



【初級～中級】初心者歓迎！ kintoneプラグインを開発しよう

2015年11月14日
株式会社ジョイゾー
代表取締役社長
四宮 靖隆

ハッシュタグ : #devsumiF
セッションID : 18-F-2

<https://github.com/joyzo/18-F-2>

```
git clone https://github.com/joyzo/18-F-2.git
```

株式会社ジョイゾー
代表取締役社長
四宮 靖隆

サイボウズ公認kintoneエバンジェリスト
kintone Café 運営事務局メンバー

1976年3月生まれ（39歳）
東海大学文学部卒
3児の父



Mr.kintone

第1回 kintone hive で
Mr. kintone
の称号をいただきました

会社紹介

会社名

株式会社ジョイゾー

所在地

東京都江東区東陽 3 – 5 – 5

設立年

2010年12月20日 6期目

従業員数

8名 (kintoneエバンジェリスト2名)

主な事業

kintone導入支援、カスタマイズ開発
サイボウズ Office/ガルーン構築支援



kintone案件にフルコミット

日本初 定額制来店型システム開発





【運営事務局】

ラジカルブリッジ
ジョイゾー
R3インスティテュート

斎藤
四宮、山下
金春

サイボウズ青野社長からの応援メッセージ

「kintone Caféの輪が全国に広がっていくことを大変うれしく感じています。作り手も、利用者も幸せになる新しいインテグレーションを生み出し続けてください。サイボウズも応援しています！」

kintone Café 開催状況

**3ヶ国 19都道府県
74回開催！※2016/4/16までの予定含む
22回参加しています！**



2014/5/29 福岡 Vol.1
2014/7/11 福岡 Vol.2
2014/10/24 福岡 Vol.3
2015/4/17 福岡 Vol.4
2015/6/24 福岡 Vol.5
2015/8/29 福岡 Vol.6
2015/10/24 福岡 Vol.7
2015/12/19 福岡 Vol.8

2015/11/28 熊本 Vol.1
2016/1/30 熊本 Vol.2

2015/9/29 鹿児島 Vol.1
2015/11/26 鹿児島 Vol.2
2016/3/17 鹿児島 Vol.3

2014/9/28 Okinawa #1
2015/2/11 沖縄 Vol.2
2015/10/10 沖縄 Vol.3
2016/1/8 沖縄 Vol.4

Korea

2014/9/2 Seoul #1

2015/6/25 大分 Vol.1
2015/12/5 大分 Vol.2

2014/11/21 北九州 Vol.1

佐賀 福岡 大分
長崎 熊本 宮崎
鹿児島

沖縄

2014/9/24 松山 #1
2015/9/26 愛媛 Vol.1
2016/1/16 愛媛 Vol.2
2016/4/16 愛媛 Vol.3

山口 島根 鳥取
広島 岡山
愛媛 香川
高知 德島

2016/1/16 関西女子会 Vol.1

2013/12/7 札幌 Vol.1 (起源)
2014/5/16 札幌 Vol.2
2014/10/25 札幌 Vol.3
2015/3/14 札幌 Vol.4
2015/6/4 札幌 Vol.5
2015/10/17 札幌 Vol.6
2015/11/30 札幌 Vol.7

北海道

2014/7/5 弘前 #1

2015/1/30 新潟 Vol.1
2015/7/4 新潟 Vol.2
2016/2/6 新潟 Vol.3

青森

秋田 岩手
山形 宮城

福島

群馬 栃木
岐阜 長野

埼玉 茨城
東京 千葉

神奈川
横浜

2015/1/24 仙台 Vol.1
2015/3/7 仙台 Vol.2
2015/4/4 仙台 Vol.3

2015/10/5 Grand Canyon Vol.1

USA

2014/12/3
@chiba みずいろ会館※女性限定

2014/8/1 千葉 (第1回)
2014/9/3 千葉 (第2回)
2014/12/12 千葉 (第3回)

2015/1/29 京橋 Vol.1
2015/4/6 京橋 Vol.2

2014/7/23 東京 Vol.1
2015/2/20 東京 Vol.2
2015/7/24 東京 Vol.3
2015/9/4 東京 Vol.4
2016/1/22 東京 Vol.5

kintone Café Grand Canyon Vol.1



kintone プラグイン を開発しよう

kintoneプラグインとは



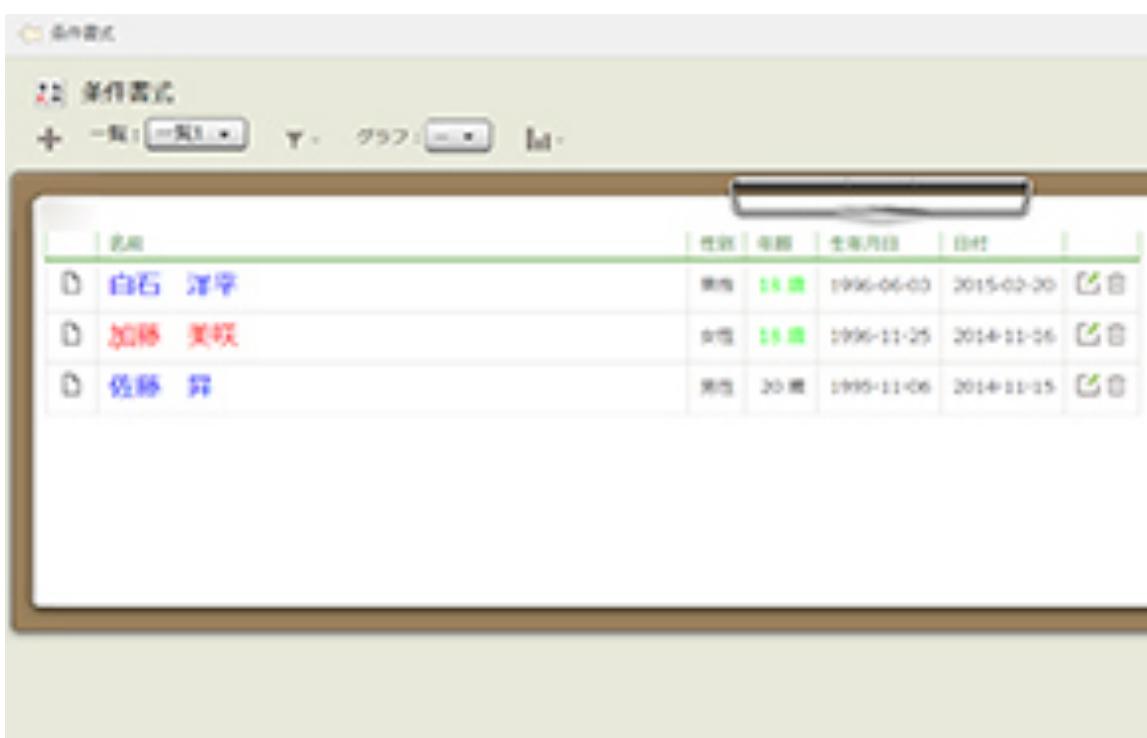
インストールするだけでkintoneを機能
拡張させることができる追加プログラム

