AUTODESK Platform Services

APS Online Training – Inventor Automation

加藤 丈博

Developer **A**dvocacy & **S**upport、Autodesk Developer Network

はじめに

Automation APIの概要、仕組み、注意点については…

- AutoCAD Automation APS Online Trainingの公開動画を参照ください。
  - AutoCAD Automationを題材にしたトレーニングですが、Automation APIの仕組みや、留意点 等はほぼ共通です。

参考ブログ記事：APS Online Training：AutoCAD Automation 収録公開

このトレーニングの内容

- 作成するアプリケーションの要件~機能やAutomation APIの仕組みから、順を追ってサンプルソースコードが何故そうなっているのか？を中心に解説します。
  - 基本的にはチュートリアルでのステップに準じて解説していきますが、説明の都合上チュートリアルの順序から前後します。

サンプルアプリケーションの開発

Autodesk Forge - Design Autom

localhost:3000

AUTODESK
Platform Services

Configure

Width:

80

Height:

Enter new height value

Input file

Choose File AutoCADSample.dwg

Existing activities

UpdateDWGParamActivity+dev

Start workitem

27°C Mostly sunny 16:16

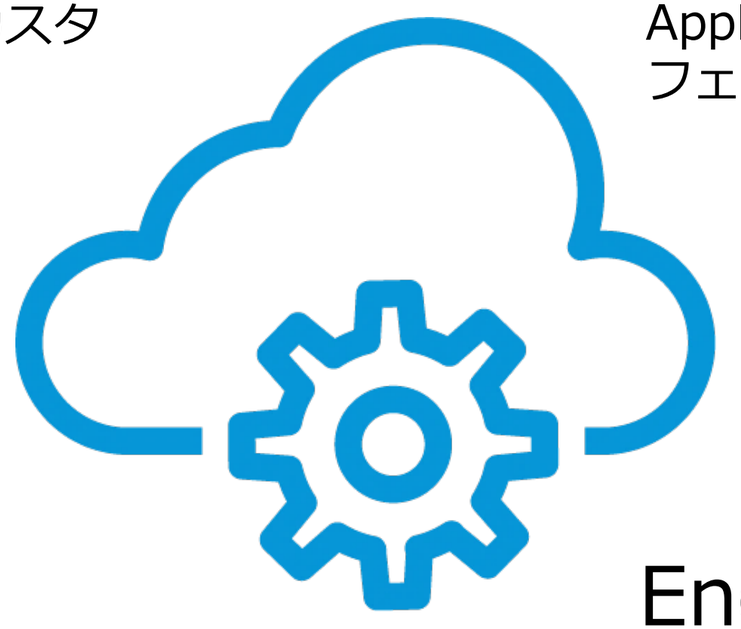
Automation APIの用語

AppBundle

各CADのAPIを使用したカスタム処理

Activity

AppBundleとWorkItemのインターフェース



WorkItem

Inventor Serverを実行するJob

Engine

各CADツールに対応する
複数バージョンのエンジン

AUTOCAD®
REVIT®
INVENTOR®
3DS MAX®
Fusion

Automation API

- **起動したコアエンジンにアドインをロード/実行**
  - AppBundle（事前登録が必要）
- **成果物としてデザインファイルや関連ファイルを作成して ダウンロードして利用**
  - Activityで入出力ファイル/パラメータを宣言（事前登録が必要)
  - WorkItem（ファイル/パラメータ渡しを実行）
- **クラウド上の CAD コアエンジンをリモートで起動**
  - WorkItem（実際の処理実行）

サンプルアプリケーションの要件

- 独自のWebアプリケーション
- 実行するAutomation APIのエンジンを選択できる
- 使用するパッケージ バンドル(ZIPファイル)を指定できる
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

サンプルアプリケーションの要件

- 独自のWebアプリケーション
- 実行するAutomation APIのエンジンを選択できる
- 使用するパッケージ バンドル(ZIPファイル)を指定できる
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

独自のWebアプリケーション

- Webアプリケーションサーバの選択

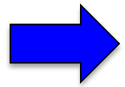
独自のWebアプリケーション

- Visual Studio ASP.NET Core Webアプリケーション作成

Setup Project

Node.js & VSCode　.NET & VSCode　**.NET & VS2022**

- Launch Visual Studio 2022, select `Create New Project`
- Enter `ASP.NET Core Empty` in templates search bar. Select and Next.
- On the next dialog, let's name it `designAutomationSample`, Next.
- On the next dialog, ensure .NET 6.0 (Long Term Support) is selected, uncheck `Configure for HTTPS`.
- Disable use of top level-statements by checking the `Do not use top-level statements` box, then click `create`.



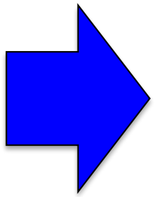
今回は.NET8に変更します

サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- 実行するAutomation APIのエンジンを選択できる
- 使用するパッケージ バンドル(ZIPファイル)を指定できる
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

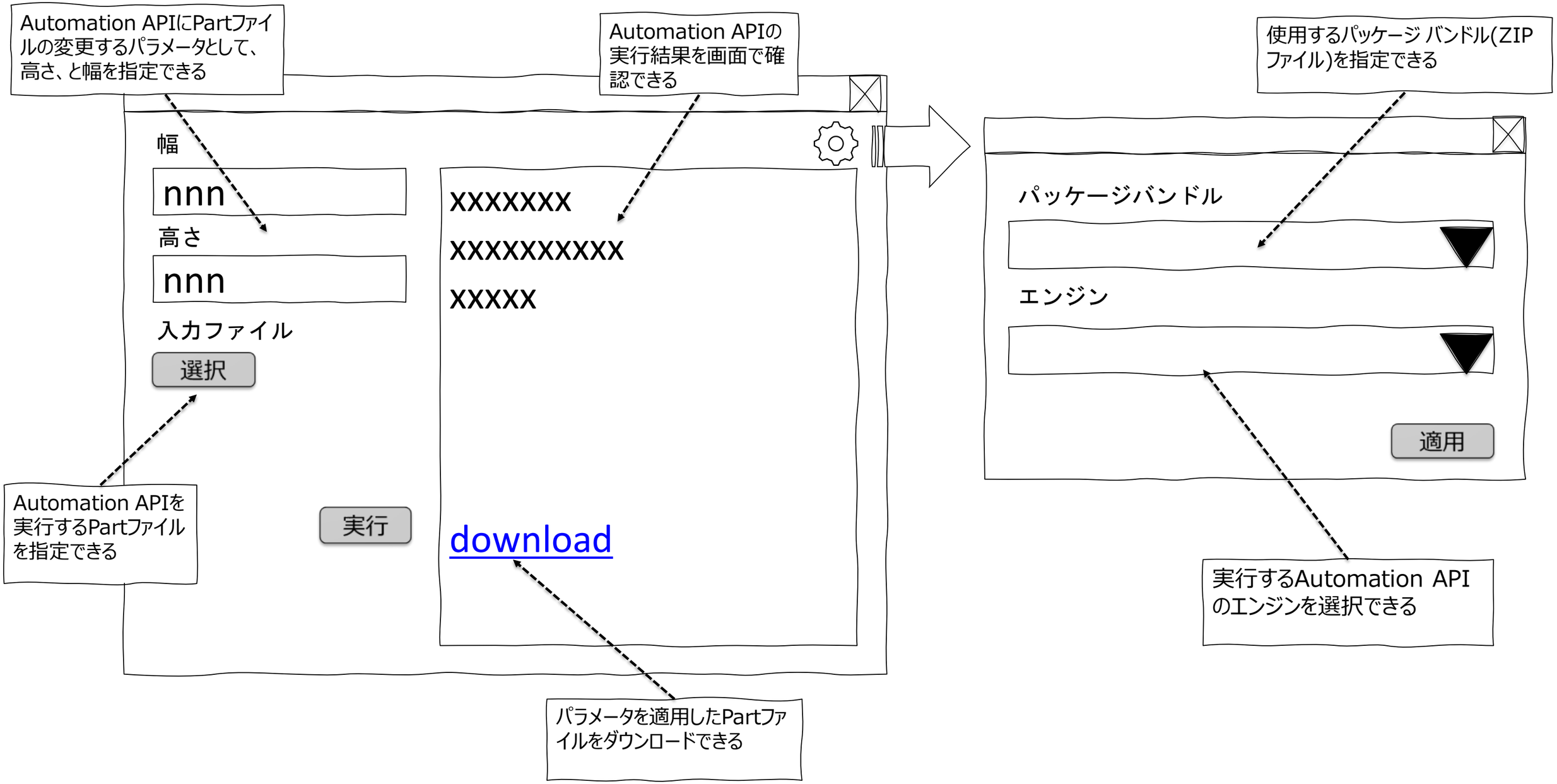
サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
  - 実行するAutomation APIのエンジンを選択できる
  - 使用するパッケージ バンドル(ZIPファイル)を指定できる
  - Automation APIを実行するPartファイルを指定できる
  - Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
  - Automation APIの実行結果を画面で確認できる
  - パラメータを適用したPartファイルをダウンロードできる



GUIを考えてみる

サンプルアプリケーションのGUIと機能



GUIの作成

- wwwrootフォルダ配下に、index.htmlを作成

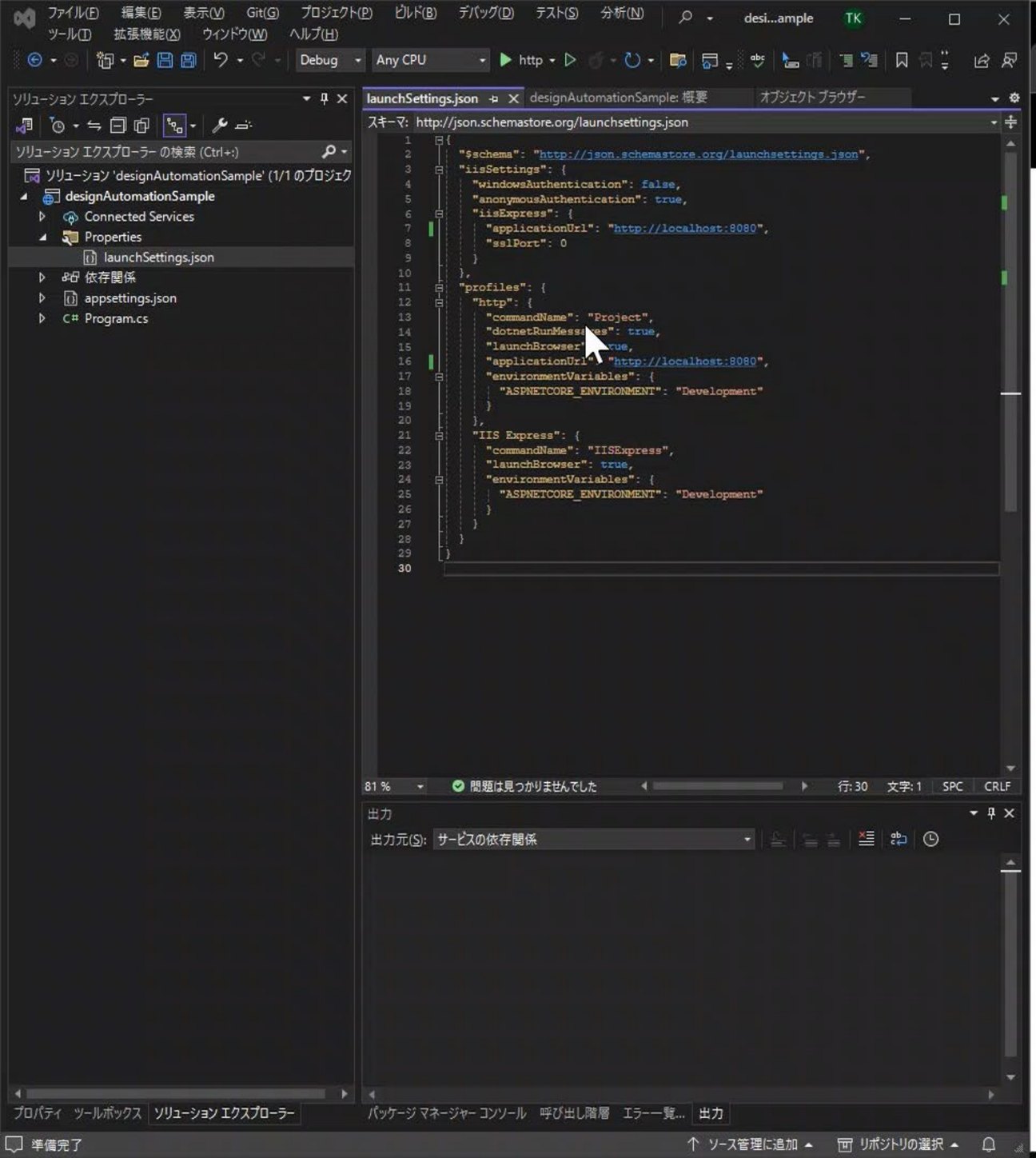
Basic UI

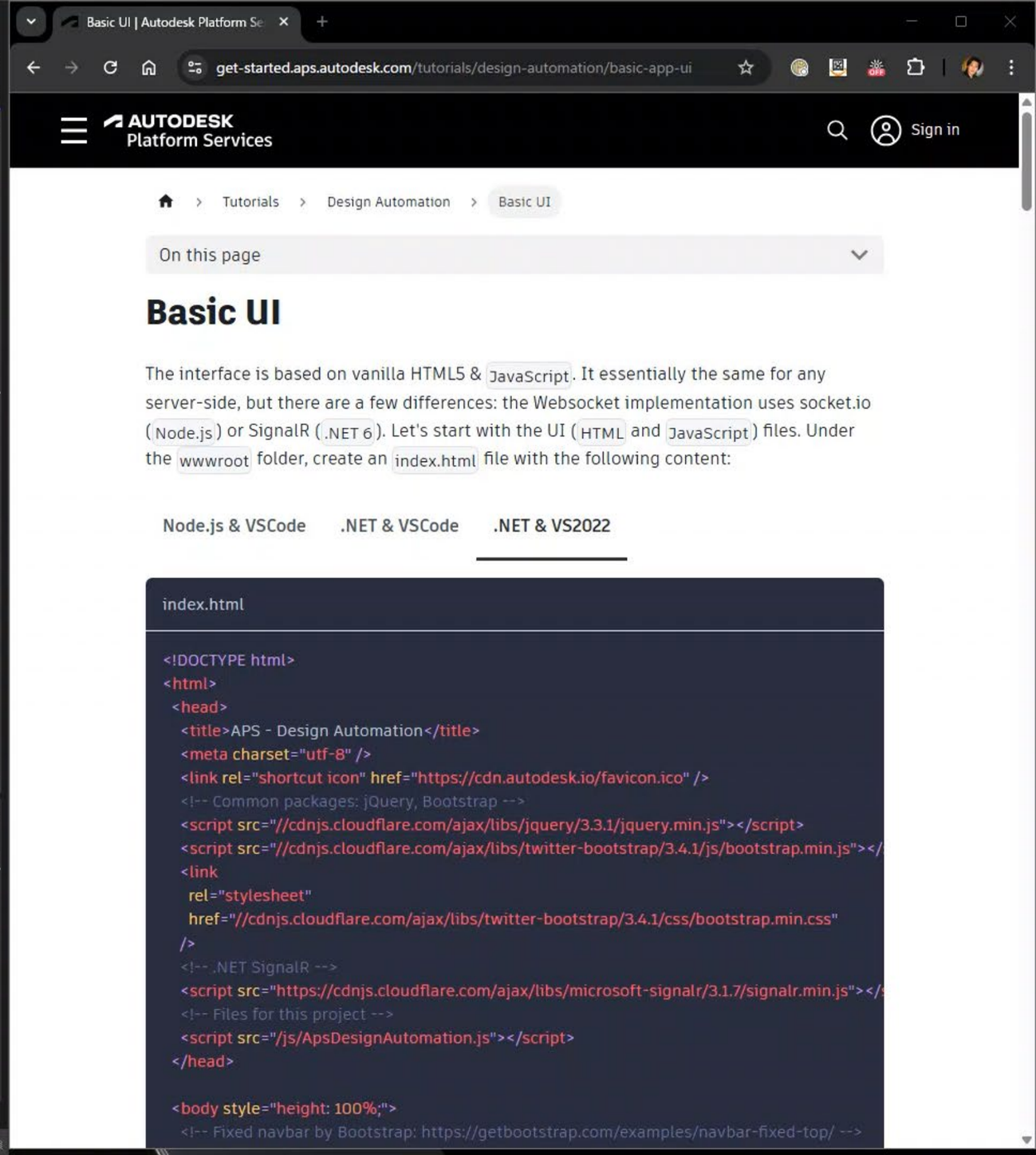
The interface is based on vanilla HTML5 & `JavaScript`. It essentially the same for any server-side, but there are a few differences: the Websocket implementation uses socket.io (`Node.js`) or SignalR (`.NET 6`). Let's start with the UI (`HTML` and `JavaScript`) files. Under the `wwwroot` folder, create an `index.html` file with the following content:

Node.js & VSCode　.NET & VSCode　**.NET & VS2022**

index.html

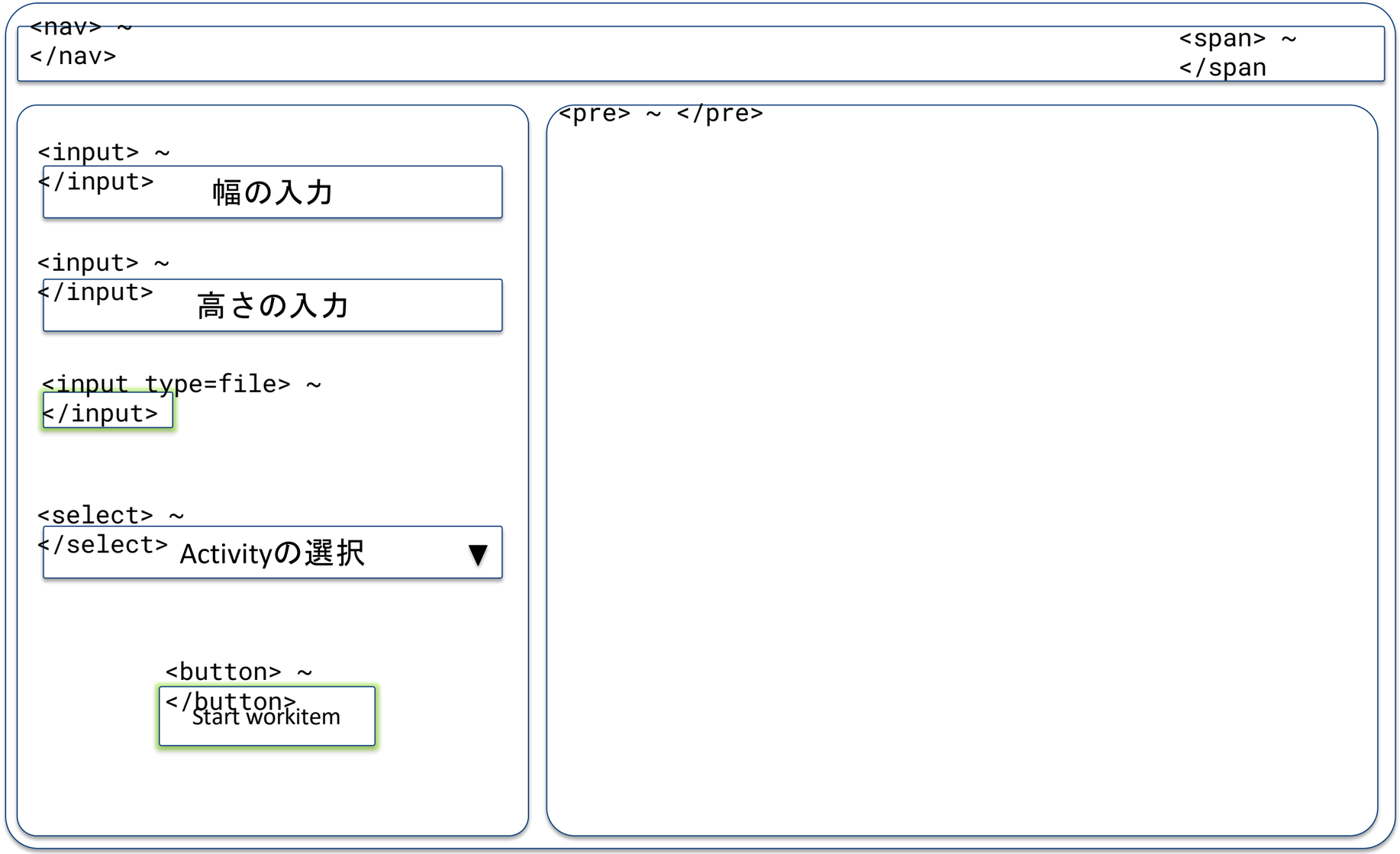
```
<!DOCTYPE html>
<html>
 <head>
  <title>APS - Design Automation</title>
  <meta charset="utf-8" />
  <link rel="shortcut icon" href="https://cdn.autodesk.io/favicon.ico" />
  <!-- Common packages: jQuery, Bootstrap -->
  <script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
```


launchSettings.json
{
"$schema": "http://json.schemastore.org/launchsettings.json",
"iisSettings": {
"windowsAuthentication": false,
"anonymousAuthentication": true,
"iisExpress": {
"applicationUrl": "http://localhost:8080",
"sslPort": 0
}
},
"profiles": {
"http": {
"commandName": "Project",
"launchBrowser": true,
"applicationUrl": "http://localhost:8080",
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
},
"IIS Express": {
"commandName": "IISExpress",
"launchBrowser": true,
"environmentVariables": {
"ASPNETCORE_ENVIRONMENT": "Development"
}
}
}
}


Basic UI | Autodesk Platform Se
get-started.aps.autodesk.com/tutorials/design-automation/basic-app-ui
AUTODESK
Platform Services
Sign in
Tutorials
Design Automation
Basic UI
On this page
Basic UI
The interface is based on vanilla HTML5 & JavaScript. It essentially the same for any server-side, but there are a few differences: the Websocket implementation uses socket.io (Node.js) or SignalR (.NET 6). Let's start with the UI (HTML and JavaScript) files. Under the wwwroot folder, create an index.html file with the following content:
Node.js & VSCode
.NET & VSCode
.NET & VS2022
index.html
<!DOCTYPE html>
<html>
<head>
<title>APS - Design Automation</title>
<meta charset="utf-8" />
<link rel="shortcut icon" href="https://cdn.autodesk.io/favicon.ico" />
<!-- Common packages: jQuery, Bootstrap -->
<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
<script src="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.4.1/js/bootstrap.min.js"></
<link
rel="stylesheet"
href="//cdnjs.cloudflare.com/ajax/libs/twitter-bootstrap/3.4.1/css/bootstrap.min.css"
/>
<!-- .NET SignalR -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/microsoft-signalr/3.1.7/signalr.min.js"></
<!-- Files for this project -->
<script src="/js/ApsDesignAutomation.js"></script>
</head>
<body style="height: 100%;">
<!-- Fixed navbar by Bootstrap: https://getbootstrap.com/examples/navbar-fixed-top/ -->

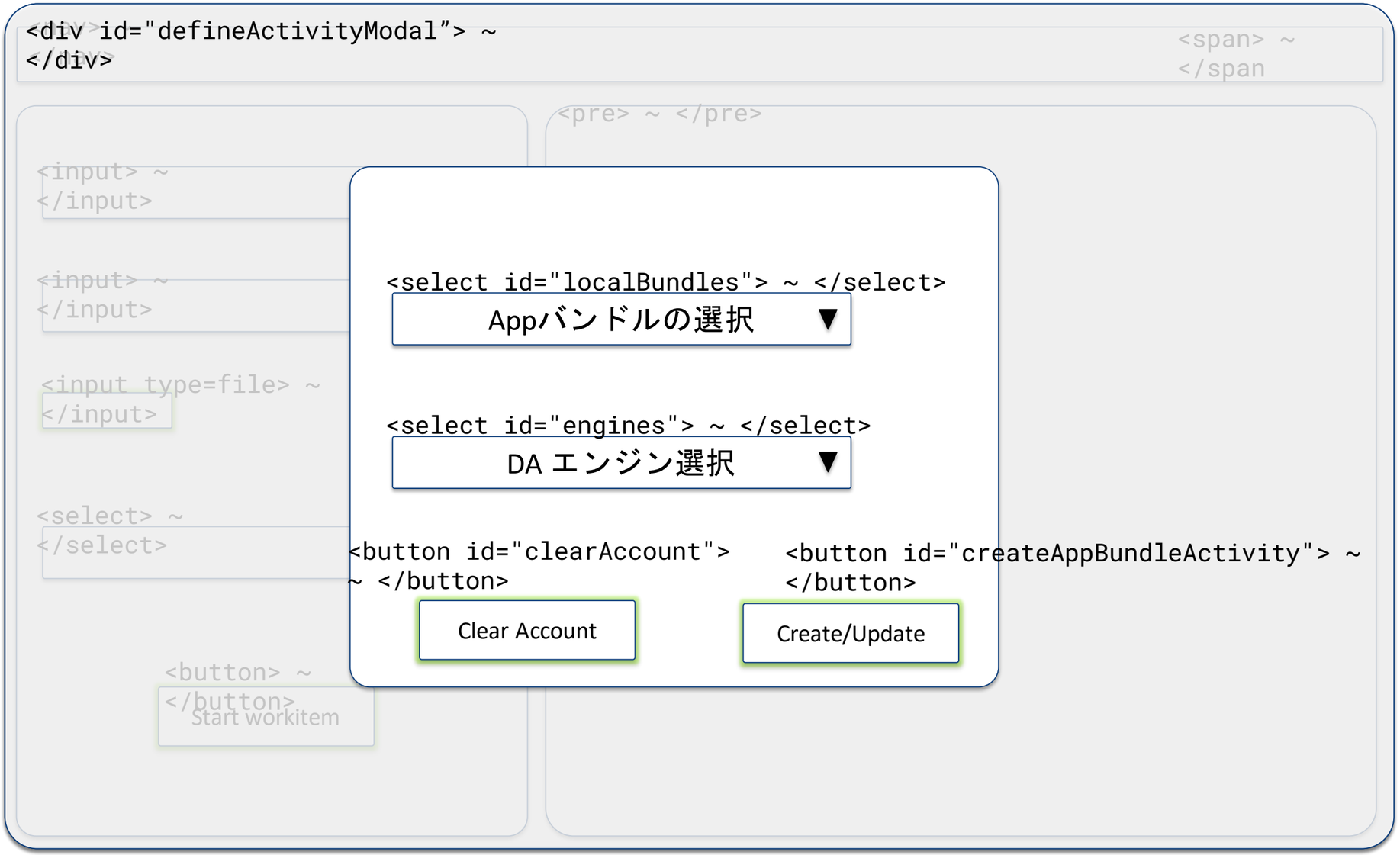
HTML の構造 メイン画面

index.html



HTML の構造 モーダルポップアップ

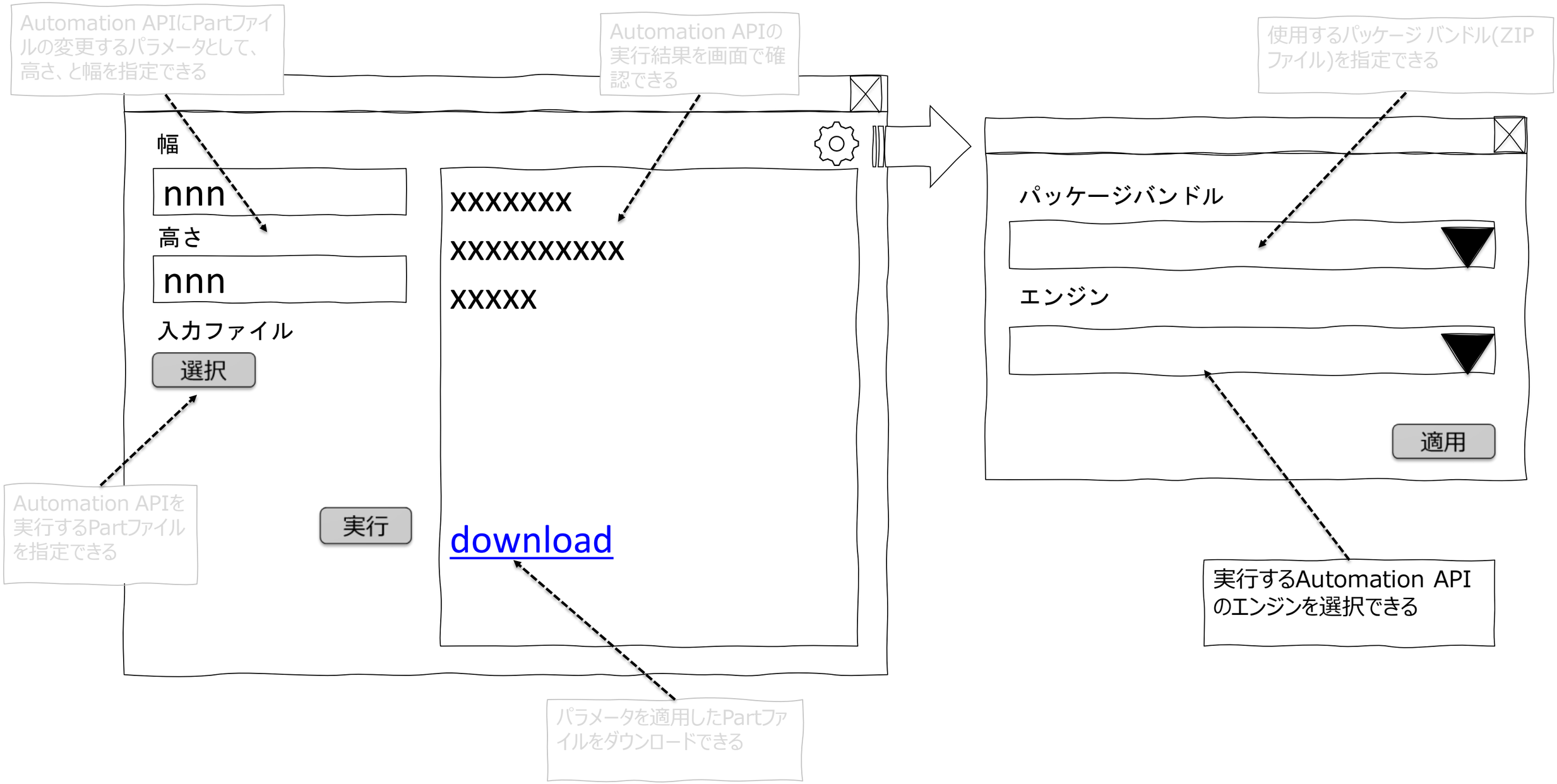
index.html



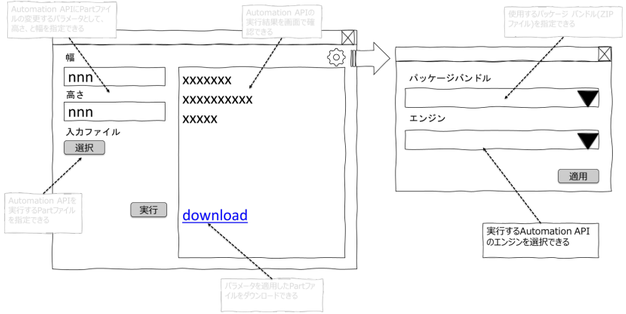
サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- 実行するAutomation APIのエンジンを選択できる
- 使用するパッケージ バンドル(ZIPファイル)を指定できる
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

