

# IRTLib Dokumentation: Software für die Verwaltung und Auslieferung computergestützter Assessments

Ulf Kroehne

Letzte Änderung: 30. Dezember 2023

Vorgeschlagene Zitierung:

Kroehne, U. (2023). IRTlib Documentation: Software for the administration and delivery of computer-based assessments [IRTLib Dokumentation: Software für die Verwaltung und Auslieferung computergestützter Assessments]. DIPF, Frankfurt am Main, Germany. <https://doi.org/10.5281/zenodo.10441352>

# Inhaltsverzeichnis

<b>1. IRTlib Software</b>	<b>5</b>
<b>2. Download &amp; Installation</b>	<b>5</b>
2.1. Offline (Windows)	5
2.1.1. Studienvorbereitung mit Offline Editor	6
2.1.2. Studiendurchführung mit Offline Player	6
2.2. Online (Docker)	7
<b>I. Vorbereitung / Preparation</b>	<b>7</b>
<b>3. Vorbereitung: Übersicht / Preparation: Overview</b>	<b>8</b>
3.1. Eingebettete Programmhilfe	9
3.2. Vorbereitung von <i>CBA ItemBuilder</i> -Inhalten	9
3.2.1. Einsprungpunkt ( <i>Task</i> )	10
3.2.2. Verhalten der Anzeige ( <i>Scaling Options</i> )	10
3.2.3. Definition des <i>Scorings</i> (Ergebnisdaten)	11
3.2.4. Integrierte Multimedia-Inhalte (Ressourcen)	11
<b>4. Vorbereitung Studien / Preparation Studies</b>	<b>13</b>
4.1. Studienverwaltung	13
4.1.1. Studien Anlegen	13
4.1.2. Weitere Funktionen und Hinweise	14
4.2. Grundlegende Konfigurationen	15
4.2.1. Einstellungen zur Studie	15
4.3. Zugang zu Studien (Login)	16
4.3.1. Konfiguration der Anmeldung	17
4.4. Anzeige von Assessment-Inhalten	19
4.4.1. Anzeigeeinstellungen	19
4.4.2. Skalierung und Ausrichtung	22
4.5. Menü für Testadministratoren	23
4.5.1. Konzept eines Testleitermenüs (Menü für Testadministratoren)	23
4.6. Abschluss von Erhebungen	26
4.6.1. Session und Session-Ende	26
<b>5. Vorbereitung: Erhebungsteile / Preparation: Study Parts</b>	<b>27</b>
5.1. Erhebungsteilverwaltung	27
5.1.1. Erhebungsteil Anlegen	27
5.1.2. Grundkonfiguration für Erhebungsteile	29
5.2. Assessmentinhalte (Items) Einfügen	30
5.2.1. Items Konfigurieren	30
5.3. Bearbeitungszeit	32
5.3.1. Zeitbegrenzung Definieren	32
5.3.2. Items nach einer Zeitbegrenzung	33
5.3.3. Items vor einer Zeitbegrenzung	34
5.4. Variablen	35
5.5. Codebook	35
5.6. ItemPool	35

5.7.	Routing innerhalb von Erhebungsteilen . . . . .	35
5.7.1.	Zusammenfassung zu Routing innerhalb von Erhebungsteilen . . . . .	35
5.7.2.	Verwendung von <i>Blockly</i> zur Ablaufsteuerung . . . . .	41
5.7.3.	Fortgeschrittene <i>Blockly</i> -Verwendung . . . . .	46
5.7.4.	Kommentieren von <i>Blockly</i> -Code . . . . .	65
5.7.5.	Darstellung von <i>Blockly</i> -Code . . . . .	66
5.8.	Routing zwischen Erhebungsteilen . . . . .	68
5.8.1.	Zusammenfassung zu Routing zwischen Erhebungsteilen . . . . .	68
 <b>II. Datenerhebung / Data Collection</b>		<b>70</b>
<b>6. Datenerhebung: Übersicht / Data Collection: Overview</b>		<b>71</b>
6.1.	Übersicht: Schritte zur Verwendung eines <i>IRTLib Player</i> zur <i>Datenerhebung</i> . . . . .	71
<b>7. Datenerhebung: Veröffentlichen &amp; Exportieren / Data Collection: Publish &amp; Export</b>		<b>72</b>
7.1.	Checkliste vor dem Veröffentlichen . . . . .	73
7.2.	Veröffentlichen & Exportieren . . . . .	73
7.2.1.	Veröffentlichen . . . . .	74
7.2.2.	Exportieren . . . . .	75
7.2.3.	Studienversionen . . . . .	75
<b>8. Datenerhebung: In IRTLib Player Importieren / Data Collection: Import into IRTLib Player</b>		<b>76</b>
8.1.	Konfiguration Importieren . . . . .	76
8.1.1.	Automatischer Import . . . . .	76
8.1.2.	Manuelles Importieren . . . . .	77
8.2.	Auslieferungen Konfigurieren . . . . .	79
8.2.1.	Desktop-Version (Windows) . . . . .	80
8.2.2.	Lokaler Server (Windows) . . . . .	81
8.2.3.	Online-Version (Docker) . . . . .	81
8.3.	Auslieferungen Testen und Freigeben . . . . .	82
8.3.1.	Vorgeschlagene Testpläne . . . . .	83
8.3.2.	Datenerhebung Durchführen . . . . .	84
<b>9. Datenerhebung: Datenaufbereitung / Data Collection: Data Post-Processing</b>		<b>85</b>
9.1.	Datenaufbereitung . . . . .	85
9.1.1.	Datenabruf mit LogFSM . . . . .	85
9.1.2.	Datenabruf über die Kommandozeile . . . . .	86
9.1.3.	Ergebnisdaten . . . . .	87
9.1.4.	Log-Daten . . . . .	87
9.1.5.	Dateien in den <i>Rohdatenarchiven</i> . . . . .	88
 <b>III. Allgemein / General</b>		<b>91</b>
<b>10. Einstellungen / Settings</b>		<b>92</b>
10.1.	Übersicht . . . . .	92
10.1.1.	Einstellungen . . . . .	92
10.1.2.	Über das Programm . . . . .	92
10.2.	Laufzeitumgebungen . . . . .	92
10.2.1.	Laufzeitumgebungen . . . . .	93
<b>11. Github Repositorien / Github Repositories</b>		<b>96</b>
11.1.	IRTLib Software . . . . .	96
11.1.1.	Download . . . . .	96
11.2.	CBA ItemBuilder . . . . .	96
11.2.1.	Download . . . . .	96

11.2.2. Source Code . . . . .	96
11.2.3. Dokumentation . . . . .	97
<b>12. Über / About</b>	<b>98</b>
12.1. Danksagung . . . . .	98
12.2. Entwicklung . . . . .	98

# 1. IRTlib Software

*IRTlib* ist eine Software zur Auslieferung computerbasierter Tests. Die Software besteht aus zwei Komponenten:

- *IRTlib Editor*: Eine Software für *Test-Autoren*, welche verwendet wird um *Studien* zu konfigurieren.
- *IRTlib Player*: Eine Software zur *Datenerhebung*, mit welcher *Zielpersonen* Aufgaben bearbeiten, die in Form einer *Studie* konfiguriert sind.

Hinweise zur Installation und Einrichtung beider Programme zur ersten Verwendung finden sich unter [Download & Installation](#).

Vor der Verwendung der *IRTlib Software* zur Konfiguration und Erstellung von Auslieferungen, müssen mit dem *CBA ItemBuilder* die Assessmentinhalte (Aufgaben, Instruktionen, Zwischenbildschirme usw.) in Form von einzelnen *Tasks* erstellt werden.

- Der *CBA ItemBuilder* kann hier heruntergeladen werden: [www.itembuilder.de/software](http://www.itembuilder.de/software)
- Eine interaktive Dokumentation des *CBA ItemBuilder* ist hier zugänglich: [cba.itembuilder.de](http://cba.itembuilder.de)

Vorgeschlagene Zitierung:

Kroehne, U. (2023). Open Computer-based Assessment with the CBA ItemBuilder. DIPF, Frankfurt am Main, Germany. <https://doi.org/10.5281/zenodo.10359757>

Die Entwicklung vom *CBA ItemBuilder* und der *IRTlib Software* wird vom *Zentrum für technologiebasiertes Assessment (TBA)* am *DIPF | Leibniz-Institut für Bildungsforschung und Bildungsinformation* koordiniert.

## 2. Download & Installation

Die *IRTlib*-Software wird für die Offline-Nutzung (z.Zt. für Windows-Betriebssysteme) und für die Online-Nutzung (in Form von *Docker-Containern*) bereitgestellt.

### 2.1. Offline (Windows)

Die *IRTlib*-Software (*IRTlib Editor* und *IRTlib Player*) für die Offline-Nutzung kann aus dem Abschnitt [Releases] des Repository <https://github.com/DIPFtba/IRTlibDeploymentSoftware> bezogen und heruntergeladen werden. Im Abschnitt [Releases](#) stehen zwei ZIP-Archive zum Download bereit.

- `TestApp.Editor.Desktop.exe`: Ist in `TestApp.Editor.Desktop.zip` und muss gestartet werden, um den *IRTlib Editor* zu verwenden.
- `TestApp.Player.Desktop.exe`: Ist in `TestApp.Player.Desktop.zip` enthalten und muss gestartet werden, um den *IRTlib Player* zu verwenden.

**i** Hinweis: Versionen aus der aktuellen Entwicklung als *Preview*.

Beachten Sie, dass der aktuellste Build im Abschnitt [Preview](#) des *Release*-Abschnitts des [repository](#) zu finden ist. Preview-Versionen sind die aktuellsten Version der Software, nach der letzten veröffentlichten Version der *IRTlib Software*. Um reproduzierbare Ergebnisse zu erhalten, sollten immer veröffentlichte Versionen verwendet werden.

**!** Hinweis: Warnmeldung beim Programmstart

Die automatisch erstellten Vorschauversionen des *IRTlib Editors* und *IRTlib Players* sind nicht signiert. Eine Warnmeldung des Betriebssystems muss akzeptiert werden, bevor die Programme ausgeführt werden können. Je nach Konfiguration des Betriebssystems können die ausführbaren Dateien der *IRTlib Software* auch als unbekannt eingestuft und vor ihrer Verwendung zusätzlich gewarnt werden.

### 2.1.1. Studienvorbereitung mit Offline Editor

Der *IRTlib Editor* für die Offline-Nutzung wird als ZIP-Archiv (z.B. `TestApp.Editor.Desktop.zip`) bereitgestellt, das entpackt werden muss. Nach dem Entpacken des Editors kann die Anwendung `TestApp.Editor.Desktop.exe` auf einem Windows-Gerät gestartet werden.

In den Abschnitten [Vorbereitung > Übersicht](#), [Vorbereitung > Studien](#) und [Vorbereitung > Erhebungsteile](#) ist dokumentiert, wie man mit Hilfe von *CBA ItemBuilder*-Items Datenerhebungen vorbereitet und konfiguriert.

### 2.1.2. Studiendurchführung mit Offline Player

Der *IRTlib Player* ist auch als Windows-Anwendung für die Offline-Nutzung verfügbar und wird als ZIP-Archiv (z.B. `TestApp.Player.Desktop.zip`) bereitgestellt. Nach dem Entpacken des *IRTlib Player* ist eine veröffentlichte Studienkonfiguration erforderlich, die zur Datenerhebung verwendet werden soll.

Nach dem Hinzufügen der als Studienkonfiguration bereitgestellten Inhalte einer veröffentlichten Studie kann die ausführbare Datei `TestApp.Player.Desktop.exe` gestartet werden (entweder mit oder ohne Startparameter).

- **Kiosk Mode:** Der *IRTlib Player* kann über die ausführbare Datei `TestApp.Player.Desktop.exe` auf dem Computer, auf dem er lokal ausgeführt wird, direkt zur Datenerhebung verwendet werden. Die *Studie* kann dazu so konfiguriert werden, dass es in einem *Kiosk Mode* auf einem Bildschirm angezeigt wird und nur über den *Task Manager* oder das *Testleitermenü* beendet werden kann (siehe *Vollbildmodus* im Abschnitt [Konfiguration zur Anzeige](#)).
- **Lokaler Server:** Der *IRTlib Player* kann aber auch als lokaler Server ausgeführt werden. Nach dem Start des Programms `TestApp.Player.Server.exe` kann eine konfigurierte *Studie* auch über *Webbrowser* oder andere Browser mit *Kiosk Mode* ausgeliefert werden (z.B. den [Safe Exam Browser](#)). Mit dieser Konfiguration lassen sich Datenerhebungen bspw. in Schulen ohne Internetverbindung aber mit einem als *Bring-in-Server* fungierenden Notebook durchführen.

In den Abschnitten [Datenerhebung > Übersicht](#), [Datenerhebung > Veröffentlichen & Export](#) und [Datenerhebung > Integration & Auslieferung](#) ist dokumentiert, wie man mit Hilfe des *IRTlib Players* in den unterschiedlichen Konstellationen Datenerhebungen durchführen kann.

## 2.2. Online (Docker)

Die *IRTlib*-Software (*IRTlib Editor* und *IRTlib Player*) für die Online-Nutzung kann als *Docker*-Container bezogen werden. Ein Beispiel ist unter <https://github.com/DIPFtba/IRTlibDeploymentSoftware> zu finden.

Um den Docker-Container zu verwenden, wird empfohlen, das Repository auf dem Zielgerät zu *klonen* und den Befehl `./start.sh` im Ordner `docker` auszuführen (erfordert installiertes `docker` und `docker compose`), um die Software zu starten.

Wenn in der Datei `docker-compose.yml` nichts geändert wird, ist der Editor über Port *8002* und die Player-Software über Port *8001* erreichbar.


In dem Abschnitt [Datenerhebung > Integration & Auslieferung](#) finden sich weitere Informationen zur Verwendung der *Docker*-Container.

## Teil I.

# Vorbereitung / Preparation

### 3. Vorbereitung: Übersicht / Preparation: Overview


Die Vorbereitung eines computerbasierten Assessments auf der Grundlage von *CBA ItemBuilder*-Inhalten beginnt mit der Verwendung des *IRTlib-Editors* zur Erstellung einer Studienkonfiguration. Dies umfasst in der Regel die folgenden Schritte:

 Optional: Verwenden eine *Runtime* für *CBA ItemBuilder* vor Version 9.9?

- **Voraussetzung:** Überprüfen Sie die Verfügbarkeit der [Runtime](#). Der *IRTlib Editor* kann zur Vorbereitung von Assessments mit im *CBA ItemBuilder* erstellten Inhalten verwendet werden. Für die Verwendung von *CBA ItemBuilder Tasks*, die in *Projektdateien* gespeichert sind, ist eine *Runtime* (d.h. die Dateien *main.\*.js* und *main.\*.css*) in der Version erforderlich, die genau der Version des *CBA ItemBuilders* entspricht, der zur Erstellung der Items verwendet wurde (z.B. 9.9.0). Bevor Sie den *IRTlib-Editor* verwenden, vergewissern Sie sich, dass die erforderliche *Runtime* enthalten ist, oder importieren Sie die *Runtime*-Dateien (siehe Abschnitt [Einstellungen](#) für Details).


Hinweis: Für die Verwendung von *CBA ItemBuilder*-Items ab Version 9.9 ist dieser Schritt in der Regel nicht notwendig.

- **Erstellen einer neuen Studie:** Der *IRTlib-Editor* wird verwendet, um sogenannte *Studien* zu konfigurieren. Die Versionen von Studien können im Editor nachverfolgt werden, Studien können dort veröffentlicht (d.h. für die Datenerfassung *versiegelt*) werden. Um mit dem *IRTlib-Editor* mit der Erstellung von Inhalten zu beginnen, muss zuerst eine Studie erstellt werden (siehe Abschnitt [Studien](#) für Details).

 Hinweis: Das Anlegen einer *Studie* ist immer notwendig.

Beachten Sie, dass mindestens eine *Studie* im *IRTlib Editor* definiert sein muss, bevor eine Studienkonfiguration zur Datenerfassung mit einem *IRTlib Player* verwendet werden kann.

- **Basiskonfigurationen für Studie festlegen (Info):** Zu den Basiskonfigurationen, die sich auf den Inhalt einer vorbereiteten Studie beziehen, gehören die Studienbezeichnung und -beschreibung, der Anmeldemodus, die Anzeigekonfiguration, das Menü für die Testadministratoren und die Angabe, wie nach Abschluss aller in einer Studie definierten Inhalte fortgefahren werden soll (siehe Abschnitt [Studien](#) für weitere Einzelheiten).
- **Erstellen eines neuen Erhebungsteils:** Jede *Studie* besteht aus einem oder mehreren *Erhebungsteilen*. *Erhebungsteile* werden als Bausteine von Assessments betrachtet, die zusammen verwaltet werden, wie z.B. Items aus einer bestimmte Domäne. *Erhebungsteile* vom Typ *CBA ItemBuilder* können verwendet werden, um *CBA ItemBuilder*-Aufgaben in einer linearen Sequenz oder mit *Blockly*-basiertem Routing zu administrieren.

 Hinweis: Das Anlegen eines *Erhebungsteils* ist immer notwendig.

Beachten Sie, dass jede *Studie* mindestens einen im *IRTlib Editor* definierten *Erhebungsteil* benötigt, bevor eine Studienkonfiguration zur Datenerhebung mit einem *IRTlib Player* verwendet werden kann.



- **Grundeinstellungen für Erhebungsteil konfigurieren (Info):** Ein *Erhebungsteil* vom Typ *CBA ItemBuilder* basiert auf einer Menge von *CBA ItemBuilder-Tasks*. Jede *CBA ItemBuilder-Projektdatei* benötigt mindestens einen *Task*, es werden aber auch Projekte mit mehreren *Tasks* unterstützt. Wenn *CBA ItemBuilder-Inhalte* mit einem gemeinsamen Zeitlimit über *Tasks* hinweg administriert werden sollen, erlauben *Erhebungsteile* die Zuordnung von Aufgaben zu einer Struktur, die Assessmentinhalte unterscheidet, welche *vor* einem zeitlich begrenzten Abschnitt administriert werden (z.B. Instruktionen, im Abschnitt *Vorspann-Items*), Inhalte die *nach* einem zeitlich begrenzten Abschnitt administriert werden (z.B. Danksagung, im Abschnitt *Nachspann-Items*) und dazwischen liegende Aufgaben mit begrenzter Zeit (*Items*, siehe Abschnitt [Erhebungsteile](#)).
- **Items Hinzufügen:** Um die Definition eines *Erhebungsteils* abzuschließen, müssen die *CBA ItemBuilder-Projektdateien* in den Abschnitt *Items* importiert werden. Standardmäßig wird davon ausgegangen, dass die Reihenfolge der *CBA ItemBuilder-Tasks* linear ist. Wenn jedoch das *Routing* für einen Studienteil aktiviert ist, kann das *Blockly*-basierte Ablaufdefinition verwendet werden, um verschiedene Testdesigns zu implementieren (z. B. mehrere Hefte, mehrstufige Tests usw.).

### 3.1. Eingebettete Programmhilfe

Für die Verwendung des *IRTLib Editors* ist eine Programmhilfe direkt in die Anwendung integriert, welche über das kleine ?-Symbol oben rechts eingeblendet werden kann.

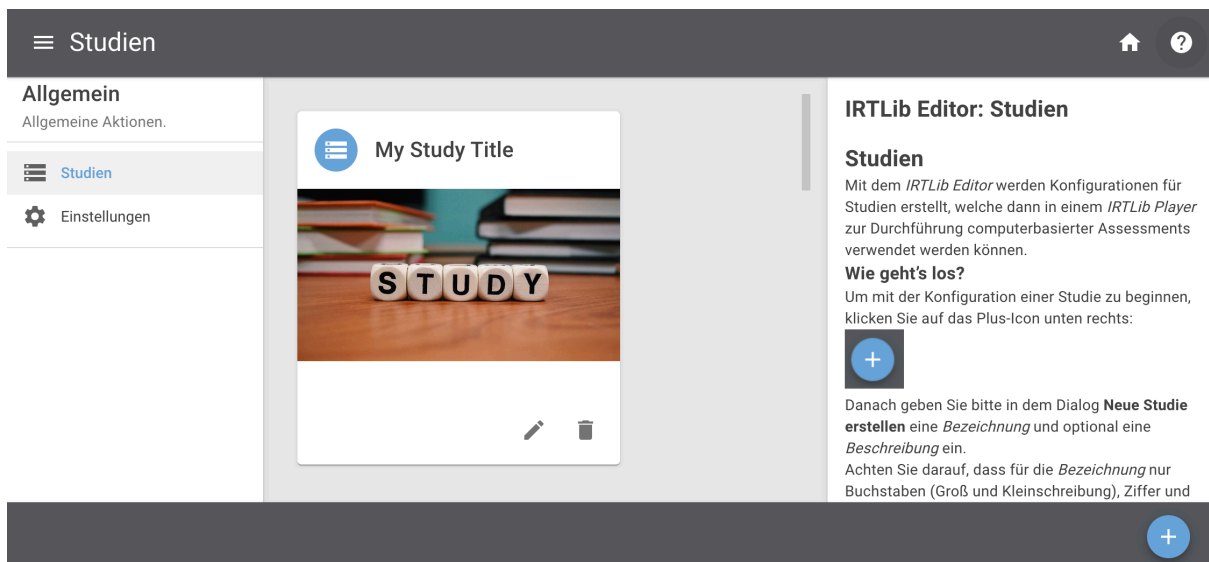


Abbildung 3.1.: Beispiel des *IRTLib Editors* mit eingeblendeter Hilfe

#### 💡 Eingebettete Programmhilfe

Die Inhalte dieser Hilfe-Seiten aus dem *IRTLib Editor* sind in diese *IRTLib Dokumentation* integriert und werden immer in diesem Rahmen mit der Überschrift *Eingebettete Programmhilfe* dargestellt.

### 3.2. Vorbereitung von CBA ItemBuilder-Inhalten

Die *IRTLib Software* wird benötigt, um die mit dem *CBA ItemBuilder* erstellten Assessmentinhalte anzuzeigen und für Datenerhebungen zu verwenden. Dafür müssen die *Projektdateien* (ZIP-Archive), die mit dem *CBA ItemBuilder* erstellt werden können, vorliegen.

### 3.2.1. Einsprungpunkt (Task)

Jede *CBA ItemBuilder-Projektdatei* muss mindestens einen *Task* definieren. Nur *Tasks* können in der *IRTlib Software* verwendet werden. Dass ein *Task* vollständig definiert ist, lässt sich in der *Preview* des *CBA ItemBuilder* leicht überprüfen:

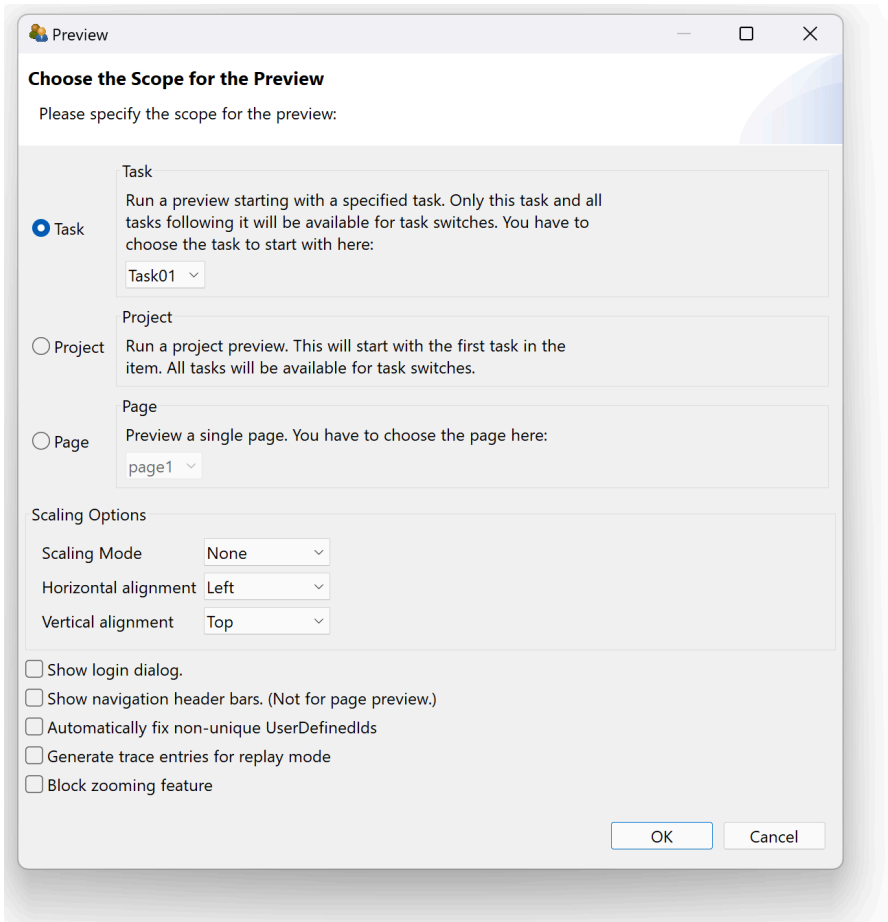


Abbildung 3.2.: Beispiel aus *CBA ItemBuilder* zur Anforderung einer *Preview* für einen *Task*

**i** Die *IRTlib Software* benötigt definierte *Tasks*.

Assessmentinhalte können mit der *IRTlib Software* aus einzelnen *Task* zusammengesetzt werden. Die Reihenfolge von *Tasks* kann statisch als *lineare Sequenz* oder als programmierter Ablauf in *Blockly* definiert werden. Aus der *IRTlib Software* heraus können keine einzelnen Seiten innerhalb von *CBA ItemBuilder-Tasks* angesteuert werden.

*CBA ItemBuilder-Projektdateien* welche nur über die Option *Project* oder *Page* in der *Preview* angezeigt werden können, lassen sich in der *IRTlib Software* nicht verwenden.

### 3.2.2. Verhalten der Anzeige (Scaling Options)

Mit Hilfe der *Preview* des *CBA ItemBuilder* kann auch geprüft werden, ob die Assessmentinhalte in der gewünschten Skalierung, die unter *Scaling Options* eingestellt werden kann, den Erfordernissen entsprechend angezeigt werden.

Einstellungen analog zur *Preview* können in dem *IRTlib Editor* für die Anzeigeeinstellungen einer *Studie* definiert werden (siehe Abschnitt [Studien](#)).

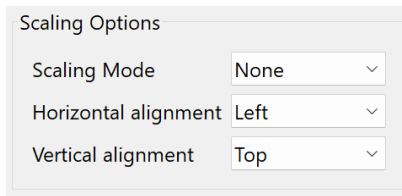


Abbildung 3.3.: Einstellungen zu *Scaling Options* aus *CBA ItemBuilder-Preview*

### 3.2.3. Definition des Scorings (Ergebnisdaten)

Die *IRTlib Software* ist dafür gedacht, Daten mit Hilfe von *CBA ItemBuilder-Task* zu erheben. Was aus der Bearbeitung eines *Tasks* an Ergebnisvariablen gespeichert wird, können *Itemautoren* in der *Scoring-Definition* eines *Tasks* definieren.

**i** Das *Scoring* muss bereits im *CBA ItemBuilder* definiert werden.

Aus der Bearbeitung von *Tasks* werden die als *Classes* definierten Ergebnisvariablen gespeichert, deren Werte entweder einzelne *Hits* sein können, oder die Übernahme von Informationen mit Hilfe des sogenannten *ResultText*-Operators im *CBA ItemBuilder*.

Mit Hilfe des eingebauten *Scoring Debug Window* sollte das *Scoring* einzelner *CBA ItemBuilder-Tasks* bereits im *CBA ItemBuilder* geprüft werden, bevor die Assessmentinhalte mit Hilfe der *IRTlib Software* zu *Studien* mit einem oder mehreren *Erhebungsteilen* kombiniert werden. Weitere Hinweise zu vorgeschlagenen Prüfungen sind im Abschnitt [Auslieferungen Testen und Freigeben](#) formuliert.

**i** Log-Daten werden automatisch gesammelt.

Ohne weitere Konfiguration werden Log-Daten in den mit dem *CBA ItemBuilder* erstellten Assessmentinhalten automatisch erfasst und über die *IRTlib Software* gesammelt.

### 3.2.4. Integrierte Multimedia-Inhalte (Ressourcen)

Die mit dem *CBA ItemBuilder* erstellten Assessmentinhalte können Multimedia-Inhalte (Bilder, Videos, Audiodateien) enthalten. Bilder und Videos werden dabei in einer Größe angezeigt, die in der entsprechenden Komponente des *CBA ItemBuilder* im *Page Editor* verwendet wird. In den *Projektdateien* werden Bilder, Videos und Audiodateien als *Ressourcen* gespeichert, sobald diese über den *Ressource Browser* eingefügt wurden. Nicht verwendete *Ressourcen* bleiben dabei in den *Projektdateien* enthalten.

**i** Dateigröße von *CBA ItemBuilder-Projektdateien* sollte möglichst klein sein.

Die Dateigröße von *CBA ItemBuilder-Projektdateien* ist für die Verwendung insbesondere in Online-Auslieferungen relevant und sollte so klein wie möglich gehalten werden.

Vor der Verwendung von *CBA ItemBuilder-Projektdateien* wird empfohlen, folgende Punkte zu berücksichtigen:

- Bilder und Videos nur in der benötigten Größe: Bilder und Videos können ohne Qualitätsverlust auf die Größe (Breite/Width und Höhe/Height) reduziert werden, in welcher sie in *CBA ItemBuilder-Projekten* auch wirklich verwendet werden.
- Bilder wenn möglich komprimieren: Ohne die Bildgröße zu verändern können Bilder häufig in der Dateigröße weiter reduziert werden.

- Videos wenn möglich komprimieren: Ohne die Video zu verändern können Videos häufig in der Dateigröße weiter reduziert werden.
- Audios nicht in höchster Qualität: Wenn nicht notwendig, sollten Audio-Dateien in ihrer Qualität soweit reduziert werden, dass sie noch akzeptabel klingen, aber bzgl. der Übertragungszeiten optimiert sind.
- Entfernen nicht verwendeter Ressourcen: Der *CBA ItemBuilder* stellt eine Schaltfläche im *Resource Browser* zur Verfügung, um nicht verwendeten *Ressourcen* automatisch zu entfernen. Diese Funktion sollte abschließend genutzt werden, damit die *Projekt-Dateien* keine unnötigen *Ressourcen* enthalten.

Ein sorgsamer Umgang mit *Ressourcen* und eine Optimierung der Dateigröße von *CBA ItemBuilder-Projektdateien* kann entscheidend für ein flüssiges und reibungsfreies Assessment sein, was sich mit der *IRTlib Software* online ausliefern und bspw. über Mobilgeräte mit beschränkter Internetbandbreite nutzen lässt.

## 4. Vorbereitung Studien / Preparation Studies

Konfigurationen, die mit dem *IRTLib-Editor* erstellt werden, werden in sogenannten *Studien* zusammengefasst. Eine *Studie* soll die Assessmentinhalte zusammenfassen, welche in einer Erhebung oder Sitzung administriert werden.

### 4.1. Studienverwaltung

Nach dem Start des *IRTLib-Editors* wird die Ansicht *Studien* angezeigt. In dieser Ansicht ist der erste Schritt zur Vorbereitung einer neuen Konfiguration das **Hinzufügen einer neuen Studie**:

<https://youtu.be/7VKf6U3oeM4>

Die erstellten *Studien* erscheinen als Karten in der Ansicht *Studien*. Beachten Sie, dass die Reihenfolge, in der die Studien in der *Studienansicht* angezeigt werden, keine Rolle spielt.