公開中の主なプラグイン

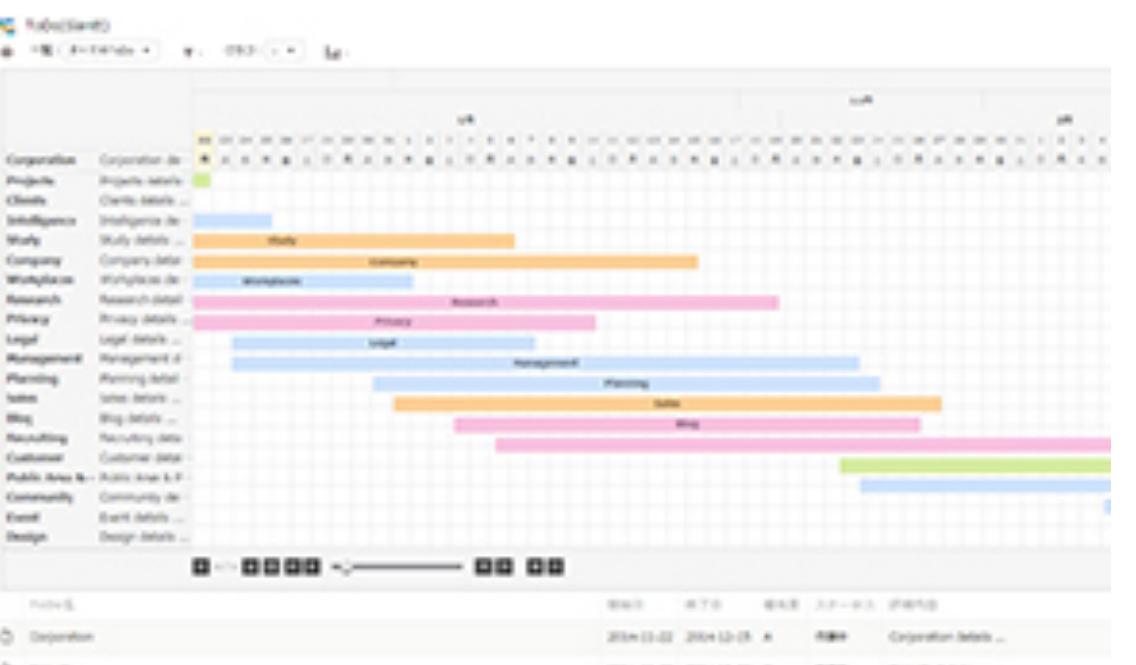


ガルーンスケジュール連携

メールワイス連携



条件付書式



ガントチャート

レコード番号	日付	作成者	操作内容
1	2014-06-13	Administrator	試用アプローチを確認版へ変更ユーザへのメール送信、一 午前中終了
2	2014-06-13	Administrator	データ導入作成、試用版: 5件 / 5件、電話版: 10件 / 10件
3	2014-06-11	Administrator	訪問件数: 5件 / 5件
4	2014-06-10	Administrator	訪問件数: 2件、電話件数: 5件 / 5件
5	2014-06-09	Administrator	お問い合わせへ添付ファイル添付。検索画面と匹敵品質、一 本目は販売中
6	2014-06-06	Administrator	訪問件数: 6件 / 10件
7	2014-06-05	Administrator	資料作成、メール件数: 3件 / 3件、電話件数: 1件 / 1件
8	2014-06-04	Administrator	資料作成、訪問件数: 5件 / 5件、電話件数: 20件 / 20件 本日は販売中
9	2014-06-03	Administrator	データ導入構築、試用版の資料準備、訪問件数: 4件 / 4件 日暮店の担当
10	2014-06-02	Administrator	メール作成、データ導入構築、訪問件数: 6件 / 6件、電 話件数: 1件 / 1件

いいね

JOYZO提供中のkintone プラグイン

●設定系プラグイン



ユーザー/組織別一覧初期設定

一番最初に表示させる一覧をユーザーや
所属組織によって変更させる

<https://www.joyzo.co.jp/plugin/>

●開発者向けプラグイン



Backlog連携

kintoneのレコード情報を
Backlogのチケットとして登録

textbox
 radio
 check
form

フィールド情報取得

kintoneのフィールド情報を
一覧で取得、CSV出力も可能

https://www.joyzo.co.jp/plugin/free_list/

① 管理画面から設定が可能

機能拡張の設定がアプリ管理画面から簡単に行なえます。
もちろん、**設定後の変更も自由**に行なえます。

② 設計変更に対して柔軟な対応

個別カスタマイズ開発よりもフィールドの追加/変更など設計変更の影響を**最小限**に抑えます。

③ 複数アプリで利用可能

同じ機能強化を複数のアプリで利用することができます。

利用可能な契約コース

ライトコース	スタンダードコース
月額 780 円 / 1ユーザー	月額 1,500 円 / 1ユーザー
年額でのお支払いも可能です 9,170円 / 1ユーザー	年額でのお支払いも可能です 17,640円 / 1ユーザー
API・JavaScriptカスタマイズ ※1,2	API・JavaScriptカスタマイズ ※1,2
アプリ数 ※3	1000個
スペース数	500個
ゲストスペース数	500個

