Protogrid

Protogrid Schnellstart-Anleitung (Erweitert)

Das Protogrid-Team freut sich, dass Sie Protogrid näher kennenlernen wollen. Diese Anleitung zeigt Ihnen wie sie eigen Logik implementieren können mit der Programmiersprache JavaScript.

Inhaltsverzeichnis

0. Voraussetzungen 1
1. Client ScriptLibraries 1
2. Erstellen der erforderlichen Komponenten 2
3. Felder identifizieren (Keys) 4
4. Feld Werte lesen 4
5. Feld Werte setzten 5
6. Fehler finden 6
7. Eigene Elemente einfügen 7
8. Direkt mit dem Back-End 7
9. Server Script Libraries 8
10. Agent Libraries 9
11. Das JSON-API 9
12. Fazit 10

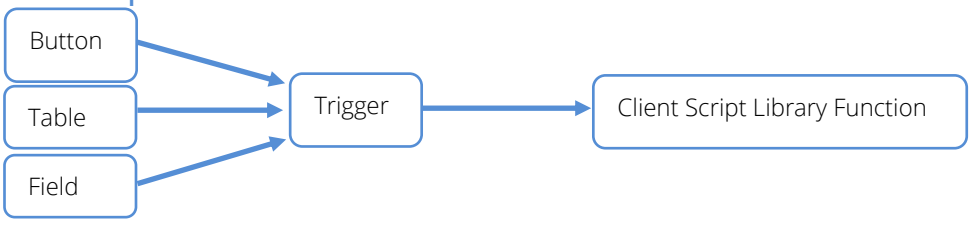
0. Voraussetzungen

Dieser Leitfaden setzt voraus, dass Sie bereits die Einführung absolviert haben und somit über eine eigene Protogrid-Umgebung mit der Anwendung „Dokumenten Ablage“ verfügen.

Darüber hinaus sind erste Erfahrungen mit JavaScript, HTML, CSS und Webtechnologien erforderlich, da wir hier nicht auf die Grundlagen eingehen werden.

Nützliche Links: https://www.javascripttutorial.net/, https://www.geeksforgeeks.org/html/html-css/

1. Client ScriptLibraries

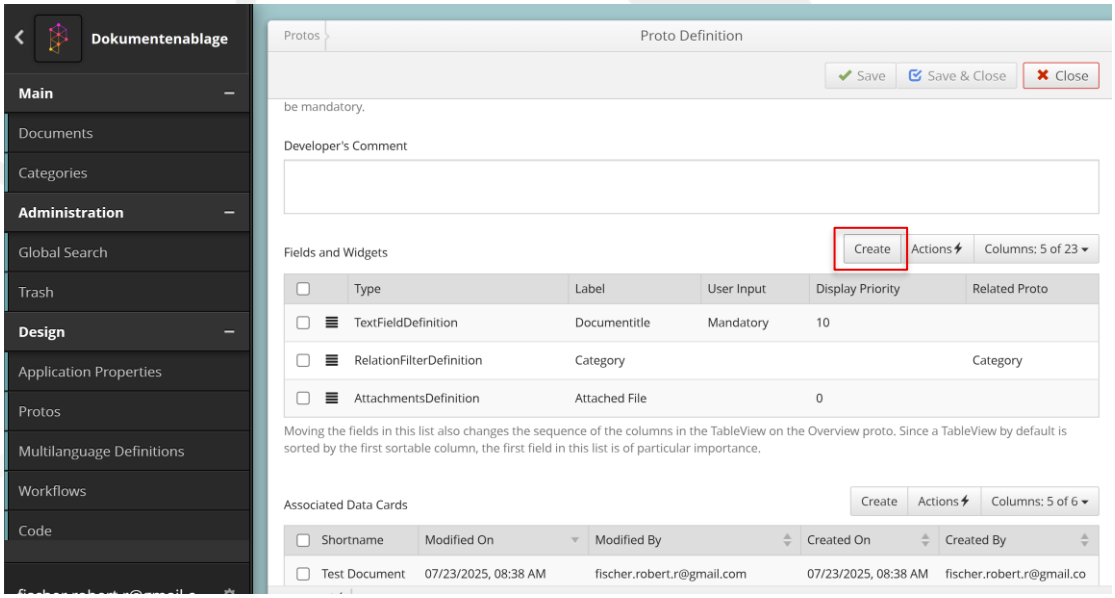


Die Client ScriptLibrary (CSL) ist die wichtigste Methode zur Implementierung benutzerdefinierter Funktionen in Protogrid. Um Ihre Funktion auszuführen, benötigen Sie einen Button, Tabelle oder Feld, welche einen Trigger aufruft, der dann eine Funktion in Ihrer CSL ausführen kann. Es ist wichtig zu