Eine detaillierte Anleitung zur Erstellung einer *Studie* findet sich hier in der eingebetteten Hilfe:

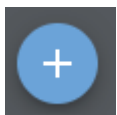
 Eingebettete Programmhilfe

#### 4.1.1. Studien Anlegen

Mit dem *IRTLib Editor* werden Konfigurationen für Studien erstellt, welche dann in einem *IRTLib Player* zur Durchführung computerbasierter Assessments verwendet werden können.

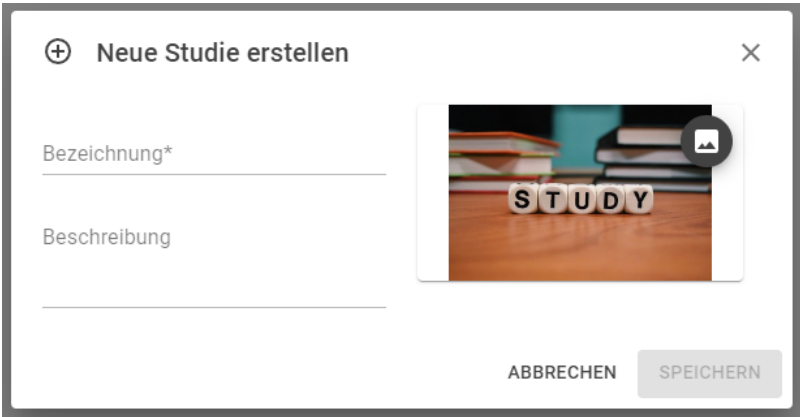
##### 4.1.1.1. Wie geht's los?

Um mit der Konfiguration einer Studie zu beginnen, klicken Sie auf das Plus-Icon unten rechts:



Danach geben Sie bitte in dem Dialog **Neue Studie erstellen** eine *Bezeichnung* und optional eine *Beschreibung* ein.

Achten Sie darauf, dass für die *Bezeichnung* nur Buchstaben (Groß- und Kleinbuchstaben), Ziffern und ein `_` erlaubt sind.



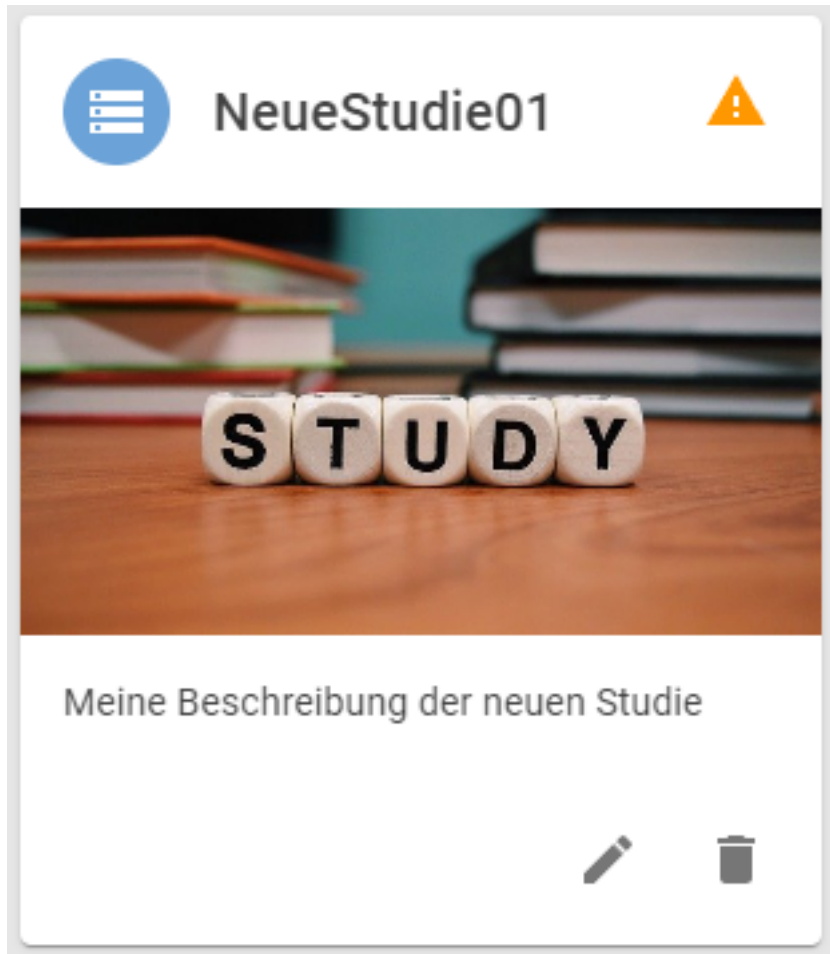
Klicken Sie anschließend auf *Speichern*.

Bei Bedarf können Sie über das folgende Icon einer Studie auch ein Bild zuordnen. Dieses Bild wird im *IRTLib Editor* für diese Studie verwendet:



#### 4.1.1.2. Wie geht's weiter?

Erstellten Studien werden in der Studienübersicht als Kacheln angezeigt:



Um nun mit der Erstellung und Konfiguration einer Studie fortzufahren, klicken Sie auf das kleine Bearbeiten-Icon:

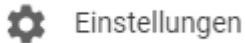


#### 4.1.2. Weitere Funktionen und Hinweise

- **Studie Löschen:** Mit dem Papierkorb-Icon können Sie Studien auch wieder löschen. Das Löschen von Studien kann nicht rückgängig gemacht werden:



- **Sprache Wechseln:** Über den Menüpunkt *Einstellungen* gelangen Sie zu dem Punkt *Allgemeine Einstellungen*, wo sie die Sprache des *IRTLib Editors* ändern können.

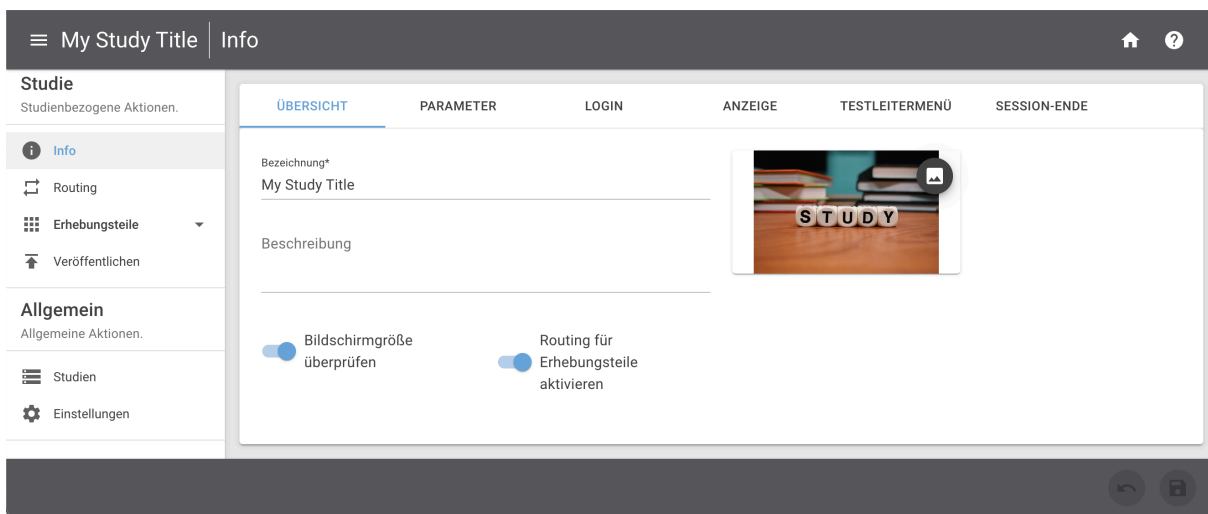


Über diesen Punkt erhalten Sie auch Zugriff auf die im *IRTLib Editor* verfügbaren *CBA ItemBuilder Runtimes* (Unterstützung für die Verwendung von *CBA ItemBuilder*-Inhalten, die mit unterschiedlichen Versionen des Programms erstellt wurden).

## 4.2. Grundlegende Konfigurationen

Die Konfigurationen einer bestimmten Studie, einschließlich Versionierung und Veröffentlichung, werden innerhalb von Studien verwaltet (d.h. nach dem Öffnen einer Studie zur Bearbeitung durch Klicken auf das Bearbeitungssymbol am unteren rechten Rand der Karte).

Erstellte Studien, die im *IRTLib Editor* in der Ansicht *Studien* angezeigt werden, können zur Bearbeitung geöffnet werden.



Detaillierte Informationen zur Grundkonfiguration einer Studie finden sich hier in der eingebetteten Hilfe:

### Eingebettete Programmhilfe

#### 4.2.1. Einstellungen zur Studie

- **Bezeichnung:** Wie soll die *Studie* benannt werden? Achten Sie darauf, dass für die Bezeichnung nur Buchstaben (Groß- und Kleinbuchstaben), Ziffern und ein `_` erlaubt sind.
- **Beschreibung:** Um eine ausführliche Beschreibung der *Studie* hinterlegen zu können, ist dieses optionle Feld vorgesehen. Hier können auch Sonderzeichen und Umlaute usw. eingegeben werden.
- **Routing für Erhebungsteile aktivieren:** *Studien* bestehen aus einem oder mehreren

*Erhebungsteilen*. Die *Erhebungsteile* werden per Default als lineare Abfolge administriert. Wenn die Option *Routing für Erhebungsteile aktivieren* ausgewählt ist, kann die Reihenfolge der *Erhebungsteile* mit einem *Blockly*-basierten Routing definiert werden. Dadurch sind dynamische Abfolgen von *Erhebungsteilen* möglich, wobei auch Aufrufparameter der Studie bspw. für die Zuordnung von unterschiedlichen Reihenfolgen genutzt werden können.


- **Bildschirmgröße überprüfen:** In Erhebungen, bei denen die Bildschirmgröße nicht bekannt ist, kann mit dieser Option ein Größenvergleich von Objekten (EC-Karte, Banknote, Personalausweis) mit Darstellungen auf dem Bildschirm vorgenommen werden.

Die Geräteprüfung erfolgt mit folgendem Dialog:

Geräteprüfung

Bitte stellen Sie die Größe Ihres Bildschirms mit dem unten abgebildeten Schieberegler ein. Dazu müssen Sie das angezeigte Objekt (Lineal, Personalausweis oder 10€-Schein) als Vergleichsobjekt bei Ihnen vorliegen haben und mit der Anzeige abgleichen. Wählen Sie dazu unten das Ihnen vorliegende Objekt aus. Bitte bestätigen Sie Ihre Einstellung abschließend mit dem Klicken auf den „Weiter“-Button.

Lineal  Personalausweis, Bankkarte o.ä.  10-Euro-Banknote WEITER



Wenn die Option *Passende Bildschirmgröße erzwingen* (im Abschnitt *Anzeige*) nicht aktiviert ist, dann kann die Testbearbeitung dennoch begonnen werden. Es wird, wenn die Auflösung zu klein ist, folgender Dialog angezeigt:

Geräteprüfung

[Die Aufgabenbearbeitung kann mit diesem Gerät grundsätzlich durchgeführt werden.](#)

Wenn Sie ein anderes Gerät nehmen können oder einen anderen Browser (wir empfehlen Ihnen Google Chrome, Firefox oder Safari) aufrufen können, würde das die Darstellung der Aufgaben noch verbessern. Wenn das nicht möglich ist, gehen Sie bitte über das Kästchen unten weiter.

Bitte klicken Sie das Kästchen links an, wenn Sie mit diesem Gerät fortfahren möchten.

WEITER

Hinweis: Diese Option ist zur Zeit nicht weiter konfigurierbar.

Wenn geänderte Einstellungen erhalten bleiben sollen müssen die Änderungen über das Disketten-Symbol gespeichert werden. Andernfalls kann das Verwerfen-Symbol verwendet werden:



### 4.3. Zugang zu Studien (Login)

Die *IRTlib Software* unterstützt verschiedene Wege, wie sich Personen (Testteilnehmer, Befragte, ...) für ein Assessment authentifizieren können. Die Konfigurationen umfassen zwei Aspekte:



- *Login-Modus*: Wird ein Zugang benötigt (Login, Login+Passwort, Passphrases/Token) oder nicht? Und wenn Zugangsdaten benötigt werden, was sind *gültige Werte*?
- *Loginquelle*: Wie wird die Login-Information abgefragt (direkte Eingabe auf der Plattform, CBA ItemBuilder Item, ...) oder übergeben (Login-Parameter oder Aufruf-Parameter)?

Detaillierte Informationen zu der Konfiguration der Anmeldung einer *Studie* finden sich hier in der eingebetteten Hilfe:

#### Eingebettete Programmhilfe

### 4.3.1. Konfiguration der Anmeldung

Im Abschnitt *Login* kann konfiguriert werden, wie Testteilnehmer, die ein Assessment starten (entweder durch Aufruf eines Links in einem Browser der auf den Online-*IRTlib Player* verweist oder durch Start des Offline-*IRTlib Players*), identifiziert oder authentifiziert werden sollen.

- **Login-Modus**: Die *IRTlib Software* unterstützt verschiedene *Login-Modi*.
  - *Zufälliger Inditifikator*: Wenn eine Sitzung zum ersten Mal gestartet wird, wird in diesem *Login-Modus* ein Identifikator generiert. Diese zufällige, aber eindeutige Zeichenkette (eine sogenannte *UUID*, d.h. ein *Universally Unique Identifier*) wird als Personenidentifikator in allen Daten (d.h. Ergebnisdaten) und allen anderen gespeicherten Daten (z.B. Log-Daten/Trace-Daten, Snapshot-Daten, etc.) verwendet.
  - *Benutzername*: Wenn von den Testteilnehmern erwartet wird, dass sie sich durch eine eindeutige Zeichenfolge identifizieren (z.B. eine Zahl oder ein Text, der als Zugangskennung verwendet wird), kann eine *Studie* mit dem *Login-Modus Benutzername* konfiguriert werden. Der Zugang zum Assessment ist dann nur möglich, wenn die als *Benutzername* eingegebene Zeichenkette gültig ist. Die zugrundeliegende Idee ist, dass die Studienkonfiguration mit einer Liste gültiger Benutzernamen geladen wird und dass ein Testteilnehmer einen gültigen Benutzernamen eingeben muss, bevor er oder sie das Assessment starten kann. Nur authentifizierte Testteilnehmer können auf die als *Studie* definierten Assessmentinhalte zugreifen und die Aufgaben bzw. Fragen beantworten.
  - *Benutzername und Passwort*: Wenn in einer *Studie* nicht nur gültige Benutzernamen, sondern ein Passwort zur Authentifizierung der Testpersonen verwendet werden soll, ermöglicht der *Login-Modus Benutzername und Passwort* eine Eingabe von Benutzernamen und Passwort. Analog zu *Benutzername* müssen dann beide Informationen in der Studienkonfiguration hinterlegt sein.
  - *Zugriffstoken*: Wenn in der Studienkonfiguration die gültigen Benutzernamen nicht gespeichert werden sollen, kann die Option *Zugriffstoken* verwendet werden. Jedes Token, das einem definierten Schema entspricht, wird dann akzeptiert und als Identifikator für die Testteilnehmer verwendet.
- **Speicher für Sessiondaten**: Bei Onlineauslieferungen kann nach einer Unterbrechung ein Assessment fortgesetzt werden. Diese Funktionalität wird bspw. auch benötigt, wenn im Browser die Seite neu geladen wird (bspw. durch Erzwingen eines *Reload/F5*, oder durch Schließen und erneutes Aufrufen der URL). Um sicherzustellen, dass Sitzungen, die von derselben Person (d.h. vom selben Browser) stammen, auch fortgesetzt werden können, kann die Software so konfiguriert werden, dass der Identifikator im Client gespeichert wird.
- **Gültige Werte**: Die *IRTlib Software* bietet für die *Login-Modi Benutzername, Benutzername + Passwort* und *Zugriffstoken* folgende Mechanismen zur Validierung von Anmeldeinformationen:
  - *Liste*: Eine Liste gültiger Berechtigungen (Benutzername oder Benutzername und Passwort, je nach Konfiguration des *Login-Modus*) kann als Teil der

Studienkonfiguration definiert werden. Die Informationen können entweder im *IRTlib Editor* bearbeitet oder aus einer CSV-Datei importiert werden. Definierte Werte können auch als CSV-Datei exportiert werden.

- *Code zur Prüfung*: Es kann eine *Blockly*-Funktion angegeben werden, welche *Wahr* zurückmeldet, wenn die übergebenen Anmeldeinformationen gültig sind (sonst *Falsch*).
- **Gruppenlogin**: Je nach *Login-Modus* dienen Benutzername oder Zugriffstoken als Personenidentifikator. Wenn die Option *Gruppenanmeldung* aktiviert wird, dann werden diese übergebenen Anmeldeinformationen zur Authentifizierung verwendet, um den Testteilnehmer als Mitglied einer Gruppe zu identifizieren (d. h. nur Testteilnehmer, die den Benutzernamen kennen, können sich als Teil der Gruppe authentifizieren). Innerhalb der Gruppe wird dann ein zusätzlicher Zufallsidentifikator generiert, um verschiedene Personen aus einer Gruppe zu unterscheiden.
- **Loginquelle**: Die *IRTlib Software* unterstützt verschiedene mögliche Optionen, wie Anmeldeinformationen bereitgestellt werden können.
  - *Plattform*: Ein Anmeldedialog wird vom *IRTlib Player* (d.h. der Plattform) angezeigt. Die Überschrift zur Eingabe der Zugangsdaten, die Beschriftung der Eingabe für Benutzernamen und Passwort, die Beschriftung des Weiter-Buttons, ein Begrüßungs- und ein Instruktionstext sowie ein Fehlertext bei fehlgeschlagenen Login-Versuchen können konfiguriert werden.
  - *Parameter*: Gültige Anmeldeinformationen für Testteilnehmer können auch über die *Befehlszeile* (d.h. Parameter beim Aufruf der Offline-Version des *IRTlib Players*) oder über URL-Parameter (d.h. Parameter beim Aufruf der *Studie* über einen Link auf eine Online-Version des *IRTlib Players*) bereitgestellt werden. In diesem Fall wird kein Anmeldedialog oder Login-Item angezeigt.
  - *Item*: Alternativ zu einem Dialog des *IRTlib Players* kann auch ein *CBA ItemBuilder*-Task konfiguriert werden, der als Login-Eingabemaske dient. Innerhalb des Items wird ein sogenannter *ExternalPageFrame* verwendet, um zur Validierung einer Eingabe einen bestimmten JavaScript-Befehl an den *IRTlib Player* zu senden (ein Beispiel finden Sie [hier](#)).

Das Login Item muss als *CBA ItemBuilder*-Projektdatei für die konfigurierte Laufzeitumgebung (Runtime) verfügbar sein und der Studienkonfiguration hinzugefügt werden. Um ein Login-Item zur Studienkonfiguration hinzuzufügen, kann der integrierte Importdialog verwendet werden. Mehr Informationen zum Importieren von *CBA ItemBuilder*-Projekten findet sich in der Hilfe zum Abschnitt *Items* eines *Erhebungsteils*.

- **Zusätzliche Parameter**: Neben der *Authentifizierung* von Testteilnehmern können die Anmeldeinformationen in der *IRTlib Software* auch als zusätzlicher Parameter hinterlegt werden, die dann bspw. in der Ablaufsteuerung verwendet werden können.
  - Parameter für Dateinamen: Der *RawDataPath* (d.h. der relative Pfad unter dem der Offline-*IRTlib Player* die Ergebnisdaten speichert) und der *MonitoringFile* (d.h. der Name der Datei in welcher der Offline-*IRTlib Player* Informationen fürs Studienmonitoring schreibt) können als Teil der Anmeldeinformationen konfiguriert werden.
  - *Blockly*-Variablen: Zusätzliche Parameter können auch als sogenannte *Preload*-Variablen mit den Anmeldeinformationen hinterlegt werden.

Tabelle 4.1.: Zusammenfassung der Optionen, die als *Konfiguration der Anmeldung* kombiniert werden können

Login Modus	Speicher für Sessiondaten	Gruppenlogin	Gültige Werte	Loginquelle	Zusätzliche Parameter
Zufälliger Inditifikator	ja	nein	nein	keine	nein
Benutzername	ja	ja	Liste oder Code	Plattform, Item + Parameter	Werte oder Parameter
Benutzername und Passwort	ja	ja	Liste oder Code	Plattform, Item + Parameter	Werte oder Parameter
Zugriffstoken	ja	ja	Schema oder Code	Plattform, Item + Parameter	Parameter

## 4.4. Anzeige von Assessment-Inhalten

*Studien* können festlegen, wie der *CBA ItemBuilder*-Inhalt dargestellt werden soll. Die Einstellungen im Abschnitt *Anzeige* können sich auf die Skalierung und die Ausrichtung der Inhalte sowie auf das Verhalten der *IRTlib Player*-Anwendung beziehen.

Detaillierte Informationen zu der Konfiguration der *Anzeige* einer *Studie* finden sich hier in der eingebetteten Hilfe:

### Eingebettete Programmhilfe

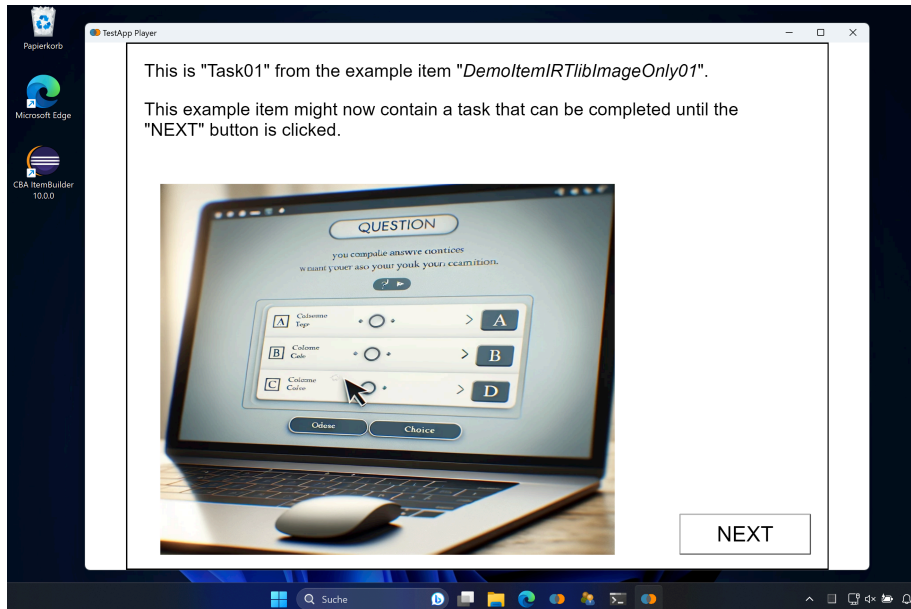
#### 4.4.1. Anzeigeeinstellungen

Für die Konfiguration der Anzeige stehen ausgewählte Optionen zur Verfügung, die sich auf die Darstellung der Assessmentinhalte und die Verwendung von *CBA ItemBuilder*-Inhalten beziehen, welche mit einem festgelegten Seitenverhältnis (Breite und Höhe) erstellt werden.

##### 4.4.1.1. Fenstermodus

In der Auswahl **Fenstermodus** kann konfiguriert werden, ob ein zusätzliches Fenster im *IRTlib Player* angezeigt wird. Die Konfiguration wird je nach Umgebung unterschiedlich umgesetzt:

- *Window*: In diesem *Fenstermodus* wird im *Offline-IRTlib Player* ein reguläres Programmfenster verwendet, im *Online-IRTlib Player* wird der Assessmentinhalt im normalen Browserbereich angezeigt, und die Adressleiste und die Navigations Schaltflächen des Browsers sind in diesem Modus sichtbar.



- **Vollbild:** Der Offline-IRTlib Player startet direkt im Vollbildmodus, wenn diese Option konfiguriert ist. Damit verbunden ist auch ein *Kiosk-Modus*, d.h. der Zugriff auf andere Programme und das (versehentliche) Beenden des Programms ist nur über den *Task Manager* möglich. Soll die Möglichkeit zum Beenden der Testung bspw. für einen Testleiter möglich sein, muss ein *Testleitermenü* konfiguriert sein.

Der Online-IRTlib Player kann Assessmentinhalte auch im Vollbildmodus anzeigen, wenn diese Option gewählt ist. Wird der Vollbildmodus im Browser verlassen, wird dann der Assessmentinhalt ausgeblendet. Da automatisiert nicht in einem Browser in den Vollbildmodus gewechselt werden kann, sieht die Zielperson zunächst folgende Nachricht der Plattform:

**Achtung - Wichtiger Hinweis**

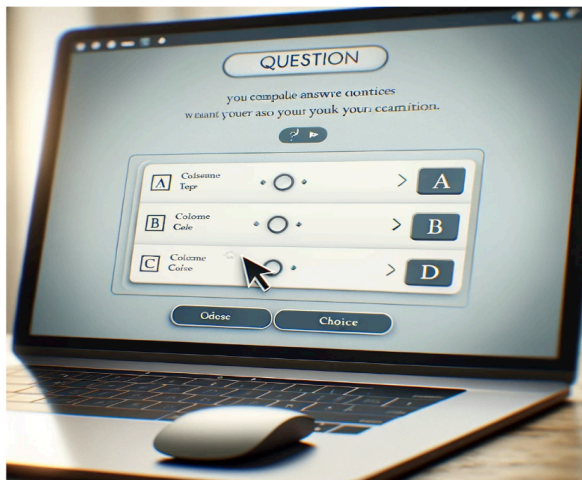
Der Test kann nur im Vollbild durchgeführt werden. Wenn Sie im Vollbildmodus ihres Browsers sind, beenden Sie diesen (z.B. mit ESC oder F11) und klicken Sie dann auf den folgenden Button, um mit dem Test fortfahren zu können.

[VOLLBILD AKTIVIEREN](#)

Durch klick auf den Button *Vollbild Aktivieren* wird der Vollbildmodus gestartet und der Assessmentinhalt dann ohne Fensterrahmen und Navigationsflächen des Browsers dargestellt. Für kurze Zeit wird dann von den Browser typischerweise ein Hinweis eingeblendet dass mit Esc der Vollbildmodus wieder beendet werden kann.

This is "Task01" from the example item "DemoltemIRTLlibImageOnly01".

This example item might be displayed in full-screen mode until the "NEXT" button is clicked.



NEXT

Beachten Sie, dass diese Funktion nur in Browsern zur Verfügung steht, die den Vollbildmodus unterstützen (insbesondere auf älteren mobilen Geräten wird der Vollbildmodus nicht vollständig unterstützt; siehe für Details z.B. auf [caniuse.com](http://caniuse.com)).

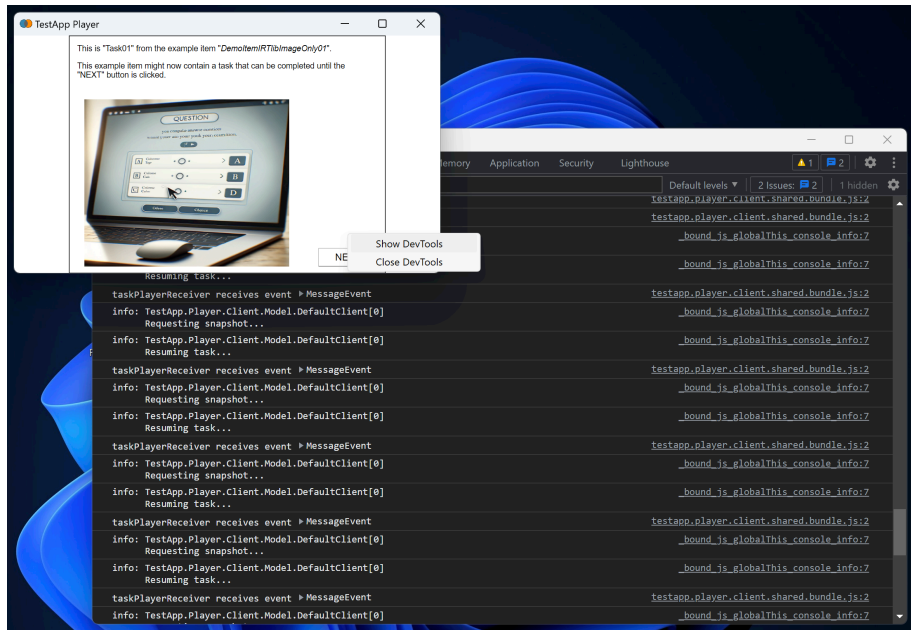
- *Vollbild, wenn unterstützt:* In diesem Modus wird das Assessment im Online-IRTLib Player nur dann im Vollbildmodus angezeigt, wenn der Browser den Vollbildmodus unterstützt. Der Inhalt des computerbasierten Assessments wird jedoch im Fenstermodus angezeigt, wenn eine Studie online bereitgestellt wird und ein Browser verwendet wird, der den Vollbildmodus nicht unterstützt. Für den IRLib Player offline ist diese Konfiguration identisch mit *Vollbild*.

#### Achtung - Wichtiger Hinweis

Der Test kann nur im Vollbild durchgeführt werden. Leider kann die Aufgabenbearbeitung auf diesem Gerät nicht durchgeführt werden, da der Browser kein Vollbildmodus unterstützt. Bitte prüfen Sie, ob Sie ein anderes Gerät (Computer oder Laptop) verwenden können!

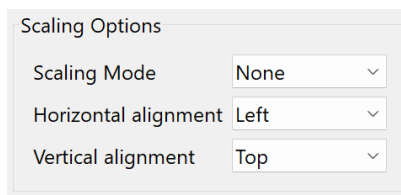
- *Debug:* Dieser Modus ermöglicht während der Testausführung den Zugriff auf die Entwicklerwerkzeuge des Browsers, die für die Fehlersuche von Softwareentwicklern vorgesehen sind.

Wenn der Offline-IRTLib Player mit einer Studie gestartet wird, welche als *Fenstermodus* den Eintrag *Debug* konfiguriert hat, dann lassen sich über die rechte Maustaste während der Aufgabenpräsentation die sogenannten Entwicklertools (*DevTools*) abrufen.



#### 4.4.2. Skalierung und Ausrichtung

Assessmentinhalte, die mit dem *CBA ItemBuilder* erstellt werden, werden für eine konkrete Größe in Pixeln (Breite mal Höhe) erstellt. Für die Darstellung auf Geräten mit unterschiedlichen Bildschirmgröße und Bildschirmauflösungen können die Inhalte dann skaliert werden. Im *CBA ItemBuilder* stehen deshalb in der *Preview* die Option unter *Scaling Options* zur Verfügung:

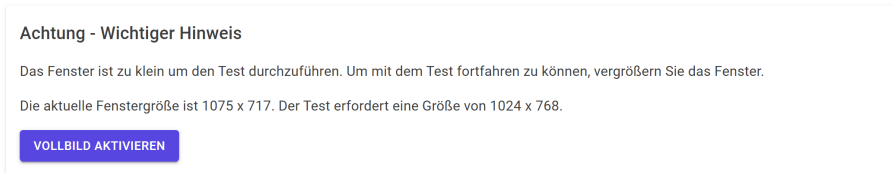


Im *IRTlib Editor* können analoge Einstellung vorgenommen werden.

- **Skalierung:** Einstellung wie Inhalte sollen angepasst werden, wenn verfügbarer Platz und Größe der Items abweichen (*Scaling Mode*).
  - *keine:* Die Inhalte werden ohne Anpassung an die verfügbare Fenster- bzw. Bildschirmgröße angezeigt (entspricht None).
  - *hochskalieren:* Inhalte werden vergrößert, damit der verfügbare Platz ausgenutzt wird (entspricht Up).
  - *runterskalieren:* Inhalte werden verkleinert, damit sie auf den Bildschirm/ins Fenster passen (entspricht Down).
  - *Fenstergröße:* Inhalte werden vergrößert und verkleinert (entspricht Both).
- **Horizontale Ausrichtung:** Die Optionen *zentriert* / *links* / *rechts* werden genutzt um Iteminhalte horizontal auszurichten, wenn die Breite von Fenster oder Bildschirm größer ist als die Breite des Inhalts.
- **Vertikale Ausrichtung:** Die Optionen *zentriert* / *oben* / *unten* werden genutzt um Iteminhalte vertikal auszurichten, wenn die Höhe von Fenster oder Bildschirm größer ist als die Höhe des Inhalts.