機能設計～実行するAutomation APIのエンジンを選択できる



機能設計～実行するAutomation APIのエンジンを選択できる



クライアント(JS)

- モーダルダイアログの表示
- エンジンのリストをリクエスト～結果を選択リストに格納

Webサーバ

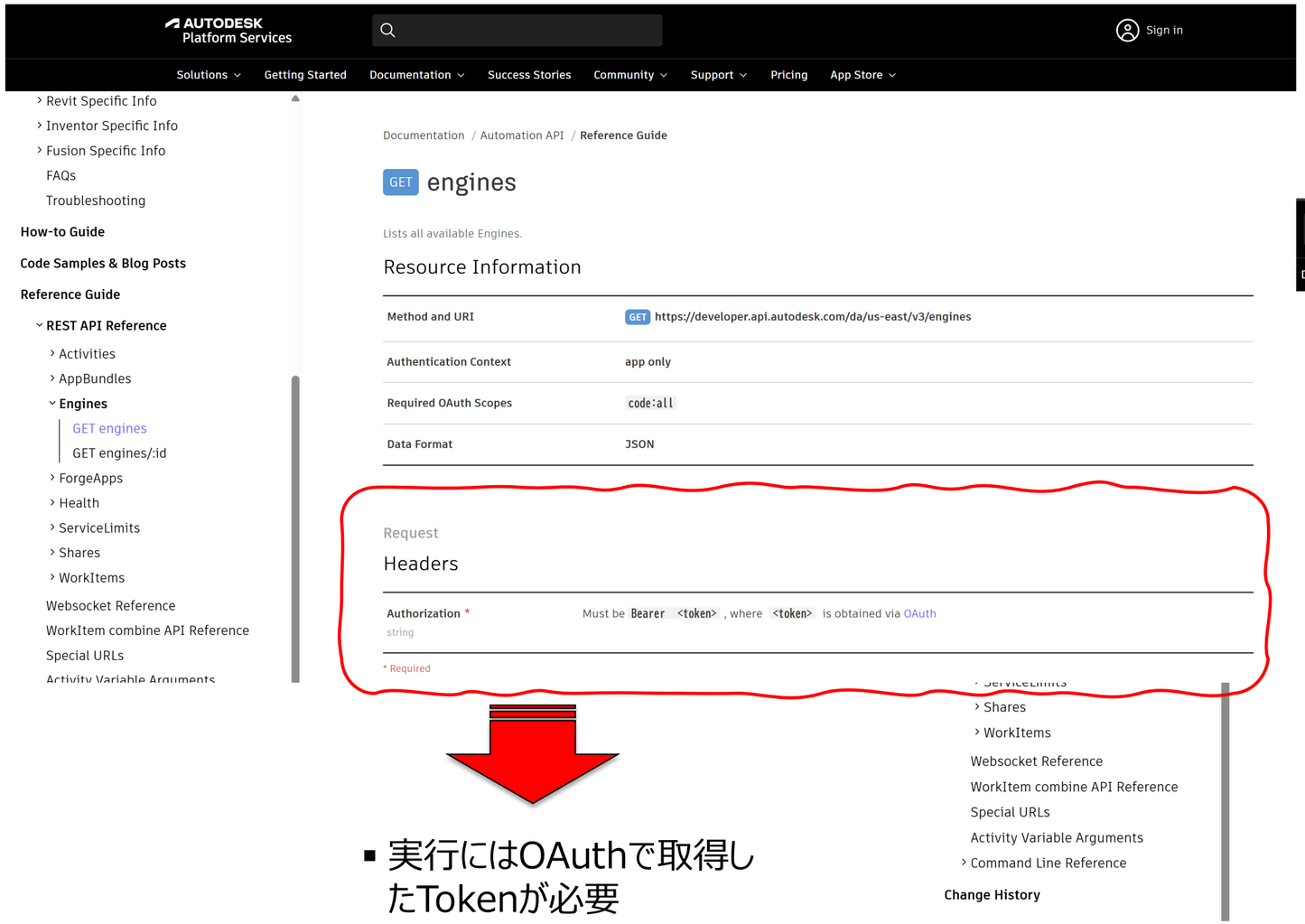
- APSのAPIを実行して、エンジンのリストを取得

APS（API）

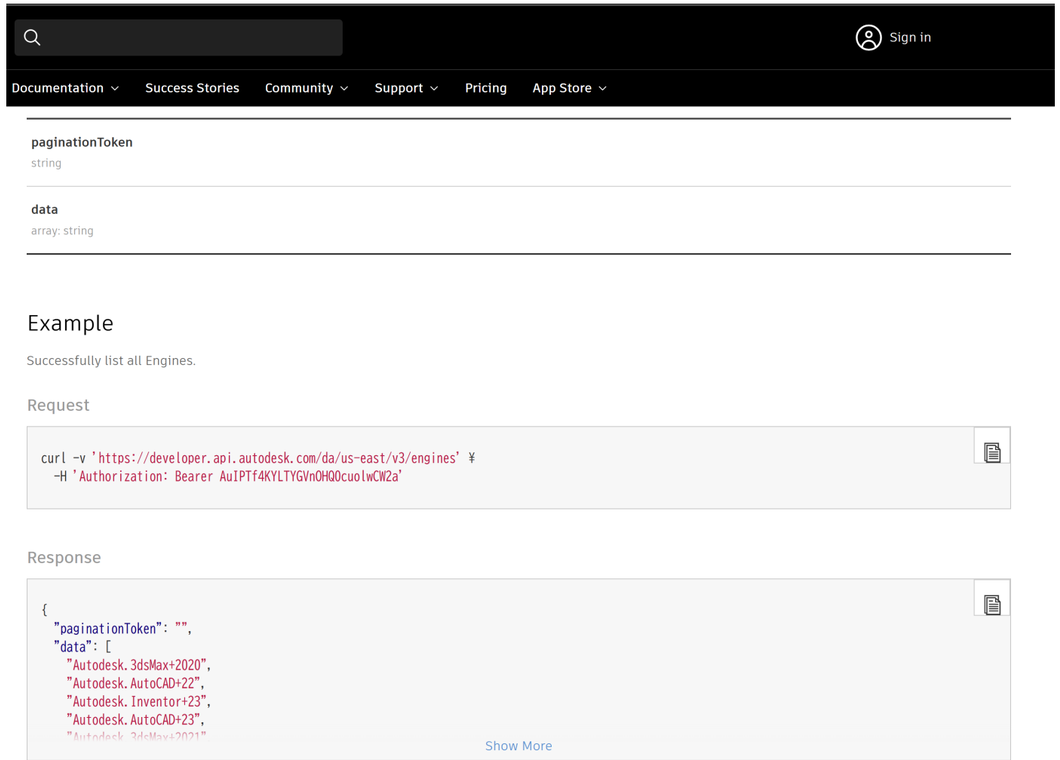
- エンジンのリストを取得するAPI

機能設計～実行するAutomation APIのエンジンを選択できる

- エンジンのリストを取得

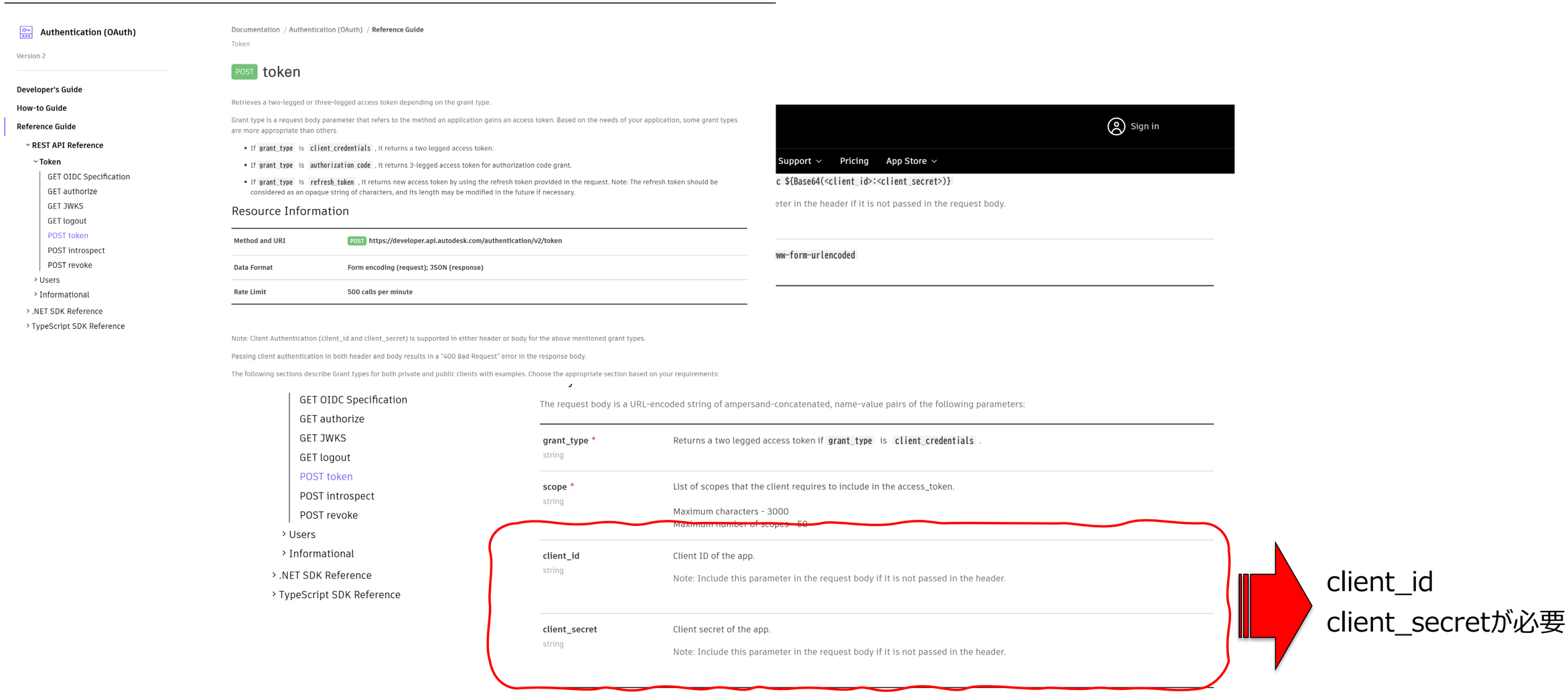


- 実行にはOAuthで取得したTokenが必要

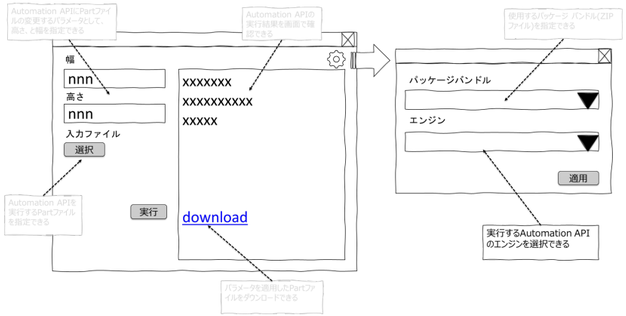


機能設計～実行するAutomation APIのエンジンを選択できる

- OauthでTokenを取得



機能設計～実行するAutomation APIのエンジンを選択できる



クライアント(JS)

- モーダルダイアログの表示
- エンジンのリストをリクエスト～結果を選択リストに格納

Webサーバ

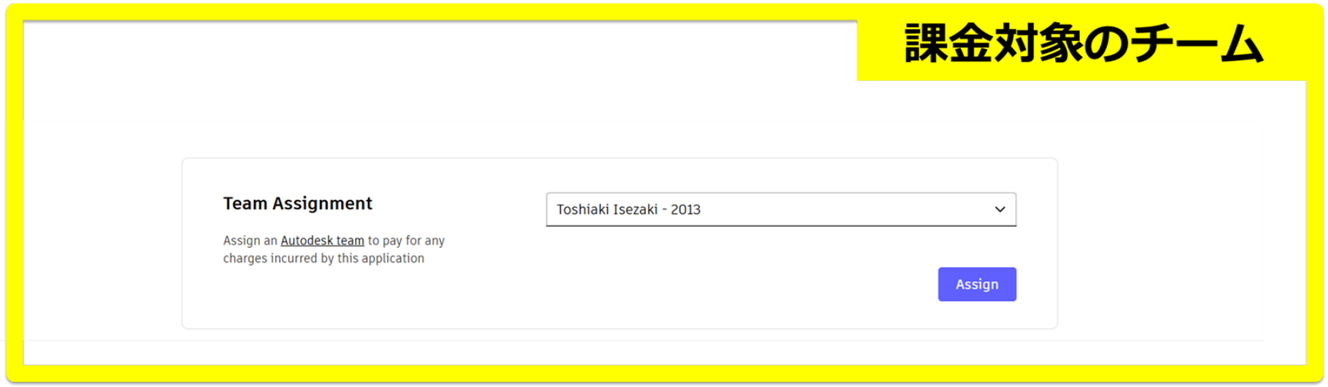
- **client_idとclient_secretを指定してOAuthでTokenを取得**
- APSのAPIを実行して、エンジンのリストを取得

APS（API）

- **OAuthでTokenを取得するAPI**
- エンジンのリストを取得するAPI

非機能要件～client idとclient_secretはどこに置く？

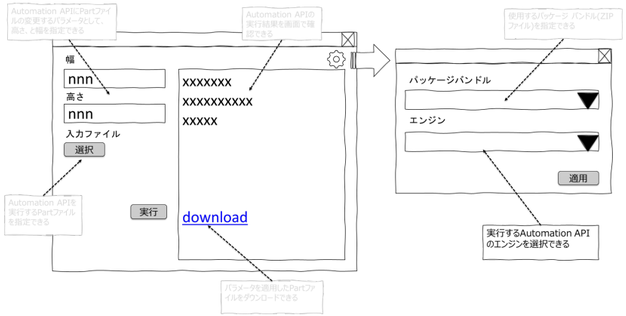
- APSの課金APIを実行した場合、Tokenを取得したclient idとclient_secretを生成したAPSアプリでTeam Assignmentの設定に従い、課金が行われます。



- とても大事な情報です。第三者から見える場所（例クライアントのJavaScriptや、wwwroot配下のフォルダ等）はNGです。

参考動画：APS Online Training：AutoCAD Automation 収録公開 の デベロッパキーの取得（約8分） 3:20～

機能設計～実行するAutomation APIのエンジンを選択できる



クライアント(JS)

- モーダルダイアログの表示
- エンジンのリストをリクエスト～結果を選択リストに格納

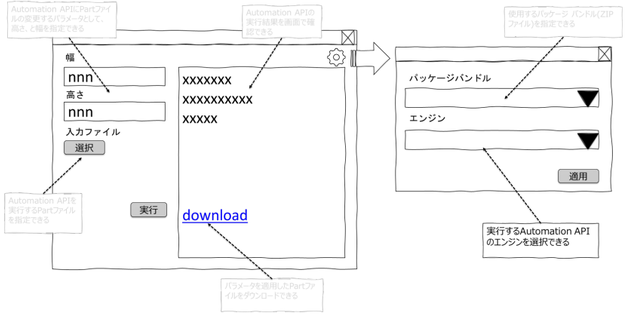
Webサーバ

- **client_idとclient_secretを、環境変数に設定**
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

APS（API）

- OAuthでTokenを取得するAPI
- エンジンのリストを取得するAPI

実装～実行するAutomation APIのエンジンを選択できる



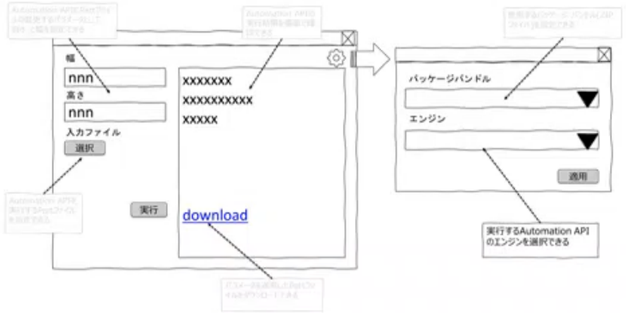
- モーダルダイアログの表示
- エンジンのリストをリクエスト～結果を選択リストに格納

ApsDesignAutomation.js

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
});

function prepareLists() {
  list("engines", "/api/aps/designautomation/engines");
}
```

```
function list(control, endpoint) {
  $("#" + control)
    .find("option")
    .remove()
    .end();
  jQuery.ajax({
    url: endpoint,
    success: function (list) {
      if (list.length === 0)
        $("#" + control).append(
          $("<option>", { disabled: true, text: "Nothing found" })
        );
      else
        list.forEach(function (item) {
          $("#" + control).append($("<option>", { value: item,
text: item }));
        });
    },
  });
}

function defineActivityModal() {
  $("#defineActivityModal").modal();
}
```

実装～実行するAutomation APIのエンジンを選択できる



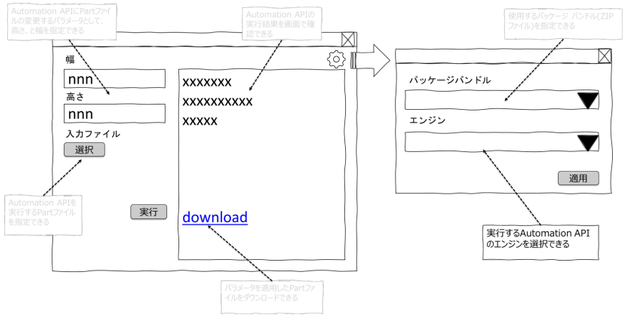
- モーダルダイアログの表示
- エンジンのリストをリクエスト～結果を選択リストに格納

ApsDesignAutomation.js

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
});

function prepareLists() {
  list("engines", "/api/aps/designautomation/engines");
}
```

```
function list(control, endpoint) {
  $("#" + control)
    .find("option")
    .remove()
    .end();
  jQuery.ajax({
    url: endpoint,
    success: function (list) {
      if (list.length === 0)
        $("#" + control).append(
          $("<option>", { disabled: true, text: "Nothing found" })
        );
      else
        list.forEach(function (item) {
          $("#" + control).append($("<option>", { value: item,
text: item }));
        });
    },
  });
}

function defineActivityModal() {
  $("#defineActivityModal").modal();
}
```

機能設計～実行するAutomation APIのエンジンを選択できる



クライアント(JS)

- ~~モーダルダイアログの表示~~
- ~~エンジンのリストをリクエスト～結果を選択リストに格納~~

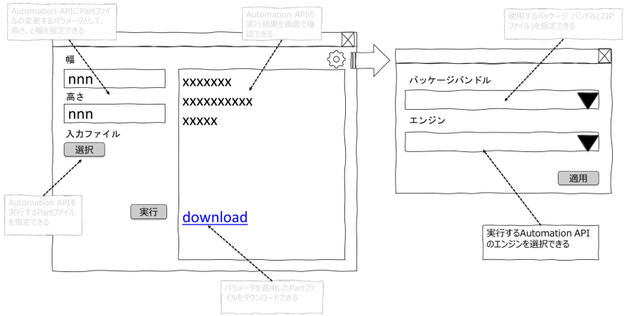
Webサーバ

- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

APS（API）

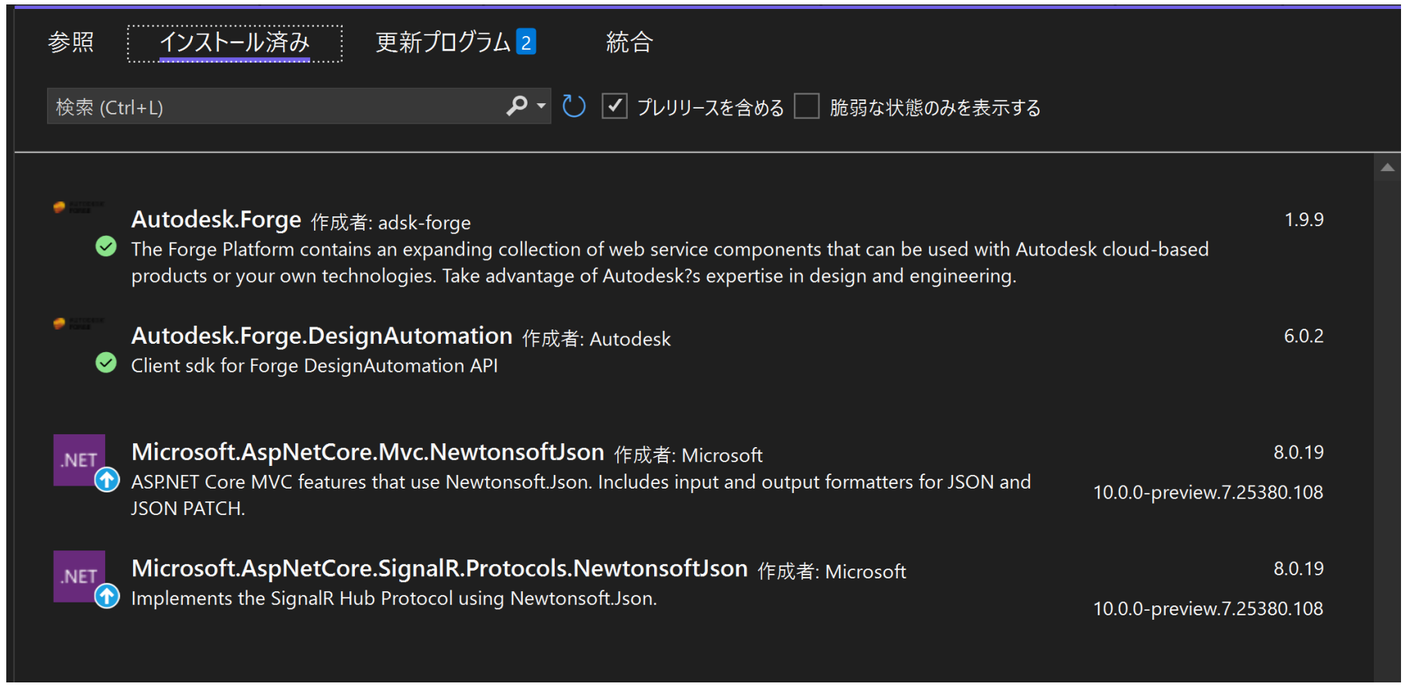
- OAuthでTokenを取得するAPI
- エンジンのリストを取得するAPI

実装～実行するAutomation APIのエンジンを選択できる



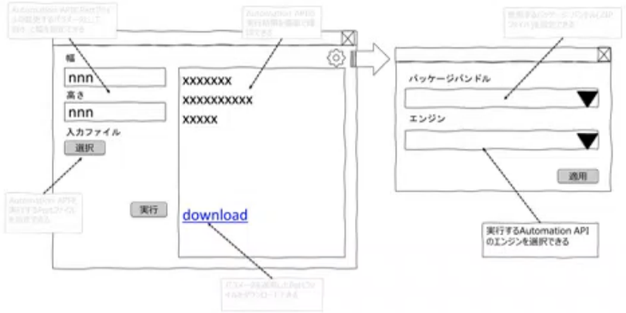
- OAuthでTokenを取得するAPI
- エンジンのリストを取得するAPI

- .NETから、Automation APIの実行、APSのAPI実行に利用可能な、Forge SDKとForge.DesignAutomation SDKをNuGetでインストール



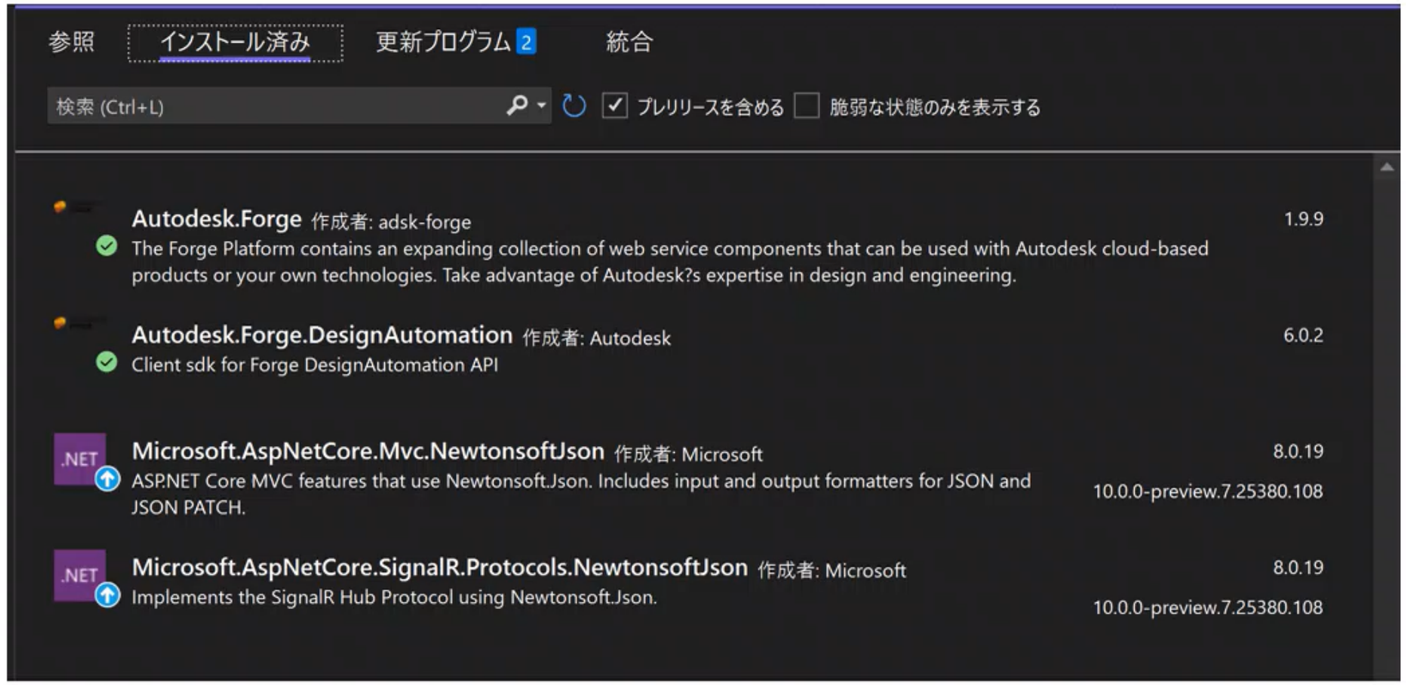
参考動画：APS Online Training：AutoCAD Automation 収録公開 の はじめに（約20分） 12:39～

