



---

## servBIRD 3.18

Handbuch für Anwendungsentwickler

---

Dokumentversion vom 20. 11. 2023

# Inhalt

<b>1</b>	<b>Über diese Dokumentation .....</b>	<b>4</b>
1.1	Änderungshistorie.....	4
1.2	Zielgruppe und Voraussetzungen .....	4
1.3	Hinweise und Konventionen .....	5
<b>2</b>	<b>Architektur und Schnittstellen .....</b>	<b>6</b>
<b>3</b>	<b>servBIRD Anbindung .....</b>	<b>7</b>
3.1	mooBIRD REST Webservice .....	7
3.1.1	Technische Dokumentation .....	7
	Funktionen .....	7
	Login .....	7
	Logout.....	8
	getUserReports .....	8
	getReportParameter .....	9
	getReportParameterByld .....	10
	startReport .....	11
	getReportJob .....	12
	getReportJobStatus.....	13
	getDownloadOutputLink.....	14
	getNativeOutput .....	14
	getErrorOutput.....	15
	addParameterSet.....	16
	getParameterSet (ID) .....	16
	getParameterSet (Bericht).....	17
3.1.2	Installation und Konfiguration im Wildfly Applikation Server.....	18
3.1.3	Hinweise zur Integration .....	18
3.1.4	Benutzer an- und abmelden.....	18
3.1.5	Bericht ausführen und Ergebnis abrufen.....	18
3.2	SOAP Webservice .....	20
<b>4</b>	<b>Portal Integration.....</b>	<b>21</b>
4.1	Vorschau Umgebung.....	21
4.2	Views.....	21
4.2.1	Einstellungen.....	21

4.2.2 Dashbird .....	21
4.2.3 BIRD Applications.....	22
4.2.4 Berichte erstellen .....	22
Weitere URL Parameter .....	22
4.2.5 Berichte planen .....	24
4.2.6 Fertige Berichte .....	24
4.2.7 Aufgaben verwalten .....	24

# 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	<ul style="list-style-type: none"> <li>• Kleinere Änderung an der Struktur</li> <li>• Übersetzungen nachgezogen</li> </ul>	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 Architektur und Schnittstellen

**TRADUI Technologies** setzt mit **servBIRD** auf zukunftssichere offene Standards auf der Basis einer Java Enterprise Architektur.

In **servBIRD** kommen Java EE7, JSF (Java Server Faces), JPA (Java Persistence API) und OSGI-Frameworks (Open Services Gateway initiative) zum Einsatz.

**servBIRD** verwendet eine sogenannte Pure-Java-Structure, so **servBIRD** auf allen Betriebssystemen läuft, welche die Java EE7-Standards vollumfänglich unterstützen.

**servBIRD** besitzt sowohl eine SOAP- als auch eine REST-Schnittstelle.

## 3 servBIRD Anbindung

### 3.1 mooBIRD REST Webservice

**servBIRD** stellt einen REST Webservice für die Integration von **servBIRD** Funktionen in andere Anwendungen (APP, Portal, etc.) zur Verfügung.

Der Name der REST Schnittstelle lautet "mooBIRD".

mooBIRD wird als zusätzliches WAR File für **servBIRD** bereitgestellt.

#### 3.1.1 Technische Dokumentation

Die technische Dokumentation der Funktionen ist in der Anwendung selbst integriert und wird unter folgender URL bereitgestellt: <http://server:port/moobird>