#### 4.4.2.1. Weitere Einstellungen

- **Passende Bildschirmgröße erzwingen:** Wenn bei *Skalierung* nicht *runterskalieren* oder *Fenstergröße* ausgewählt ist, kann über diese Option erzwungen werden, dass man nur dann mit der Aufgabenbearbeitung starten kann, wenn die verfügbare Größe des Fensters bzw. Bildschirms größer ist als die benötigte Breite/Höhe der Items. Andernfalls wird folgende Nachricht angezeigt:



Hinweis: Die Einstellungen zur Anzeige beziehen sich auf alle *Erhebungsteile* innerhalb einer *Studie*. Werden in einem *IRTlib-Player* mehrere Studien konfiguriert, müssen die Einstellungen zueinander passen, d.h. es ist nicht möglich mit einer Instanz eines *IRTlib-Players* gleichzeitig eine Studie im *Fenstermodus: Fenster* oder im *Fenstermodus: Vollbild* zu administrieren.

Wenn geänderte Einstellungen erhalten bleiben sollen, müssen die Änderungen über das Disketten-Symbol gespeichert werden. Andernfalls kann das Verwerfen-Symbol verwendet werden:



## 4.5. Menü für Testadministratoren

Wenn die Durchführung von Assessments begleitet durch Testleiter oder Interviewer erfolgt, können Funktionen passwortgeschützt für Testleiter definiert werden.

### Warnung

Auch wenn Sie die Funktionalität eines Testadministratormenüs für die Durchführung Ihrer Datenerfassung nicht benötigen, sollten Sie dennoch ein Testadministratormenü definieren, wenn Sie planen, mit dem *IRTlib Player* Daten offline zu erfassen. Nur so können Sie sicherstellen, dass Sie die Anwendung im Falle unvorhergesehener Ereignisse ohne den Task-Manager (und ohne möglichen Datenverlust) beenden können.

Detaillierte Informationen zu der Konfiguration des *Testleitermenüs* finden sich hier in der eingebetteten Hilfe:

### Eingebettete Programmhilfe

#### 4.5.1. Konzept eines Testleitermenüs (Menü für Testadministratoren)

Die Konfiguration des Menüs für Testadministratoren erfolgt in zwei Schritten. Zunächst muss eine Tastenkombination definiert werden, mit der das Testleitermenü aufgerufen werden kann. Wird diese Tastenkombination während der Testbearbeitung gedrückt, erscheint ein Fenster zur Passworteingabe. Testadministratoren geben das (nur) ihnen bekannte Passwort ein und erhalten so Zugriff auf ausgewählte Funktionen. Zu diesem Zweck müssen in einem zweiten Schritt eine oder mehrere Rollen im *IRTlib Editor* definiert werden.

#### 4.5.1.1. Zugriff für Testleitung

Zunächst muss eine Tastenkombination definiert werden.

- **Taste:** Die Konfiguration der Tastenkombination für das Testleitermenü erfordert zunächst die Definition einer Taste. Um eine Taste festzulegen klickt man in das Feld und drückt die Taste, welche für das Testleitermenü verwendet werden soll.
- **Modifikatoren** (Alt, Strg und Shift): Für eine Taste kann dann zusätzlich festgelegt werden, ob eine oder mehrere Modifikatoren gedrückt werden müssen damit das Testleitermenü geöffnet wird.

Beispiel:

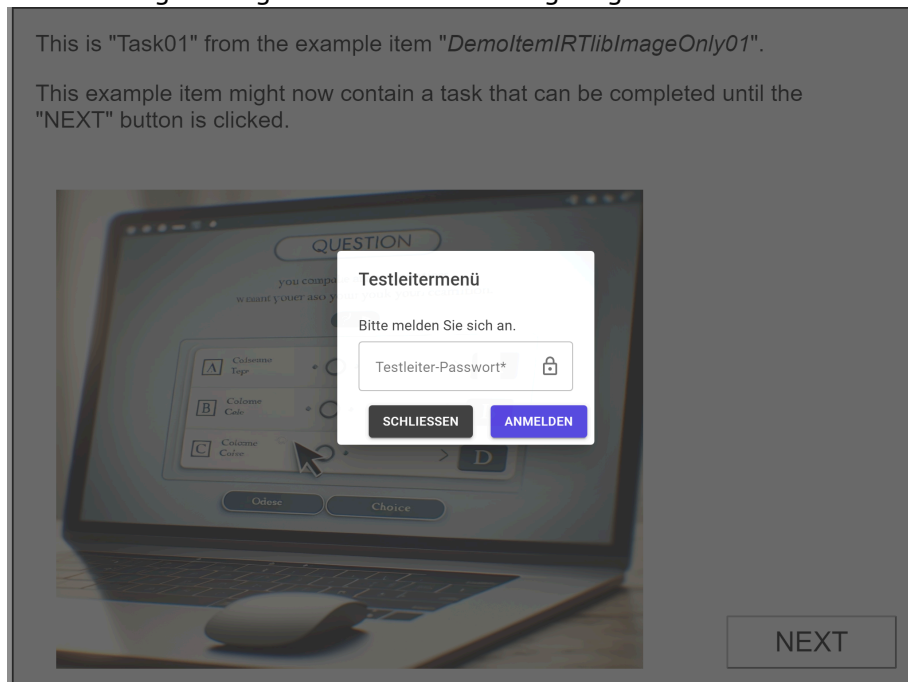
- Die folgende Konfiguration definiert die Tastenkombination Strg + Shift+ X:

Taste*	<input type="checkbox"/>	Alt
X	<input checked="" type="checkbox"/>	Strg
	<input checked="" type="checkbox"/>	Shift

Die definierte Tastenkombination öffnet nur die Option zur Eingabe eines Passwort für Testleiter während der Testbearbeitung im *IRTlib Player*. Um die Funktion zu nutzen, ist ein Passwort notwendig, welches zusammen mit einer Rolle im zweiten Schritt definiert wird.

#### 4.5.1.2. Rollen

Nach dem Aufruf der definierten Tastenkombination wird während der Testbearbeitung die Aufforderung zur Eingabe eines Passworts angezeigt:

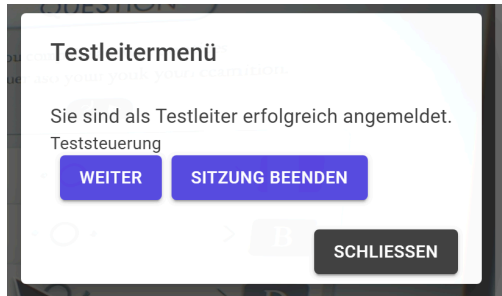


Welche Funktionen nun wirklich zugreifbar sind wird dadurch gesteuert, welches Passwort eingegeben wird. Nur wenn ein gültige Passwort bekannt ist, können Funktionen der Testleitung aufgerufen werden.

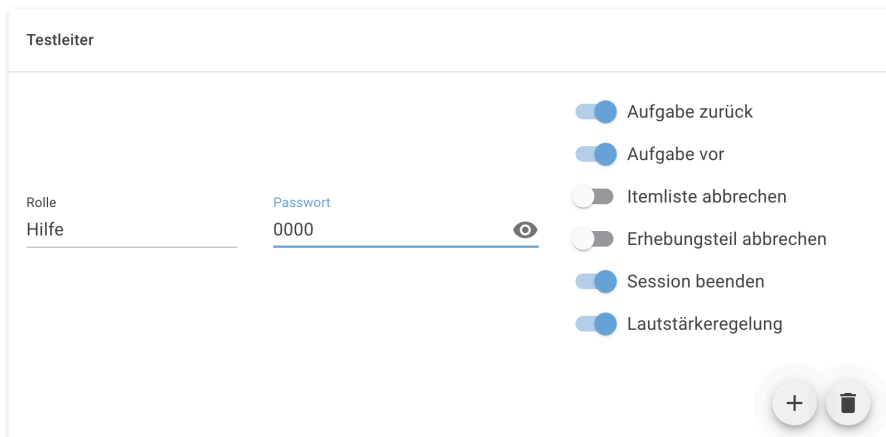
Beispiel:



- In der folgenden Konfiguration können Testleiter mit diesem Passwort zur nächsten Aufgabe springen (*Weiter*) oder die Anwendung beenden (*Sitzung beenden*):



Um eine Rolle zu definieren, muss zunächst das +-Symbol unten rechts geklickt werden. Danach kann der Name einer Rolle und ein Passwort definiert:



Der Name der Rolle dient nur der Dokumentation. Entscheidend für die Funktionalität ist die Vergabe eines eindeutigen Passwortes und die Auswahl einer oder mehrerer der folgenden Funktionen:

- **Aufgabe zurück:** Ermöglicht die Navigation zur vorherigen Aufgabe.
- **Aufgabe vor:** Ermöglicht die Navigation zur nächsten Aufgabe.
- **Itemliste abrechnen:** Ermöglicht die Abarbeitung der aktuellen Itemliste abubrechen. Diese Option ist insbesondere sinnvoll, wenn in einem *Erhebungsteil* die Option *Routing* aktiviert ist und die Definition von *CBA ItemBuilder* Tasks mit Hilfe von Itemlisten umgesetzt ist.
- **Erhebungsteil abrechnen:** Ermöglicht den Abbruch des aktuellen Erhebungsteils.
- **Sitzung beenden:** Ermöglicht das Beenden der aktuellen Sitzung.
- **Lautstärkeregelung:** Ermöglicht die Änderung der Lautstärke.

Die Audiodatei, welche zur Kontrolle der Audioausgabe abgespielt wird nachdem die Lautstärke verändert wurde, kann im Abschnitt *Audio für Soundtest* eingefügt und in der Studienkonfiguration hinterlegt werden.

Wenn geänderte Einstellungen erhalten bleiben sollen, müssen die Änderungen über das Disketten-Symbol gespeichert werden. Andernfalls kann das Verwerfen-Symbol verwendet werden:



## 4.6. Abschluss von Erhebungen

Für die Integration von Assessments in externe Abläufe besteht die Möglichkeit, zu konfigurieren, wie nach der Bearbeitung der Assessmentinhalte in einer *Session* vorgegangen werden soll, was also am *Session-Ende* geschehen wird.

### Eingebettete Programmhilfe

#### 4.6.1. Session und Session-Ende

Eine *Session* bezieht sich auf die Durchführung einer Erhebung mit *einer* Person zu *einem* bestimmten Zeitpunkt. Der in einer Session angezeigte Inhalt entspricht einer konfigurierten *Studie*, wie sie im *IRTlib Editor* erstellt werden kann. Nachdem alle in einer *Studie* definierten Erhebungsteile durchgeführt worden sind, wird das *Session-Ende* erreicht.

##### 4.6.1.1. Konfiguration des Session-Endes

Was nach einem *Session-Ende* erfolgt, d.h. wie sich der *IRTlib Players* am Ende einer Sitzung verhält, kann mit den folgenden Optionen festgelegt werden:

- **Neue Session starten:** Es wird eine neue Sitzung gestartet. Dieses Verhalten ist nicht sinnvoll, wenn die Anmeldedaten übergeben werden (entweder als *Startup-Parameter* oder als *URL-Parameter*).
- **Endtext anzeigen:** Wenn diese Option ausgewählt ist, zeigt die Plattform den konfigurierten Text an. Der Text kann als *Nachricht auf Endseite* konfiguriert werden.
- **End-Item anzeigen:** Analog zu einem *Login-Item* kann auch ein *CBA ItemBuilder-Item* definiert werden, das am Ende einer Sitzung angezeigt wird.

Das *End-Item* kann schließlich das Beenden des Offline-*IRTlib Players* anstoßen. Ein Beispiel für ein *End-Item* mit dem notwendigen JavaScript-Aufruf findet sich [hier](#).

- **Redirect to Exit URL (Redirect zu Exit-Url):** Bei Online-Lieferungen mit dem *IRTlib Player* ist es möglich, auf eine URL umzuleiten. Die *Weiterleitungs-URL* kann dann konfiguriert werden.

##### 4.6.1.2. Weitere Optionen

- **Session ID kann wiederverwendet werden:** Wenn diese Option aktiviert ist, können mehrere Datenerfassungen mit einer Session-ID administriert werden.

Wenn geänderte Einstellungen erhalten bleiben sollen müssen die Änderungen über das Disketten-Symbol gespeichert werden. Andernfalls kann das Verwerfen-Symbol verwendet werden:



## 5. Vorbereitung: Erhebungsteile / Preparation: Study Parts

Assessments, die mit der *IRTLib Software* administriert werden, bestehen aus sogenannten *Erhebungsteilen*. Nach der Konfiguration einer Studie muss zumindest ein *Erhebungsteil* angelegt werden.


### 5.1. Erhebungsteilverwaltung

Nach dem Erstellen einer *Studie* erfolgt in der Ansicht *Erhebungsteile* als nächster Schritt zur Vorbereitung einer Testauslieferung das **Hinzufügen eines neuen Erhebungsteils**:

<https://youtu.be/YFgu8uz8nkc>

Die erstellten *Erhebungsteile* erscheinen als Karten in der Ansicht *Erhebungsteile*. Wenn Studien aus mehreren Erhebungsteilen bestehen, kann für *lineare Abläufe* die Reihenfolge der Erhebungsteile in der Ansicht *Erhebungsteile / Übersicht* angepasst werden. Sollen *Erhebungsteile* in Abhängigkeit von Variablen, z.B. übergebene *Preload*-Variablen oder andere *Blockly*-Variablen gesteuert werden, kann alternativ ein Routing zwischen Erhebungsteilen konfiguriert werden.

Eine detaillierte Anleitung zur Erstellung von *Erhebungsteilen* findet sich hier in der eingebetteten Hilfe:

 Eingebettete Programmhilfe

#### 5.1.1. Erhebungsteil Anlegen

Mit dem *IRTLib Editor* werden Konfigurationen für *Studien* erstellt, welche dann in einem *IRTLib Player* zur Durchführung computerbasierter Assessments verwendet werden können. *Studien* bestehen aus einem oder mehreren *Erhebungsteilen*.

##### 5.1.1.1. Wie geht's?

Nachdem eine *Studie* angelegt ist, kann nun über Plus-Icon unten rechts ein *Erhebungsteil* hinzugefügt werden:



Danach geben Sie bitte in dem Dialog **Neuen Erhebungsteil erstellen** eine *Bezeichnung* und optional eine *Beschreibung* ein.

Achten Sie darauf, dass für die *Bezeichnung* wieder nur Buchstaben (Groß- und Kleinschreibung), Ziffern und ein `_` erlaubt sind.

Klicken Sie anschließend auf *Speichern*.

⊕ Neuen Erhebungsteil erstellen ×

Bezeichnung\* \_\_\_\_\_

Beschreibung \_\_\_\_\_

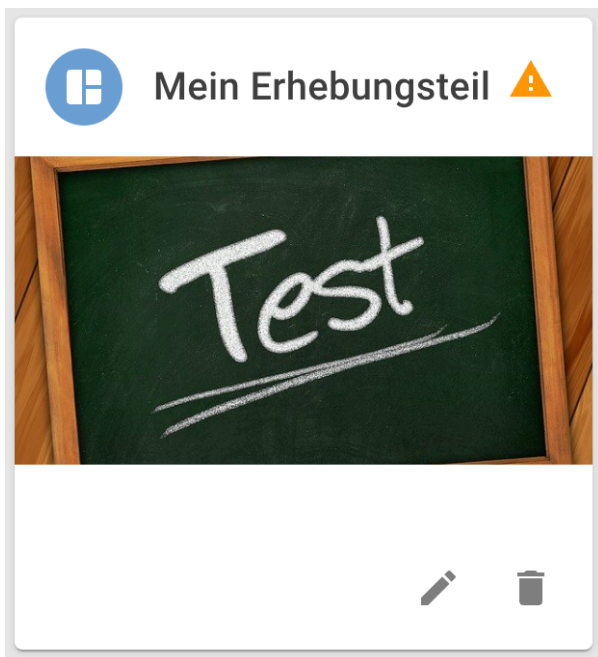
ABBRECHEN SPEICHERN

Bei Bedarf können Sie über das folgende Icon einem *Erhebungsteil* auch ein Bild zuordnen. Dieses Bild wird im *IRTLib Editor* für diesen Erhebungsteil verwendet:



#### 5.1.1.2. Erhebungsteil Bearbeiten

Erstellten *Erhebungsteile* werden in der Erhebungsteilübersicht als Kacheln angezeigt:



- Um nun mit der Konfiguration eines *Erhebungsteils* fortzufahren, klicken Sie auf das kleine Bearbeiten-Icon:



- **Erhebungsteil Löschen:** Mit dem Papierkorb-Icon können Sie *Erhebungsteile* auch wieder löschen. Das Löschen von *Erhebungsteilen* kann nicht rückgängig gemacht werden:



### 5.1.1.3. Erhebungsteile Sortieren

Wenn in der Konfiguration einer *Studie* in der Ansicht *Info* (Abschnitt *Übersicht*) die Option *Routing für Erhebungsteile aktivieren* nicht ausgewählt ist, dann werden *Erhebungsteile* in der Reihenfolge administriert, in welcher sie in der Erhebungsteilverwaltung angezeigt werden.

- **Erhebungsteile Verschieben:** Um per drag-and-drop die Reihenfolge von *Erhebungsteilen* zu verändern, muss zunächst über folgendes Toggle-Icon der Modus *Reihenfolge Ändern* aktiviert werden:



Danach können die Kacheln in die gewünschte Reihenfolge gebracht werden. Der Modus *Reihenfolge Ändern* wird beendet, wenn das Disketten-Symbol angeklickt oder die Änderungen verworfen werden:



Die Reihenfolge von *Erhebungsteilen* kann in der Ansicht *Erhebungsteile* verändert werden:

<https://youtu.be/Ag0IcETZTdM>

Vor dem Hinzufügen bzw. Auswählen von CBA ItemBuilder Projekten, wie im Abschnitt *Assessmentinhalte* (Items) beschrieben, können in der Ansicht *Info* ausgewählte Einstellungen konfiguriert werden.

Eine detaillierte Beschreibung findet sich hier in der eingebetteten Hilfe:

Eingebettete Programmhilfe

## 5.1.2. Grundkonfiguration für Erhebungsteile

### 5.1.2.1. Bezeichnung und Beschreibung

- **Bezeichnung:** Die interne Bezeichnung des Erhebungsteils, welche im *IRTlib Editor* zur Bearbeitung und zur Definition des Ablaufs angezeigt wird. Bezeichnungen dürfen keine Sonderzeichen, Leerzeichen und Umlaute enthalten und nicht mit einer Ziffer beginnen.
- **Beschreibung:** Optionale, zusätzliche Beschreibung eines Erhebungsteils.

### 5.1.2.2. Routing innerhalb von Erhebungsteilen

- **Routing aktivieren:** Die konfigurierten Assessmentinhalte im Abschnitt *Items* können als lineare Sequenz, d.h. in der konfigurierten Reihenfolge administriert werden. Soll eine davon abweichende Reihenfolge verwendet werden, kann hier die Option *Routing aktivieren* gewählt werden. Die Reihenfolge kann dann im Abschnitt *Routing* als visuelles Programm spezifiziert werden.

### 5.1.2.3. Weitere Einstellungen

- **Snapshot verwenden:** Damit CBA ItemBuilder *Tasks* mehrfach besucht werden können, wird deren Inhalt beim Verlassen des Items in sogenannten *Snapshots* gespeichert. *Snapshots* können auch dazu verwendet werden, um die Inhalte eines CBA ItemBuilder *Tasks* zu einem späteren Zeitpunkt erneut darzustellen. Diese Option sollte nur dann deaktiviert werden, wenn es einen gewichtigen Grund gibt und die Konsequenzen (d.h. die dann nicht gespeicherten *Snapshot*-Daten) sorgfältig bedacht wurden.

Das Hinzufügen und Verwalten von CBA ItemBuilder Projekte innerhalb des *IRTlib-Editors* erfolgt im Abschnitt Items.

#### Hinweis zur Zeitbegrenzung

Für die Administration von zeitbegrenzten Erhebungsteilen kann unter Bearbeitungszeit ein Zeitlimit definiert werden. Wenn die Option *Bearbeitungszeit begrenzen* aktiviert ist, können ein oder mehrere *Tasks* definiert werden, welche bei einem *Timeout* angezeigt werden. Außerdem können im Abschnitt Vorspann-Items(s) und Nachspann-Item(s) Inhalte definiert werden, welche vor bzw. nach dem zeitbegrenzten Teil administriert werden.

## 5.2. Assessmentinhalte (Items) Einfügen

Die Inhalte, welche in einem Erhebungsteil vom Typ *CBA ItemBuilder* verwendet werden sollen, werden über den *IRTlib Editor* in die Konfiguration übernommen, d.h. die mit dem *IRTlib Editor* erstellte Konfiguration enthält auch die *CBA ItemBuilder Project Files*. Für das Hinzufügen oder Aktualisierung von *CBA ItemBuilder* Projekten steht die Ansicht *Items* zur Verfügung.

Eine detaillierte Beschreibung findet sich hier in der eingebetteten Hilfe:

#### Eingebettete Programmhilfe

### 5.2.1. Items Konfigurieren

#### 5.2.1.1. Grundfunktionen

- **Importieren von CBA ItemBuilder-Projektdateien:** Der *IRTlib Editor* pflegt eine *Liste bekannter Items*, zu denen *CBA ItemBuilder*-Projektdateien die noch nicht bekannt sind hinzugefügt werden können. Um eine Projektdatei hinzuzufügen, öffnet man zunächst mit dem +-Symbol die *Liste bekannter Items* und wählt dann die Schaltfläche *Importieren* aus.



- **Aktualisieren bereits importierter CBA ItemBuilder-Projektdateien:** Wenn eine *CBA ItemBuilder*-Projektdatei bereits in der *Liste der bekannten Items* enthalten ist, können die Projektdateien aktualisiert werden. Sie werden dann nicht zusätzlich zur *Liste der bekannten Items* hinzugefügt, sondern die bereits vorhandene *CBA ItemBuilder*-Projektdatei wird in einer neueren Version hinterlegt. Um ein Item zu aktualisieren, muss es zunächst in der Liste der Items eines *Erhebungsteils* ausgewählt werden. Dadurch wird das Aktualisieren-Symbol

aktiv. In dem sich dann öffnenden Dialog *Item aktualisieren*, kann über die Schaltfläche *Importieren* eine aktualisierte Version einer *CBA ItemBuilder*-Projektdatei hinzugefügt werden.



↑ IMPORTIEREN

- **Vorschau von CBA ItemBuilder-Projektdateien:** Die in einem *Erhebungsteil* hinzugefügten Items können direkt im *IRTlib Editor* in einer eingebauten Preview-Funktion angesehen werden. Um ein Item anzusehen, muss es zunächst in der Liste der Items eines *Erhebungsteils* ausgewählt werden. Danach kann die *Vorschau* über das Augen-Symbol aufgerufen werden:



- **Exportieren von CBA ItemBuilder-Projektdateien:** *CBA ItemBuilder*-Projektdateien, die in den *IRTlib Editor* importiert wurden, können zur weiteren Bearbeitung mit dem *CBA ItemBuilder* exportiert werden. Um ein ausgewähltes Item aus der Liste der Items eines *Erhebungsteils* zu exportieren, kann das Download-Symbol aufgerufen werden:



- **Löschen von CBA ItemBuilder-Projektdateien:** Die in *Erhebungsteilen* eingefügten Items können aus einem *Erhebungsteil* wieder gelöscht werden. Durch das Löschen-Symbol wird das Item aus einem *Erhebungsteil* entfernt, es verbleibt aber in der *Liste bekannter Items*:



Hinweis: Es ist bisher nicht möglich, *CBA ItemBuilder*-Projektdateien aus der *Liste bekannter Items* wieder zu löschen. Diese Funktionalität ist nicht notwendig, da *CBA ItemBuilder*-Projektdateien vom *IRTlib-Editor* nur dann in die Konfiguration einer *Studie* übernommen werden, wenn *Tasks* aus einer *CBA ItemBuilder*-Projektdatei in einem *Erhebungsteil* verwendet werden.

#### 5.2.1.2. Sortieren von Items (Linearer Ablauf)

- **Sortieren CBA ItemBuilder-Projektdateien:** Wenn für einen *Erhebungsteil* die Option *Routing aktivieren* nicht ausgewählt ist, dann kann in der Liste der Items die Reihenfolge über die folgende Schaltfläche angepasst werden:



Die Items dann exakt so administriert, wie sie für einen *Erhebungsteil* in dieser Liste aufgeführt sind.

Hinweis: Änderungen an der Sicht *Items* müssen über das Disketten-Symbol gespeichert oder mit dem Rückgängig-Symbol verworfen werden:



## 5.3. Bearbeitungszeit

Wenn die Administration einer linearen Folge von *CBA ItemBuilder Tasks* mit einer begrenzten Bearbeitungszeit administriert werden sollen, lässt sich dies durch Definieren einer maximalen Bearbeitungszeit (in Sekunden) umsetzen. Soll bspw. ein Testinhalt maximal 28 min. administriert werden, wird als *Bearbeitungszeit* eine Zeit von 1680 Sekunden definiert. Die Nachricht, welche beim Ablaufen der Bearbeitungszeit angezeigt werden soll, lässt sich als ein (oder mehrere) *CBA ItemBuilder Tasks* definieren.

Eine detaillierte Beschreibung findet sich hier in der eingebetteten Hilfe:

 Eingebettete Programmhilfe

### 5.3.1. Zeitbegrenzung Definieren

*Erhebungsteile* ohne *Routing* können auf einfache Weise einen zeitbegrenzten Abschnitt enthalten. Dafür wird in der Sicht *Bearbeitungszeit* die Option *Bearbeitungszeit begrenzen* aktiviert und ein Zeitlimit in Sekunden (>0) eingetragen.

Für eine Zeitbegrenzung werden vier Gruppen von *CBA ItemBuilder Tasks* unterschieden, die an unterschiedlichen Stellen im *IRTlib Editor* definiert werden. In der Sicht *Items* (analog zu nicht zeitbegrenzten *Erhebungsteilen*) werden die Items, für welche die Zeitbegrenzung gelten soll, definiert:

- **Items:** Items die so lange angezeigt werden, bis das Zeitlimit erreicht wurde.



In der Ansicht *Bearbeitungszeit* kann zusätzlich definiert werden:

- **Timeout-Items:** Items die nur angezeigt werden, wenn die zeitbegrenzten Items nicht in der begrenzten Bearbeitungszeit abgeschlossen wurden.

Als einzelne Sichten der Konfiguration von *Erhebungsteile* können schließlich folgende Tasks definiert werden:

- **Vorspann-Items:** Items die vor dem zeitbegrenzten Abschnitt angezeigt werden.
- **Nachspann-Items:** Items die nach dem zeitbegrenzten Abschnitt angezeigt werden.

In allen genannten Dialogen stehen die Symbole für folgende Operationen zur Verfügung:

- Hinzufügen: 
- Aktualisieren: 



- Vorschau/Preview:



- Download/Export:



- Löschen:



- Sortieren:



Hinweis: Komplexere Designs mit ggf. mehreren Timern lassen sich mit dem *IRTlib Editor* umsetzen, wenn die Option *Routing aktivieren* in der Übersichtsansicht zu einem *Erhebungsteil* aktiviert ist. Hinweis: Änderungen an der Sicht *Bearbeitungszeit* müssen über das Disketten-Symbol gespeichert oder mit dem Rückgängig-Symbol verworfen werden:



### Vorspann-/Nachspann-Items

Ein zentrales Konzept für die Umsetzung von Zeitbegrenzungen in der *IRTlib Software* ist die Trennung der zeitbegrenzten Items, und zusätzlicher Assessmentinhalte, die *vor* oder *nach* dem zeitbegrenzten Teil administriert werden.

- Items die *nach* einem potentiell zeitbegrenzten Abschnitt eines Erhebungsteils administriert werden, werden als *Nachspann-Items* bezeichnet.

### Eingebettete Programmhilfe

#### 5.3.2. Items nach einer Zeitbegrenzung

Der *Erhebungsteile* erlauben die Definition von Items in verschiedenen Abschnitten. Items in diesem Abschnitt *Nachspann-Item(s)* werden nach den Items angezeigt, welche im Abschnitt *Items* eines *Erhebungsteils* definiert sind. Die Trennung in *Nachspann-Item(s)* und *Items* ist besonders sinnvoll, wenn unter *Bearbeitungszeit* eine Zeitbegrenzung aktiviert ist.

Um Items in dem Abschnitt *Nachspann-Item(s)* zu konfigurieren, stehen die folgenden Optionen zur Verfügung:

- Hinzufügen:



- Aktualisieren:



- Vorschau/Preview:



- Download/Export:



- Löschen:



- Sortieren:



Hinweis: Änderungen an der Sicht *Nachspann-Item(s)* müssen über das Disketten-Symbol gespeichert oder mit dem Rückgängig-Symbol verworfen werden:



- Items die *vor* einem potentiell zeitbegrenzten Abschnitt eines Erhebungsteils administriert werden, werden als *Vorspann-Items* bezeichnet.

#### 💡 Eingebettete Programmhilfe

### 5.3.3. Items vor einer Zeitbegrenzung

Der *Erhebungsteile* erlauben die Definition von Items in verschiedenen Abschnitten. Items in diesem Abschnitt *Vorspann-Item(s)* werden vor den Items angezeigt, welche im Abschnitt *Items* eines *Erhebungsteils* definiert sind. Die Trennung in *Vorspann-Item(s)* und *Items* ist besonders sinnvoll, wenn unter *Bearbeitungszeit* eine Zeitbegrenzung aktiviert ist.

Um Items in dem Abschnitt *Vorspann-Item(s)* zu konfigurieren, stehen die folgenden Optionen zur Verfügung:

- Hinzufügen:



- Aktualisieren:



- Vorschau/Preview:



- Download/Export:



- Löschen:



- Sortieren:



Hinweis: Änderungen an der Sicht *Vorspann-Item(s)* müssen über das Disketten-Symbol gespeichert oder mit dem Rückgängig-Symbol verworfen werden:



## 5.4. Variablen

! Under Development

Diese Funktion ist gerade in Entwicklung.

💡 Eingebettete Programmhilfe

(Diese Funktionalität ist gerade noch in *Entwicklung*.)

## 5.5. Codebook

! Under Development

Diese Funktion ist gerade in Entwicklung.

💡 Eingebettete Programmhilfe

(Diese Funktionalität ist gerade noch in *Entwicklung*.)

## 5.6. ItemPool

! Under Development

Diese Funktion ist gerade in Entwicklung.

💡 Eingebettete Programmhilfe

(Diese Funktionalität ist gerade noch in *Entwicklung*.)

## 5.7. Routing innerhalb von Erhebungsteilen

Wenn *CBA ItemBuilder*-Tasks nicht in einer linearen Abfolge administriert werden sollen, die im Vorhinein feststeht und für alle Testpersonen identisch ist, dann kann die Funktion *Routing* der *IRTlib Software* verwendet werden.