詳しく見る

プログラミングなしで
簡単アプリ作成！

REST API、JavaScript/CSSで
カスタマイズして更に便利に！

カスタマイズを行いますのでスタンダードプランのみ対応しています

(出典 : kintone公式サイト [<https://kintone.cybozu.com/jp/price/>])

条件付き必須フィールド指定 プラグイン

今回の作るプラグイン

指定したフィールドに値を入力すると別に指定したフィールドを必須項目にする

活動タイトル 1

サイボウズ社訪問

活動内容 1

必須項目です

- ① **plugin-sdkの準備**
- ② **プラグイン用ファイルの作成**
- ③ **マニフェストファイルの作成**
- ④ **設定画面機能の実装**
- ⑤ **カスタマイズ機能の実装**
- ⑥ **パッケージング**
- ⑦ **kintoneへの適用**

① plugin-sdkの準備

② プラグイン用ファイルの作成

③ マニフェストファイルの作成

④ 設定画面機能の実装

⑤ カスタマイズ機能の実装

⑥ パッケージング

⑦ kintoneへの適用

plugin-sdkの準備

The screenshot shows the GitHub repository page for 'kintone / plugin-sdk'. The main content is the README.md file, which contains the following text:

公式SDKはこちら

<https://github.com/kintone/plugin-sdk>

Requirement

- A bash shell
- zip and openssl libraries are required on Windows using Cygwin.

How to Use

```
$ package.sh <plug-in dir> [<key file>]
```

Output Files

今回はこちら

<https://github.com/jozzo/18-F-2>

```
git clone https://github.com/jozzo/18-F-2.git
```

プラグイン開発の流れ

- ① plugin-sdkの準備
- ② プラグイン用ファイルの作成
- ③ マニフェストファイルの作成
- ④ 設定画面機能の実装
- ⑤ カスタマイズ機能の実装
- ⑥ パッケージング
- ⑦ kintoneへの適用

plugin-sdkのファイル構成

フォルダ/ファイル名	内 容
css	cssファイル格納フォルダ
html	設定画面用HTMLファイル格納フォルダ
image	アイコンファイル格納フォルダ
js	JavaScriptファイル格納フォルダ
manifest.json	パッケージング設定ファイル

- 各ファイルサイズには上限値がある
アイコン、CSS、JSファイル：512KB
HTMLファイル：64KB
- アイコンファイルはpng,jpg,gif,bmpのいずれか
- 設定画面とカスタマイズのJavaScriptファイルは別に作成
- JavaScriptファイルの文字コードはUTF-8

今回のファイル構成

フォルダ	ファイル名	内 容
css	51-us-default.css	kintoneのデザインと調和させるCSSファイル
html	config.html	設定画面用HTMLファイル
image	icon.png	アイコンファイル
js	config.js	設定画面用JSファイル
	desktop.js	PC用力スタマイズJSファイル

プラグイン開発の流れ

- ① plugin-sdkの準備
- ② プラグイン用ファイルの作成
- ③ **マニフェストファイルの作成**
- ④ 管理画面機能の実装
- ⑤ カスタマイズ機能の実装
- ⑥ パッケージング
- ⑦ kintoneへの適用

manifest.json

プラグインをパッケージングする際に必要な構成ファイル

記述を間違えるとパッケージングに失敗、もしくは
kintoneへの適用時にエラーになります。

マニフェストファイルの作成

主なパラメータ

パラメータ	必須	型	説明
manifest_version	○	Number	プラグインのマニフェストバージョン
version	○	Number	プラグインのバージョン
type	○	String	プラグインのタイプ ※”APP”と指定
name	○	Object.<String>	ロケールをキーとする各言語のプラグイン名
name.<locale>	○	String	指定されたロケールのプラグイン名
description	○	Object.<String>	ロケールをキーとする各言語の説明文
description.<locale>	○	String	指定されたロケールの説明文

マニフェストファイルの作成

パラメータ	必須	型	説明
icon	○	String	アイコンファイル
desktop	—	Object.<String>	ファイル種類をキーとするPC用カスタマイズファイル
desktop.js	—	Array.<String>	PC用JavaScriptファイル・URLの配列
desktop.css	—	Object.<String>	PC用cssファイル・URLの配列
mobile	—	String	ファイル種類をキーとするスマホ用カスタマイズファイル
mobile.js	—	Object.<String>	スマホ用JavaScriptファイル・URLの配列
mobile.css	—	String	スマホ用cssファイル・URLの配列

マニフェストファイルの作成

パラメータ	必須	型	説明
config	—	Object.<String>	ファイル種類をキーとする設定用カスタマイズファイル
config.js	—	Array.<String>	設定用JavaScriptファイル・URLの配列
config.css	—	Object.<String>	設定用CSSファイル・URLの配列
config.required_params	—	Array.<String>	設定画面で必須とするパラメータの配列