#### Funktionen

Derzeit stehen folgende Funktionen zur Verfügung:

reportmanagement : Operations about reports			<a href="#">Show/Hide</a>   <a href="#">List Operations</a>   <a href="#">Expand Operations</a>   <a href="#">Raw</a>
POST	/reportmanagement/addParameterSet/{reportfolder}/{reportdesignfile}/{parametersetname}	addParameterSet	
GET	/reportmanagement/getDownloadOutputLink/{jobid}	getDownloadOutputLink	
GET	/reportmanagement/getErrorOutput/{jobid}	getErrorOutput	
GET	/reportmanagement/getNativeOutput/{jobid}/{native}	getNativeOutput	
GET	/reportmanagement/getParameterSet/{paramsetid}	getParameterSet	
GET	/reportmanagement/getParameterSet/{reportfolder}/{reportdesignfile}/{paramsetname}	getParameterSet	
GET	/reportmanagement/getReportJob/{jobid}	getReportJob	
GET	/reportmanagement/getReportJobStatus/{jobid}	getReportJobStatus	
GET	/reportmanagement/getReportParameter	getReportParameters	
GET	/reportmanagement/getReportParameterById	getReportParameters	
GET	/reportmanagement/getUserReports	getUserReports	
POST	/reportmanagement/login	login	
POST	/reportmanagement/logout	logout	
POST	/reportmanagement/startReport	startReport	

#### Login

Benutzer authentifizieren

**POST** /reportmanagement/login login

**Implementation Notes**  
User login

**Response Class**  
string

Response Content Type: application/json

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
username	(required)	<b>Username</b>	query	string
password	(required)	<b>Password</b>	query	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
500	Internal Server Error	
401	Username or password is wrong	

## Logout

### Benutzer abmelden

**POST** /reportmanagement/logout logout

**Implementation Notes**  
User logout

**Response Class**  
string

Response Content Type: application/json

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
username			header	string
authToken			header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
401	Wrong header parameters!	

## getUserReports

Kategorien, deren Berichte, Parameter und Parameter-Sets

GET </reportmanagement/getUserReports> getUserReports

**Implementation Notes**  
Get all user reports as tree structure

**Response Class**  
Model | Model Schema

```

{
  "children": [
    "CategoryTreeNode"
  ],
  "categoryTreeItems": [
    {
      "paramSet": [
        {
          "id": 0,
          "name": "",
        }
      ]
    }
  ]
}
    
```

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
locale	<input type="text" value="de_DE"/>	Locale code for language selection, like de_DE or en_EN	query	string
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Parameter username is null	
404	User not found	
500	Internal Server Error	

## getReportParameter

Alle Parameter zu einem Bericht (Parameter Berichtsdesign)

GET </reportmanagement/getReportParameter> getReportParameters

**Implementation Notes**  
Get all parameters for a report

**Response Class**  
Model | Model Schema

```
[
  {
    "name": "",
    "displayname": "",
    "value": "",
    "nullValue": false,
    "type": "",
    "scalartype": "",
    "group": "",
    "displaygroup": ""
  }
]
```

Response Content Type

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
reportFile	<input type="text" value="(required)"/>	Reportfile name concate with relative category path, like /Category /ReportFile.rptdesign	query	string
locale	<input type="text" value="de_DE"/>	Locale code for language selection, like de_DE or en_EN	query	string
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Parameter reportFile is null	
500	Internal Server Error	

## getReportParameterById

Alle Parameter zu einem Bericht (Parameter Berichtsdesign) über die Berichts Id

GET </reportmanagement/getReportParameterById> getReportParameters

**Implementation Notes**  
Get all parameters for a report

**Response Class**  
Model | Model Schema

```
[
  {
    "name": "",
    "displayname": "",
    "value": "",
    "nullValue": false,
    "type": "",
    "scalartype": "",
    "group": "",
    "displaygroup": ""
  }
]
```

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
reportId	<input type="text" value="-1"/>	<b>Report ID</b>	query	integer
locale	<input type="text" value="de_DE"/>	Locale code for language selection, like de_DE or en_EN	query	string
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Parameter reportFile is null	
500	Internal Server Error	