Eine detaillierte Beschreibung zum *Routing innerhalb von Erhebungsteilen* findet sich hier in der eingebetteten Hilfe:

💡 Eingebettete Programmhilfe

### 5.7.1. Zusammenfassung zu Routing innerhalb von Erhebungsteilen

Die Reihenfolge von *CBA ItemBuilder*-Aufgaben kann hier mit Hilfe von *Blockly* (also einer Form des visuellen Programmierens) definiert werden. *Blockly*-basierte Ablaufsteuerung ist verfügbar, wenn bei einem Erhebungsteil die Option *Routing aktivieren* ausgewählt ist. Die Option ist im Abschnitt *Info* eines Erhebungsteils zu finden. Ist sie aktiviert, enthält der Erhebungsteil den Eintrag *Routing*.

### 5.7.1.1. Beispiele

Die Grundidee zur Verwendung von *Blockly* für die Definition von Abläufen in *computerbasierten Assessments* soll zunächst mit einigen Beispielen illustriert werden.

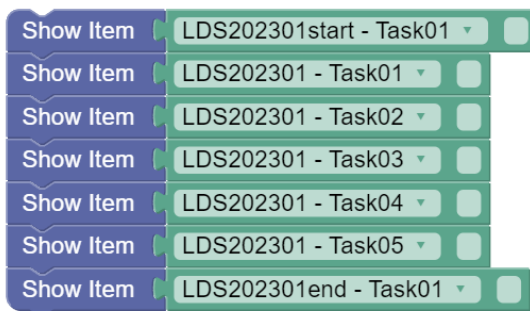
- **Beispiel für linearen Ablauf**

Basierend auf den einem Erhebungsteil hinzugefügten *CBA ItemBuilder Tasks* in der Ansicht *Items* entspricht eine lineare Folge der *Tasks* der folgenden *Blockly*-Definition:



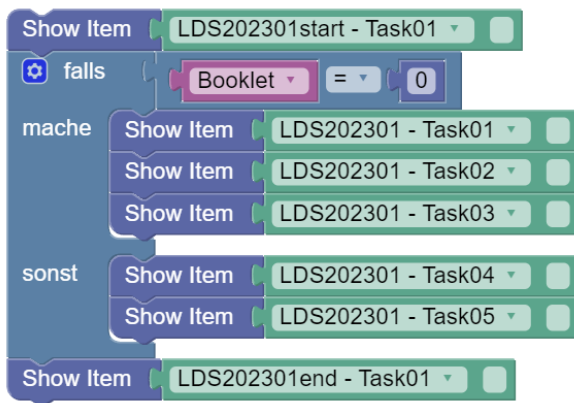
Dem *Blockly*-Element *Show Items* wird eine Liste der *CBA ItemBuilder Tasks* übergeben, die mit dem Operator *erzeuge Liste mit* erstellt wird. Die Liste wird in der dargestellten Reihenfolge abgearbeitet, wobei jeder *CBA ItemBuilder Tasks* solange dargestellt wird, bis das *NEXT\_TASK-Command* ausgeführt wird.

Eine äquivalente Formulierung einer linearen Sequenz kann auch mit mehreren *Show Items*-Blöcken erfolgen, wenn keine Zurücknavigation notwendig ist:



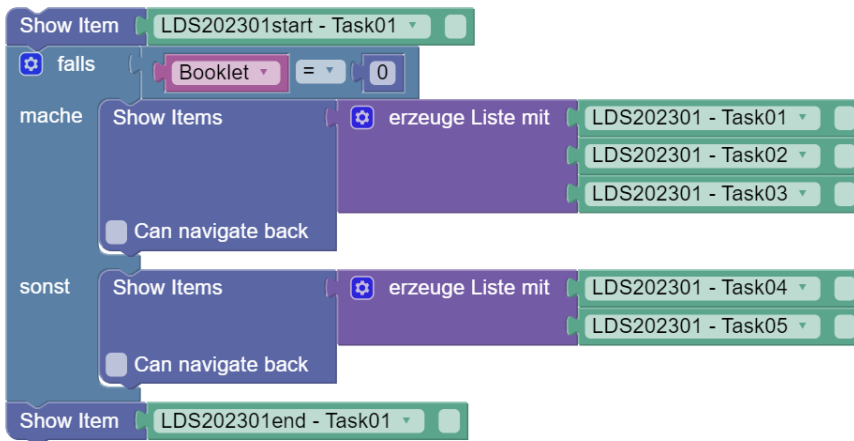
- **Beispiel für einfache Testhefte**

Mit Hilfe einer Variable (hier: *Booklet*) und einer einfachen *falls/mache*-Bedingung lässt sich daraus nun ein Ablauf definieren, welcher je nach Wert der Variable unterschiedliche Items administriert:



Die Items für Start und Ende werden immer administriert, die Tasks 1-3 nur, wenn die Variable *Booklet* den Wert 0 hat, die Tasks 4 und 5, wenn die Variable *Booklet* einen von 0 verschiedenen Wert hat.

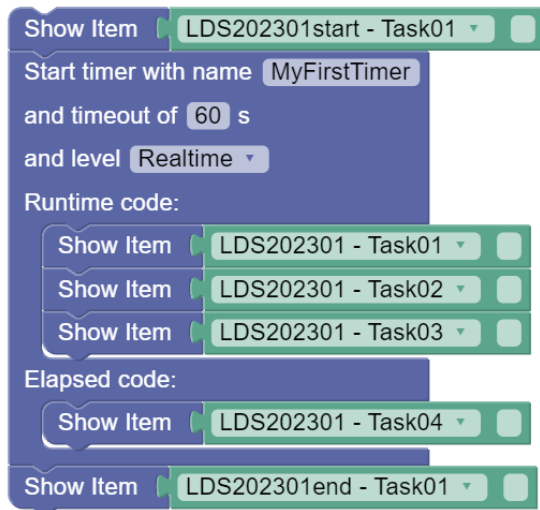
Der identische Ablauf lässt sich alternativ auch unter Verwendung des *Blockly*-Operators für das Anzeigen von Itemlisten erstellen:



Beide Varianten sind bzgl. der Funktionalität völlig äquivalent, die zweite Vorgehensweise mit Listen erlaubt aber die Verwendung der Option Zurücknavigation innerhalb der booklet-spezifischen Tasks.

- **Beispiel für Ablauf mit Zeitbegrenzung**

Um mit Hilfe der *Blockly*-Configuration zeitbegrenzte Abschnitte innerhalb eines Erhebungsteils umzusetzen, kann die folgende *Blockly*-Komponente verwendet werden:



Jeder Ablauf beginnt mit einem nicht zeitbegrenzten Start-Tasks und endet mit einem ebenfalls nicht zeitbegrenzten End-Tasks. Dazwischen läuft eine Zeitbegrenzung für einen Abschnitt mit der Bezeichnung *MyFirstTimer*, der eine Zeitbegrenzung für 60 Sekunden hat.

Die Tasks 1, 2 und 3 werden in dem Abschnitt *Runtime code* mit einer Zeitbegrenzung angezeigt. Tritt ein Timeout auf, d.h. werden die drei Tasks nicht innerhalb der 60 Sekunden bearbeitet, wird (ebenfalls ohne Zeitbegrenzung) der Task 4 angezeigt.

- **Beispiel für einfaches Booklet-Design mit Zeitbegrenzung**

Bei vielen Items kann die Definition von *Booklet Designs*, d.h. Taskreihenfolgen mit balancierten Positionen, durch Funktionen bzw. Listen vereinfacht werden.

Wenn keine Zurücknavigation notwendig ist, können Funktionen für die Definition von Clustern verwendet werden:

setze VBooklet auf ganzzahlige Zufallszahl zwischen 1 und 6

ShowInstructionItems

Start timer with name default and timeout of 1800 s and level Realtime

Runtime code:

- falls VBooklet = 1
  - mache ShowItemsC1, ShowItemsC2
- sonst falls VBooklet = 2
  - mache ShowItemsC2, ShowItemsC3
- sonst falls VBooklet = 3
  - mache ShowItemsC1, ShowItemsC3
- sonst falls VBooklet = 4
  - mache ShowItemsC2, ShowItemsC1
- sonst falls VBooklet = 5
  - mache ShowItemsC3, ShowItemsC2
- sonst
  - ShowItemsC3, ShowItemsC1

Elapsed code:

- ShowTimeoutItems
- ShowFarewellItems

um ShowInstructionItems # Show Item Statements for Instruction Items

um ShowItemsC1 # Show Item Statements for C1 Items

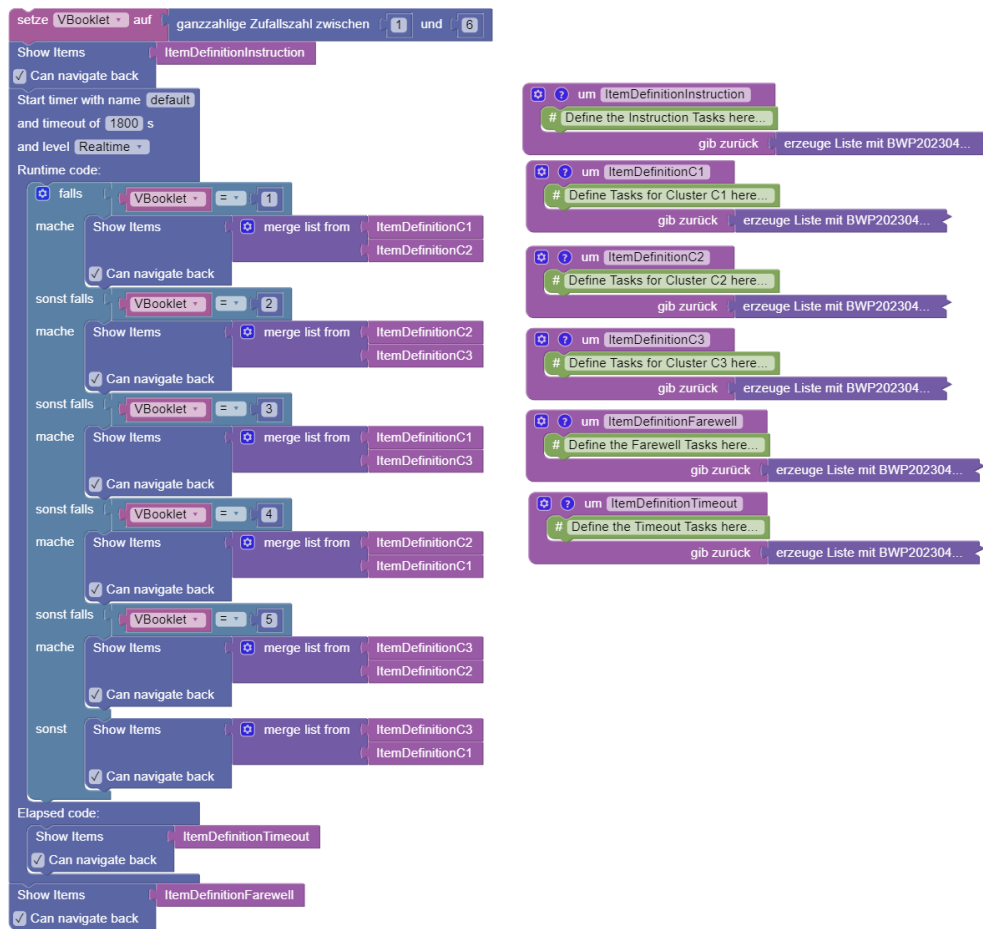
um ShowItemsC2 # Show Item Statements for C1 Items

um ShowItemsC3 # Show Item Statements for C1 Items

um ShowTimeoutItems # Show Item Statements for Timeout Items

um ShowFarewellItems # Show Item Statements for Farewell Items

Mit Zurücknavigation können die Funktionen Listen von Tasks zurückgeben:

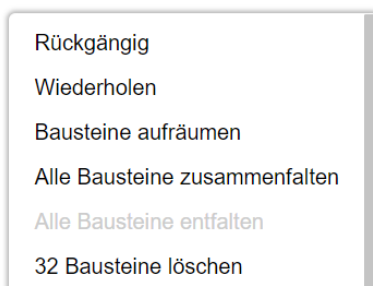


Weitere Informationen siehe [hier](#).

### 5.7.1.2. Hinweise zur Verwendung des Blockly-Editors

Die Definition von Abläufen erfolgt in dem visuellen *Blockly*-Editor. Die Ausführung beginnt mit dem Element, welches am weitesten oben ausgerichtet ist. Wenn nötig, kann der Arbeitsbereich mit der Funktion Aufräumen automatisch ausgerichtet werden. Zum Hinzufügen von *Blockly*-Operatoren können diese per Drag-and-Drop aus der Palette gezogen werden.

- **Löschen:** Zum Löschen von Operatoren können diese auf den Papierkorb gezogen werden. Ausgewählte *Blockly*-Elemente können auch über die Taste *Entf(ernen)* gelöscht werden. Alternativ können ausgewählte *Blockly*-Elemente auch über Kontextmenü gelöscht werden.
- **Redo-/Undo:** Innerhalb des *Blockly*-Editor können einzelne Aktionen rückgängig gemacht werden. Dafür kann die Tastenkombination *Strg + Z* verwendet werden. Mit *Strg + Y* wird eine Aktion wiederholt. Durch einen Klick in einen leeren Bereich des *Blockly*-Editors ist der Zugriff auf ein Kontextmenü möglich, welches ebenfalls die Optionen für *Rückgängig* und *Wiederholen* bereithält:



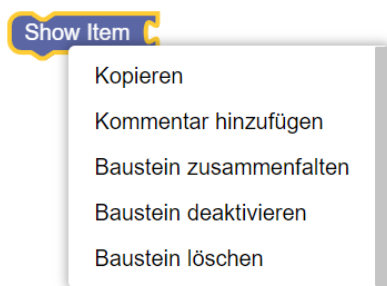


- **Speichern:** Anpassungen im *Blockly*-Editor müssen gespeichert werden. Dafür steht unten rechts das Disketten-Symbol zur Verfügung:



Sollen die Änderung (insgesamt) verworfen werden, kann unten rechts das Verwerfen-Symbol verwendet werden.

- **Zoom:** Die Ansicht im Arbeitsbereich kann mit den Icons + vergrößert und mit – verkleinert werden.
- **Kontextmenü:** Weitere Optionen sind über die rechte Maustaste (Kontextmenü) im *Blockly*-Editor verfügbar. Um diese Funktionen aufzurufen, muss auf ein *Blockly*-Element ein Sekundärklick (rechte Maustaste) durchgeführt werden:
  - Kopieren dupliziert das ausgewählte *Blockly*-Element, inklusive aller verbundener Elemente.
  - Kommentieren von Blöcken ist möglich.
  - Blöcke können deaktiviert/aktiviert werden.
  - Manche Block-Typen erlauben die Darstellungsform extern/intern zu wechseln.
  - Blöcke, welche weitere Blöcke enthalten, können zusammengefallen/entfaltet werden.
  - Das Löschen von Blöcken ist auch über das Kontextemü möglich.



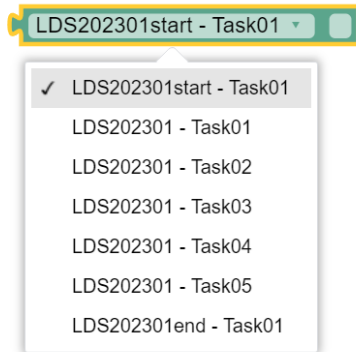
Einige *Blockly*-Elemente stellen im Kontextmenü auch einen Eintrag *Hilfe* zur Verfügung, welcher auf allgemein zugängliche *Blockly*-Dokumente (<https://github.com/google/blockly/wiki/>) verweist.

### 5.7.2. Verwendung von Blockly zur Ablaufsteuerung

Die Grundfunktionen für die Nutzung der *Blockly*-Umgebung zur Steuerung von Assessments finden sich im Abschnitt *Session*.

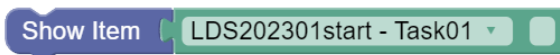
### 5.7.2.1. Einzelne Items anzeigen

Auf *CBA ItemBuilder*-Tasks, die in der Ansicht *Items* für einen Erhebungsteil importiert wurden, kann in der Ablaufsteuerung, wie in den Beispielen oben gezeigt, mit Hilfe des folgende *Blockly*-Elements für *Tasks* zugegriffen werden:



Das Element, welches im Abschnitt *Session* der Palette des *Blockly*-Editors zu finden ist, kann durch die Auswahlliste konfiguriert werden. Jedes *Blockly*-Element für *Tasks* kann auf genau einen konkreten Task verweisen, d.h. in der Regel besteht eine Ablaufdefinition aus mehrerer solcher Elemente.

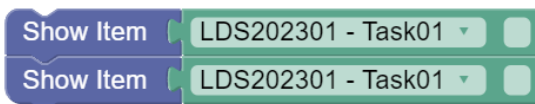
*Blockly*-Elemente für *Tasks* können nicht direkt in den Ablauf eingefügt werden, sondern werden zusammen mit einem *Show Item*-Element verwendet:



Das Beispiel für einfache Testhefte illustriert, dass Abläufe in der *Blockly*-Definition häufig durch eine Abfolge von mehreren *Show Item*-Operatoren definiert werden. *Show Item*-Operatoren können dabei in Bedingungen und Schleifen, sowohl innerhalb des Hauptablaufs als auch innerhalb von Funktionen eingefügt werden.

### 5.7.2.2. Verwendung von Geltungsbereichen (Scopes)

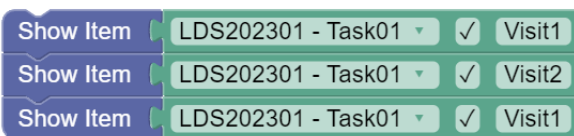
Mit Hilfe der *Blockly*-basierten Ablaufsteuerung ist es auch möglich, *CBA ItemBuilder*-Tasks mehrfach innerhalb eines Ablaufs zu administrieren:



Dabei wird beim erneuten Aufruf eines Items der Zustand aus dem letzten Besuch wiederhergestellt, d.h. die Bearbeitung wird fortgesetzt. Sollen Items mehrfach neu, d.h. unbearbeitet vorgelegt werden, kann das automatische wiederherstellen nicht gewünscht sein. Dafür kann optional die Checkbox für die Angabe eines *Scopes* (Geltungsbereich) aktiviert werden:



Wird nichts weiter angegeben, wird das Item im "Default"-Scope administriert. Alternativ kann ein Text definiert werden, wie in folgendem Beispiel zu sehen:



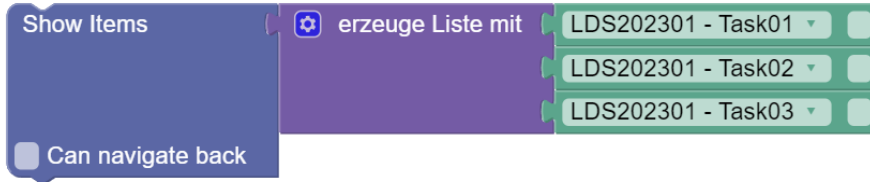
Beim ersten Besuch wird der Task in dem *Scope* "Visit1" dargestellt. Danach folgt eine neue, unabhängige Darstellung des Tasks in einem anderen *Scope* ("Visit2"). Im dritten Aufruf wird der Task wieder mit den Daten dargestellt, die beim ersten Besuch bereits gesammelt wurden (d.h.

der *Scope* "Visit1" wird erneut verwendet).

### 5.7.2.3. Anzeigen mehrerer Items (Itemlisten)

Wie im Beispiel für linearen Ablauf zu sehen, können lineare Tests auch über Listen von Tasks dargestellt werden.

Listen können mit dem *Blockly*-Operator *Show Items* verwendet werden:

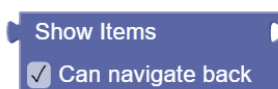


- **Zurücknavigation:** Das *Show Items*-Element für Listen kann über die Eigenschaft *Can navigate back* konfiguriert werden. Ist diese Eigenschaft ausgewählt, dann können *CBA ItemBuilder-Tasks* mit dem *Command* `BACK_TASK` eine Navigation zum vorherigen *CBA ItemBuilder Tasks* anfordern.
- **Abbrechen von Listen:** Die Verwendung von Listen erlaubt auch das Abbrechen von Listen. Listen können über zwei Wege abgebrochen werden:
  - Das *Command* `CANCEL_TASK`, welches innerhalb von *CBA ItemBuilder Tasks* verwendet werden kann, wird aufgerufen.
  - Im Testleitermenü, welches für die Studie konfiguriert und ggf. über den *Blockly*-Operator Testleitermenü bearbeiten angepasst wurde, wird die Funktion *Itemliste abbrechen* aufgerufen.

Die Administration einer Itemliste wird dadurch abgebrochen, und die Abarbeitung des *Blockly*-Ablaufs nach dem *Show Items*-Block fortgesetzt.

### 5.7.2.4. Anzeige von Items mit Speicherung der Ergebnisse

Die Operatoren *Show Item* (für einzelne Items) und *Show Items* (für Itemlisten) sind auch als Operatoren für Wertzuweisungen verfügbar:

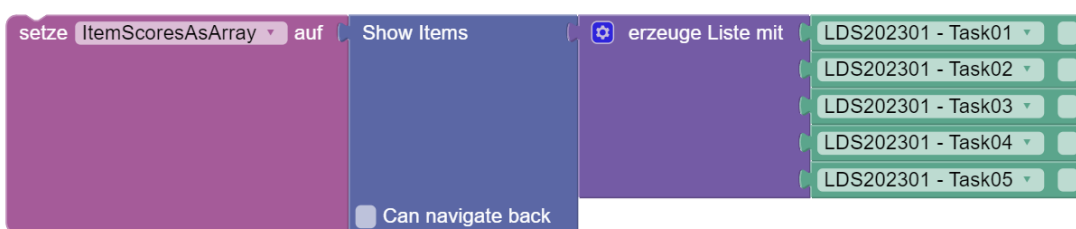


Mit deren Hilfe lassen sich Ergebnisse der Itembearbeitung zu Variablen (String oder Array) zuweisen, und dann für die Ablaufsteuerung auswerten.

- Einzelner Task:



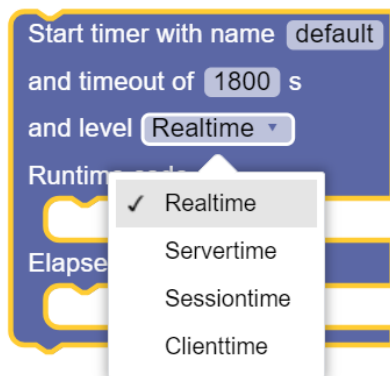
- Liste von Tasks:



### 5.7.2.5. Definition von Zeitbegrenzungen

Wie im Beispiel Ablauf mit Zeitbegrenzung bereits illustriert, kann mit dem *Blockly*-Block *Start time with name* die Zeitbegrenzte Administration von Items umgesetzt werden.

Das *Blockly*-Element *Start timer with name* erlaubt die Definition von Zeitbegrenzungen. Jede Zeitbegrenzung kann einen eigenen Namen haben. Zusätzlich muss die Zeit in Sekunden angegeben werden. Darüber kann definiert werden, welche Art von Zeit verwendet werden soll:



- **Realtime:** Der Timer läuft in Echtzeit. Wird nicht durch Downtimes des Servers oder einen Sessionneustart beeinflusst.
- **Servertime:** Der Timer läuft in Serverzeit. Wird nicht durch einen Sessionneustart beeinflusst, berücksichtigt aber keine Downtime des Servers.
- **Sessiontime:** Der Timer läuft innerhalb einer Session. Wird bei einer Unterbrechung durch Server-Downtime oder einen Session-Neustart unterbrochen.
- **Clienttime:** Der Timer läuft nur in Clientzeit und wird auch bei Pausen der Session unterbrochen.

Schließlich können zwei Stellen mit weiteren *Blockly*-Operatoren (wie bspw. ein oder mehrere *Show Item*-Blöcke zum Anzeigen einzelner Items oder ein oder mehrere *Show Items*-Blöcke zum Anzeigen von Listen) gefüllt werden:

- **Runtime code:** Diese Blöcke werden ausgefüllt, bis die definierte Zeit abgelaufen ist.
- **Elapsed code:** Diese Blöcke werden nur ausgefüllt, wenn der *Runtime code* nicht innerhalb der Zeit beendet wurde.

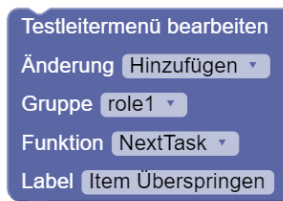
### 5.7.2.6. Blockly-Operatoren für das Testleitemenü

In der Studiendefinition können Funktionen des Testleitemenüs für eine oder mehrere Rollen angelegt werden. Rollen stellen unterschiedliche Funktionen zusammen, die mit Hilfe des vom Testleiter einzugebenden Passworts unterschieden werden können.

**Anpassen von Standardfunktionen:** Folgende Standardfunktionen können für eine Studie im Abschnitt *Info / Testleitemenü* definiert werden:

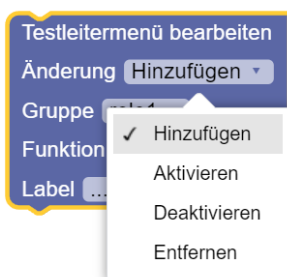
- *Navigation:* Aufgabe vor / Aufgabe zurück
- *Listen:* Itemliste abrechnen
- *Beenden:* Erhebungsteil beenden und Session beenden
- *Lautstärkeregelung:* Einstellen der Audiolautstärke während des Assessments

Während der Bearbeitung eines Erhebungsteils kann in der Ablaufsteuerung mit Hilfe des folgenden *Blockly*-Operators das Testleitemenü kontextspezifisch angepasst werden:



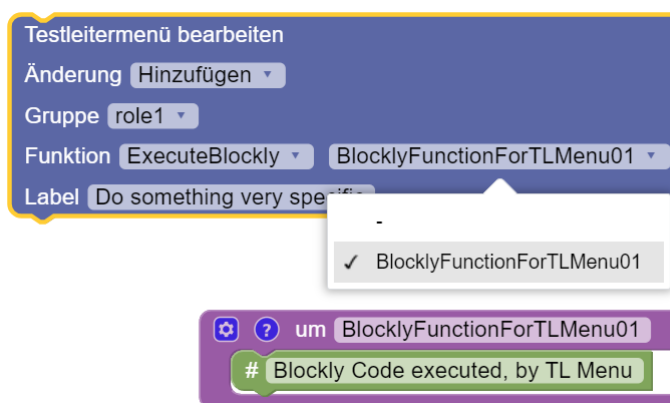
Das Testleitemenü kann für jede der Standardfunktionen (im Bereich *Funktion*) für eine Rolle (im Bereich *Gruppe*) sowohl die Beschriftung der Schaltfläche (im Bereich *Label*) geändert werden:

- *Hinzufügen*: Funktion wird im Testleitemenü ergänzt
- *Entfernen*: Funktion wird aus dem Testleitemenü entfernt
- *Deaktivieren*: Funktion wird im Testleitemenü deaktiviert
- *Aktivieren*: Funktion wird im Testleitemenü aktiviert



Der Aufruf dieses *Blockly*-Operators im Testablauf definiert das Verhalten des Testleitemenüs im weiteren Testablauf. Im Unterschied zu *Entfernen* bleiben *deaktivierte* Funktionen im Testleitemenü sichtbar, können aber (bis sie wieder *aktiviert* werden) nicht ausgeführt werden.

**Verwenden von Blockly-Funktionen im Testleitemenü:** Der *Blockly*-Operator für das Bearbeiten des Testleitemenüs enthält im Abschnitt *Funktion* auch die Option zum Ausführen von *Blockly*-Code (*ExecuteBlockly*):

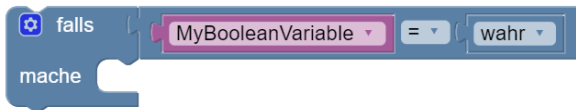


Wenn *ExecuteBlockly* ausgewählt ist, kann in dem *Blockly*-Element *Testleitemenü bearbeiten* eine innerhalb des *Blockly*-Editors definierte Funktion ausgewählt werden. Die in dieser Funktion definierten *Blockly*-Operatoren werden dann ausgeführt, wenn ein Testleiter die entsprechende Schaltfläche zur Laufzeit im Testleitemenü auswählt.

### 5.7.3. Fortgeschrittene Blockly-Verwendung

#### 5.7.3.1. Ablaufsteuerung mit Bedingungen

Der Abschnitt *Logic* enthält den *Blockly*-Operator *falls/mache*, welcher zur Umsetzung von Bedingungen im Ablauf verwendet werden kann. Bedingungen sind logische Ausdrücke, bspw. die Prüfung ob eine Preload-Variable einen bestimmten Wert hat:



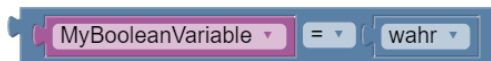
Nur wenn die Bedingung (*falls*) erfüllt ist, werden die *Blockly*-Operatoren ausgeführt, welche innerhalb des Bedingungsblocks definiert sind (d.h. neben *mache*). In dem Beispiel wird geprüft, ob eine boolsche Variable den Wert *wahr* hat.

Die Bedingung wird dabei als separater Block definiert, die mit dem *Blockly*-Operator *falls/mache* verbunden ist. Hier die beiden Komponenten separat:

- Bedingung:



- Logischer Ausdruck:

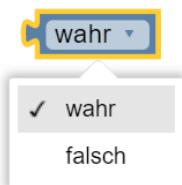


#### 5.7.3.2. Verwendung logischer Ausdrücke

Logische Ausdrücke in Bedingungen basieren entweder auf Wertevergleichen oder Rückgaben von Funktionen. Wertevergleiche können mit folgendem *Blockly*-Element realisiert werden:



Die beiden Slots können mit Werten gefüllt werden. Für boolsche Werte (*wahr/falsch*) steht ein entsprechendes *Blockly*-Element im Abschnitt *Logic* bereit:



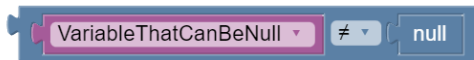
Bedingungen sind auch mit Variablen von anderem Datentyp möglich:



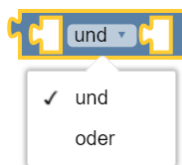
Für numerische Werte gibt es ein entsprechendes *Blockly*-Element im Abschnitt *Math*, welcher Operatoren für Zahlen und einfache mathematische Operationen enthält:

42

Mit dessen Hilfe und einer numerischen Variable lässt sich folgende Bedingung formulieren: Aus technischen Gründen kann es auch notwendig sein zu prüfen, ob eine Variable noch gar keinen Wert hat. Das kann durch Verwendung der *Blockly*-Komponente `null` umgesetzt werden:



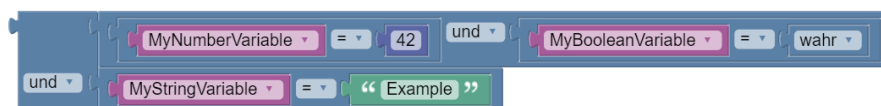
**Kombination von logischen Ausdrücken:** Einzelne Bedingungen oder logische Ausdrücke können mit folgendem *Blockly*-Element aus dem Abschnitt *Logik* verbunden werden:



Dabei steht eine *und* sowie eine *oder*-Verknüpfung der Aussagen zur Auswahl. Die *und*-Verknüpfung ist wahr, wenn beide Ausdrücke wahr sind, die *oder*-Verknüpfung ist wahr, wenn mindestens einer von beiden Ausdrücken (oder beide Ausdrücke) wahr sind.

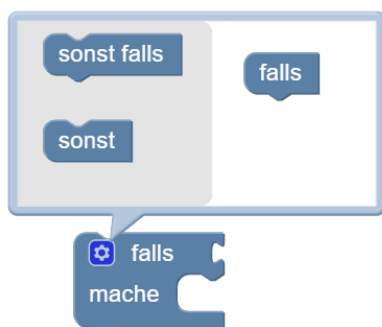


Mehrere *logische Ausdrücke* können in einander geschachtelt werden:



Hinweis: Für eine übersichtlichere Darstellung ist bei der äußeren *und*-Verknüpfung die externe Darstellung gewählt.

