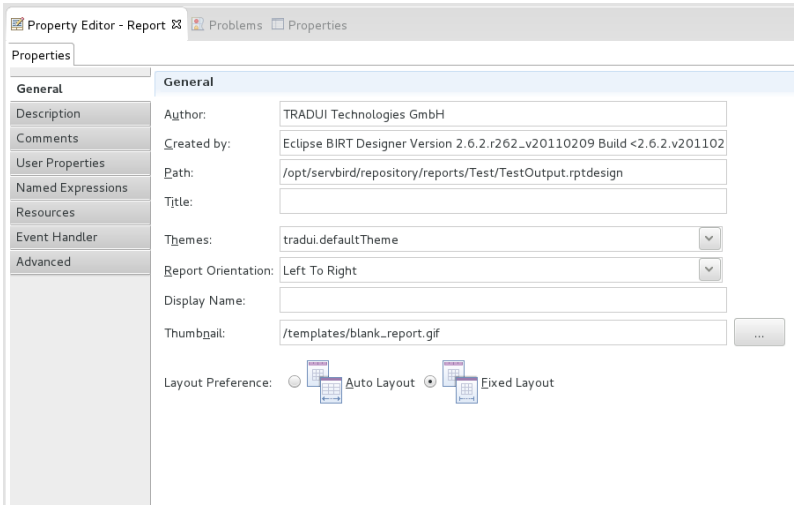
Beispiel

In diesem Block wird ein Beispiel zum jeweiligen Kontext aufgeführt.

2 BIRT Report Properties

2.1 Standard Properties

An fast allen Objekten in BIRT können vordefinierte Eigenschaften gesetzt werden. Beispielsweise können am Reportobjekt Titel und Autor des Reports definiert werden.



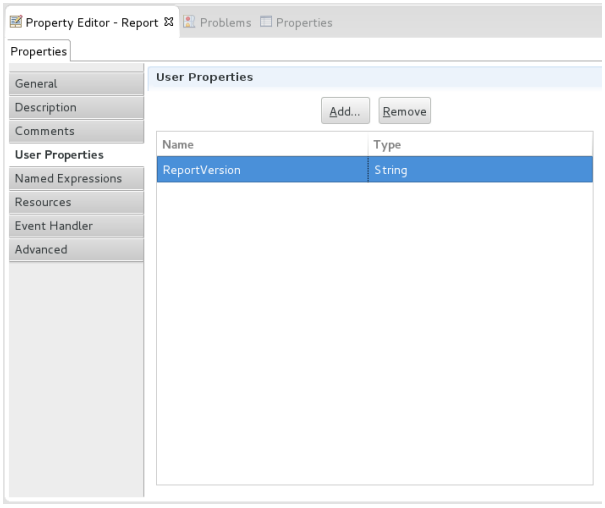
2.2 User Properties

Zusätzlich können an fast allen Objekten in BIRT benutzerdefinierte Eigenschaften gesetzt werden. Mittels solcher User Properties kann das Aussehen und Verhalten von Reports und deren Parametern innerhalb von **servBIRD** beeinflusst werden. User Properties werden im Eclipse Report Designer in zwei Schitten definiert: Zuerst wird ein User Property hinzugefügt und danach wird die Eigenschaft mit einem Wert versehen.

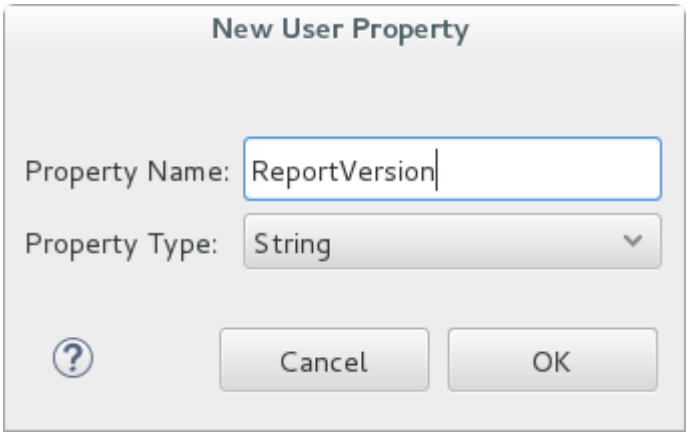
> **servBIRD** wertet ausschließlich User Properties der Objekte "Report" und "Parameter" aus. An anderen Objekten definierte User Properties werden von **servBIRD** ignoriert.

2.2.1 Hinzufügen von User Properties

Wählen Sie im View **Outline** das gewünschte Objekt und öffnen Sie danach im View **Property Editor** den Tab **User Properties**. Hier können Sie mittels des Buttons **Add** ein User Property hinzufügen.

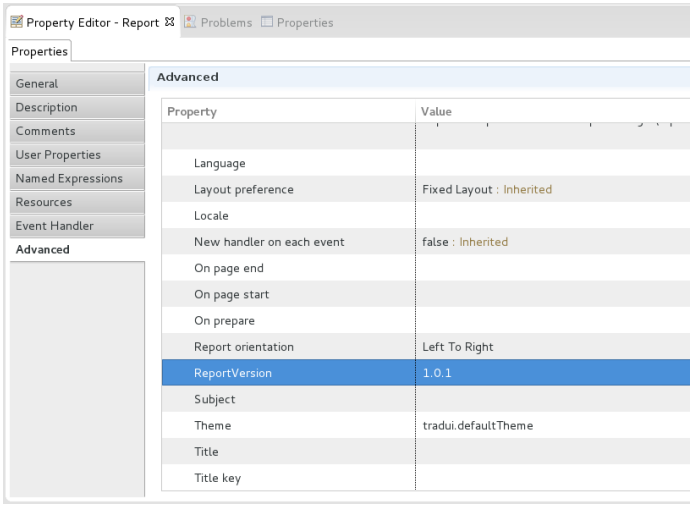


Im erscheinenden Dialogfenster werden Sie aufgefordert den Namen und den Datentyp des User Propertys festzulegen. Mit einem Klick auf den Button **OK** wird das User Property schießlich angelegt.



2.2.2 Setzen des Wertes für ein User Property

Wählen Sie im View **Property Editor** den Tab **Advanced**. In der alphabetisch sortierten Liste erscheinen alle definierten User Properties. Klicken Sie in die Value-Spalte der entsprechenden Zeile, um einen Wert für das User Property zu setzen.



2.3 Report Properties

Hier werden allgemeine Einstellungen am ReportDesign beschrieben, die von **servBIRD** ausgelesen und gegebenenfalls im Portal anzeigt werden.

2.3.1 Auf einen Blick: Übersicht über die Standard Properties und User Properties an Report Designs

Standard Properties				
Beschreibung	**Name**	**Typ**	**Beispielwert**	**Zu finden unter**
Titel des Reports	Title	String	Personalstatistik	Properties → General → Title
Autor des Reports	Author	String	Max Mustermann	Properties → General → Author
Beschriebung des Reports	Description	String	Dieser Report gibt Aufschluss über ...	Properties → Description → Description
User Properties				

Beschreibung	Name	Typ	Beispielwert	Zu finden unter
Version des Reports	ReportVersion	String	1.0	Properties → User Properties; Properties → Advanced
Unterstützte Ausgabeformate	SupportedFormats	String	PDF,DOC,XLS_SPUDSOFT,XLSX	Properties → User Properties; Properties → Advanced
Vorausgewähltes Ausgabeformat	StandardFormat	String	PDF	Properties → User Properties; Properties → Advanced
Generierungsverfahren	GenerationMethod	String	TwoPass	Properties → User Properties; Properties → Advanced
Maximale Aufbewahrungsdauer der Ausgabedokumente	DocumentMaxAge	Integer	180	Properties → User Properties; Properties → Advanced
Report als Master Report verarbeiten	MasterReport	Boolean	true	Properties → User Properties; Properties → Advanced
LdapManager Plugin verwenden	UseLdapPlugin	Boolean	true	Properties → User Properties; Properties → Advanced
HTTP Endpunkte verwenden	UseHttpEndpoint	Boolean	true	Properties → User Properties; Properties → Advanced

2.3.2 Eigenschaften

Titel des Reports

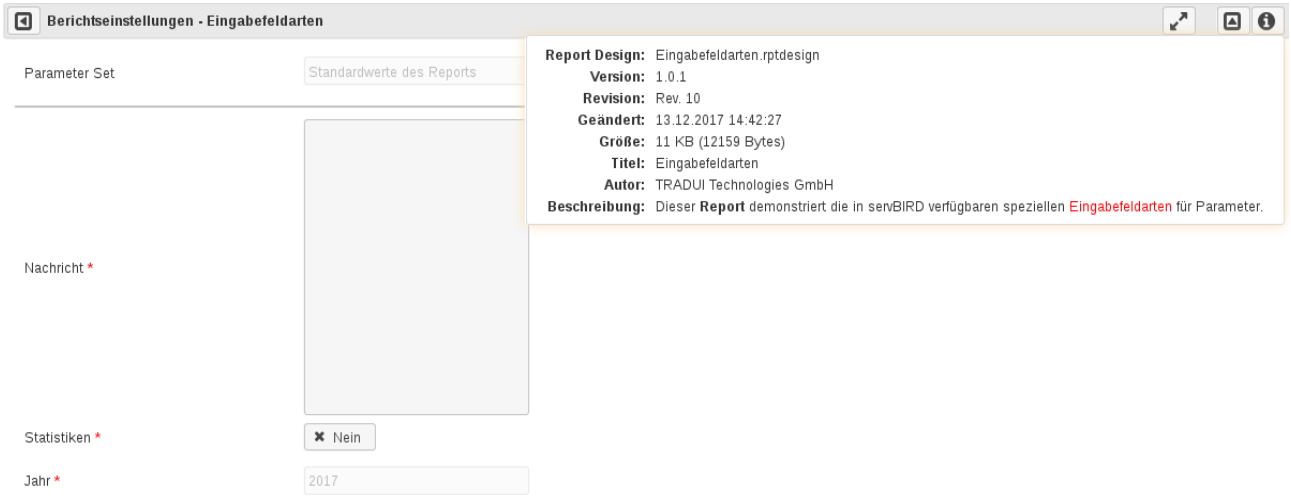
Der Titel des Reports wird im Eclipse Report Designer am Report unter **Properties** → **General** → **Title** eingestellt.

Sobald ein Titel definiert ist, zeigt **servBIRD** den Reporttitel im Kategoriebrowser an.

Autor des Reports

Der Autor des Reports wird im Eclipse Report Designer am Report unter **Properties** → **General** → **Author** eingestellt.

Die Information wird innerhalb von **servBIRD** in den Informationen zum Report angezeigt.



Beschreibung des Reports

Die Beschreibung des Reports wird im Eclipse Report Designer am Report unter **Properties** → **Description** → **Description** hinterlegt. Formatierungen mittels HTML sind möglich.

Die Information wird innerhalb von **servBIRD** in den Informationen zum Report angezeigt.

Version des Reports

Für Reports kann eine Versionsinformation hinterlegt werden. Dies erleichtert die Identifikation unterschiedlicher Reportversionen.

Zum Hinzufügen der Versionsinformation muss am Report das User Property ReportVersion mit dem gewünschten Wert definiert werden.

- User Property ReportVersion vom Typ String am Report definieren
- Als Wert für das Property wird unter unter **Advanced** ein gewünschter Wert eingetragen

Die Information wird innerhalb von **servBIRD** in den Informationen zum Report angezeigt

2.3.3 Verhalten

Unterstützte Ausgabeformate

Für Reports kann eine Liste von möglichen Ausgabeformaten festgelegt werden.

Zur Definition der erlaubten Ausgabeformate muss am Report das User Property SupportedFormats mit einer kommasepartierten Liste von Ausgabeformaten als Wert definiert werden. Mögliche Ausgabeformate sind: HTML, PDF, DOC, DOCX, EXCEL, XLS_SPUDSOFT, XLSX, ODS, ODT, ODP, PPT, PPTX, NATIVE.

- User Property SupportedFormats vom Typ String am Report definieren
- Als Wert für das Property wird unter unter **Advanced** eine kommasepartierte Liste von Ausgabeformaten eingetragen

Haben Sie die möglichen Ausgabeformate mittels dieses User Propertys gesetzt, so bietet **servBIRD** zur Ausführung des Reports nur noch die erlaubten Formate an.

Vorausgewähltes Ausgabeformat

Für Reports kann ein bestimmtes Ausgabeformat als Standard festgelegt werden.

Zur Definition Standardausgabeformats muss am Report das User Property StandardFormat mit dem gewünschten Ausgabeformat als Wert definiert werden. Mögliche Ausgabeformate sind: HTML, PDF, DOC, DOCX, EXCEL, XLS_SPUDSOFT, XLSX, ODS, ODT, ODP, PPT, PPTX, NATIVE.

- User Property StandardFormat vom Typ String am Report definieren
- Als Wert für das Property wird unter unter **Advanced** das gewünschte Ausgabeformat eingetragen

Haben Sie das Standardausgabeformat mittels dieses User Propertys gesetzt, so wird in **servBIRD** zur Ausführung des Reports das spezifizierte Ausgabeformat vorausgewählt sein.

Generierungsverfahren

Für Reports, die über mooBIRD angestoßen werden (WEBSERVICE Jobs), kann das Generierungsverfahren umgestellt werden. Soll der Report durch BIRT in getrennten Run-Tasks und Render-Tasks (anstatt im gemeinsamen RunAndRender-Task) generiert werden, so kann hier der Wert "TwoPass" gesetzt werden.

- User Property "GenerationMethod" vom Typ String am Report definieren
- Als Wert für das Property wird unter unter **Advanced** den Wert "TwoPass" eingetragen

Wird der Report auf diese Weise generiert, so stehen zusätzliche Informationen, wie beispielsweise die Anzahl der generierten Seiten, am Report Job zur Verfügung.

Maximale Aufbewahrungsdauer der Ausgabedokumente

Für die Ausgabedokumente des Reports kann mittels dieses User Properties die maximale Aufbewahrungsdauer in Minuten festgelegt werden.

Zur Definition der Aufbewahrungsdauer muss am Report das User Property DocumentMaxAge hinzugefügt und als Wert die gewünschte Aufbewahrungsdauer in Minuten definiert werden.

- User Property DocumentMaxAge vom Typ Integer am Report definieren
- Als Wert für das Property wird unter unter **Advanced** die gewünschte Dauer in Minuten definiert

Haben Sie auf diesem Wege die maximale Aufbewahrungsdauer mittels dieses User Propertys gesetzt, so wird **servBIRD** die von diesem Report erzeugten Dokumente nach erreichen der Maximaldauer automatisch löschen.

Report als Master Report verarbeiten

Dieses Property weist **servBIRD** an, die Jobs dieses Reports als Ausführungstyp "MASTER" auszuführen. Dieser JobTyp besitzt eine eigenständige Report Warteschlange, deren Sloteinstellungen in den Konfigurationseinstellungen separat konfiguriert werden können. Weiterhin wird **servBIRD** diese Jobs innnerhalb der Jobliste als Mappe anzeigen und alle Sub-Report-Jobs innerhalb dieser Mappe bündeln. Auf diese Weise können aus dem Master Report heraus, mittels **TRADUI mooBIRD** gestartete, weitere Report Jobs in geordneter Form im Portal dargestellt werden. Dies ist vor allem bei Verwendung des ServBIRD Manager Plugins und dem darin enthaltenen ControllerManager wünschenswert, wenn auf diese Weise beispielsweise mit dem PdfManager Plugin aus Einzelreports bestehende zusammengesetzte PDF Dokumente erstellt werden.

Zur Definition des Reports als Master Report muss am Report das User Property MasterReport hinzugefügt und als Wert "true" definiert werden.

- User Property MasterReport vom Typ Boolean am Report definieren
- Als Wert für das Property wird unter unter **Advanced** der Wert "true" definiert

Haben Sie auf diesem Wege den Report als Master Report gekennzeichnet, so wird **servBIRD** diesen Report als Job Typ "MASTER" ausführen und die von diesem Report gestarteten Sub-Report-Jobs im Portal zusammenfassen.

LdapManager Plugin verwenden

Dieses Property veranlasst **servBIRD** dazu die vorhandenen (aktiven) Security Controller Konfigurationen zur Verwendung mit dem LdapManager Plugin für den Report bei dessen Ausführung bereitzustellen. Mit dem LdapManager Plugin ist es möglich Verzeichnisdienstabfragen auszuführen um beispielsweise nach Benutzern oder Gruppen zu suchen.

Damit das LdapManager Plugin im Report mit den Security Controller Konfigurationen in **servBIRD** verwendet werden kann, muss das User Property UseLdapPlugin hinzugefügt und als Wert "true" definiert werden.

- User Property UseLdapPlugin vom Typ Boolean am Report definieren
- Als Wert für das Property wird unter unter **Advanced** der Wert "true" definiert

Wenn Sie das User Property am Report definiert haben, so kann bei der Ausführung des Reports auf **servBIRD** das LdapManager Plugin verwendet werden.

HTTP Endpunkte verwenden

Dieses Property veranlasst **servBIRD** dazu die vorhandenen (aktiven) HTTP Endpunkt Konfigurationen zur Verwendung mit dem servBIRD Plugin für den Report bei dessen Ausführung bereitzustellen. Auf diese Weise ist es möglich die Informationen der in **servBIRD** definierten HTTP Endpunkte im Report zu verwenden.

Damit die HTTP Endpunktionformationen von **servBIRD** an den Report weitergegeben werden, muss das User Property UseHttpEndpoint hinzugefügt und als Wert "true" definiert werden.

- User Property UseHttpEndpoint vom Typ Boolean am Report definieren
- Als Wert für das Property wird unter unter **Advanced** der Wert "true" definiert

Wenn Sie das User Property am Report definiert haben, so kann bei der Ausführung des Reports auf **servBIRD** das servBIRD Plugin die HTTP Endpunktinformationen auslesen.

2.4 Parameter Properties

Sie können das Verhalten und Aussehen der Parameter im Portal bei der Berichtsentwicklung steuern.

Hinweis

Bitte achten Sie bei der Verwendung von User Properties und deren Werten auf korrekte Groß-/Kleinschreibung.

2.4.1 Auf einen Blick: Übersicht über die User Properties an Berichtsparametern

Eingabefeldarten					
Beschreibung	**Name**	**Typ**	**Wert**		**Gültig für**
Autovervollständigung	DisplayType	String	AutoComplete		Alle Parameter
Boolean Button	DisplayType	String	BooleanButton		Alle Parameter
Textbereich	DisplayType	String	TextArea		Alle Parameter
HTML Editor	DisplayType	String	HTMLEditor		Alle Parameter
Text Editor	DisplayType	String	TextEditor		Alle Parameter
Code Editor	DisplayType	String	CodeEditor		Alle Parameter
Datei Upload	DisplayType	String	FileUpload		Alle Parameter
PickList	DisplayType	String	PickList		Alle Parameter
RadioButton	DisplayType	String	RadioButton		Alle Parameter
Textausgabe	DisplayType	String	TextOutput		Alle Parameter
Erscheinungsbild					
Beschreibung	**Name**	**Typ**	**Wert**	**Einheit**	**Gültig für**
Parametereingabefeld: Höhe	ParamHeight	Integer	42	Pixel	HTML Editor, Text Editor
Parametereingabefeld: Breite	ParamWidth	Integer	42	Pixel	Textbox, Textbereich, HTML Editor, Text Editor, Combobox, Listenparameter, PickList
Parametereingabefeld: Zeilenanzahl	ParamRows	Integer	42	Anzahl Zeilen	Textbereich

Parametereingabefeld: Anzuzeigende Einträge	PortalHeight	Integer	42	Anzahl Einträge	Combobox, Listenparameter, PickList
Spalte mit Namen des Parameters ausblenden	HideParamName	Boolean	true		Alle Parameter
Auswahlbuttons für Multivalue Parameter ausblenden	ListBoxSelectionButtons	Boolean	false		Listenparameter
Icons für Auswahlwerte anzeigen	LabelIcons	Boolean	true		Combobox, Listenparameter, PickList
Verhalten					
Beschreibung	**Name**	**Typ**	**Wert**	**Einheit**	**Gültig für**
Optionaler Parameter	Optional	Boolean	true		Alle Parameter
Unveränderlicher Parameter	ReadOnly	Boolean	true		Alle Parameter
Bedingt unveränderlicher Parameter	ReadOnlyCondition	String	('[paramLand]' == 'Wert' && '[paramStadt]' == 'Wert')		Alle Parameter
Maximale Parameterlänge	ParamMaxLength	String	42	Anzahl Zeichen	Eingabefelder, Textbereich, HTML Editor, Text Editor, Code Editor
Selektive Sichtbarkeit für servBIRD Rollen	PortalParamSecurityRoles	String	role1,role2,role3		Alle Parameter
Filter	ListBoxFilter	Boolean	true		Comboboxen und Listenparameter
Filtermodus	ListBoxFilterMode	String	startsWith || contains || endsWith		Comboboxen und Listenparameter mit aktiviertem Filter
Bedingt sichtbarer Parameter	VisibilityCondition	String	('[paramLand]' == 'Wert' && '[paramStadt]' == 'Wert')		Alle Parameter

Auslösender Parameter	UpdatesParameters	String	parameterName1,parameterName2		Alle Parameter
Null-Checkbox ausblenden	HideNullCheckbox	Boolean	true		Alle Parameter

2.4.2 Eingabefeldarten

Autovervollständigung für Parameter (Autocomplete)

Für Comboboxen bzw. Listboxen kann die Darstellung statt einer Auswahlliste zu einer AutocompleteBox geändert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "AutoComplete" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "AutoComplete" unter Advanced eingetragen werden

Die Autocomplete-Liste wird erst angezeigt, sobald ein Buchstabe ins Eingabefeld eingegeben wurde.