## startReport

Bericht starten

POST </reportmanagement/startReport> startReport

**Implementation Notes**  
Starts a report in servBIRD

**Response Class**  
string

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
body	<input type="text" value="(required)"/>	<b>Request config to start report</b>	body	Model   Model Schema
	Parameter content type: <input type="text" value="application/json"/>			<pre>{   "reportFilePath": "",   "reportId": 0,   "owner": "",   "outputFormat": "",   "parameterSetId": 0,   "reportParameters": [     {       "key": "",       "value": ""     }   ] }</pre> <p>Click to set as parameter value</p>
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong request parameter	
404	Owner not found	
500	Internal Server Error	
401	You are not allowed to execute this report	

## getReportJob

ReportJob Objekt abfragen

GET </reportmanagement/getReportJob/{jobid}> getReportJob

**Implementation Notes**  
Gets a single reportjob

**Response Class**  
Model | Model Schema

```

{
  "inputFile": "",
  "reportDesignFilePath": "",
  "parent": "ReportJob",
  "designFilePath": "",
  "designFileName": "",
  "outputFileName": "",
  "displayOutputFileName": "",
  "ownerUserID": "",
  "reportLocale": "",

```

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
jobid	<input type="text" value="-1"/>	<b>JobID</b>	path	integer
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
412	ReportJob status is null	
400	Wrong path parameter 'jobid'	
500	Internal Server Error	

## getReportJobStatus

Berichts Generierungsstatus Running, Success oder Failed

GET </reportmanagement/getReportJobStatus/{jobid}> getReportJobStatus

**Implementation Notes**  
Gets the actual report job status

**Response Class**  
string

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
jobid	<input type="text" value="-1"/>	<b>JobID</b>	path	integer
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong path parameter 'jobid'	
500	Internal Server Error	

## getDownloadOutputLink

Link auf das Ausgabeformat (HTML, PDF, etc.), wenn der Bericht erfolgreich generiert wurde

GET
/reportmanagement/getDownloadOutputLink/{jobid}
getDownloadOutputLink

**Implementation Notes**  
If job success, you will get the Download-Link to the associated reportjob output document. If job failed or canceled you will get the error output as string representation

**Response Class**  
string

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
jobid	<input type="text" value="-1"/>	<b>JobID</b>	path	integer
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong path parameter 'jobid'	
500	Internal Server Error	

## getNativeOutput

Byte-Stream Ausgabeformat (native Excel), wenn der Bericht erfolgreich generiert wurde.

GET </reportmanagement/getNativeOutput/{jobid}/{native}> getNativeOutput

**Implementation Notes**  
Get the requested native outfile to the associated reportjob as stream

**Response Class**  
Model | Model Schema

```

{
  "path": "",
  "name": "",
  "canonicalPath": "",
  "parent": "",
  "absolute": false,
  "absoluteFile": "File",
  "absolutePath": "",
  "canonicalFile": "File",
  "freeSpace": 0,
}
    
```

Response Content Type

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
jobid	<input type="text" value="-1"/>	<b>JobID</b>	path	integer
native	<input type="text" value="(required)"/>	<b>NativeFileName</b>	path	string
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong path parameter 'jobid'	
500	Internal Server Error	

## getErrorOutput

Ausgabe der Fehlermeldung, falls die Berichtsgenerierung fehlschlug.

GET </reportmanagement/getErrorOutput/{jobid}> getErrorOutput

**Implementation Notes**  
If job failed, you will get the error output as string representation.

**Response Class**  
string

Response Content Type

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
jobid	<input type="text" value="-1"/>	<b>JobID</b>	path	integer
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong path parameter 'jobid'	
500	This job has no error output	

## addParameterSet

Für einen Bericht ein neues Parameter Set anlegen

POST
/reportmanagement/addParameterSet/{reportfolder}/{reportdesignfile}/{parametersetname}
addParameterSet

**Implementation Notes**  
Add a new paramset

**Response Class**  
string

Response Content Type:

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
reportfolder	<input type="text" value="(required)"/>	ReportFolder	path	string
reportdesignfile	<input type="text" value="(required)"/>	ReportDesignFile -> REPORTDESIGNFILE.rptdesign	path	string
parametersetname	<input type="text" value="(required)"/>	ParameterSetName	path	string
owner	<input type="text"/>	Owner	query	string
globalparamset	<input type="text" value="false (default)"/>	GlobaleParamSet	query	boolean
locale	<input type="text" value="de_DE"/>	Locale	query	string
body	<div style="border: 1px solid #ccc; padding: 5px; min-height: 50px;">(required)</div>	Parameters	body	Model   Model Schema
				<div style="border: 1px solid #ccc; padding: 5px; background-color: #fff9c4; font-family: monospace; font-size: 0.9em;"> <pre>[   {     "key": "",     "value": ""   } ]</pre> </div> <p style="font-size: 0.8em; margin-top: 5px;">Click to set as parameter value</p>
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

Parameter content type:

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong parameter	
404	Owner not found	
500	An general error occurred	
401	Username or password is wrong	

## getParameterSet (ID)

Parameter Set ausgeben über die Parameter Set Id.

**GET** /reportmanagement/getParameterSet/{paramsetId} getParameterSet

**Implementation Notes**  
Returns parameter set by id.

**Response Class**  
string

Response Content Type

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
paramsetId	<input type="text" value="-1"/>	ParamSetId	path	integer
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong path parameter 'paramSetId'	
403	You have no permission for the requested paramset	
404	ParameterSet not found	
500	An general error occured	
401	Username or password is wrong	

## getParameterSet (Bericht)

Alle Parameter Sets eines Berichts ausgeben.

**GET** /reportmanagement/getParameterSet/{reportfolder}/{reportdesignfile}/{paramsetName} getParameterSet

**Implementation Notes**  
Returns parameter set by name.

**Response Class**  
string

Response Content Type

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
reportfolder	<input type="text" value="(required)"/>	ReportFolder	path	string
reportdesignfile	<input type="text" value="(required)"/>	ReportDesignFile -> REPORTDESIGNFILE.rptdesign	path	string
paramsetName	<input type="text" value="(required)"/>	ParamSetName	path	string
username	<input type="text"/>		header	string
authToken	<input type="text"/>		header	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
400	Wrong path parameter 'paramSetName'	
403	You have no permission for the requested paramset	
404	ParameterSet not found	
500	An general error occured	
401	Username or password is wrong	

### 3.1.2 Installation und Konfiguration im Wildfly Applikation Server

Um mooBIRD auf dem Wildfly Applikation Server zu deployen ist es notwendig folgende Zeile in der Datei *"standalone.conf"* (Linux) bzw. *"standalone.conf.bat"* (Windows) einzutragen:

```
JAVA_OPTS="$JAVA_OPTS  
-Dcom.sun.jersey.server.impl.cdi.lookupExtensionInBeanManager=true"
```

Dazu muss mooBIRD.war im Deployment-Verzeichnis bereitgestellt werden.

### 3.1.3 Hinweise zur Integration

In den folgenden Abschnitten wird auf einige Punkte hingewiesen, die bei der Verwendung der Funktionen der REST-Schnittstelle zu beachten sind.

Die detaillierten Angaben zu den Funktionen und Parametern entnehmen sie bitte dem vorangegangenen Abschnitt.

### 3.1.4 Benutzer an- und abmelden

Zunächst ist es notwendig einen Benutzer zu authentifizieren. Rufen Sie dazu die Methode `login` auf. Bei erfolgreicher Authentifizierung wird ein sogenannter `AuthToken` zurückgegeben.

Ohne gültiges Token können vom Benutzer keine weiteren mooBIRD Funktionen ausgeführt werden. Das generierte Token, sowie der Benutzername müssen in sämtlichen Funktionsaufrufen, als Header-Parameter übergeben werden, ansonsten erhalten Sie eine Fehlermeldung.