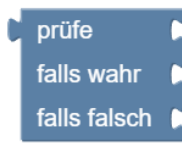
**Mehrere Bedingungen (sonst falls / sonst):** Durch Klick auf das kleine Zahnrad-Symbol eines Bedingungsblocks (*falls/mache*) kann dieser konfiguriert werden:



Durch das Hinzufügen eines Abschnitts *sonst falls* können kann eine weitere Bedingung hinzugefügt werden. Die in einem *sonst falls* Abschnitt definierte Bedingung wird geprüft, wenn die vorherigen Bedingungen (*falls*) nicht erfüllt sind. Ist eine Bedingung erfüllt, werden die definierten *Blockly*-Operatoren ausgeführt.

Durch das Hinzufügen eines Abschnitts *sonst* können Blöcke hinzugefügt werden, welche dann ausgeführt werden, wenn keine der Bedingungen erfüllt ist.

**Prüfe-Operator:** Für Wertzuweisungen in Abhängigkeit von einer Bedingung stellt der *Blockly*-Editor eine speziellen Operator *prüfe-falls wahr-falls falsch* zur Verfügung:



Der Operator kombiniert eine Wertzuweisung mit einem logische Ausdruck:



In diesem Beispiel wird der String-Variablen *MyStringVariable* der Wert *Yes* zugewiesen, falls die boolsche Variable *MyBooleanVariable* den Wert *wahr* hat. Wenn *MyBooleanVariable* den Wert *falsch* hat, wird *MyStringVariable* der Wert *No* zugewiesen.

**Negation:** Um einen logischen Ausdruck umzukehren (Negation) steht folgender *Blockly*-Operator zur Verfügung:



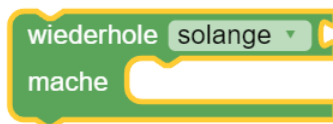
### 5.7.3.3. Ablaufsteuerung mit Schleifen

Die mehrfache Ausführung von *Blockly*-Operatoren (und der damit darstellbaren Aktionen) ist mit Schleifen möglich. Der Abschnitt *Loops* der *Palette* enthält die dafür notwendigen *Blockly*-Elemente.

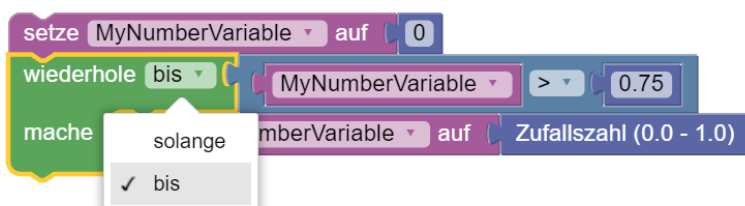
**Wiederhole n-mal:** Der folgende *Blockly*-Operator kann verwendet werden, um die Ausführung von Blöcke n-mal zu wiederholen:



**Wiederhole solange:** Schleifen können auch solange wiederholt werden *bis* eine Bedingung zutrifft (oder *solange* eine Bedingung zutrifft):

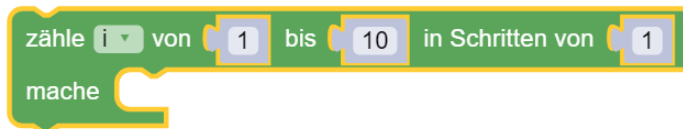


Beispiel:





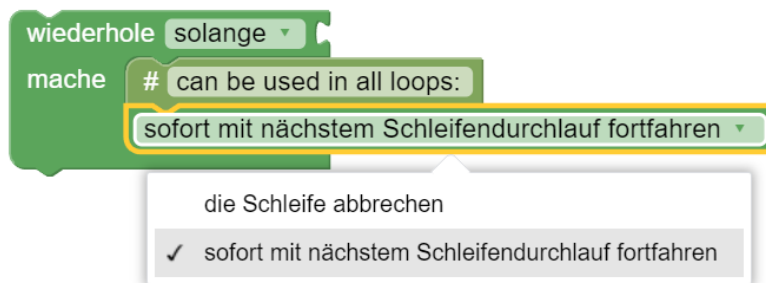
**Zähle von/bis:** Schleife mit Hilfsvariablen:



**Für jeden Wert aus Liste:** Schleife über alle Werte einer Liste:



**Schleifen vorzeitig abbrechen:** Folgendes *Blockly*-Element kann genutzt werden, um eine Schleife (vorzeitig) abzubrechen oder um vorzeitig mit dem nächsten Schleifendurchlauf zu beginnen:



#### 5.7.3.4. Operatoren für Zahlen und einfache mathematische Funktionen

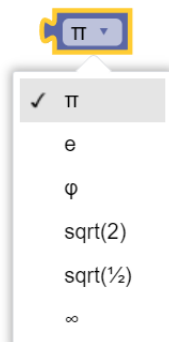
Der Abschnitt *Math* der *Palette* enthält *Blockly*-Elemente zur Verwendung von Zahlen und einfachen mathematischen Funktionen.

##### Ausdrücke

- Zahlen: Ganzzahlen / Dezimalzahlen

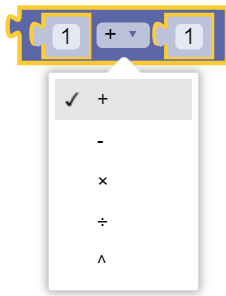


- Symbole: Spezielle Symbole oder Konstanten:



## Basale Funktionen

- Addition, Subtraktion, Multiplikation, Division und Potenzfunktion von zwei Argumenten:



Schachtelung ist möglich, z.B.:



- Division mit Rest:



- Ob eine Zahl gerade ist, kann mit diesem *Blockly*-Element geprüft werden:

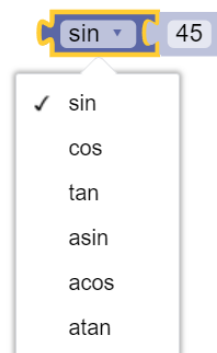


- Mit dem folgenden *Blockly*-Element, kann eine Zahl auf einen Bereich begrenzt werden:

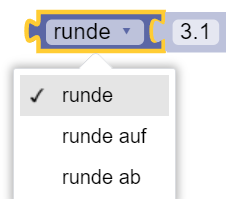


## Eingebaute Funktionen

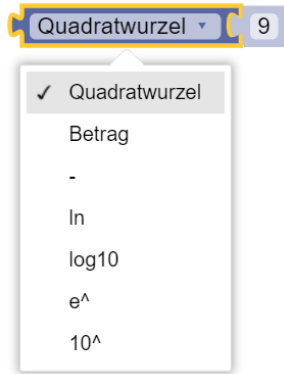
- Trigonometrische Funktionen:



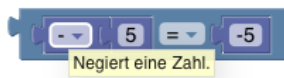
- Runden von Werten:



- Weitere Funktionen:



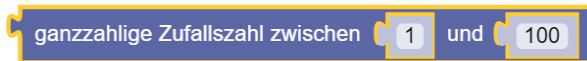
Hinweis: – erlaubt das Negieren numerischer Werte, wie in folgendem Beispiel zu sehen ist:



In dem Beispiel zu sehen ist der *Tooltip* für die in *Blockly* verfügbare Hilfe und ein Beispiel, wo die Zahl 5 mit dem --Operator in die Zahl -5 umgewandelt wird. Die äußere Bedingung (Negieren von 5 ergibt -5) ist also wahr.

**Erzeugung von Zufallszahlen:** Für die Erstellung von Zufallszahlen stehen zwei *Blockly*-Elemente zur Verfügung:

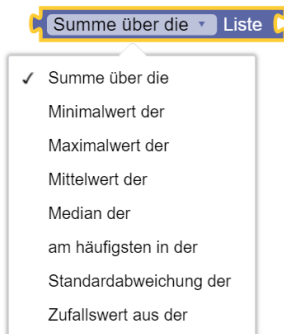
- Ganzzahlen (in Wertebereich):



- Zufallszahl zwischen 0 und 1:



**Numerische Funktionen für Listen:** Vordefinierte Funktionen für Listen umfassen:



Hinweise:

- Weitere Funktionen lassen sich, wenn benötigt, mit Schleifen für Listen umsetzen.
- Bei der Verwendung der Funktionen ist darauf zu achten, dass die Listenfunktion nur für Listen mit numerischen Datentypen anwendbar sind!

### 5.7.3.5. Operatoren für Text und einfache String-Operationen

Der Abschnitt *Text* der *Palette* enthält *Blockly*-Elemente zur Verwendung Zeichenketten.

**Ausdrücke:** Zum Erstellen von Text steht folgender Operator zur Verfügung:

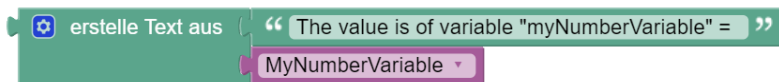


**Verketteten:** Verschiedene Operatoren können verwendet werden, um Text zusammenzufügen und zu Variablen zuzuweisen:

- Einen Text an eine Variable anfügen:



- Texte (und Variablenwerte) verketteten und an andere *Blockly*-Operatoren weitergeben:



- Zusammengeführten Texte eine Variable zuweisen:



**Textlänge:** Die Länge einer Zeichenkette kann mit folgendem *Blockly*-Operator ermittelt werden:



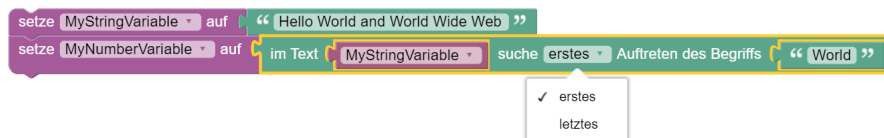
**Prüfung auf leeren String:** Leere String-Variablen können daran erkannt werden, dass die Anzahl der Zeichen 0 ist.



Alternativ kann dafür auch der folgende *Blockly*-Operator verwendet werden:



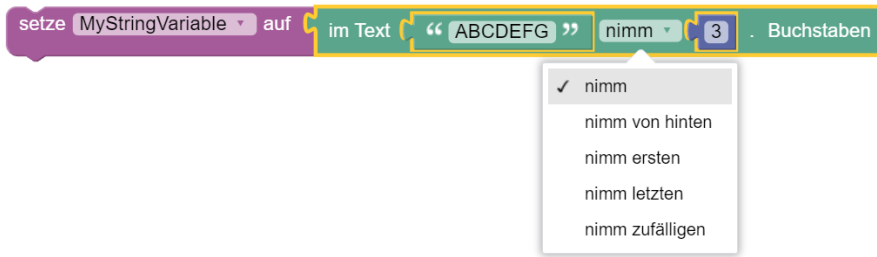
**Position in String finden:** Ein Operator, der *im Text* (der per Variable oder als Ausdruck übergeben wird) das *erste* oder *letzte Auftreten eines Begriffs* sucht, kann wie folgt verwendet werden:



Zurückgegeben wird dabei die Position des *Begriffs* innerhalb der Zeichenkette (d.h. *im Text*).

**Teilzeichenketten bilden:** Der folgende Operator nimmt aus der übergebenen Zeichenkette *im Text* die ersten Buchstaben. Die Anzahl der Buchstaben wird dabei ebenfalls übergeben.

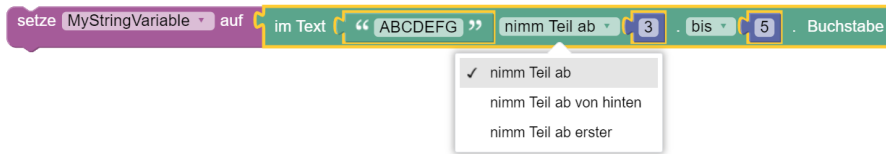
- Beispiel (hier wird, wenn die Option *nimm ersten* ausgewählt ist der Variable *MyStringVariable* der Text ABC, d.h. die ersten drei Buchstaben der Zeichenkette ABCDEFG) zugewiesen:



Buchstabenfunktion	Parameter N	Bedeutung
nimm	Ja	Es werden die ersten N Buchstaben zurückgeliefert
nimm von hinten	Ja	Es werden die letzten N Buchstaben zurückgeliefert
nimm ersten	Nein	Es wird der erste Buchstabe zurückgeliefert (entspricht nimm mit N=1)
nimm letzten	Nein	Es wird der letzte Buchstabe zurückgeliefert (entspricht nimm von hinten mit N=1)
nimm zufälligen	Nein	Es wird ein zufälliger Buchstabe zurückgeliefert

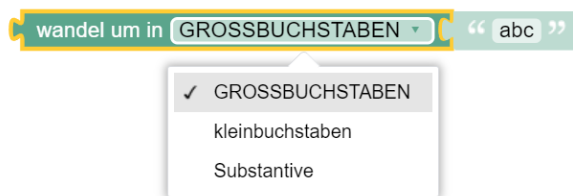
Buchstaben aus einer Zeichenkette kann man auch mit folgendem Operator entnehmen, und bspw. einer Variablen zuweisen:

- Beispiel (hier können bspw. die Zeichen 3 bis 5 aus einer Zeichenkette entnommen werden):



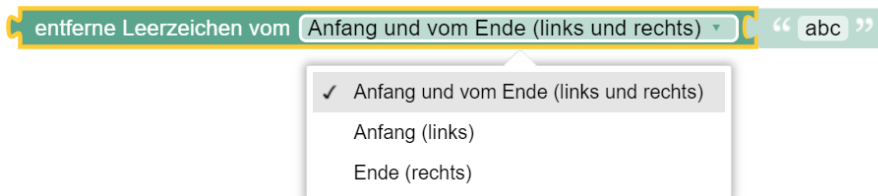
**Texte Verändern:** Vorhandene Texte (entweder als Ausdrücke oder aus Variablen vom *Datentyp String*) können durch die Anwendung von Operatoren verändert werden.

- Folgender Operator kann verwendet werden, um Text in Grossbuchstaben oder in Kleinbuchstaben umzuwandeln:



Die Option Substantive konvertiert die übergebene Zeichenfolge in eine Folge von Worten mit großem Anfangsbuchstaben (außer Zeichenketten, die vollständig in Großbuchstaben geschrieben sind).

- Führende, abschließende oder führende und abschließende Leerzeichen können durch folgenden Operator entfernt werden:



### 5.7.3.6. Operatoren für Zeiten und einfache Zeit-Operationen

Der Abschnitt *Date & Time* der *Palette* enthält *Blockly*-Elemente zur Verwendung von Zeiten innerhalb von Ablaufdefinitionen.

**Festhalten von Zeitpunkten:** Variablen vom *Datentyp DateTime* können Zeitstempel zugewiesen werden.

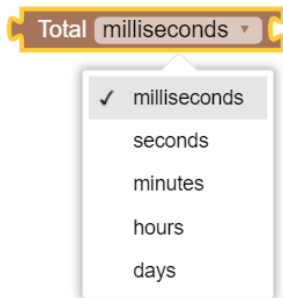
**Ermitteln von Zeitdifferenzen:** Vollständiges Beispiel: Folgender *Blockly*-Code misst die Zeit für die Bearbeitung von Task 1 bis 4. Dafür wird zunächst der Startzeitpunkt festgehalten, und nach der Bearbeitung der Aufgaben wird die Zeitdifferenz ermittelt und in Sekunden umgewandelt:

```

# Variable "StartTimeSection1" stores timestamp
setze StartTimeSection1 auf Now
# Section 1: Tasks 1-4
Show Item LDS202301 - Task01
Show Item LDS202301 - Task02
Show Item LDS202301 - Task03
Show Item LDS202301 - Task04
# Variable "SecondsSection1" stores time difference
setze SecondsSection1 auf Total seconds - Now - StartTimeSection1

```

### Umrechnen von Zeitmaßen



### 5.7.3.7. Operatoren für Listen

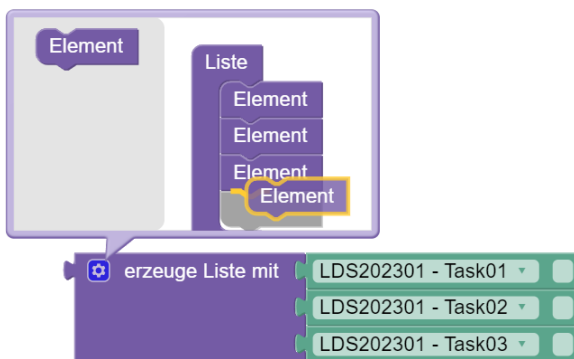
Der Abschnitt *Lists* der *Palette* enthält *Blockly*-Elemente zur Erstellung und Verwendung von Listen.

**Liste erstellen:** Es stehen verschiedene Optionen zur Verfügung, wie Listen erstellt werden können.

- Listen können aus bestehenden Elementen erstellt werden:



Die Anzahl der Elemente des Operators *erzeuge Liste mit* kann per Drag-and-Drop konfiguriert werden, nachdem das *Zahnrad*-Symbol angeklickt wurde:

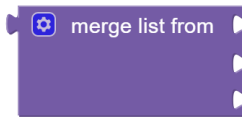


Bei der Erstellung von Listen ist darauf zu achten, dass der *Blockly*-Editor keine Prüfung der Datentype vornimmt. Listen mit Werten unterschiedlicher Datentypen können (fälschlicher Weise) erstellt werden, führen aber zu keinem funktionierenden Testablauf.

- Listen können durch Wiederholung eines Elements erstellt werden:



**Verbinden von Listen:** Bestehende Listen können zusammengeführt werden mit folgendem Operator:



**Teillisten:** Aus Listen kann mit folgendem Operator eine Teilliste ausgewählt werden:



Weitere Optionen des Operators für *bis*: *bis von hinten* und *bis letztes*.

**Eigenschaften von Listen:** Folgende Operatoren stehen zur Verfügung um Eigenschaften einer Liste abzufragen:

- Folgender Operator gibt *wahr* zurück, wenn die verbundene Liste leer ist:



- Folgender Operator gibt die Länge der Liste zurück:

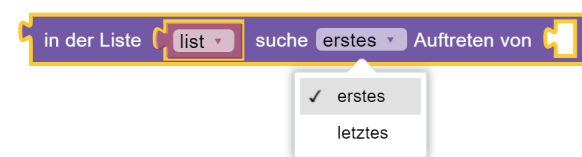


- Folgender Operator gibt die distinkten Elemente einer Liste zurück



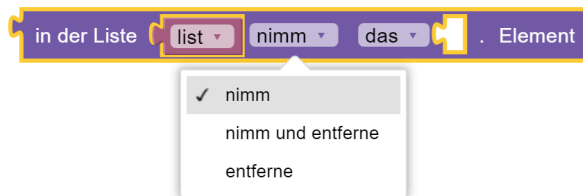
**Suchen und Ersetzen:** Folgende Operatoren stehen zum Suchen und Ersetzen von Elementen in Listen zur Verfügung:

- Folgender Operator findet Elemente in Listen:



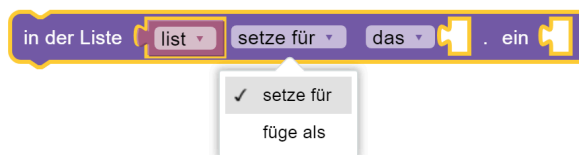
- Folgender Operator gibt / entfernt oder ersetzt in einer Liste und gibt das Element zurück:





Weitere Optionen des Operators für *das*: *von hinten das / Erste / Letzte* und *Zufällig*.

- Folgender Operator ersetzt und fügt in einer Liste ein:



Weitere Optionen des Operators für *das*: *von hinten das / Erste / Letzte* und *Zufällig*.

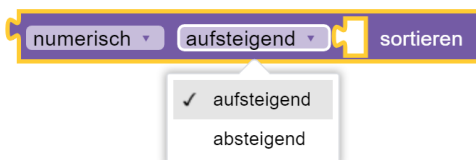
**Umwandlung von Listen und Text:** Liste und Text können über Trennzeichen umgewandelt werden.

- Folgender Operator erstellt einen Text aus einer Liste oder eine Liste aus einem Text:



**Listen Sortieren:** Elemente in Listen können auch sortiert werden.

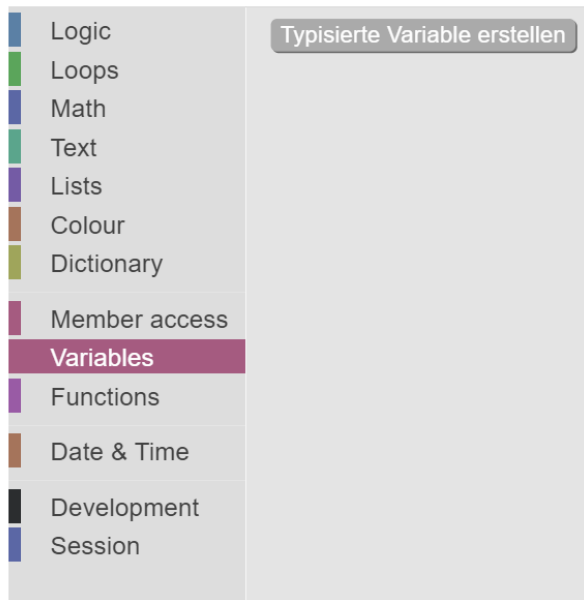
- Folgender Operator gibt die distinkte Elemente einer Liste zurück:



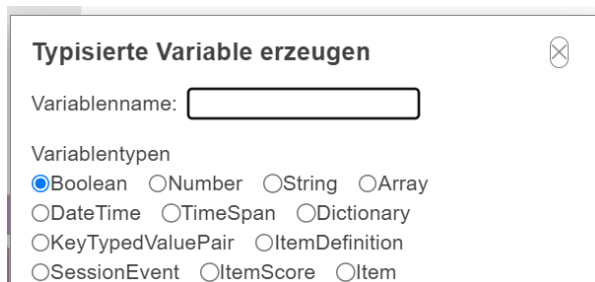
### 5.7.3.8. Blockly-Variablen

Der Abschnitt *Variables* der *Palette* enthält *Blockly*-Elemente zur Erstellung und Verwendung von Variablen.

**Variable Erstellen:** Um eine *Blockly*-Variable zu erstellen enthält die *Palette* die *Typisierte Variable Erstellen*:



- *Blockly*-Variablen haben immer einen *Variablenamen* und *Datentyp*:



**Einfache Datentypen und Wertzuweisungen:** Folgende basale Datentypen werden unterstützt:

- *Boolean*: Logische Wahrheitswerte und Logische Ausdrücke (*wahr* oder *falsch*)



- *Number*: Datentyp für Zahlenwerte (mit und ohne Dezimalstelle)

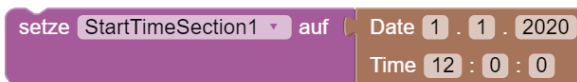


- *String*: Textwerte bzw. Zeichenketten



Für Zeiten werden folgende Datentypen bereitgestellt:

- *DateTime*: Datum und Uhrzeit



- *TimeSpan*: Zeitspanne

setze TimeSpanSection1 auf

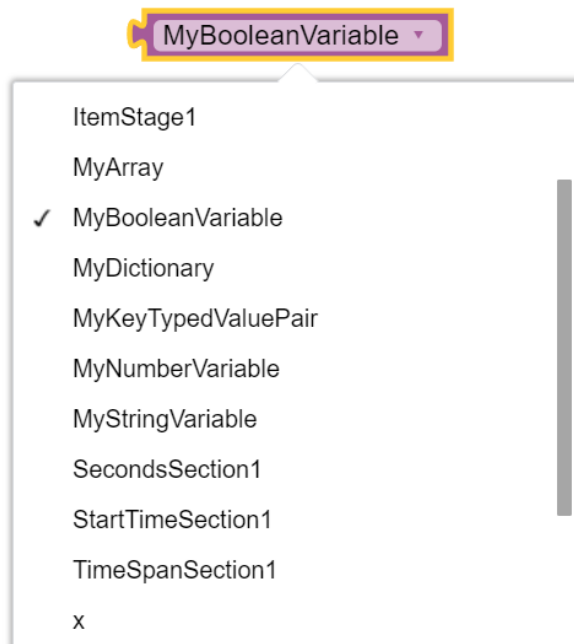
**Datentypen für mehrere Werte:** Neben den basalen Datentypen werden auch Datentypen für mehrere Werte unterstützt:

- *Array*: Datentyp für Listen

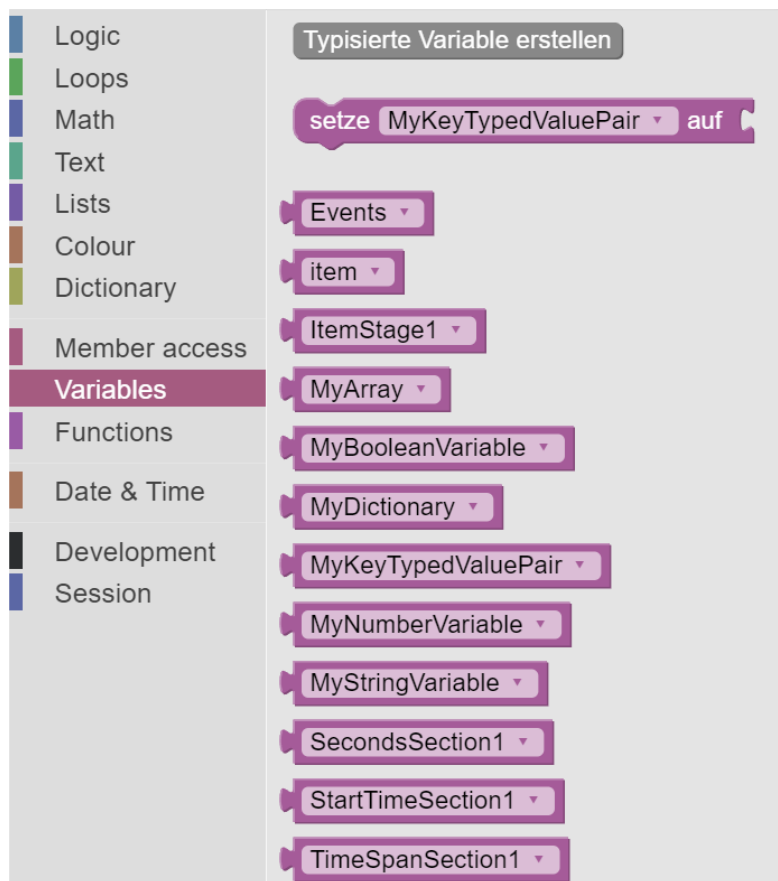


- *Dictionary*: (Dokumentation fehlt)
- *KeyTypedValuePairs*: (Dokumentation fehlt)

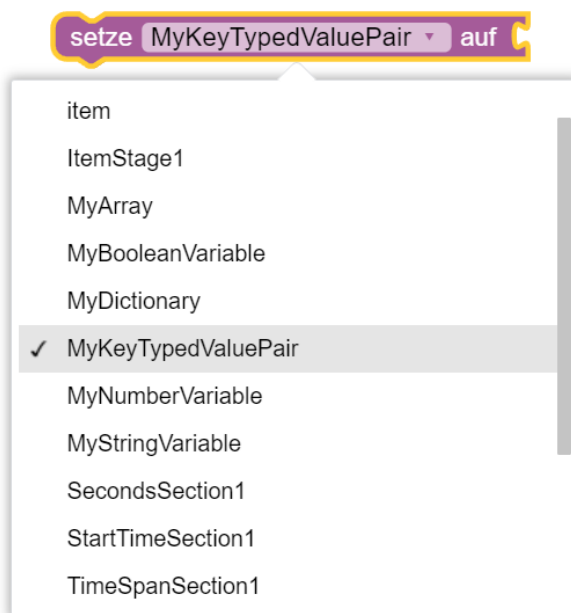
**Variablenwerte Verwenden:** Für die Verwendung von Variablenwerten, können *Blockly*-Elemente mit *Eingängen* folgende Komponenten aufnehmen:



- Welche Variable verwendet wird, kann dabei ausgewählt werden. Für definierte Variablen findet sich dafür jeweils auch ein *Blockly*-Element im Abschnitt *Variables* der *Palette*:



- In der Palette findet sich auch ein *Blockly*-Element vom Typ *setze ... auf*. In diesem kann ebenfalls ausgewählt werden, welchen Wert der Variable es setzt:

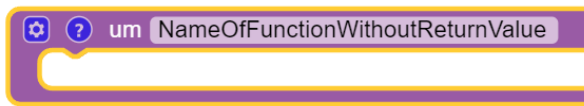


### 5.7.3.9. Blockly-Funktionen

Der Abschnitt *Functions* der *Palette* enthält *Blockly*-Elemente zur Verwendung von Funktionen innerhalb von Ablaufdefinitionen. Funktionen kombinieren *Blockly*-Code, so dass dieser nur einmal definiert aber mehrfach verwendet werden kann.

**Definieren von Funktionen:** Es können zwei verschiedene Formen von Funktionen definiert werden.

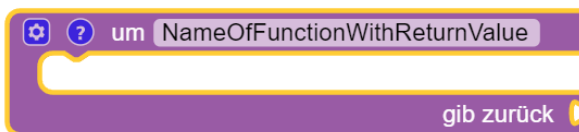
- Funktionen ohne Rückgabewert:



Funktionen ohne Rückgabewert können, damit sie angerufen werden, im Ablauf einfach mit vorherigen und nachfolgenden *Blockly*-Elementen verbunden werden (d.h. sie haben eine Verbindung nach oben und unten):



- Funktionen mit Rückgabewert:



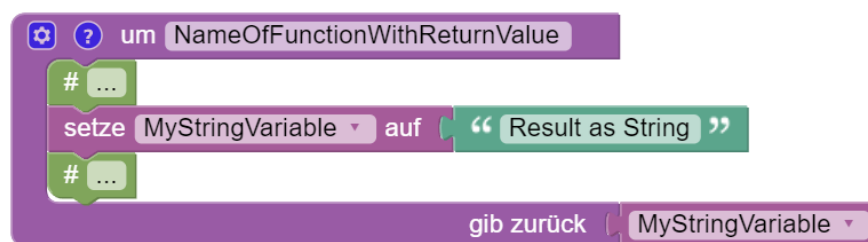
Funktionen mit Rückgabewert können in einem Zuweisungs-Block aufgerufen werden (d.h. sie haben eine Verbindung nach links):



Zu welchem Typ eine Zuweisung sinnvoll ist, hängt von dem Typ des Rückgabewerts ab.

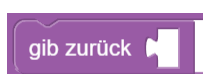
**Definieren von Rückgabewerten von Funktionen:** Funktionen werden durch spezielle *Blockly*-Elemente definiert, die an einer beliebigen Stelle im Code-Editor eingefügt werden können.