実装～実行するAutomation APIのエンジンを選択できる



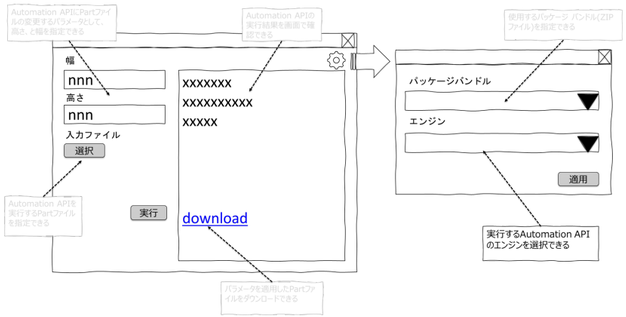
- OAuthでTokenを取得するAPI
- エンジンのリストを取得するAPI

- .NETから、Automation APIの実行、APSのAPI実行に利用可能な、Forge SDKとForge.DesignAutomation SDKをNuGetでインストール



参考動画：APS Online Training：AutoCAD Automation 収録公開 の はじめに（約20分） 12:39～

実装～実行するAutomation APIのエンジンを選択できる



- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

▪client_idとclient_secretを、launchSettings.jsonと、appsettings.user.jsonに記述

- Inside the `launchSettings.json` go to `profiles\designAutomationSample\environmentVariables` add following environments.

```
ASPNETCORE_URLS:"http://localhost:8080"
APS_CLIENT_ID:"Your Id Here"
APS_CLIENT_SECRET:"Your Secret Here"
```

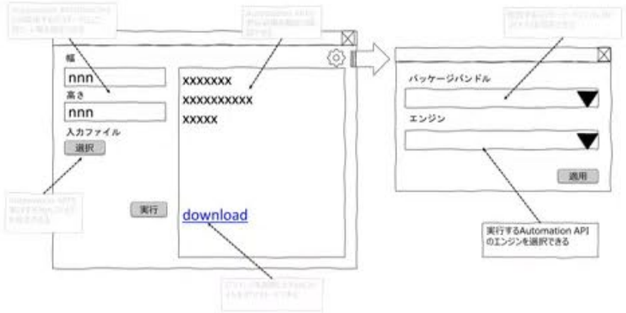
- Select `designAutomationSample` project, right-click `Add` -> `New Item`, select `JSON`.
- Name it as `appsettings.user.json`.
- Add following configuration settings, this is required by DesignAutomation SDK to create oAuth and run API requests.

appsettings.user.json

```
{
 "Forge": {
  "ClientId": "your client id",
  "ClientSecret": "your secret"
 }
}
```

参考動画：APS Online Training：AutoCAD Automation 収録公開 の デベロッパキーの取得（約8分） の～5:05

実装～実行するAutomation APIのエンジンを選択できる



- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

▪client_idとclient_secretを、launchSettings.jsonと、appsettings.user.jsonに記述

- Inside the `launchSettings.json` go to `profiles\designAutomationSample\environmentVariables` add following environments.

```
ASPNETCORE_URLS:"http://localhost:8080"
APS_CLIENT_ID:"Your Id Here"
APS_CLIENT_SECRET:"Your Secret Here"
```

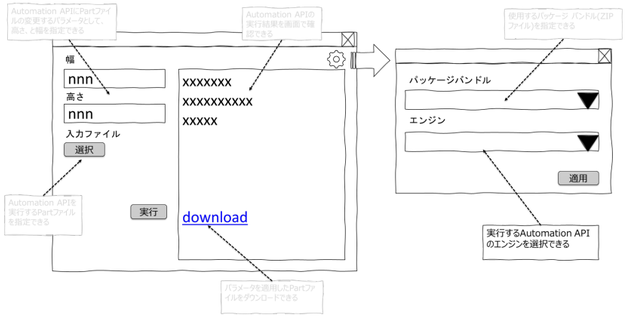
- Select `designAutomationSample` project, right-click `Add` -> `New Item`, select `JSON`.
- Name it as `appsettings.user.json`.
- Add following configuration settings, this is required by DesignAutomation SDK to create oAuth and run API requests.

appsettings.user.json

```
{
  "Forge": {
    "ClientId": "your client id",
    "ClientSecret": "your secret"
  }
}
```

参考動画：APS Online Training：AutoCAD Automation 収録公開 の デベロッパキーの取得（約8分） の～5:05

実装～実行するAutomation APIのエンジンを選択できる



- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

▪OAuthController.csを追加

OAuthController.cs

Create a Controllers folder, which will host the WebAPI Controllers. We'll need an `access token` to read & write input & output files to OSS Buckets. Under Controllers folder, create a `OAuthController.cs` file with the following content.

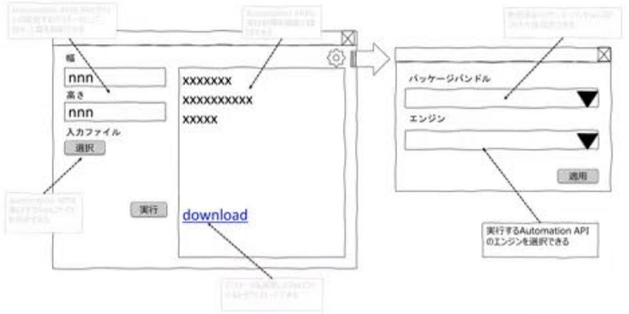
OAuthController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Autodesk.Forge;

namespace designAutomationSample.Controllers
{
  [ApiController]
  public class OAuthController : ControllerBase
  {
    // As both internal & public tokens are used for all visitors
    // we don't need to request a new token on every request, so let's
    // cache them using static variables. Note we still need to refresh
    // them after the expires_in time (in seconds)
    private static dynamic InternalToken { get; set; }
```

```
/// <summary>
/// Get access token with internal (write) scope
/// </summary>
public static async Task<dynamic> GetInternalAsync()
{
  if (InternalToken == null || InternalToken.ExpiresAt < DateTime.UtcNow)
  {
    InternalToken = await Get2LeggedTokenAsync(new Scope[] { Scope.BucketCreate, Sco
    InternalToken.ExpiresAt = DateTime.UtcNow.AddSeconds(InternalToken.expires_in);
  }

  return InternalToken;
}

/// <summary>
/// Get the access token from Autodesk
/// </summary>
private static async Task<dynamic> Get2LeggedTokenAsync(Scope[] scopes)
{
  TwoLeggedApi oauth = new TwoLeggedApi();
  string grantType = "client_credentials";
  dynamic bearer = await oauth.AuthenticateAsync(
    GetAppSetting("APS_CLIENT_ID"),
    GetAppSetting("APS_CLIENT_SECRET"),
    grantType,
    scopes);
  return bearer;
}
```

Forge SDKで、APSのRest APIを叩いてTokenを取得

実装～実行するAutomation APIのエンジンを選択できる



- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

▪OAuthController.csを追加

OAuthController.cs

Create a Controllers folder, which will host the WebAPI Controllers. We'll need an `access token` to read & write input & output files to OSS Buckets. Under Controllers folder, create a `OAuthController.cs` file with the following content.

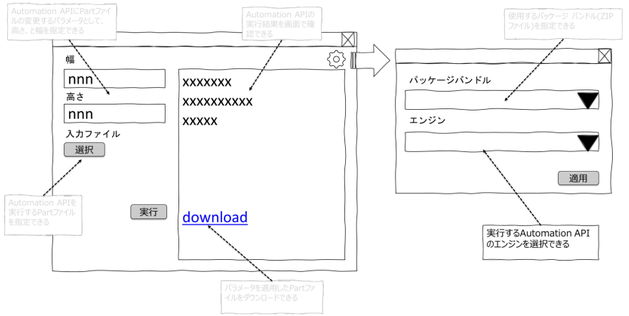
OAuthController.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Autodesk.Forge;

namespace designAutomationSample.Controllers
{
  [ApiController]
  public class OAuthController : ControllerBase
  {
    // As both internal & public tokens are used for all visitors
    // we don't need to request a new token on every request, so let's
    // cache them using static variables. Note we still need to refresh
    // them after the expires_in time (in seconds)
    private static dynamic InternalToken { get; set; }
```

```
/// <summary>
/// Get access token with internal (write) scope
/// </summary>
public static async Task<dynamic> GetInternalAsync()
{
  if (InternalToken == null || InternalToken.ExpiresAt < DateTime.UtcNow)
  {
    InternalToken = await Get2LeggedTokenAsync(new Scope[] { Scope.BucketCreate, Sco
    InternalToken.ExpiresAt = DateTime.UtcNow.AddSeconds(InternalToken.expires_in);
  }

  return InternalToken;
}

/// <summary>
/// Get the access token from Autodesk
/// </summary>
private static async Task<dynamic> Get2LeggedTokenAsync(Scope[] scopes)
{
  TwoLeggedApi oauth = new TwoLeggedApi();
  string grantType = "client_credentials";
  dynamic bearer = await oauth.AuthenticateAsync(
    GetAppSetting("APS_CLIENT_ID"),
    GetAppSetting("APS_CLIENT_SECRET"),
    grantType,
    scopes);
  return bearer;
}
```

Forge SDKで、APSのRest APIを叩いてTokenを取得

実装～実行するAutomation APIのエンジンを選択できる



- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

ApsDesignAutomation.js

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
});

function prepareLists() {
  list("engines", "/api/aps/designautomation/engines");
}
```

- GetAvailableEngines

To define a bundle we also nee
engines.

▪DesignAutomationController.csを追加

Node.js & VSCode　.NET & VSCode　.NET & VS2022

Under `Controllers` folder create a `DesignAutomationController.cs` with the following content. This is just the class, we'll define the endpoints later, but note the DesignAutomationHub at the end, which allow us push notifications to the client via SignalR.

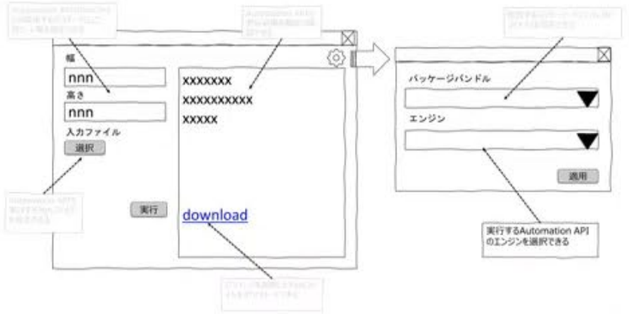
DesignAutomationController

```
using Autodesk.Forge;
using Autodesk.Forge.Client;
using Autodesk.Forge.DesignAutomation;
using Autodesk.Forge.DesignAutomation.Model;
using Autodesk.Forge.Model;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.SignalR;
using Newtonsoft.Json;
using Newtonsoft.Json.Linq;
```

```
/// <summary>
/// Return a list of available engines
/// </summary>
[HttpGet]
[Route("api/aps/designautomation/engines")]
public async Task<List<string>> GetAvailableEngines()
{
    dynamic oauth = await OAuthController.GetInternalAsync();
    List<string> allEngines = new List<string>();
    // define Engines API
    string paginationToken = null;
    while (true)
    {
        Page<string> engines = await _designAutomation.GetEnginesAsync(paginationToken);
        allEngines.AddRange(engines.Data);
        if (engines.PaginationToken == null)
            break;
        paginationToken = engines.PaginationToken;
    }
    allEngines.Sort();
    return allEngines; // return list of eng
}
```

Forge.DesignAutomation SDKで、APSのRest APIを叩いて利用可能なEngineを取得

実装～実行するAutomation APIのエンジンを選択できる



- client_idとclient_secretを、環境変数に設定
- client_idとclient_secretを指定してOAuthでTokenを取得
- APSのAPIを実行して、エンジンのリストを取得

▪DesignAutomationController.csを追加

Node.js & VSCode　.NET & VSCode　.NET & VS2022

Under `Controllers` folder create a `DesignAutomationController.cs` with the following content. This is just the class, we'll define the endpoints later, but note the DesignAutomationHub at the end, which allow us push notifications to the client via SignalR.

DesignAutomationController

```
using Autodesk.Forge;
using Autodesk.Forge.Client;
using Autodesk.Forge.DesignAutomation;
using Autodesk.Forge.DesignAutomation.Model;
using Autodesk.Forge.Model;
using Microsoft.AspNetCore.Hosting;
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.SignalR;
using Newtonsoft.Json;
```

ApsDesignAutomation.js

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
});

function prepareLists() {
  list("engines", "/api/aps/designautomation/engines");
}
```

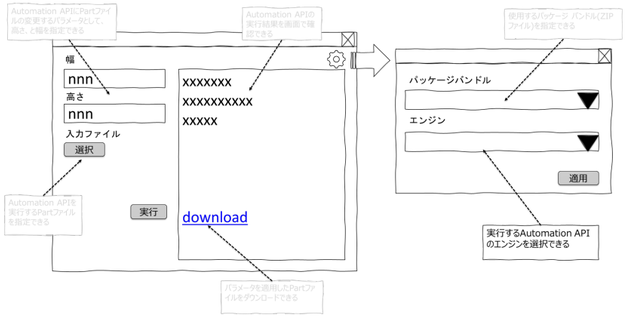
- GetAvailableEngines

To define a bundle we also nee
engines.

```
/// <summary>
/// Return a list of available engines
/// </summary>
[HttpGet]
[Route("api/aps/designautomation/engines")]
public async Task<List<string>> GetAvailableEngines()
{
    dynamic oauth = await OAuthController.GetInternalAsync();
    List<string> allEngines = new List<string>();
    // define Engines API
    string paginationToken = null;
    while (true)
    {
        Page<string> engines = await _designAutomation.GetEnginesAsync(paginationToken);
        allEngines.AddRange(engines.Data);
        if (engines.PaginationToken == null)
            break;
        paginationToken = engines.PaginationToken;
    }
    allEngines.Sort();
    return allEngines; // return list of eng
}
```

Forge.DesignAutomation SDKで、APSのRest APIを叩いて利用可能なEngineを取得

実装～実行するAutomation APIのエンジンを選択できる



▪Program.csとStartup.csの実装

Now open the Program.cs and add the following namespaces

```
using Autodesk.Forge.Core;
using Autodesk.Forge.DesignAutomation;
```

Then replace the `Program.cs` content with the following. This tells our application to load Forge Client ID & Secret from the environment variables defined above.

```
using Autodesk.Forge.Core;
using Autodesk.Forge.DesignAutomation;
using Microsoft.AspNetCore;

namespace designAutomationSample
{
  public class Program
  {
    public static void Main(string[] args)
    {
      CreateHostBuilder(args).ConfigureAppConfiguration(builder =>
      {
        builder.AddJsonFile($"appsettings.user.json", optional: true);
```

Forge.DesignAutomation SDKが、ディベロッパーキーを環境変数から取得できるようにロードする処理

Now open the Startup.cs (create it if needed) and add the following namespace

```
using Microsoft.AspNetCore.Mvc;
```

Then replace the content of the Startup class with the following code, which enables static file server (HTML & JS) and SignalR, used to push notifications to the client.

```
// This method gets called by the runtime. Use this method to add services to the container.
// For more information on how to configure your application, visit https://go.microsoft.com/fw
public void ConfigureServices(IServiceCollection services)
{
```

Webサーバがエンドポイントを公開するために、MVC等のサービスの有効化、エンドポイントの追加等を行う処理を記述

実装～実行するAutomation APIのエンジンを選択できる

- Program.cs

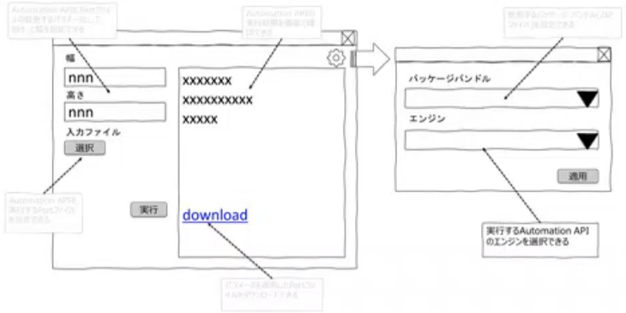
```
using Autodesk.Forge.Core;
using Autodesk.Forge.DesignAutomation;
using Microsoft.AspNetCore;


namespace designAutomationSample
{
    public class Program
    {
        public static void Main(string[] args)
        {
            CreateHostBuilder(args).ConfigureAppConfiguration(builder =>
            {
                builder.AddJsonFile($"appsettings.user.json", optional: true);
                builder.AddEnvironmentVariables();

            }).ConfigureServices((hostContext, services) =>
            {
                services.AddDesignAutomation(hostContext.Configuration);
            }).Build().Run();
        }

        public static IWebHostBuilder CreateHostBuilder(string[] args) =>
            WebHost.CreateDefaultBuilder(args)
                .UseStartup<Startup>();
    }
}
```

実装～実行するAutomation APIのエンジンを選択できる

- Startup.cs

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.DependencyInjection;

namespace designAutomationSample
{
    internal class Startup
    {
        // This method gets called by the runtime. Use this method to add services to the container.
        // For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddMvc(options => options.EnableEndpointRouting =
false).SetCompatibilityVersion(CompatibilityVersion.Version_3_0).AddNewtonsoftJson();
            services.AddSignalR().AddNewtonsoftJsonProtocol(opt => {
                opt.PayloadSerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;
            });
        }
        // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.UseFileServer();
            app.UseMvc();
            app.UseRouting();
            app.UseEndpoints(routes =>
            {
                routes.MapHub<Controllers.DesignAutomationHub>("/api/signalr/designautomation");
            });
        }
    }
}
```

実装～実行するAutomation APIのエンジンを選択できる



▪Program.csとStartup.csの実装

Now open the Program.cs and add the following namespaces

```
using Autodesk.Forge.Core;
using Autodesk.Forge.DesignAutomation;
```

Then replace the `Program.cs` content with the following. This tells our application to load Forge Client ID & Secret from the environment variables defined above.

```
using Autodesk.Forge.Core;
using Autodesk.Forge.DesignAutomation;
using Microsoft.AspNetCore;

namespace designAutomationSample
{
  public class Program
  {
    public static void Main(string[] args)
    {
      CreateHostBuilder(args).ConfigureAppConfiguration(builder =>
      {
        builder.AddJsonFile($"appsettings.user.json", optional: true);
```

Forge.DesignAutomation SDKが、ディベロッパーキーを環境変数から取得できるようにロードする処理

Now open the Startup.cs (create it if needed) and add the following namespace

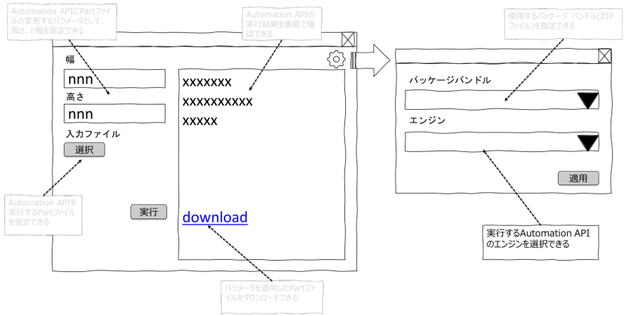
```
using Microsoft.AspNetCore.Mvc;
```

Then replace the content of the Startup class with the following code, which enables static file server (HTML & JS) and SignalR, used to push notifications to the client.

```
// This method gets called by the runtime. Use this method to add services to the container.
// For more information on how to configure your application, visit https://go.microsoft.com/fw
public void ConfigureServices(IServiceCollection services)
{
```

Webサーバがエンドポイントを公開するために、MVC等のサービスの有効化、エンドポイントの追加等を行う処理を記述

機能設計～実行するAutomation APIのエンジンを選択できる



クライアント(JS)

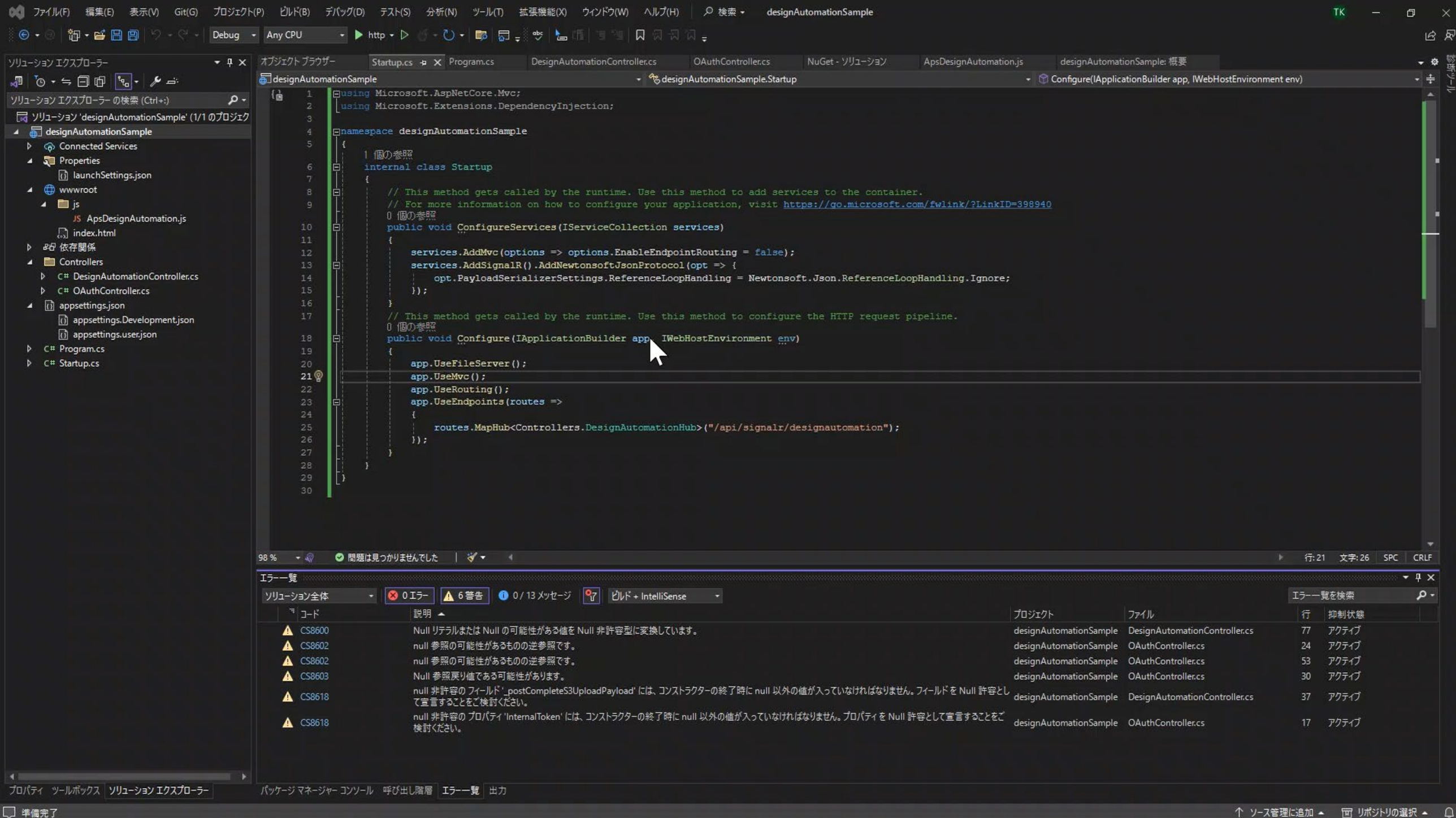
- ~~モーダルダイアログの表示~~
- ~~エンジンのリストをリクエスト～結果を選択リストに格納~~

Webサーバ

- **~~client_idとclient_secretを、環境変数に設定~~**
- **~~client_idとclient_secretを指定してOAuthでTokenを取得~~**
- **~~APSのAPIを実行して、エンジンのリストを取得~~**

APS（API）

- ~~OAuthでTokenを取得するAPI~~
- ~~エンジンのリストを取得するAPI~~


designAutomationSample
Startup.cs
Program.cs
DesignAutomationController.cs
OAuthController.cs
ApsDesignAutomation.js
designAutomationSample.Startup
Configure(IApplicationBuilder app, IWebHostEnvironment env)
using Microsoft.AspNetCore.Mvc;
using Microsoft.Extensions.DependencyInjection;
namespace designAutomationSample
{
internal class Startup
{
// This method gets called by the runtime. Use this method to add services to the container.
// For more information on how to configure your application, visit https://go.microsoft.com/fwlink/?LinkID=398940
public void ConfigureServices(IServiceCollection services)
{
services.AddMvc(options => options.EnableEndpointRouting = false);
services.AddSignalR().AddNewtonsoftJsonProtocol(opt => {
opt.PayloadSerializerSettings.ReferenceLoopHandling = Newtonsoft.Json.ReferenceLoopHandling.Ignore;
});
}
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
app.UseFileServer();
app.UseMvc();
app.UseRouting();
app.UseEndpoints(routes =>
{
routes.MapHub<Controllers.DesignAutomationHub>("/api/signalr/designautomation");
});
}
}
ソリューション エクスプローラー
ソリューション 'designAutomationSample' (1/1 のプロジェクト)
designAutomationSample
Connected Services
Properties
launchSettings.json
wwwroot
js
ApsDesignAutomation.js
index.html
依存関係
Controllers
DesignAutomationController.cs
OAuthController.cs
appsettings.json
appsettings.Development.json
appsettings.user.json
Program.cs
Startup.cs
エラー一覧
0 エラー
6 警告
0 / 13 メッセージ
コード
説明
プロジェクト
ファイル
行
抑制状態
CS8600 Null リテラルまたは Null の可能性がある値を Null 非許容型に変換しています。 designAutomationSample DesignAutomationController.cs 77 アクティブ
CS8602 null 参照の可能性があるものの逆参照です。 designAutomationSample OAuthController.cs 24 アクティブ
CS8602 null 参照の可能性があるものの逆参照です。 designAutomationSample OAuthController.cs 53 アクティブ
CS8603 Null 参照戻り値である可能性があります。 designAutomationSample OAuthController.cs 30 アクティブ
CS8618 null 非許容の フィールド '_postCompleteS3UploadPayload' には、コンストラクターの終了時に null 以外の値が入っていなければなりません。フィールドを Null 許容として宣言することをご検討ください。 designAutomationSample DesignAutomationController.cs 37 アクティブ
CS8618 null 非許容の プロパティ 'InternalToken' には、コンストラクターの終了時に null 以外の値が入っていなければなりません。プロパティを Null 許容として宣言することをご検討ください。 designAutomationSample OAuthController.cs 17 アクティブ
準備完了

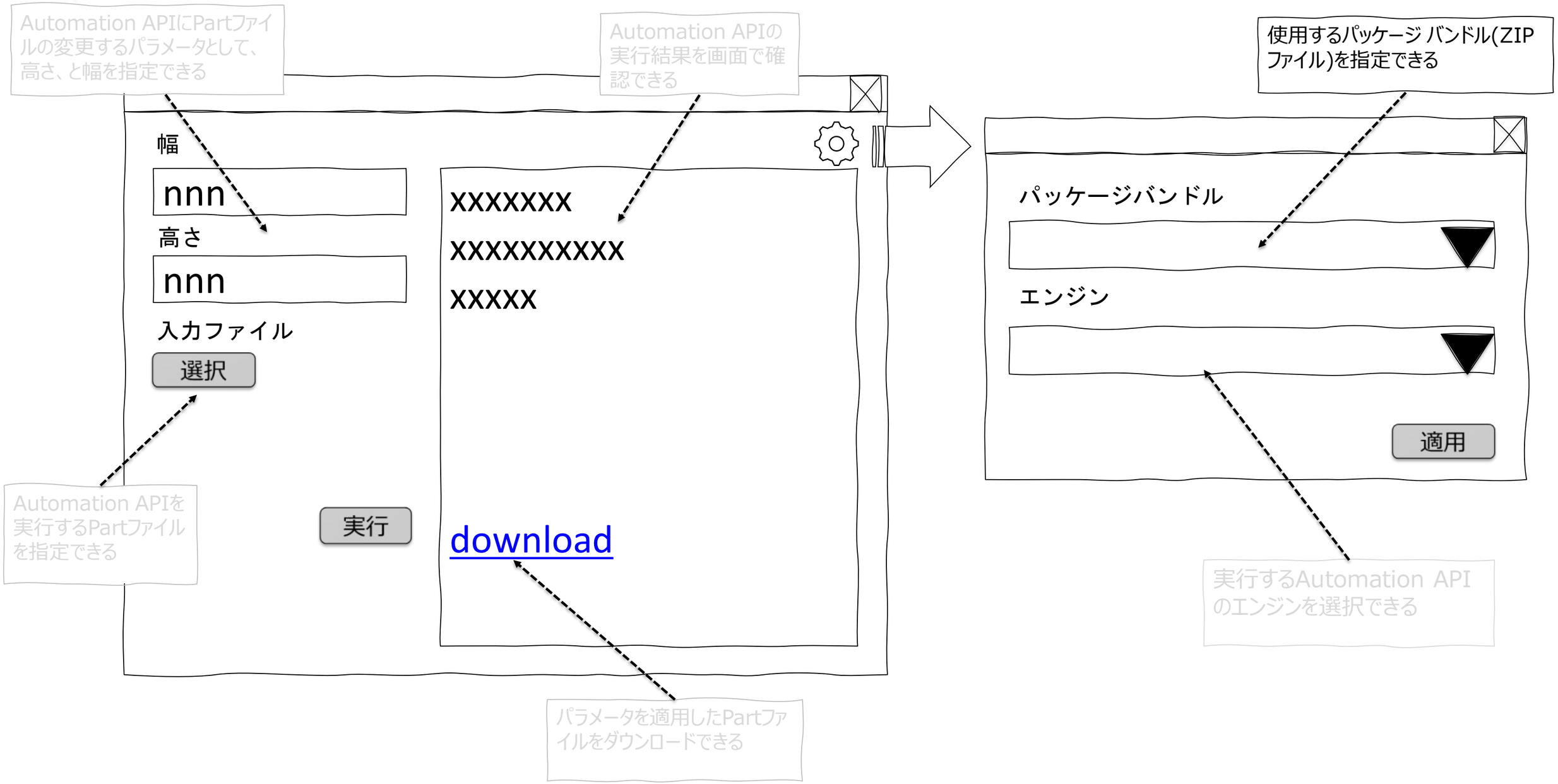
サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- ~~実行するAutomation APIのエンジンを選択できる~~
- 使用するパッケージ バンドル(ZIPファイル)を指定できる
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

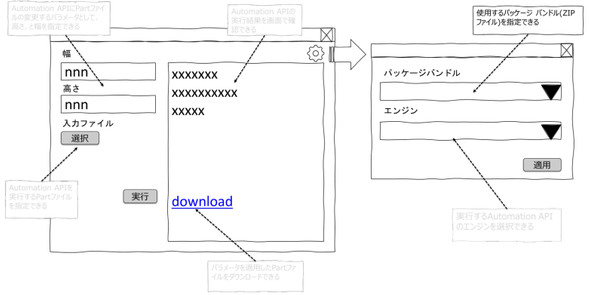
サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- ~~実行するAutomation APIのエンジンを選択できる~~
- 使用するパッケージ バンドル(ZIPファイル)を指定できる
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

機能設計～使用するパッケージ バンドル(ZIPファイル)を指定できる



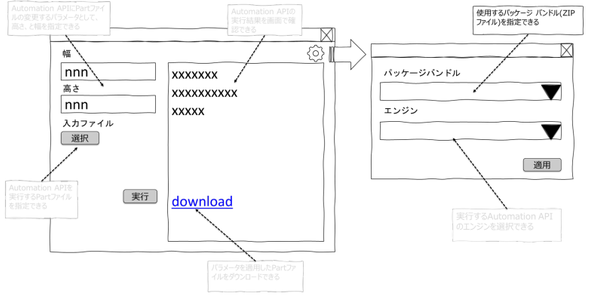
機能設計～使用するパッケージ バンドル(ZIPファイル)を指定できる



- 使用するパッケージ バンドル(ZIPファイル)を指定できる

クライアント(JS)	Webサーバ	APS（API）
▪ パッケージバンドルをリクエスト～結果を選択リストに格納	▪ wwwroot/bundlesフォルダ配下のzipファイルのリストを取得	

機能設計～使用するパッケージ バンドル(ZIPファイル)を指定できる



- 使用するパッケージ バンドル(ZIPファイル)を指定できる

▪ApsDesignAutomation.jsとDesignAutomationController.csに処理を追加

DesignAutomationController.cs

- GetLocalBundles

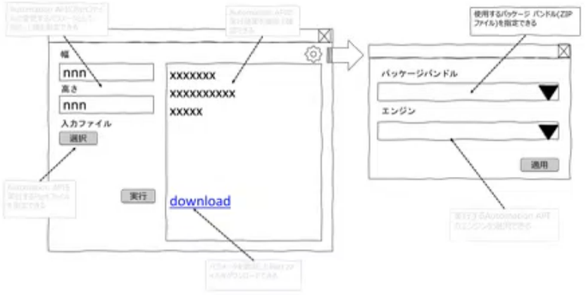
Look at the `bundles` folder and return a list of .ZIP files.