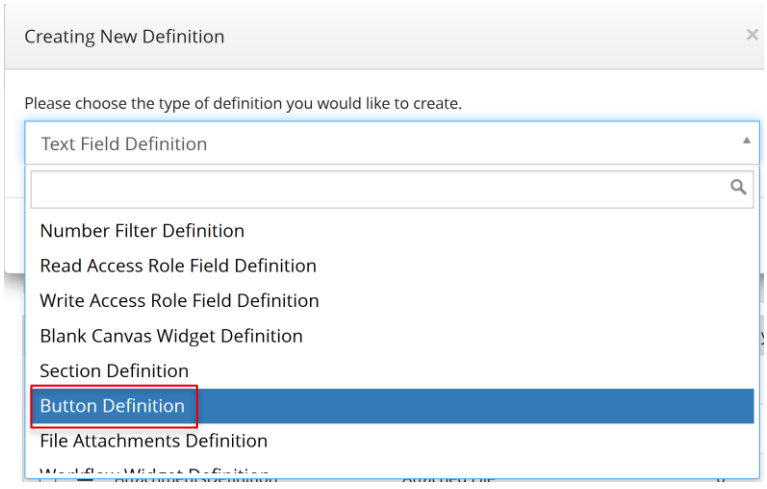
wissen, dass CSLs nur ausgeführt werden können, wenn der Benutzer auf einen Button klickt, und dass sie lokal auf dem Gerät des Benutzers ausgeführt werden, sodass der Benutzer auch den Code sehen kann. In späteren Kapiteln werden weitere Möglichkeiten vorgestellt. Die CSLs dienen dazu, die Benutzeroberfläche für Berechnungen, Komfortfunktionen oder andere Zwecke zu manipulieren.

2. Erstellen der erforderlichen Komponenten

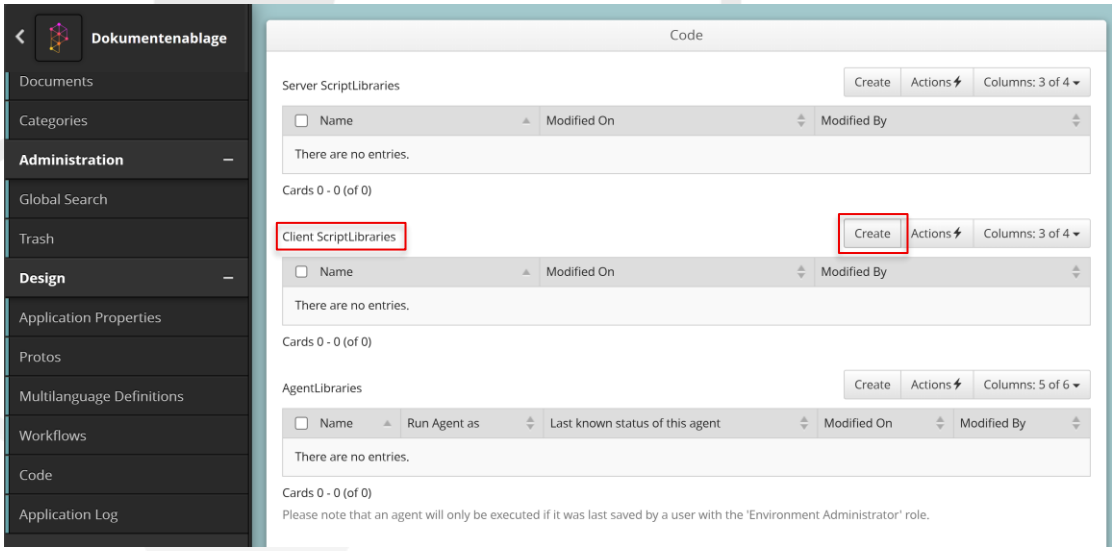
Schauen wir uns einmal an, wie man einen Button erstellt. Gehen Sie zu „Protos“ und wählen Sie das Daten-Proto „Dokument“ aus, das wir im letzten Tutorial definiert haben. Scrollen Sie dann nach unten zu „Felder und Widgets“ und klicken Sie auf „Erstellen“.