Nach dem Aufruf der `logout` Funktion wird das Token für den Benutzer ungültig.

Ein weiterer Aufruf der Funktion `login` bewirkt, dass für diesen Benutzer ein neues Token generiert wird:

### 3.1.5 Bericht ausführen und Ergebnis abrufen

Um einen Bericht auszuführen und das Ausgabedokument abzurufen, sind folgende Schritte notwendig:

POST /reportmanagement/login login

**Implementation Notes**  
User login

**Response Class**  
string

Response Content Type: application/json

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
username	admin	Username	query	string
password	admin	Password	query	string

**Response Messages**

HTTP Status Code	Reason	Response Model
200	OK	
500	Internal Server Error	
401	Username or password is wrong	

[Try it out!](#) [Hide Response](#)

**Request URL**

```
http://192.168.2.120:8080/moobird/service/reportmanagement/login?username=admin&password=admin
```

**Response Body**

```
4b27a46c-71b3-476c-a13c-fb9c4c4f3c9d
```

**Response Code**

```
200
```

**Response Headers**

```
{
  "Server": "Apache-Coyote/1.1",
  "Access-Control-Allow-Origin": "*",
  "Access-Control-Allow-Methods": "GET, POST, DELETE, PUT",
  "Access-Control-Allow-Headers": "Content-Type",
  "Content-Type": "text/plain",
  "Transfer-Encoding": "chunked",
  "Date": "Tue, 10 May 2016 10:23:24 GMT"
}
```

### Beispiel Aufruf der Login Methode

1. Funktion: startReport  
(diese Funktion erwartet ein JSON-Objekt als Parameter, siehe Abschnitt "Funktionen")
2. Ergebnis: ReportJobID
3. Funktion: getReportJobStatus(ReportJobId)
  - a. Ergebnis: Running
    - i. Bericht wird noch ausgeführt
  - b. Ergebnis: Success
    - i. Funktion: getDownloadOutputLink(ReportJobId)
    - ii. Ergebnis: Link auf die Ausgabedatei
  - c. Ergebnis: Failed
    - i. Funktion: getErrorOutput(ReportJobID)
    - ii. Ergebnis: Fehlermeldung, gemäß dem TRADUI Toolbox Logging, im HTML Format

## 3.2 SOAP Webservice

Über einen Webservice (SOAP) kann **servBIRD** Fachanwendungen mit Berichten versorgen. Die Implementierung des Aufrufs der Webservice Schnittstelle muss in der Fachanwendung erfolgen. Die Schnittstelle unterstützt das Ausführen, Parametrisieren und Abrufen der Berichte über **servBIRD**.

**servBIRD** unterstützt, aus kompatibilitätsgründen, zwei Varianten der Implementierung des Webservices:

- Ohne optimierte Übertragung binärer Daten
- Mit optimierter Übertragung binärer Daten

Die Erfahrung hat gezeigt dass bestimmte Applikationen nur die "nicht optimierte" Variante unterstützen. Der Unterschied besteht auch im Funktionsumfang. Die "optimierte" Variante unterstützt auch das Streamen von Ausgabedokumenten, während die andere Variante nur Links zum Download zur Verfügung stellen kann.

Die WSDL ersterer Variante können Sie sich unter folgender Url anschauen:

<http://HOST:PORT/BirdWebService/BirdWebServiceClassic/BirdWebServiceClassicBean?wsdl>

Die zweite Variante unter:

<http://HOST:PORT/BirdWebService/BirdWebService/BirdWebServiceBean?wsdl>

Die vollständige Funktionsreferenz können Sie mit dieser HTML Datei im Anhang herunterladen:



## 4 Portal Integration

Für die Integration von **servBIRD** in ein Unternehmensportal steht zu jedem Menüpunkt von **servBIRD** ein direkter View zur Verfügung. Diese Views arbeiten ohne die **servBIRD** Menüsteuerung und sind jeweils gesondert über deren URL abrufbar.