マニフェストファイルの作成

```
"manifest_version": 1,  
"version": 1, ← 整数のみ  
"type": "APP",  
"name": {  
    "ja": "条件付き必須入力チェックプラグイン",  
    "en": "Required input check check plug-in", ← 64文字以内  
    "zh": "Required input check check plug-in"  
},  
"description": {  
    "ja": "指定したフィールドに値が入っている際に別のフィールドに値が入っているかをチェックする  
    プラグインです。",  
    "en": "This plug-in will check that contains the value to a pair of field when that contains the value to the  
    specified field.",  
    "zh": "This plug-in will check that contains the value to a pair of field when that contains the value to the  
    specified field."  
},  
"200文字以内"
```

マニフェストファイルの作成

```
"icon": "image/check.png",
"desktop": {
    "js": [
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",
        "js/desktop.js"
    ]
},
"config": {
    "html": "html/config.html",
    "js": [
        "js/config.js"
    ],
    "css": [
        "css/51-us-default.css",
        "css/config.css"
    ],
    "required_params": ["field-1", "require-field-1"]
}
```

- ローカルパス、URL指定どちらも対応
- ローカルパスはmanifest.jsonが保存されているディレクトリからの相対パス

← 必須項目とするパラメータを指定

ファイルパス指定時のポイント

- ① HTMLファイルとアイコンは
ローカルパスのみ
- ② URL指定のほうがメリットがある
 - ・ファイルサイズの制限(512KB)をうけない
 - ・コード変更時の再パッケージングが不要

プラグイン開発の流れ

- ① plugin-sdkの準備
- ② プラグイン用ファイルの作成
- ③ マニフェストファイルの作成
- ④ **設定画面機能の実装**
- ⑤ カスタマイズ機能の実装
- ⑥ パッケージング
- ⑦ kintoneへの適用

設定画面機能の実装

```
"icon": "image/check.png",
"desktop": {
    "js": [
        "js/desktop.js"
    ]
},
"config": {
    "html": "html/config.html",
    "js": [
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",
        "js/config.js"
    ],
    "css": [
        "css/51-us-default.css",
        "css/config.css"
    ],
    "required_params": ["field-1", "require-field-1"]
}
```

- ① HTMLファイルの作成
- ② JS、CSSファイルの作成
- ③ 必須項目の指定

① HTMLファイルの作成

② JS、CSSファイルの作成

③ 必須項目の指定

完成イメージ

カテゴリー：
プラグイン

条件付き必須入力チェックプラグイン
Check!!

詳細
バージョン：1

必須フィールド組み合わせ-1
必須元フィールド*： 必須対象フィールド*：

必須フィールド組み合わせ-2
必須元フィールド： 必須対象フィールド：

保存する **キャンセル**

html/config.html

```
<div class="block">
  <div class="kintoneplugin-label">必須フィールド組み合わせ-1</div>
  <span class="kintoneplugin-row">必須元フィールド<span class="kintoneplugin-require">*</span> :
  </span>
  <div class="kintoneplugin-input-outer">
    <input id ="field-1" class="kintoneplugin-input-text" type="text"></input>
  </div>
  <span class="kintoneplugin-row">必須対象フィールド<span class="kintoneplugin-require">*</span> :
  </span>
  <div class="kintoneplugin-input-outer">
    <input id ="require-field-1" class="kintoneplugin-input-text" type="text"></input>
  </div>
</div>
```

HTMLファイルの作成

ドロップダウンリスト

表示例



記述例（HTML）

```
1 <div class="kintoneplugin-dropdown-outer">
2   <div class="kintoneplugin-dropdown">
3     <div class="kintoneplugin-dropdown-selected">
4       <span class="kintoneplugin-dropdown-selected-name">ドロップダウン</span>
5     </div>
6   </div>
7 </div>
8 <div class="kintoneplugin-dropdown-list">
9   <div class="kintoneplugin-dropdown-list-item kintoneplugin-dropdown-list-item-selected">
10    <span class="kintoneplugin-dropdown-list-item-name">オプションA</span>
11  </div>
12  <div class="kintoneplugin-dropdown-list-item">
13    <span class="kintoneplugin-dropdown-list-item-name">オプションB</span>
14  </div>
15  <div class="kintoneplugin-dropdown-list-item">
16    <span class="kintoneplugin-dropdown-list-item-name">オプションC</span>
17  </div>
18 </div>
```

51-us-default.css

新デザインと同じUIを実装する
ことができるCSSファイル

旧デザインをしばらく使いたい場合は、CSS
ファイルを「51-js-default.css」に
差し替えるだけでOK
※できれば新UIに移行しましょう！

HTMLやjQueryベースでの
記述サンプルと表示例が参考になるサイト

https://joyzo.github.io/kintone_css/

HTMLファイルの作成

html/config.html

```
<div class="block">
  <div class="kintoneplugin-label">必須フィールド組み合わせ-1</div>
  <span class="kintoneplugin-row">必須元フィールド<span class="kintoneplugin-require">*</span> :
  </span>
  <div class="kintoneplugin-input-outer">
    <input id ="field-1" class="kintoneplugin-input-text" type="text"></input>
  </div>
  <span class="kintoneplugin-row">必須対象フィールド<span class="kintoneplugin-require">*</span> :
  </span>
  <div class="kintoneplugin-input-outer">
    <input id ="require-field-1" class="kintoneplugin-input-text" type="text"></input>
  </div>
</div>
```

ボタンを作成

保存する

キャンセル

```
<div class="block">
  <button type="button" id="check-plugin-submit" class="kintoneplugin-button-dialog-ok">
    保存する </button>
  <button type="button" id="check-plugin-cancel" class="kintoneplugin-button-dialog-cancel">
    キャンセル </button>
</div>
```

マニフェストファイルに記述

```
"config": {  
    "html": "html/config.html",  
    "js": [  
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",  
        "js/config.js"  
    ],  
    "css": [  
        "css/51-us-default.css",  
        "css/config.css"  
    ],  
    "required_params": ["field-1", "require-field-1"]  
}
```