- *Rückgabewerte* können für Funktionen mit Rückgabewert definiert werden. Der Rückgabewert kann direkt an die Funktionsdefinition neben *gib zurück* angefügt werden:

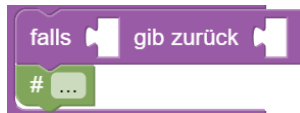


Ergänzend stehen die folgenden zwei *Blockly*-Elemente zur Verfügung, die nur innerhalb einer Funktionsdefinition (mit Rückgabewert) verwendet werden können:

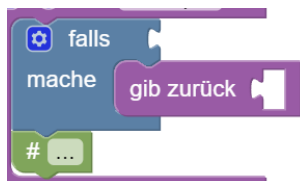
- Der Operator *gib zurück* erlaubt die Rückgabe eines Wertes. Danach können innerhalb der Funktion keine weiteren *Blockly*-Elemente in den Ablauf platziert werden (d.h. der *gib zurück*-Operator hat keine Verbindung nach unten):



- Der Operator *falls gib zurück* Operator gibt einen Wert nur dann zurück, wenn eine Bedingung erfüllt ist. Ist die Bedingung erfüllt, endet die Abarbeitung des Ablaufs in der Funktion, ist die Bedingung nicht erfüllt, wird die Bearbeitung fortgesetzt (d.h. der *falls gib zurück*-Operator hat eine Verbindung nach unten):



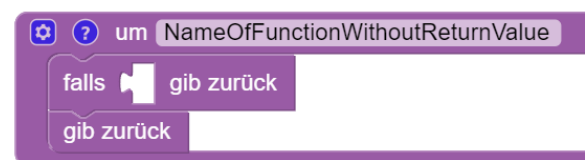
- Der *falls gib zurück* Operator ist also identisch mit folgender Kombination von Operatoren:



- Beide Operatoren (*falls gib zurück* und *gib zurück*) können nicht außerhalb von Funktionen verwendet werden:

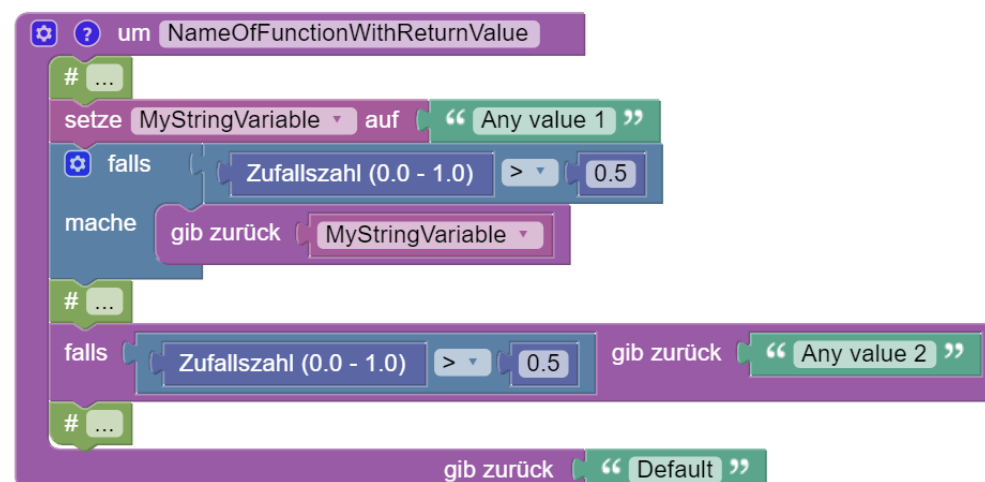


- Die beide Operatoren (*falls gib zurück* und *gib zurück*) können innerhalb von Funktionen ohne Rückgabewert verwendet werden, um die Abarbeitung von Funktionen zu beenden (aber nicht zum Rückgeben von Werten):



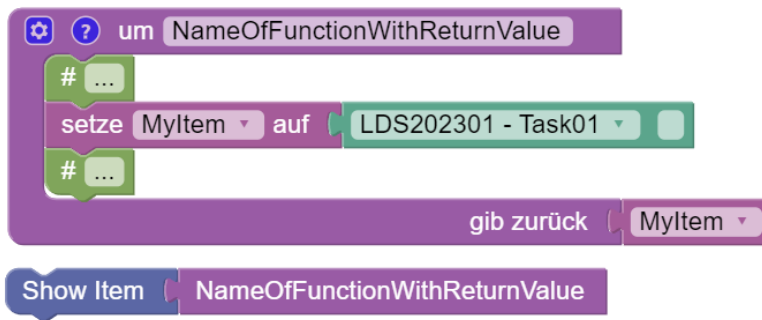
Beispiel:

- Die folgende Funktion gibt den Wert der Variablen *MyStringVariable* (*Any value 1*) in 50% der Fälle zurück (d.h. wenn eine erste gezogene Zufallsvariable größer 0.5 ist). In den anderen 50% der Fälle, wird eine weitere Zufallsvariable gezogen, und wenn diese größer 0.5 ist, dann wird der Text *Any value 2* zurückgegeben. Ist auch dies nicht der Fall, dann wird der Text *Default* zurückgegeben:

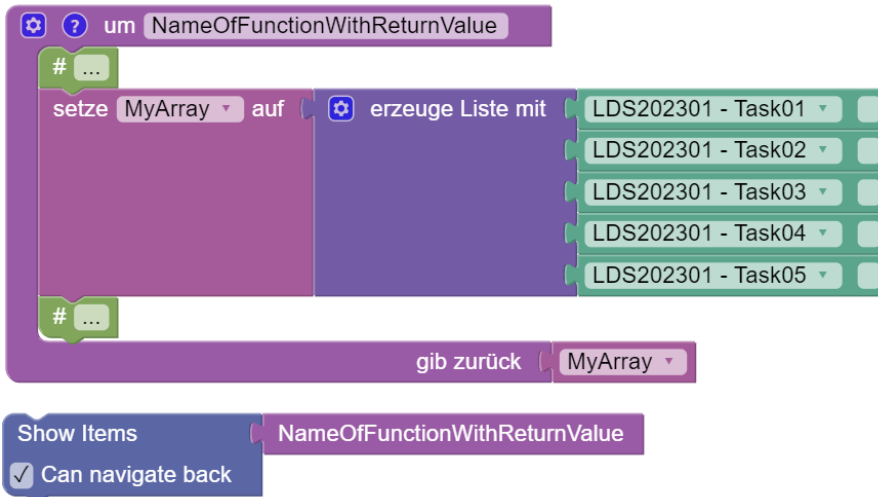


Rückgabewerte sind typisiert. Die Ablaufsteuerung unterstützt auch Funktionen, die ...

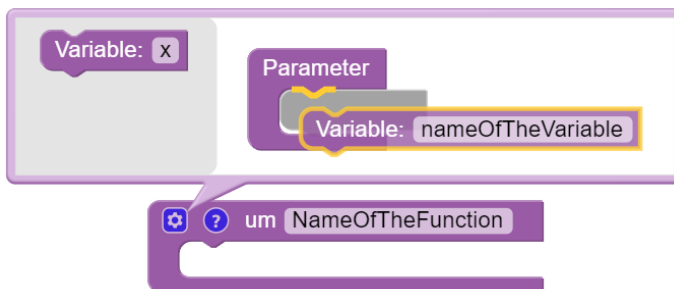
- ... einzelne *Tasks* zurückgeben:



- ... Listen von *Tasks* zurückgeben:

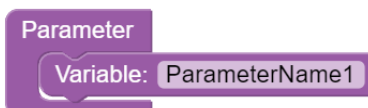


**Definieren von Aufrufparametern von Funktionen:** Funktionen können auch Parameter verwenden, die beim Aufruf der Funktion zu übergeben sind (*Aufrufparameter*). Die Definition von Aufrufparametern ist nach einem Klick auf das kleine Zahnrad-Symbol eines Funktions-Blocks möglich:



Passend zur Definition der Parameter, erfolgt dann der Aufruf der Funktion durch Übergabe:

- Definition eines Parameters:

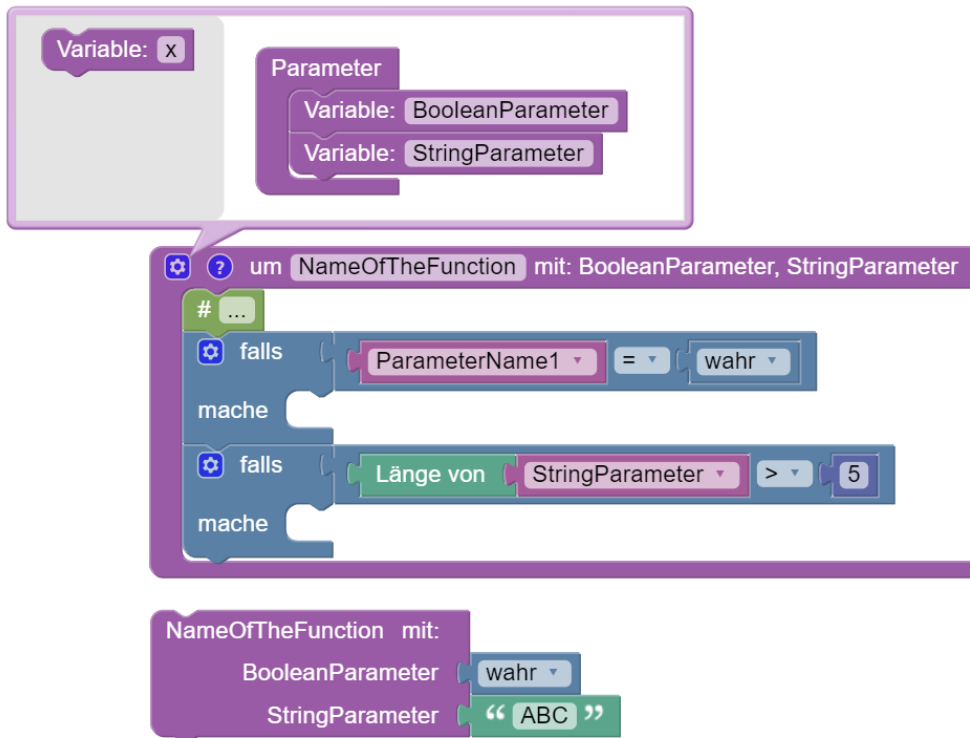


- Aufruf der Funktion mit Angabe von Wert:

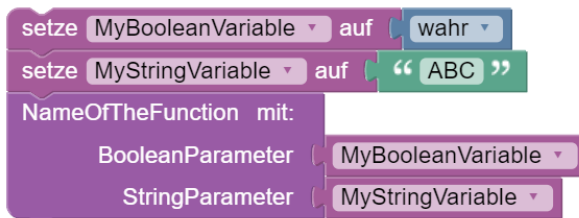


Beispiel:

- Das folgende Beispiel zeigt eine Funktion mit zwei Parametern, deren Verwendung innerhalb der Funktion am Beispiel von Bedingungen und den Aufruf der Funktion mit festen Werten darstellt:



- Alternativ kann die Funktion natürlich auch mit Variablen aufgerufen werden:



### 5.7.3.10. Nutzung von Itemergebnissen in der Ablaufsteuerung

(Dokumentation folgt)

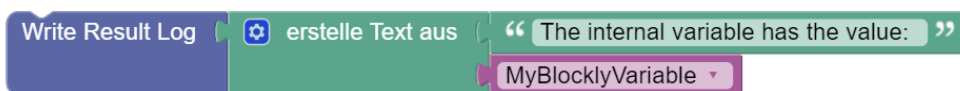
### 5.7.3.11. Blockly-Operatoren zum Kodieren fehlender Werte

(Dokumentation folgt)

### 5.7.3.12. Blockly-Operatoren zum schreiben von Daten

(Dokumentation folgt)

**Log-Daten:** Folgender Operator kann genutzt werden, um Informationen direkt in die Log-Daten zu speichern:





**Ergebnis-Daten:** (Dokumentation folgt)

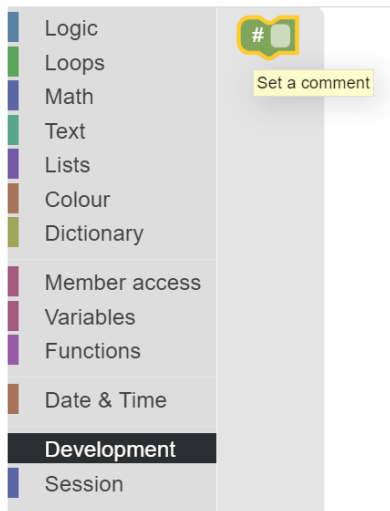
**Monitoring-Daten:** (Dokumentation folgt)

### 5.7.4. Kommentieren von Blockly-Code

Der *IRTLib Editor* unterstützt zwei verschiedene Optionen zur Kommentierung von *Blockly*-Code.

#### 5.7.4.1. Kommentare als Blockly-Elemente

Kommentare, die im Ablauf dauerhaft sichtbar sein sollen, können über die Platte im Abschnitt *Development* hinzugefügt werden:

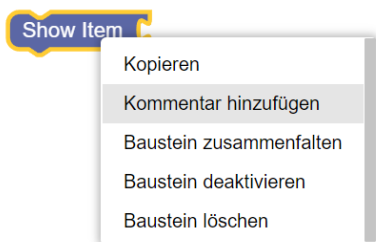


Diese Kommentare können wie Blockly-Operatoren verschoben werden und zeigen einzeiligen Kommentartext.

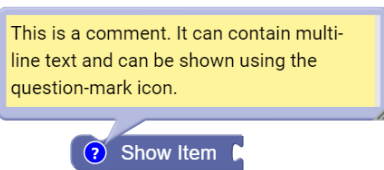


#### 5.7.4.2. Ausführliche Kommentare an Blockly-Elementen

Für ausführlichere Kommentare kann über das Kontextmenü jeder Block mit einem Kommentar hinzugefügt (und wenn vorhanden gelöscht) werden:



Diese Kommentare können mehrere Zeilen umfassen und werden dargestellt, wenn auf das kleine ?-Icon eines Blocks geklickt wurde.

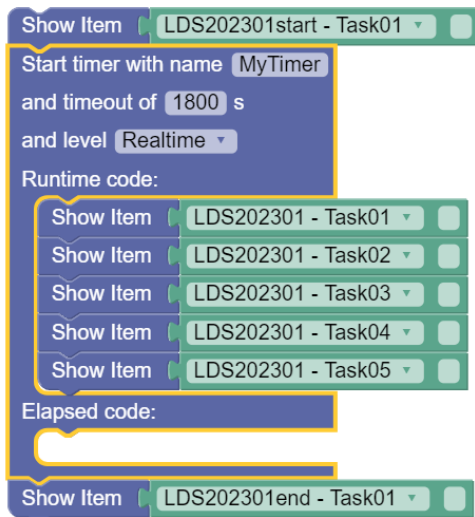


## 5.7.5. Darstellung von Blockly-Code

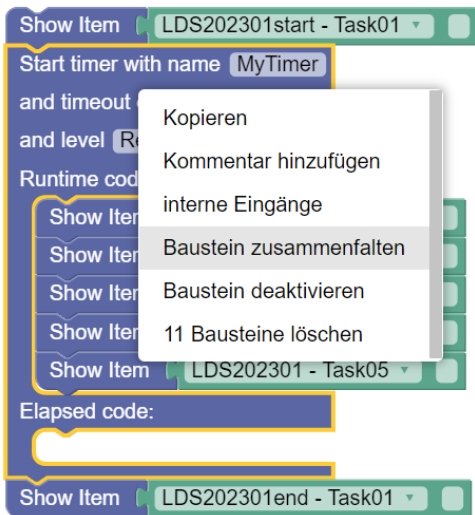
### 5.7.5.1. Entfalten / Zusammenfallen

Große und komplexe Abläufe können im *Blockly*-Editor unter Umständen unübersichtlich werden. Um für eine Betrachtung nicht benötigte *Blockly*-Elemente auszublenden, ohne die Funktion der Ablaufdefinition zu verändern, können Blöcke *zusammengefallen* werden: Das wird in folgendem Beispiel illustriert:

- Entfaltete (d.h. vollständige) Darstellung des markierten Blocks:



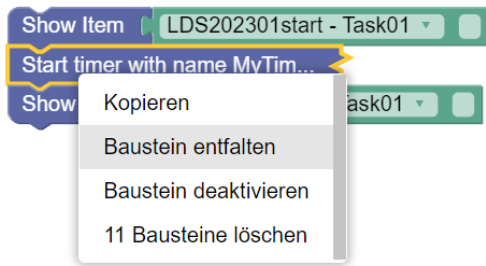
- Option zum *zusammenfallen* des Blocks im Kontextmenü:



- *Zusammengefallene* Darstellung des Blocks innerhalb der Ablaufdefinition:



- Option zum *entfalten* des Blocks im Kontextmenü:



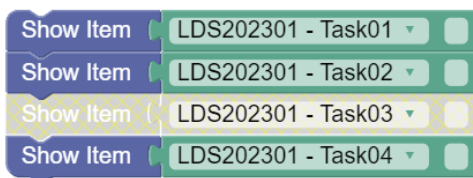
Das *zusammenfallen* / *entfalten* von *Blockly*-Elementen ändert nichts an der Funktion einer Ablaufdefinition und dient nur der übersichtlicheren Anordnung von komplexen Ablaufdefinitionen.

### 5.7.5.2. Deaktivieren / Aktivieren

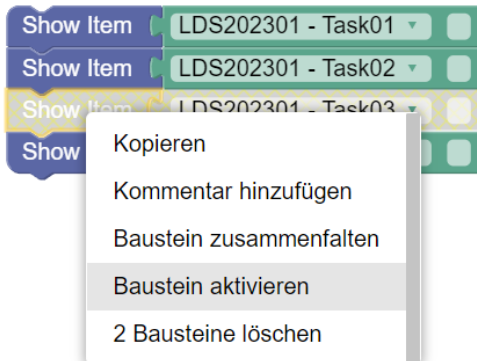
Hinweis: Diese Funktion ist gerade noch in Entwicklung.

Der *Blockly*-Editor bietet die Option, *Blockly*-Elemente, statt sie zu löschen, nur zu deaktivieren. Deaktivierte *Blockly*-Elemente bleiben in der Ablaufdefinition enthalten, werden aber nicht ausgeführt.

In folgendem Beispiel ist der Block zum Anzeigen des Tasks 3 deaktiviert, d.h. es werden nur Task 1, 2 und 4 angezeigt:

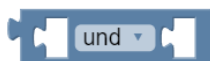


Aktivieren bzw. Deaktivieren von *Blockly*-Elementen erfolgt über das Kontextmenü:



**Internal / External:** Einige *Blockly*-Elemente mit *Eingängen* (d.h. Stellen, an denen man weitere Blöcke verbinden kann) erlauben zwischen zwei Darstellungsformen zu wechseln.

- Internal: Die *Eingänge* sind innerhalb der Blöcke angeordnet.

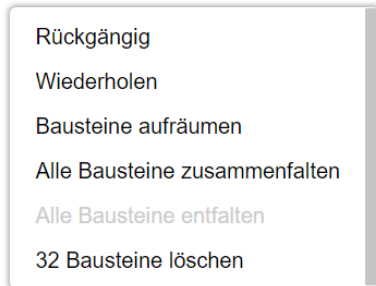


- Exnternal: Die *Eingänge* sind an der Seite der Blöcke angeordnet.



Beide Darstellungsformen sind bzgl. der Funktionalität äquivalent.

**Aufräumen:** Im Kontextmenü des *Blockly*-Editors, welches durch Klick in einen leeren Bereich geöffnet werden kann, ist die Funktion *Bausteine aufräumen* enthalten:



Durch Aufruf von *Bausteine aufräumen* werden alle *Blockly*-Elemente im *Blockly*-Editor vertikal untereinander ausgerichtet.

## 5.8. Routing zwischen Erhebungsteilen

Wenn mehrere *Erhebungsteile* für eine *Studie* definiert sind, kann die Abfolge von Erhebungsteilen definiert werden, in welcher Befragte oder Testpersonen die Inhalte der *Erhebungsteile* präsentiert bekommen.

Neben einfachen linearen Abläufen können Abläufe von mehreren Erhebungsteilen auch mit einem *Blockly*-basierten Routing konfiguriert werden.

Eine detaillierte Beschreibung zum *Routing zwischen Erhebungsteilen* findet sich hier in der eingebetteten Hilfe:

### 📌 Eingebettete Programmhilfe

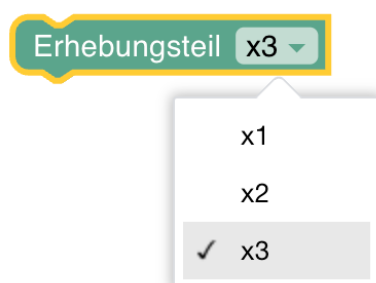
#### 5.8.1. Zusammenfassung zu Routing zwischen Erhebungsteilen

Die Reihenfolge von *Erhebungsteilen* kann mit Hilfe von *Blockly* definiert werden (analog zur Definition der Reihenfolge von *Items* innerhalb von *Erhebungsteilen*). Diese Option ist verfügbar, wenn in der Grundkonfiguration zu einer Studie (in der Ansicht *Übersicht*) die Option *Routing für Erhebungsteile aktivieren* gewählt ist.

Für die allgemeinen Grundlagen zur Verwendung von *Blockly* im *IRTlib Editor* siehe die Hilfe zum *Routing innerhalb von Erhebungsteilen*.

Funktionen, die nur im *Routing zwischen Erhebungsteilen* zur Verfügung stehen, sind:

- Erhebungsteil Anzeigen



Dieser *Blockly*-Operator ersetzt *Show Item* innerhalb von Erhebungsteilen.

- Erfolgreiches Login



Dieser *Blockly*-Operator hat den Wert *wahr*, wenn vor der Anzahl der maximalen Versuche (hier: unendlich, d.h. unbegrenzt oft) gültige Login-Informationen angegeben wurden.

Hinweis: Änderungen an der Sicht *Routing* zwischen *Erhebungsteilen* müssen über das Disketten-Symbol gespeichert oder mit dem Rückgängig-Symbol verworfen werden:



**Teil II.**

## **Datenerhebung / Data Collection**

# 6. Datenerhebung: Übersicht / Data Collection: Overview

## 6.1. Übersicht: Schritte zur Verwendung eines IRTlib Player zur Datenerhebung

Nachdem eine *Studie* mit dem *IRTlib Editor* angelegt und konfiguriert wurde, muss eine finalisierte *Version* dieser Konfiguration erstellt werden. Versionen *versiegeln* und vervollständigen alle Konfigurationen und haben eine eindeutige Versionsnummer (bezeichnet als *Revision*). Die Verwendung von *Revisionen* macht die Durchführung von Datensammlungen mit dem *IRTlib Editor* und *IRTlib Player* reproduzierbar, da die Revisionsnummer einer Konfiguration auch in den Datensätzen gespeichert wird.

- **Konfigurationen Prüfen:** Vor der Fertigstellung und dem *Versiegeln* einer Version ist es ratsam, alle Einstellungen noch einmal zu überprüfen. Der *IRTlib-Editor* stellt dafür eine zusätzliche Funktion der *Validierung* zur Verfügung.
- **Versiegelte Version Erstellen:** Wenn keine weiteren Änderungen erforderlich sind, kann die Version *versiegelt* werden. Dies geschieht durch das Auswählen der noch nicht gespeicherten Änderungen und einen Klick auf das Schloss-Symbol im *IRTlib Editor* in der Ansicht *Veröffentlichen*, in welcher die *Studienrevisionen* angezeigt werden.
- **Version Exportieren:** Versionen von Studien, die im *IRTlib-Editor* verfügbar sind, können exportiert werden. Der Export der Konfiguration ist notwendig, bevor diese mit dem *IRTlib-Player* verwendet werden können. Beim Exportieren wird die vollständige Studienkonfiguration inklusive der importierten *CBA ItemBuilder*-Inhalte als ZIP-Archiv heruntergeladen.
- **Studie in IRTlib Player Importieren:** Exportierte Versionen von Studien aus dem *IRTlib-Editor* können zur Verwendung in einen *IRTlib-Player* importiert werden. Für einzelne *Studien* gibt es einen automatischen Modus, wenn mehrere *Studien* in einem *IRTlib Player* simultan verwendet werden sollen, kann diese manuell erfolgen.
- **Testen der Studie:** Bevor mit der eigentlichen Datenerfassung begonnen werden kann, sollte jede Konfiguration zunächst mit synthetischen Testfällen (d.h. systematisch) getestet werden.

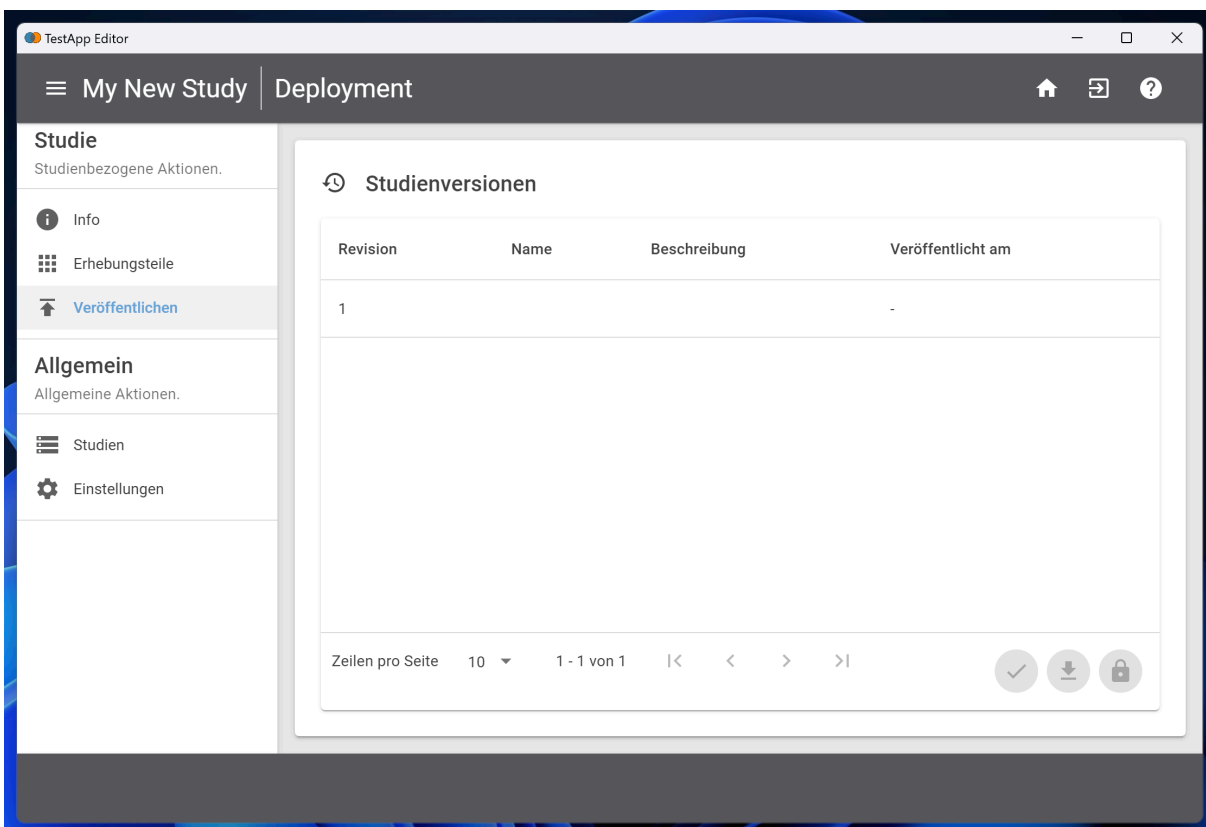
Falls in einer exportierten Studie noch Probleme festgestellt werden, ist es möglich, zur Vorbereitung (siehe [Vorbereitung von Studien](#) und [Vorbereitung von Erhebungsteilen](#)) zurückzukehren, die Studiendefinition und die Konfiguration der Testteile zu ändern, eine weitere versiegelte Version zu erstellen und mit der geänderten Konfiguration fortzufahren.

## 7. Datenerhebung: Veröffentlichen & Exportieren / Data Collection: Publish & Export

Die Konfiguration von *Studien* und den darin enthaltenen *Erhebungsteil(en)* erfolgt im *IRTlib Editor*. Änderungen werden innerhalb des *IRTlib Editors* immer gespeichert, wenn das Disketten-Symbol unten rechts geklickt wird. Während der Vorbereitung einer Studie speichert man die Änderungen, wenn sie übernommen werden sollen. Aber sobald die Vorbereitung abgeschlossen ist, sollten Änderungen nicht mehr möglich sein oder zumindest nachvollzogen werden, damit die Version im *IRTlib Editor* der Version im *IRTlib Player* entspricht.

Um diesen Prozess der Verwendung von *IRTlib Editor* und *IRTlib Player* zu unterstützen, ist folgendes Konzept implementiert. Damit zu jeder Zeit eindeutig benennbar ist, mit welcher Konfiguration einer *Studie* eine Datenerhebung durchgeführt wird, muss die Konfiguration vor der Übertragung an einen *IRTlib Player* versiegelt werden.

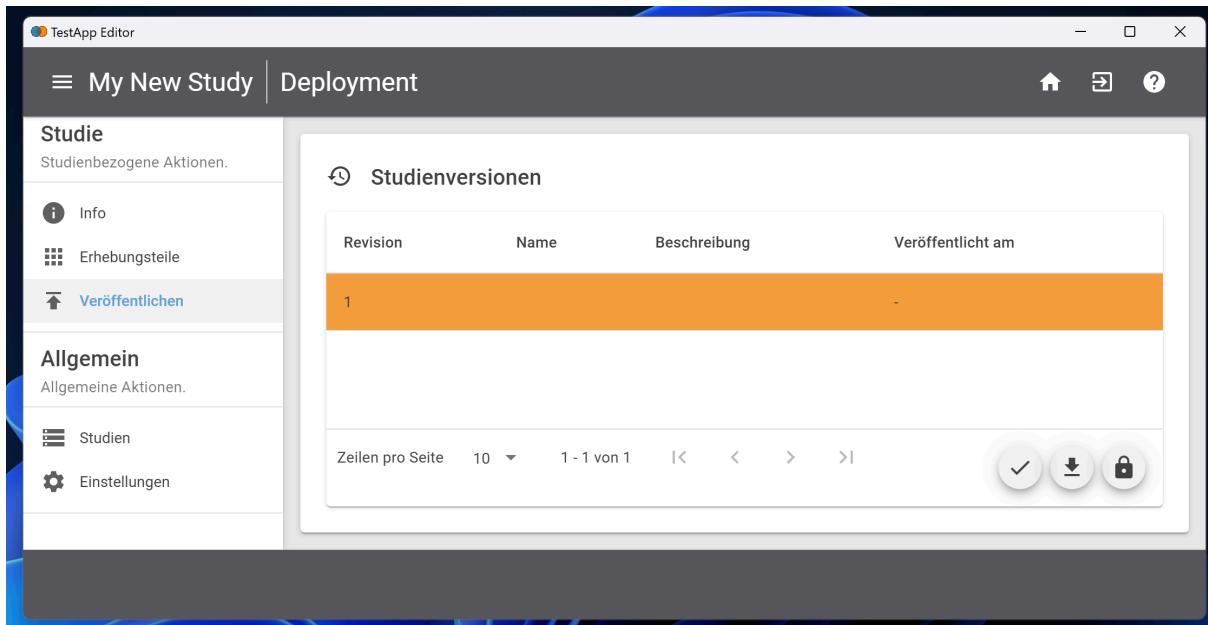
Das geschieht in der Ansicht *Veröffentlichen* einer Studie, in welcher die *Studienversionen* aufgelistet werden. Bei einer neuen *Studie* sieht diese Ansicht zunächst so aus:



In diesem Zustand können Sie Änderungen an der Einstellung der *Studie* und aller enthaltenen *Erhebungsteile* vornehmen.

Die Schaltflächen zum *Validieren*, *Herunterladen* und *Veröffentlichen* von Studien sind ausgegraut, weil keine Revision markiert ist. Durch Klick auf die Zeile mit der noch nicht veröffentlichten *Revision 1* sind die Schaltflächen aktivierbar:





Bevor Sie nun fortfahren, prüfen Sie bitte, ob Sie an alles gedacht haben. Dafür dient die folgende *Checkliste*.

## 7.1. Checkliste vor dem Veröffentlichen

- Ist die *Anmeldung* konfiguriert?

Damit nach dem Start des *IRTlib Players* die richtige *Studie* gestartet werden kann, muss ein zu der geplanten Verwendung passender *Loginmodus* konfiguriert sein. Die Definition des *Loginmodus* ist in der [Studienkonfiguration](#) im Abschnitt *Login* möglich.

- Ist ein *Testleitermenü* konfiguriert?