Folgende Seiten werden als eigene Haupt-Views zur Verfügung gestellt:

- Einstieg und Hilfe
- dashBIRD
- BIRD Applications
- Berichte abrufen
- Berichte planen
- Fertige Berichte
- Aufgaben verwalten
- Einstellungen

Des Weiteren existiert ein gesonderter Login-View, auf den bei Bedarf automatisch umgeleitet wird.

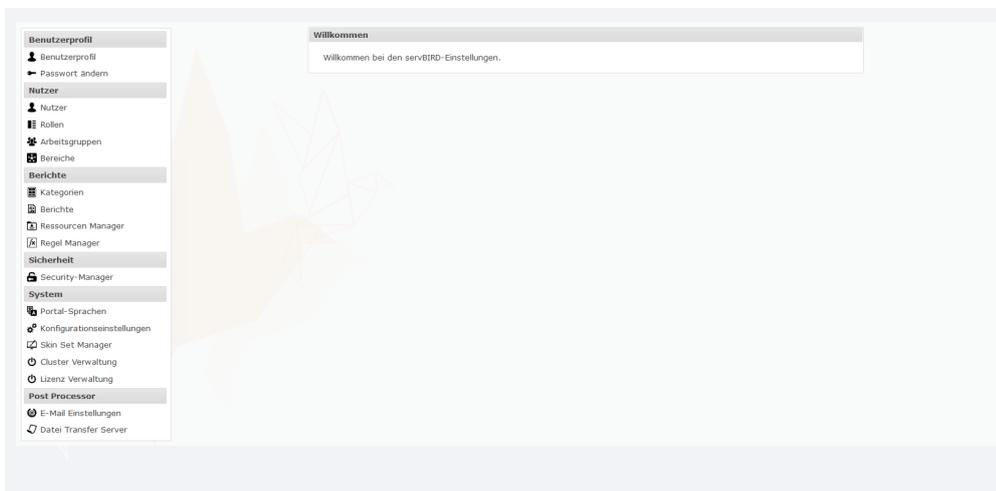
### 4.1 Vorschau Umgebung

Auf der Unterseite „servBIRD/test/home.faces“, kann man sich eine mögliche Integration der Views anschauen und selbst testen.

### 4.2 Views

#### 4.2.1 Einstellungen

Link: [servBIRD/admin/index.faces?intview=true](http://servBIRD/admin/index.faces?intview=true)



#### 4.2.2 Dashbird

Link: [servBIRD/portal/dashbird.faces?intview=true](http://servBIRD/portal/dashbird.faces?intview=true)

Um ein Dashboard direkt zu generieren, muss die nachfolgende URL verwendet und zusätzliche URL Parameter verwendet werden (Der Parameter "dashboardid" ist für diese Funktionalität erforderlich. Der Parameter "paramsetid" ist nur erforderlich wenn das Dashboard die Eingabe von Parameterwerten erfordert und der aufrufende Nutzer für dieses Dashboard kein Parameter Set als Favorit gesetzt hat.