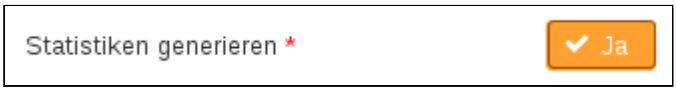
User Property		
Name	**Typ**	**Wert**
DisplayType	String	AutoComplete

Boolean Button (Aktivierungsknopf)

Für Parameter vom Datentyp Boolean kann eine Darstellung als Button spezifiziert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "BooleanButton" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "BooleanButton" unter Advanced eingetragen werden



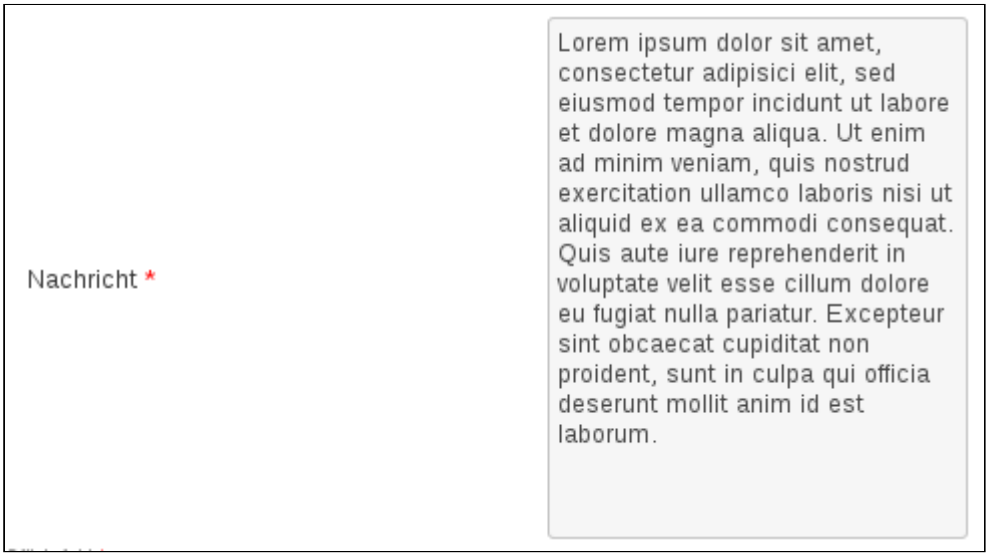
User Property		
Name	**Typ**	**Wert**
DisplayType	String	BooleanButton

Textbereich (Textarea)

Für Textboxen kann die Darstellung von einem einfachen Eingabefeld zu einem Textbereich geändert werden. Auf diese Weise können später auch längere Texte bequem eingegeben werden. Die Höhe des Textbereichs vergrößert sich automatisch, wenn die Größe des eingegebenen Inhalts dies erfordert.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "TextArea" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "TextArea" unter Advanced eingetragen werden



User Property		
Name	**Typ**	**Wert**
DisplayType	String	TextArea

Mittels folgender User Properties kann der Datei Upload Dialog weiter angepasst werden:

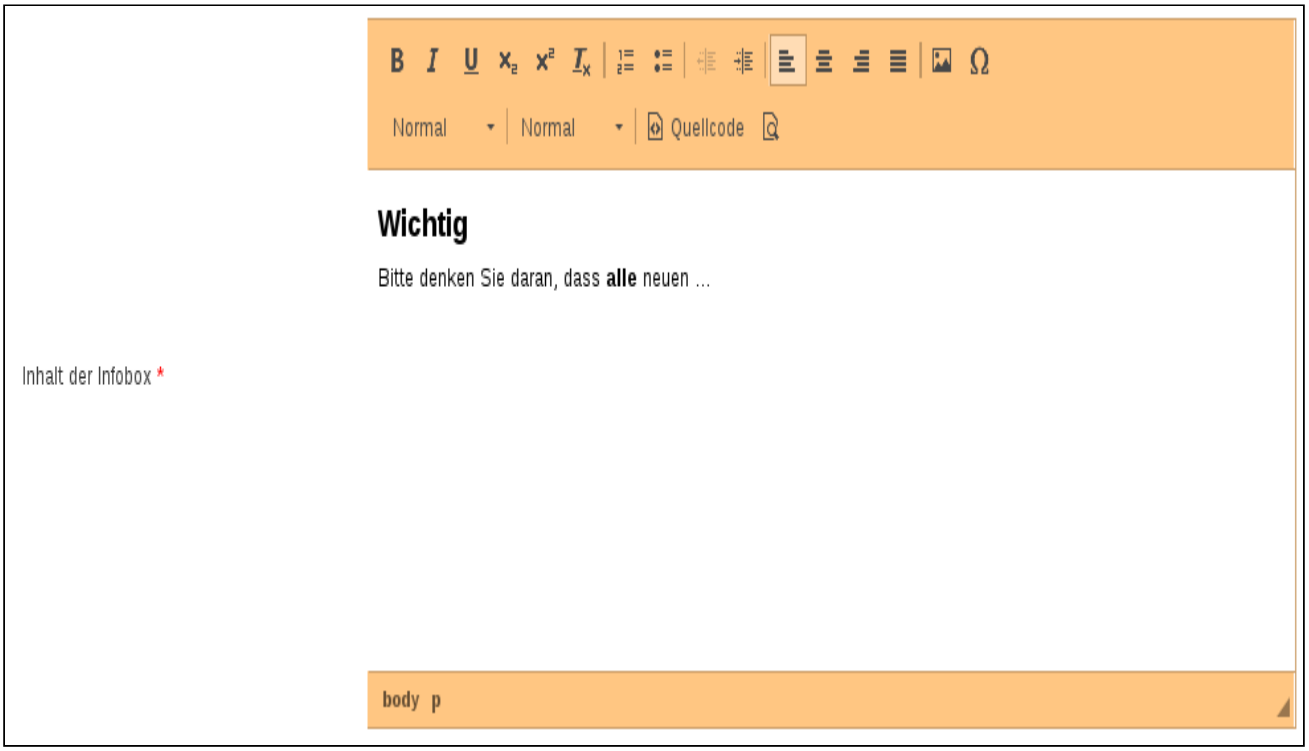
User Property	**Typ**	**Beschreibung (User Property)**	**Beispielwert**	**Beschreibung (Wert)**
ParamWidth	Integer	Parametereingabefeld: Breite	500	Breite in Pixeln
ParamRows	Integer	Parametereingabefeld: Zeilenanzahl	10	Anzahl Zeilen
ParamMaxLength	String	Maximale Parameterlänge	256	Maximale Anzahl an Zeichen, die vom Parameter akzeptiert werden

HTML Editor

Für Textboxen kann eine Darstellung als HTML Editor festgelegt werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "HTMLEditor" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "HTMLEditor" unter Advanced eingetragen werden



User Property		
Name	**Typ**	**Wert**
DisplayType	String	HTMLEditor

Mittels folgender User Properties kann der HTML Editor weiter angepasst werden:

User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
HTMLEditorToolbar	String	Steuert die verfügbaren Button Icons in der Toolbar des Editors	[['Source','Preview'],['Bold', 'Italic']]	Verfügbare Buttons können hier eingesehen werden: https://ckeditor.com/latest/samples/toolbarconfigurator/#basic Besondere Elemente: '-' Trenner '/' Neue Zeile Hinweis: Wird der Wert auf den String "null" gesetzt, so werden alle verfügbaren Buttons zur Toolbar hinzugefügt.

User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
HTMLEditorContentCss	String	Eigene CSS Styles für den Inhalt des Editorfensters	mystyles.css	relativer Pfad zu css File (ausgehend von repository/resources/portal); Wenn Property nicht gesetzt, wird immer repository/resources/portal/html_editor_example.css verwendet.
HTMLEditorSkin	String	Skin für den Editor (vor allem Toolbar und Fußzeile)	office2013	Mögliche Werte: flat, icy_orange, kama, minimalist, moono-lisa, moono, Moono_blue, moonocolor, moono-dark, office2013. Wenn Property nicht gesetzt, wird moono-lisa verwendet.
HTMLEditorInterfaceColor	String	Die Farbe des Editors (vor allem Toolbar und Fußzeile). Funktioniert nicht mit jedem Skin. (Beim Standardskin "moono-lisa" funktioniert es.)	#ffa030	Farbwert als hex String. Wenn Property nicht gesetzt, wird die Primärfarbe des servBIRD Portals verwendet.
HTMLEditorCustomConfig	String	Die Konfiguration des Editors über ein JavaScript Configfile setzen	myconfig.js	relativer Pfad zu js File (ausgehend von repository/resources/portal); Wenn Property nicht gesetzt, wird immer repository/resources/portal/html_editor_config.js verwendet.
ParamWidth	Integer	Parametereingabefeld: Breite	42	Breite in Pixeln
ParamHeight	Integer	Parametereingabefeld: Höhe	42	Höhe in Pixeln (Toolbar muss von Reportentwickler mitbedacht/aufgeschlagen werden)
ParamMaxLength	String	Maximale Parameterlänge	42	Maximale Anzahl an Zeichen, die vom Parameter akzeptiert werden

Text Editor

Als Alternative für den HTML Editor kann für Textboxen kann eine Darstellung als Text Editor festgelegt werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "TextEditor" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "TextEditor" unter Advanced eingetragen werden

User Property		
Name	**Typ**	**Wert**
DisplayType	String	TextEditor

Code Editor

Für Parameter vom Datentyp String kann eine Darstellung als Code Editor spezifiziert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "CodeEditor" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "CodeEditor" unter Advanced eingetragen werden

User Property		
Name	**Typ**	**Wert**
DisplayType	String	CodeEditor

Mittels folgender User Properties kann der Code Editor weiter angepasst werden:

User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
CodeEditorMode	String	Legt den Modus für das Syntax Highlighting fest	text/x-sql	Der Wert kann entweder der Name eine CodeMirror Modes, oder ein MIME Type sein, der von einem der CodeMirror Modes unterstützt wird

User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
CodeEditorTheme	String	Theme für den Editor	eclipse	Mögliche Werte: 3024-day, 3024-night, abcdef, ambiance, ambiance-mobile, base16-dark, base16-light, bespin, blackboard, cobalt, colorforth, dracula, eclipse, elegant, erlang-dark, hopscotch, icecoder, isotope, lesser-dark, liquibyte, material, mbo, mdn-like, midnight, monokai, neat, neo, night, panda-syntax, paraiso-dark, paraiso-light, pastel-on-dark, railscasts, rubyblue, seti, solarized, the-matrix, tomorrow-night-bright, tomorrow-night-eighties, ttcn, twilight, vibrant-ink, xq-dark, xq-light, yeti, zenburn. Eine Vorschau der Themes kann hier eingesehen werden. Wenn Property nicht gesetzt, wird "eclipse" verwendet.
CodeEditorLineNumbers	Boolean	Anzeige von Zeilennummern	true	Der Wert "true" aktiviert die Anzeige von Zeilennummern. Standardeinstellung ist "false".

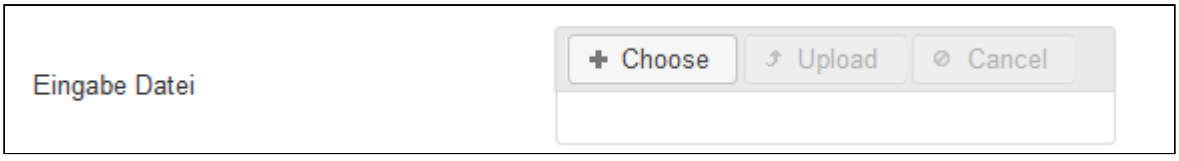
User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
CodeEditorExtraKeys	String	Erlaubt das Hinzufügen, von Tastenkombinationen für Funktionalitätserweiterungen	{'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.sql);}}	Der Beispielwert fügt der Tastenkombination "Strg+Leertaste" die integrierte Code Completion für SQL hinzu. Mögliche Werte für integrierte Code Completions sind: {'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.javascript);}} {'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.coffeescript);}} {'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.xml);}} {'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.html);}} {'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.css);}} {'Ctrl-Space': function(cm) {CodeMirror.showHint(cm, CodeMirror.hint.sql);}}
ParamMaxLength	String	Maximale Parameterlänge	42	Maximale Anzahl an Zeichen, die vom Parameter akzeptiert werden

Datei Upload

Für Parameter vom Datentyp String kann eine Darstellung als Datei Upload Dialog spezifiziert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "FileUpload" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "FileUpload" unter Advanced eingetragen werden



User Property		
Name	**Typ**	**Wert**
DisplayType	String	FileUpload

Mittels folgender User Properties kann der Datei Upload Dialog weiter angepasst werden:

User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
AllowedTypes	String	Erlaubte Dateitypen für Datei Upload Dialog	.jpg, .gif, .png	Dateinamenerweiterungen als komma-separierte Liste
SizeLimit	String	Erlaubte Dateigröße für Datei Upload Dialog	800000	Dateigröße in Bytes. Der Standardwert ist 10000000

Information

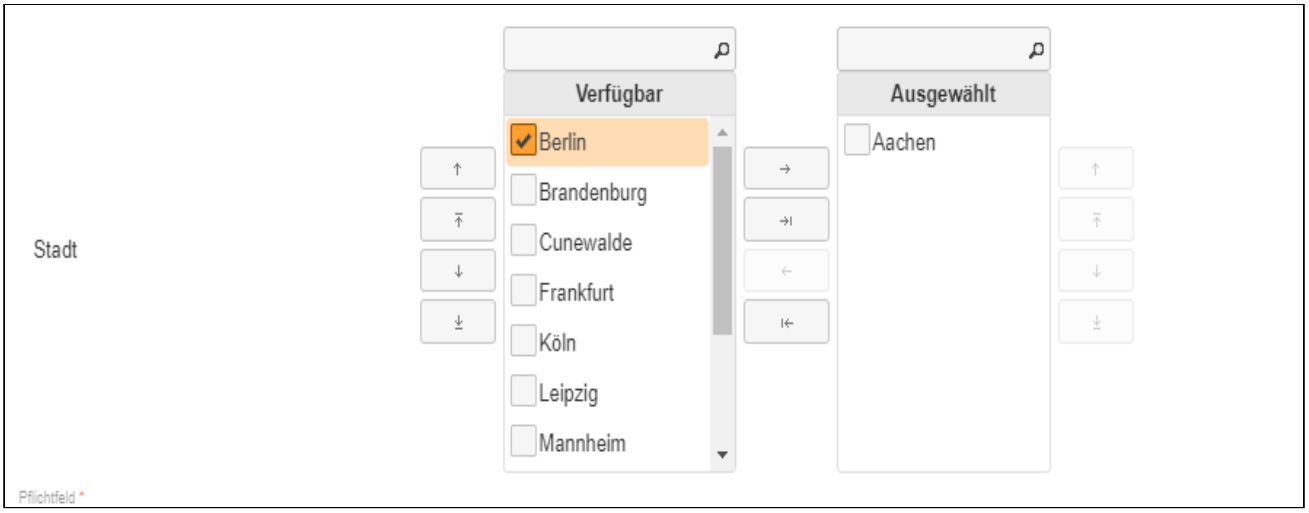
servBIRD wird bei Berichtsausführung den Pfad zur hochgeladenen Datei und (durch das Pipe-Symbol "|" separiert) den ursprünglichen Dateinamen als Parameterwert an den Report übergeben. Beispiel: "/var/tradui/servbird/repository/input/2020/01/17/2b00042f7481c7b056c4b410d28f33cf.jpg|logo.jpg"

PickList

Für Listboxen mit Mehrfachauswahl kann die Darstellung statt einer Auswahlliste zu einer Picklist geändert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "PickList" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "PickList" unter Advanced eingetragen werden
- Der Display type des Parameters muss auf "List Box" gesetzt sein
- Am Parameter muss die Option "Allow multiple values" aktiviert sein



User Property		
Name	**Typ**	**Wert**
DisplayType	String	PickList

Mittels folgender User Properties kann die PickList weiter angepasst werden:

User Property	Typ	Beschreibung (User Property)	Beispielwert	Beschreibung (Wert)
ShowControls	Boolean	Anzeige von Source- und Target Control Buttons bei Source- und Targetlisten	false	Der Wert "false" deaktiviert die Anzeige von Control Buttons. Standardeinstellung ist "true".
showTransferAll	Boolean	Anzeige der Buttons "Alle Elemente verschieben"	false	Der Wert "false" deaktiviert die Anzeige der Buttons "Alle Elemente verschieben". Standardeinstellung ist "true".
ShowCheckBox	Boolean	Anzeige von Kontrollkästchen bei Elementen der Source- und Targetlisten	false	Der Wert "false" deaktiviert die Anzeige von Kontrollkästchen. Standardeinstellung ist "true".
PortalHeight	Integer	Anzahl der gleichzeitig dargestellten Zeilen (= Höhe der PickList)	7	Es werden nur Werte von 3 bis 25 unterstützt. Standardwert ist 7.
ParamWidth	Integer	Breite der PickList in Pixeln	500	Standardwert ist 500. Auf Grund des Platzbedarfs einer PickList sind ausschließlich Werte größer 500 sinnvoll.

RadioButton

Für ListBoxen mit Einzelwerten kann die Darstellung statt einer Auswahlliste zu Radio Buttons geändert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "RadioButton" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "RadioButton" unter Advanced eingetragen werden

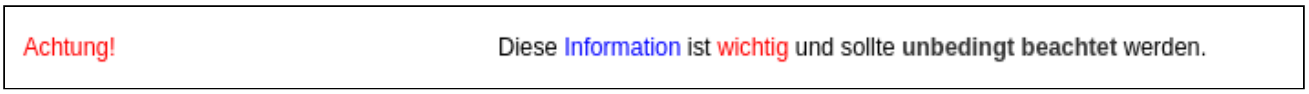
User Property		
Name	**Typ**	**Wert**
DisplayType	String	RadioButton

Textausgabe

Anstelle einer Parametereingabemaske können mittels eines Textausgabeparameters beliebige Informationen angezeigt werden. Diese können zur Formatierung beliebiges HTML Markup enthalten.

Zum Anzeigen von Informationen ist am entsprechenden Parameter das User Property DisplayType mit dem Wert "TextOutput" zu definieren.

- User Property DisplayType vom Typ String am Parameter definieren
- Als Wert für das Property muss "TextOutput" unter Advanced eingetragen werden
- Der Parameter darf nicht als "Is Required" definiert sein
- Der Prompt-Text des Parameters wird in der Parametermaske in der Spalte für den Parameternamen angezeigt
- Der Default-Wert des Parameters bestimmt, was an Stelle des Parametereingabefeldes angezeigt wird



User Property		
Name	**Typ**	**Wert**
DisplayType	String	TextOutput

Erscheinungsbild

Parametereingabefeld: Höhe

Für HTML-Editor und Text-Editor Parameter kann die Höhe definiert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property ParamHeight mit entsprechendem Wert zu definieren.

- User Property ParamHeight vom Typ Integer am Parameter definieren
- Wert für das Property (Höhe in Pixeln) unter Advanced eintragen

User Property		
Name	**Typ**	**Wert**
ParamHeight	Integer	42

Parametereingabefeld: Breite

Für Textbox, Textbereich, Combobox, Listen, HTML-Editor, Text-Editor und PickList Parameter kann die Breite definiert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das UserProperty ParamWidth mit entsprechendem Wert zu definieren.

- User Property ParamWidth vom Typ Integer am Parameter definieren
- Wert für das Property (Breite in Pixeln) unter Advanced eintragen

User Property		
Name	**Typ**	**Wert**
ParamWidth	Integer	42

Parametereingabefeld: Zeilenanzahl

Für Textbereich Parameter kann die Höhe definiert werden.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property ParamRows mit entsprechendem Wert zu definieren.

- User Property ParamRows vom Typ Integer am Parameter definieren
- Wert für das Property (Anzahl der anzuzeigenden Zeilen) unter Advanced eintragen

User Property		
Name	**Typ**	**Wert**
ParamRows	Integer	42

Parametereingabefeld: Anzuzeigende Einträge

Für Listboxen und PickLists kann die Höhe definiert werden, indem die Anzahl der gleichzeitig anzuzeigenden Einträge definiert wird. Sollten sich mehr Elemente in der Auswahlliste befinden, so wird ein Scrollleiste angezeigt.

Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property PortalHeight mit entsprechendem Wert zu definieren.

- User Property PortalHeight vom Typ Integer am Parameter definieren

- Wert für das Property (Anzahl anzuzeigender Einträge) unter Advanced eintragen
  (Die PickList unterstützt ausschließlich Werte von 3 bis 25)

User Property		
Name	**Typ**	**Wert**
PortalHeight	Integer	42

Spalte mit Namen des Parameters ausblenden

Über ein User Property am Parameter soll die Spalte mit dem Namen des Parameters ausgeblendet werden.

User Property		
Name	**Type**	**Wert**
HideParamName	Boolean	true

Auswahlbuttons für ListBox Parameter ausblenden

Für multivalue ListBox Parameter können die "Alles auswählen" und "Nichts auswählen" Buttons über ein User Property ausgeblendet werden. Der Standardwert ist true , d.h die Buttons werden angezeigt.

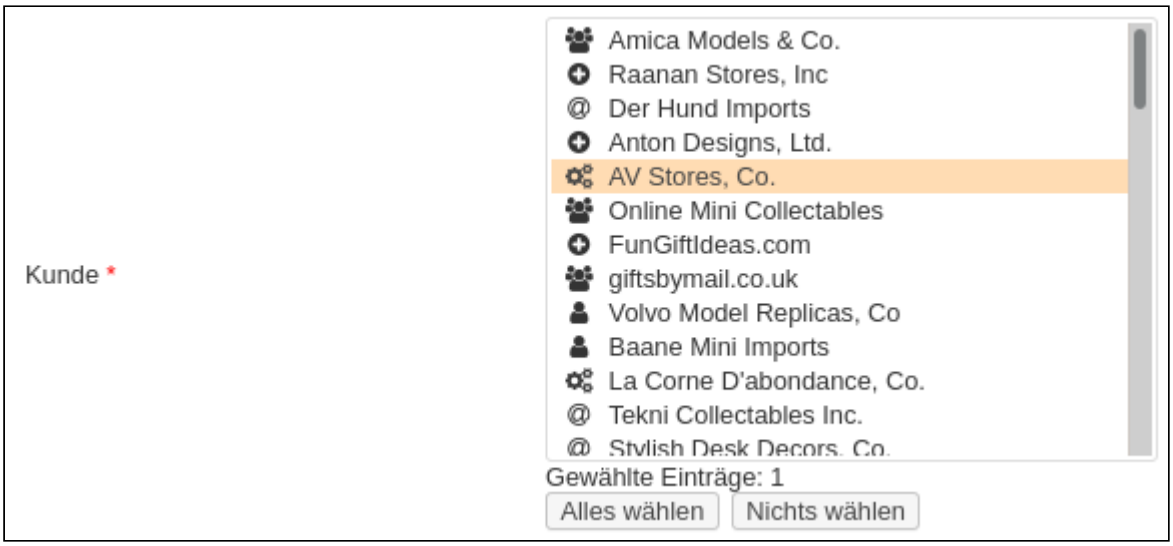
Zum Anpassen der Anzeige ist am entsprechenden Parameter das User Property ListBoxSelectionButtons mit entsprechendem Wert zu definieren.

- User Property ListBoxSelectionButtons vom Typ Boolean am Parameter definieren
- Wert für das Property (ob die Buttons angezeigt werden sollen) unter Advanced eintragen

User Property		
Name	**Type**	**Wert**
ListBoxSelectionButtons	Boolean	false

Icons für Auswahlwerte anzeigen

Vor den Auswahlwerten von ListBox und PickList Parametern können Icons angezeigt werden.



Um die Funktionalität zu aktivieren ist am entsprechenden Parameter das User Property LabelIcons zu definieren und mit dem Wert "true" zu belegen. Zusätzlich müssen die Labels der Auswahlliste mit einem Präfix versehen werden.

- User Property LabelIcons vom Typ Boolean am Parameter definieren
- Wert für das Property (ob die Icons angezeigt werden sollen) unter Advanced eintragen
- Am Parameter müssen die Labels der Auswahlliste mit einem Präfix nach folgendem Schema beginnen: [icon <CLASS>]
  Anstatt dem Platzhalter "<CLASS>" muss eine gültige Iconklasse verwendet werden. Verwendbare Icons finden sich in der Liste verfügbarer Icons.

Beispiel: Auswahllistenlabels für die Darstellung von Icons

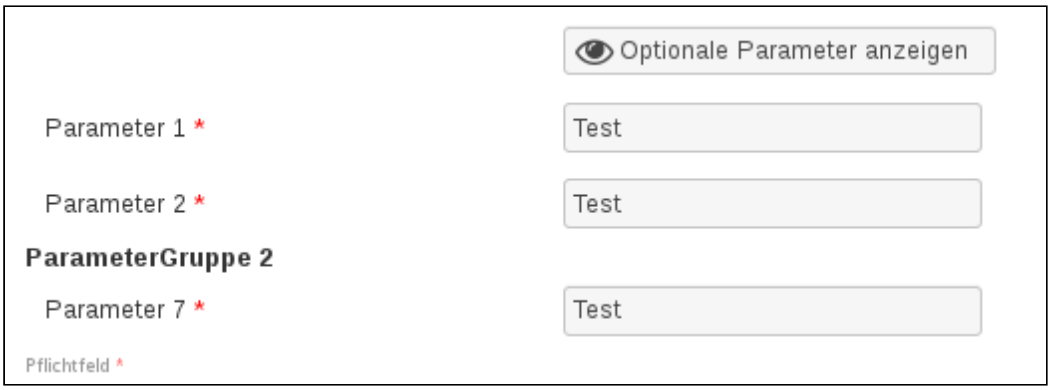
[icon fa-users]Amica Models & Co.
[icon fa-plus-circle]Raanan Stores, Inc
[icon fa-at]Der Hund Imports

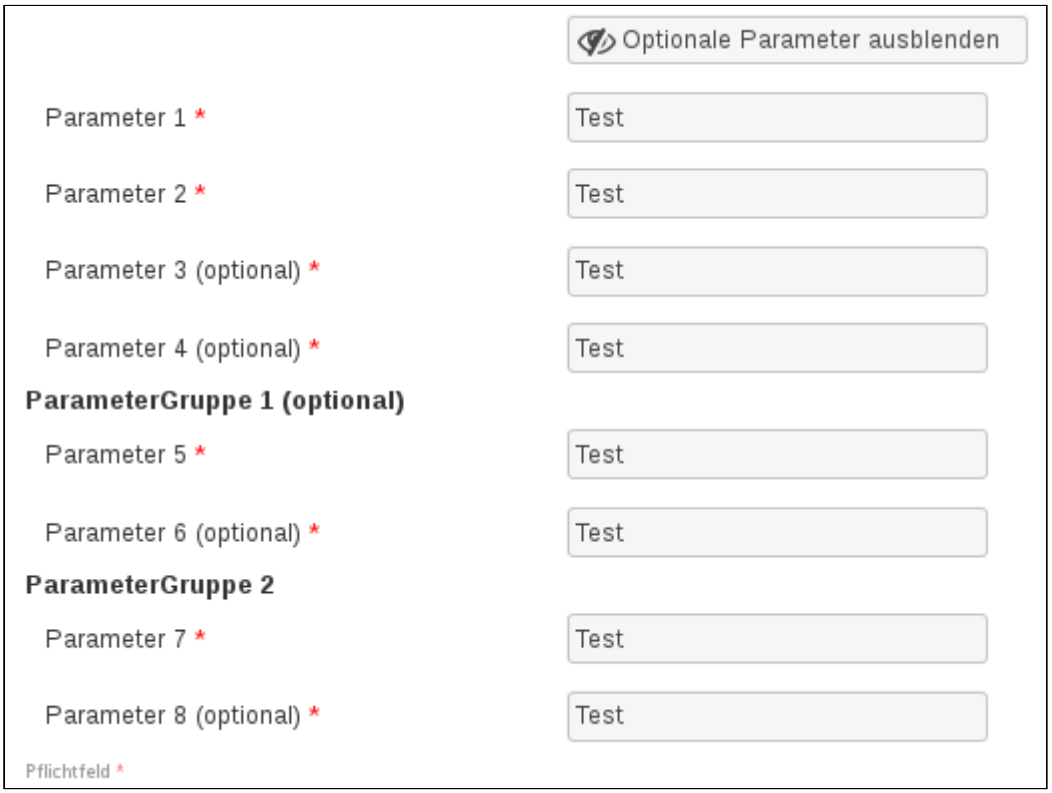
2.4.3 Verhalten

Optionale Parameter

Optionale Parameter / Parametergruppen werden standardmäßig ausgeblendet. Im Portal erscheint dann ein Button zum Ein-/Ausblenden der optionalen Parameter.