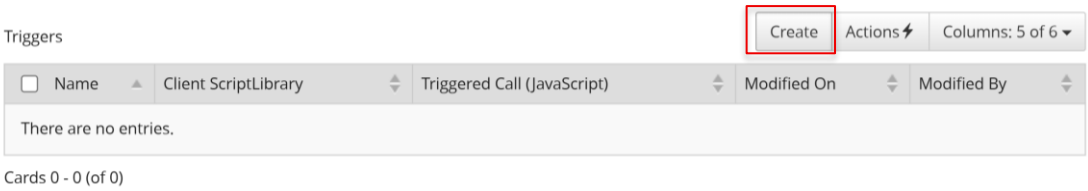
Wählen Sie dann „Button Definition“ und geben Sie dem Button einen Namen, der angezeigt wird, wenn Sie das Proto öffnen. Wir wählen „Check Title“ und für den Hilfetext „Überprüft, ob der Dokumenttitel lang genug ist“ für dieses Beispiel.



Um eine CSL zu erstellen, gehen Sie zu „Code“ und scrollen Sie nach unten zu „Client ScriptLibraries“ und erstellen Sie eine neue CSL. Geben Sie der CSL einen aussagekräftigen Namen, in diesem Beispiel wählen wir „Check Functions“. Speichern Sie anschliessend die erstellte CSL.



Um einen Trigger zu erstellen, scrollen Sie nach unten zu „Trigger“ und erstellen Sie einen neuen Trigger. Geben Sie dem Trigger einen aussagekräftigen Namen, in diesem Beispiel wählen wir „Title Check Trigger“. Die anderen Felder können Sie vorerst leer lassen.



Als Sie die CSL erstellt haben, ist Ihnen vielleicht das grosse Feld unten mit der Bezeichnung „Code“ aufgefallen. Hier speichern wir unsere benutzerdefinierte Logik, die in JavaScript geschrieben ist. Es enthält bereits viele nützliche Funktionen, die wiederverwendet werden können. Weiter unten finden Sie eine Liste, die alle verfügbaren Funktionen beschreibt. Eine wichtige Funktion, die Sie beachten sollten, ist „after_load()“, die jedes Mal ausgeführt wird, wenn das Proto geöffnet oder neu geladen wird.

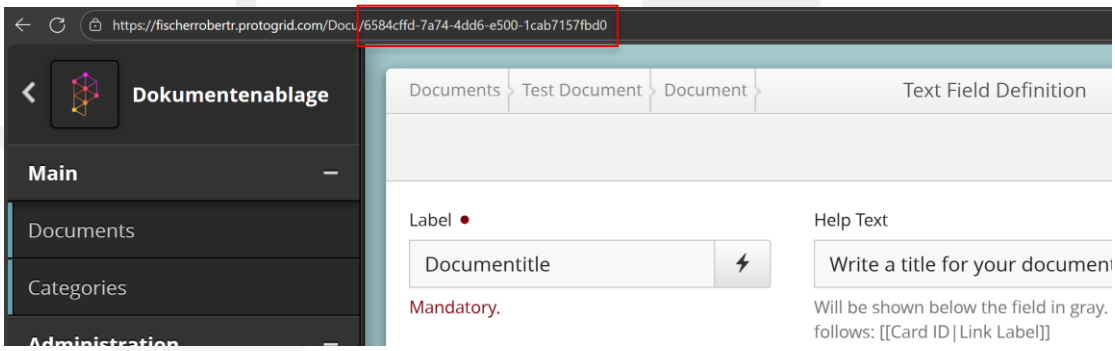
Nun erstellen wir im Feld „Code“ oben eine neue Funktion namens „check_documentTitle()“.

```
function check_documentTitle(){
    // Get document title
    var documentTitle;

    // check length and set output field.
    if(documentTitle.length > 5){

    }else{

    }
}
```

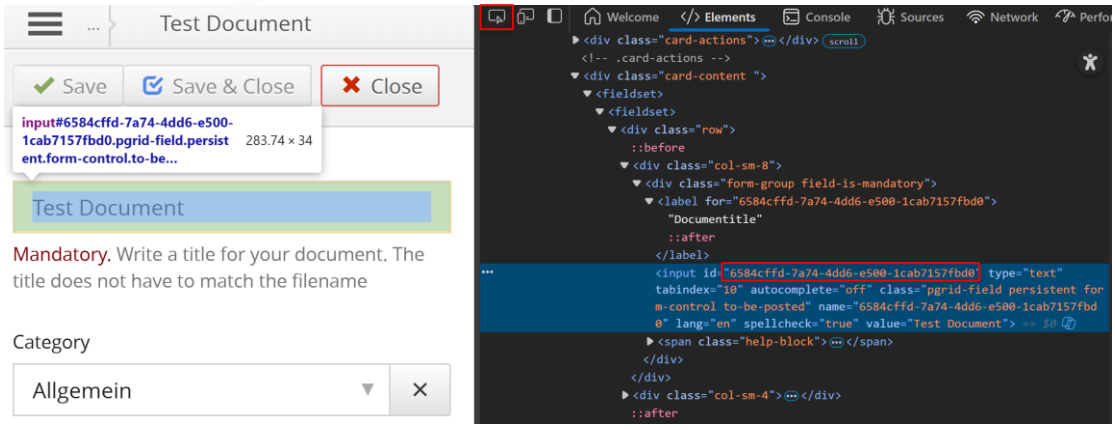
Nun könnte man sich fragen, wie wir auf den Dokumenttitel zugreifen, da die Funktion keine Eingabevariablen akzeptiert. Diese Frage werden wir im nächsten Kapitel beantworten.