```
/// <summary>
/// Names of app bundles on this project
/// </summary>
[HttpGet]
[Route("api/appbundles")]
public string[] GetLocalBundles()
{
   // this folder is placed under the public folder, which may expose the bundles
   // but it was defined this way so it be published on most hosts easily
   return Directory.GetFiles(LocalBundlesFolder, "*.zip").Select(Path.GetFileNameWithoutExtension).ToArray();
}
```

ApsDesignAutomation.js

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
});

function prepareLists() {
  list("engines", "/api/aps/designautomation/engines");
  list("localBundles", "/api/appbundles");
}
```

機能設計～使用するパッケージ バンドル(ZIPファイル)を指定できる



- 使用するパッケージ バンドル(ZIPファイル)を指定できる

▪ApsDesignAutomation.jsとDesignAutomationController.csに処理を追加

DesignAutomationController.cs

- GetLocalBundles

Look at the `bundles` folder and return a list of .ZIP files.

```
/// <summary>
/// Names of app bundles on this project
/// </summary>
[HttpGet]
[Route("api/appbundles")]
public string[] GetLocalBundles()
{
   // this folder is placed under the public folder, which may expose the bundles
   // but it was defined this way so it be published on most hosts easily
   return Directory.GetFiles(LocalBundlesFolder, "*.zip").Select(Path.GetFileNameWithoutExtension).ToArray();
}
```

ApsDesignAutomation.js

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
});

function prepareLists() {
  list("engines", "/api/aps/designautomation/engines");
  list("localBundles", "/api/appbundles");
}
```

サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- ~~実行するAutomation APIのエンジンを選択できる~~
- ~~使用するパッケージ バンドル(ZIPファイル)を指定できる~~
- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- ~~実行するAutomation APIのエンジンを選択できる~~
- ~~使用するパッケージ バンドル(ZIPファイル)を指定できる~~

⋮ Automation APIの仕組み上必要な機能がある？

- Automation APIを実行するPartファイルを指定できる
- Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる
- Automation APIの実行結果を画面で確認できる
- パラメータを適用したPartファイルをダウンロードできる

Automation API

実行する処理の定義（ひな型：カスタム処理の内容、入出力）を登録し、具体的な入出力の値・ファイルを指定して、実行する

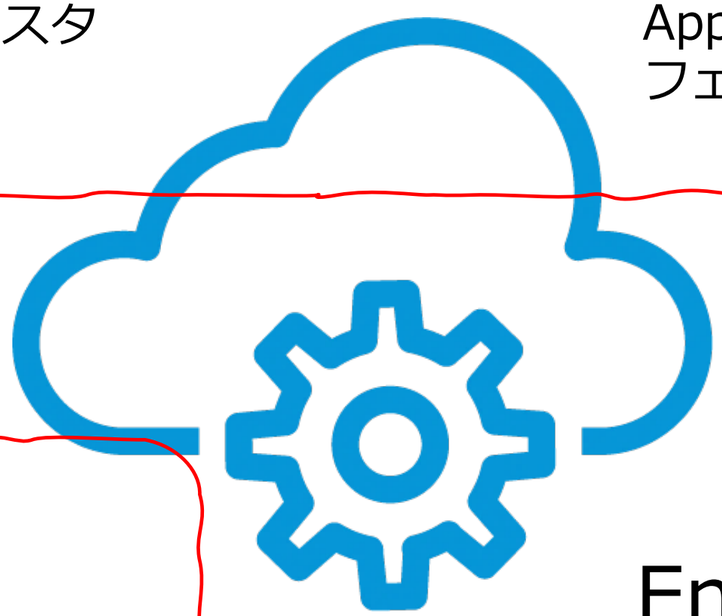
定義

AppBundle

各CADのAPIを使用したカスタム処理

Activity

AppBundleとWorkItemのインターフェース



インスタンス

WorkItem

Inventor Serverを実行するJob

Engine

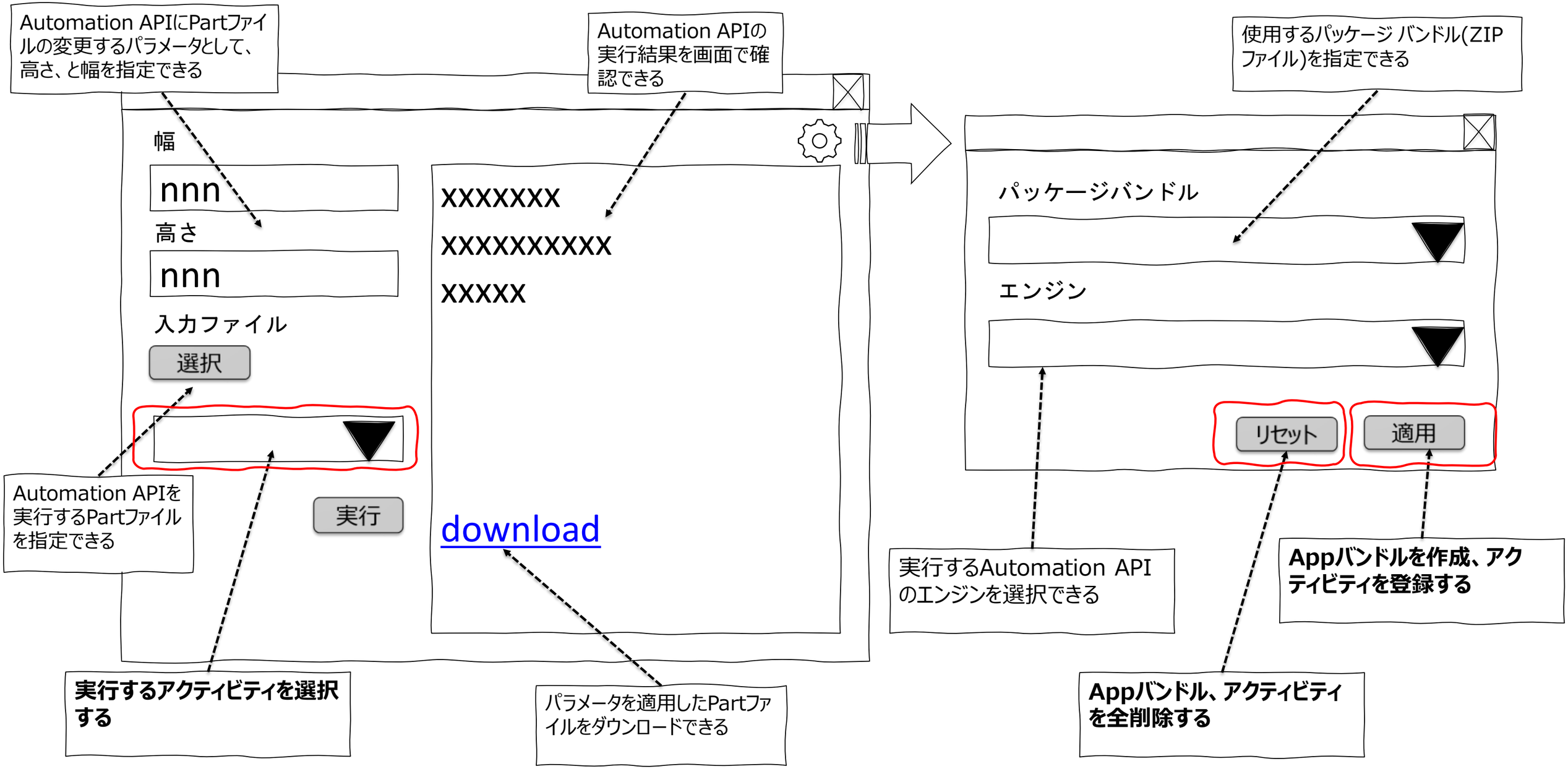
各CADツールに対応する複数バージョンのエンジン

AUTOCAD® REVIT®

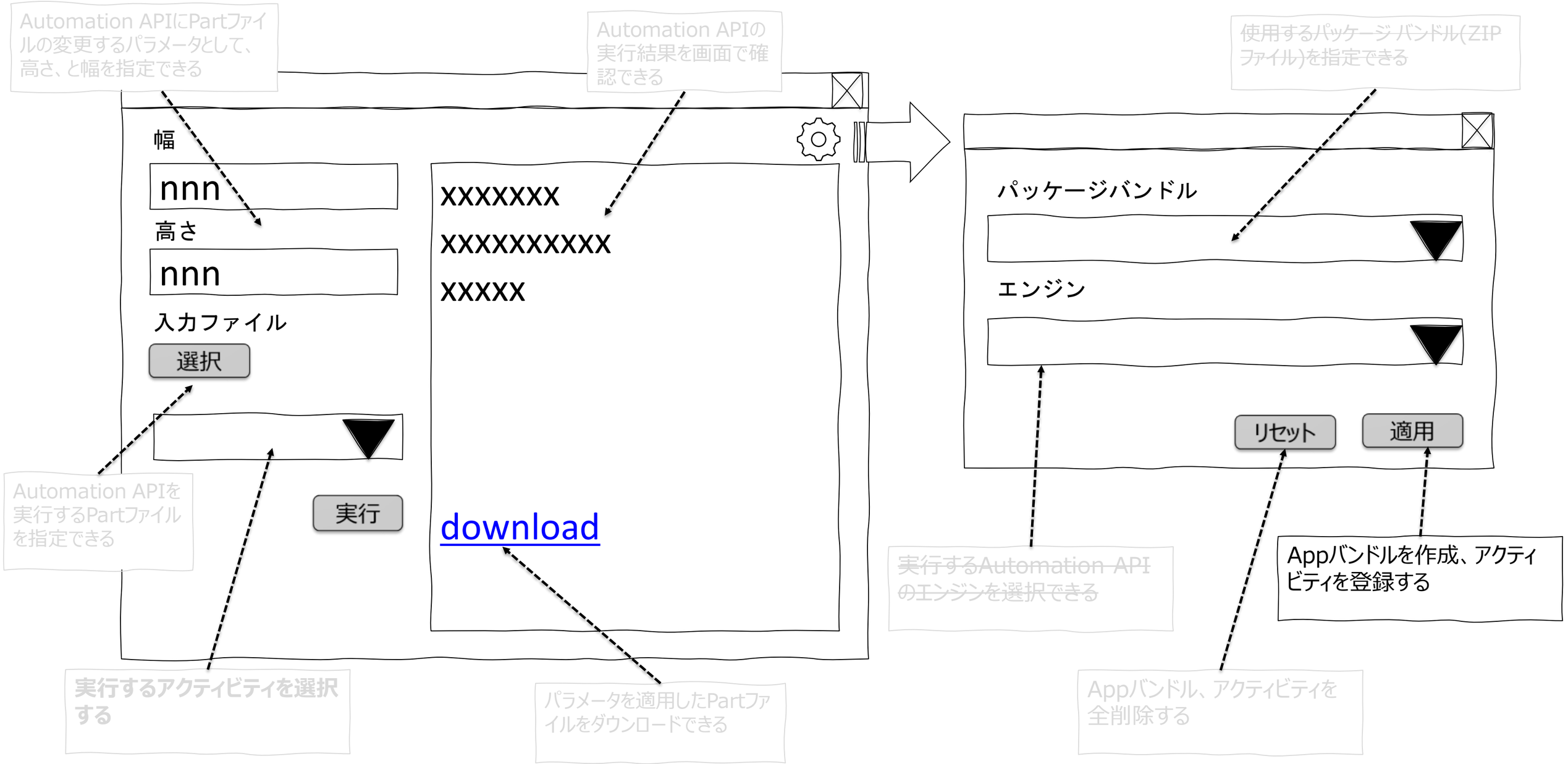
INVENTOR® 3DS MAX®

Fusion

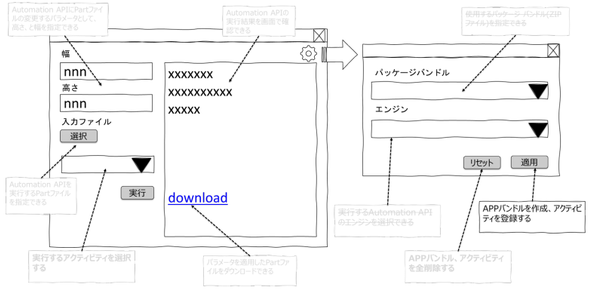
サンプルアプリケーションのGUIと機能



サンプルアプリケーションのGUIと機能

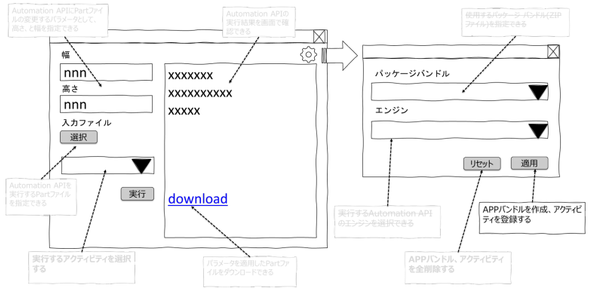


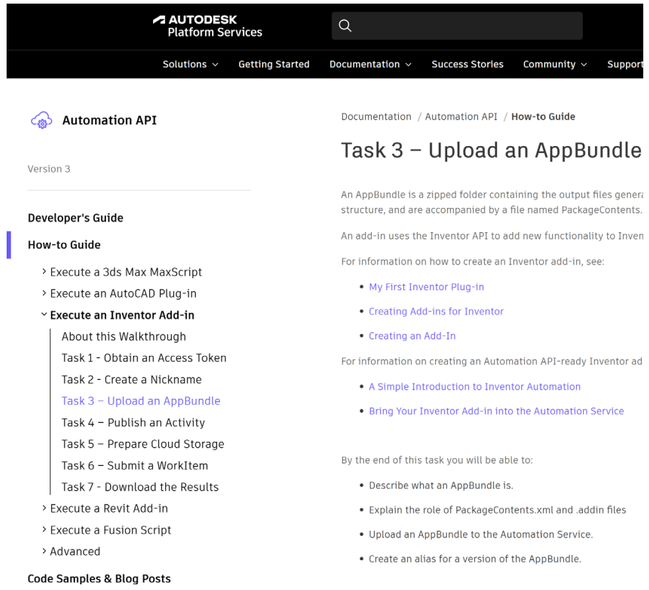
機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)	Webサーバ	APS（API）
▪ 選択したエンジンとZIPファイルでAppバンドルの作成を要求 ▪ 作成したAPPバンドルでアクティビティの作成を要求	▪ Appバンドルを作成 ▪ アクティビティを作成	▪ ？？？？

機能設計～Appバンドルを作成、アクティビティを登録する





You will use the following operations to handle AppBundles in this task:

HTTP Request	Description
POST /appbundles	Registers a new AppBundle.
POST /appbundles/{id}/aliases	Creates a new alias for the AppBundle.
POST /appbundles/{id}/versions	Creates a new version of the AppBundle.
PATCH /appbundles/{id}/aliases/{aliasId}	Modify alias details.

- More information on AppBundles can be found here.

Step 1 – Download or build an Inventor AppBundle

利用手順概略

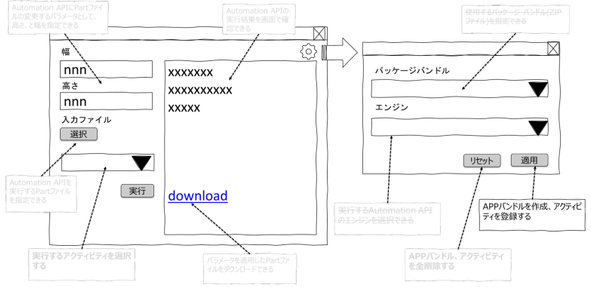
１．AppBundle名とエンジン名を指定してAppバンドルを作成

２．作成したAppバンドルに、パッケージ バンドル(ZIPファイル)をアップロード（作成時に戻り値で取得したURLを使用）

３．エイリアスを作成

参考リファレンス：Documentation /Automation API /How-to Guide Task 3 – Upload an AppBundle to the Automation Service

機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)

- 選択したエンジンとZIPファイルでAPPバンドルの作成を要求
- 作成したAPPバンドルでアクティビティの作成を要求

Webサーバ

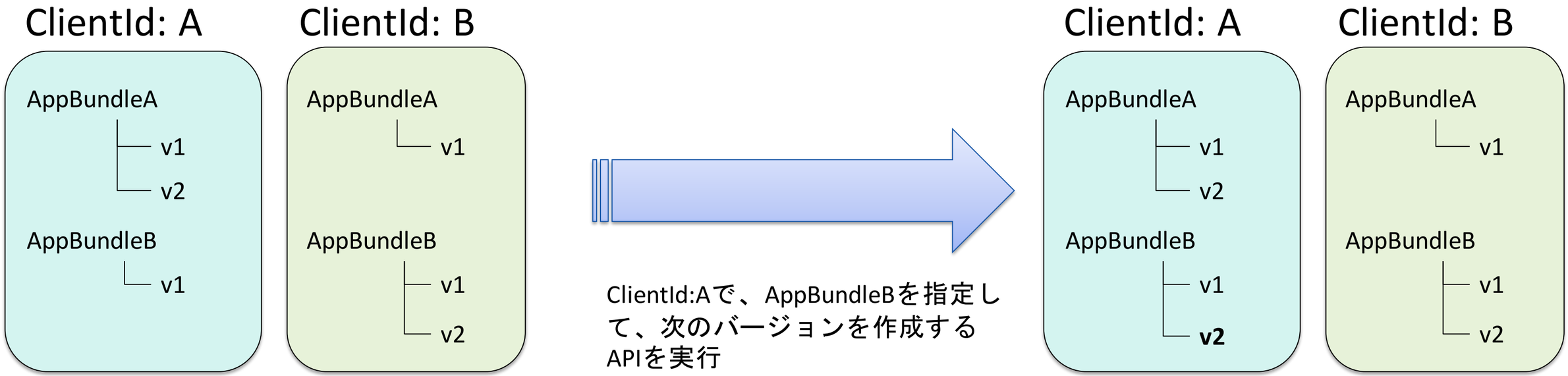
- Appバンドルを作成
  - APPバンドルを作成
  - ZIPファイルをアップロード
  - エイリアスを作成
- アクティビティを作成

APS（API）

- Appバンドルを作成するAPI
- エイリアスを作成するAPI

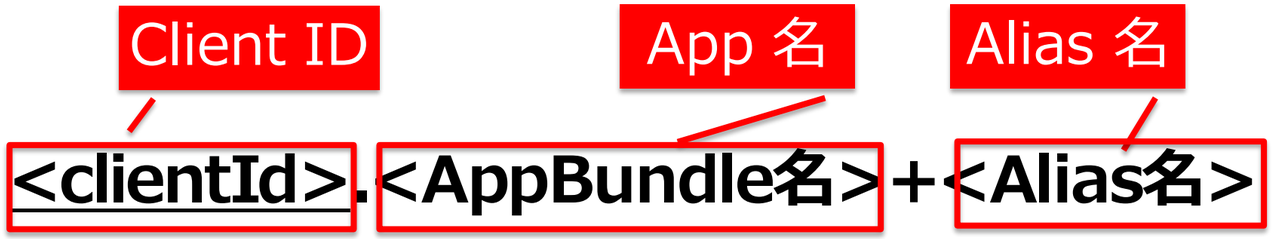
AppBundle名とバージョンについて

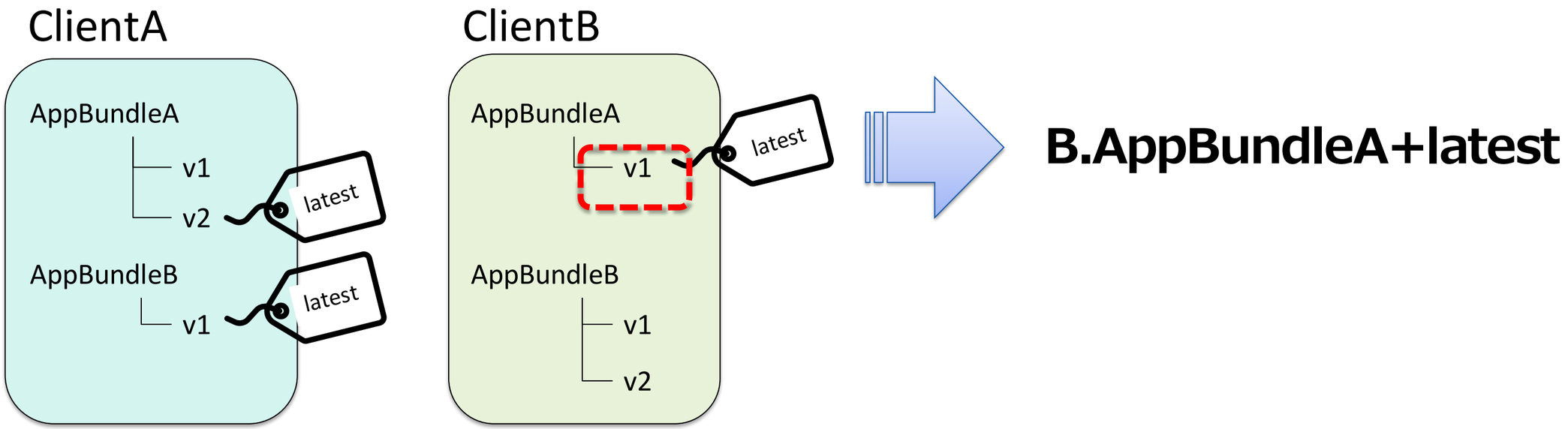
- AppBundle新規作成時には、指定したAppBundle名でv1が作成される。
- 同一名のAppBundleがある場合はバージョンを積むAPIを実行して次のバージョンを作成する（指定したエンジンによらず、 AppBundle名で識別される）
- Appバンドル名の有効範囲は、APIを実行するClientId内



AppBundleのバージョンとエイリアスについて

- AppBundleの各バージョンには、バージョンを識別するためのエイリアス（任意の名前：タグのような物）を付加できる
- エイリアスの付与は任意
- Activityから、使用するAppBundleを指定する場合は、以下の情報を使用
  - ClientID（またはNickName）
  - AppBundle名
  - Alias





ClientID と Nickname（ニックネーム）

- ClientIdに対してNicknameのマッピング登録が可能（**任意**）
  - 例）

    CleintId“nqpwqsDLFGkSO6LgA2mvaSXy5AeH5VSJ“ の NickName“Inventor_Tutorial”を登録

    “nqpwqsDLFGkSO6LgA2mvaSXy5AeH5VSJ**.UpdateIPTParam+dev**“

    “Inventor_Tutorial**.UpdateIPTParam+dev**“

  - 注意：Nickname 登録後 Client ID 名指定は使用不可

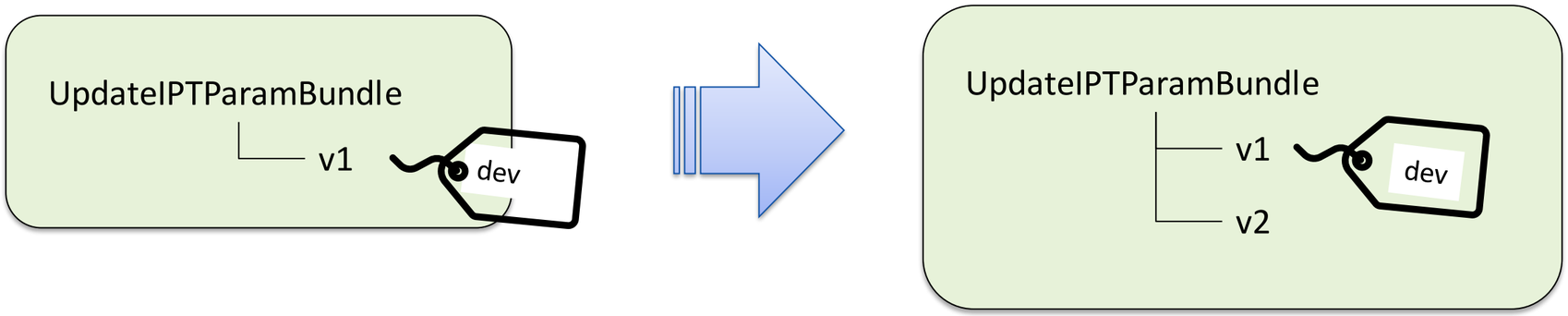
機能設計～Appバンドルを作成、アクティビティを登録する

- Appバンドル名：指定した“Zipファイル名+AppBundle“
- Alias名：dev

AppBundle登録時の動作

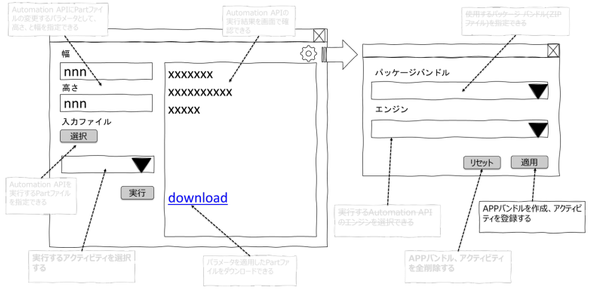
1．ClientId. Appバンドル名+devが存在しない場合
   - Appバンドルを新規登録（V1を作成）
   - V1にAlias：devを付与

2．ClientId. Appバンドル名+devが存在する場合
   - Appバンドルの次のバージョンを作成（今の最新+1を作成）
   - 作成したバージョンにAlias：devを付け替え



3．zipパッケージバンドルファイルをアップロード

機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)

- 選択したエンジンとZIPファイルでAPPバンドルの作成を要求
- 作成したAPPバンドルでアクティビティの作成を要求

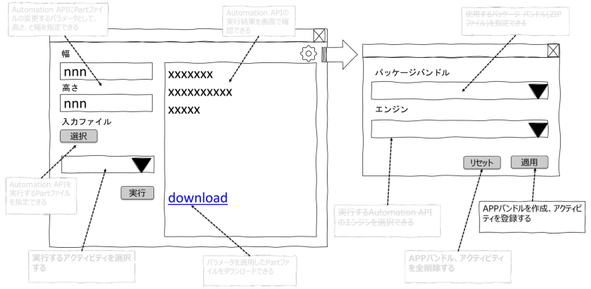
Webサーバ

- **Appバンドルを作成or新規バージョンを作成して、エイリアスを付与**
- アクティビティを作成

APS（API）

- **既存のAppバンドルを確認するAPI**
- Appバンドルを作成するAPI
- エイリアスを作成するAPI
- **APPバンドルのバージョンを作成するAPI**
- **エイリアスを付け替えるAPI**

機能設計～Appバンドルを作成、アクティビティを登録する



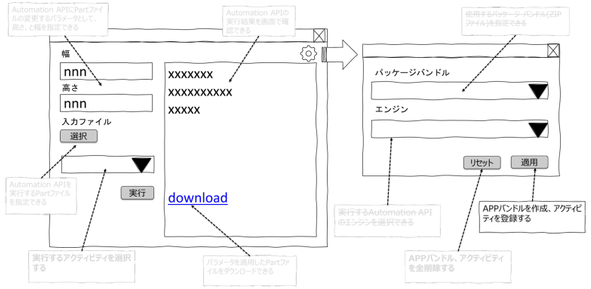
- 選択したエンジンとZIPファイルでAPPバンドルの作成を要求
- 作成したAPPバンドルでアクティビティの作成を要求

▪ApsDesignAutomation.jsの処理

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
  $("#createAppBundleActivity").click(createAppBundleActivity);
});

function createAppBundleActivity() {
  startConnection(function () {
    writeLog("Defining appbundle and activity for " +
$("#engines").val());
    $("#defineActivityModal").modal("toggle");
    createAppBundle(function () {
      createActivity(function () {
        prepareLists();
      });
    });
  });
}
```

```
function createAppBundle(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/appbundles",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
    zipFileName: $("#localBundles").val(),
    engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("AppBundle: " + res.appBundle + ", v" + res.version);
      if (cb) cb();
    },
  });
}
```