- User Property Optional vom Typ Boolean am Parameter definieren
- Wert für das Property auf "true" setzen





User Property		
Name	**Typ**	**Wert**
Optional	Boolean	true

Unveränderliche Parameter

Parameter können als unveränderlich gekennzeichnet werden. Nutzer können die Felder dann sehen, aber nicht aktiv verändern.

Um ein Feld entsprechend zu markieren, müssen Sie am Parameter ein User Property ReadOnly mit dem Wert "true" belegen.

- UserProperty ReadOnly vom Typ Boolean am Parameter definieren
- Wert für das Property auf "true" setzen

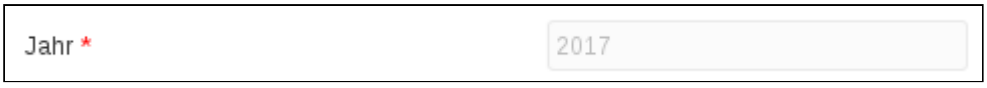
User Property		
Name	**Typ**	**Wert**
ReadOnly	Boolean	true

Bedingt unveränderliche Parameter

Parameter können in Abhängigkeit von anderen Parameterwerten als unveränderlich gekennzeichnet werden. Nutzer können die Felder dann bei zutreffender Bedingung zwar sehen, aber nicht aktiv verändern.

Um einen Parameter entsprechend zu markieren, müssen Sie am Parameter ein User Property ReadOnlyCondition anlegen und als Wert eine Bedingung in JavaScript Syntax angeben. Innerhalb der Bedingung können andere Parameterwerte mittels eckigen Klammern verwendet werden.

- User Property ReadOnlyCondition vom Typ String am Parameter definieren
- Als Wert für das Property eine Bedingung in JavaScript Syntax angeben
- Beispiel: ('[paramLand]' == 'Deutschland' && '[paramStadt]' == 'Berlin')
- Ergebnis: Dieser Parameter wird in dem Fall unveränderlich, wenn der Parameter "paramLand" auf den Wert "Deutschland" und der Parameter "paramStadt" auf den Wert "Berlin" eingestellt ist



User Property		
Name	**Typ**	**Wert**
ReadOnlyCondition	String	('[paramLand]' == 'Deutschland' && '[paramStadt]' == 'Berlin')

Hinweis

Bitte beachten Sie, dass in der ReadOnlyCondition keine Multivalue Parameter verwendet werden können.

Warnung

Bitte beachten Sie, dass durch den Einsatz von bedingten Parametern und auslösenden Parametern Wechselwirkungen zwischen Parametern definiert werden. Achten Sie daher besonders darauf, dass Sie keine zyklischen Abhängigkeiten generieren indem sich Parameter gegenseitig beeinflussen.

Maximale Parameterlänge

Für Standardeingabefelder und Textboxen kann die maximale Eingabelänge geändert werden (Default ist 2147483647).

- User Property ParamMaxLength vom Typ String am Parameter definieren
- Wert für das Property auf die gewünschte maximale Anzahl Zeichen setzen

User Property		
Name	**Typ**	**Wert**
ParamMaxLength	String	200

Filter

Die selektierbaren Werte von Comboboxen und Listenparametern können mittels eines Filters eingeschränkt werden, damit der Benutzer nach für ihn relevanten Werten filtern kann.

- User Property ListBoxFilter vom Typ Boolean am Parameter definieren
- Wert für das Property auf "true" setzen

User Property		
Name	**Typ**	**Wert**
ListBoxFilter	Boolean	true

Mit einem zusätzlichen User Property kann zudem die Filtermethode angepasst werden:

User Property	Typ	Beispielwert	Beschreibung (Wert)
ListBoxFilterMode	String	startsWith	Mögliche Werte sind • startsWith • contains • endsWith

Bedingt sichtbare Parameter

Parameter können in Abhängigkeit von anderen Parameterwerten ein- bzw. ausgeblendet werden.

Um einen Parameter entsprechend zu markieren, müssen Sie am Parameter ein User Property VisibilityCondition anlegen und als Wert eine Bedingung in JavaScript Syntax angeben. Innerhalb der Bedingung können andere Parameterwerte mittels eckigen Klammern verwendet werden.

- User Property VisibilityCondition vom Typ String am Parameter definieren
- Als Wert für das Property eine Bedingung in JavaScript Syntax angeben
- Beispiel: ('[paramLand]' == 'Deutschland' && '[paramStadt]' == 'Berlin')
- Ergebnis: Dieser Parameter ist nur in dem Fall sichtbar, wenn der Parameter "paramLand" auf den Wert "Deutschland" und der Parameter "paramStadt" auf den Wert "Berlin" eingestellt ist

User Property		
Name	**Typ**	**Wert**
VisibilityCondition	String	('[paramLand]' == 'Deutschland' && '[paramStadt]' == 'Berlin')

Hinweis

Bitte beachten Sie, dass in der VisibilityCondition keine Multivalue Parameter verwendet werden können.

Warnung

Bitte beachten Sie, dass durch den Einsatz von bedingten Parametern und auslösenden Parametern Wechselwirkungen zwischen Parametern definiert werden. Achten Sie daher besonders darauf, dass Sie keine zyklischen Abhängigkeiten generieren indem sich Parameter gegenseitig beeinflussen.

Auslösende Parameter

Parameter können Einfluss auf andere Parameter, bzw. deren Auswahlmöglichkeiten und/oder Defaultwerte, ausüben. Auf diese Weise lassen sich ähnliche (aber flexiblere) Ergebnisse erzielen, wie mit Cascading Parametern.

Über ein User Property UpdatesParameters kann eine kommaseparierte Liste von Parameternamen angegeben werden, die durch diesen Parameter beeinflusst werden.

- User Property UpdatesParameters vom Typ String am Parameter definieren
- Als Wert eine kommaseparierte Liste von Parameternamen angegeben
- Beispiel: paramLand,paramStadt
- Ergebnis: Wird der Parameterwert dieses Parameters verändert, so werden die Parameter "paramLand" und "paramStadt" neu geladen (und somit deren Defaultwerte und Auswahllisten aktualisiert)

User Property		
Name	**Typ**	**Wert**
UpdatesParameters	String	parameterName1,parameterName2

Hinweis

Bitte beachten Sie, dass ein auslösender Parameter keine als "hidden" gekennzeichneten Parameter beeinflussen kann.

Warnung

Bitte beachten Sie, dass durch den Einsatz von bedingten Parametern und auslösenden Parametern Wechselwirkungen zwischen Parametern definiert werden. Achten Sie daher besonders darauf, dass Sie keine zyklischen Abhängigkeiten generieren indem sich Parameter gegenseitig beeinflussen.

Null-Checkbox ausblenden

In der Parametereingabemaske bietet servBIRD bei Parametern, die als "nicht required" definiert sind, rechts neben dem Eingabefeld eine Checkbox an, um für diesen Parameter überhaupt keinen Wert an den Report weiterzugeben. Da es in manchen Fällen wüschenswert ist, diese Checkbox nicht anzuzeigen, kann diese durch ein User Property ausgeblendet werden. In diesem Fall würde für den Parameter immer ein Wert an den Report weitergegeben - im Falle eines leeren Parametereingabefeldes wäre dies beispielsweise ein leerer String.

Um an einem "nicht required" Parameter die Null-Checkbox auszublenden, müssen Sie am Parameter ein User Property HideNullCheckbox mit dem Wert "true" belegen.

- UserProperty HideNullCheckbox vom Typ Boolean am Parameter definieren
- Wert für das Property auf "true" setzen

User Property		
Name	**Typ**	**Wert**
HideNullCheckbox	Boolean	true

Berechtigungen auf Parameter

Für Parameter, die nur für User mit bestimmten Rollen sichtbar sein sollen, ist jeweils am Parameter oder an der Parametergruppe ein User Property zu definieren.

- User Property "PortalParamSecurityRoles" vom Typ String am/an Parameter/-gruppe definieren
- Wert für das Property ist der Rollenname bzw. eine kommagetrennte Liste von Rollennamen

User Property		
Name	**Typ**	**Wert**
PortalParamSecurityRoles	String	role1,role2,role3

3 Spezialparameter

In **servBIRD** ist es möglich zusätzliche Funktionalitäten über spezielle Parameter bereitzustellen. Hierzu wird einem BIRT Report ein neuer Parameter mit den unten genannten Eigenschaften hinzugefügt. Dieser Parameter wird nicht, wie ein herkömmlicher BIRT Parameter zur Anzeige gebracht, sondern erfüllt eine besondere Funktion.

3.1 Default-Wert

Wenn ein Default-Wert für ein Parameter aus der Datenbank kommen soll, kann das über einen zusätzlichen Parameter erfolgen. Der zusätzliche Parameter muss vom Namen mit dem Präfix "default_" beginnen und der hintere Teil identisch mit dem Names des Parameters sein, dessen Wert gesetzt werden soll. Der neue Parameter ist eine dynamische Listbox vom richtigen Datentyp, nicht required, nicht hidden und das Dataset muss genau einen Wert liefern.

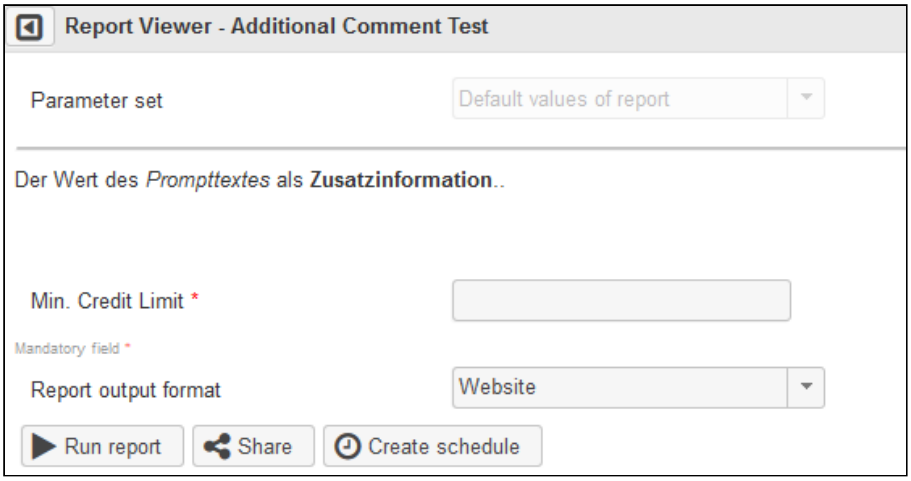
Der Parameter wird von **servBIRD** automatisch auf Hidden gesetzt und damit nicht angezeigt.

3.2 Zusatzinfo Parameter

Mit dem Zusatzinfo Parameter kann in **servBIRD** ein Informationstext über der Parametereingabemaske dargestellt werden. Hierbei kann HTML Markup verwendet werden, um den Text in der gewünschten Formatierung darstellen zu lassen. So können Sie z.B. mittels "**
" einen Zeilenumbruch hinzufügen. Der anzuzeigende Text kann über den Default-Wert des Parameters definiert werden. Sollte der Default-Wert des Parameters leer sein, wird **servBIRD den Prompt-Text des Parameters anzeigen. Die Voraussetzungen für einen Parameter mit dieser Funktionalität sind:

- Parametername: paramSERVBIRDAdditionalComment
- Hidden: true
- Default value: Der gewünscht Text, der über der Parametereingabemaske dargestellt werden soll

Ein Parameter mit oben genannten Eigenschaften und dem beispielhaften Default-Wert "Der Wert des Prompttextes als Zusatzinformation.." wird so angezeigt:



3.3 Debug-Parameter

Parameter die mit dem Präfix "**DEBUG_**" beginnen, können vom Portal generell ausgeblendet werden. Über die Konfigurationseinstellungen von **servBIRD** kann eingestellt werden, ob diese Debug-Parameter angezeigt oder ausgeblendet werden sollen.

4 Berichtskontext Variablen

Über den ApplicationContext vom Report gibt der Server Informationen an den Report und der Report kann Funktionen des Servers steuern.

4.1 Informationen zur Reportausführung abgreifen

4.1.1 Aktueller User

Der aktuelle User kann aus dem ApplicationContext des Reports ausgelesen werden.

```
glUser = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("USER_NAME"))
{
    glUser = appContext.get("USER_NAME");
}
```

4.1.2 User-Rollen

Die Rollen des Users (genauer: deren Namen, bzw. im Fall von LDAP Nutzern deren unterscheidbarer Name (DN)) werden als Semikolon-getrennte Liste in einem String geliefert.

```
glUserRoles = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("USER_ROLES_DN"))
{
    glUserRoles = appContext.get("USER_ROLES_DN");
}
```

Die Rollen des Users (genauer: deren Anzeigenamen) werden zusätzlich als Komma-getrennte Liste in einem String geliefert.

```
glUserRoles = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("USER_ROLES"))
{
    glUserRoles = appContext.get("USER_ROLES");
}
```

4.1.3 Job-ID

Die JobID ist vom Typ Long.

```
glJobID = 0;
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("REPORT_JOBID"))
{
    glJobID = appContext.get("REPORT_JOBID");
}
```

4.1.4 Parameter Set

Der Name des aktuell ausgewählten Parameter Set's ist vom Typ String

```
glParamSet = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("USER_PARAM_SET"))
{
    glParamSet  = appContext.get("USER_PARAM_SET");
}
```

4.1.5 Ausführungstyp

Der Ausführungstyp des aktuellen Berichts bzw. Jobs ist vom Typ String

```
glExecutionTyp = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("REPORT_EXECUTIONTYPE"))
{
    glExecutionTyp= appContext.get("REPORT_EXECUTIONTYPE");
}
```

4.1.6 Ausgabeformat

Das in servBIRD gewählte Ausgabeformat des aktuellen Berichts bzw. Jobs ist vom Typ String.

Mögliche Werte sind: PDF, PPT, HTML, EXCEL, DOC, ODT, ODS, ODP, DOCX, PPTX, XLS_SPUDSOFT, XLSX

```
glOutputFormat = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("REPORT_OUTPUTFORMAT"))
{
    glOutputFormat = appContext.get("REPORT_OUTPUTFORMAT");
}
```

4.1.7 Dashboard/Cockpit Pfad

Der Ausgabedatei Pfad eines Dashboard/Cockpit ist vom Typ String

```
glDashboardPath = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("REPORT_DASHLET_PATH"))
{
    glDashboardPath = appContext.get("REPORT_DASHLET_PATH");
}
```

4.1.8 Bericht wird auf servBIRD ausgeführt

Prüft ob der aktuelle Bericht auf servBIRD ausgeführt wird und ist vom Typ Boolean.

```
glRunOnServer = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("RUN_ON_SERVER"))
{
    glRunOnServer = appContext.get("RUN_ON_SERVER");
}
```

4.1.9 Hostname des Servers

Der Hostname des Servers, der den Bericht generiert, ist vom Typ String.

```
glHostName = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("HOST_NAME"))
{
    glHostName = appContext.get("HOST_NAME");
}
```

4.1.10 Systemname der servBIRD Installation

Der Systemname der servBIRD Installation, die den Bericht generiert, (ist für alle Nodes eines servBIRD Clusters gleich) und ist vom Typ String.

```
glSystemName = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("SYSTEM_NAME"))
{
    glSystemName = appContext.get("SYSTEM_NAME");
}
```

4.1.11 Native Ausgabedokumente

Der Ausgabedatei Pfad inklusive des Namens der erzeugten nativen Dokumente vom Typ String.

```
glNativeFileName = "";
var appContext = reportContext.getAppContext();   // HashMap
if (appContext != null && appContext.containsKey("REPORT_NATIVE_FILE_NAME"))
{
    glNativeFileName = appContext.get("REPORT_NATIVE_FILE_NAME");
}
```

4.2 Ausgabedateiname setzen

Der angezeigte Dateiname im Bereich **Fertige Berichte** kann vom Report beeinflusst werden, sodass z.B. Parameter in den Dateinamen einfließen.

```
// reportContext.getAppContext().put("SERVBIRD_DISPLAYOUTPUTNAME", "neuer Name");
var sdt = Formatter.format(params["paramStichtag"].value, "yyyy-MM");
reportContext.getAppContext().put("SERVBIRD_DISPLAYOUTPUTNAME",
ExpertsCommon.getReportDesignFileWithoutExt(reportContext) + "_" + sdt);
```

4.3 Dokumenten Eigner (Owner) überschreiben/setzen

Um den eigentlichen Eigner eines Dokumentes (z.B. innerhalb einer Parameter-Loop) zu überschreiben/zu setzen gibt es das Attribut `PROFESSIONAL_OWNER`.

Dieses kann über den Application Context innerhalb eines Reports gesetzt werden.

```
reportContext.getAppContext().put("PROFESSIONAL_OWNER", "max.mustermann");
```

> **Hinweis**
>
> Es muss sich um einen gültigen Benutzernamen handeln (z.B. der LDAP Benutzername).

5 Interaktive Berichte (BIRD Interactives)

5.1 Applikationen entwickeln

Das Modul **BIRD Applications** ermöglicht die Nutzung von BIRT Berichten als Grundlage für Webformulare, sodass diese im **servBIRD**-Portal ausgeführt werden können.
Anzeigen, Erfassen und Speichern von entsprechenden Formulardaten ermöglicht die Einrichtung von komplexen Prozessen im **servBIRD**-Portal.
Zusätzlich lassen sich über die Formulare Daten hinzufügen, ändern und löschen.

5.1.1 Voraussetzungen

Für dieses Modul ist der Einsatz der **Toolbox** mit dem Modul **Data Manager** erforderlich.

> **Information**
>
> **BIRD Applications** stellt ein eigenes Modul dar und muss gesondert lizenziert werden. Für Angebote, Preise und weitere Informationen kontaktieren Sie uns bitte über sales@tradui.de.

5.1.2 Hauptbericht

Der Hauptbericht einer Application stellt die "Startseite" der Anwendung dar. Hier könnte beispielsweise, je nach Anwendungsfall der Anwendung, zu bearbeitende Datensätze tabellarisch aufgelistet werden. Jede Zeile könnte verschiedene Buttons enthalten, die auf Formularberichte verweisen und auf diese Weise unterschiedliche Operationen auf den Datensätzen erlauben.

> **Hinweis**
>
> **servBIRD** erkennt den Hauptbericht immer an der Datei-Endung **.app.rptdesign**. (Beispiel: **Auftragsübersicht.app.rptdesign**)

Reportdesign Properties

Folgende Properties sind obligatorisch und müssen im Hauptbericht gesetzt sein:

Art	Schlüssel	Typ	Wert (Bespiel)	Beschreibung
General Property	title	String	Mein Dashboard	Der Name der Application, mit dem sie auch im **servBIRD** angezeigt wird. Dieser muss eindeutig sein, da er als ID der Application verwendet wird.

Art	Schlüssel	Typ	Wert (Bespiel)	Beschreibung
User Property	SERV.INTERACTIVE.GROUP	String	Meine Gruppe	Der Name der Gruppe der Application. Die Application wird im **servBIRD** innerhalb dieser Gruppe angezeigt werden. Eine Gruppe ist als eine Art Ordner zu verstehen.
User Property	SERV.INTERACTIVE.COLOR	String	#FFAAFF	Der hex-Code für die Farbe des Symbols, mit dem die Application in **servBIRD** angezeigt wird.
User Property	SERV.INTERACTIVE.ICON	String	fa-user	Der Klassenname für ein FontAwesome Icon dass das Symbol zieren soll, mit dem die Application in **servBIRD** angezeigt wird. Siehe hierzu: Liste verfügbarer Icons
User Property	SERV.INTERACTIVE.SORT	Integer	1	Die Sortiernummer für Auflistung dieser Application in eigener Gruppe

Hinweis

Achten Sie darauf, für unterschiedliche Applications unterschiedliche Titel zu vergeben. Wird im **servBIRD** wiederholt eine Application mit gleichem Titel hochgeladen, so wird die bereits vorhandene überschrieben. Neue Versionen einer Application sollten demnach also den gleichen Titel tragen, damit alte Versionen automatisch aktualisiert werden.

Icon- und Vorschaubilder

Um den Wiedererkennungswert der Application innerhalb von **servBIRD** zu steigern, kann vom Reportentwickler ein Icon und ein Vorschaubild für die Application gesetzt werden. Hierzu müssen diese als Bilddateien im Application zip-Archiv vorliegen und wie folgt benannt sein:

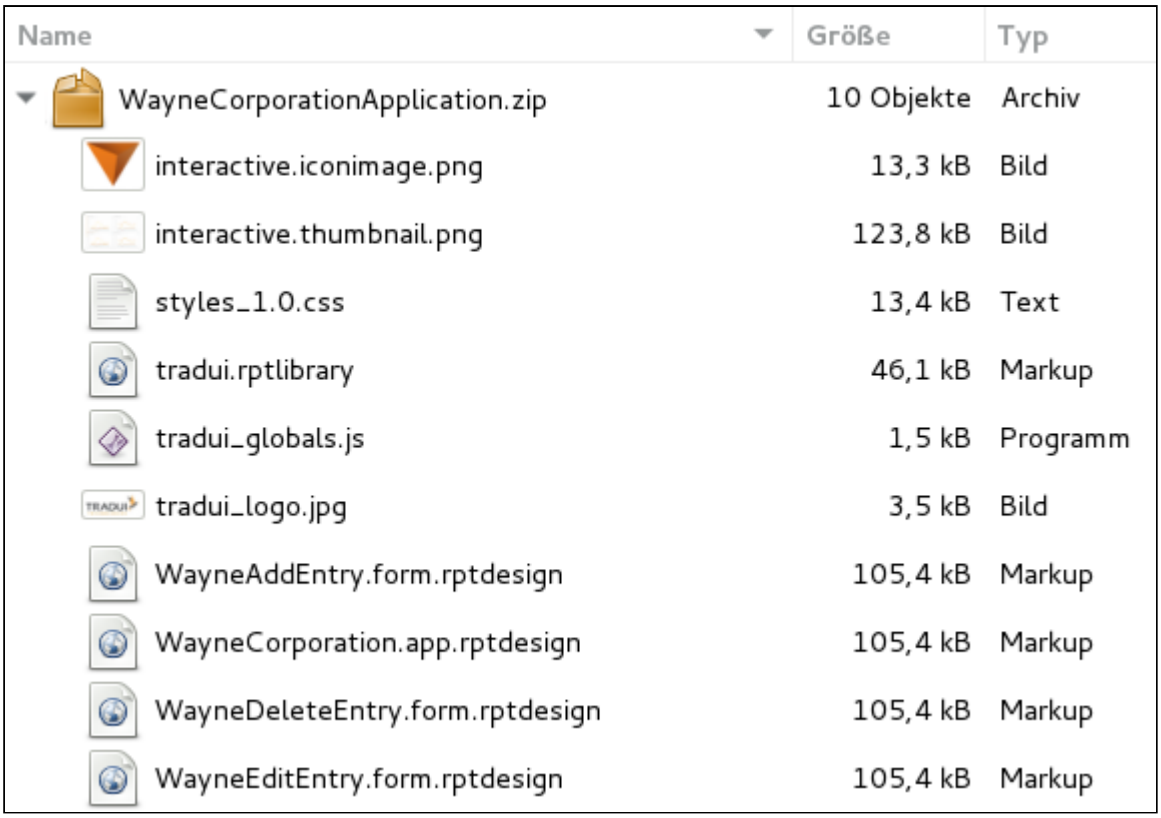
Art	Erlaubte Dateinamen
Icon	interactive.iconimage.png interactive.iconimage.jpg interactive.iconimage.gif
Vorschaubild	interactive.thumbnail.png interactive.thumbnail.jpg interactive.thumbnail.gif

Hinweis

Pro Application ist jeweils nur ein Icon- und Vorschaubild erlaubt.

5.1.3 Paketierung

Damit die Applikation in **servBIRD** importiert werden kann, muss es bestimmte Anforderungen an Struktur und Aufbau erfüllen. Zum einen müssen spezielle User Properties im Hauptreport gesetzt sein und dessen Dateiname auf *.application.rptdesign* enden. Zum anderen müssen die verwendeten Ressourcen und alle Designfiles des Dashboards als zip-Archiv komprimiert werden. Im Folgenden ist der schematische Aufbau eines Applikation zip-Archivs dargestellt.