3. Felder identifizieren (Keys)

Jedes Feld, das wir für ein Proto definieren, wird durch einen Key identifiziert. Der Key ist ein eindeutiges Wort/eine eindeutige Zeichenfolge, die für jedes Feld generiert wird. Um den Schlüssel für ein beliebiges Feld zu finden, ist es am bequemsten, wenn wir zum zugehörigen Daten-Proto gehen und das entsprechende Feld im Abschnitt „Felder und Widgets" öffnen. Wenn wir dann in die URL-Leiste des Browsers schauen, sehen wir den Key für dieses Feld. In unserem Fall wäre das „6584cffd-7a74-4dd6-e500-1cab7157fbd0".



Eine alternative Methode wäre, ein Dokumenten Card (nicht das Proto) zu öffnen und die Entwicklertools des Browsers zu verwenden, um den Schlüssel zu finden. In der Regel werden die Tools durch Drücken der Taste F12 geöffnet. Verwenden Sie im HTML-Code der Website das Auswahlwerkzeug, um das gewünschte Feld auszuwählen, und lesen Sie den Schlüssel unter der Bezeichnung „id" ab.



4. Feld Werte lesen

Nachdem wir nun wissen, wie man ein Feld identifiziert, können wir mit dem Auslesen von Werten beginnen. Wenn Sie sich den bereits generierten Code der CSL angesehen haben, ist Ihnen vielleicht die Funktion „csl_helpers.get_value()" aufgefallen. Wir übergeben einfach den Schlüssel, und die Funktion gibt den Wert des Feldes zurück.

```
    // Get document title
    var documentTitle = csl_helpers.get_value("6584cffd-7a74-4dd6-e500-1cab7157fbd0");
```

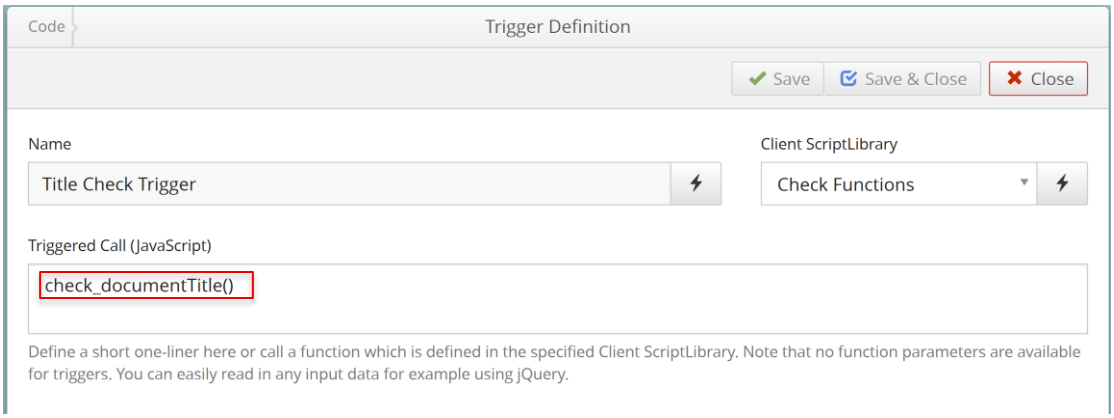
5. Feld Werte setzten

Um einen Wert in ein Feld zu schreiben, gibt es auch eine vordefinierte Hilfsfunktion namens „csl_helpers.set_value()“, die analog funktioniert. Zunächst müssen wir jedoch eine neue Textfelddefinition für das Datenproto des Dokuments erstellen, in der wir die Ausgabe unserer Funktion anzeigen können. In diesem Beispiel nennen wir sie „Validierung“ und lassen alle anderen Felder leer. Nachdem wir den Schlüssel für dieses neue Feld identifiziert haben, können wir unseren Code wie folgt vervollständigen:

```
function check_documentTitle(){
    // Get document title
    var documentTitle = csl_helpers.get_value("6584cffd-7a74-4dd6-e500-1cab7157fbd0");

    // check length and set output field.
    if(documentTitle.length > 5){
        csl_helpers.set_value("e6003db3-3f86-4287-d454-3a2adbb02873", "Document Title is
correct.");
    }else{
        csl_helpers.set_value("e6003db3-3f86-4287-d454-3a2adbb02873", "Document Title is too
short. Should contain more than 5 characters.");
    }
}
```

Beachten Sie, dass „csl_helpers.set_value()“ zwei Eingabeargumente benötigt, den Key und den zu setzenden Wert, und keinen Rückgabewert liefert.