機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)

- ~~**選択したエンジンとZIPファイルでAPPバンドルの作成を要求**~~
- 作成したAPPバンドルでアクティビティの作成を要求

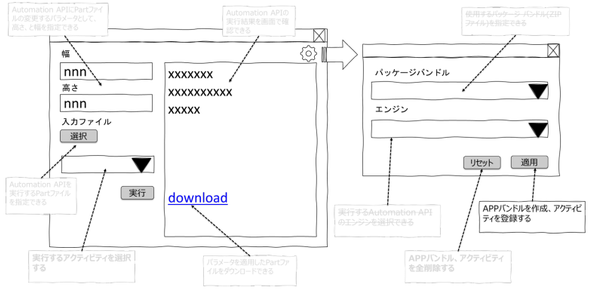
Webサーバ

- Appバンドルを作成or新規バージョンを作成して、エイリアスを付与
- アクティビティを作成

APS（API）

- 既存のAppバンドルを確認するAPI
- Appバンドルを作成するAPI
- エイリアスを作成するAPI
- APPバンドルのバージョンを作成するAPI
- エイリアスを付け替えるAPI

機能設計～Appバンドルを作成、アクティビティを登録する



- Appバンドルを作成or新規バージョンを作成して、エイリアスを付与
- アクティビティを作成

▪DesignAutomationController.jsの処理

- CreateAppBundle

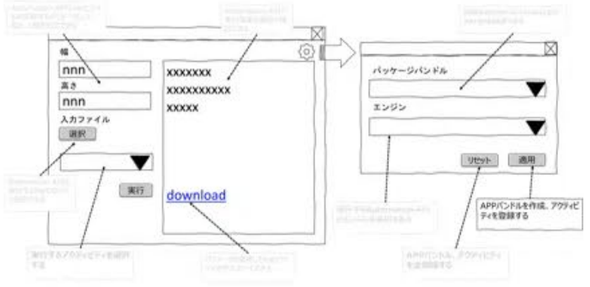
That's where we actually define a new AppBundle:

```
/// <summary>
/// Define a new appbundle
/// </summary>
[HttpPost]
[Route("api/aps/designautomation/appbundles")]
public async Task<IActionResult> CreateAppBundle([FromBody]JObject appBundleSpecs)
{
  // basic input validation
  string zipFileName = appBundleSpecs["zipFileName"].Value<string>();
  string engineName = appBundleSpecs["engine"].Value<string>();

  // standard name for this sample
  string appBundleName = zipFileName + "AppBundle";

  // check if ZIP with bundle is here
  string packageZipPath = Path.Combine(LocalBundlesFolder, zipFileName + ".zip");
  if (!System.IO.File.Exists(packageZipPath)) throw new Exception("Appbundle not found at " + packageZipPath);

  // get defined app bundles
  Page<string> appBundles = await _designAutomation.GetAppBundlesAsync();

  // check if app bundle is already define
  dynamic newAppVersion;
  string qualifiedAppBundleId = string.Format("{0}.{1}+{2}", NickName, appBundleName, Alias);
```

ApsDesignAutomation.js

```
function createAppBundle(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/appbundles",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
    zipFileName: $("#localBundles").val(),
    engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("AppBundle: " + res.appBundle + ", v" + res.version);
      if (cb) cb();
    },
  });
}
```

機能設計～Appバンドルを作成、アクティビティを登録する



- Appバンドルを作成or新規バージョンを作成して、エイリアスを付与
- アクティビティを作成

▪DesignAutomationController.jsの処理

- CreateAppBundle

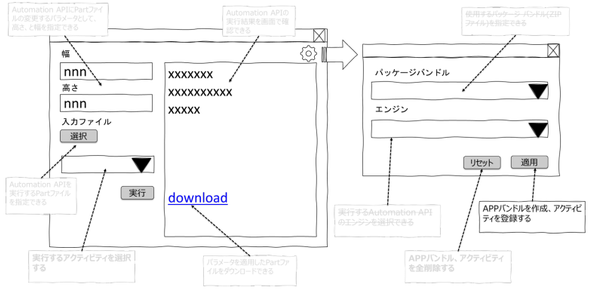
That's where we actually define a new AppBundle:

```
/// <summary>
/// Define a new appbundle
/// </summary>
[HttpPost]
[Route("api/aps/designautomation/appbundles")]
public async Task<IActionResult> CreateAppBundle([FromBody]JObject appBundleSpecs)
{
  // basic input validation
  string zipFileName = appBundleSpecs["zipFileName"].Value<string>();
  string engineName = appBundleSpecs["engine"].Value<string>();

  // standard name for this sample
  string appBundleName = zipFileName + "AppBundle";

  // check if ZIP with bundle is here
  string packageZipPath = Path.Combine(LocalBundlesFolder, zipFileName + ".zip");
  if (!System.IO.File.Exists(packageZipPath)) throw new Exception("Appbundle not found at " + packageZipPath);

  // get defined app bundles
  Page<string> appBundles = await _designAutomation.GetAppBundlesAsync();

  // check if app bundle is already define
  dynamic newAppVersion;
  string qualifiedAppBundleId = string.Format("{0}.{1}+{2}", NickName, appBundleName, Alias);
```

ApsDesignAutomation.js

```
function createAppBundle(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/appbundles",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
    zipFileName: $("#localBundles").val(),
    engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("AppBundle: " + res.appBundle + ", v" + res.version);
      if (cb) cb();
    },
  });
}
```

機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)

- ~~選択したエンジンとZIPファイルでAPPバンドルの作成を要求~~
- 作成したAPPバンドルでアクティビティの作成を要求

Webサーバ

- **~~Appバンドルを作成or新規バージョンを作成して、エイリアスを付与~~**
- アクティビティを作成

APS（API）

- **~~既存のAppバンドルを確認するAPI~~**
- **~~Appバンドルを作成するAPI~~**
- **~~エイリアスを作成するAPI~~**
- **~~APPバンドルのバージョンを作成するAPI~~**
- **~~エイリアスを付け替えるAPI~~**

Activityとは

- Automation 処理の入力と出力の定義
  - 入力
    - CADファイル
    - パラメータ
    - スクリプト
  - 出力
    - 処理結果のファイル
  - 利用するコアエンジンと実行引数（コマンドライン文字列）定義
    - どのAppBundleを利用するのか
    - 入力CADファイルは？

Inventorコアエンジンのオプション

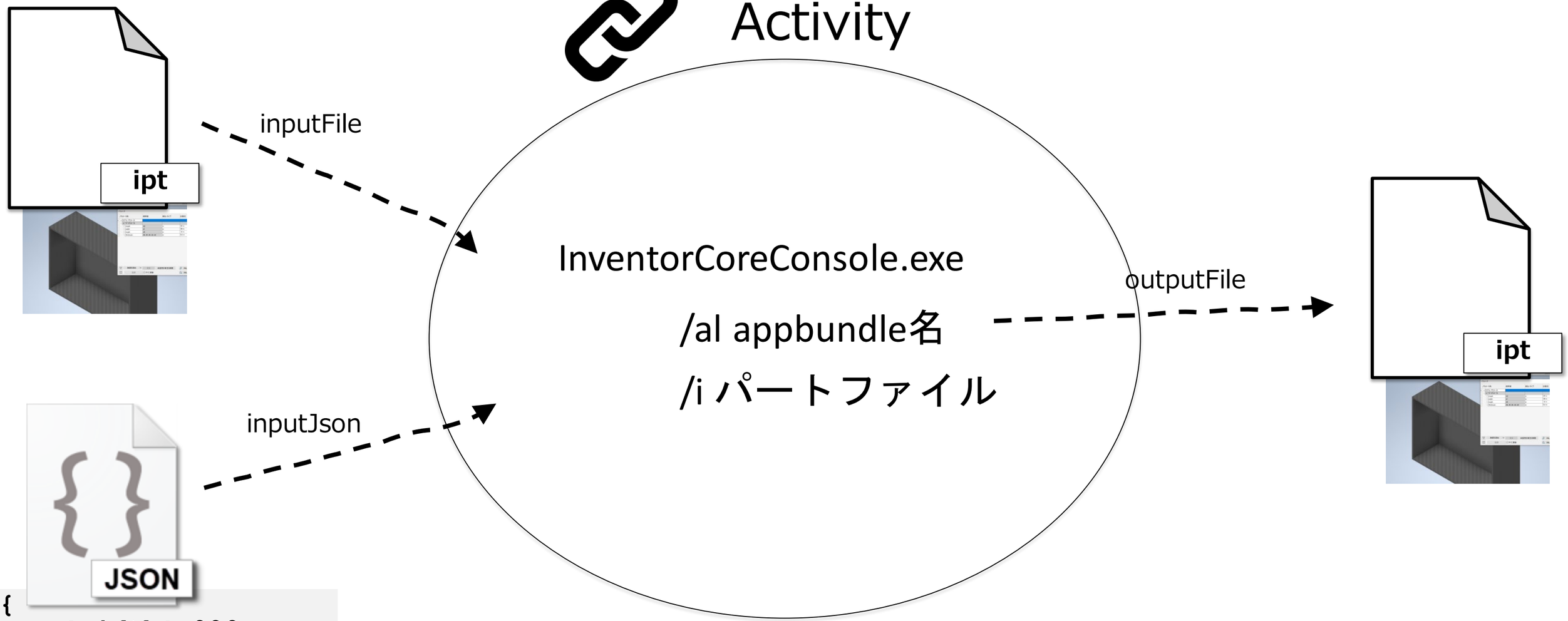
InventorCoreConsole.exe

- ✓ Design Automation for Inventor用のコアエンジン
- ✓ クライアント版Inventorには同梱されていない
- ✓ 起動引数で指定されたAppBundleをロードし、AppBundleのメソッド（のどちらか）を実行
  - Run(Document doc) ：所定起動引数のみ場合
  - RunWithArguments(Document doc, NameValueMap map)： 所定引数以外を指定した場合。所定以外の引数はNameValueMapにキー“-1”, “-2”…で順に格納される
- ✓ InventorCoreConsole.exeの所定起動引数
  - /al：ロードするAppBundleのパス
  - /i：ロードするドキュメントのパス
  - /s：実行するスクリプトファイルへのパス

API Reference : Inventor Command Line Reference

機能設計～Appバンドルを作成、アクティビティを登録する

- 入出力




```
{
    'width':999,
    'height':555
}
```

機能設計～Appバンドルを作成、アクティビティを登録する

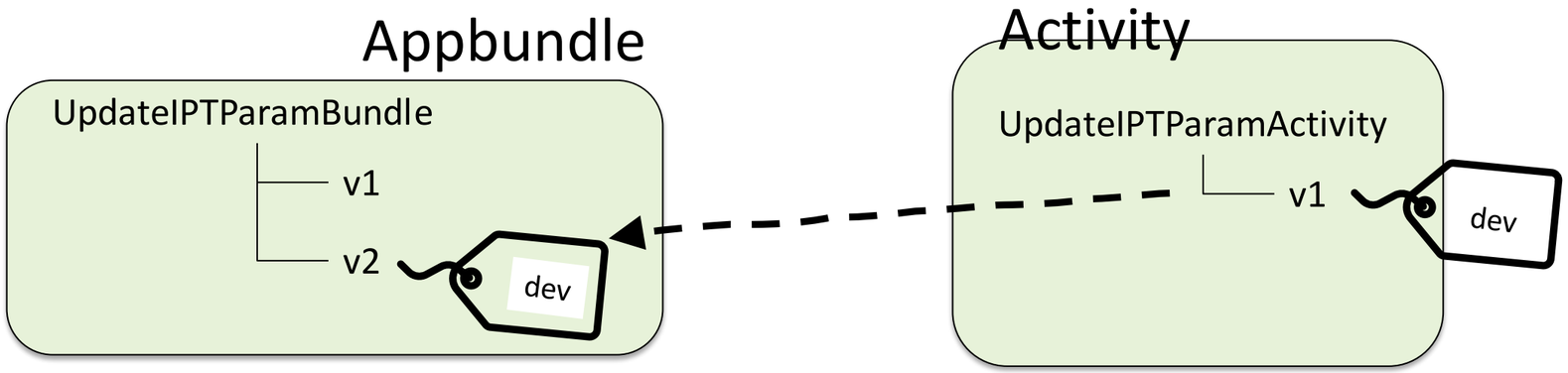
- Activity名：指定した“Zipファイル名+Activity”
- Alias名：dev

Activity登録時の動作

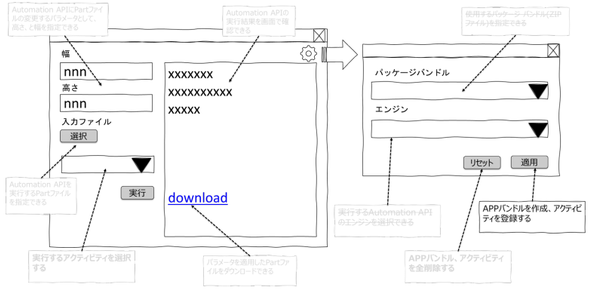
ClientId. Activity名+devが存在しない場合は以下を実行※存在する場合は処理をスキップ

- Activityを新規登録（V1を作成）
  - 利用するAppBundleには、**ClientId.Appbundle名+dev**を指定
  - 入力：CADファイル、パラメータ（JSON形式）
  - 出力：パラメータを適用した、CADファイル
- V1にAlias：devを付与

- Activityから見ると、常に最新のAppbundleを参照する形になる(Alias:devのAppbundleを指定しているため）



機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)

- ~~選択したエンジンとZIPファイルでAPPバンドルの作成を要求~~
- 作成したAPPバンドルでアクティビティの作成を要求

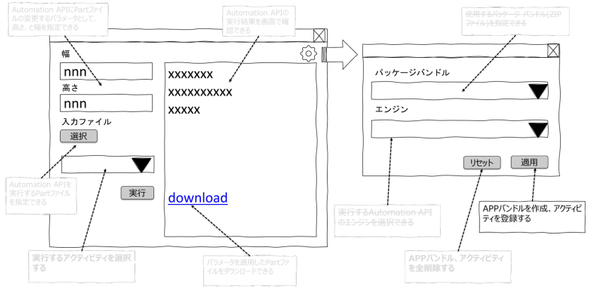
Webサーバ

- ~~Appバンドルを作成or新規バージョンを作成して、エイリアスを付与~~
- アクティビティを作成

APS（API）

- ~~既存のAppバンドルを確認するAPI~~
- ~~Appバンドルを作成するAPI~~
- ~~エイリアスを作成するAPI~~
- ~~APPバンドルのバージョンを作成するAPI~~
- ~~エイリアスを付け替えるAPI~~
- **既存のActivityを確認するAPI**
- **Activityを作成するAPI**

機能設計～Appバンドルを作成、アクティビティを登録する



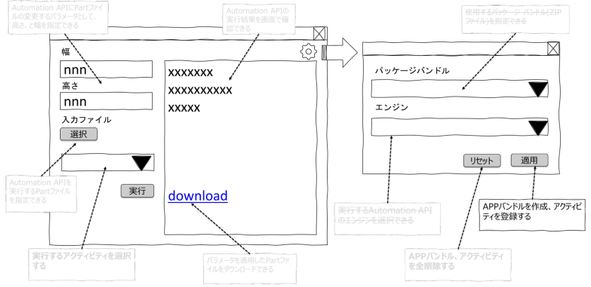
- ~~Appバンドルを作成or新規バージョンを作成して、エイリアスを付与~~
- アクティビティを作成

▪ApsDesignAutomation.jsの処理

```
$(document).ready(function () {
  prepareLists();
  $("#defineActivityShow").click(defineActivityModal);
  $("#createAppBundleActivity").click(createAppBundleActivity);
});

function createAppBundleActivity() {
  startConnection(function () {
    writeLog("Defining appbundle and activity for " +
$("#engines").val());
    $("#defineActivityModal").modal("toggle");
    createAppBundle(function () {
      createActivity(function () {
        prepareLists();
      });
    });
  });
}
```

```
function createActivity(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/activities",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
      zipFileName: $("#localBundles").val(),
      engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("Activity: " + res.activity);
      if (cb) cb();
    },
  });
}
```

機能設計～Appバンドルを作成、アクティビティを登録する



- Appバンドルを作成or新規バージョン
  を作成して、エイリアスを
- アクティビティを作成

- DesignAutomationController.csの処理

- EngineAttributes

To define the activity we'll need the executable and the default file extensi
function provides it (from the engine name).

```
/// <summary>
/// Helps identify the engine
/// </summary>
private dynamic EngineAttributes(string engine)
{
  if (engine.Contains("3dsMax")) return new { commandLine = "$(engine.pa
  if (engine.Contains("AutoCAD")) return new { commandLine = "$(engine.p
  if (engine.Contains("Inventor")) return new { commandLine = "$(engine.p
  if (engine.Contains("Revit")) return new { commandLine = "$(engine.path
  throw new Exception("Invalid engine");
}
```

- CreateActivity

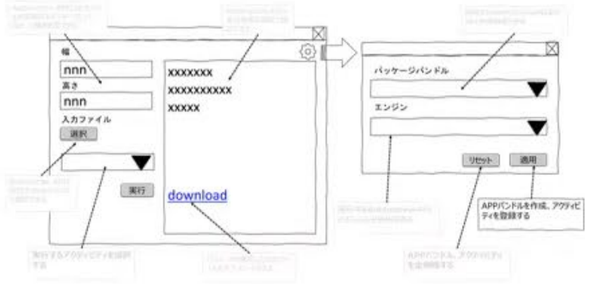
Define a new activity with an input file, input data (JSON) and an output file.

```
/// <summary>
/// Define a new activity
/// </summary>
[HttpPost]
[Route("api/aps/designautomation/activities")]
public async Task<IActionResult> CreateActivity([FromBody]JObject activitySpecs)
{
  // basic input validation
  string zipFileName = activitySpec  FileName"].Value<string>();
  string engineName = activitySpecs[  ne"].Val

  // standard name for this sample
  string appBundleName = zipFileName + "Appb
  string activityName = zipFileName + "Activity";

  //
  Page<string> activities = await _designAutoma
  string qualifiedActivityId = string.Format("{0}.{
  if (!activities.Data.Contains(qualifiedActivityId)
  {
    // define the activity
    // ToDo: parametrize for different engines...
    dynamic engineAttributes = EngineAttributes
    string commandLine = string.Format(engineA
    Activity activitySpec = new Activity()
    {
      Id = activityName,
      Appbundles = new List<string>() { string.Fo
      CommandLine = new List<string>() { comm
      Engine = engineName,
      Parameters = new Dictionary<string, Paramet
```

ApsDesignAutomation.js

```
function createActivity(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/activities",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
      zipFileName: $("#localBundles").val(),
      engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("Activity: " + res.activity);
      if (cb) cb();
    },
  });
}
```

機能設計～Appバンドルを作成、アクティビティを登録する



- Appバンドルを作成or新規バンドルを作成して、エイリアスを
- アクティビティを作成

- DesignAutomationController.csの処理

- EngineAttributes

To define the activity we'll need the executable and the default file extensi function provides it (from the engine name).

```
/// <summary>
/// Helps identify the engine
/// </summary>
private dynamic EngineAttributes(string engine)
{
  if (engine.Contains("3dsMax")) return new { commandLine = "$(engine.pa
  if (engine.Contains("AutoCAD")) return new { commandLine = "$(engine.p
  if (engine.Contains("Inventor")) return new { commandLine = "$(engine.p
  if (engine.Contains("Revit")) return new { commandLine = "$(engine.path)
  throw new Exception("Invalid engine");
}
```

- CreateActivity

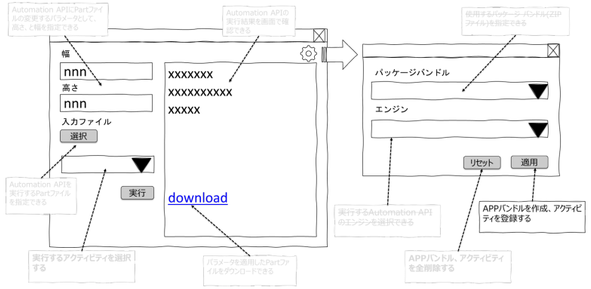
Define a new activity with an input file, input data (JSON) and an output file.

```
/// <summary>
/// Define a new activity
/// </summary>
[HttpPost]
[Route("api/aps/designautomation/activities")]
public async Task<IActionResult> CreateActivity([FromBody]JObject activitySpecs)
{
  // basic input validation
  string zipFileName = activitySpec FileName"].Value<string>();
  string engineName = activitySpecs[ ne"].Val

  // standard name for this sample
  string appBundleName = zipFileName + "Appb
  string activityName = zipFileName + "Activity";

  //
  Page<string> activities = await _designAutomat
  string qualifiedActivityId = string.Format("{0}.{
  if (!activities.Data.Contains(qualifiedActivityId)
  {
    // define the activity
    // ToDo: parametrize for different engines...
    dynamic engineAttributes = EngineAttributes
    string commandLine = string.Format(engineA
    Activity activitySpec = new Activity()
    {
      Id = activityName,
      Appbundles = new List<string>() { string.Fo
      CommandLine = new List<string>() { comm
      Engine = engineName,
      Parameters = new Dictionary<string, Parame
```

ApsDesignAutomation.js

```
function createActivity(cb) {
  jQuery.ajax({
    url: "api/aps/designautomation/activities",
    method: "POST",
    contentType: "application/json",
    data: JSON.stringify({
      zipFileName: $("#localBundles").val(),
      engine: $("#engines").val(),
    }),
    success: function (res) {
      writeLog("Activity: " + res.activity);
      if (cb) cb();
    },
  });
}
```

機能設計～Appバンドルを作成、アクティビティを登録する



クライアント(JS)