Hinweis

Alle Inhalte des zip-Archivs, die kein Reportdesign oder spezielles Thumbnail/Iconimage Bild sind, werden von servBIRD als Reportressourcen interpretiert und im **servBIRD** Repository im Ordner *resources* abgelegt. Das gilt selbstverständlich auch für Ressourcen, die in Unterordnern organisiert sind. Sollten sich Ressourcenverzeichnis von **servBIRD** bereits gleichnamige Dateien befinden, so werden diese überschrieben.

5.1.4 Formularberichte entwickeln

Formular-Berichte sind spezielle BIRT-Berichte, die Eingabemasken im **servBIRD**-Portal bereitstellen. Die Eingabefelder werden hierbei über BIRT-Parameter abgebildet. Der Formular-Bericht selbst übernimmt bei der Ausführung die Validierung und Verarbeitung der Formulardaten. Im Gegensatz zum "normalen" BIRT-Bericht wird kein Ausgabeformat im Sinne eines Berichts erzeugt.

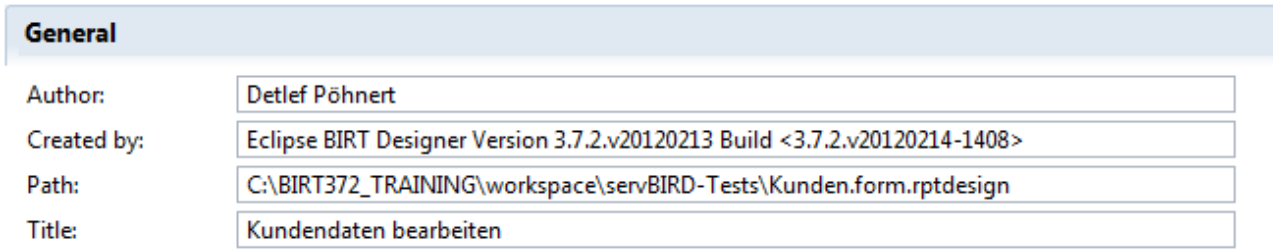
> **Hinweis**
>
> **servBIRD** erkennt Formularberichte immer an der Datei-Endung **.form.rptdesign**. (Beispiel: **Aufragserfassung.form.rptdesign**)

Formular-Berichte erstellen

Erstellen sie einen BIRT-Bericht und legen Sie für jedes Eingabefeld einen Parameter an. Sie können auch Hidden-Parameter verwenden, wenn Sie zum Beispiel eindeutige Datensatz-Ids für ihr Formular benötigen. Alle sichtbaren Parameter werden als Eingabefelder des Formulars angezeigt.

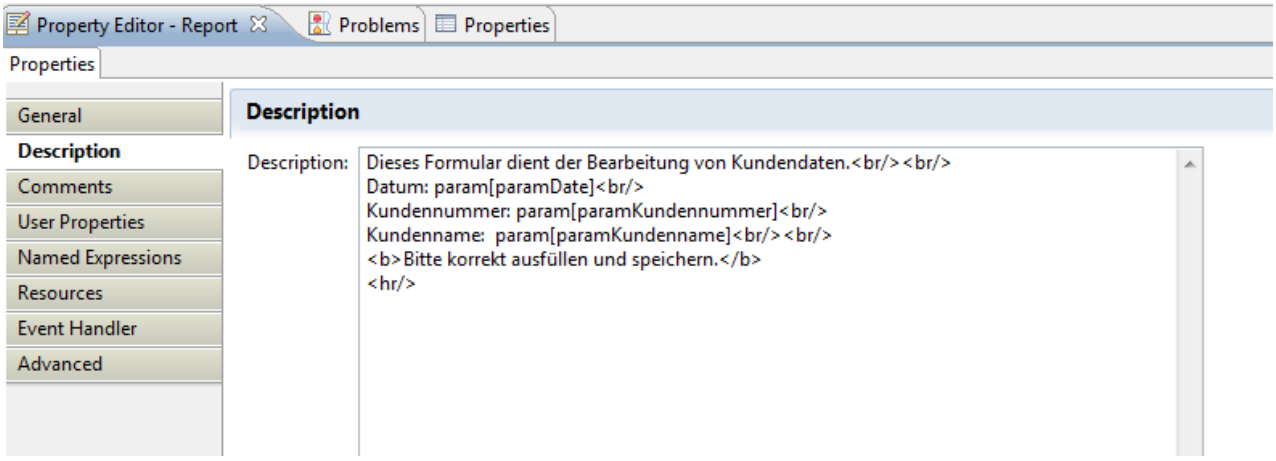
Formular-Titel festlegen

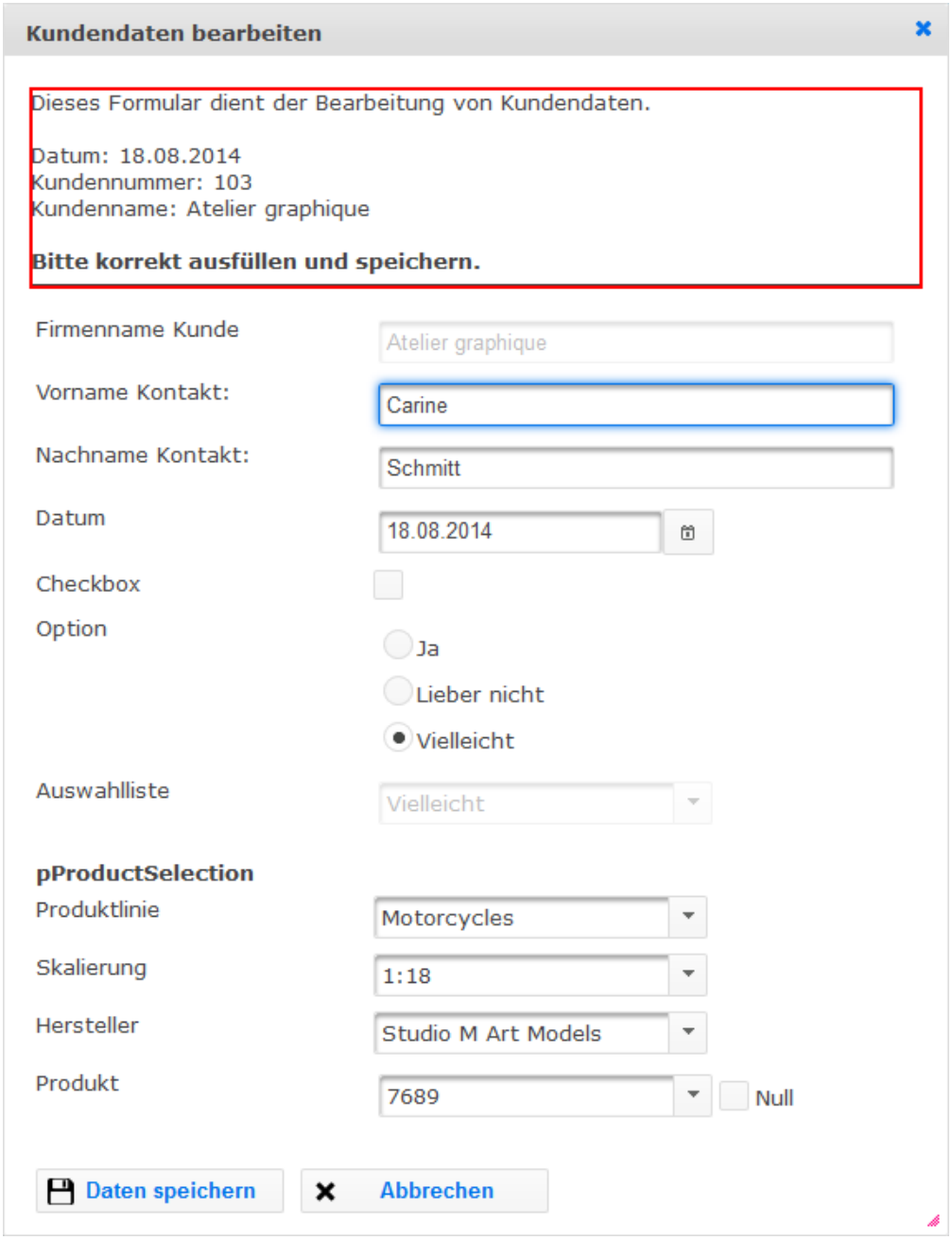
Den Default-Formular-Titel können Sie überschreiben, indem Sie im Formular-Bericht einen Titel hinterlegen. Dieser wird dann auch als Formular-Titel angezeigt.



Formular-Beschreibung festlegen

Formulare zeigen im oberen Teil eine Beschreibung an, wenn diese im Formular-Report hinterlegt ist.



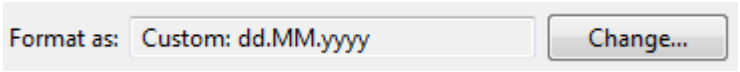


Parameter in der Formular-Beschreibung verwenden

Sie können in der Beschreibung die Werte von Parametern anzeigen - auch unsichtbare Parameter.

Der Syntax dafür lautet: **param[Parametername]**

In der Anzeige werden die Parameter so angezeigt, wie es im Display-Format des Parameters definiert wurde.

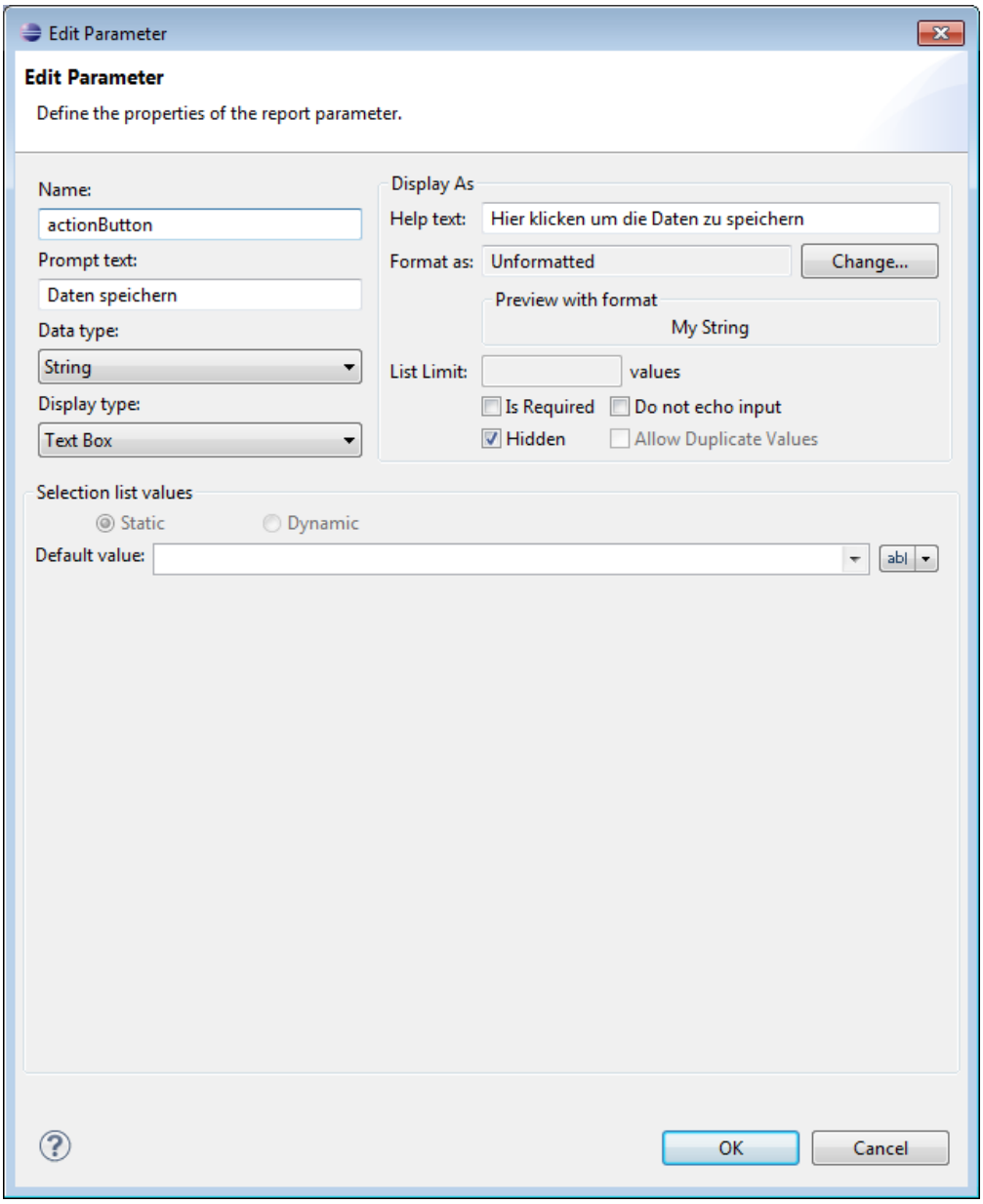


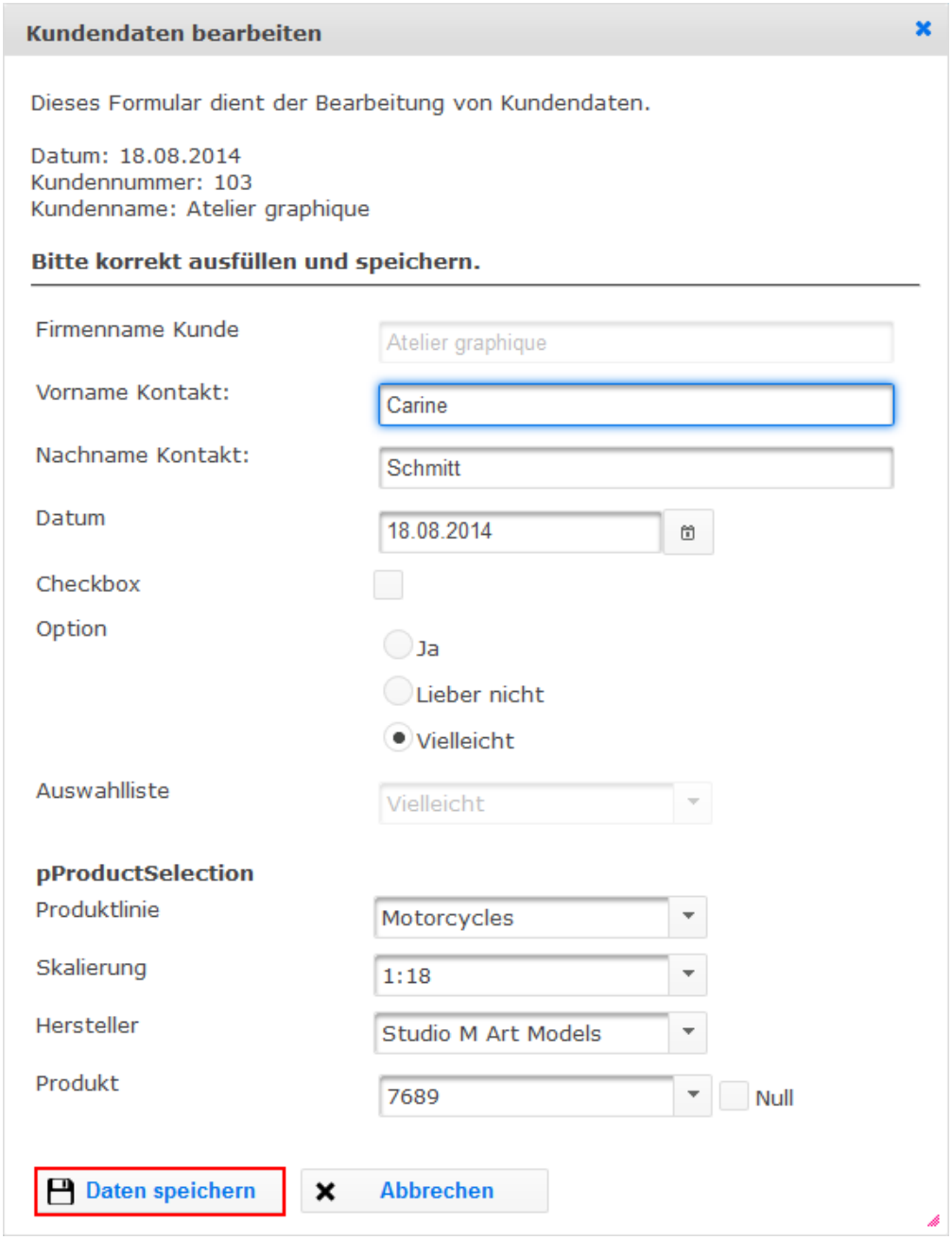
Action-Button beschriften

Jedes Formular hat standardmäßig einen **OK** und einen **Abbrechen**-Button. Der Button für **OK** kann textlich geändert werden, wenn es die Formular-Funktion erfordert. Dazu gehen Sie wie folgt vor:

Legen Sie im Formular-Bericht einen unsichtbaren Parameter mit dem Namen **actionButton** an.

Der Promt-Text ist die Beschriftung für den Action-Button. Der Hilfe-Text erscheint, wenn man sich mit der Maus über dem Action-Button befindet. Hier sollte eine treffende Beschreibung hinterlegt sein, was der Button bewirkt.





Der Parameter **actionButton** kann auch am Hauptbericht der Application verwendet werden.

Buttons mit eigenen JavaScript Aufrufen versehen

Jedes Formular hat standardmäßig einen **OK** und einen **Abbrechen**-Button. Sie haben die Möglichkeit bei Interaktion mit diesen Buttons eigene JavaScript Aufrufe auszuführen. Dazu gehen Sie wie folgt vor:

Legen Sie im Formular-Bericht einen unsichtbaren Parameter mit dem Namen **actionButtonScript** bzw. **cancelButtonScript** an.

Der Promt-Text ist der entsprechende Event des Buttons, an dem der JavaScript Aufruf ausgelöst wird. Die möglichen Werte sind:

Event	Beschreibung
complete	Löst aus, nachdem der Formularbericht gestartet wurde. (Standardwert, wenn nicht angegeben)
click	Löst aus, sobald der Button geklickt wird.
success	Löst aus, nachdem der Formularbericht erfolgreich beendet wurde wurde. Löst nicht bei fehlerhafter Ausführung aus.

Der am Parameter hinterlegte Default-Value ist der JavaScript Code der bei Auslösen des Events aufgerufen werden soll.

> Der Parameter **actionButtonScript** kann auch am Hauptbericht der Application verwendet werden.

JavaScript Callback zum Aufruf nach der Formularausführung definieren

servBIRD kann nach der Formularausführung eine JavaScript Callback Funktion aufrufen, mittels derer Sie auf gewünschte Weise auf die Formularausführung reagieren können. Dazu gehen Sie wie folgt vor:

Legen Sie im Formular-Bericht einen unsichtbaren Parameter mit dem Namen **formSuccessfulCallback** bzw. **formFailedCallback** an.

Der am Parameter hinterlegte Default-Value ist der JavaScript Code der nach erfolgreicher Formularausführung (formSuccessfulCallback) bzw. fehlerhafter Formularausführung (formFailedCallback) ausgeführt wird.

Darstellung des Formulars beeinflussen

Sie können die Darstellung von Formularen mittels css beeinflussen. Dazu gehen Sie wie folgt vor:

Legen Sie im Formular-Bericht einen unsichtbaren Parameter mit dem Namen **formStyles** an.

Als Promt-Text können CSS Styles hinterlegt werden, die auf das Panel mit der Parametermaske angewendet werden.

Als Help-Text können CSS Styles hinterlegt werden, die auf die Panels mit der Fehlermeldung nach fehlerhafter Formularausführung angewendet werden.

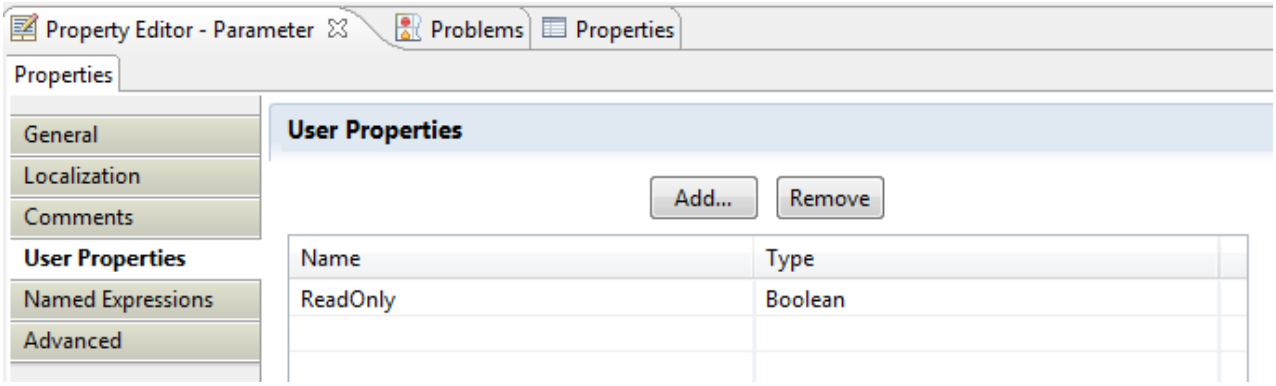
Beispielwerte:

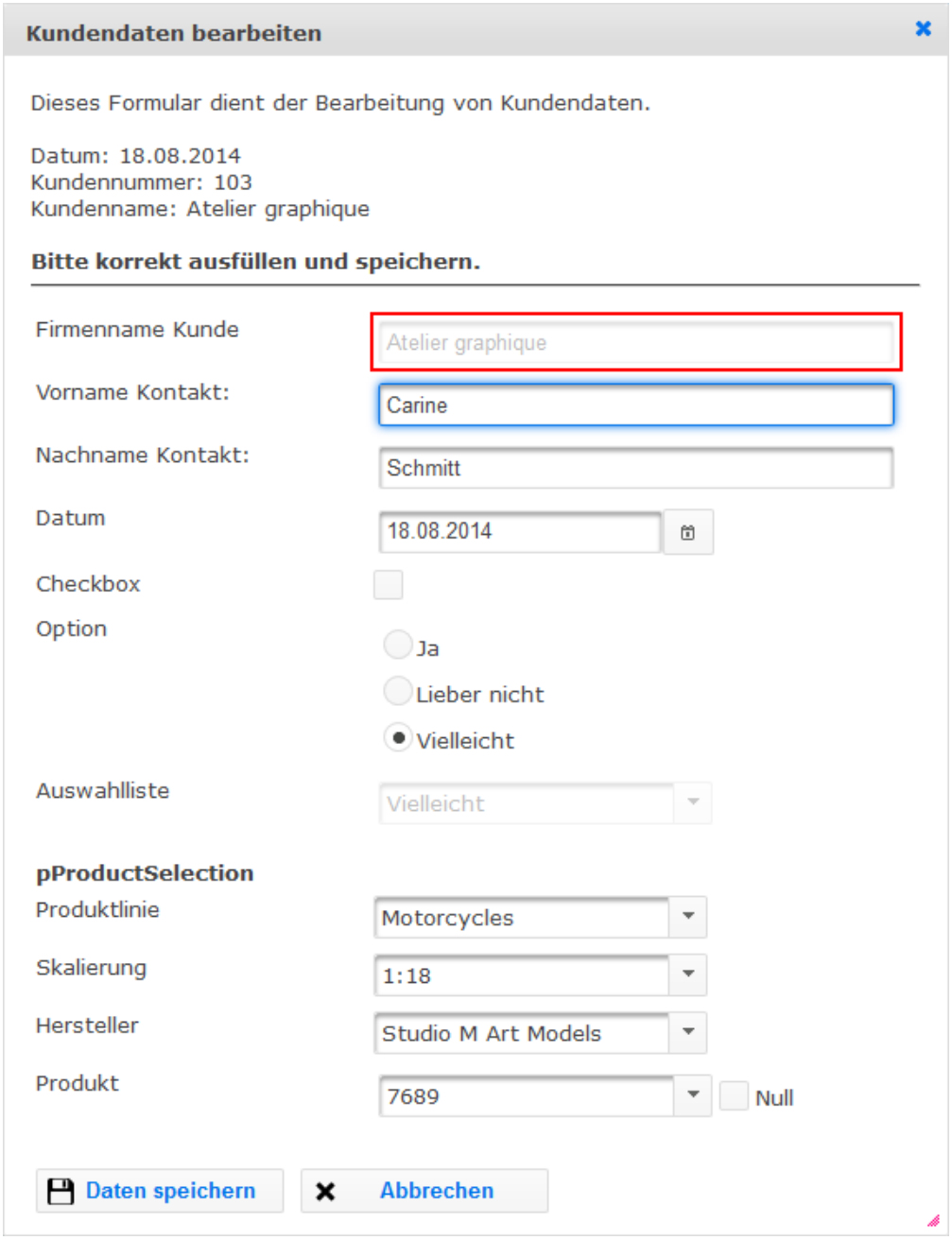
Parametereigenschaft	Wert	Auswirkung
Prompt text	width: 1000px; background: green;	Parametermaske mit Breite von 1000px und grüner Hintergrundfarbe
Help text	width: 750px;	Fehlermeldungspanel mit Breite von 750px

Read-Only für Eingabefelder festlegen

Es ist möglich Formular-Felder als Read Only zu markieren. Nutzer können die Felder dann sehen, aber nicht aktiv verändern.