- ① HTMLファイルの作成
- ② JS、CSSファイルの作成
- ③ 必須項目の指定

JSファイル作成時の注意点

プラグイン設定画面から値の取り出しを行う
ためにプラグインIDの指定が必要

プラグインID = 各プラグインに一意に割り当てられた32桁のID

```
jQuery.noConflict();  
  
(function($,PLUGIN_ID) {  
    kintone.plugin.getConfig(PLUGIN_ID);  
})(jQuery,kintone.$PLUGIN_ID);
```

プラグインの設定内容を保存する関数

kintone.plugin.app.setConfig(config,callback)

引数	必須	型	説明
config	○	Object	キーと値を対にしたオブジェクトで、プラグインに保存する設定を指定します。値は文字列で指定します。 例： { "key1": "value1", "key2": "value2" }
callback	—	関数	設定の保存が完了したあとに実行する関数を指定します。callback 関数の指定を省略した場合、設定の保存が完了すると、プラグインの一覧画面に移動します。

プラグインの設定内容を取得する関数

kintone.plugin.app.getConfig(pluginId)

引数	必須	型	説明
pluginId	○	String	設定情報を取得するプラグインのIDを指定します。

js/config.jsの7行目と26行目を編集

```
(function($,PLUGIN_ID) {
  "use strict";

  $(document).ready(function(){
    var conf = kintone.plugin.app.getConfig(PLUGIN_ID); //プラグインの設定情報を取得
    ~省略~

    kintone.plugin.app.setConfig(config); //設定情報を保存
    ~省略~

  });
})(jQuery,kintone.$PLUGIN_ID);
```

```
var conf = kintone.plugin.app.getConfig(PLUGIN_ID); //プラグインの設定情報を取得  
  
//既に値が設定されている場合はフィールドに値を設定する  
if (conf){  
    $('#field-1').val(conf['field-1']);  
    $('#require-field-1').val(conf['require-field-1']);  
    $('#field-2').val(conf['field-2']);  
    $('#require-field-2').val(conf['require-field-2']);  
}
```

取得した設定情報を
テキストフィールドにセット

```
// 「保存する」ボタン押下時に入力情報を設定する
$('#check-plugin-submit').click(function() {
    var config = [];
    //元フィールドとチェック先フィールドのどちらかが入力されていない場合は対のフィールドは空にセット
    config['field-1'] = $('#field-1').val();
    config['require-field-1'] = $('#require-field-1').val();
    config['field-2'] = ($('#require-field-2').val()) ? $('#field-2').val() : "";
    config['require-field-2'] = ($('#field-2').val()) ? $('#require-field-2').val() : "";

    kintone.plugin.app.setConfig(config); //設定情報を保存
});
```

入力された値をプラグインの
設定情報として保存

マニフェストファイルに記述

```
"config": {  
    "html": "html/config.html",  
    "js": [  
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",  
        "js/config.js"  
    ],  
    "css": [  
        "css/51-us-default.css",  
        "css/config.css"  
    ],  
    "required_params": ["field-1", "require-field-1"]  
}
```

- ① HTMLファイルの作成
- ② JS、CSSファイルの作成
- ③ 必須項目の指定

マニフェストファイルに記述

```
"config": {  
    "html": "html/config.html",  
    "js": [  
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",  
        "js/config.js"  
    ],  
    "css": [  
        "css/51-us-de...  
        "css/config.c...  
    ],  
    "required_params": ["field-1", "require-field-1"]  
}
```

必須フィールド組み合わせ-1の
必須元フィールドと必須対象フィール
ドを必須扱いに

プラグイン開発の流れ

- ① plugin-sdkの準備
- ② プラグイン用ファイルの作成
- ③ マニフェストファイルの作成
- ④ 設定画面機能の実装
- ⑤ カスタマイズ機能の実装
- ⑥ パッケージング
- ⑦ kintoneへの適用

カスタマイズ機能の実装

```
"icon": "image/check.png",
"desktop": {
    "js": [
        "js/desktop.js"
    ]
},
"config": {
    "html": "html/config.html",
    "js": [
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",
        "js/config.js"
    ],
    "css": [
        "css/51-us-default.css",
        "css/config.css"
    ],
    "required_params": ["field-1", "require-field-1"]
}
```

カスタマイズ機能の実装

日付

2016-02-

活動タイトル 1

サイボウズ社訪問

活動内容 1

必須項目です

指定したフィールドに値を入力すると別
に指定したフィールドを必須項目にする

JSファイルの作成

js/desktop.jsの16,17行目を編集

```
// 指定されたフィールドに値が入っている場合に指定のフィールドの入力チェックを行う  
var config = kintone.plugin.app.getConfig(PLUGIN_ID); //設定を読み込む  
  
//設定値読み込み  
if (!config) return false;  
  
var record = event.record; //レコード情報  
//フィールド値を取得  
var field_1 = record[config['field-1']]['value'];  
var check_field_1 = record[config['require-field-1']]['value'];  
  
var field_2 = (config['field-2']) ? record[config['field-2']]['value'] : "";  
var check_field_2 = (config['require-field-2']) ? record[config['require-field-2']]['value'] : "";
```

設定画面で登録した値を取得

JSファイルの作成

23行目を編集

```
if(field_1 && !check_field_1){ //フィールドパターン1の必須チェック  
    record[config['require-field-1']].error = "必須項目です";  
}  
  
if(field_2 && !check_field_2){ //フィールドパターン2の必須チェック  
    record[config['require-field-2']].error = "必須項目です";  
}
```

フィールドにエラーを表示

```
(function (PLUGIN_ID) {  
    ~省略~  
})(kintone.$PLUGIN_ID);
```