Wenn Sie nun auf den erstellten Button klicken, werden Sie feststellen, dass nichts passiert. Das liegt daran, dass wir zunächst alle erstellten Elemente verknüpfen müssen. Gehen Sie zunächst erneut zur Registerkarte „Code“ und öffnen Sie den erstellten Trigger. Unter „Client ScriptLibrary“ wählen wir „Check Functions“ oder wie auch immer Sie Ihre CSL genannt haben. In das darunter liegende Textfeld mit dem Namen „Triggered Call (JavaScript)“ geben wir „check_documentTitle()“ ein. Dadurch wird die zuvor definierte Funktion ausgeführt, aber nichts anderes aus unserer CSL.



Gehen Sie dann zur Definition unseres Buttons im Daten Proto und setzen Sie den Trigger auf „Title Check trigger“. Es werden auch die zugehörige CSL und die aufgerufene Funktion in eckigen Klammern angezeigt.

Wenn nun alles korrekt durchgeführt wurde, können Sie den Button drücken, und das Validierungsfeld sollte je nach Länge des Dokumenttitels eine entsprechende Ausgabe anzeigen. Ist dies nicht der Fall, hilft Ihnen das nächste Kapitel weiter. Es sollte jedoch trotzdem gelesen werden.

6. Fehler finden

Wenn nichts passiert, sollten Sie zunächst Folgendes überprüfen:

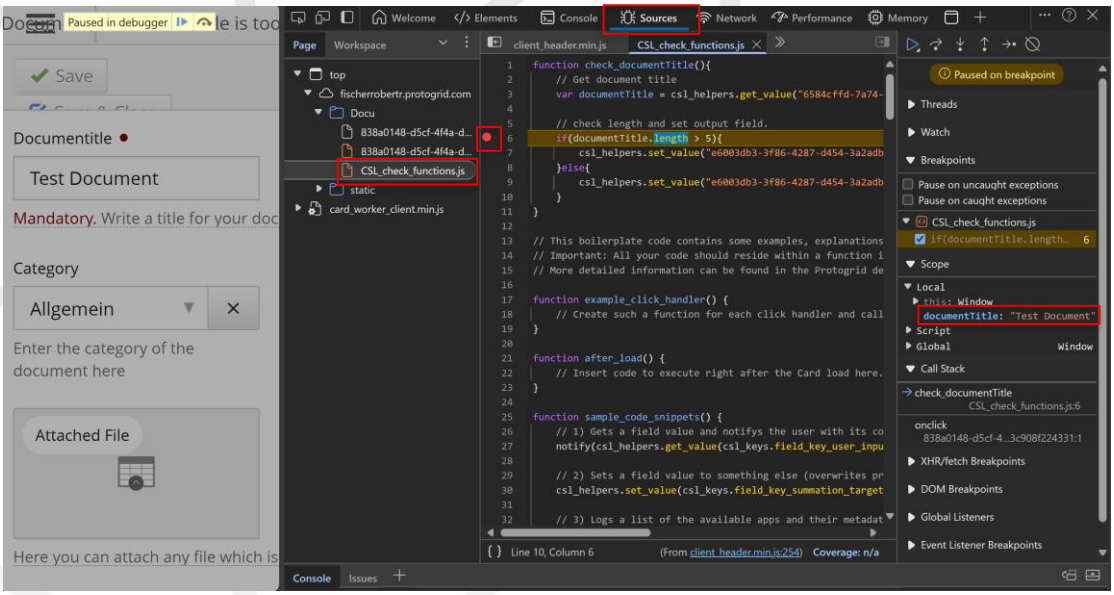
- Ist der Button mit dem richtigen Trigger verknüpft?
- Ist der Trigger mit der richtigen CSL verknüpft?
- Ruft der Trigger die richtige Funktion auf?
- Ist der Funktionsname korrekt und ohne Tippfehler geschrieben?
- Folgt auf den Funktionsnamen eine Klammer? Es sollte „check_documentTitle()" heissen und NICHT „check_documentTitle".