Um ein Feld entsprechend zu markieren, müssen Sie am Parameter ein User Property **ReadOnly** als **Boolean** erstellen und es mit dem Wert **true** belegen.



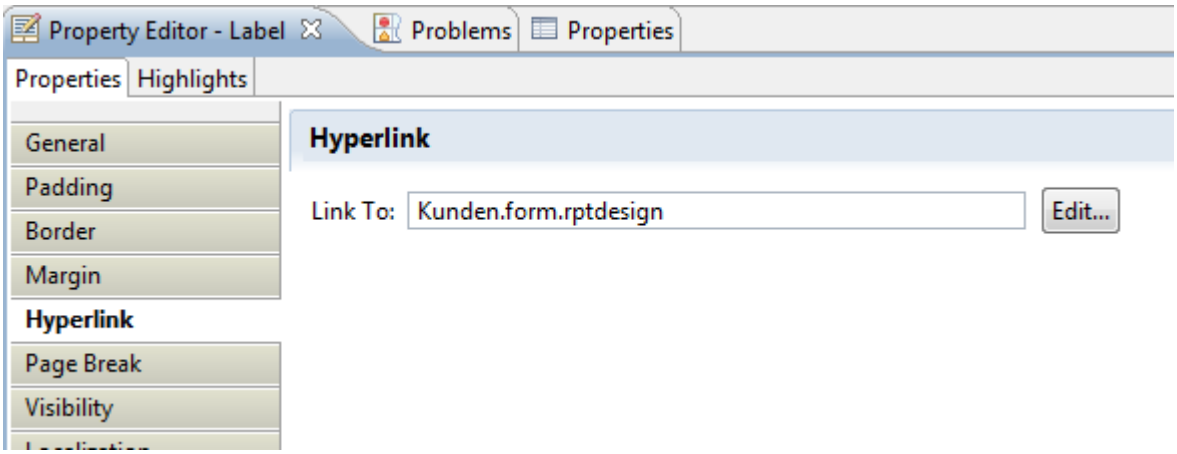


5.1.5 Formular-Aufruf in einen Standard-Bericht integrieren

Um ein Formular im **servBIRD**-Portal aufrufen zu können, muss in einen Standard-BIRT-Bericht ein entsprechender Hyperlink erstellt werden. Dazu gehen Sie wie folgt vor:

Selektieren Sie im Bericht das Element welches ein Formular öffnen soll und wählen Sie dann im Property Editor den Eintrag **Hyperlink** an.

Klicken Sie dort auf **Edit...**

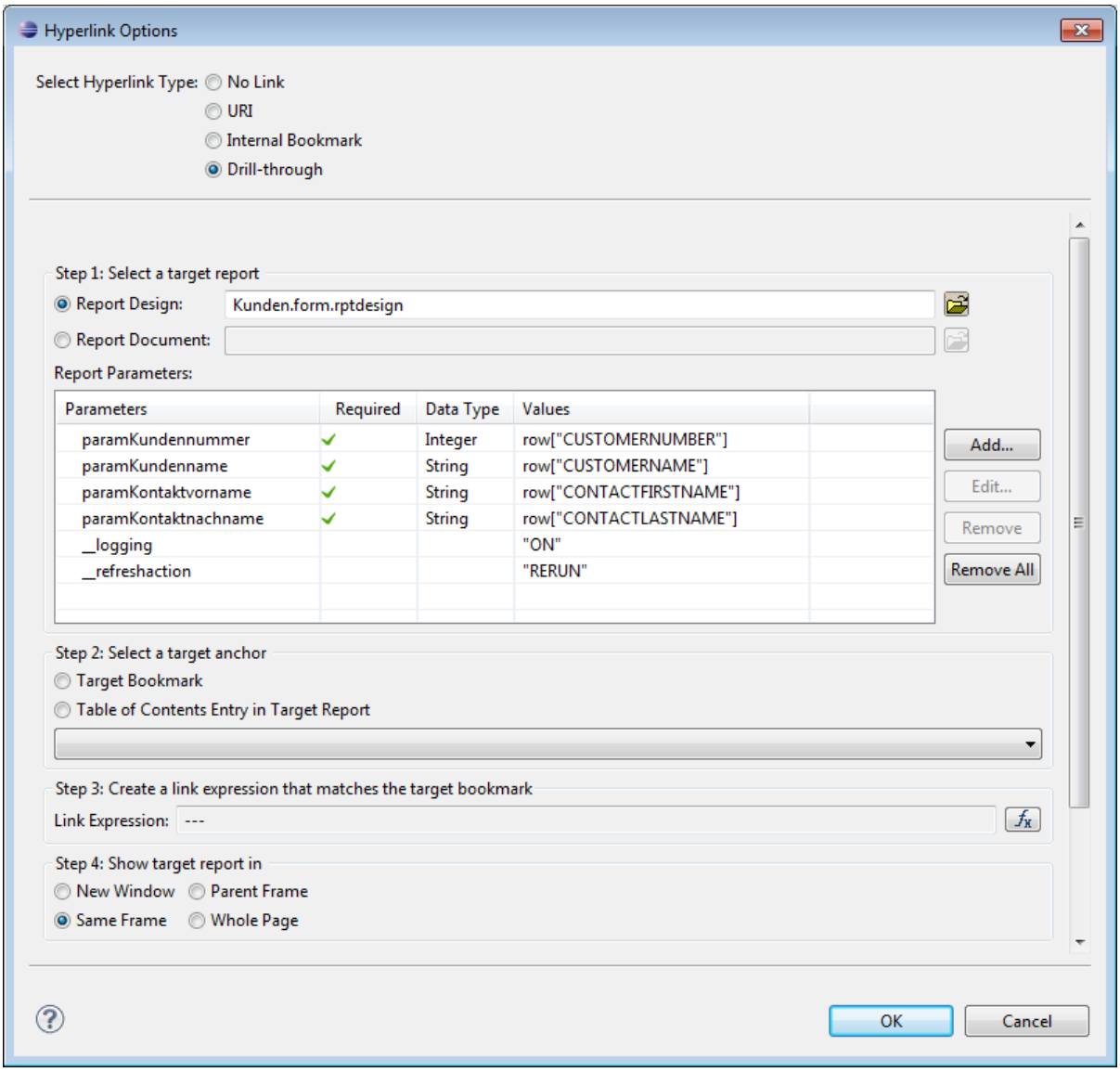


Es öffnet sich der Dialog mit den Hyperlink Optionen.

Selektieren Sie hier als zuerst den Formular-Bericht, der aufgerufen werden soll.

Danach folgen die Parameter. Hier können Sie Werte aus dem Bericht an das Formular weiterreichen.

Achten Sie darauf, dass unter Step 4 Same Window ausgewählt ist.



Es gibt besondere Parameter die das Verhalten des Formulars steuern.

Der Parameter "__refreshaction" mit der Belegung "RERUN" sorgt dafür, dass nach dem Ausführen des Formulars der aufrufende Bericht aktualisiert wird. Standardmäßig muss man dies manuell erfolgen.

Ein weiterer Parameter ist für Entwickler interessant. Mit "__logging" und dem Wert "ON" wird das Logging für das Formular aktiviert. Formularaufrufe werden damit im Serverlog von **servBIRD** protokolliert.

5.2 Cockpits entwickeln

5.2.1 Reportdesign Properties

Folgende Properties sind obligatorisch und müssen im zentralen Cockpit-Designfile gesetzt sein:

Art	Schlüssel	Typ	Wert (Bespiel)	Beschreibung
General Property	title	String	Mein Cockpit	Der Name des Cockpits, mit dem es auch im **servBIRD** angezeigt wird. Dieser muss eindeutig sein, da er als ID des Cockpits verwendet wird.
User Property	SERV.INTERACTIVE.GROUP	String	Meine Gruppe	Der Name der Gruppe des Cockpits. Das Cockpit wird im **servBIRD** innerhalb dieser Gruppe angezeigt werden. Eine Gruppe ist als eine Art Ordner zu verstehen.
User Property	SERV.INTERACTIVE.COLOR	String	#FFAAFF	Der hex-Code für die Farbe des Symbols, mit dem das Cockpit in **servBIRD** angezeigt wird.
User Property	SERV.INTERACTIVE.SORT	Integer	1	Die Sortiernummer für Auflistung dieses Cockpits in seiner Gruppe

Achten Sie darauf, für unterschiedliche Cockpits unterschiedliche Titel zu vergeben. Wird im **servBIRD** wiederholt ein Cockpit mit gleichem Titel hochgeladen, so wird das bereits vorhandene überschrieben. Neue Versionen eines Cockpits sollten demnach also den gleichen Titel tragen, damit alte Versionen automatisch aktualisiert werden.

5.2.2 Icon- und Vorschaubilder

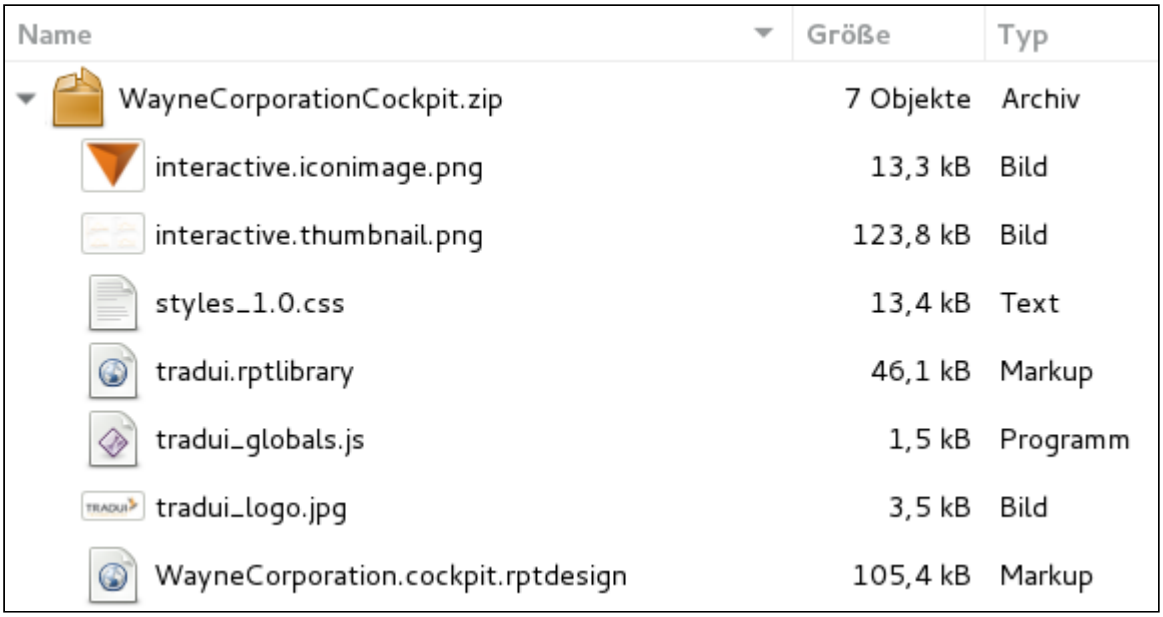
Um den Wiedererkennungswert des Cockpits innerhalb von **servBIRD** zu steigern, kann vom Reportentwickler ein Icon und ein Vorschaubild für das Cockpit gesetzt werden. Hierzu müssen diese als Bilddateien im Cockpit zip-Archiv vorliegen und wie folgt benannt sein:

Art	Erlaubte Dateinamen
Icon	interactive.iconimage.png interactive.iconimage.jpg interactive.iconimage.gif
Vorschaubild	interactive.thumbnail.png interactive.thumbnail.jpg interactive.thumbnail.gif

Pro Cockpit ist jeweils nur ein Icon- und Vorschaubild erlaubt.

5.2.3 Paketierung

Damit das Cockpit in **servBIRD** importiert werden kann, muss es bestimmte Anforderungen an Struktur und Aufbau erfüllen. Zum einen müssen spezielle User Properties im Hauptreport gesetzt sein und dessen Dateiname auf *.cockpit.rptdesign* enden. Zum anderen müssen die verwendeten Ressourcen und alle Designfiles des Cockpits als zip-Archiv komprimiert werden. Im Folgenden ist der schematische Aufbau eines Cockpit zip-Archivs dargestellt.



> Alle Inhalte des zip-Archivs, die kein Reportdesign oder spezielles Thumbnail/Iconimage Bild sind, werden von servBIRD als Reportressourcen interpretiert und im **servBIRD** Repository im Ordner *resources* abgelegt. Das gilt selbstverständlich auch für Ressourcen, die in Unterordnern organisiert sind. Sollten sich Ressourcenverzeichnis von **servBIRD** bereits gleichnamige Dateien befinden, so werden diese überschrieben.

5.3 Dashboards entwickeln (für dashBIRD)

In diesem Abschnitt wird erklärt wie Sie sogenannte Dashboard-Reports zur Nutzung mit dem Modul **dashBIRD** erstellen. Dashboard-Reports sind spezielle BIRT-Reports die unter **servBIRD** Dashboard-Funktionalitäten ermöglichen.

Für die vollständige Dokumentation und den Entwicklungsleitfaden für dashBIRD gehen Sie bitte auf diesen Bereich.

5.3.1 Reportdesign Properties

Folgende Properties sind obligatorisch und müssen im zentralen Dashboard-Designfile gesetzt sein:

Art	Schlüssel	Typ	Wert (Bespiel)	Beschreibung
General Property	title	String	Mein Dashboard	Der Name des Dashboards, mit dem es auch im **servBIRD** angezeigt wird. Dieser muss eindeutig sein, da er als ID des Dashboards verwendet wird.
User Property	SERV.INTERACTIVE.GROUP	String	Meine Gruppe	Der Name der Gruppe des Dashboards. Das Dashboard wird im **servBIRD** innerhalb dieser Gruppe angezeigt werden. Eine Gruppe ist als eine Art Ordner zu verstehen.
User Property	SERV.INTERACTIVE.COLOR	String	#FFAAFF	Der hex-Code für die Farbe des Symbols, mit dem das Dashboard in **servBIRD** angezeigt wird.
User Property	SERV.INTERACTIVE.SORT	Integer	1	Die Sortiernummer für Auflistung dieses Dashboards in seiner Gruppe

Achten Sie darauf, für unterschiedliche Dashboards unterschiedliche Titel zu vergeben. Wird im **servBIRD** wiederholt ein Dashboard mit gleichem Titel hochgeladen, so wird das bereits vorhandene überschrieben. Neue Versionen eines Dashboards sollten demnach also den gleichen Titel tragen, damit alte Versionen automatisch aktualisiert werden.

5.3.2 Icon- und Vorschaubilder

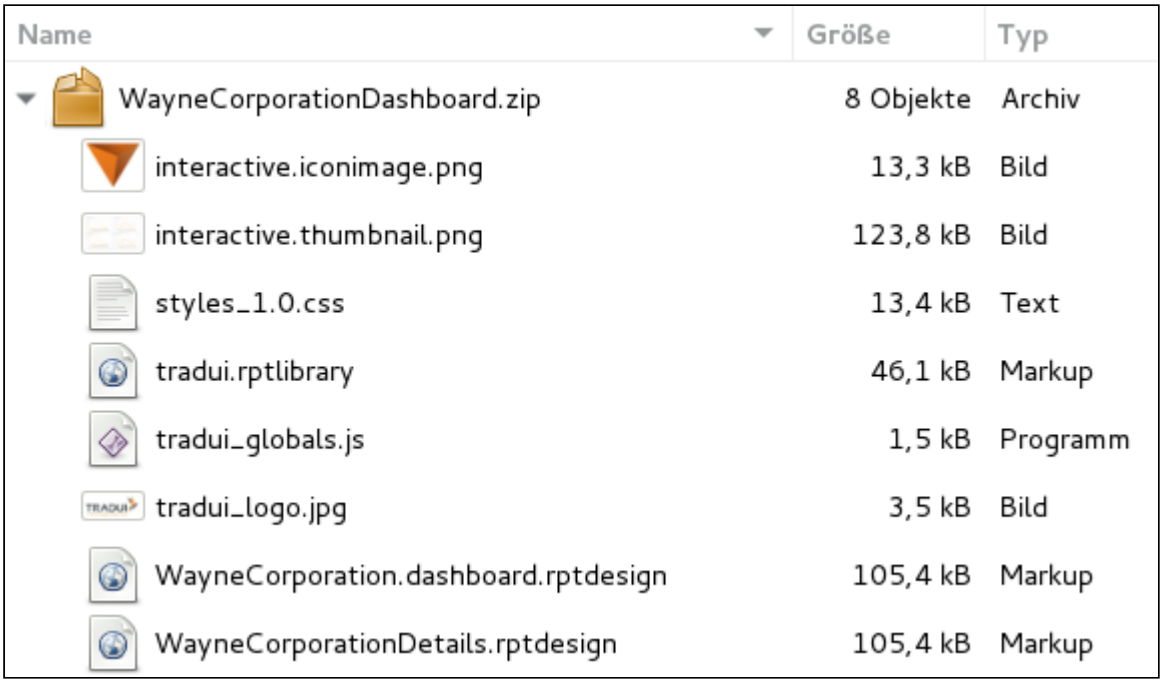
Um den Wiedererkennungswert des Dashboards innerhalb von **servBIRD** zu steigern, kann vom Reportentwickler ein Icon und ein Vorschaubild für das Dashboard gesetzt werden. Hierzu müssen diese als Bilddateien im Dashboard zip-Archiv vorliegen und wie folgt benannt sein:

Art	Erlaubte Dateinamen
Icon	interactive.iconimage.png interactive.iconimage.jpg interactive.iconimage.gif
Vorschaubild	interactive.thumbnail.png interactive.thumbnail.jpg interactive.thumbnail.gif

Pro Dashboard ist jeweils nur ein Icon- und Vorschaubild erlaubt.

5.3.3 Paketierung

Damit das Dashboard in **servBIRD** importiert werden kann, muss es bestimmte Anforderungen an Struktur und Aufbau erfüllen. Zum einen müssen spezielle User Properties im Hauptreport gesetzt sein und dessen Dateiname auf *.dashboard.rptdesign* enden. Zum anderen müssen die verwendeten Ressourcen und alle Designfiles des Dashboards als zip-Archiv komprimiert werden. Im Folgenden ist der schematische Aufbau eines Dashboard zip-Archivs dargestellt.



> Alle Inhalte des zip-Archivs, die kein Reportdesign oder spezielles Thumbnail/Iconimage Bild sind, werden von servBIRD als Reportressourcen interpretiert und im **servBIRD** Repository im Ordner *resources* abgelegt. Das gilt selbstverständlich auch für Ressourcen, die in Unterordnern organisiert sind. Sollten sich Ressourcenverzeichnis von **servBIRD** bereits gleichnamige Dateien befinden, so werden diese überschrieben.

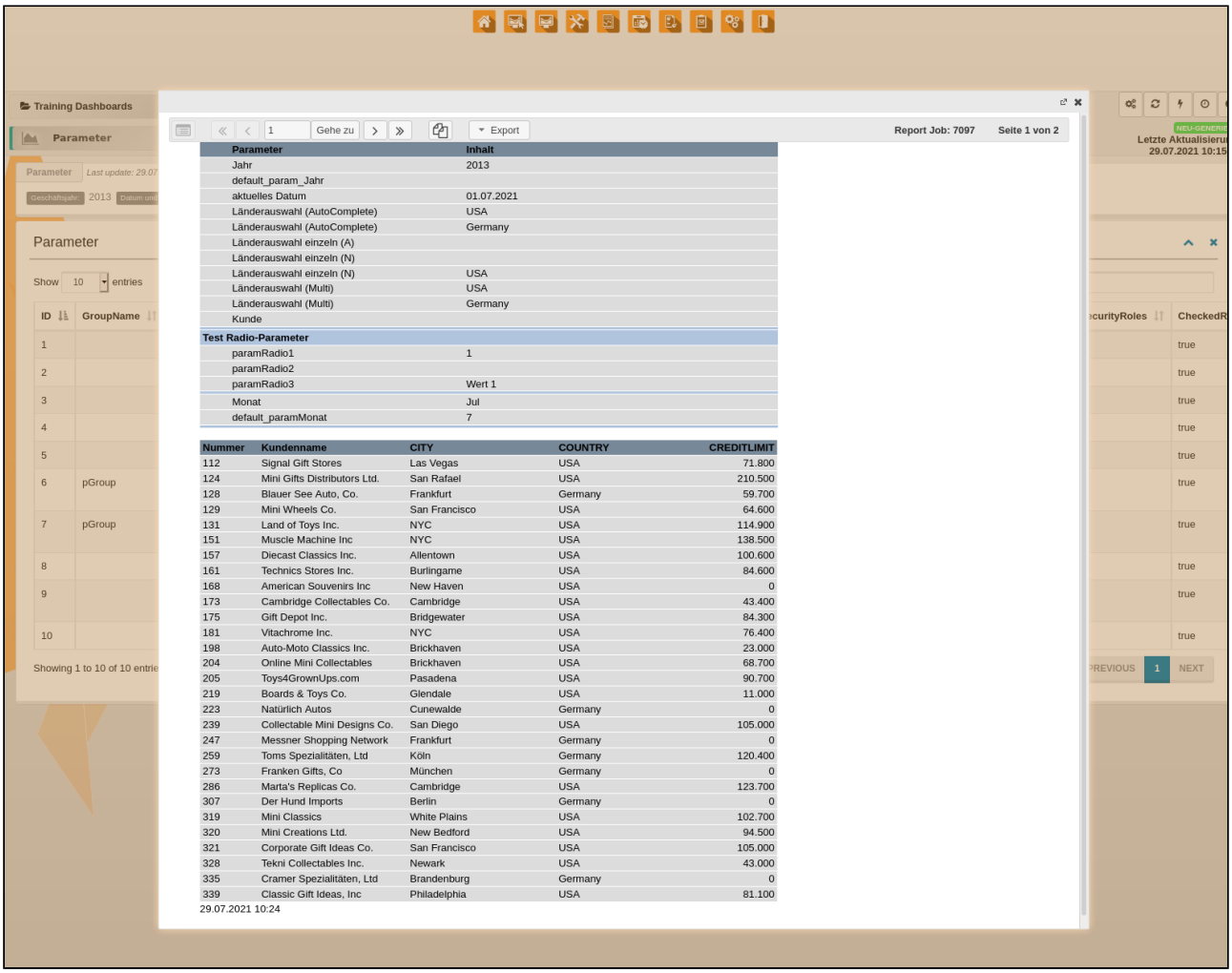
.

5.4 Erweiterte Script-Funktionalitäten

In BIRD Interactives Inhalten (BIRD Applications bzw. Dashboards) ist es möglich, mittels JavaScript Aufrufen zusätzliche Funktionalitäten auszulösen bzw. zu verwenden.