プラグインIDの指定/重複対応

```
// 指定されたフィールドに値が入っている場合に指定のフィールドの入力チェックを行う
function checkRequire(event){

    var config = kintone.plugin.app.getConfig(PLUGIN_ID); //設定を読み込む
    if (!config) return false;
```

kintone.plugin.app.getConfig 関数
でプラグインID(**PLUGIN_ID**)を指定して
設定情報を取得

```
function checkRequire(event){  
    ~省略~  
}  
  
kintone.events.on(['app.record.create.submit',  
    'app.record.edit.submit',  
    'app.record.index.edit.submit'], checkRequire);
```

新規作成 / 編集の保存前、
一覧からのレコード編集の保存前
イベント時に実行

イベントハンドラーを登録する関数

kintone.events.on(event, handler(event))

パラメーター	指定する値	説明
event	文字列	イベントハンドラーをバインドする対象のイベントタイプまたは
	文字列の配列	イベントタイプの配列
handler(event)	Function(Object)	イベント発生時に実行されるハンドラー

レコードを保存する前のイベント

```
kintone.events.on('app.record.create.submit', handler);
```

レコード一覧表示後のイベント

```
kintone.events.on('app.record.index.show', handler);
```

複数のイベントをまとめて実行

```
kintone.events.on(['app.record.index.show', 'app.record.create.submit'], handler);
```

```
function checkRequire(event){  
    ~省略~  
}  
  
kintone.events.on(['app.record.create.submit',  
    'app.record.edit.submit',  
    'app.record.index.edit.submit'], checkRequire);
```

ハンドラーにはeventオブジェクトを渡せる

eventオブジェクトからkintoneの レコード情報が取得できる

パラメーター	指定する値	説明
appId	数値	アプリID
record	オブジェクト	ユーザー入力のデータを保持したレコードオブジェクト

```
var record = event.record; //レコード情報を取得
```

```
//フィールド値を取得
```

```
var field_1 = record[config['field-1']]['value'];
```

```
var check_field_1 = record[config['require-field-1']]['value'];
```

```
var field_2 = (config['field-2']) ? record[config['field-2']]['value'] : "";
```

```
var check_field_2 = (config['require-field-2']) ? record[config['require-field-2']]['value'] : "";
```

recordオブジェクトから
指定されたフィールドの値を取得

```
if(field_1 && !check_field_1){ //フィールドパターン1の必須チェック  
    record[config['require-field-1']].error = "必須項目です";  
}  
  
if(field_2 && !check_field_2){ //フィールドパターン2の必須チェック  
    record[config['require-field-2']].error = "必須項目です";  
}  
  
return event;
```

フィールドのerrorにメッセージを
代入してeventオブジェクトをreturn
するとエラー扱いとして保存がキャンセル

マニフェストファイルに記述

```
"desktop": {  
    "js": [  
        "js/desktop.js"  
    ]  
},
```

プラグイン開発の流れ

- ① plugin-sdkの準備
- ② プラグイン用ファイルの作成
- ③ マニフェストファイルの作成
- ④ 設定画面機能の実装
- ⑤ カスタマイズ機能の実装
- ⑥ **パッケージング**
- ⑦ kintoneへの適用

package.sh

kintoneに読み込める形式の
ZIPファイルを生成するシェルスクリプト

シェルなのでWindowsの場合はちょっと
面倒・・・・

コーディングもシェル実行もできる
Cloud9の利用がオススメ

新規作成用コマンド

package.sh <manifest.jsonが置かれているフォルダ>

**kintone-pluginディレクトリに移って
以下のコマンドを実行**

sh ./package.sh plugin/

PLUGIN_DIR must not contain dot files or directories.

/Documents/kintone-plugin/plugin/.DS_Store

**Macの場合、こんなメッセージが出た場合は
.DS_Storeファイルを手動で削除**

結果例

Plugin ID: godoojndmlcajgabgldkpbifjipioplgo

Plugin file: /Users/y-shinomiya/Documents/kintone-plugin/plugins/godoojndmlcajgabgldkpbifjipioplgo/plugin.zip

Private key file: /Users/y-shinomiya/Documents/kintone-plugin/keys/plugin.godoojndmlcajgabgldkpbifjipioplgo.ppk

項目	説明
Plugin ID	プラグインID
Plugin file	パッケージングされたプラグインファイル
Private key file	パッケージングの際に作成された秘密鍵

結果例

Plugin ID: godoojndmlcajgabgldkpbifjipioplgo

Plugin file: /Users/y-shinomiya/Documents/kintone-plugin/plugins/godoojndmlcajgabgldkpbifjipioplgo/plugin.zip

Private key file: /Users/y-shinomiya/Documents/kintone-plugin/keys/plugin.godoojndmlcajgabgldkpbifjipioplgo.ppk

Pluginを更新する際に必要なので
絶対に無くさないように！！

Plugin file

パッケージングされたPluginファイル

Private key file

パッケージングの際に作成された秘密鍵

2回目以降のパッケージングコマンド

`package.sh <manifest.jsonが置かれているフォルダ> <生成された秘密鍵ファイル>`

秘密鍵ファイルを指定しないと
新たなプラグインとして作成されてしまう

実行例

`sh ./package.sh plugin/ keys/plugin.godoojndmlcajgabgldkpbifjiplgo.ppk`

プラグイン開発の流れ

- ① plugin-sdkの準備
- ② プラグイン用ファイルの作成
- ③ マニフェストファイルの作成
- ④ 設定画面機能の実装
- ⑤ カスタマイズ機能の実装
- ⑥ パッケージング
- ⑦ **kintoneへの適用**

結果例

Plugin ID: godoojndmlcajgabgldkpbifjipioplgo

Plugin file: /Users/y-shinomiya/Documents/kintone-plugin/plugins/godoojndmlcajgabgldkpbifjipioplgo/plugin.zip

Private key file: /Users/y-shinomiya/Documents/kintone-plugin/plugins/godoojndmlcajgabgldkpbifjipioplgo/keys/plugin_godoojndmlcajgabgldkpbifjipioplgo_privatekey.pem