Seite für direkte-Dashboard Ausführung: `servBIRD/portal/dashbirdrun.faces`

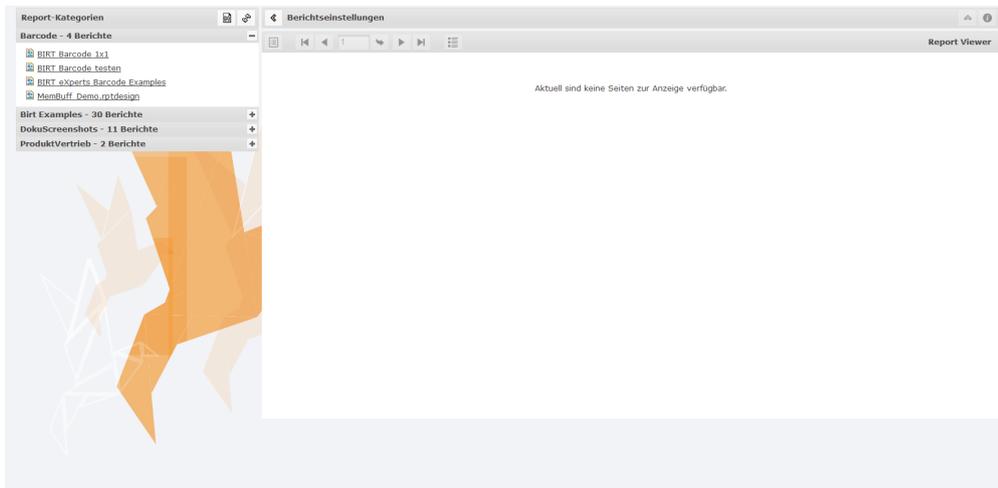
URL Parameter	Beschreibung	Beispiel
dashboardid	Die ID des aufzurufenden Dashboards	<code>servBIRD/portal/dashbirdrun.faces?dashboardid=42</code>
paramsetid	Die ID des zu verwendenden Parameter Sets	<code>servBIRD/portal/dashbirdrun.faces?dashboardid=42&amp;paramsetid=128</code>

### 4.2.3 BIRD Applications

Link: `servBIRD/portal/birdapplications.faces?intview=true`

### 4.2.4 Berichte erstellen

Link: `servBIRD/portal/runreport.faces?intview=true`



Weitere URL Parameter

URL Parameter	Beschreibung	Syntax
run	Direktausführung eines Berichts	<code>servBIRD/portal/runreport.faces?intview=true&amp;run=true&amp;reportfolder=/CATEGORY/&amp;reportname=REPORTFILENAME.rptdesign</code>
reportfolder	Das Kategorieverzeichnis des Report Designs	<code>servBIRD/portal/runreport.faces?intview=true&amp;run=true&amp;reportfolder=/CATEGORY/&amp;reportname=REPORTFILENAME.rptdesign</code>

URL Parameter	Beschreibung	Syntax
reportname	Der Dateiname des Report Designs	servBIRD/portal/runreport.faces?intview=true&run=true&reportfolder=/CATEGORY/&reportname=REPORTFILENAME.rptdesign
outputFormat	Das Ausgabeformat Optional. Standardwert: "HTML"	servBIRD/portal/runreport.faces?intview=true&outputFormat=PDF&run=true&reportfolder=/CATEGORY/&reportname=REPORTFILENAME.rptdesign
showCategoryViewer	Steuert Anzeige des Reportbrowsers Optional. Standardwert: "true"	servBIRD/portal/runreport.faces?intview=true&run=true&showCategoryViewer=false&reportfolder=/CATEGORY/&reportname=REPORTFILENAME.rptdesign
-- parametername	Übergabe von Parameterwerten	servBIRD/portal/runreport.faces?intview=true&run=true&showCategoryViewer=false&reportfolder=/CATEGORY/&reportname=REPORTFILENAME.rptdesign&--param1=value1&--param2=value2
multiValueDelimiter	Trennzeichen für Multivalue-Parameterwerte Optional. Standardwert: " " (%7C)	servBIRD/portal/runreport.faces?intview=true&run=true&multiValueDelimiter=%3B&reportfolder=/CATEGORY/&reportname=REPORTFILENAME.rptdesign&--param1=germany%3Bfrance%3Bsweden

**Hinweis**

Parameterwerte müssen bei der Übergabe via URL entsprechend dem MIME Format "application/x-www-form-urlencoded" codiert sein.