5.4.1 Direktausführung eines Reports im Overlay Viewer

Aus einem BIRD Interactive heraus kann ein Bericht direkt ausgeführt und in einem Overlay Viewer angezeigt werden.



runJobInViewer

```
runJobInViewer([
{name:'--__report', value:'\/categoryfolder\/filename.rptdesign'},      // relativer Pfad zum Report Design unterhalb von "repository/reports/" (Pflichtparameter)
{name:'--__viewerSize', value:'FULLSCREEN'},                           // Viewer in voller Größe öffnen (optional)
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

5.4.2 Direktausführung eines Reports in einem neuen Browsertab

Aus einem BIRD Interactive heraus kann ein Bericht direkt ausgeführt und in einem neuen Browsertab angezeigt werden.

runJobInNewTab

```
runJobInNewTab([
{name:'--__report', value:'\/categoryfolder\/filename.rptdesign'},      // relativer
Pfad zum Report Design unterhalb von "repository/reports/" (Pflichtparameter)
{name:'--__outputFormat', value:'PDF'},                                 // das zu
verwendende Ausgabeformat für den Bericht (Pflichtparameter)
{name:'--__nativeFile', value:'nativeoutputfilename.xlsx'},             // die durch
den OutputManager erzeugte native Excel-Datei, die im neuen Tab bereitgestellt werden
soll (optional)
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

5.4.3 Direktausführung eines Reports im Hintergrund

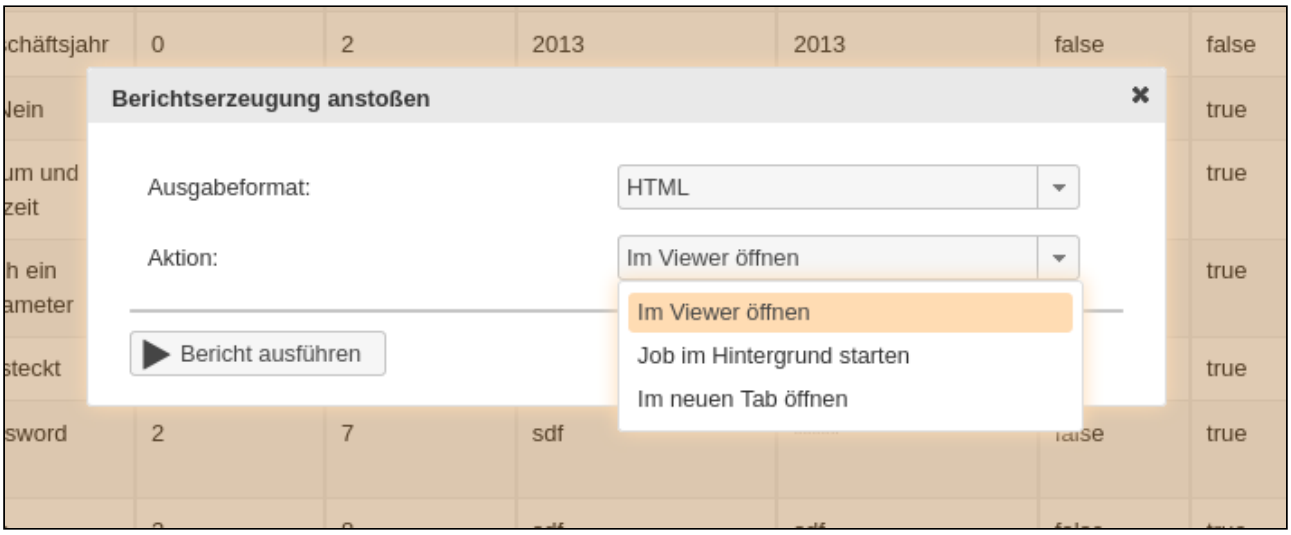
Aus einem BIRD Interactive heraus kann ein Bericht direkt im Hintergrund ausgeführt werden. Das dadurch erzeugte Ausgabedokument kann nach Fertigstellung im Bereich "Fertige Berichte" eingesehen werden.

runJobInBackground

```
runJobInBackground([
{name:'--__report', value:'\/categoryfolder\/filename.rptdesign'},      // relativer
Pfad zum Report Design unterhalb von "repository/reports/" (Pflichtparameter)
{name:'--__outputFormat', value:'PDF'},                                 // das zu
verwendende Ausgabeformat für den Bericht (Pflichtparameter)
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

5.4.4 Auswahldialog zur Ausführung eines Reports anzeigen (JobCreator)

Aus einem BIRD Interactive heraus kann ein Dialog angezeigt werden, der die Ausführung eines bestimmten Berichts erlaubt. Der Nutzer kann zwischen unterschiedlichen Ausgabeformaten und Ausführungsarten (im Hintergrund / im Viewer / im neuen Tab) wählen.



openJobCreatorDialog

```
openJobCreatorDialog([
{name:'--__report', value:'\/categoryfolder\/filename'},        // relativer Pfad zum
Report Design unterhalb von "repository/reports/" (Pflichtparameter)
{name:'--__outputFormats', value:'PDF,HTML'},                  // kommaseparierte
Liste von zur Auswahl stehenden Ausgabeformaten (optional; standard: HTML, PDF, XLSX)
{name:'--__viewerSize', value:'FULLSCREEN'},                   // Viewer in voller
Größe öffnen (optional)
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

5.4.5 Formularausführung anstoßen

Während in Applications und Cockpits üblicherweise zur Ausführung von Formularberichten die BIRT eigene Drill-Through-Mechanik via Hyperlinks verwendet wird, so kann das Öffnen eines Formulars auch direkt via JavaScript erfolgen. Auf diese Weise ist es möglich auch in Dashboards die Formularmechaniken von **servBIRD** zu verwenden.

openReportForm

```
openReportForm([
{name:'--__report', value:'\/groupfolder\/dashboardfolder\/filename.form.rptdesign'},
/ relativer Pfad zum Report Design des Formulars unterhalb von "repository/
dashboards/" (Pflichtparameter)
{name:'--__run', value:'true'},                                    // Die Anzeige der
Formularparametermaske überspringen und das Formular direkt ausführen
{name:'--__quiet', value:'true'},                                  // Bei
Direktausführung das Formular komplett unsichtbar Hintergrund ausführen (Anmerkung:
Verwendung von JavaScript Callback Funktionalitäten des Formulars nicht möglich)
{name:'--__closeonerror', value:'true'},                           // Den Formulardialog
automatisch schließen, wenn bei der Formularausführung ein Fehler aufgetreten ist
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

5.4.6 BIRD Application öffnen

Aus einem BIRD Interactive heraus kann direkt eine bestimmte BIRD Application aufgerufen werden. Die Application kann via ID oder via Pfad definiert werden.

openApplication (via Pfad)

```
openApplication([
{name:'--__reportfolder', value:'\/groupfolder\/applicationfolder'},    // relativer
Pfad zum Verzeichnis der Application unterhalb von "repository/
applications/" (Pflichtparameter)
{name:'--__reportname', value:'filename.app.rptdesign'},               // Dateiname
des Application Report Designs (Pflichtparameter)
{name:'--__run', value:'true'}                                         // Die
Anzeige der Parametermaske überspringen und die Application direkt ausführen
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

openApplication (via Application ID)

```
openApplication([
{name:'--__aid', value:'42'},                                          // Die ID der
aufzurufenden Application (Pflichtparameter)
{name:'--__run', value:'true'}                                         // Die
Anzeige der Parametermaske überspringen und die Application direkt ausführen
{name:'--paramDateStart', value:'2019-11-13 00:00:00.000'},
{name:'--paramZeitraum', value:'week'},
{name:'--paramDateEnde', value:'2019-11-21 00:00:00.000'}]);
```

6 Postprozessoren über den Report Context erzeugen

6.1 Dateiupload über einen Report auslösen und steuern

> **Information**
>
> Diese Funktionalität steht ab **servBIRD** Version 3.18.9 zur Verfügung.

> **Hinweis**
>
> Voraussetzung zur Nutzung dieses Features ist, dass in **servBIRD** ein funktionierender Dateiserver eingerichtet wurde.

Über einen Bericht kann ein Dateiupload Postprozesser erzeugt und gesteuert werden.

Hierzu muss im Bericht eine HashMap<String,String> mit den benötigten Einstellungen erzeugt und im Report Context platziert werden. Um mehrere Dateiuploads auf unterschiedlichen Servern durchzuführen können mehrere HashMaps zu einer Liste hinzugefügt und anschließend im Report Context platziert werden.

Folgende Schlüssel werden hierfür benötigt:

Schlüssel	Information
UPLOAD_SERVER_TECH_NAME	Der technische Name des zu verwendenden Upload Servers
UPLOAD_REMOTE_PATH	Der relative Pfad innerhalb dessen die Datei(en) abgelegt werden sollen
UPLOAD_FILE_FILTER	Ein Filterstring mit dem nur bestimmte Dateitypen hochgeladen werden können

Beispielwerte für den Schlüssel "UPLOAD_FILE_FILTER":

Datei Filter	Auswirkung
pdf	berücksichtigt alle Dateien mit der Endung ".pdf"
pdf,doc	berücksichtigt alle Dateien mit der Endung ".pdf" und ".doc"
*	berücksichtigt alle Dateien

Beispiel 1: Upload von PDF und XLSX Ausgabedateien auf einen Upload Server

initialize() Methode im Bericht

```
1  fileUploadMap = new java.util.HashMap();
2
3  fileUploadMap.put("UPLOAD_SERVER_TECH_NAME", "f2a8211e");
4  fileUploadMap.put("UPLOAD_REMOTE_PATH","unterverzeichnis1/
   unterverzeichnis2");
5  fileUploadMap.put("UPLOAD_FILE_FILTER","pdf,xlsx");
6
7  reportContext.getAppContext().put("TT.SRVBRD.POSTPRC.FILE",
   fileUploadMap);
```

Beispiel 2: Upload von PDF Dateien auf zwei unterschiedliche Upload Server

initialize() Methode im Bericht

```
1  fileUploadMapList = new java.util.ArrayList();
2
3  fileUploadMap1 = new java.util.HashMap();
4  fileUploadMap1.put("UPLOAD_SERVER_TECH_NAME", "f2a8211e");
5  fileUploadMap1.put("UPLOAD_REMOTE_PATH","unterverzeichnis1/
   unterverzeichnis2");
6  fileUploadMap1.put("UPLOAD_FILE_FILTER","pdf");
7
8  fileUploadMap2 = new java.util.HashMap();
9  fileUploadMap2.put("UPLOAD_SERVER_TECH_NAME", "e03c8bb0");
10 fileUploadMap2.put("UPLOAD_REMOTE_PATH","verzeichnisname");
11 fileUploadMap2.put("UPLOAD_FILE_FILTER","pdf");
12
13 fileUploadMapList.add(fileUploadMap1);
14 fileUploadMapList.add(fileUploadMap2);
15
16 reportContext.getAppContext().put("TT.SRVBRD.POSTPRC.FILE",
   fileUploadMapList);
```

6.2 E-Mail Versand über einen Report auslösen und steuern

Hinweis

Vor der Nutzung dieses Features muss ein Mail Server/Relay angebunden sein. Andernfalls wird servBIRD Fehler werfen.

E-Mail Benachrichtigung nach der Berichtsgenerierung kann über den Bericht ausgelöst und gesteuert werden.

Hierzu muss im Bericht eine HashMap<String,String> mit den benötigten Einstellungen erzeugt und im Report Context platziert werden. Um mehrere E-Mails mit unterschiedlichen Inhalten zu verschicken, können mehrere HashMaps zu einer Liste hinzugefügt und anschließend im Report Context platziert werden.

Die Schlüssel für folgende Felder sind **vordefiniert** und sie können nicht durch benutzerdefinierte Schlüssel ersetzt werden:

Schlüssel	Information
MAIL	Empfänger-Adresse ist **Pflichtfeld** für ein E-Mail Post Prozessor, falls keinen E-Mail Verteiler Namen angegeben.
MAILINGLIST	E-Mail Verteiler Name gilt als **Pflichtfeld** für ein E-Mail Post Prozessor, falls keine Empfänger-Adresse angegeben.
TEMPLATE_TECHNAME	E-Mail Vorlage mit angegebenem technischen Namen wird verwendet falls sie in **servBIRD** vorhanden ist. Falls nicht, dann wird die Standardvorlage verwendet.
SUBJECT	Der Betreff wird in zur Verfügung stehende Vorlage eingefügt. Falls nicht angegeben, dann wird zur Verfügung stehender Vorlagen-Betreff verwendet.
BODY	Der Inhalt wird in zur Verfügung stehende Vorlage beigetragen. Falls nicht angegeben, dann wird nur der Inhalt der zur Verfügung stehenden Vorlage verwendet.
BCC	E-Mail Adresse(n) der BCC Empfänger
CC	E-Mail Adresse(n) der CC Empfänger
FROM	E-Mail Adresse des Absenders
REPLY_TO	E-Mail Antwortadresse(n) die gesetzt werden, wenn Nutzer auf die von servBIRD gesendeten E-Mail antworten
ADD_ATTACHMENTS	Ausgabedateien inkl. Native Dateien werden an Email angehängt falls dieser Schlüssel auf "true" gesetzt ist.

Alle Felder bis auf "MAIL" und "MAILINGLIST" sind optional und werden gegebenenfalls mit Standardwerten gefüllt. Weitere benutzerdefinierte Schlüssel und Werte für in festgelegter Vorlage bestehende Platzhalter müssen im Report Context vorhanden sein, sodass der Inhalt der E-Mail ordentlich dargestellt werden kann. E-Mail relevante Informationen können mittels einer HashMap<String, String> dem ApplicationContext vom Report übergeben werden:

initialize() Methode im Bericht

```
myMailMap = new java.util.HashMap();

//eins der folgenden zwei Felder ist erforderlich
myMailMap.put("MAIL", "recipient@domain.tld");
//myMailMap.put("MAILINGLIST","verteiler1");
//myMailMap.put("MAILINGLIST","verteiler1,verteiler2");

//optional
myMailMap.put("BCC", "recipient1@domain.tld,recipient2@domain.tld");
myMailMap.put("SUBJECT", "Email-Versand");
myMailMap.put("BODY", "E-Mail Inhalt...");
myMailMap.put("FROM", "sender@domain.tld");
myMailMap.put("REPLY_TO", "responsibleuser@domain.tld");
myMailMap.put("TEMPLATE_TECHNAME", "a1b2c3d4");
//benutzerdefinierte Schlüssel
myMailMap.put("CUSTOM_KEY_FIRSTNAME", "Max");
myMailMap.put("CUSTOM_KEY_LASTNAME", "Mustermann");
myMailMap.put("CUSTOM_KEY_GREETING", "Hallo");
myMailMap.put("CUSTOM_KEY_CLIENT", "Fr. Musterfrau");
//attachment
myMailMap.put("ADD_ATTACHMENTS","true");

reportContext.getAppContext().put("TT.SRVBRD.POSTPRC.EMAIL", myMailMap);
```

Beispiel Email-Vorlage

*%{*CUSTOM_KEY_GREETING*} %{*CUSTOM_KEY_CLIENT*},*

Ihr Bericht wurde ausgeführt.

MfG

*%{*CUSTOM_KEY_*FIRSTNAME} %{*CUSTOM_KEY_*LASTNAME}*

Beispiel: Versenden mehrerer E-Mails:

initialize() Methode im Bericht

```
1  mailMapList = new java.util.ArrayList();
2
3  myMailMap1 = new java.util.HashMap();
4  myMailMap1.put("MAIL", "recipient@domain.tld");
5  myMailMap1.put("SUBJECT", "Email-Betreff");
6  myMailMap1.put("BODY", "E-Mail Inhalt...");
7  myMailMap1.put("FROM", "sender@domain.tld");
8  myMailMap1.put("REPLY_TO", "responsibleuser@domain.tld");
9  myMailMap1.put("TEMPLATE_TECHNAME", "a1b2c3d4");
10 myMailMap1.put("ADD_ATTACHMENTS","true");
11
12 myMailMap2 = new java.util.HashMap();
13 myMailMap2.put("MAIL", "recipient2@domain.tld");
14 myMailMap2.put("SUBJECT", "Wichtige Information");
15 myMailMap2.put("BODY", "Anderer E-Mail Inhalt...");
16 myMailMap2.put("FROM", "sender@domain.tld");
17 myMailMap2.put("REPLY_TO", "responsibleuser@domain.tld");
18 myMailMap2.put("TEMPLATE_TECHNAME", "d342f1c3");
19 myMailMap2.put("ADD_ATTACHMENTS","false");
20
21 mailMapList.add(myMailMap1);
22 mailMapList.add(myMailMap2);
23
24 reportContext.getAppContext().put("TT.SRVBRD.POSTPRC.EMAIL", mailMapList);
```

6.3 Freigabebenachrichtigung über den Report Context steuern

Information

Diese Funktionalität steht ab **servBIRD** Version 3.18.9 zur Verfügung.

Über einen Bericht kann ein Freigabe Postprozesser erzeugt werden. Hierbei werden alle anderen Postprozessoren zurückgehalten und nicht sofort ausgeführt. Stattdessen werden ausgewählte Benutzer per E-Mail benachrichtigt und aufgefordert die zurückgehaltenen Postprozessoren freizugeben. Erst danach werden diese ausgeführt.

Um einen Freigabe Postprozesser anzulegen muss im Bericht eine HashMap<String,String> mit den benötigten Einstellungen erzeugt und im Report Context platziert werden.

Folgende Schlüssel stehen hierfür zur Verfügung:

Schlüssel	Information
OWNER	Legt fest, dass der Eigentümer des Report Jobs per Mail benachrichtigt werden soll
RECIPIENT	Der Loginname (oder mehrere Loginnamen als kommaseparierte Liste) eines servBIRD Benutzers, der per Mail benachrichtigt werden soll

Schlüssel	Information
MAILINGLIST	Der Name eines in servBIRD vorhandenen Mailverteilers (oder mehrere Mailverteiler als kommaseparierte Liste) dessen Mitglieder per Mail benachrichtigt werden sollen
MAIL	Die E-Mail-Adresse (oder mehrere E-Mail-Adressen als kommaseparierte Liste) eines servBIRD Benutzers, der per Mail benachrichtigt werden soll

Mindestens einer der oben genannten Schlüssel muss gesetzt sein, damit eine E-Mail-Benachrichtigung versendet werden kann. Zudem müssen alle definierten zu benachrichtigenden Benutzer, Mailverteiler und E-Mail-Adressen in **servBIRD** vorhanden sein. Andernfalls wird der Postprozessor nach dessen Ausführung in den Status "fehlgeschlagen" versetzt und ausschließlich die vorhandenen Benutzer per E-Mail benachrichtigt.

Beispiel:

initialize() Methode im Bericht

```
releaseMap = new java.util.ArrayList();

// Den Eigentümer des Jobs benachrichtigen und um Freigabe bitten
releaseMap.put("OWNER", "true");

// Zusätzlich den servBIRD Benutzer mit dem Loginnamen "max.mustermann"
benachrichtigen und um Freigabe bitten
releaseMap.put("RECIPIENT","max.mustermann");
// Die Angabe mehrere Loginnamen ist auch möglich
// releaseMap.put("RECIPIENT","max.mustermann,moritz.mustermann")

// Zusätzlich den servBIRD Benutzer mit der E-Mail-Adresse
"xyz@example.com" benachrichtigen und um Freigabe bitten
releaseMap.put("MAIL","xyz@example.com");
// Die Angabe mehrerer Mailadressen ist auch möglich
//
releaseMap.put("MAIL","abc@example.com,def@example.com,xyz@example.com");

// Zusätzlich alle Mitglieder des servBIRD Mailverteilers mit dem Namen
"Mailverteiler1" benachrichtigen und um Freigabe bitten
releaseMap.put("MAILINGLIST","Mailverteiler1");
// Die Angabe mehrerer Mailverteiler ist auch möglich
releaseMap.put("MAILINGLIST",
"Mailverteiler1,Mailverteiler2,Mailverteiler3");

reportContext.getAppContext().put("TT.SRVBRD.POSTPRC.RELEASE",
releaseMap);
```

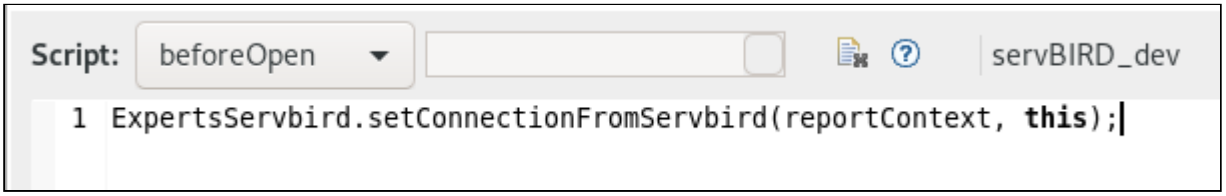
7 Datenbankverbindung aus dem Connection Manager verwenden

Sie können in Ihren Berichten Datenbankverbindungen verwenden, die im **servBIRD** Connection Manager hinterlegt sind.

Voraussetzung ist, dass ...

- im **servBIRD** eine Connection existiert, die als Namen den Namen der Data Source des Reports trägt
- die im **servBIRD** hinterlegte Connection aktiviert ist
- das servBIRD Plugin auf dem Server installiert ist

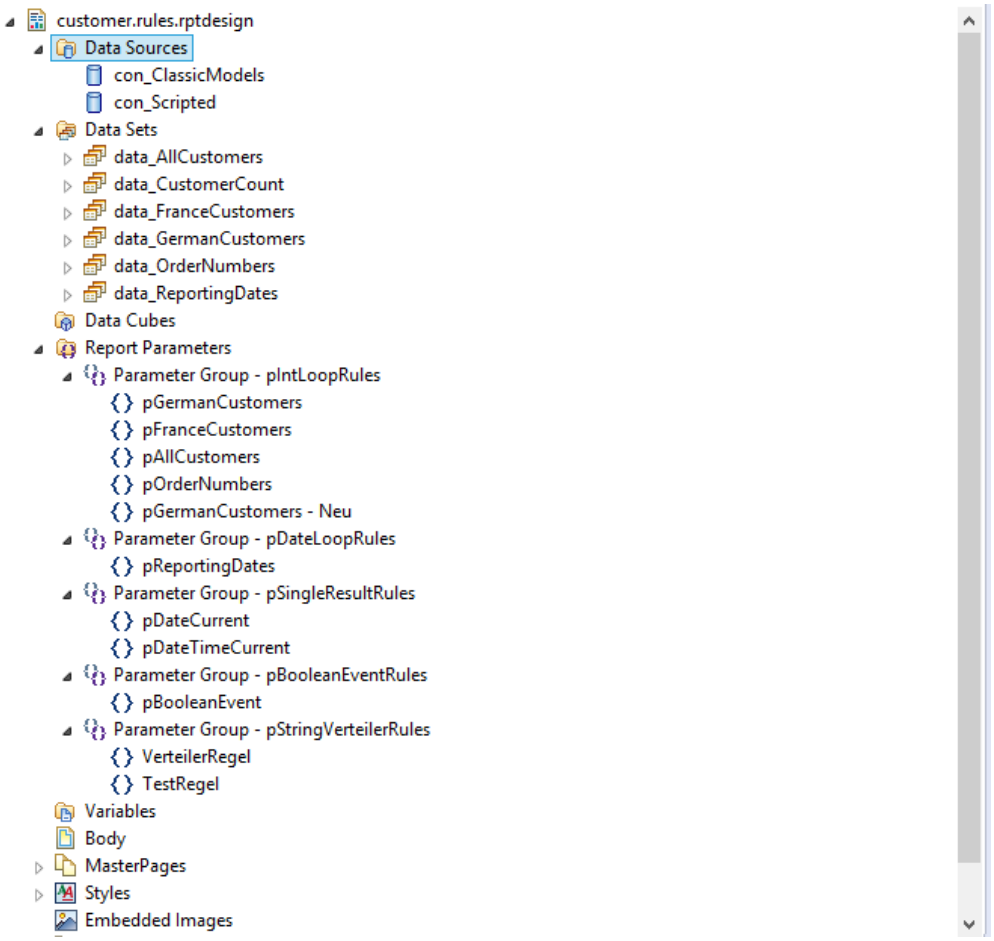
Um eine in **servBIRD** hinterlegte Connection im Report zu verwenden, muss im beforOpen() Script der Data Source die Funktion "setConnectionFromServbird" aufgerufen werden:



Wird dieser Bericht nun innerhalb von **servBIRD** ausgeführt, so werden die in der Data Source angegebenen Datenbankverbindungsdaten mit denen, die durch servBIRD übergeben werden, überschrieben.

8 Parameter Regeln entwickeln

Für die Implementierung eigener Regeln werden die Standard BIRT Techniken in einem rptdesign genutzt, um Parameter zu definieren.
In Berichten mit der Endung ".rules.rptdesign" werden entsprechende Parameter zentral definiert, die dann in **servBIRD** zur Auswahl angeboten werden, wenn ein erweitertes Scheduling definiert werden soll. Das bedeutet, die Regeln werden einmal zentral definiert und bei jeder Zuordnung in einem geplanten Job individuell genutzt.



8.1 Regeln erstellen

Wenn man eine Standardregel definieren möchte, sodass die Regel genau einen Wert eines beliebigen Datentyps zurückliefert, sind folgende Schritte durchzuführen:

Erstellen Sie im ersten Schritt eine Data Source (mit bestehender Datenbank oder Scripted Datasource). Erzeugen Sie nun basierend auf der eben erstellten Data Source ein Data Set, in welchem Sie beliebige Werte abfragen oder selbst per Script erzeugen.

Im nächsten Schritt erzeugen Sie einen neuen Parameter. Der Name des Parameters ist der Name der Regel. Die Regel-Info wird über den Prompttext gefüllt, der Helptext liefert die Beschreibung der Regel.