- ~~選択したエンジンとZIPファイルでAPPバンドルの作成を要求~~
- **~~作成したAPPバンドルでアクティビティの作成を要求~~**

Webサーバ

- ~~Appバンドルを作成or新規バージョンを作成して、エイリアスを付与~~
- **~~アクティビティを作成~~**

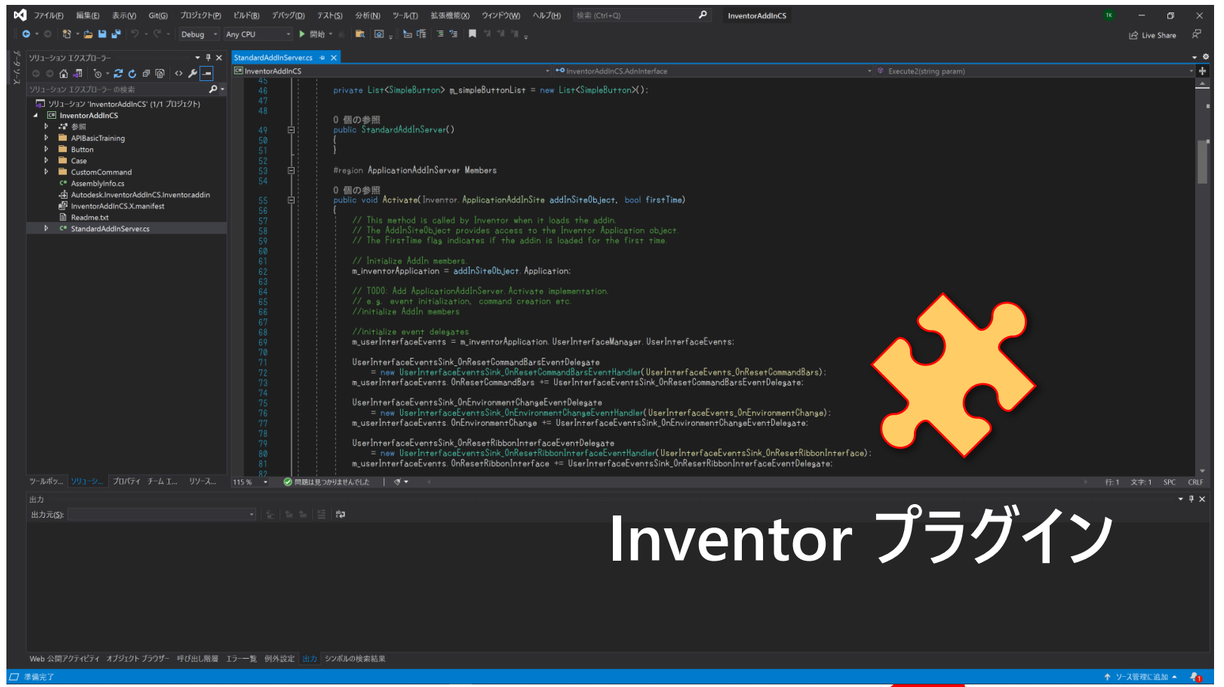
APS（API）

- ~~既存のAppバンドルを確認するAPI~~
- ~~Appバンドルを作成するAPI~~
- ~~エイリアスを作成するAPI~~
- ~~APPバンドルのバージョンを作成するAPI~~
- ~~エイリアスを付け替えるAPI~~
- **~~既存のActivityを確認するAPI~~**
- **~~Activityを作成するAPI~~**

プラグインの作成

Inventor Automation API プラグインの開発環境

- Inventor SDK
  - C#、VB.NET
- ターゲットプラットフォーム
  - ～Inventor 2024：.NET Framework 4.6～4.8
  - Inventor 2025～：.NET 8
- Microsoft Visual Studio Professional
  - 各バージョンに一致するアセンブリ出力が可能なもの
- 参照ライブラリ
  - autodesk.inventor.interop.dll
  - Nuget パッケージ
    - Autodesk.Forge.DesignAutomation.Inventor.Utils
    - Newtonsoft.Json （JSONファイルのR/W）



Inventor Automation API のコアエンジンとプラグインのバージョン

- デスクトップ製品と同期したコアエンジンバージョン
  - AppPackageはエンジンバージョンに合わせた作成が必須
  - コアエンジンIDの形式はコアエンジン毎に異なる
  - 2025年9月10日現在（Inventor 2021 ～2026が対象）

"Autodesk.Inventor+2021" ⇒ Inventor 2021
"Autodesk.Inventor+2022" ⇒ Inventor 2022
"Autodesk.Inventor+2023" ⇒ Inventor 2023
"Autodesk.Inventor+2024" ⇒ Inventor 2024
“Autodesk.Inventor+2025” ⇒ Inventor 2025

.Net Framework

"Autodesk.Inventor+2025_Net8" ⇒ Inventor 2025
“Autodesk.Inventor+2026” ⇒ Inventor 2026

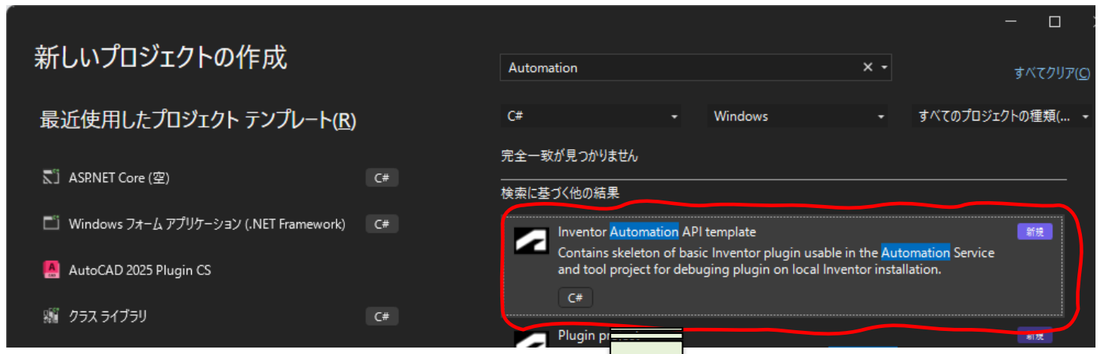
.Net 8

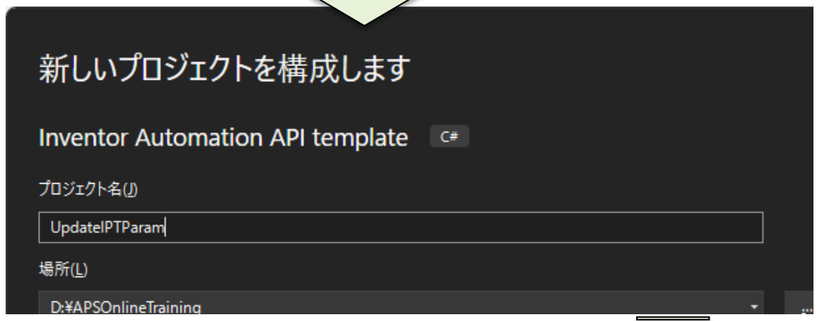
参考ブログ記事：Design Automation for Inventor 2026 コアエンジンは.NET 8版のみとなります

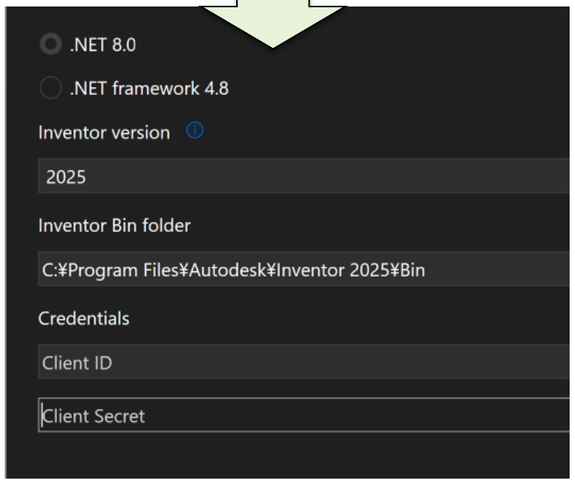
プラグインの作成 – Visual Studio Template

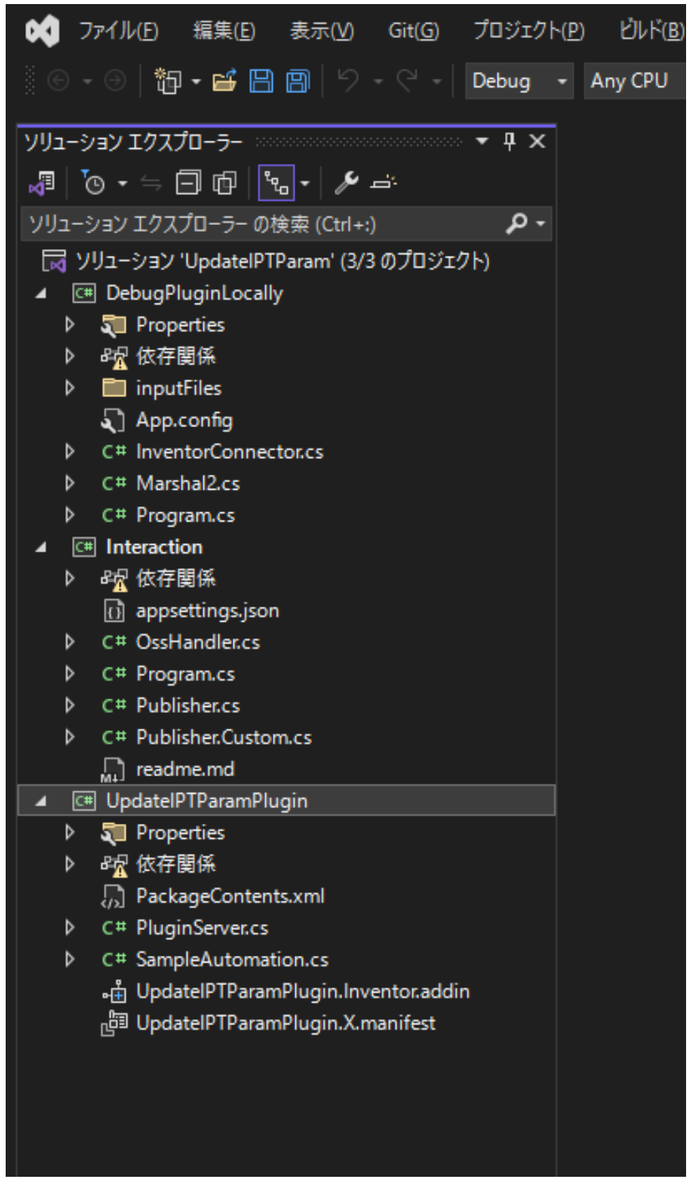
参考ブログ記事：Design Automation API for Inventor - Visual Studio template のご紹介

プラグインの作成 – Visual Studio Template



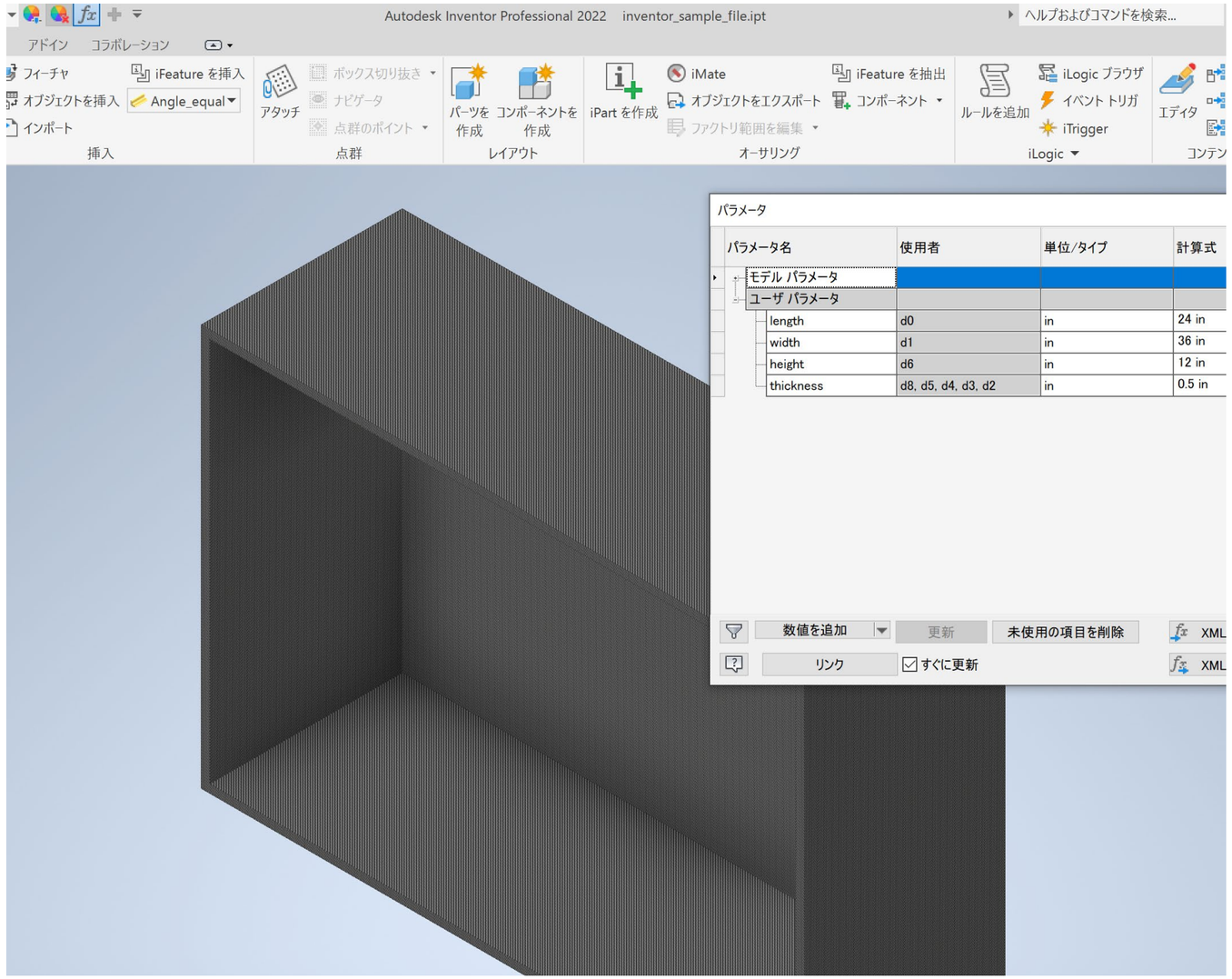






参考ブログ記事：Design Automation API for Inventor - Visual Studio template のご紹介

UpdateIPTParam Plug-In



widthとheightパラメータを
指定した値に変更するプラグイン

Inventor プラグインをAppBundleにパッケージ化

- カスタムコマンドを実行するバイナリファイルとサポートファイルをパッケージ化
- AppBundleにアップロードして使用

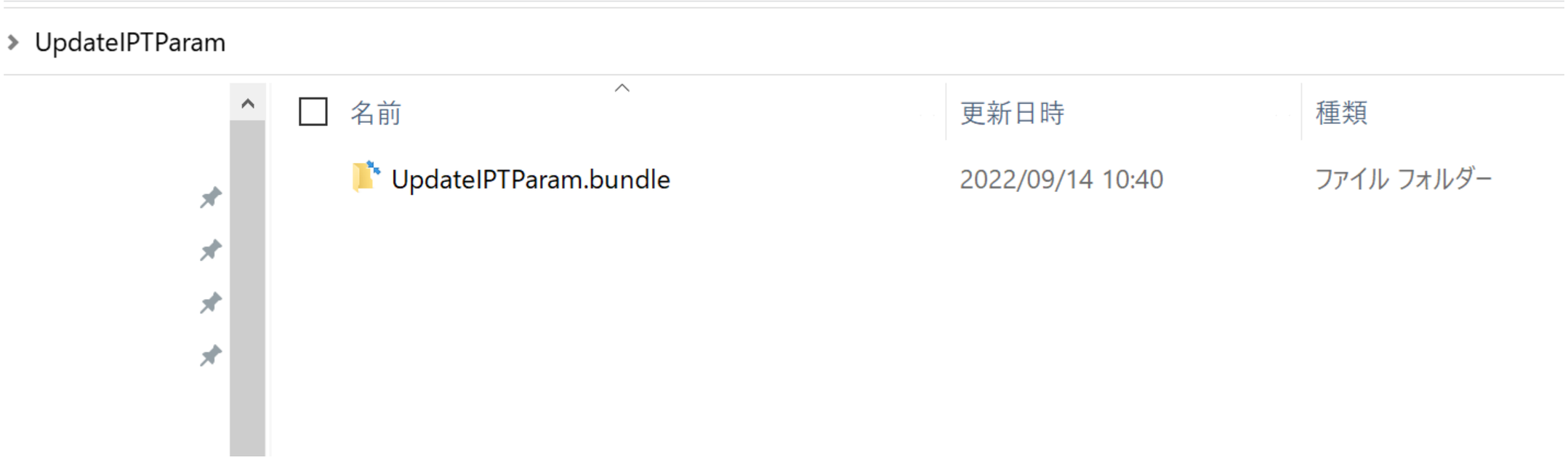
```
¥UpdateIPTParam.bundle
        + PackageContents.xml
        ¥Contents
                + UpdateIPTParam.dll
                + UpdateIPTParam.Inventor.addin
```

```
Platform="Inventor"
ModuleName="./Contents/ UpdateIPTParam.Inventor.addin"
```

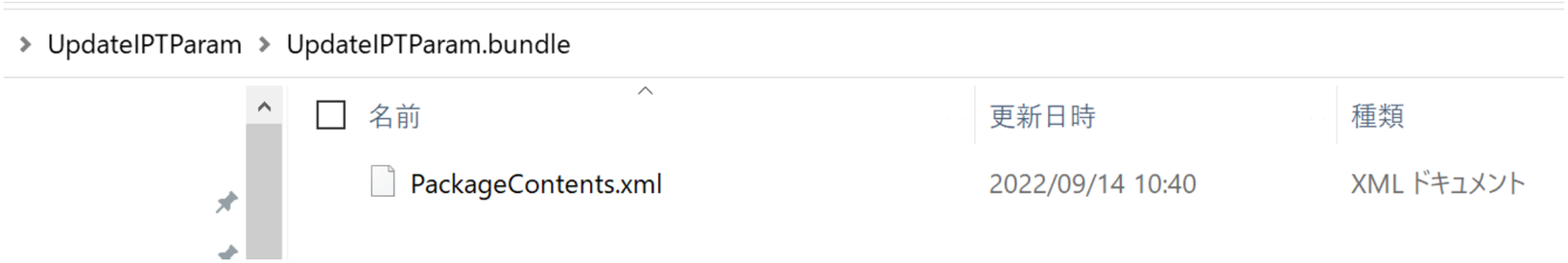
App Bundleの作成の手順

1. *.bundleフォルダの作成
2. PackageContents.xmlの作成
3. *.bundle¥contentsフォルダを作成し、アセンブリと.addinファイルをコピー
4. .addinファイルを修正
5. Zipアーカイブを作成

1. *.bundleフォルダの作成



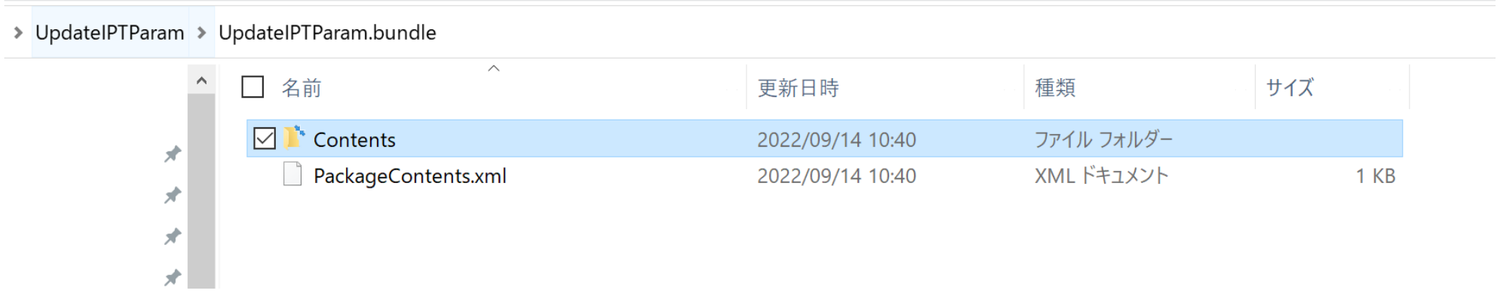
2. PackageContents.xmlの作成



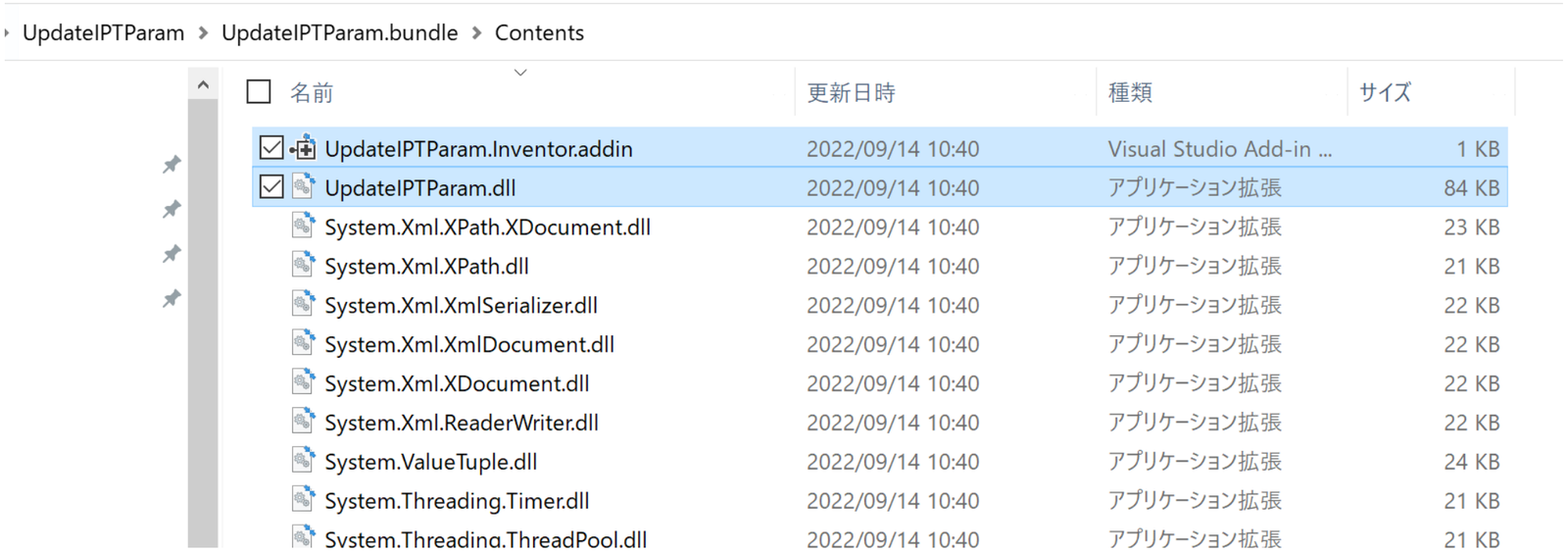

```
<?xml version="1.0" encoding="utf-8" ?>
<ApplicationPackage SchemaVersion="1.0" Version="1.0" ProductCode="{c803510f-2486-4dd7-b477-a236127d25ed}"
Name="UpdateIPTParam" Description="UpdateIPTParamPlugin Plugin" Author="Design Automation for Inventor">
   <CompanyDetails Name="Autodesk, Inc" Phone="415.555.5555" Url="learnforge.autodesk.io" Email="forge.help@autodesk.com" />
   <Components>
      <!-- For Inventor Engine, "Platform" attribute must be "Inventor" -->
      <RuntimeRequirements  OS="Win64" Platform="Inventor" />
      <!-- For Inventor Plug-in, the "Module" attribute must point to the .addin manifest file. -->
      <ComponentEntry LoadOnAutoCADStartup="False" LoadOnCommandInvocation="False"
         AppDescription="UpdateIPTParam App Package. "
         ModuleName="./Contents/UpdateIPTParam.Inventor.addin" AppName="UpdateIPTParam"/>
   </Components>
   <EnvironmentVariables>
   </EnvironmentVariables>
</ApplicationPackage>
```

3. *.bundle¥contentsフォルダを作成し、アセンブリと.addinファイルをコピー

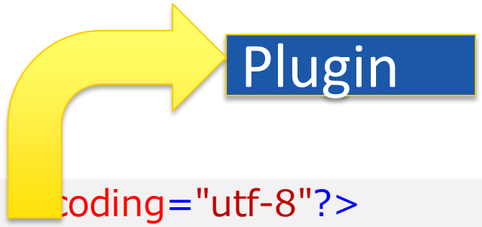
“Contents” フォルダを作成



アセンブリと.addinファイルをコピー

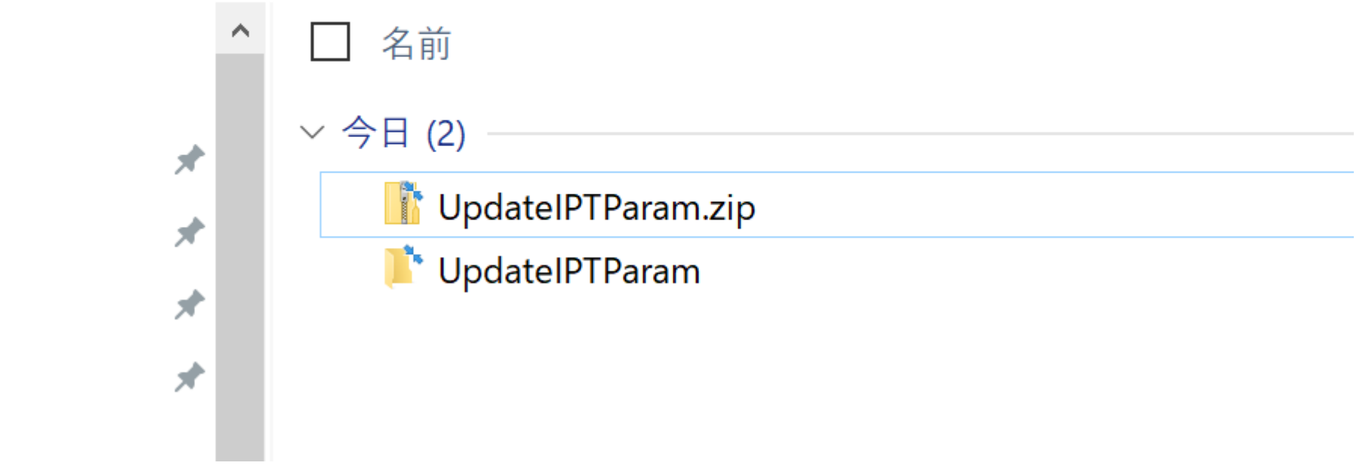


4. .addinファイルを修正




```
<?xml version="1.0" encoding="utf-8"?>
    <!-- Type attribute is same as Type registry key (Standard, Translator, Plugin (Server only) -->
    <Addin Type="Standard">
    <ClassId>{c803510f-2486-4dd7-b477-a236127d25ed}</ClassId>
    <ClientId>{c803510f-2486-4dd7-b477-a236127d25ed}</ClientId>
    <!-- Both of the following fields should be translated. NO OTHER FIELDS SHOULD BE TRANSLATED! -->
    <DisplayName>UpdateIPTParam</DisplayName>
    <Description>UpdateIPTParam</Description>
    <!-- Assumes that SimpleAddIn.dll is underneath Inventor¥bin -->
    <Assembly>UpdateIPTParam.dll</Assembly>
    <SupportedSoftwareVersionGreaterThan>17..</SupportedSoftwareVersionGreaterThan>
    <LoadOnStartUp>1</LoadOnStartUp>
    <Hidden>0</Hidden>
</Addin>
```

5. Zipアーカイブを作成



Inventor アドインのプラグイン化のポイント

1. GUIに依存した処理を除去
2. *Inventor.Application* の参照を*InventorServer*に変更
3. *InventorServer*を引数に取るコンストラクタの実装
4. *Run*、*RunWithArguments*メソッドに自動化処理を実装
5. COM Automationのサポート

参考ブログ記事：Forge Online - Design Automation : Inventor タスクの自動化

プラグインの作成 – Create Plugin

SampleAutomation.cs

Open the `SampleAutomation.cs` file and copy the following content to it. This is where the parameters are updated under the `Run` method.