Siehe hierzu: <https://www.w3.org/TR/html4/interact/forms.html#h-17.13.4.1>

**Hinweis**

Parameterwerte für Datumparameter müssen in folgenden Formaten übergeben werden:

Date: yyyy-MM-dd

Time: HH:mm:ss

DateTime: yyyy-MM-dd HH:mm:ss

### Information

Multivalue-Parameterwerte können alternativ auch durch Mehrfachdefinition des Parameters in der URL angegeben werden. Beispiel:

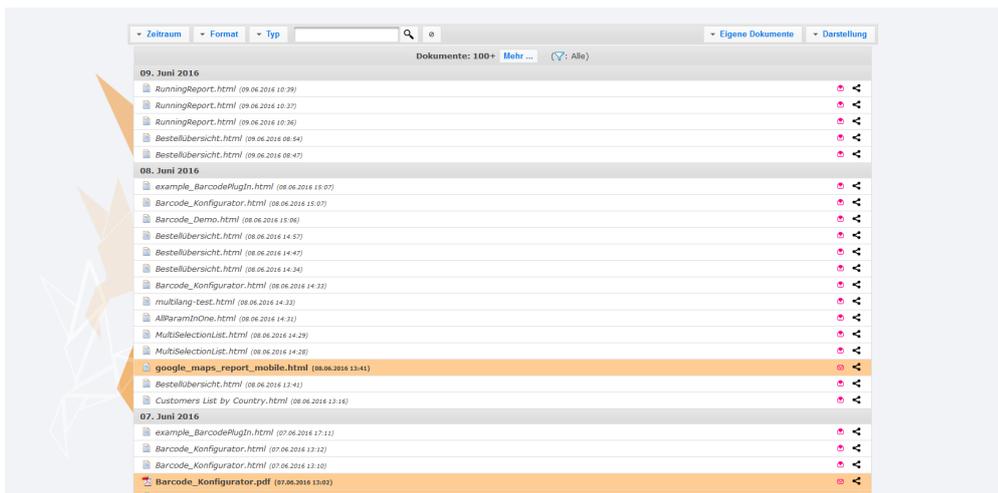
```
servBIRD/portal/runreport.faces?intview=true&run=true&reportfolder=/CATEGORY/
&reportname=REPORTFILENAME.rptdesign&--param1=germany&--param1=france&--param1=sweden
```

## 4.2.5 Berichte planen

Link: [servBIRD/portal/planschedulereport.faces?intview=true](http://servBIRD/portal/planschedulereport.faces?intview=true)

## 4.2.6 Fertige Berichte

Link: [servBIRD/portal/showreportdocuments.faces?intview=true](http://servBIRD/portal/showreportdocuments.faces?intview=true)



## 4.2.7 Aufgaben verwalten

Link: [servBIRD/portal/showcompletereportjobs.faces?intview=true](http://servBIRD/portal/showcompletereportjobs.faces?intview=true)

Enthält auch Verweise auf laufende & wartende Berichte, sowie auf Zeitpläne.

### Hinweise zur Integration

Bis zur Inbetriebnahme der SSO Funktionalität wird man bei der ersten Anmeldung bzw. bei Ablauf der Session auf den Login-View umgeleitet.

Die Views sollten innerhalb eines <iFrame> oder über ein AJAX-Get in das Portal integriert werden. Hierbei ist es wichtig, dass bei jedem neuen Seitenaufruf der Frame explizit neu geladen wird.

Ein Beispiel für die Integration innerhalb eines <iFrame>:

#### iFrame Codebeispiel

```
<iFrame id="iFrame" src="localhost:8080/servBIRD/portal/intviews/  
showreportjobs.faces" style="max-width: 1600px; margin: auto;" frameborder="0"  
width="100%" height="980px" scrolling="no" ></iFrame>
```

Die Hintergrundfarbe der Views muss (sofern gewünscht) manuell angepasst werden.

Dazu wechselt man in den Administrationsbereich auf **Skin Set Manager**, aktiviert das Skin vom Typ „System Portal Skin“ und ändert die folgenden beiden Styles dementsprechend:

In diesem Fall wäre der innere, sowie äußere Hintergrund „weiß“.