Wenn im Offline *IRTlib Player* der Kiosk-Modus aktiviert ist, kann das Beenden der Anwendung ohne konfiguriertes *Testleitermenü* schwierig bzw. unmöglich sein. Die Definition von Tastenkombination und Passwörtern mit Rollen erfolgt in der [Studienkonfiguration](#) im Abschnitt *Testleitermenü*.

- Sind die *Items* eingefügt?

Die Assessmentinhalte werden in einem oder mehreren [Erhebungsteilen](#) konfiguriert. Die meisten *CBA ItemBuilder-Tasks* werden sich im Abschnitt *Items* eines *Erhebungsteils* befinden.

- Sind die Laufzeitumgebungen (*Runtimes*) vorhanden?

Die Konfiguration von *Runtimes* erfolgt in den [Einstellungen](#).

Wenn Sie diese Checkliste geprüft haben, können Sie fortsetzen, wie im nächsten Abschnitt beschrieben.

## 7.2. Veröffentlichen & Exportieren

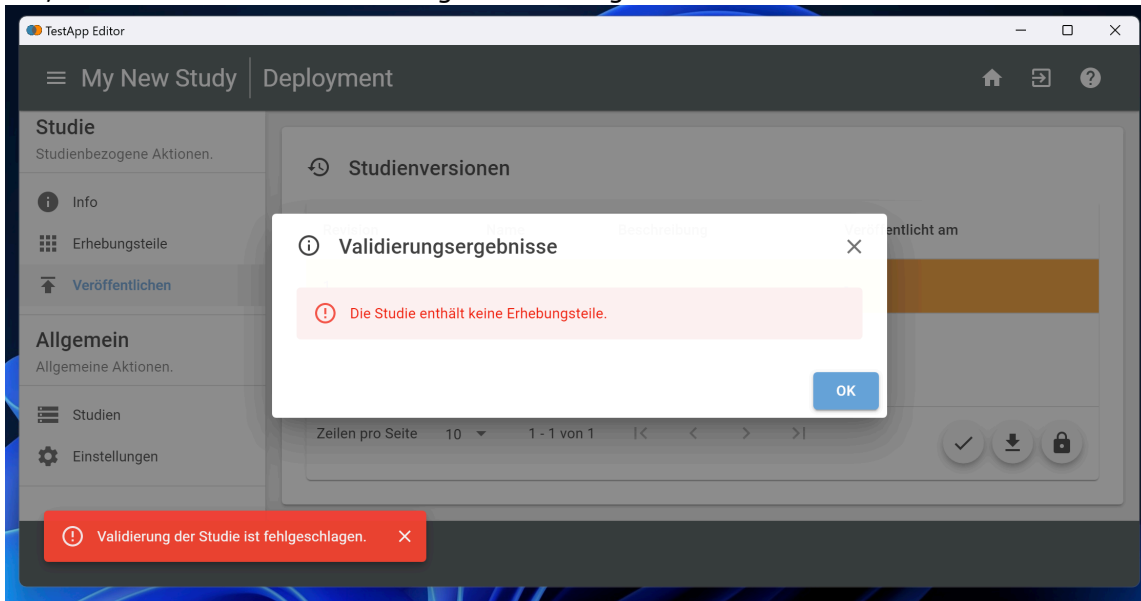
Der Prozess zum *Validieren*, *Veröffentlichen* und *Herunterlade* von Studienkonfigurationen ist in der eingebetteten Hilfe beschrieben:

### 7.2.1. Veröffentlichen

Bevor eine *Version* einer *Studie* zu veröffentlichen wird, kann über die Schaltfläche *Validieren* geprüft werden, ob die *Studie* fehlerfrei konfiguriert wurde.



Eine *Studie* muss beispielsweise immer mindestens einen *Erhebungsteil* enthalten. Ist das nicht der Fall, erscheint beim *Validieren* die folgende Meldung:



Hinweis: Werden keine Fehler beim *Validieren* gefunden, wird keine weitere Nachricht angezeigt und die *Studie* kann veröffentlicht werden.

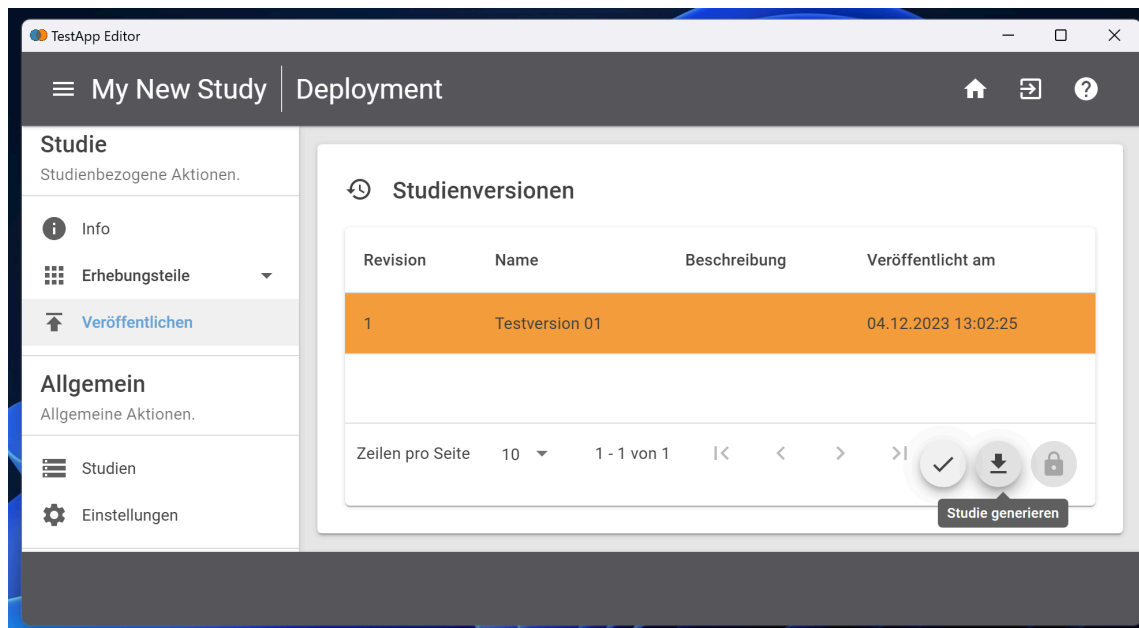
Wenn keine Fehler mehr enthalten sind und die *Studie* für den Export vorbereitet werden soll, kann eine *Version* erstellt werden. Dazu dient die folgende Schaltfläche:



Danach erscheint der folgende Dialog:



Versionen benötigen eine *Bezeichnung* und können optional mit einer *Beschreibung* dokumentiert werden. Nach dem Bestätigen mit *Speichern* erscheint die *Studie* mit einem *Veröffentlichungsdatum* in der Liste:



## 7.2.2. Exportieren

Veröffentlichte Versionen können über die folgende Schaltfläche *Version* heruntergeladen werden:



Nach dem Betätigen dieser Schaltfläche erscheint ein *Save As / Speichern unter* - Dialog, und die vollständige Studienkonfiguration kann als ZIP-Archiv gespeichert werden. Dieses ZIP-Archiv kann dann in einem nächsten Schritt verwendet werden, um es in einen *IRTLib Player* zu laden.

## 7.2.3. Studienversionen

Die *IRTLib Software* ist so erstellt, dass im Datensatz immer die *Revision* der *Studie* gespeichert wird, sodass nachvollzogen werden kann, mit welcher Konfiguration (d.h. welche *CBA ItemBuilder*-Inhalten und Einstellungen im *IRTLib Editor*) Daten erhoben werden.

**Unveröffentlichte Versionen:** Es ist auch möglich, nicht veröffentlichte Versionen herunterzuladen und in einem *IRTLib Player* auszuprobieren. Zur Sicherheit wird dann aber eine farbliche Markierung während der Testung angezeigt, damit klar ersichtlich ist, dass es sich hierbei nur um eine Vorabversion handelt. Mit nicht veröffentlichten Studien sollten niemals Daten erhoben werden.

**Änderungen veröffentlichter Versionen:** Versionen *versiegeln* die Konfiguration, und machen Änderungen durch eine *Revisionsnummer*, eine *Bezeichnung* und eine optionale *Beschreibung* nachvollziehbar. Änderungen nach dem Versiegeln einer Version sind aber weiterhin möglich. Bevor veränderte Versionen für eine Datenerhebung verwendet werden können, müssen diese aber ebenfalls versiegelt werden, d.h. eine neue *Version* muss über *Veröffentlichen* erstellt werden.

Eine Datenerhebung mit einem *IRTLib Player* kann erfolgen, wenn eine veröffentlichte *Studie* aus dem *IRTLib Editor* als ZIP-Archiv heruntergeladen wurde.

# 8. Datenerhebung: In IRTlib Player Importieren / Data Collection: Import into IRTlib Player

## 8.1. Konfiguration Importieren

Im Folgenden ist beschrieben, wie eine mit einem *IRTlib Editor* erstellte Studienkonfiguration, die als ZIP-Archiv vorliegt, verwendet werden kann.

### Veröffentlichte Version erforderlich

Für die Datenerhebung mit einem *IRTlib Player* ist eine veröffentlichte Version einer *Studie* notwendig.

Wenn eine *versiegelte* Studienkonfiguration aus dem *IRTlib Editor* exportiert wurde, kann diese in einen *IRTlib Player* integriert werden.

Zurzeit werden zwei Optionen unterstützt:

- Automatischer Import aus ZIP-Archiv
- Manueller Import aus ZIP-Archiv


Der *Automatische Import* ist nur für die erste *Studie* in einem *IRTlib Player* möglich. Sollen mehrere *Studien* in einem *IRTlib Player* parallel verwendet werden, muss ein *Manueller Import* durchgeführt werden.


### 8.1.1. Automatischer Import

Für einen *automatischen Import* einer als ZIP-Archiv vorliegenden *Studie* in einem Offline *IRTlib Player*, kann der Player zunächst über die ausführbare Datei `TestApp.Player.Desktop.exe` gestartet werden.

Wurde dieser *IRTlib Player* noch nicht mit einer *Studie* konfiguriert (d.h. wurde der Player bspw. wie unter [Download](#) beschrieben direkt von dem [Github](#)-Repositoy heruntergeladen), dann erscheint folgender Dialog:

#### Keine Studie vorhanden

 Dieses Programm benötigt eine Studie. Stellen Sie diese bitte bereit oder beenden Sie das Programm.

 **STUDIE BEREITSTELLEN**

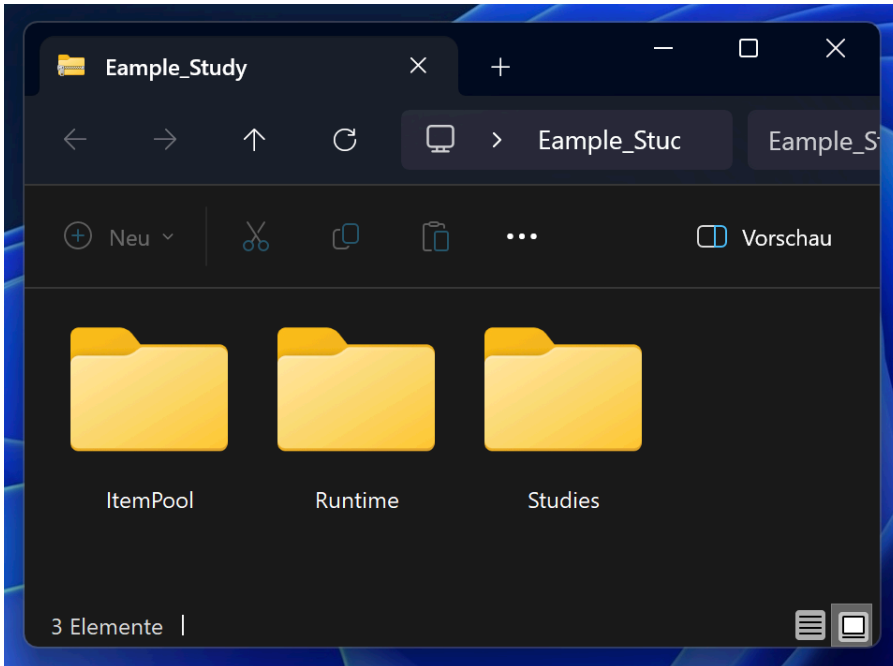
**BEENDEN**

Durch Klicken der Schaltfläche *Studie Bereitstellen* kann das ZIP-Archiv direkt geöffnet werden. Es wird dann automatisch in den *IRTlib Player* eingefügt und kann über die im Abschnitt *Login* der [Studie](#) konfigurierte Weise verwendet werden.

## 8.1.2. Manuelles Importieren

Wenn kein automatischer Import möglich oder gewollt ist, kann der Inhalt des ZIP-Archivs einer *Studie* auch manuell in das dafür vorgesehene Verzeichnis des *IRTlib Players* integriert werden.

Jedes ZIP-Archiv mit einer *Studienkonfiguration*, welches aus dem *IRTlib Editor* exportiert wurde, enthält drei Verzeichnisse. Die ZIP-Archive lassen {StudyName.zip} sich z.B. mit dem *Windows Explorer* öffnen:



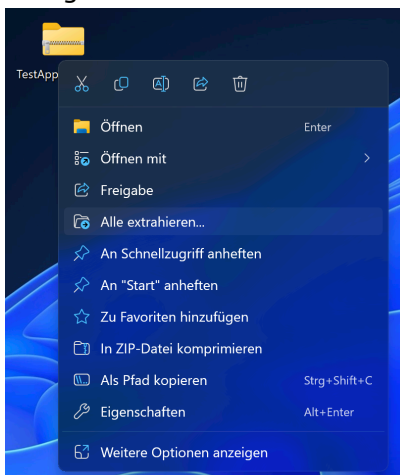
Um die *Studie* in einen *IRTlib Player* zu integrieren, kann der Inhalt dieser drei Verzeichnisse nun z.B. in das Programmverzeichnis eines Offline *IRTlib Players* integriert werden.

---

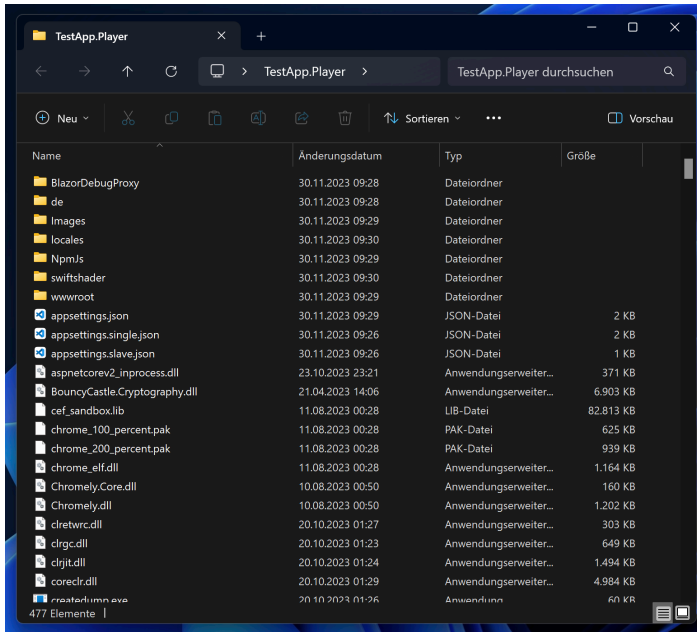
### Schritt Beschreibung

---

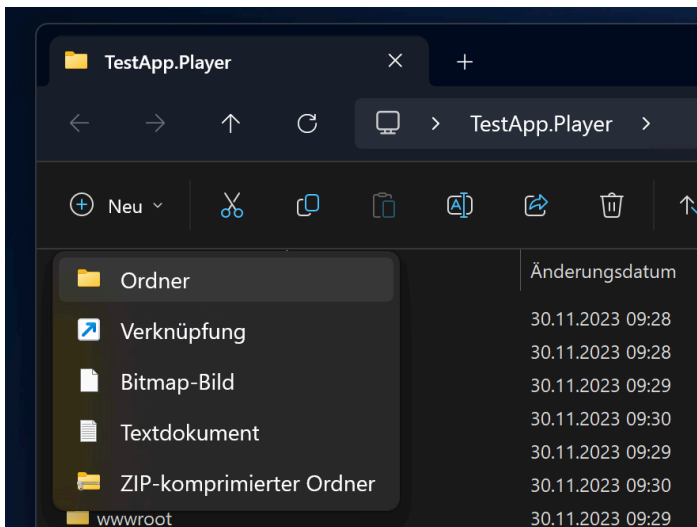
1. Player (TestApp.Player.zip) entpacken. Das kann bspw. über den *Windows Explorer* erfolgen:



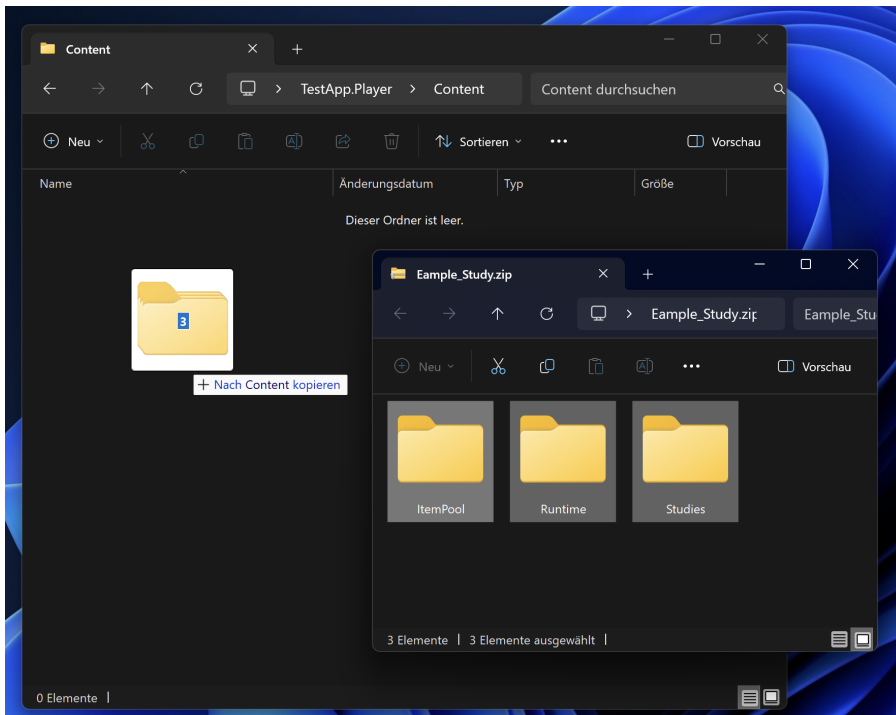
2. In das entpackte Verzeichnis navigieren:



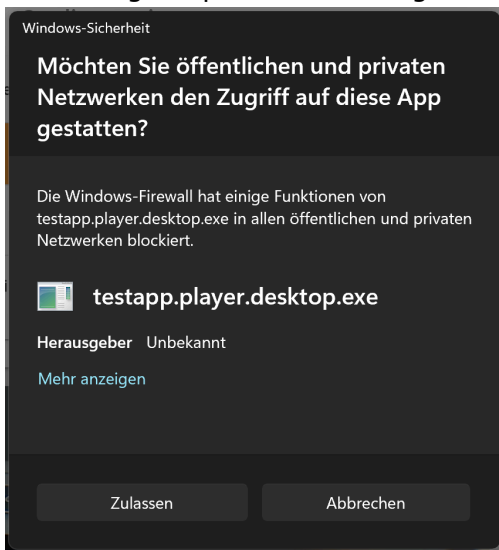
- Erstellen Sie einen neuen Ordner Content innerhalb des Player-Ordners (d.h. TestApp.Player/Content/). Wenn bereits eine Studie konfiguriert ist, dann existiert der Ordner Content bereits.



- Kopieren Sie die drei Ordner ItemPool, Runtime und Studies aus der heruntergeladenen Studie {StudyName.zip} in den Content-Ordner des Players.



5. Starten der Datei `TestApp.Player.Desktop.exe`
6. Wenn nötig akzeptieren Sie die folgende Warnung:



Analog zu dem hier beschriebenen Vorgehen erfolgt auch der Import von *Studien* in einen Online *IRTlib Player*. Hierfür ist für die Vorbereitung Zugriff auf das im `docker-compose.yml`-File definierte *Volume /app/Content* notwendig.

## 8.2. Auslieferungen Konfigurieren

Die mit einem *IRTlib Editor* erstellten Studienkonfigurationen sind mit unterschiedlichen Varianten des *IRTlib Players* verwendbar.

Im Moment werden drei Varianten bereitgestellt:

- Desktop-Version (Windows)
- Lokale Server-Version (Windows)

- Online Version (Docker)

### 8.2.1. Desktop-Version (Windows)

Basale Konfigurationen des *IRTlib Player* (Darstellung in einem Fenstermodus vs. Vollbildmodus) sind Teil der Studienkonfiguration. Nur Studien mit gleichen Einstellungen bzgl. der Darstellung können in einem Offline *IRTlib Player* gleichzeitig verwendet werden. Mehrere Kopien eines Offline *IRTlib Players* auf einem Computer sind möglich.

Um den (Offline) *IRTlib Player* auf einem Computer zu starten, muss die ausführbare Datei `TestApp.Player.Desktop.exe` gestartet werden.

**Datenspeicherung:** Die während einer Datenerhebung mit dem Offline *IRTlib Player* erhobenen Daten werden in einem Verzeichnis lokal gespeichert. Das Verzeichnis und der Dateiname für die Rohdatenspeicherung sind in der *Studie* konfiguriert und können mithilfe von Startup-Parametern angepasst werden. Als Dateiname der *Rohdatenarchive* wird der Benutzername bzw. die erstellte UUID verwendet. Sollte in einem (Offline) *IRTlib Player* der Benutzername mehrfach verwendet werden, d.h. wenn beim Beenden des *IRTlib Players* bereits ein *Rohdatenarchiv* mit diesem Dateinamen existiert, wird dieses nicht überschrieben, sondern es wird ein Suffix angehängt (z.B. `PersonIdentifizier_1.zip`).

**Startup-Parameter:** Für die Integration des Offline *IRTlib Players* in programmierte Abläufe sind möglich. Anmeldedaten (Benutzername, Benutzername + Passwort, Token), die in einer *Studie* konfiguriert sind, können als so-genannte *Startup-Parameter* übergeben werden. Diese Parameter werden dann an den Aufruf von `TestApp.Player.Desktop.exe` angehängt.

Beispiel:

```
TestApp.Player.Desktop.exe /RawDataFolder="..\myDataFolder"
```

Gültige Startup-Parameter sind:

- `/AutoLoginCreateWithTest="{StudienName}"`: Fordert die Administration der Studie mit der Bezeichnung `{StudienName}` an.
- `/AutoLoginUserName="{PersonIdentifizier}"`: Übergibt die Anmeldeinformation `{PersonIdentifizier}` als Benutzername.
- `/MyBlocklyVariable="123"`: Übergibt den Wert 123 für die *Blockly-Variable* `MyBlocklyVariable`.
- `/MonitoringFile="..\last-run.json"`: Pfad und Dateiname der so-genannten *Monitoring-Datei*.
- `/RawDataFolder="..\myDataFolder"`: Pfad zu dem Verzeichnis, in welchem die *Rohdatenarchive* gespeichert werden.

Mehrere Startup-Parameter nacheinander sind möglich (getrennt durch Leerzeichen).

**Prüfung von Voraussetzungen:** Der *IRTlib Player* sollte auf Windows-Computern (aktuell unterstützt ab *Windows 10*) ohne weitere Installation oder Laufzeitvoraussetzungen ausgeführt werden können. Spezielle Konfigurationen, Virens Scanner usw. können die Ausführung aber unterbinden. Ein Tool zum Prüfen von Voraussetzungen kann bspw. aufbauend auf diesem Beispiel ([IRTlibReadiness](#)) erstellt werden.

#### ! Wichtiger Hinweis

Die bereitgestellte Windows-Version des *IRTlib Player* ermöglicht einen einfachen *Kiosk-Modus*, der nur für Computer mit nur einem Bildschirm (z.B. Notebooks) vorgesehen ist. Für eine Prüfungssichere Kiosk-Lösung kann die Offline-Version des *IRTlib Players* als *lokaler Server* mit weiterer Software (wie bspw. dem [Safe Exam Browser](#)) kombiniert werden.



## 8.2.2. Lokaler Server (Windows)

Die über das [Github](#)-Repository im Abschnitt *Releases* verfügbaren ZIP-Archive des Offline *IRTlib Players* enthalten parallel zu der Anwendung mit integriertem Browser (`TestApp.Player.Desktop.exe`) auch eine *lokale Server-Version*, welche über die ausführbare Datei `TestApp.Player.Server.exe` gestartet werden kann.

### ! Wichtiger Hinweis

Die Version `TestApp.Player.Server.exe` ist gedacht für einen Offline-Betrieb in Bring-in Netzwerken, z.B. wenn WLAN-Router und Server-Notebook in Schulen gebracht werden. Diese Version ist nicht für Online-Ehreibungen gedacht (wofür die Docker-Version bereitgestellt wird).

## 8.2.3. Online-Version (Docker)

**Integration:** Für die Durchführung einer Datenerhebung sollte das *Docker-Image* des *IRTlib Players* nur über https-gesicherte Verbindungen erreichbar sein. Das kann bspw. mithilfe eines als *Reverse Proxy* konfigurierten zusätzlichen *nginx* realisiert werden.

**Zugriff auf Editor:** Für unbefugte Personen, die nicht mit der Studienvorbereitung betraut sind, darf kein Zugriff auf den *IRTlib Editor* möglich sein. Über das [Github](#)-Repository werden *IRTlib Editor* und *IRTlib Player* bereitgestellt. Für eine operative Datenerhebung ist es nicht notwendig, den *IRTlib Editor* online zu betreiben, da die Studienvorbereitung auch mit der Offline-Version des *IRTlib Editors* erstellt werden kann. Wird der *IRTlib Editor* online gehostet, muss er vor dem Zugriff Unbefugter geschützt werden.

**Zugriff auf Verzeichnisse:** In den im `docker-compose.yml`-File definierten *Volumes* `/app/Content` (*IRTlib Player*) und `/app/data` (*IRTlib Editor*) sind die Iteminhalte hinterlegt, die für ein Assessment konfiguriert werden. Um den Schutz von Instrumenten sicherzustellen, darf kein Zugriff für Unbefugte auf diese *Volumes* möglich sein.

**Datenspeicherung:** Die während einer Datenerhebung mit dem Online *IRTlib Player* erhobenen Daten werden in dem *Volume* `app/result` gespeichert. Sie können von dort als Verzeichnisse (ein Verzeichnis je *Session*) oder als *Rohdatenarchive* über eine *API* abgerufen werden (wenn ein *API-key* definiert ist).

### ! Hinweis

Bei Verwendung der *Docker-Container* können Assessmentinhalte und Daten online zugreifbar sein. Assessmentinhalte sind nur über den in der Studienkonfiguration definierten *Loginmodus* geschützt. Personendaten und Assessmentinhalte könne zusätzlich zugreifbar sein, wenn ein *API-key* definiert ist.

**Konfiguration des Players:** Zentral für die technische Konfiguration des Players ist die Datei `appsettings.json`, welche im Verzeichnis `TestApp.Player` enthalten ist. In dieser Datei können drei verschiedene *API-Keys* (also Zugriffsschlüssel) hinterlegt werden, in dem sie in folgender JSON-Struktur hinterlegt werden, bevor der *Docker-Container* gestartet wird:

```
"API": {
  "ExternalExportKey": "",
  "DevelopmentKey": "",
  "LoginManagementKeys": []
}
```

Die *API-Keys* haben die folgenden Funktionen:

- `ExternalExportKey`: Über diesen *Schlüssel* erhält man Zugang zu den mit dem *IRTLib Player* erhobenen Daten. Der Zugriff auf die Daten kann bspw. über das R-Paket `LogFSM` erfolgen, wie im Abschnitt *Datenabruf* beschrieben.

#### **i** Routen zum Direkten Zugriff

Die Liste der bearbeiteten Session, d.h. die *Session-Identifizier* lassen sich mit einem *API-Key* für `ExternalExportKey` über folgenden Aufruf als JSON abrufen:

`https://{U}/{S}/api/session/?apiKey={K}`

- {U} ist die URL des *IRTLib Players*
- {S} ist der Bezeichner der *Studie*
- {K} ist die `ExternalExportKey` wie in der `appsettings.json` definiert

Mit einer bekanntem *Session-Identifizier* lassen sich dann die Rohdaten über folgende Abruf mit einem *API-Key* für `ExternalExportKey` beziehen:

`https://{U}/{S}/api/session/{ID}/result?apiKey={K}`

- {ID} ist der *Session-Identifizier* (z.B. der Benutzername, je nach Konfiguration des Logins)

- `DevelopmentKey`: Dieser *API-Key* ist für das Anpassen von Studienkonfigurationen in einem laufenden Player vorgesehen.

#### **!** Under Development

Diese Funktion ist gerade in Entwicklung.

- `LoginManagementKeys`: Diese Liste von *API-Keys* ist für das Anpassen Zugangsdaten (Accounts) in einem laufenden Player vorgesehen.

#### **!** Under Development

Diese Funktion ist gerade in Entwicklung.

**Monitoring:** (Eine Methode zur Überwachung von *Docker-Containern* ist in Entwicklung).

## 8.3. Auslieferungen Testen und Freigeben

Mit der Integration einer *Studienkonfiguration* in einen *IRTLib Player* ist die Vorbereitung noch nicht abgeschlossen. Bevor eine Datenerhebung mit der *IRTLib Software* begonnen werden kann, sollten folgende Tests durchlaufen werden:

**(Tests innerhalb der CBA ItemBuilder-Preview):** Bevor die Konfiguration einer *Studie* und eines *Erhebungsteils* mit *CBA ItemBuilder-Tasks* erfolgt, sollte diese im Hinblick auf Darstellung, Funktionalität und Scoring bereits in der *Preview* des *CBA ItemBuilder* getestet sein.

**Funktionale Tests:** Insbesondere wenn *CBA ItemBuilder-Tasks* mit der Auslieferungsplattform interagieren (wie bspw. *Login-Items*), sollten funktionale Tests (d.h. Tests von konkreten Funktionen) in der Auslieferungssoftware erfolgen. Das betrifft auch die Navigation zwischen Items und natürlich die in der Auslieferungsumgebung konfigurierte Ablaufsteuerung.

**Cross-Browser-Testung:** Werden Studien nicht mit dem Offline *IRTLib Player* durchgeführt (welcher seinen eigenen Browser mitbringt) und insbesondere wenn über sogenannte *ExternalPageFrames* (d.h. *iFrames*) neu oder spezifisch programmierte JavaScript/HTML5-Inhalte innerhalb der *CBA ItemBuilder-Tasks\** verwendet werde, sollte eine Testung in verschiedenen Browsern erfolgen.

**Performanz-Tests:** Wenn große Mediendateien (Videos, Audiodateien) in den *CBA ItemBuilder-Tasks* enthalten sind, kann es ratsam sein, die Durchführbarkeit des Assessments auch unter ungünstigen Netzwerkbedingungen (z.B. geringe Bandbreite, lange Latenzen, Verbindungsabbrüche usw.) zu testen.

**Last-Tests:** Wenn sehr viele Testteilnehmer parallel (online) getestet werden sollen, kann es ratsam sein, das Lastverhalten der Auslieferung (und bspw. die für den *Docker*-Container verfügbaren Ressourcen) vorab abzustimmen.