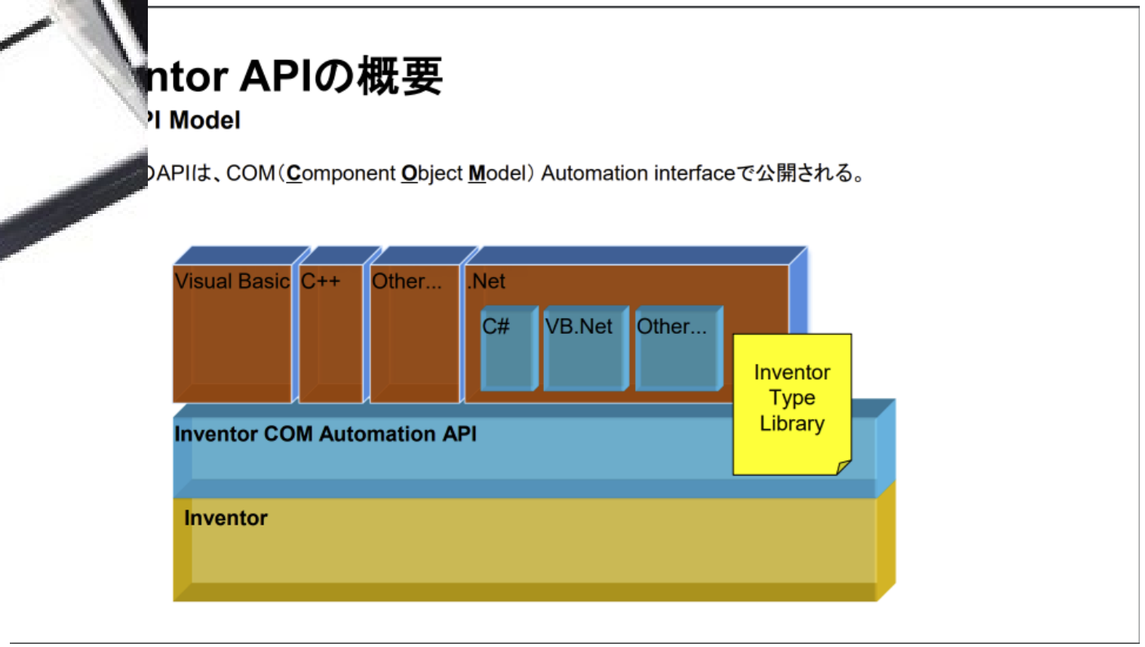
SampleAutomation.cs

```
using Inventor;
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Threading;

namespace UpdateIPTParam
{
    [ComVisible(true)]
    public class SampleAutomation
    {
        private InventorServer m_server;
        public SampleAutomation(InventorServer app) { m_server = app; }

        public void Run(Document doc)
        {
            try
            {
                // update parameters in the doc
                ChangeParameters(doc);

                // generate outputs
                var docDir = System.IO.Path.GetDirectoryName(doc.FullFileName);

                // save output file
                var documentType = doc.DocumentType;
                if (documentType == DocumentTypeEnum.kPartDocumentObject)
                {
                    // the name must be in sync with OutputIpt localName in Activity
                    var fileName = System.IO.Path.Combine(docDir, "outputFile.ipt");

                    // save file
                    doc.SaveAs(fileName, false);
                }
            }
            catch (Exception e) { LogTrace("Processing failed: {0}", e.ToString()); }
        }
```

プラグインの作成 – Create Plugin

```
/// <summary>
/// Change parameters in Inventor document.
/// </summary>
/// <param name="doc">The Inventor document.</param>
/// <param name="json">JSON with changed parameters.</param>
public void ChangeParameters(Document doc)
{
    var theParams = GetParameters(doc);

    Dictionary<string, string> parameters = JsonConvert.DeserializeObject<Dictionary<string, string>>(System.IO.File.ReadAllText("params.json"));
    foreach (KeyValuePair<string, string> entry in parameters)
    {
        try
        {
            Parameter param = theParams[entry.Key.ToLower()];
            param.Expression = entry.Value;
        }
        catch (Exception e) { LogTrace("Cannot update {0}: {1}", entry.Key, e.Message); }
    }
    doc.Update();
}
```

Inventor APIの学習
～Inventor 2026 API トレーニング マテリアル



参考ブログ記事：[Autodesk Inventor 2026 の APIトレーニングマテリアル]

作成済みのバンドル

Choose the engine

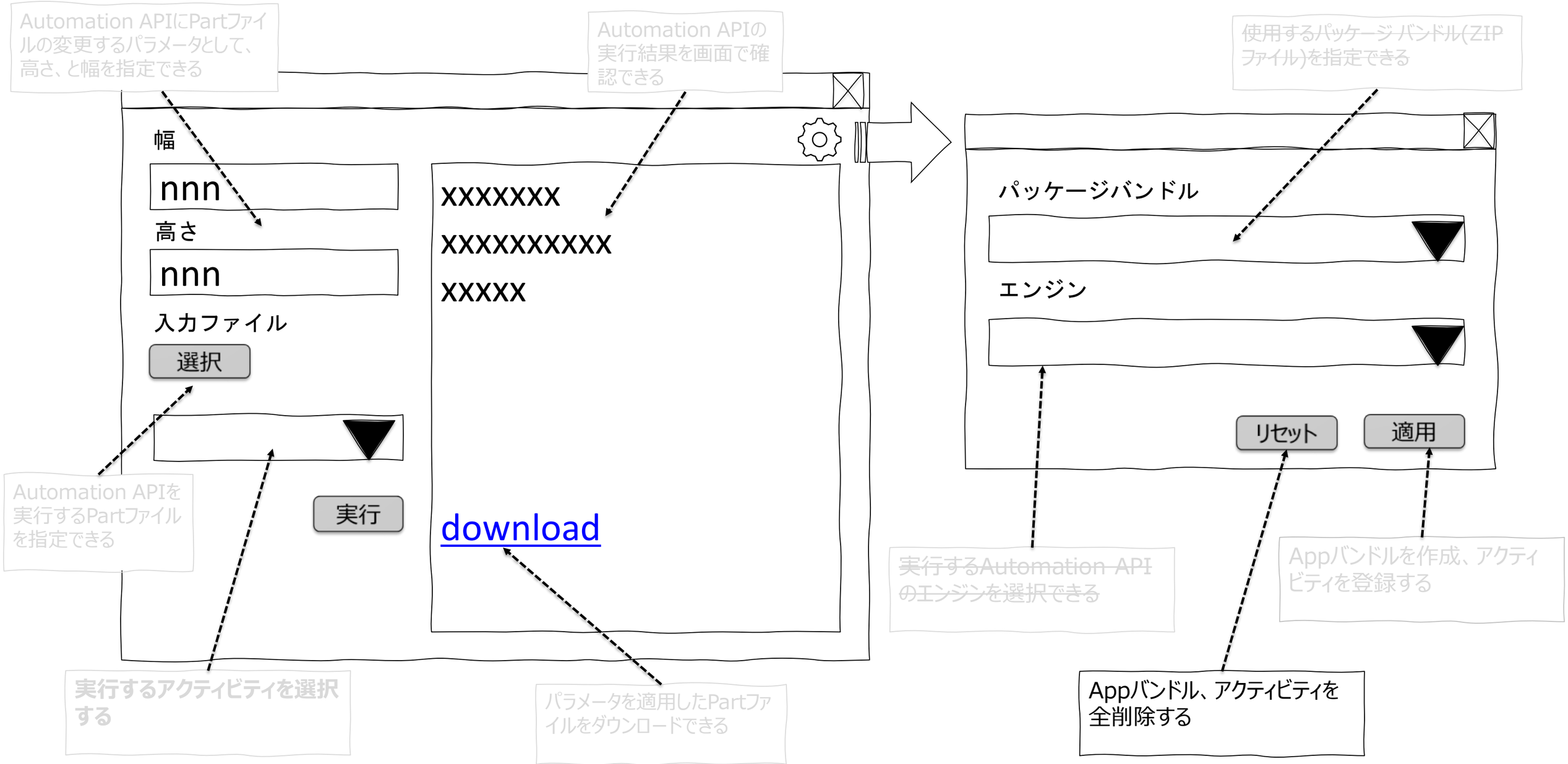
AutoCAD Plugin | **Inventor Plugin** | Revit Plugin | 3ds Max Plugin

This step will help you create a basic Inventor plugin. For more information, please visit My First Inventor Plugin tutorial.

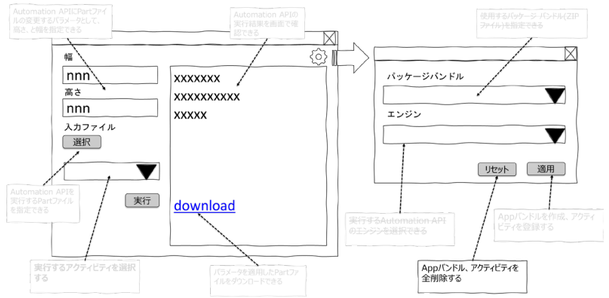
> You may download the Bundle ZIP into the `bundles/` (Node.js) or `/designAutomationSample/wwwroot/bundles` (.NET Core) folder and skip to **Upload Plugin Bundle** section.

Prerequisites

サンプルアプリケーションのGUIと機能

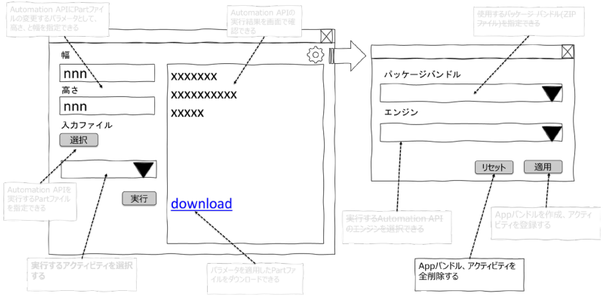


機能設計～Appバンドル、アクティビティを全削除する



クライアント(JS)	Webサーバ	APS（API）
▪ 全APPバンドル、アクティビティの削除を要求	▪ Appバンドル、Activityを削除	▪ Appバンドル、Activityを削除するAPI

機能設計～Appバンドル、アクティビティを全削除する



- Appバンドル、Activityを削除するAPI

Documentation / Automation API / **Reference Guide**

DELETE forgeapps/:id

Delete all data associated with the given app.

All AppBundles and Activities are DELETED.

This may take up to 2 minutes. During this time the app will not be able to make successful requests.

Resource Information

Method and URI	DELETE **https://developer.api.autodesk.com/da/us-east/v3/forgeapps/:id**
Authentication Context	**app only**
Required OAuth Scopes	`code:all`
Data Format	**JSON**

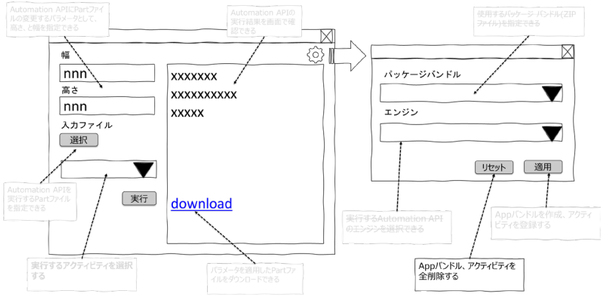
Request

Headers

Authorization * string	Must be `Bearer <token>` , where `<token>` is obtained via OAuth

* Required

機能設計～Appバンドル、アクティビティを全削除する



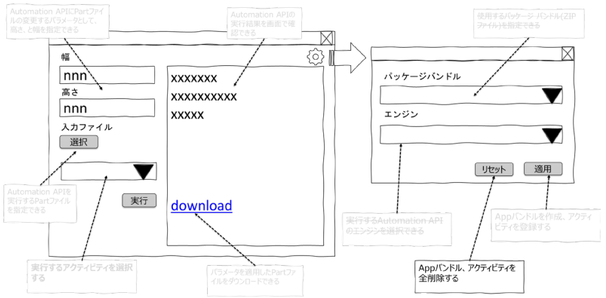
- 全APPバンドル、アクティビティの削除を要求

ApsDesignAutomation.jsの処理

```
$(document).ready(function () {
  prepareLists();
  $("#clearAccount").click(clearAccount);
  $("#defineActivityShow").click(defineActivityModal);
  $("#createAppBundleActivity").click(createAppBundleActivity);

});
```

```
function clearAccount() {
  if (
    !confirm(
      "Clear existing activities & appbundles before start. " +
      "This is useful if you believe there are wrong settings on your    account." +
      "¥n¥nYou cannot undo this operation. Proceed?"
    )
  )
  return;

  jQuery.ajax({
    url: "api/aps/designautomation/account",
    method: "DELETE",
    success: function () {
      prepareLists();
      writeLog("Account cleared, all appbundles & activities deleted");
    },
  });
}
```

機能設計～Appバンドル、アクティビティを全削除する



- Appバンドル、Activityを削除

- DesignAutomationController.csの処理

- ClearAccount

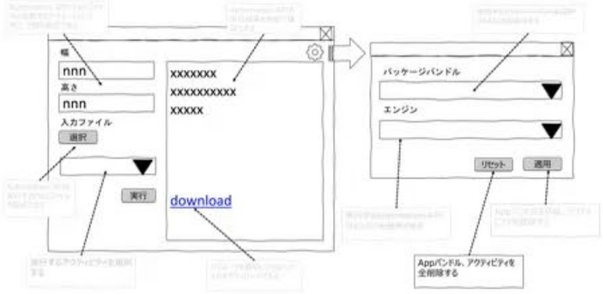
Last, but not least, to help you test, this api removes all appbundles and activities f
account.

```
/// <summary>
/// Clear the accounts (for debugging purposes)
/// </summary>
[HttpDelete]
[Route("api/aps/designautomation/account")]
public async Task<IActionResult> ClearAccount()
{
  // clear account
  await _designAutomation.DeleteForgeAppAsync("me");
  return Ok();
}
```

ApsDesignAutomation.js

```
function clearAccount() {
  if (
    !confirm(
      "Clear existing activities & appbundles before start. " +
      "This is useful if you believe there are wrong settings on your    account." +
      "¥n¥nYou cannot undo this operation. Proceed?"
    )
  )
  return;

  jQuery.ajax({
    url: "api/aps/designautomation/account",
    method: "DELETE",
    success: function () {
      prepareLists();
      writeLog("Account cleared, all appbundles & activities deleted");
    },
  });
}
```

機能設計～Appバンドル、アクティビティを全削除する



- Appバンドル、Activityを削除

- DesignAutomationController.csの処理

- ClearAccount

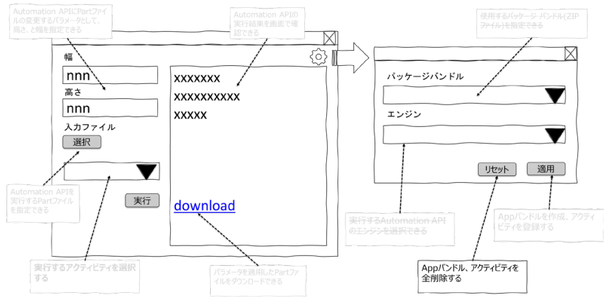
Last, but not least, to help you test, this api removes all appbundles and activities f
account.

```
/// <summary>
/// Clear the accounts (for debugging purposes)
/// </summary>
[HttpDelete]
[Route("api/aps/designautomation/account")]
public async Task<IActionResult> ClearAccount()
{
  // clear account
  await _designAutomation.DeleteForgeAppAsync("me");
  return Ok();
}
```

ApsDesignAutomation.js

```
function clearAccount() {
  if (
    !confirm(
      "Clear existing activities & appbundles before start. " +
      "This is useful if you believe there are wrong settings on your    account." +
      "¥n¥nYou cannot undo this operation. Proceed?"
    )
  )
  return;

  jQuery.ajax({
    url: "api/aps/designautomation/account",
    method: "DELETE",
    success: function () {
      prepareLists();
      writeLog("Account cleared, all appbundles & activities deleted");
    },
  });
}
```

機能設計～Appバンドル、アクティビティを全削除する



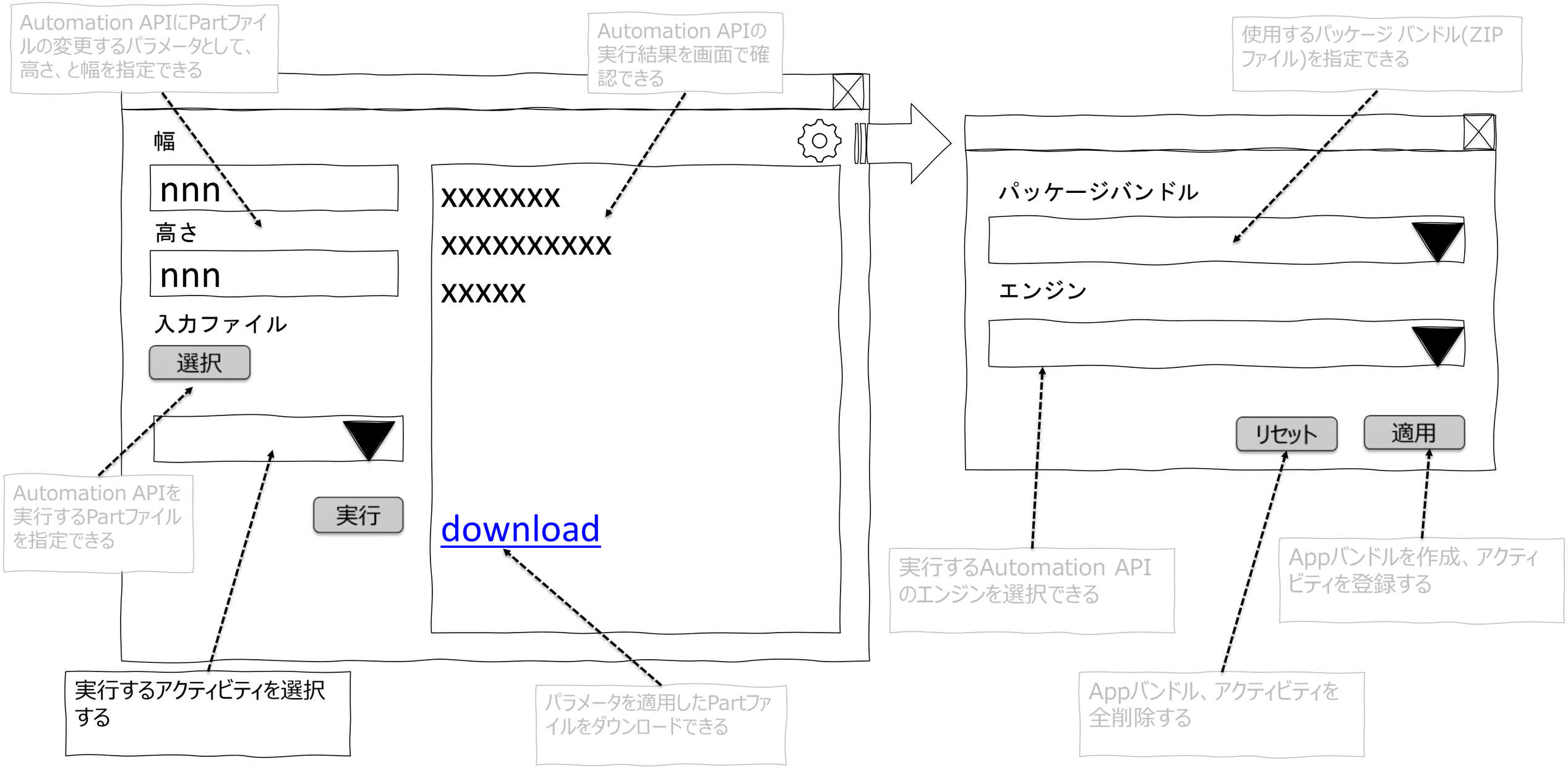
クライアント(JS)　　Webサーバ　　APS（API）

- ~~全APPバンドル、アクティビティの削除を要求~~

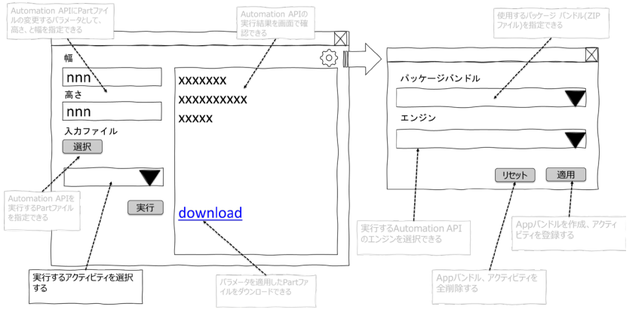
- ~~Appバンドル、Activityを削除~~

- ~~Appバンドル、Activityを削除するAPI~~

サンプルアプリケーションのGUIと機能

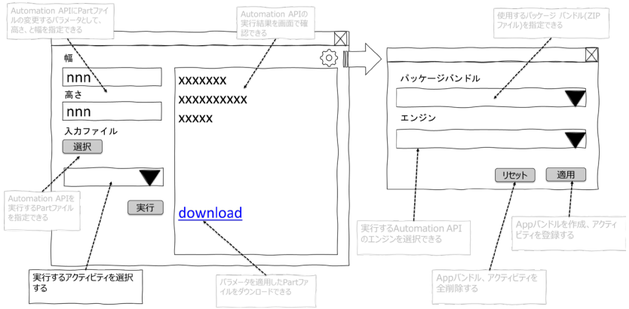


機能設計～実行するアクティビティを選択する



クライアント(JS)	Webサーバ	APS（API）
▪ **全Activityを取得する**	▪ **全Activityを取得する**	▪ **全Activityを取得するAPI**

機能設計～実行するアクティビティを選択する



- 全Activityを取得するAPI

GET activities

Lists all available Activities, including Activities shared with this app.

Resource Information

Method and URI	**GET https://developer.api.autodesk.com/da/us-east/v3/activities**
Authentication Context	**app only**
Required OAuth Scopes	`code:all`
Data Format	**JSON**

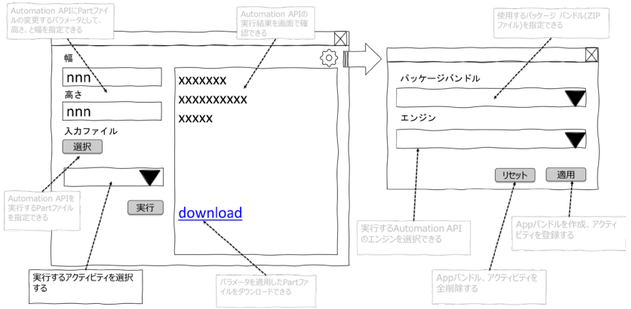
Request

Headers

Authorization * string	Must be `Bearer <token>`, where `<token>` is obtained via OAuth

* Required

機能設計～実行するアクティビティを選択する



- 全Activityを取得する

▪ApsDesignAutomation.jsとDesignAutomationController.csの処理

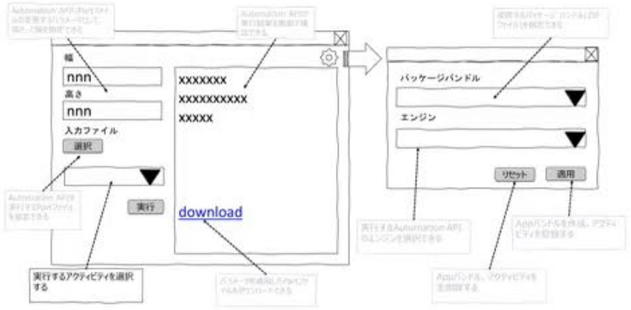
```
function prepareLists() {
list("activity", "/api/aps/designautomation/activities");
list("engines", "/api/aps/designautomation/engines");
list("localBundles", "/api/appbundles");
}
```

- GetDefinedActivities

We'll also need a method to return all defined activities. Note that returns only those defined by you (we use the APS Client Id as nick name, which then appears as a prefix).

```
/// <summary>
/// Get all Activities defined for this account
/// </summary>
[HttpGet]
[Route("api/aps/designautomation/activities")]
public async Task<List<string>> GetDefinedActivities()
{
  // filter list of
  Page<string> activities = await _designAutomation.GetActivitiesAsync();
  List<string> definedActivities = new List<string>();
  foreach (string activity in activities.Data)
    if (activity.StartsWith(NickName) && activity.IndexOf("$LATEST") == -1)
      definedActivities.Add(activity.Replace(NickName + ".", String.Empty));

  return definedActivities;
}
```

機能設計～実行するアクティビティを選択する



- 全Activityを取得する

▪ApsDesignAutomation.jsとDesignAutomationController.csの処理

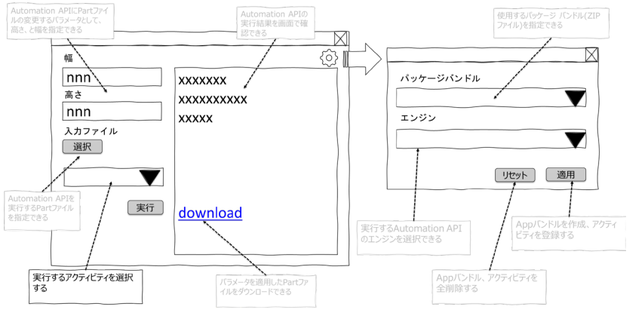
```
function prepareLists() {
list("activity", "/api/aps/designautomation/activities");
list("engines", "/api/aps/designautomation/engines");
list("localBundles", "/api/appbundles");
}
```

- GetDefinedActivities

We'll also need a method to return all defined activities. Note that returns only those defined by you (we use the APS Client Id as nick name, which then appears as a prefix).