Protogrid zeigt keine Fehlermeldungen an, wenn diese nicht festgelegt sind.

Wenn Sie eine fehlerhafte Ausgabe haben oder glauben, dass das Problem auf Ihren JavaScript-Code zurückzuführen ist, können wir die Entwicklertools verwenden. Hier erklären wir den Vorgang für alle Browser, die auf Chromium basieren (Chrome, Edge usw.), aber der Vorgang ist wahrscheinlich für andere Browser ähnlich.

Öffnen Sie die Entwicklertools erneut mit F12 und navigieren Sie zur Registerkarte „Sources" und dann links zu >top >yourenvironment.protogrid.com >Docu. Hier sehen Sie einige JavaScript-Dateien. Diejenigen, die mit „CSL_" beginnen, sind Ihre verknüpften CSLs. Wenn sie hier nicht angezeigt werden, sind sie nicht korrekt verknüpft. Klicken Sie dann auf die CSL, die Sie untersuchen möchten, in unserem Fall wäre das „CSL_check_functions.js".

Dann sehen Sie erneut den Code, den Sie zuvor geschrieben haben. Wenn Sie mit der Maus kurz links neben den Zeilennummern verweilen, erscheinen rote Punkte. Sie können daraufklicken, um einen davon an einer bestimmten Zeile zu setzen. Diese Punkte werden „Breakpoints" genannt und unterbrechen die Ausführung Ihres Codes an dieser bestimmten Zeile. Ich habe meinen hier bei der if-Anweisung gesetzt. Das bedeutet, dass, wenn ich den Button in meinem Proto drücke, der Code ausgeführt wird und kurz vor der if-Anweisung stoppt. Sobald die Ausführung pausiert, können wir rechts unter >Scope >Local alle Variablen sehen, die wir definiert haben, sowie ihre Werte zu diesem Zeitpunkt. Die aktuellen Werte werden auch neben dem JavaScript-Code angezeigt.



Dies ist alles, was Sie benötigen, um Fehler in Ihrem Code zu finden. Die anderen Registerkarten in den Entwicklertools können Sie ignorieren, da sie für die Entwicklung benutzerdefinierter Protogrid-Funktionen nicht benötigt werden. Beachten Sie, dass alle Änderungen, die Sie hier am Code vornehmen, nicht gespeichert werden. Diese sollten stattdessen im Code Feld der entsprechenden CSL vorgenommen werden.

7. Eigene Elemente einfügen

In Protogrid ist es möglich, eigenen HTML/CSS-Code in jede Card einzufügen, für die eine CSL aktiviert ist. Dies kann mithilfe von jQuery erfolgen. Wir verwenden „$(document).ready()“, bevor wir die Einfügung ausführen, um sicherzustellen, dass die Website vollständig geladen ist, und verwenden dann die Funktion „append()“, um unseren HTML-Code einzufügen.

In diesem Beispiel habe ich ein neues leeres Widget erstellt, über dessen Key wir dann ein farbiges Element einfügen können. Das Element wird eingefügt, sobald wir auf die Schaltfläche „Validieren“ klicken, da ich den neuen Code in dieselbe Funktion eingefügt habe.

```
$(document).ready(function() {
    $('#8c93347e-2d6a-42d1-cb17-9a883711df9e').append('<button style="color: red">Click
    Me</button>');
});
```

8. Direkt mit dem Back-End

Wir können jQuery auch benutzen, um direkt mit dem Back-End von Protogrid zu interagieren mit dem JSON-API. In diesem Beispiel senden wir eine Email an eine vordefinierte Person wenn wir den «validate» Button drücken. Wir schreiben den folgenden code in dieselbe Funktion.

```
$.post({
    url: "https://example.protogrid.com/api/v2/apps/appc55b3773-5503-4c41-a9aa-
3a561c40fa04/mailsend",
    data: JSON.stringify({
        "to": ["User 1 <user1@example.com>", "User 2 <user2@example.ch>"],
        "subject": "Reminder",
        "body_text_plain": "Reminder: Document has been validated",
```