このファイルをkintoneにインポート

項目	説明
Plugin ID	プラグインID
Plugin file	パッケージングされたプラグインファイル
Private key file	パッケージングの際に作成された秘密鍵

kintoneへの適用



その他

利用する機能の選択

プラグイン

新旧デザインの切り替え

スマートフォンでの表示

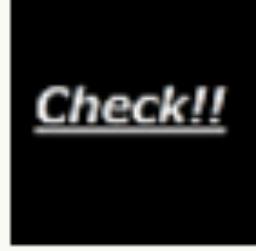
The screenshot shows the kintone System Management interface with the 'Plugins' tab selected. A red box highlights the 'Import' button ('読み込む') at the top of the page. Below it, there's a table with columns for 'Plugin Name' ('Plugin名') and 'Description' ('説明'). A brief description in Japanese is visible on the right side of the screen.

kintoneシステム管理画面から
plugin.zipを読み込む

kintoneへの適用

プラグイン

読み込んだプラグイン

プラグイン名	説明	利用中のアプリ
条件付き必須入力チェックプラグイン 	指定したフィールドに値が入っている際に対応する他のフィールドに値が入っているかをチェックするプラグインです。	

「読み込んだプラグイン」に表示されていることを確認

サンプルアプリの作成

The screenshot shows the kintone System Management interface. At the top, there is a navigation bar with icons for three, home, bell, and star, followed by the text "試用期間：残り30日" (Trial Period: 30 days left), a blue "購入" (Purchase) button, a gear icon, and a question mark icon. Below the navigation bar, the text "kintoneシステム管理" is displayed. On the left, a sidebar menu is open under the heading "kintoneシステム管理". The "アプリ" (App) section is expanded, showing "アプリ管理" (App Management) and "アプリテンプレート" (App Template). The "アプリテンプレート" option is highlighted with a red rectangle. Below this, the "スペース" (Space) section is expanded, showing "スペース管理" (Space Management), "スペーステンプレート" (Space Template), and "スレッドのアクション" (Thread Action). On the right side of the interface, there is a toolbar with buttons for "作成" (Create), "読み込む" (Import) [highlighted with a red rectangle], "書き出す" (Export), and "削除" (Delete). The text "kintoneシステム管理" is also present above the toolbar. Below the toolbar, the text "アプリテンプレートの一覧" (List of App Templates) is displayed.

kintoneシステム管理画面から
「日報_デブサミ用.zip」を読み込む

サンプルアプリの作成



ポータル画面から「日報_カスタマイズ」
アプリを作成

サンプルアプリの作成

日付 担当者

2016-02-15  

ログインユーザー 

活動タイトル 1

活動内容 1

**活動タイトル 1 (title 1)
に値が入力されると
活動内容 1 (body 1)を
必須項目にする**

プラグインの設定

The screenshot shows the Joyzo application management interface. On the left, there is a sidebar with the following options:

- 高度な設定
- 言語ごとの名称
- JavaScript / CSSでカスタマイズ
- Plugin** (This option is highlighted with a red box)
- APIトークン

On the right, under the 'Plugin' section, there is a button labeled 'Pluginを追加' (Add Plugin) which is also highlighted with a red box. Below this button, there is a message: 'Plugin名' (Plugin Name) and '利用中のPluginはありません。' (No active Plugins).

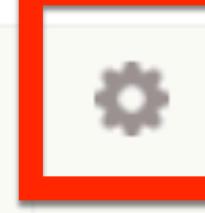
アプリ管理画面の「Plugin」から
作成したPluginを追加

プラグインの設定



アプリ管理画面の「プラグイン」から
作成したプラグインを追加

プラグインの設定

プラグイン名	設定	説明	
条件付き必須入力チェックプラグイン <small>CheckIt!</small>	 必須項目が設定されていません。	指定したフィールドに値が入っている際に別のフィールドに値が入っているかをチェックするプラグインです。	

プラグインの設定画面を開く

プラグインの設定

カテゴリー：
プラグイン

条件付き必須入力チェックプラ
グイン

Check!!

詳細
バージョン：1

必須フィールド組み合わせ-1

必須元フィールド* :

必須対象フィールド* :

必須フィールド組み合わせ-2

必須元フィールド :

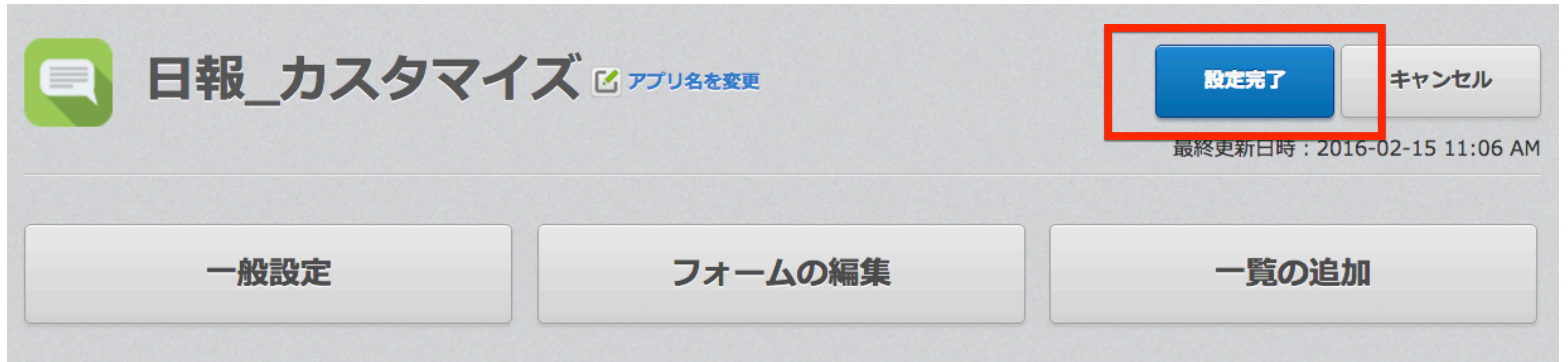
必須対象フィールド :