**Datenablageprüfung:** In jedem Fall sollte die Passung der Scoring-Definition innerhalb der *CBA ItemBuilder-Tasks* und der Konfiguration im *IRTlib Editor* geprüft und eine Datenablageprüfung gemacht werden. Damit ist gemeint, dass vor Erhebungsstart synthetische Klickmuster (also Antworten) eingegeben und mit den im Datensatz gespeicherten Antworten verglichen werden. Um bei der späteren Überprüfung Eingabefehler gut erkennen zu können, hat es sich bewährt, für die Datenablageprüfung Bildschirmvideos parallel aufzuzeichnen.

**Smoke-Test:** Die abschließende Form des Testens ist ein Durchlauftest in dem fertig konfigurierten Setting aus *Studie* in einem *IRTlib Player*. Die Erhebung sollte richtig dargestellt werden und ein lesbare *Rohdatenarchiv* sollte entstehen.

### 8.3.1. Vorgeschlagene Testpläne

#### Hinweis

Die Bereitstellung der freien Forschungssoftware *IRTlib Editor* und *IRTlib Player* erfolgt ohne Gewähr und es kann keine Haftung für fehlende Daten, Datenverlust oder kompromittierte Daten usw. übernommen werden.

Allgemeingültige Empfehlungen zu (unbedingt) notwendigen Tests sind schwer zu formulieren, die folgende Tabelle ist deshalb als unverbindliche Empfehlung zu verstehen, welche im konkrete Fall abzuwägen ist.

Test	Empfehlung	Bedingung
<i>CBA ItemBuilder-Preview</i>	Immer	(Fehler in Darstellung, Verhalten und Scoring, die sich bereits bei der Itemerstellung finden lassen, sollten vor der Erstellung einer Studienkonfiguration systematisch getestet und ausgeschlossen werden.)
Funktionale Tests	Bei Bedarf	Nur wenn <i>IRTlib Player</i> und <i>CBA ItemBuilder-Tasks</i> interagieren müssen und bezogen auf im <i>IRTlib Editor</i> definierte Funktionalität (z.B. antwortabhängige Verzweigungen).
Cross-Browser-Testung	Bei Bedarf	Nur wenn <i>IRTlib Player</i> online verwendet wird und die Browser nicht den Browsern entsprechen, welche bereits für die <i>CBA ItemBuilder-Preview</i> verwendet wurden.
Performanz-Tests	Bei Bedarf	Wenn große Multimedia-Teile enthalten sind oder wenn mit schlechter Netzwerkverbindung zu rechnen ist.
Last-Tests	Bei Bedarf	Nur wenn Online <i>IRTlib Player</i> mit vielen parallelen Tests administriert werden soll.
Datenablageprüfung	Immer	Prüfung aller Daten (inkl. Log-Daten, wenn diese für die Auswertung benötigt werden).
Smoke-Test	Immer	Smoke-Testung bei jeder Version, vor allem um versehentliche Konfigurationsfehler in <i>letzter Minute</i> auszuschließen.

### **8.3.2. Datenerhebung Durchführen**

Ist die *Studie* in einem *IRTlib Player* konfiguriert und getestet, kann die Datenerhebung erfolgen. Je länger die Feldzeit ist, desto wichtiger sind regelmäßige Backups der erhobenen Daten bzw. ein regelmäßiger Abzug der gesammelten Rohdatenarchive von der Erhebungshardware.

# 9. Datenerhebung: Datenaufbereitung / Data Collection: Data Post-Processing

## 9.1. Datenaufbereitung

Daten werden vom *IRTLib Player* in *Rohdatenarchiven* pro *Session* (d.h. pro Testdurchführung mit einer *Studie*) gespeichert. Die *Rohdatenarchive* sind ZIP-Archive, deren Dateinamen dem *Benutzernamen* oder dem *Universally Unique Identifier* (UUID) entsprechen. Abweichungen von diesem Schema sind möglich, wenn zum Zeitpunkt der Speicherung bereits ein *Rohdatenarchiv* mit diesem Dateinamen vorhanden war. In diesem Fall werden die Daten vom *IRTLib Player* nicht überschrieben, sondern es wird ein Suffix *\_1, \_2, ...* angehängt, bis der Dateiname verwendbar ist.

- **Offline IRTlib Player:** Wenn nicht anders konfiguriert wurde, werden die Ergebnisdaten im Verzeichnis `Temp/{Studien-Name}/Results` gespeichert. Die *Rohdatenarchive* werden erstellt, wenn eine *Session* beendet, d.h. die letzte definierte *CBA ItemBuilder-Task* mit `NEXT_TASK` verlassen wird. Erst wenn die *Rohdatenarchive* erstellt wurden, ist ein *Fortsetzen* der angefangenen *Session*, wie es beispielsweise bei einem Computerabsturz notwendig sein kann, nicht mehr möglich.

Analog verhält es sich auch, wenn die Offline-Version des *IRTLib Player* als lokaler Server eingesetzt wird. Die *Rohdatenarchive* werden nach der Testbearbeitung im Verzeichnis `Temp/{Studien-Name}/Results` gespeichert.

Die Sammlung von Daten aus den Offline *IRTLib Playern* entspricht dem Einsammeln der *Rohdatenarchive*, die auf den verschiedenen Geräten erhoben werden.

### Hinweis

Da die Offline *IRTLib Player* untereinander nicht in Verbindung stehen, können - je nach *Loginmodus* - identische Anmeldedaten in verschiedenen *IRTLib Playern* parallel erstellt werden. Nach einer Datenerhebung müssen die *Rohdatenarchive* deshalb mit Sorgfalt zusammengefügt und ggf. durch Unterordner getrennt werden.

- **Online IRTlib Player:** Der Online-Player sammelt die Daten, wenn nicht anders konfiguriert, in dem *Volume*, welches für die Ergebnisdaten konfiguriert ist (vgl. `/app/results` in [docker-compse.yml-File](#)). Jede *Session* wird dort in einem separaten Unterverzeichnis gespeichert und kann von Administratoren, die Zugriff auf das *Volume* haben (!), heruntergeladen werden.

Wenn ein *API-key* für den Datenzugriff definiert ist, kann der *Download* der Ergebnisdaten auch über das R-Paket [LogFSM](#) erfolgen.

### 9.1.1. Datenabruf mit LogFSM

Dazu kann mit folgendem Aufruf zunächst (einmalig) das R-Paket installiert werden:

```
source("http://logfsm.com/latest")
```

Danach ist der Download der *Rohdatenarchive* mit folgendem R-Skript möglich:

```
library(LogFSM)

if (!dir.exists(paste0(getwd(),"/in/")))
  dir.create(paste0(getwd(),"/in/"))

if (!dir.exists(paste0(getwd(),"/out/")))
  dir.create(paste0(getwd(),"/out/"))

SECRET_KEY <- "(your secret key)"
API_URL <- "(your API-URL)"

LogFSM::TransformToUniversalLogFormat(inputfolders = paste0(getwd(),"/in/"),
                                     inputformat = "irtlibv01a",
                                     zcsvoutput = paste0(getwd(),"/out/data_csv.zip"),
                                     stataoutput = paste0(getwd(),"/out/data_dta.zip"),
                                     spssoutput = paste0(getwd(),"/out/data_sav.zip"),
                                     key = SECRET_KEY,
                                     web = API_URL,
                                     outputtimestampformatstring="dd.MM.yyyy HH:mm:ss.fff")

results <- read.csv(unz(paste0(getwd(),"/out/data_csv.zip"), "Results.csv"),
                  sep=";", encoding = "UTF-8")
```

#### *Datenabruf und Konvertierung der Daten mit LogFSM*

Mit dem Aufruf der Funktion `TransformToUniversalLogFormat` aus dem Paket `LogFSM` werden die Daten heruntergeladen und in das spezifizierte Verzeichnis `infolders` abgelegt, wenn ein API-Key (`key`) und eine API-Url (`web`) übergeben werden.

#### **i** Hinweis zur `SECRET_KEY` und `API_URL`

Der Wert für `SECRET_KEY` muss dem Eintrag entsprechen, der bei der Konfiguration des *Docker*-Images in der `appsettings.json` als `ExternalExportKey` definiert wurde, siehe Abschnitt [Online-Version \(Docker\)](#).

Der Wert für die `API_URL` bildet sich nach folgendem Schema:  
`https://{U}/{S}/api/session/`

- `{U}` ist die URL des *IRTlib Players*
- `{S}` ist der Bezeichner der *Studie*

Die Funktion `TransformToUniversalLogFormat` aus dem Paket `LogFSM` (oder analog dem im Folgenden beschriebenen *Kommandozeilenwerkzeug*) können auch verwendet werden, um bereits lokal vorhandene Rohdatenarchive auszulesen.

### 9.1.2. Datenabruf über die Kommandozeile

Die für den Datenabruf und die Datenkonvertierung über `LogFSM` verwendete Anwendung `TransformToUniversalLogFormat` ist als Konsolen-Anwendung aus dem *Releases*-Abschnitt von <https://github.com/kroehne/LogFSM/> verfügbar.

Darüber lassen sich Datenabruf und Datenumwandlung auch ohne R ausführen.

### ! In Entwicklung

Eine zertifizierte Version von `TransformToUniversalLogFormat` für Apple ist im Moment in Entwicklung.

### 9.1.3. Ergebnisdaten

Wenn die Daten über LogFSM von einem Online *IRTlib Player* abgerufen oder Offline eingesammelt wurden, liegen sie am Ende in einem Verzeichnis vor. Je *Session* (d.h. je Person oder Person x Zeitpunkt) als *Rohdaten*-Archiv.

Die Funktion `TransformToUniversalLogFormat` in LogFSM oder über die Kommando-Zeile kann auch verwendet werden, um die *Rohdatenarchive* aus einem Verzeichnis auszulesen und die Ergebnisdaten zu extrahieren:

```
library(LogFSM)

if (!dir.exists(paste0(getwd(), "/out/")))
  dir.create(paste0(getwd(), "/out/"))

LogFSM::TransformToUniversalLogFormat(inputfolders = paste0(getwd(), "/in/"),
                                     inputformat = "irtlibv01a",
                                     zcsvoutput = paste0(getwd(), "/out/data_csv.zip"),
                                     stataoutput = paste0(getwd(), "/out/data_dta.zip"),
                                     spssoutput = paste0(getwd(), "/out/data_sav.zip"),
                                     outputtimestampformatstring="dd.MM.yyyy HH:mm:ss.fff")

results <- read.csv(unz(paste0(getwd(), "/out/data_csv.zip"), "Results.csv"),
                  sep=";", encoding = "UTF-8")
```

### 9.1.4. Log-Daten

Die Konvertierung der Daten mit `TransformToUniversalLogFormat` in LogFSM oder über die Kommando-Zeile wandelt die erhobenen Log-Daten, welche von den *CBA ItemBuilder-Tasks* bereitgestellt werden, in folgende Formate um:

- *Flat and Sparse Log-Data Table*: Eine große Tabelle (als CSV, Stata, SPSS) mit einer Zeile je Event. Da die eventspezifischen Attribute (d.h. die unterschiedlichen zusätzlichen Informationen, die von einem Event vorhanden sind) sich auf viele Spalten verteilen, die aber jeweils nur je *Event-Typ* gefüllt sind, ist diese Tabelle zwar *flach*, aber ggf. auch sehr *löchrig*.
- *Universal Log-Format*: Alternativ enthalten die von LogFSM bzw. dem Kommandozeilen-Werkzeug `TransformToUniversalLogFormat` erstellten ZIP-Archive auch einzelne Datensatz-Tabellen je *Event-Typ*. Die eventspezifischen Attribute in diesen Tabellen sind weniger *löchrig* (d.h. sie enthalten nur fehlende Werte für optionale Attribute) und können zu einer *Flat and Sparse Log-Data Table* kombiniert werden, wenn erforderlich.
- *XES (eXtensible Event Stream)*: Die Log-Daten können auch in dem standardisierten XML-Format (<https://xes-standard.org/>) umgewandelt werden.

### **i** Hinweis zu Zeitstempeln

Die mit der *IRTLib Software* erhobenen Zeitstempel sind im *UTC-Format (Coordinated Universal Time)*.

## 9.1.5. Dateien in den Rohdatenarchiven

Die *Rohdatenarchive* enthalten folgende Dateien:

- *Trace.json*: Log-Daten (*Traces*) wie von der *CBA ItemBuilder-Runtime* geliefert, zusammen mit dem Kontext aus dem *IRTLib Player*.

Die Datei enthält folgende Struktur, mit Komma getrennt. Die Datei ist kein gültiges JSON, bis nicht das letzte Komma entfernt und ein [ vor und ein ] nach dem Inhalt eingefügt wird.

Der Eintrag *Trace* enthält die Log-Daten (*Traces*) in Päckchen (wie von der *CBA ItemBuilder-Runtime* geliefert) quotiert (d.h. " wird als `\u0022` dargestellt). Die *TraceId* ist ein Zähler, welcher die übermittelten Päckchen durchzählt. *Timestamp* ist der Zeitstempel der Übermittlung. *SessionId* der Benutzername oder die *UUID (PersonIdentifizier)*. Der *Context* gibt über den Namen des *CBA ItemBuilder-Projekts*, *Task* und *Scope* eine Referenz zum *Assessmentinhalt (Element)*. Unter *Assemblies* sind die Informationen zum verwendeten *IRTLib Player* gespeichert und *StudyRevision* verweist auf die *Revision* einer (veröffentlichten) *Studie*.

```
{
  "Trace": "(TRACE-JSON)",
  "TraceId": 1,
  "Timestamp": "2023-12-04T20:53:06.297Z",
  "SessionId": "(SESSION-ID OR USERNAME)",
  "Context": {
    "Item": "(PROJECT NAME)",
    "Task": "(TASK NAME)",
    "Scope": "(SCOPE)",
    "Preview": ""
  },
  "Assemblies": [
    {
      "Name": "TestApp.Player.Desktop",
      "Version": "(APPLICATION VERSION)",
      "GitHash": "(APPLICATION BUILD HASH)"
    }
  ],
  "StudyRevision": "(STUDY REVISION)"
},
```

- *Snapshot.json*: Snapshot-Daten wie von der *CBA ItemBuilder-Runtime* geliefert, zusammen mit dem Kontext aus dem *IRTLib Player*.

Die Datei enthält folgende Struktur, mit Komma getrennt. Die Datei ist kein gültiges JSON, bis nicht das letzte Komma entfernt und ein [ vor und ein ] nach dem Inhalt eingefügt wird.

Der Eintrag *Snapshot* enthält die Snapshot-Informationen (wie von der *CBA ItemBuilder-Runtime* geliefert) quotiert (d.h. " wird als `\u0022` dargestellt). Der *ContextFlag* gibt an, wie der *CBA ItemBuilder-Task* verlassen wurde (*NextTask*, *PreviousTask* oder *Cancel*).



Timestamp ist der Zeitstempel der Übermittlung. *SessionId* der Benutzername oder die UUID (*PersonIdentifier*). Der *Context* gibt über Name des *CBA ItemBuilder*-Projekts, *Task* und *Scope* eine Referenz zum Assessmentinhalt (*Element*). Unter *Assemblies* sind die Informationen zum verwendeten *IRTlib Player* gespeichert und *StudyRevision* verweist auf die *Revision* einer (veröffentlichten) *Studie*.

```
{
  "Snapshot": "(SNAPSHOT-JSON)",
  "ContextFlag": "NextTask",
  "ContextScope": 0,
  "Timestamp": "2023-12-04T20:53:06.497Z",
  "SessionId": "(SESSION-ID OR USERNAME)",
  "Context": {
    "Item": "(PROJECT NAME)",
    "Task": "(TASK NAME)",
    "Scope": "(SCOPE)",
    "Preview": ""
  },
  "Assemblies": null,
  "StudyRevision": null
},
```

- *ItemScore.json*: Scoring-Information (wie von *CBA ItemBuilder-Runtime* geliefert).

Die Datei enthält folgende Struktur, mit Komma getrennt. Die Datei ist kein gültiges JSON, bis nicht das letzte Komma entfernt und ein [ vor und ein ] nach dem Inhalt eingefügt wird.

Der Eintrag *ItemScore* enthält den *ItemScore* (wie von der *CBA ItemBuilder-Runtime* geliefert) quotiert (d.h. " wird als `\u0022` dargestellt). Der *ContextFlag* gibt an, wie der *CBA ItemBuilder-Task* verlassen wurde (*NextTask*, *PreviousTask* oder *Cancel*). *Timestamp* ist der Zeitstempel der Übermittlung. *SessionId* der Benutzername oder die UUID (*PersonIdentifier*). Der *Context* gibt über Name des *CBA ItemBuilder*-Projekts, *Task* und *Scope* eine Referenz zum Assessmentinhalt (*Element*). Unter *Assemblies* sind die Informationen zum verwendeten *IRTlib Player* gespeichert und *StudyRevision* verweist auf die *Revision* einer (veröffentlichten) *Studie*.

```
{
  "ItemScore": "(SCORING-JSON)",
  "ContextFlag": "NextTask",
  "ContextScope": 0,
  "Timestamp": "2023-12-04T20:53:06.474Z",
  "SessionId": "(SESSION-ID OR USERNAME)",
  "Context": {
    "Item": "(PROJECT NAME)",
    "Task": "(TASK NAME)",
    "Scope": "(SCOPE)",
    "Preview": ""
  },
  "Assemblies": [
    {
      "Name": "TestApp.Player.Desktop",
      "Version": "(APPLICATION VERSION)",
      "GitHash": "(APPLICATION BUILD HASH)"
    }
  ],
},
```

```
"StudyRevision": "(STUDY REVISION)"  
},
```

- `Session.json`: Die Datei enthält Daten des *IRTlib Players*, welche die Ausführung der *Session* beschreiben.
- `Log.json`: Log-Events der *IRTlib Player* (enthält u.A. Log-Informationen zur Verarbeitung des *Blockly-Routings*).
- `browser.log`: Ausgabe der Konsole, die während der Bearbeitung der Aufgaben im Browser gesammelt wurden (unstrukturierter Text, für Entwickler).
- `server.log`: Log-Ausgaben des Servers des *IRTlib Players* (unstrukturierter Text, für Entwickler)
- `Keyboard.json`: Tastatureingaben und Zeitstempel.
- `Monitoring.json`: Kopie der Monitoring-Datei, welche erstellt wurde.

**Teil III.**

**Allgemein / General**

# 10. Einstellungen / Settings

Die Einstellungen des *IRTLib Players* werden als Teil der *Studienkonfiguration* festgelegt. Der *IRTLib Editor* hat eine kleine Anzahl von Einstellungen.

## 10.1. Übersicht

Die *IRTLib Software* wird kontinuierlich weiterentwickelt. Informationen über die laufende Version können im Abschnitt *Über das Programm* abgerufen werden.

 Eingebettete Programmhilfe

### 10.1.1. Einstellungen

In diesem Bereich können Einstellungen vorgenommen werden, welche die Arbeit mit dem *IRTLib Editor* und alle Studien betreffen.

#### 10.1.1.1. Runtimesverwaltung

Um mit dem IRTLib Editor Studien zu konfigurieren, welche CBA ItemBuilder Inhalte verwenden, ist je Version die passende Laufzeitumgebung (*Runtime*) notwendig. Aktuelle getestete Version der CBA ItemBuilder Runtime sind bereits im Editor hinterlegt, in diesem Bereich können jedoch auch Runtimes für andere Versionen des CBA ItemBuilders oder aktualisierte bzw. korrigierte Runtimes in den Editor importiert werden.

Runtimes, die im Editor verfügbar sind, werden beim Veröffentlichen von Studien automatisch als Teil der Studienkonfiguration integriert und stehen auf diesem Weg aus dem *IRTLib Player* zur Verfügung.

#### 10.1.1.2. Allgemeine Einstellungen

Ändern Sie in diesem Abschnitt die Sprache für den Editor. Die hier ausgewählte Einstellung hat keinen Einfluss auf die Sprache der Assessmentinhalte in den konfigurierten Studien.

### 10.1.2. Über das Programm

Finden Sie unter der Schaltfläche **Versionsinfo** eine Zusammenfassung der letzten Änderungen und Angaben zur aktuellen Programmversion.

Wenn eine *Preview*-Version verwendet wird (d.h. eine Version der *IRTLib Software* die Änderungen enthält die seit dem letzten *Release* gemacht wurden, dann können die Programmversionen über den Build-Hash identifiziert werden:

## 10.2. Laufzeitumgebungen

Die *IRTLib Software* kann mit CBA ItemBuilder Aufgaben verschiedener CBA ItemBuilder Versionen verwendet werden. Die erforderliche **Runtime** (d.h. die Verbindung zwischen den CBA ItemBuilder-

## Versionsinfo & Changelog

Version: 1.0.0 (b1cd92b0)

Abbildung 10.1.: Beispiel aus Dialog *Über das Programm* mit dem Build-Hash b1cd92b0

Aufgaben und der *IRTLib Software*) ist Teil der Studienkonfiguration, damit der *IRTLib Player* sicher weiß, wie er CBA ItemBuilder-Aufgaben einer bestimmten Version verwenden kann.

### Eingebettete Programmhilfe

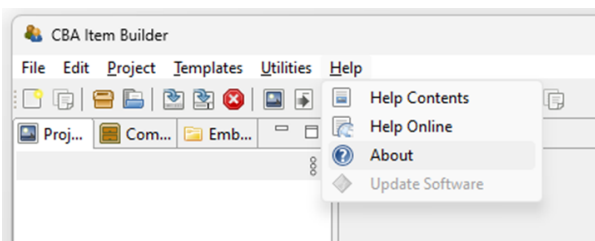
#### 10.2.1. Laufzeitumgebungen

Um mit dem *IRTLib Editor Studien* zu konfigurieren, welche *CBA ItemBuilder* Inhalte verwenden, ist je Version die passende Laufzeitumgebung (*Runtime*) notwendig. Aktuelle getestete Version der *CBA ItemBuilder Runtime* sind bereits im *IRTLib Editor* hinterlegt, in diesem Bereich können jedoch auch *Runtimes* für andere Versionen des *CBA ItemBuilders* oder aktualisierte bzw. korrigierte *Runtimes* in den *IRTLib Editor* importiert werden.

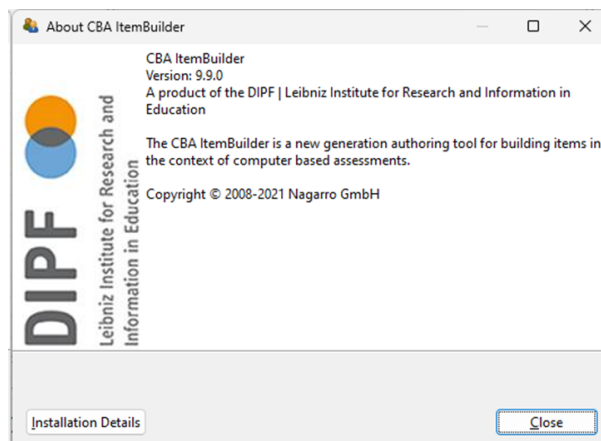
##### 10.2.1.1. CBA ItemBuilder Version Prüfen

Es ist wichtig zu wissen, welche Version des *CBA ItemBuilders* für die Erstellung der Items (d.h. der *CBA ItemBuilder*-Projektdateien) verwendet wurde. Im Zweifelsfall kann man diese Information z.B. im *About Dialog* des *CBA ItemBuilders* finden:

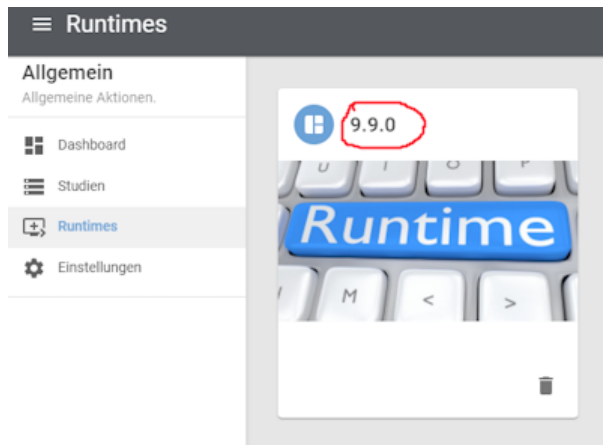
- Schritt 1: Öffnen Sie den "About"-Dialog über das "Help"-Menü



- Schritt 2: Suchen Sie die Versionsnummer im Dialog (hier 9.9.0)



Die Versionsnummer muss in den *Einstellungen* des *IRTlib-Editors* im Abschnitt *Laufzeiten* als eine der Karten aufgeführt sein:



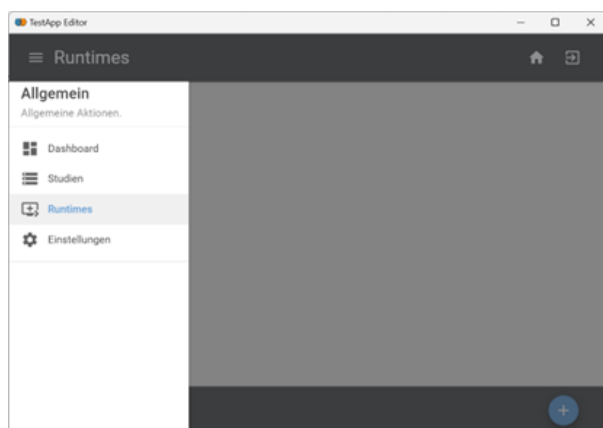
### 10.2.1.2. Laufzeitdateien importieren

Wenn die entsprechende Runtime nicht bereits im Editor enthalten ist, kann eine neue/zusätzliche *Runtime* importiert werden. Studienkonfigurationen, die mit dem *IRTlib-Editor* erstellt/bearbeitet werden, können mehrere *Runtimes* für verschiedene Versionen enthalten.

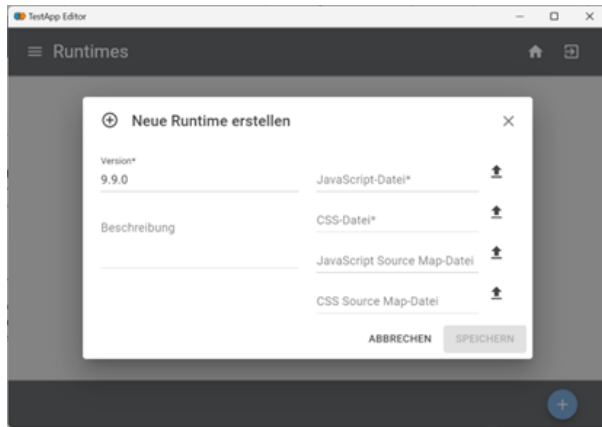
- Schritt 1: Um eine Runtime zu integrieren, werden eine JavaScript- und eine CSS-Datei benötigt. Diese Dateien können hier heruntergeladen werden:

<https://cba.itembuilder.de/appendix-tables.html#previous-versions>

- Schritt 2: Entpacken Sie die heruntergeladene *Runtime*, die verwendet werden soll.
- Schritt 3: Navigieren Sie zum Abschnitt *Runtimes*:



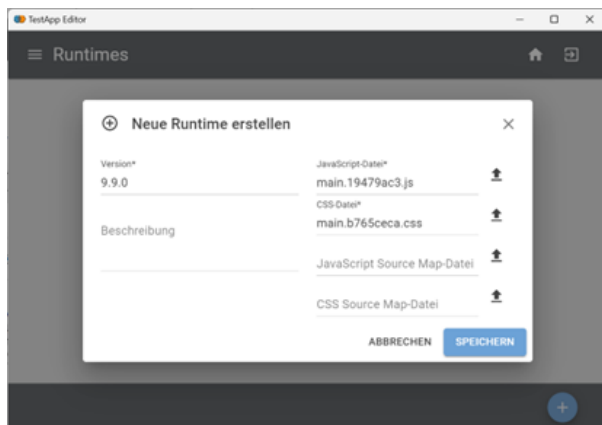
- Schritt 4: Drücken Sie den Button "+" (unten rechts)
- Schritt 5: Geben Sie die Versionsnummer mit drei Stellen ein (z.B. 9.9.0):



- Schritt 6: Wählen Sie die Datei `main.*.js` aus dem ZIP-Archiv aus, das die Laufzeitumgebung enthält. Beachten Sie, dass das \* dem Hash der Datei entspricht (d.h. der vollständige Dateiname sieht aus wie `main.19479ac3.js`)
- Schritt 7: Wählen Sie die Datei `main.*.css` aus dem ZIP-Archiv, das die Runtime enthält. Beachten Sie, dass der \* dem Hash der Datei entspricht (d.h. der vollständige Dateiname sieht aus wie `main.b765ceca.css`)

Hinweis: Das Feld *Description* und die beiden zusätzlichen *Map-Files* (für JavaScript Source und für CSS Source) sind optional.

- Schritt 8: Drücken Sie die Schaltfläche *Speichern*, um den Import der *Runtime* abzuschließen:



Nach dem Import werden die unterstützten CBA ItemBuilder-Versionen im Abschnitt *Runtime* aufgelistet. Um eine *Runtime* für eine bestimmte Version zu löschen, klicken Sie auf das *Papierkorb*-Symbol unten rechts auf der "Karte" und bestätigen Sie mit *Löschen*.

# 11. Github Repositorien / Github Repositories

## 11.1. IRTLib Software

Die *IRTlib Software* ist freie *Forschungssoftware* im Sinne von *Open Science*. Sie kann für nicht-kommerzielle Anwendungen verwendet werden.

Vorgeschlagene Zitierung:

Kroehne, U. (2023). IRTlib Documentation: Software for the administration and delivery of computer-based assessments [IRTlib Dokumentation: Software für die Verwaltung und Auslieferung computergestützter Assessments]. DIPF, Frankfurt am Main, Germany. <https://doi.org/10.5281/zenodo.10441352>

### Hinweis

Übersetzung: Wenn Sie uns bei der Übersetzung dieser Software helfen wollen, finden Sie [hier](#) weitere Informationen.

### 11.1.1. Download

- Aktuelle Versionen der IRTlib Software (Windows und Docker): [GitHub](#)
- Dokumentation: [GitHub](#)

## 11.2. CBA ItemBuilder

Die *IRTlib Software* erlaubt die Administration von Assessmentinhalten, die mit dem CBA ItemBuilder erstellt wurden.

### 11.2.1. Download

- Aktuelle Versionen des CBA ItemBuilder (Windows): <https://www.itembuilder.de/software>

### 11.2.2. Source Code

Source Code und Material zum CBA ItemBuilder sind aufgeteilt auf mehrere Repositorien:

- CBA ItemBuilder (Desktop Anwendung): [GitHub](#) (In Vorbereitung / noch privat)
- Laufzeitumgebung / Runtime: [GitHub](#) (In Vorbereitung / noch privat)
- Ausführungsumgebung / Execution Environment für Entwickler: [GitHub](#) (In Vorbereitung / noch privat)
- Technische Dokumentation: [GitHub](#) (In Vorbereitung / noch privat)
- Technische Beispielitems: [GitHub](#) (In Vorbereitung / noch privat)



### 11.2.3. Dokumentation

#### Online-Dokumentation

- HTML (interaktiv): <https://cba.itembuilder.de>
- PDF (statisch): [Open-Assessments-with-CBA-ItemBuilder.pdf](#)
- Quellen [GitHub](#) (In Vorbereitung / noch privat)

#### Vorgeschlagene Zitierung:

Kroehne, U. (2023). Open Computer-based Assessment with the CBA ItemBuilder. DIPF, Frankfurt am Main, Germany. <https://doi.org/10.5281/zenodo.10359757>

# 12. Über / About

## 12.1. Danksagung

An diesem Manual mitgeholfen haben:

- Maximilian Sattler
- Carla Burkart

**!** In Entwicklung

- Überarbeitung, sprachliche Korrektur und Übersetzung der Anleitung sind gerade in Arbeit...

## 12.2. Entwicklung

Die Entwicklung der *IRTlib Software* erfolgt bei [Software-Driven](#).