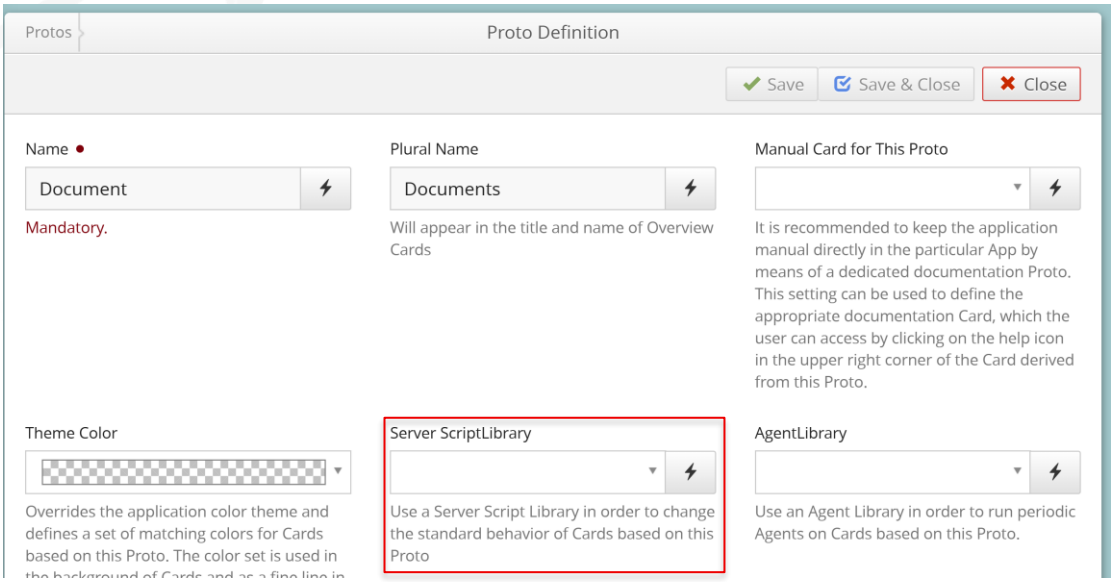
```
    }),
    contentType: 'application/json; charset=utf-8'
}).done(function (response) {
    alert("Data: " + JSON.stringify(response));
});
```

Wichtig zu wissen ist, dass der Email Relay für das Environment gesetzt wird auf den Email Server der eigenen Organisation. Detaillierte Informationen finden Sie hier.

9. Server Script Libraries

Server Script Libraries werden, wie der Name schon sagt, auf dem Server statt auf dem Client ausgeführt. Sie sollten nur für Funktionen verwendet werden, die ein erhöhtes Mass an Sicherheit erfordern, da der Code für den Benutzer nicht sichtbar ist.

Um eine Server Script Library zu erstellen, gehen Sie zur Registerkarte „Code", klicken Sie auf „Erstellen" und geben Sie ihr einen Namen. Auch hier erhalten Sie wieder vorgenerierten Code. Die beiden wichtigsten Funktionen, auf die Sie achten sollten, sind „on_render()" und „on_save()", die beide ausgeführt werden, wenn das Proto auf dem Server gerendert oder gespeichert wird. Ihre gesamte Logik muss im Hauptteil dieser Methoden geschrieben werden. Die neu definierte Bibliothek wird nur auf Protos ausgeführt, wo sie innerhalb des Daten-Protos verknüpft wurde, in dem Sie bereits alle Ihre Felder und Widgets definiert haben.



Die Funktion „get_shortname()" gibt den Kurznamen einer verknüpften Card zurück. Kurznamen werden als Darstellung der Card in Relationen- und Tag-Feldern verwendet und standardmässig von Protogrid automatisch generiert, sofern sie nicht im Hauptteil dieser Funktion überschrieben werden.

Dies ist nützlich in Verbindung mit der Funktion „get_sorting_string()", die eine Zeichenfolge zurückgibt, die zum Sortieren und Ordnen von Cards in einem Relationenfeld verwendet wird. Denken Sie an das Dropdown-Menü, das wir im Primer erstellt haben.

Die Funktion „get_title()" wird von Protogrid verwendet, um den Titel festzulegen, der oben auf der Card angezeigt wird. Sie können hier auch Ihre eigenen Regeln implementieren.

10. Agent Libraries

Agent Libraries sind die einzige Möglichkeit, Code ohne vorherige Benutzerinteraktion auszuführen. Sie werden alle 60 Minuten ausgeführt und dürfen nicht länger als 1 Minute laufen. Das Protogrid-Team kann dies auf Anfrage erhöhen.

Um dies auszuführen, müssen Sie eine Agent Library erstellen, ihr einen Namen geben, ein Konto ohne Administratorrechte verknüpfen und die Library mit dem Proto verknüpfen, das sie ausführen soll. Es ist wichtig, ein Konto mit minimalen Zugriffsrechten zu verknüpfen, da die Library von diesem Konto aus ausgeführt wird.