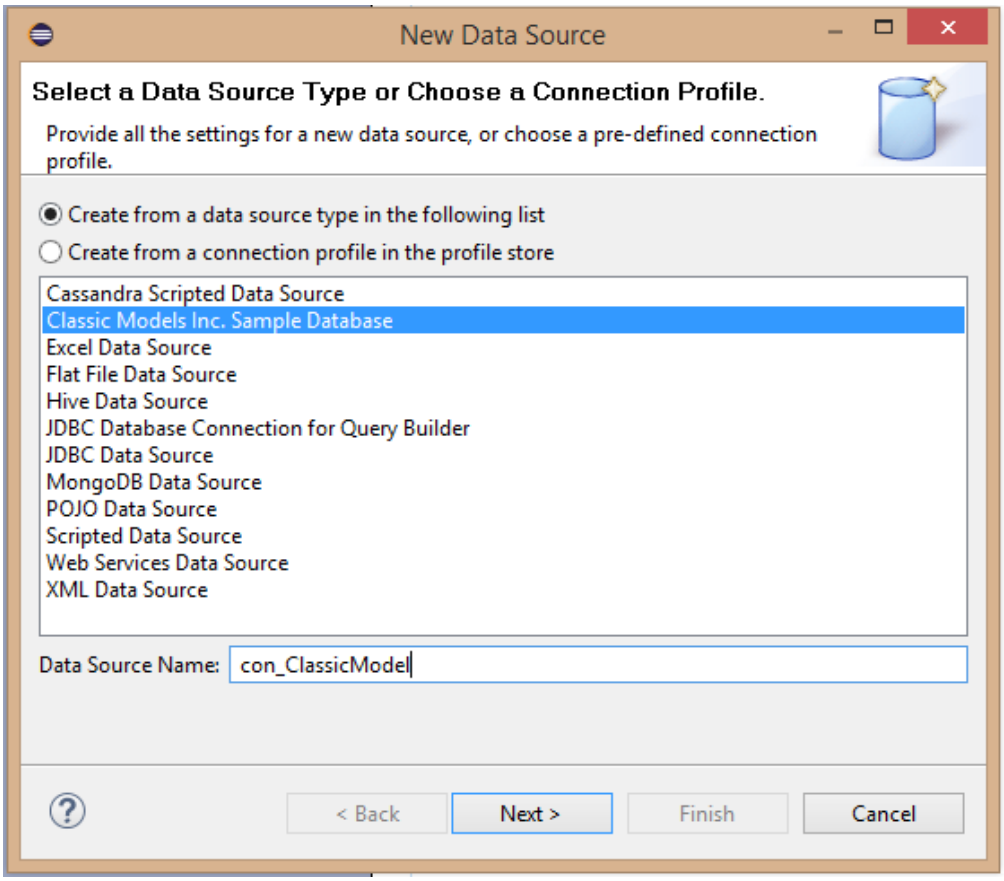
Der Datentyp des Parameters definiert den Rückgabetyp der Regel, d.h. die Regel kann auch später nur auf solche Berichtsparameter mit dem selben Typ angewendet werden. Den Rückgabewert der Regel bestimmen Sie über ein Script oder eine Konstante.

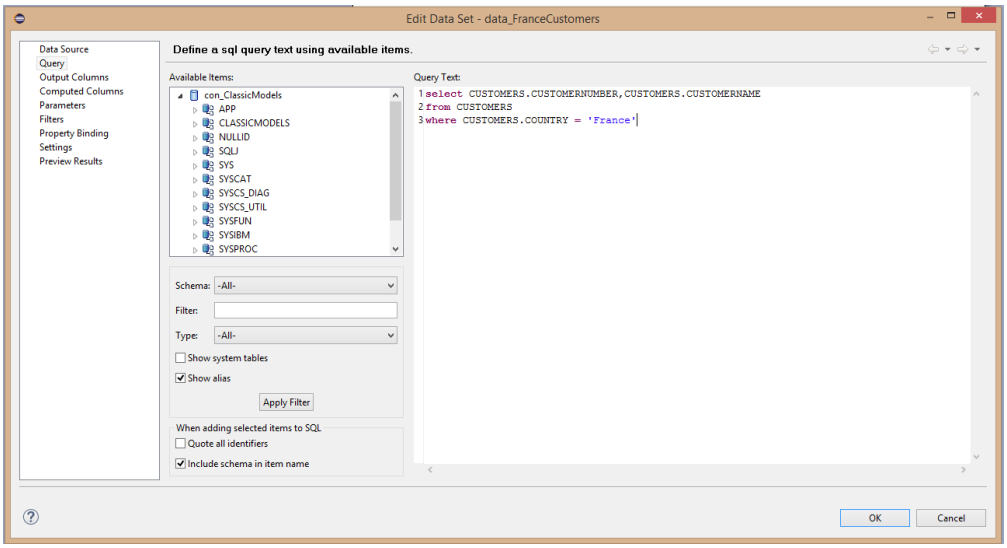
8.1.1 Loop Regeln

Wenn man eine Loop Regel erstellen möchte, ist zu definieren, wie die Mengenverarbeitung im Detail ausgeführt werden soll. Grundlage einer Regel ist das Ergebnis eins Data Sets, das die entsprechende Menge an Werten liefert. Je geliefertem Wert, entsteht ein Job.
Im folgenden Abschnitt wird anhand der Classic Models Datenbank bzw. per Scripted Data Source beschrieben, wie man eine Loop-Regel erstellen kann.

Erstellen Sie im ersten Schritt eine Data Source auf die Classic Models Datenbank. Erzeugen Sie nun ein Data Set mit beispielsweise folgende Abfrage:

```
select CUSTOMERS.CUSTOMERNUMBER, CUSTOMERS.CUSTOMERNAME
from CUSTOMERS
where CUSTOMERS.COUNTRY = 'France'
```

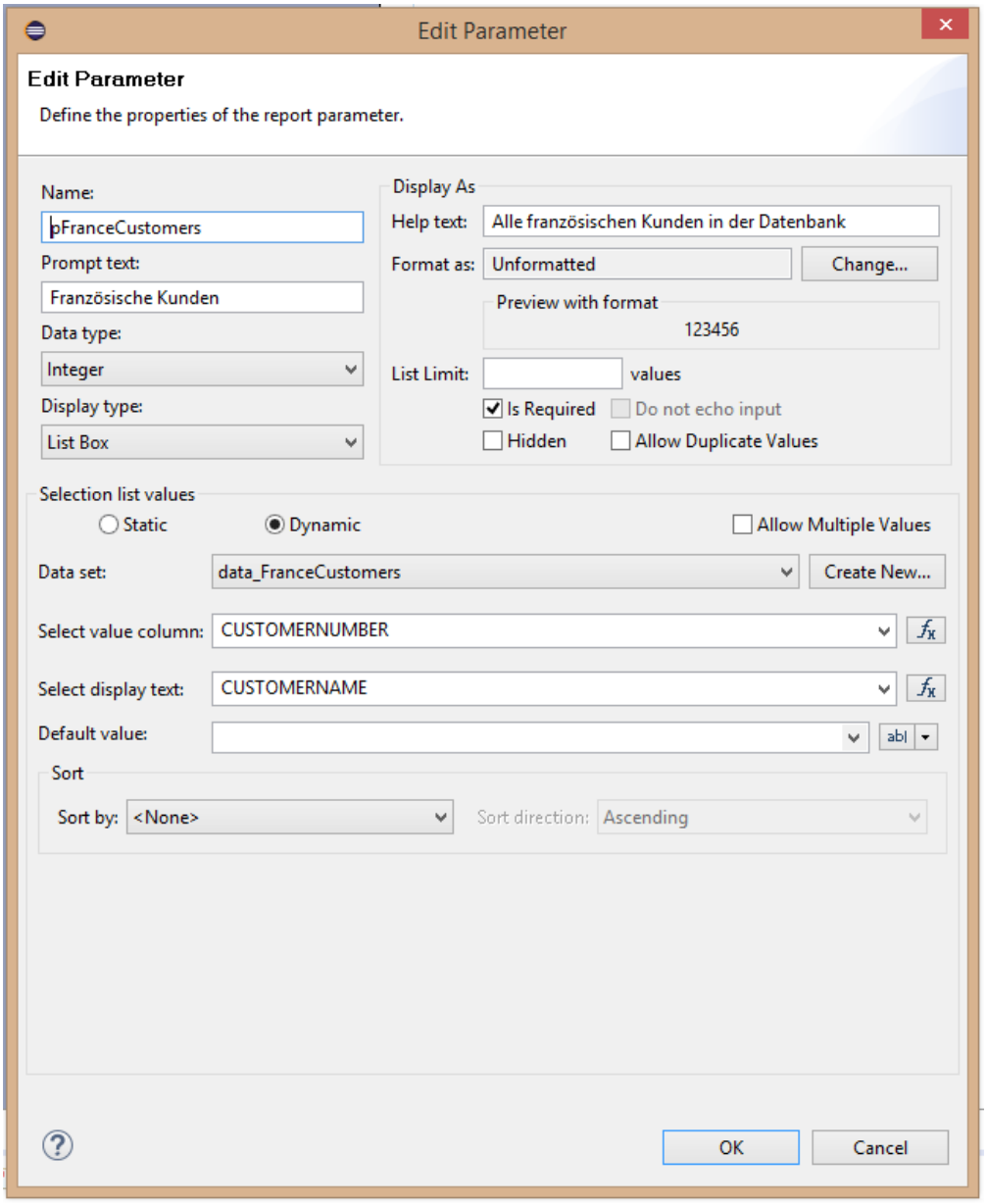




Die Abfrage liefert alle französischen Kundennummern und Kundennamen.

Im nächsten Schritt erzeugen Sie den Regel Parameter. Da in diesem Beispiel eine Loop-Regel erstellt wird, d.h. die Regel liefert mehrere Ergebnisse, wird der Display type zu einer List Box. Da der Parameter mit einem Data Set verknüpft werden soll wählt man im unteren Abschnitt den Radio Button "dynamic".

Wählen Sie nun das eben erstellte Data Set, welches alle französischen Kunden selektiert. Als Value Column, wird nun die Kundennummer ausgewählt (CUSTOMERNUMBER), als Display Text der Name des Kunden (CUSTOMERNAME). Der Datentyp der Spalte CUSTOMERNUMBER ist Integer, d.h. die Regel gibt einen Integer Wert zurück.

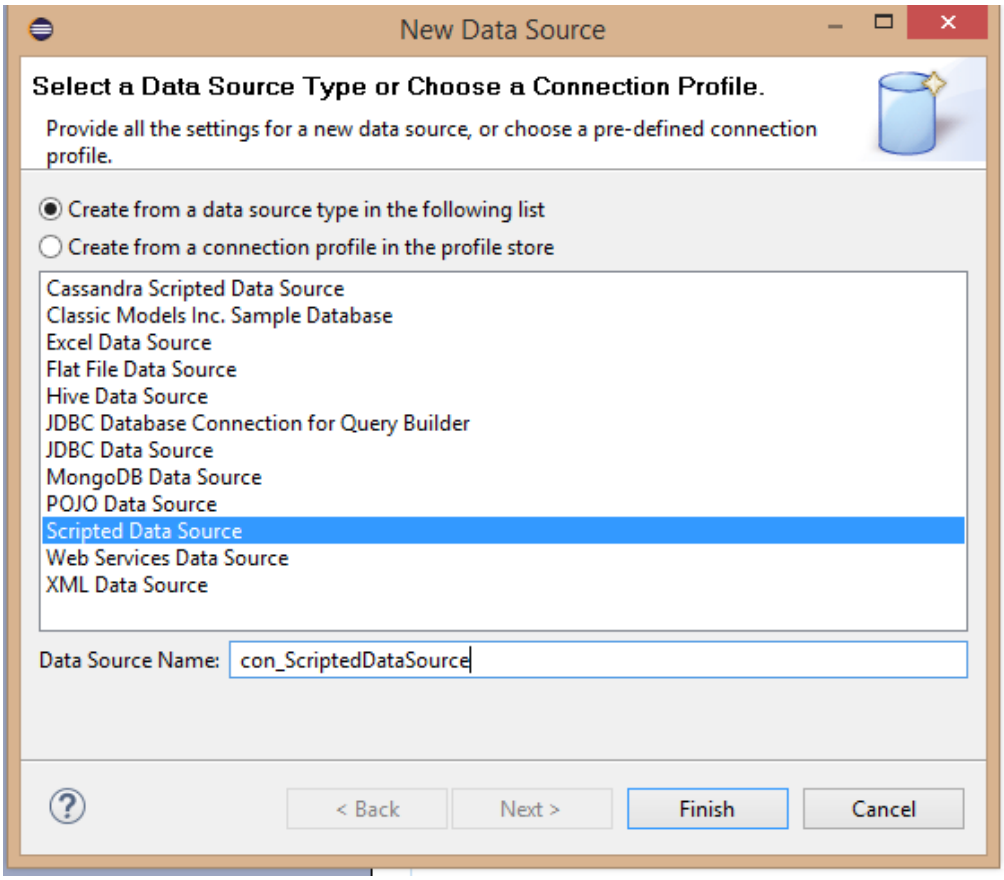


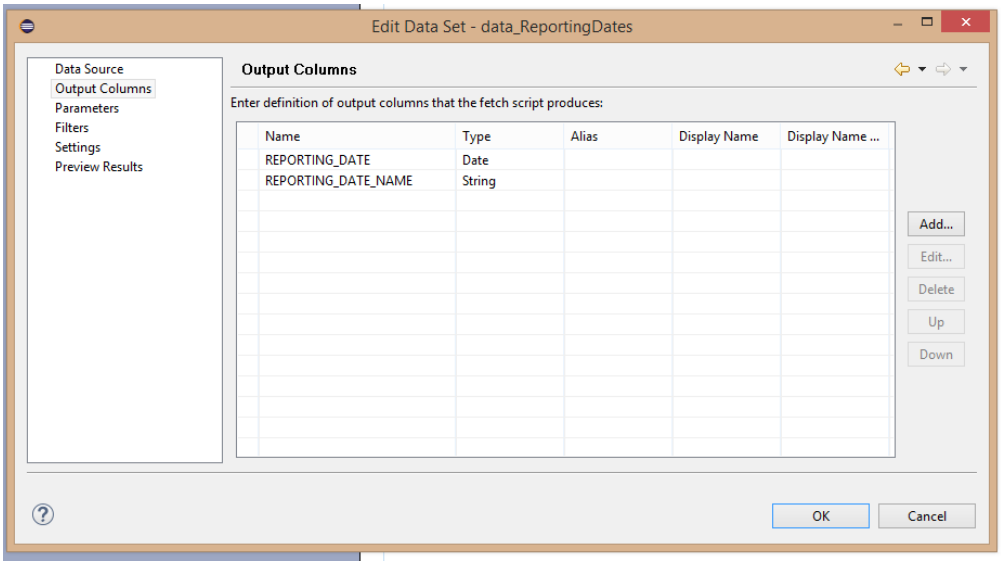
Wenn Sie die erstellte Regel nun im **servBIRD** mit einem Berichts-Parameter verknüpfen, wird beim Ausführen einer Schedule Definition, für jeden französischen Kunden (selektiert über seine Kundennummer) ein Job erstellt.

Es kann Anwendungsfälle geben, in denen Sie Regeln mit selbst definierten bzw. konstanten Rückgabewerten erstellen möchten. Im folgenden Beispiel werden vier konstante Werte vom Typ Date erzeugt.

Erstellen Sie im ersten Schritt eine Scripted Data Source. Erzeugen Sie nun, basierend auf dem eben erstellten Scripted Data Source, ein Scripted Data Set. Öffnen Sie das Data Set und legen die gewünschten Spalten inklusive Datentyp an. In diesem Beispiel sind das:

- REPORTING_DATE Typ: DATE
- REPORTING_DATE_NAME Typ: STRING





Selektieren Sie weiterhin das Scripted Data Set und wechseln auf den Reiter "Script". Nun müssen Sie per Skript die eben erstellten Spalten mit Daten füllen. In diesem Beispiel sollen insgesamt vier Reporting Tage mit je einem festen Datum angelegt werden. Im Script-Bereich "Open" initialisieren Sie eine Zähler-Variable "glDateCount".

ScriptedDataset:open

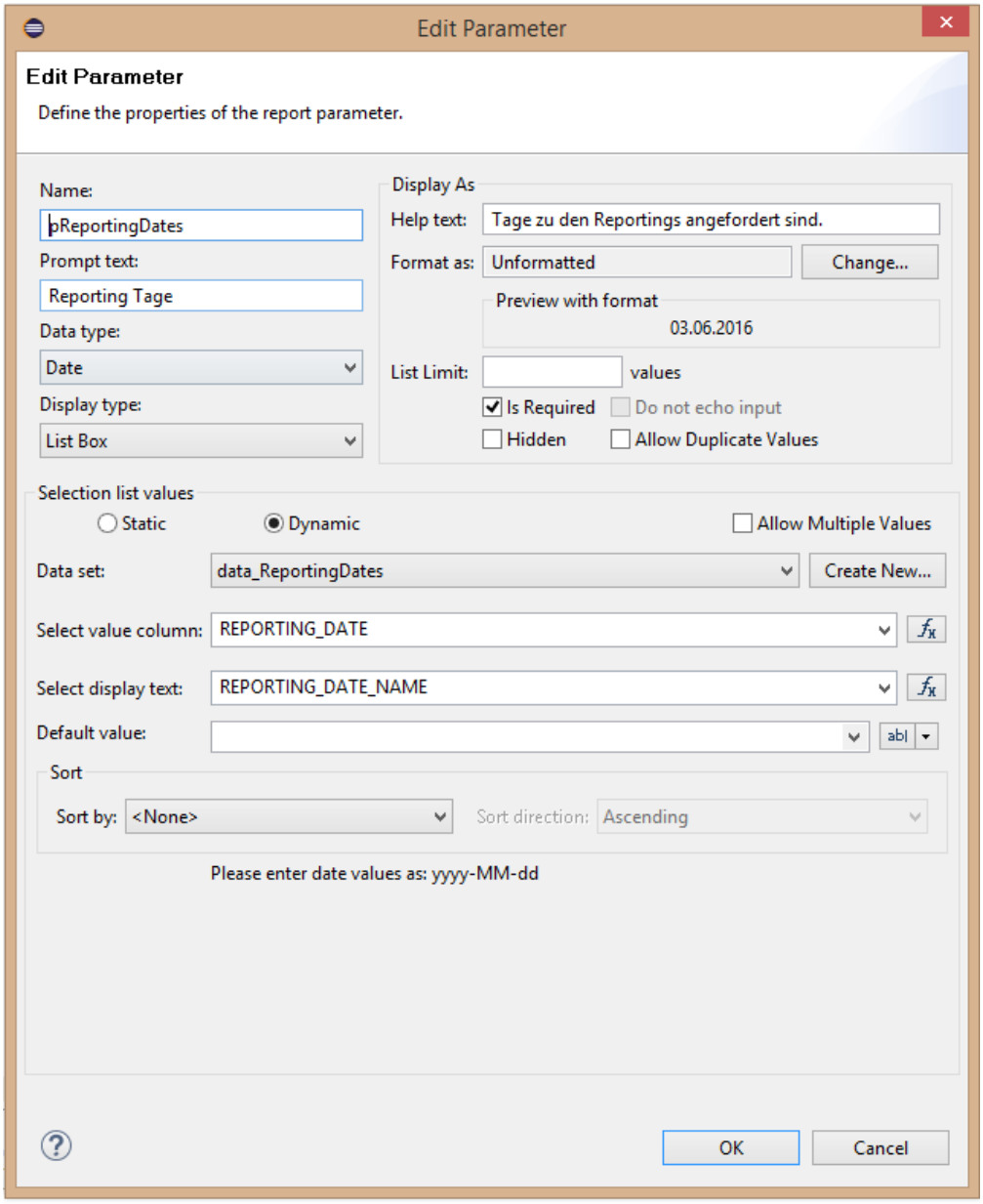
```
glDateCount = 0;
```

Im Script Bereich "Fetch" initialisieren Sie die Daten. Das heißt pro Durchlauf wird ein fixes Datum erstellt. Hat man den vierten Tag erreicht, wird die Initialisierung abgebrochen.

ScriptedDataset:fetch

```
if(glDateCount > 3) return false;
if(glDateCount == 0)
{
    row.REPORTING_DATE = new java.util.Date(100,2,14);
    row.REPORTING_DATE_NAME = "Reporting Tag 1";
}
else if (glDateCount == 1)
{
    row.REPORTING_DATE = new java.util.Date(100,5,18);
    row.REPORTING_DATE_NAME = "Reporting Tag 2";
}
else if (glDateCount == 2)
{
    row.REPORTING_DATE = new java.util.Date(100,8,12);
    row.REPORTING_DATE_NAME = "Reporting Tag 3";
}
else if (glDateCount == 3)
{
    row.REPORTING_DATE = new java.util.Date(100,11,11);
    row.REPORTING_DATE_NAME = "Reporting Tag 4";
}
glDateCount++;
return true;
```

Die Erzeugung des Parameters erfolgt analog, wie im vorherigen Abschnitt beschrieben. Sie verknüpfen das eben erstellte Scripted Data Set mit dem Parameter. Als Value Column, wird nun die Datumsspalte ausgewählt (REPORTING_DATE), als Display Text der Name des Reporting Tages (REPORTING_DATE_NAME). Der Datentyp der Spalte der REPORTING_DATE ist Date, d.h. die Regel gibt einen Date Wert zurück.



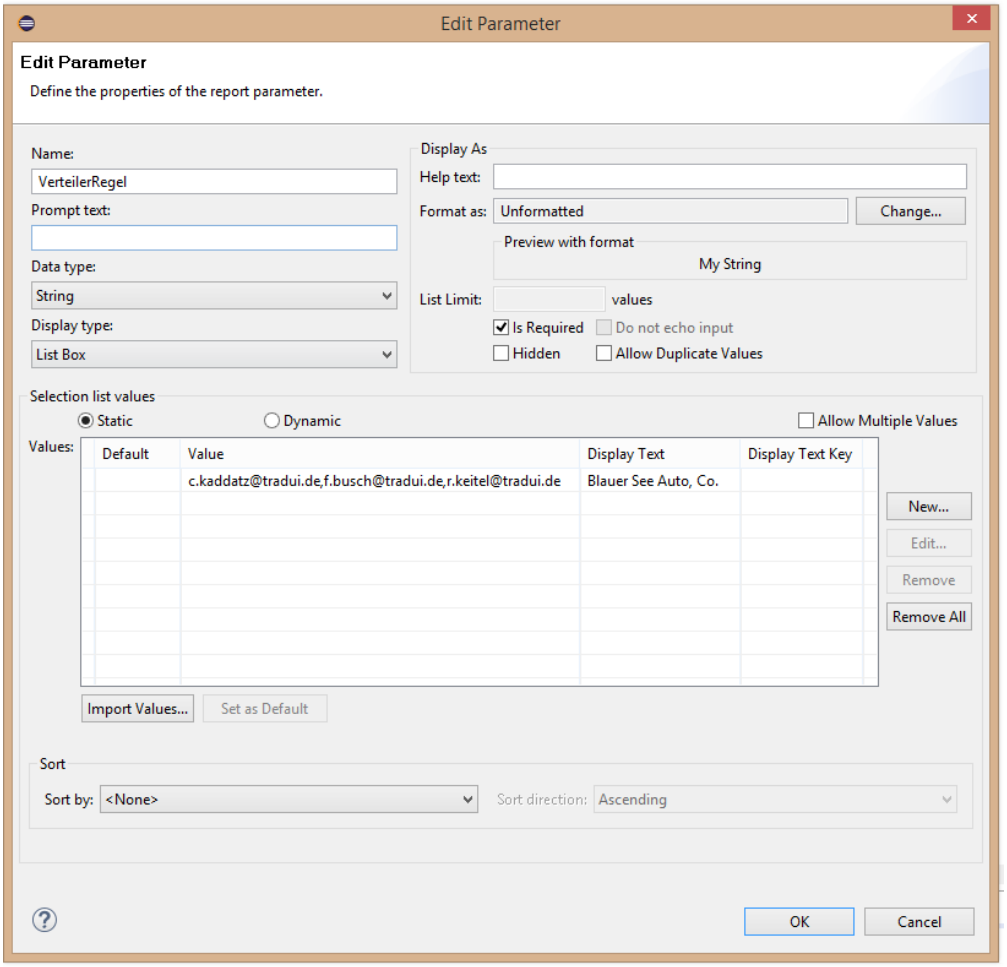
Hinweis

servBIRD evaluiert bei Zeitplanausführung alle Loopregeln gleichzeitig und in unbestimmter Reihenfolge. Das bedeutet, dass einzelne Loopregeln nicht von Ergebnissen aus anderen Loopregeln abhängig sein dürfen. Bitte beachten Sie dies bei der Erstellung und dem Einsatz von Loopregeln.

8.1.2 Verteiler Regeln

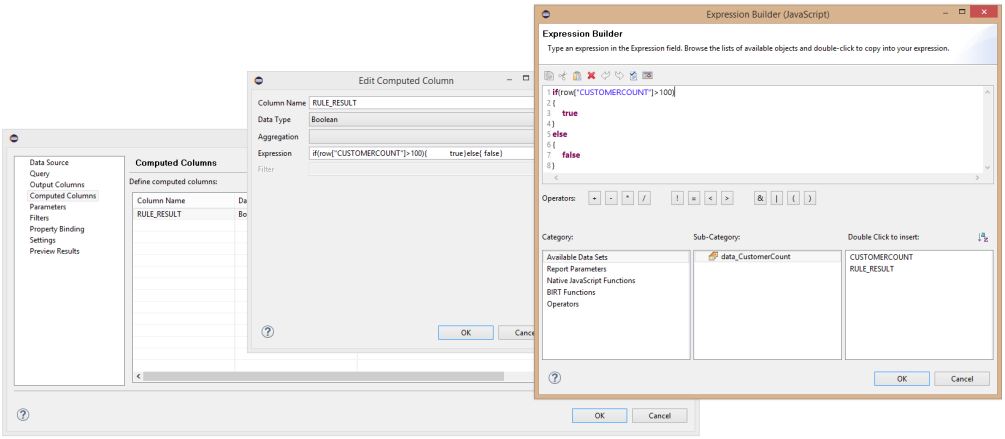
Verteiler-Regeln können auf die gleiche Art und Weise erstellt werden wie Loop-Regeln. Der Rückgabetyp der Regel ist jedoch immer ein String. Die Spalte aus dem Data Set, welche die E-Mail Adresse des Empfängers bereitstellt, fungiert als Value Column. Der Display Text kann entweder als einfaches Label oder zur Verknüpfung mit einer

Parameter Loop-Regel genutzt werden. Letzteres erfordert, dass in dem Display Text aus der Verteiler Regel und der Parameter Loop Regel, jeweils die selbe Spalte, aus dem selben DataSet, zugewiesen wird. Es ist auch möglich mehrere E-Mail Adressen komma-separiert, in eine Zeile einzutragen, beispielsweise wenn mehrere Empfänger mit den aus der Parameter Loop Regel entstandenen Bericht zugesendet bekommen sollen.