保存する **キャンセル**



必須元フォールドに「title1」
必須対象フィールドに「body1」
と入力し、「保存する」をクリック

プラグインの設定



「設定完了」でアプリの設定を反映させる

※これを忘れるとなにかの設定が反映されないので注意

動作確認

日報_カスタマイズ

保存

担当者

2016-02-15

活動タイトル 1

活動内容 1

テスト

「活動タイトル 1」に値を入力し、「活動内容 1」は空のまま保存

日報_カスタマイズ

保存

担当者

2016-02-15

system39@joyzo.co.jp

活動タイトル 1

活動内容 1

テスト

活動内容 1

必須項目です

「活動内容 1」の下にエラーメッセージが表示される

設定画面のフィールドコード指定を
テキストではなくプルダウンから
選択可能にしましょう

機能拡張

The screenshot shows a user interface for plugin configuration. On the left, there's a sidebar with navigation items like '日報_カスタマイズ', 'アプリの設定', 'プラグイン', and 'プラグインの設定'. Below this, there are sections for '条件付き必須入力チェック' (Conditional Required Input Check) and 'Check!!'. At the bottom of the sidebar are '詳細' and 'バージョン: 1'. The main area has two sections: '必須フィールド組み合わせ-1' and '必須フィールド組み合わせ-2'. Each section has a dropdown for '必須元フィールド*' and another for '必須対象フィールド*'. A large orange banner at the bottom contains the text: 'デプロイAPIを使ってフィールド情報を取得し、プルダウンを作成'.

活動内容2
更新日時
作成者
活動タイトル1
担当者
ステータス
作業者
日付
作成日時
レコード番号
活動タイトル2
カテゴリー^{*}
更新者
✓ 活動内容1

必須フィールド組み合わせ-1

必須元フィールド*: 活動タイトル1

必須対象フィールド*: 活動内容2

必須フィールド組み合わせ-2

必須元フィールド: 活動タイトル2

必須対象フィールド: 活動内容2

デプロイAPIを使ってフィールド情報を取得し、プルダウンを作成

manifest.jsonを編集

```
"config": {  
    "html": "html/config_advance.html",  
    "js": [  
        "https://js.cybozu.com/jquery/1.11.1/jquery.min.js",  
        "js/config_advance.js"  
    ],  
    "css": [  
        "css/51-us-default.css"  
    ],  
    "required_params": ["field-1", "require-field-1"]  
}
```

2回目以降のパッケージングコマンド

package.sh <manifest.jsonが置かれているフォルダ> <生成された秘密鍵ファイル>

1回目で作成された秘密鍵ファイルを指定

実行例

sh ./package.sh plugin/ keys/plugin.godoojndmlcajgabgldkpbifjiplgo.ppk

機能拡張

その他

利用する機能の選択

プラグイン

新旧デザインの切り替え

スマートフォンでの表示

The screenshot shows the 'kintoneシステム管理' (System Management) interface with the 'Plugin' tab selected. A red box highlights the 'Import' button ('読み込む') in the top navigation bar. Below it, there's a section titled 'Plugin' and a table header for 'Plugin Store' with columns for 'Plugin Name' and 'Description'. A large orange callout box covers the bottom half of the screenshot, containing the explanatory text.

kintoneシステム管理画面から
plugin.zipを読み込む

プリとGaroonのスケジュールを
Garoonの予定を閲覧したり、 Ga
す。

```
//フィールド情報を取得
kintone.api(kintone.api.url('/k/v1/preview/app/form/fields', true), 'GET', {'app': kintone.app.getId()},
function(resp) {

},function(error){ //一覧の取得に失敗
alert("プラグイン設定画面の生成に失敗しました。");
return;
});
```

フィールド一覧を取得するデプロイ系API
をkintone REST APIでリクエスト

kintone REST APIでリクエスト

kintone.api(pathOrUrl,method,params,callback,errback)

kintone REST APIの GET/POST/PUT/DELETE が実行できるAPI

パラメーター	指定する値	説明
pathOrUrl	文字列	kintone REST APIのパス or kintone.api.urlで取得したURL
method	文字列	使用するメソッド (GET/POST/PUT/DELETE)
params	オブジェクト	APIに渡すパラメータ
callback	関数	成功時に実行されるコールバック関数
errback	関数	失敗時に実行されるコールバック関数

フィールド一覧取得API

運用環境の設定取得用URL

https://(サブドメイン名).cybozu.com/k/v1/app/form/fields.json

テスト環境の設定取得用URL

https://(サブドメイン名).cybozu.com/k/v1/preview/app/form/fields.json

パラメーター	指定する値	説明
app	整数	アプリIDの
lang	文字列	取得する名称の言語（省略可）

```
kintone.api(kintone.api.url('/k/v1/preview/app/form/fields', true), 'GET', {'app': kintone.app.getId()}, function(resp) {  
    //取得したフィールド一覧をプルダウンにセット  
    for (var key in resp.properties){ ← for..in文でオブジェクトのプロパティを取り出す  
        $('[name=field-1]').append($('option').html(resp.properties[key].label).val(resp.properties[key].code));  
        $('[name=require-field-1]').append($('option').html(resp.properties[key].label).val(resp.properties[key].code));  
        $('[name=field-2]').append($('option').html(resp.properties[key].label).val(resp.properties[key].code));  
        $('[name=require-field-2]').append($('option').html(resp.properties[key].label).val(resp.properties[key].code));  
    }  
}
```

レスポンスパラメータは**properties**の
後にフィールドコードが入る

例) **resp.properties.title1.code**

**Enjoy IT,
Enjoy LIFE**