Die Agent Library wird auf jedem verknüpften Proto mit einem DateTime-Feld ausgeführt, das in der Vergangenheit liegt und noch nicht ausgeführt wurde. Sie führt alles aus, was in „on_schedule()" definiert ist.

Der Hauptanwendungsfall für Agent-Bibliotheken besteht darin, alle Protos mit zeitkritischen Informationen zu überprüfen und Massnahmen zu ergreifen, wenn sie abgelaufen sind.

11. Das JSON-API

Die JSON-API ermöglicht es uns, über HTTP-Anfragen auf sichere Weise direkt mit Protogrid-Daten zu interagieren. Hier zeigen wir Ihnen, wie dies in Python funktioniert, aber es ist auch mit jeder anderen Programmiersprache möglich, die HTTP-Anfragen unterstützt. Um zu beginnen, installieren Sie die Requests-Bibliothek, indem Sie folgenden Befehl in der Befehlszeile ausführen:

```
pip install requests
```

Dann schreiben wir in einer neuen Python-Datei den folgenden Code.

```python
import requests
url = "https://yourenvironmentname.protogrid.com/api/v2/authenticate"
headers = dict(username="testuser@example.com", password="test_password")
req = requests.post(url, headers=headers)
response = req.text
cookie = req.cookies['session']

print(response)
```

Wenn Sie die URL lesen, sehen wir, dass wir unsere Anfrage an eine Sub Page richten. In diesem Fall rufen wir den Authentifizierungsdienst auf. Im Header geben wir auch den Benutzernamen und das Passwort an. Der Server antwortet dann, wenn die Authentifizierung erfolgreich war. Er setzt auch ein Sitzungscookie, das es uns ermöglicht, Anfragen an andere Unterseiten zu stellen, ohne uns erneut authentifizieren zu müssen, wenn wir das Sitzungscookie angeben. Daher speichern wir es als Variable.

Wenn die Anmeldung erfolgreich war, sollte in der Ausgabe etwa Folgendes zu sehen sein:

```
{"errors":[],"protogrid_environment_version":"2.16.7e","result":"Login successful!"}
```

Jetzt, da wir das Sitzungscookie haben, können wir einen Aufruf an einen bestimmten Endpunkt senden. In unserem Beispiel verwenden wir „api/v2/apps“, wodurch alle Apps und deren Informationen zurückgegeben werden, die Sie auf der Startseite Ihrer Protogrid-Umgebung sehen können.

```
# The cookie variable was set above in the authentication example.
url = "https:// yourenvironmentname.protogrid.com/api/v2/apps"
req = requests.get(url, cookies={"session":cookie})
response = req.text

print(response)
```

Sie können nicht nur Informationen lesen, sondern auch neue Cards erstellen. Dazu benötigen Sie die Anwendungs-ID. Diese finden Sie unten links unter >Diese Card >Eigenschaften anzeigen. Anschliessend können Sie den Key des Protos der Card, die Sie erstellen möchten, verwenden und alle Felder ausfüllen. Sie müssen nicht alle Felder ausfüllen, ausser den Pflichtfeldern.

```
# The cookie variable was set above in the authentication example.
url = "https:// yourenvironmentname.protogrid.com/api/v2/apps/app77344ff9-0633-4c6d-b2e4-
96c3f8968448/cards"
payload = {
  "proto_key": "afa1c744-0289-4666-bb02-9a03fd25fc01",
  "design_elements": [
    {
      "definition_key": "6584cffd-7a74-4dd6-e500-1cab7157fbd0",
      "value": "JSON_API_DOCUMENT"
    },
    {
      "definition_key": "e6003db3-3f86-4287-d454-3a2adbb02873",
      "value": "JSON_TEST"
    },
  ]
}
response = requests.post(url=url, cookies={"session":cookie}, json=payload)
```

Eine vollständige Liste aller verfügbaren Endpunkte und detailliertere Informationen zur JSON-API finden Sie im Wiki.

12. Fazit

Jetzt kennen Sie alle Möglichkeiten der automatisierten Interaktion mit Protogrid. Während dieses Prozesses haben Sie Code auf der Client- und Serverseite ausgeführt, direkt mit der JSON-API interagiert und zeitgesteuerte Aufgaben im Hintergrund ausgeführt. Ausführlichere Informationen zur gesamten Funktionalität von Protogrid finden Sie unter protogrid.wiki.

Möchten Sie etwas Komplexeres implementieren, wissen aber nicht genau, ob und wie dies möglich ist? Das Protogrid-Team steht Ihnen jederzeit zur Verfügung. Senden Sie eine E-Mail an protogrid-customer-support@ategra.ch oder rufen Sie uns einfach unter +41 44 392 21 20 an.