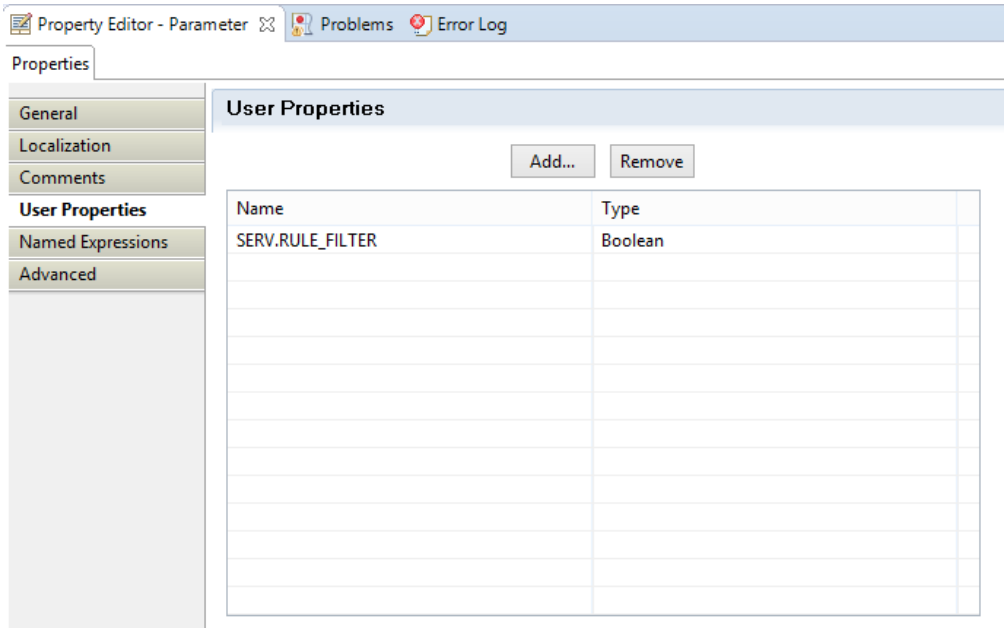
8.1.3 Event Regeln

Um eine Event-Regel zu entwickeln, befolgen Sie zunächst die Schritte zur Erzeugung einer Standard-Regel. Zuerst wird ein Data Set, mit entsprechenden Abfragen erstellt. Dann wird zusätzlich eine Computed Column vom Typ Boolean erstellt. Die eigentliche Regel wird als Skript auf die eben erstellte Computed Column implementiert. Diese liefert für die Event Regel entweder "true" als Bedingung für Ausführen oder "false" für nicht Ausführen zurück.

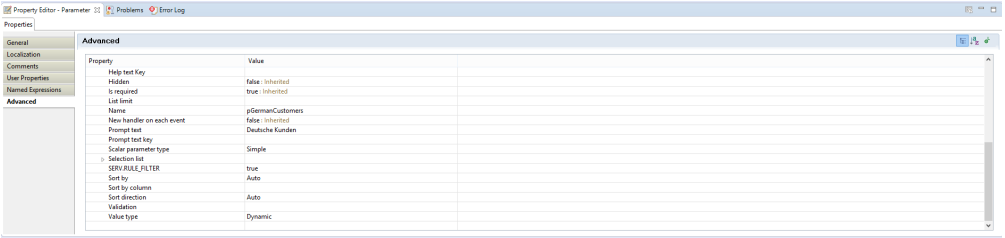


8.2 Filter definieren

Voraussetzung um einen Regel-Filter zu erstellen, ist als Grundlage eine Mengen-Regel (Loop- oder Verteilerregel) erforderlich. Der Filter muss explizit für die Loop-Regel aktiviert werden. Dazu muss am Regel Parameter ein User Property mit dem Namen "SERV.RULE_FILTER" erstellt werden.



Unterhalb des Reiters "Advanced" muss das Property mit "true" aktiviert werden.



Wie man einen Filter im **servBIRD** Portal definiert erfahren Sie im Abschnitt Regel Manager. Um die Filter Bedingung auszulesen wählen Sie nun das dem Regel-Parameter zugehörige Data Set aus. Dann wechseln Sie in die Script-

Ansicht und selektieren Before-Open Phase. Die Filter-Bedingung muss nun über folgenden Java Script ausgelesen werden:

:beforeOpen

```
filter = reportContext.getAppContext().get("RULE_FILTER");

if(filter!=null)
{
    filter = filter.replace("COLUMN_NAME", "CUSTOMERNUMBER");
    this.queryText += " and " + filter;
}
```

Der Rest des Codes muss dem Anwendungsfall (d.h. der Data Set Query) entsprechend implementiert werden. Im folgenden Abschnitt ein Beispiel:

Die Filter-Bedingung soll das Ergebnis der Loop-Regel "Alle deutschen Kunden" auf fünf bestimmte Kunden, die anhand ihrer Kundennummer ausgewählt wurden, einschränken.

Die Filter-Bedingung lautet:

```
CUSTOMERNUMBER IN (409, 415, 443, 459, 477)
```

Die Abfrage der ursprünglichen Regel lautet:

```
SELECT CUSTOMERS.CUSTOMERNUMBER, CUSTOMERS.CUSTOMERNAME FROM CUSTOMERS WHERE
 CUSTOMERS.COUNTRY = "Germany";
```

Nun erweitert man das Java Script um folgende Zeilen:

```
if(filter!=null)
{
    this.queryText  += " and " + filter;
}
```

Mit "this.queryText" kann das aktuelle Statement ausgelesen und entsprechend bearbeitet werden. Das Filter-Statement muss an die Abfrage angefügt werden. Der Java Script Code kann je nach Anwendungsfall bzw. SQL-Abfrage variieren.

9 Internationalisierung

Sie haben die Möglichkeit Berichte für unterschiedliche Sprachen zu entwickeln.

Das bedeutet, Sie können Beschriftungen (Labels) und Textmarken in sogenannte Property-Dateien auslagern und dadurch verschiedene Sprachen unterschiedlichem Zielgruppen bereitstellen.

> **Hinweis**
>
> Properties Files müssen bis einschließlich Java 8 mittels ISO 8859-1 kodiert sein. Alle nicht ASCII Zeichen müssen als Unicode Escape Characters (z.B. "\u0123") angegeben werden. Bei Verwendung anderer Kodierungen werden die Übersetzungen nicht richtig angezeigt.

9.1 Übersetzung erstellen

Erstellen Sie eine Datei folgendem Schema: "reportname_de.properties". "_de" steht dabei für den sogenannten "Country Code", also in diesem Fall "de" = "Deutschland".

In dieser Datei fügen Sie die Beschriftungen in Form von Schlüssel=Wert ein.

beispielbericht_de.properties

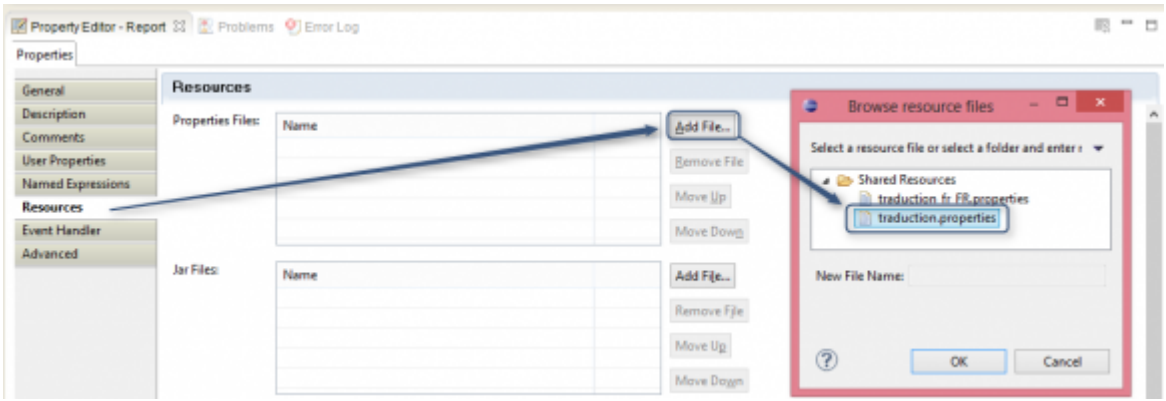
```
label1=Ich bin eine Beschriftung
example=Beispiel
```

Erstellen Sie nun eine weitere Datei mit dem "Country Code" für englischsprachige Beschriftungen: "_en":

beispielbericht_en.properties

```
label1=I am a Label
example=Example
```

Sie können innerhalb Ihres Report-Projektes einen beliebigen Ordner für die .properties-Dateien wählen, dennoch bietet es sich laut Konvention an einen "i18n"-Ordner zu erstellen.



9.2 Übersetzungen integrieren

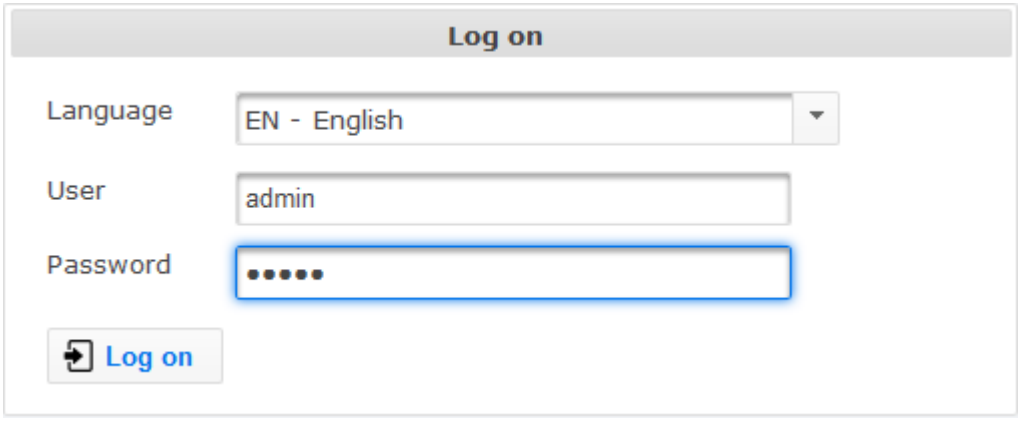
Wenn Sie im Bericht ein Label, einen Text oder die Beschriftungen eines Parameters dynamisch anhand der Lokalisierung (locale) übersetzen wollen, müssen Sie an dieser Stelle den Schlüssel aus der Übersetzung nutzen.

Im Javascript können Sie folgenden Code dafür nutzen:

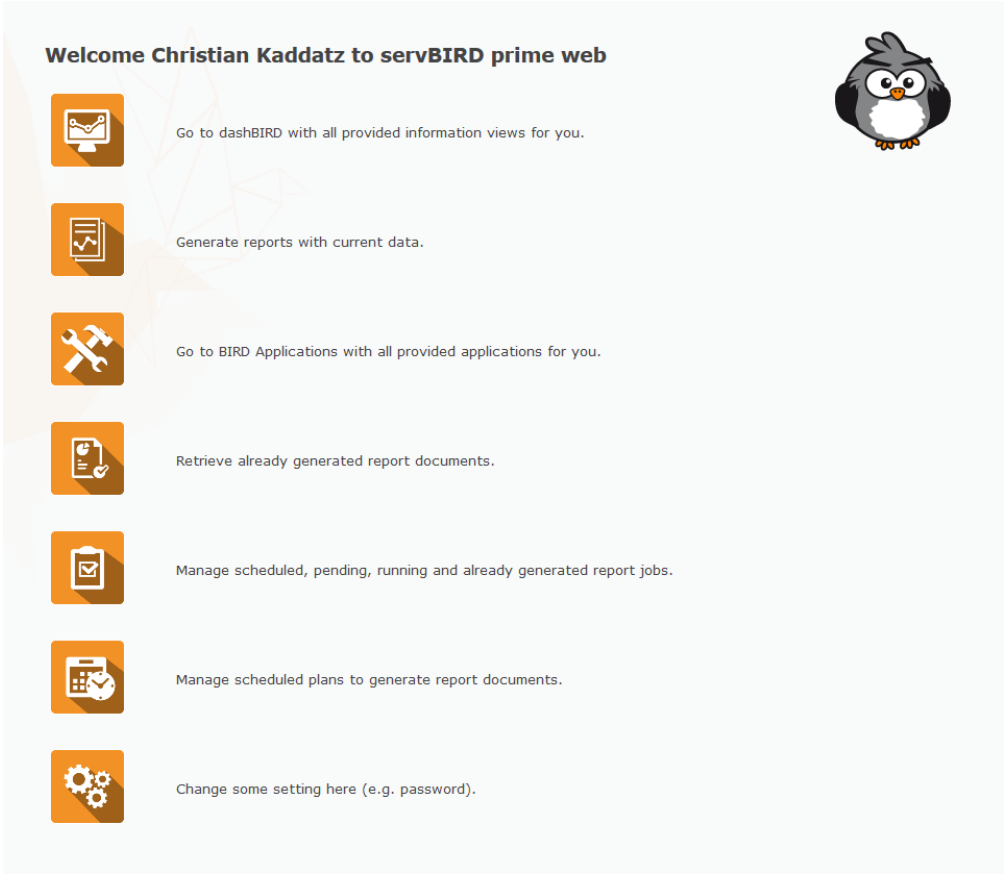
```
reportContext.getMessage("label1", reportContext.getLocale());
```

9.3 Anzeige im servBIRD

In der Regel erkennt **servBIRD** Ihre eingestellte Sprache anhand der "Locale" aus dem Browser. Diese wird im Login-Fenster selektiert.



Nun wird die Oberfläche von **servBIRD** und Berichte die eine Übersetzung in der gewählten Sprache besitzen, in der gewählten Sprache dargestellt.



10 Anzeigeformate für Datumsparameter

Die Parametereingabefelder in **servBIRD** bieten bei Datumsparametern einige Oberflächenelemente, die den Komfort bei der Eingabe von Datumsparametern erhöhen. Für Daten steht eine Kalenderkomponente zur schnellen Auswahl eines Datums bereit. Für Zeiten erlauben Schieberegler einen komfortable Werteeingabe.

Das Format, in dem Datumsparameterwerte angezeigt werden, kann im Report Designer am Parameter festgelegt werden. Hierzu dient die Einstellung "Format as". Dort kann aus verschiedenen Formatvorlagen gewählt werden oder ein eigenes Formatierungsmuster festgelegt werden.

Im Folgenden werden die durch **servBIRD** unterstützten Anzeigeformate für Datumsparameter aufgelistet.

Datentyp	Formatvorlage	Eingestelltes Format	Tatsächlich verwendetes Format	Beschreibung
Date	Unformatted		d. MMMM yyyy	
Date	Medium Date		dd.MM.yyyy	
Date	Long Date		d. MMMM yyyy	
Date	Short Date		dd.MM.yy	
Date	General Date	d. MMMM yyyy HH:mm:ss z	d. MMMM yyyy HH:mm:ss	Buchstabe "z" wird nicht unterstützt und wird daher ignoriert
Date	Custom Format (Jahr mit nur einem 'y')	y	Jahr als yyyy	Buchstabe "y" wird nicht unterstützt und wird daher durch "yyyy" ersetzt
Date Time	Unformatted		d. MMMM yyyy HH:mm:ss	
Date Time	Medium Date		dd.MM.yyyy	
Date Time	Long Date		d. MMMM yyyy	
Date Time	Short Date		dd.MM.yy	
Date Time	General Date	d. MMMM yyyy HH:mm:ss z	d. MMMM yyyy HH:mm:ss	Buchstabe "z" wird nicht unterstützt und wird daher ignoriert
Date Time	Long Time	HH:mm:ss z	HH:mm:ss	Buchstabe "z" wird nicht unterstützt und wird daher ignoriert
Date Time	Short Time		HH:mm	

Datentyp	Formatvorlage	Eingestelltes Format	Tatsächlich verwendetes Format	Beschreibung
Date Time	Medium Time		HH:mm:ss	
Date Time	Custom Format (Jahr mit nur einem 'y')	y	Jahr als yyyy	Buchstabe "y" wird nicht unterstützt und wird daher durch "yyyy" ersetzt
Date Time	Custom Format (Zeit mit AM/PM Information "a")	MMMM yyyy HH:mm:ss a	MMMM yyyy HH:mm:ss	Buchstabe "a" wird nicht unterstützt und wird daher ignoriert
Date Time	Custom Format (Zeit mit Millisekunden Information "SSS")	MMMM yyyy HH:mm:ss.SSS	MMMM yyyy HH:mm:ss	Formatbestandteil "SSS" wird nicht unterstützt und wird daher ignoriert
Date Time	Custom Format (Zeit nur in Minuten "mm")	mm	MMMM yyyy HH:mm:ss	Format "mm" wird nicht unterstützt und wird daher ignoriert
Date Time	Custom Format (Zeit nur in Sekunden "ss")	ss	MMMM yyyy HH:mm:ss	Format "ss" wird nicht unterstützt und wird daher ignoriert
Time	Unformatted		HH:mm:ss	
Time	Long Time	HH:mm:ss z	HH:mm:ss	Buchstabe "z" wird nicht unterstützt und wird daher ignoriert
Time	Short Time		HH:mm	
Time	Medium Time		HH:mm:ss	
Time	Custom Format (Zeit mit AM/PM information "a")	HH:mm:ss a	HH:mm:ss	Buchstabe "a" wird nicht unterstützt und wird daher ignoriert
Time	Custom Format (Zeit mit millisecond information "SSS")	HH:mm:ss.SSS	HH:mm:ss	Formatbestandteil "SSS" wird nicht unterstützt und wird daher ignoriert
Time	Custom Format (Zeit mit nur minutes "mm")	mm	HH:mm:ss	Format "mm" wird nicht unterstützt und wird daher ignoriert
Time	Custom Format (Zeit mit nur seconds "ss")	ss	HH:mm:ss	Format "ss" wird nicht unterstützt und wird daher ignoriert

11 Liste verfügbarer Icons

In servBIRD können an unterschiedlichen Stellen Icons genutzt werden um den Wiedererkennungswert von Elementen zu steigern. Nachfolgend sind die verfügbaren Icons mit Ihrem Klassennamen augeführt.

Als PDF herunterladen

fa-bed
fa-buysellads
fa-cart-arrow-down
fa-cart-plus
fa-connectdevelop
fa-dashcube
fa-diamond
fa-facebook-official
fa-forumbee
fa-heartbeat
fa-hotel (alias)
fa-leanpub
fa-mars
fa-mars-double
fa-mars-stroke
fa-mars-stroke-h
fa-mars-stroke-v
fa-medium
fa-mercury
fa-motorcycle
fa-neuter
fa-pinterest-p
fa-sellsy
fa-server
fa-ship
fa-shirtsinbulk
fa-simplybuilt
fa-skyatlas
fa-street-view
fa-subway
fa-train
fa-transgender
fa-transgender-alt
fa-user-plus
fa-user-secret
fa-user-times
fa-venus
fa-venus-double
fa-venus-mars
fa-viacoin
fa-whatsapp
fa-adjust
fa-adn
fa-align-center
fa-align-justify
fa-align-left
fa-align-right
fa-ambulance
fa-anchor
fa-android
fa-angellist
fa-angle-double-down
fa-angle-double-left
fa-angle-double-right
fa-angle-double-up
fa-angle-down
fa-angle-left
fa-angle-right
fa-angle-up
fa-apple
fa-archive
fa-area-chart
fa-arrow-circle-down
fa-arrow-circle-left
fa-arrow-circle-o-down
fa-arrow-circle-o-left
fa-arrow-circle-o-right
fa-arrow-circle-o-up
fa-arrow-circle-right
fa-arrow-circle-up
fa-arrow-down
fa-arrow-left
fa-arrow-right
fa-arrow-up
fa-arrows
fa-arrows-alt
fa-arrows-h
fa-arrows-v
fa-asterisk
fa-at
fa-automobile
fa-backward
fa-ban
fa-bank
fa-bar-chart
fa-bar-chart-o
fa-barcode
fa-bars
fa-beer
fa-behance
fa-behance-square
fa-bell
fa-bell-o
fa-bell-slash
fa-bell-slash-o
fa-bicycle
fa-binoculars
fa-birthday-cake
fa-bitbucket
fa-bitbucket-square
fa-bitcoin
fa-bold
fa-bolt
fa-bomb
fa-book
fa-bookmark
fa-bookmark-o
fa-briefcase
fa-btc
fa-bug
fa-building
fa-building-o
fa-bullhorn
fa-bullseye
fa-bus
fa-cab
fa-calculator
fa-calendar
fa-calendar-o
fa-camera
fa-camera-retro
fa-car
fa-caret-down
fa-caret-left
fa-caret-right
fa-caret-square-o-down
fa-caret-square-o-left
fa-caret-square-o-right
fa-caret-square-o-up
fa-caret-up
fa-cc
fa-cc-amex
fa-cc-discover
fa-cc-mastercard
fa-cc-paypal
fa-cc-stripe
fa-cc-visa
fa-certificate
fa-chain
fa-chain-broken
fa-check
fa-check-circle
fa-check-circle-o
fa-check-square
fa-check-square-o
fa-chevron-circle-down
fa-chevron-circle-left
fa-chevron-circle-right
fa-chevron-circle-up
fa-chevron-down
fa-chevron-left
fa-chevron-right
fa-chevron-up
fa-child
fa-circle
fa-circle-o
fa-circle-o-notch
fa-circle-thin
fa-clipboard
fa-clock-o
fa-close
fa-cloud
fa-cloud-download
fa-cloud-upload
fa-cny
fa-code
fa-code-fork
fa-codepen
fa-coffee
fa-cog
fa-cogs
fa-columns
fa-comment
fa-comment-o
fa-comments
fa-comments-o
fa-compass
fa-compress
fa-copy
fa-copyright
fa-credit-card
fa-crop
fa-crosshairs
fa-css3
fa-cube
fa-cubes
fa-cut
fa-cutlery
fa-dashboard
fa-database
fa-dedent
fa-delicious
fa-desktop
fa-deviantart
fa-digg
fa-dollar
fa-dot-circle-o
fa-download
fa-dribbble
fa-dropbox
fa-drupal
fa-edit
fa-eject
fa-ellipsis-h
fa-ellipsis-v
fa-empire
fa-envelope
fa-envelope-o
fa-envelope-square
fa-eraser
fa-eur
fa-euro
fa-exchange
fa-exclamation
fa-exclamation-circle
fa-exclamation-triangle
fa-expand
fa-external-link
fa-external-link-square
fa-eye
fa-eye-slash
fa-eyedropper
fa-facebook
fa-facebook-square
fa-fast-backward
fa-fast-forward
fa-fax
fa-female
fa-fighter-jet
fa-file
fa-file-archive-o
fa-file-audio-o
fa-file-code-o
fa-file-excel-o
fa-file-image-o
fa-file-movie-o
fa-file-o
fa-file-pdf-o
fa-file-photo-o
fa-file-picture-o
fa-file-powerpoint-o
fa-file-sound-o
fa-file-text
fa-file-text-o
fa-file-video-o
fa-file-word-o
fa-file-zip-o
fa-files-o
fa-film
fa-filter
fa-fire
fa-fire-extinguisher
fa-flag
fa-flag-checkered
fa-flag-o
fa-flash
fa-flask
fa-flickr
fa-floppy-o
fa-folder
fa-folder-o
fa-folder-open
fa-folder-open-o
fa-font
fa-forward
fa-foursquare
fa-frown-o
fa-futbol-o
fa-gamepad
fa-gavel
fa-gbp
fa-ge
fa-gear
fa-gears
fa-gift
fa-git
fa-git-square
fa-github
fa-github-alt
fa-github-square
fa-gittip
fa-glass
fa-globe
fa-google
fa-google-plus
fa-google-plus-square
fa-google-wallet
fa-graduation-cap
fa-group
fa-h-square
fa-hacker-news
fa-hand-o-down
fa-hand-o-left
fa-hand-o-right
fa-hand-o-up
fa-hdd-o
fa-header