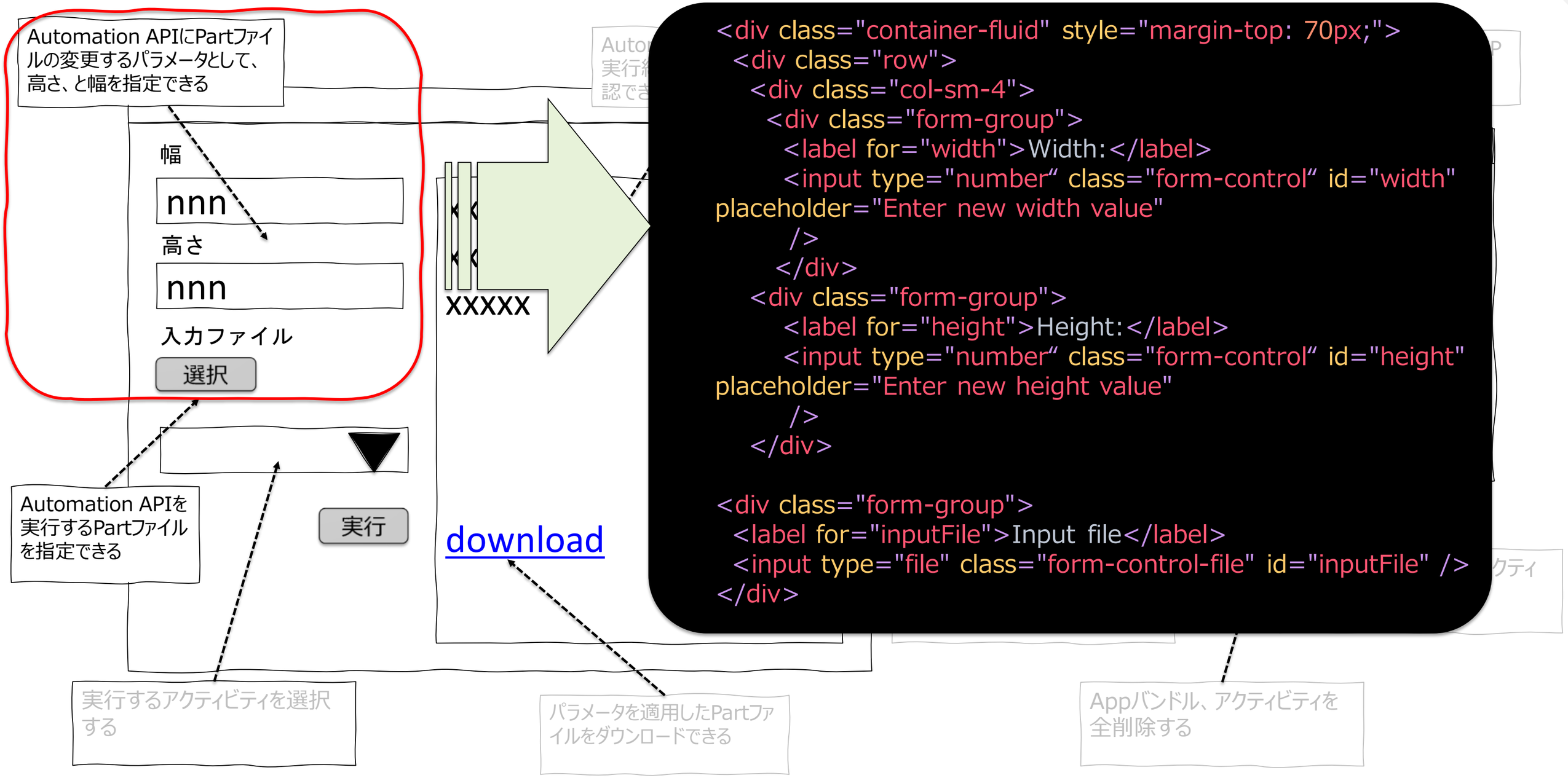
```
/// <summary>
/// Get all Activities defined for this account
/// </summary>
[HttpGet]
[Route("api/aps/designautomation/activities")]
public async Task<List<string>> GetDefinedActivities()
{
  // filter list of
  Page<string> activities = await _designAutomation.GetActivitiesAsync();
  List<string> definedActivities = new List<string>();
  foreach (string activity in activities.Data)
    if (activity.StartsWith(NickName) && activity.IndexOf("$LATEST") == -1)
      definedActivities.Add(activity.Replace(NickName + ".", String.Empty));

  return definedActivities;
}
```

機能設計～実行するアクティビティを選択する



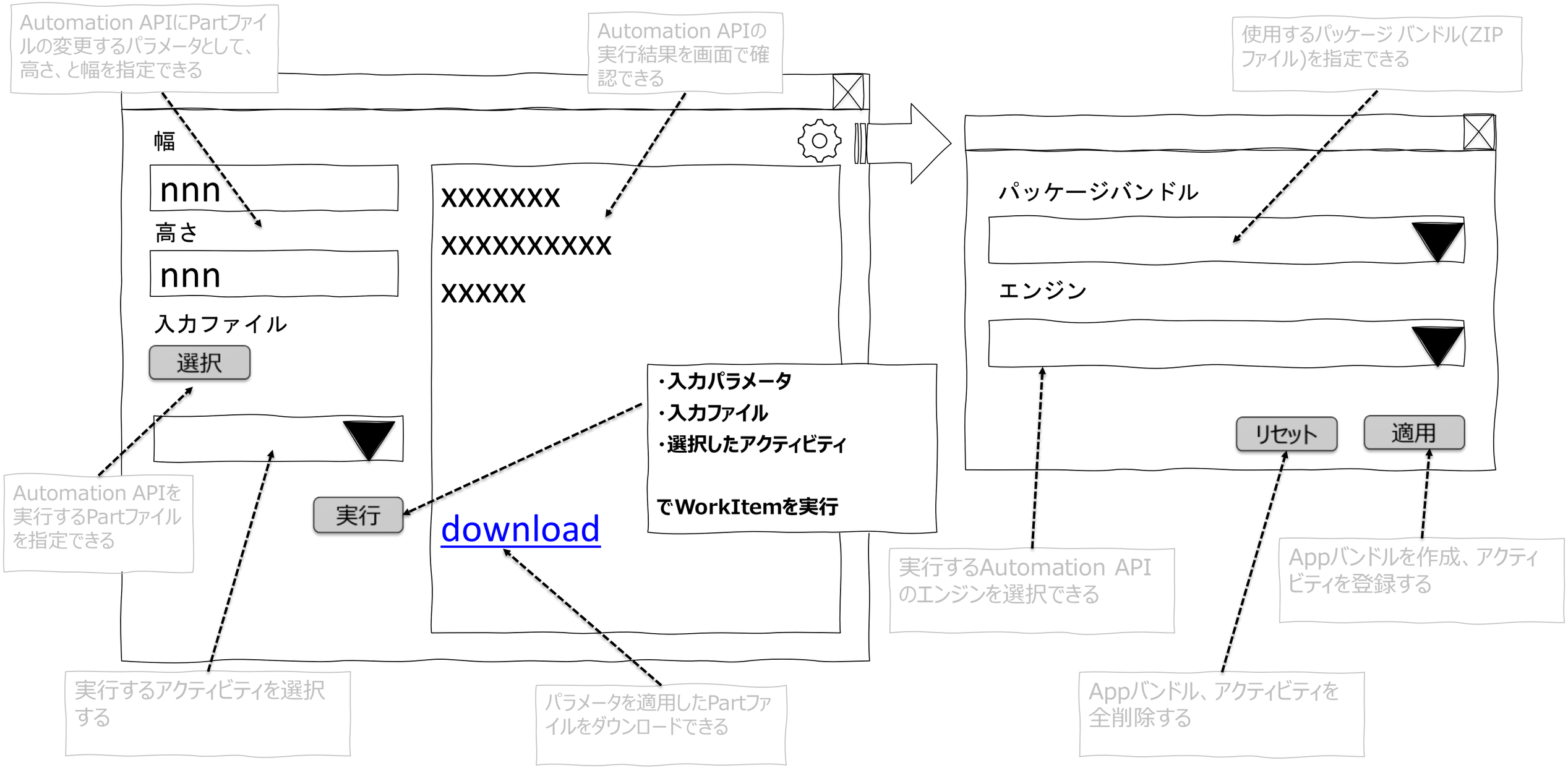
クライアント(JS)	Webサーバ	APS（API）
~~・全Activityを取得する~~	~~・全Activityを取得する~~	~~・全Activityを取得するAPI~~

サンプルアプリケーションのGUIと機能




```
<div class="container-fluid" style="margin-top: 70px;">
  <div class="row">
    <div class="col-sm-4">
      <div class="form-group">
        <label for="width">Width:</label>
        <input type="number“ class="form-control“ id="width"
placeholder="Enter new width value"
        />
      </div>
    <div class="form-group">
        <label for="height">Height:</label>
        <input type="number“ class="form-control“ id="height"
placeholder="Enter new height value"
        />
    </div>

<div class="form-group">
  <label for="inputFile">Input file</label>
  <input type="file" class="form-control-file" id="inputFile" />
</div>
```

サンプルアプリケーションのGUIと機能

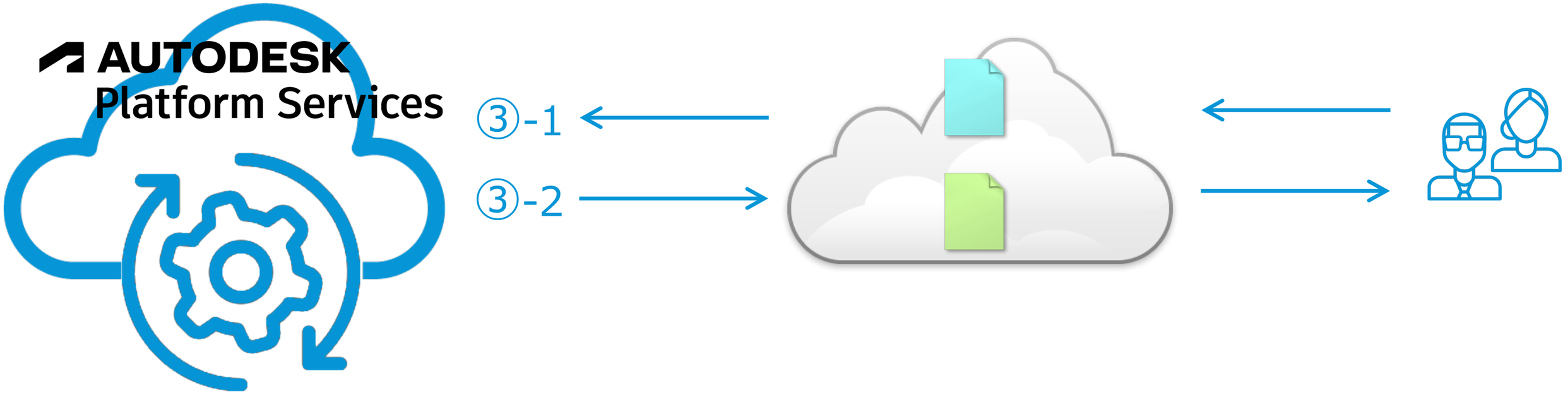


WorkItem – ワークアイテム

- 実際にクラウド上で処理されるタスクの単位
- 定義済 Activity を参照する WorkItem を作成
- Activity の応じた入出力パラメータ値の指定
  - 入力パラメータ
  - 入出力ファイル
- WorkItem の進捗状態の監視と処理終了の検出

入出力ファイル指定は Direct-to-S3 か署名付き URL

- **Automation API 用 Direct-to-S3指定**
  - **OSS Bucket の場合：**
    urn:adsk.objects:os.object:<Bucket name>**/**<Object key>



参考ブログ記事：Direct-to-S3 アプローチを簡素化する新機能、 さまざまな Activity と WorkItem

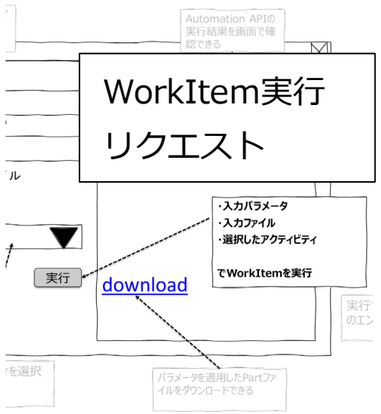
WorkItem 実行時に起こること

- AppBundle（アドイン）は実行時に作成の仮想環境に展開
- Automation API は WorkItem で指定された値を利用
  - 指定のクラウド ストレージから ipt をダウンロード（任意）
  - **指定の値を持つ JSON ファイルを作成**

```
{
    'width':999,
    'height':555
}
```

  - 成果ファイルを指定のクラウド ストレージへ保存（アップロード）
    - iptや PDF など
- WorkItem 終了後に仮想環境は自動削除（含む使用ファイル）
  - キャッシュして後日の WorkItem で使用することは不可

機能設計～WorkItemを実行



- Activity名
- 幅、高さパラメータ
- ファイル

ASP.NET Core

WorkItem実行

- 幅、高さパラメータ
- 入力ファイル URN
- 出力ファイル URN

WorkItem

Appbundle展開

パラメータ
ファイル作成

Buket作成

Upload

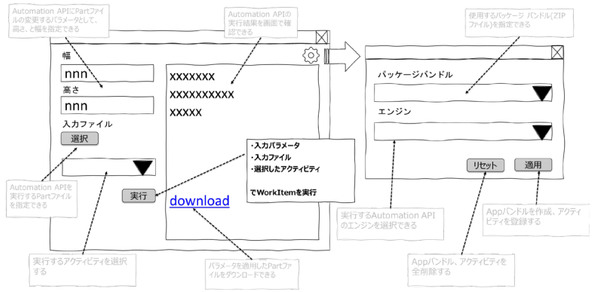
入力ファイル
Dowload

コアエンジン起動

結果ファイル作成

Upload

機能設計～WorkItemを実行



クライアント(JS)

- 選択したActivity名、幅、高さパラメータ、ファイルを指定してWorkItemの実行をリクエストする

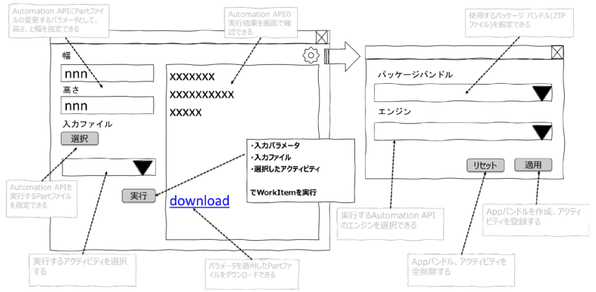
Webサーバ

- クライアントから送られたパラメータを元に以下を実行
  - ファイルをサーバ内に一時保存
  - OSSBucketを作成
  - OSSBuketにファイルをアップロード
  - 幅、高さパラメータをJSON形式に変換
  - WorkItemを実行

APS（API）

- OSSBucketを作成
- OSSにアップロード
- WorkItemを実行する

機能設計～WorkItemを実行



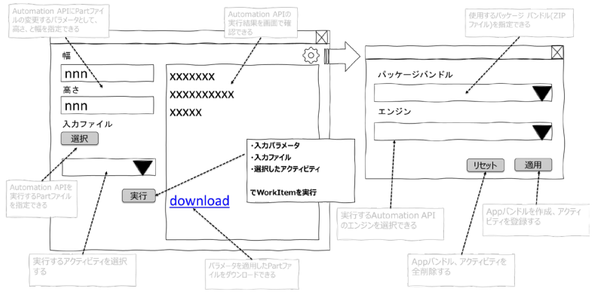
- 選択したActivity名、幅、高さパラメータ、ファイルを指定してWorkItemの実行をリクエストする

▪ApsDesignAutomation.jsの処理

```
$(document).ready(function () {
  prepareLists();
  $("#clearAccount").click(clearAccount);
  $("#defineActivityShow").click(defineActivityModal);
  $("#createAppBundleActivity").click(createAppBundleActivity);
  $("#startWorkitem").click(startWorkitem);

});
```

```
function startWorkitem() {
  var inputFileField = document.getElementById("inputFile");
  if (inputFileField.files.length === 0) {
    alert("Please select an input file");
    return;
  }
  if ($("#activity").val() === null) {
    alert("Please select an activity");
    return;
  }
  var file = inputFileField.files[0];
  startConnection(function () {
    var formData = new FormData();
    formData.append("inputFile", file);
    formData.append("data",JSON.stringify({
      width: $("#width").val(), height: $("#height").val(),
      activityName: $("#activity").val(),
      browserConnectionId: connectionId,
    }));
    writeLog("Uploading input file...");
    $.ajax({
      url: "api/aps/designautomation/workitems",
      data: formData,  processData: false, contentType: false,
      type: "POST",
      success: function (res) {
        writeLog("Workitem started: " + res.workItemId);
      },
    });
  });
}
```

機能設計～WorkItemを実行



- クライアントから送られたパラメータを元に以下を実行
  - ファイルをサーバ内に一時保存
  - OSSBucketを作成
  - OSSBuketにファイルをアップロード
  - 幅、高さパラメータをJSON形式に変換
  - WorkItemを実行

▪DesignAutomationController.csの処理

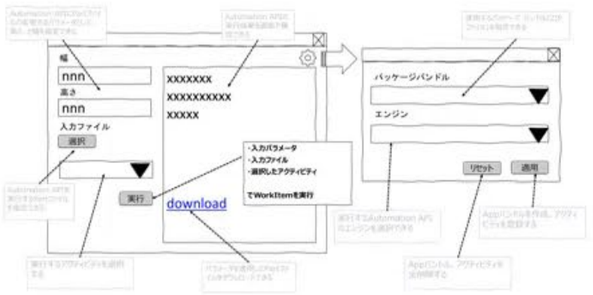
- StartWorkitem

This is where we actually start the Design Automation. This endpoint also uploads the input file to an OSS Bucket and define that the output should be saved at the same bucket. To help you identify the files, both input and output uses the same original file name, but with a suffix (input or output) plus a time stamp.

```
/// <summary>
/// Direct To S3
/// ref : https://aps.autodesk.com/blog/new-feature-support-direct-s3-migration-inputoutpu
/// </summary>
static void onUploadProgress(float progress, TimeSpan elapsed, List<UploadItemDesc> obje
{
  Console.WriteLine("progress: {0} elapsed: {1} objects: {2}", progress, elapsed, string.Join(",
}
public static async Task<string> GetObjectId(string bucketKey, string objectKey, dynamic oa
{
  try
  {
    ObjectsApi objectsApi = new ObjectsApi();
    objectsApi.Configuration.AccessToken = oauth.access_token;
    List<UploadItemDesc> uploadRes = await objectsApi.uploadResources(bucketKey,
      new List<UploadItemDesc> {
        new UploadItemDesc(objectKey, await System.IO.File.ReadAllBytesAsync(fileSaveP
      },
      null,
      onUploadProgress,
      null);
    Console.WriteLine("**** Upload object(s) response(s):");
    DynamicDictionary objValues = uploadRes[0].completed;
    objValues.Dictionary.TryGetValue("objectId", out var id);
```

```
/// <summary>
/// Input for StartWorkitem
/// </summary>
public class StartWorkitemInput
{
  public IFormFile inputFile { get; set; }
  public string data { get; set; }
}

/// <summary>
/// Start a new workitem
/// </summary>
[HttpPost]
[Route("api/aps/designautomation/workitems")]
public async Task<IActionResult> StartWorkitem([FromForm] StartWorkitemInput input)
{
  // basic input validation
  JObject workItemData = JObject.Parse(input.data);
  string widthParam = workItemData["width"].Value<string>();
  string heigthParam = workItemData["height"].Value<string>();
  string activityName = string.Format("{0}.{1}", NickName, workItemData["activityName"].V
  string browserConnectionId = workItemData["browserConnectionId"].Value<string>();

  // save the file on the server
  var fileSavePath = Path.Combine(_env.ContentRootPath, Path.GetFileName(input.inputFile
  using (var stream = new FileStream(fileSavePath, FileMode.Create)) await input.inputFile.

  // OAuth token
  dynamic oauth = await OAuthController.GetInternalAsync();

  // upload file to OSS Bucket
  // 1. ensure bucket existis
  string bucketKey = NickName.ToLower() + "-designautomation";
  BucketsApi buckets = new BucketsApi();
  buckets.Configuration.AccessToken = oauth.access_token;
```

機能設計～WorkItemを実行



- クライアントから送られたパラメータを元に以下を実行
  - ファイルをサーバ内に一時保存
  - OSSBucketを作成
  - OSSBuketにファイルをアップロード
  - 幅、高さパラメータをJSON形式に変換
  - WorkItemを実行

▪DesignAutomationController.csの処理

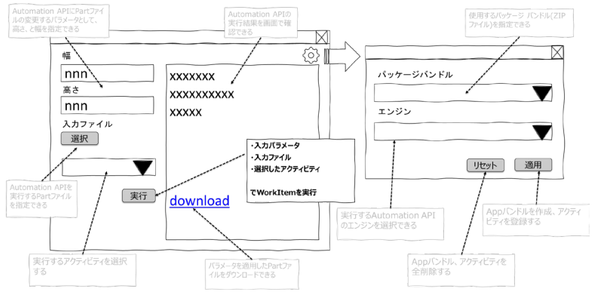
- StartWorkitem

This is where we actually start the Design Automation. This endpoint also uploads the input file to an OSS Bucket and define that the output should be saved at the same bucket. To help you identify the files, both input and output uses the same original file name, but with a suffix (input or output) plus a time stamp.

```
/// <summary>
/// Direct To S3
/// ref : https://aps.autodesk.com/blog/new-feature-support-direct-s3-migration-inputoutp
/// </summary>
static void onUploadProgress(float progress, TimeSpan elapsed, List<UploadItemDesc> obje
{
  Console.WriteLine("progress: {0} elapsed: {1} objects: {2}", progress, elapsed, string.Join("
}
public static async Task<string> GetObjectId(string bucketKey, string objectKey, dynamic oa
{
  try
  {
    ObjectsApi objectsApi = new ObjectsApi();
    objectsApi.Configuration.AccessToken = oauth.access_token;
    List<UploadItemDesc> uploadRes = await objectsApi.uploadResources(bucketKey,
      new List<UploadItemDesc> {
        new UploadItemDesc(objectKey, await System.IO.File.ReadAllBytesAsync(fileSaveP
      },
      null,
      onUploadProgress,
      null);
    Console.WriteLine("***** Upload object(s) response(s):");
    DynamicDictionary objValues = uploadRes[0].completed;
    objValues.Dictionary.TryGetValue("objectId", out var id);
```

```
/// <summary>
/// Input for StartWorkitem
/// </summary>
public class StartWorkitemInput
{
  public IFormFile inputFile { get; set; }
  public string data { get; set; }
}

/// <summary>
/// Start a new workitem
/// </summary>
[HttpPost]
[Route("api/aps/designautomation/workitems")]
public async Task<IActionResult> StartWorkitem([FromForm] StartWorkitemInput input)
{
  // basic input validation
  JObject workItemData = JObject.Parse(input.data);
  string widthParam = workItemData["width"].Value<string>();
  string heigthParam = workItemData["height"].Value<string>();
  string activityName = string.Format("{0}.{1}", NickName, workItemData["activityName"].V
  string browserConnectionId = workItemData["browserConnectionId"].Value<string>();

  // save the file on the server
  var fileSavePath = Path.Combine(_env.ContentRootPath, Path.GetFileName(input.inputFile
  using (var stream = new FileStream(fileSavePath, FileMode.Create)) await input.inputFile.

  // OAuth token
  dynamic oauth = await OAuthController.GetInternalAsync();

  // upload file to OSS Bucket
  // 1. ensure bucket existis
  string bucketKey = NickName.ToLower() + "-designautomation";
  BucketsApi buckets = new BucketsApi();
  buckets.Configuration.AccessToken = oauth.access_token;
```

機能設計～WorkItemを実行



クライアント(JS)

- ~~選択したActivity名、幅、高さパラメータ、ファイルを指定してWorkItemの実行をリクエストする~~

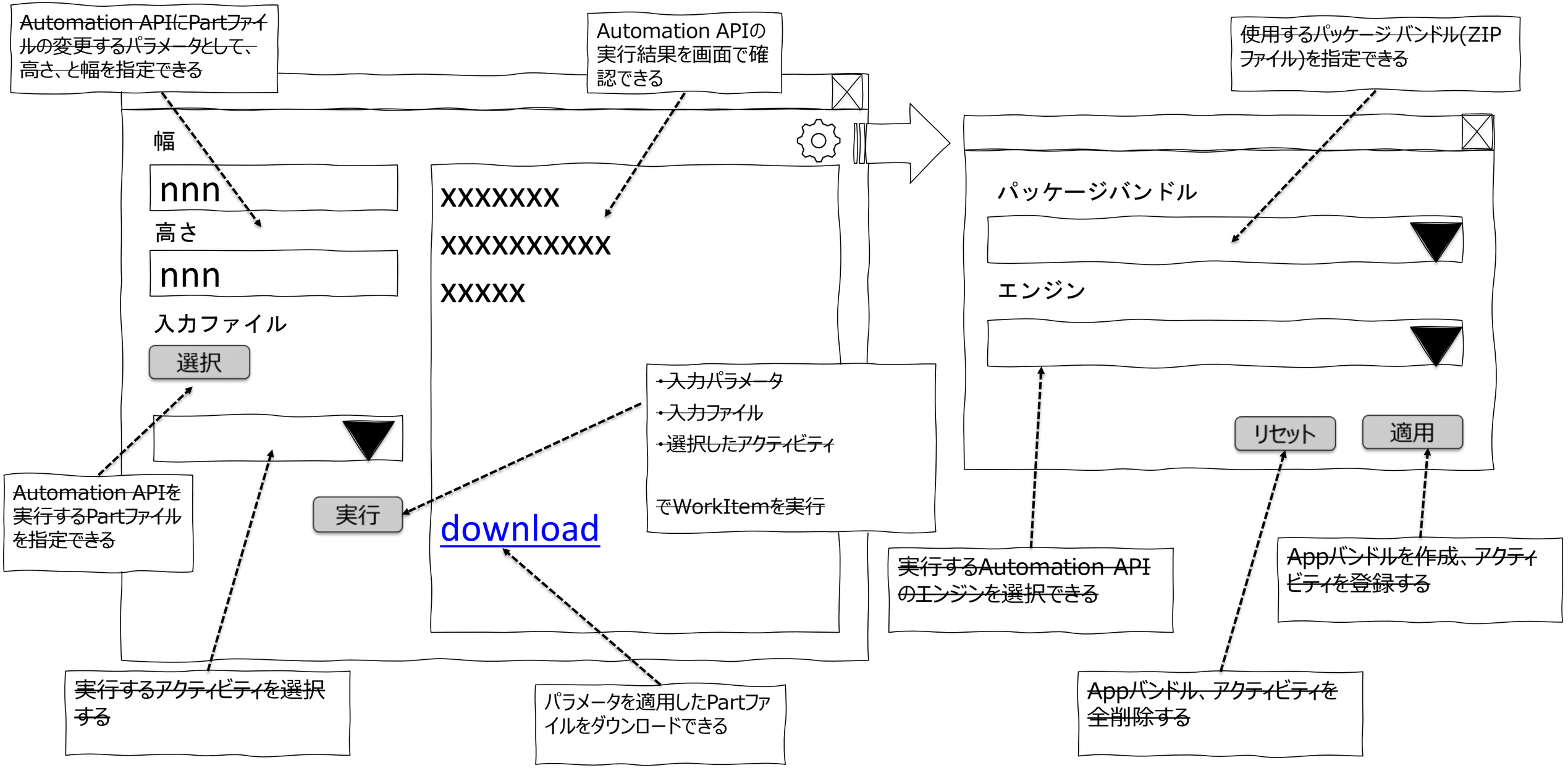
Webサーバ

- ~~クライアントから送られたパラメータを元に以下を実行~~
  - ~~ファイルをサーバ内に一時保存~~
  - ~~OSSBucketを作成~~
  - ~~OSSBuketにファイルをアップロード~~
  - ~~幅、高さパラメータをJSON形式に変換~~
  - ~~WorkItemを実行~~

APS（API）

- ~~OSSBucketを作成~~
- ~~OSSにアップロード~~
- ~~WorkItemを実行する~~

サンプルアプリケーションのGUIと機能

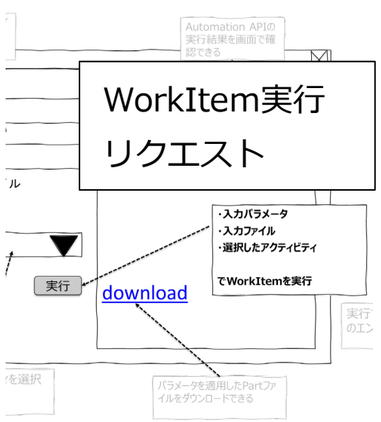


WorkItem 処理終了判定と処理レポートについて

- POST workitems エンドポイント呼び出し
  - 非推奨：150 rpm（150 回/1 分間）が上限
  - レスポンス ボディの **reportUrl**：WorkItem 毎に作成されるレポートログ

- OnComplete コールバックの実装
  - 推奨：要サーバー実装、ただし、クライアント⇔サーバー間トラフィックが低減
  - 開発時のクライアント実装では ngrok ツール等の利用が必要

機能設計～Automation APIの実行結果を画面で確認できる



ASP.NET Core

WorkItem実行

WorkItem

Appbundle展開

パラメータ
ファイル作成

JSON

入力ファイル
Dowlload

結果ファイル作成

Upload

同期での処理（HTTPリクエスト～レスポンス）は無理がありそう

POST workitems エンドポイントを繰り返して呼び出し

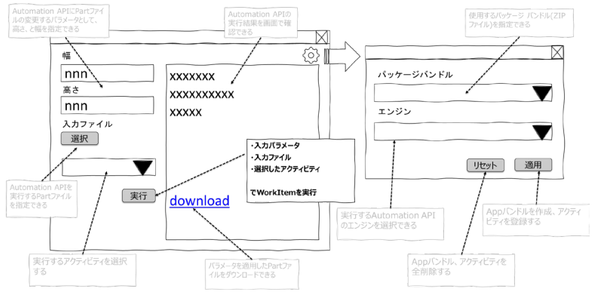
ソケット通信で、Webサーバ
⇒クライアントへ結果を送信

Rate Limit 厳守のお願い

- 1分間あたりのエンドポイント呼び出し数制限
  - 429 レスポンス ステータス受信時に適切に対応する必要あり
  - 例）Automation API WorkItem 実行時のポーリング

参考ブログ記事：Rate Limit（レート制限、呼び出し回数制限）

機能設計～ Automation APIの実行結果を画面で確認できる



クライアント(JS)

- ソケット通信のコネクションを開く
- サーバからプッシュ送信されたWorkItemの情報を表示

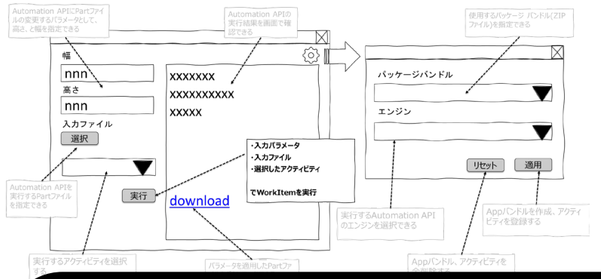
Webサーバ

- WorkItemの状態を確認する
- 取得したWorkItemの状態をクライアントにプッシュ送信

APS（API）

- WorkItemの状態を確認する

機能設計～ Automation APIの実行結果を画面で確認できる

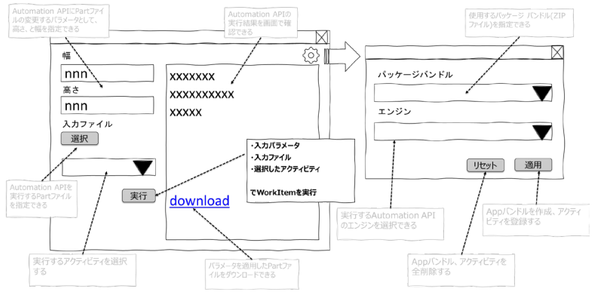


- ソケット通信のコネクションを開く
- サーバからプッシュ送信された WorkItemの情報を表示

```
function startWorkitem() {
  var inputFileField = document.getElementById("inputFile");
  if (inputFileField.files.length === 0) {
    alert("Please select an input file");
    return;
  }
  if ($("#activity").val() === null) {
    alert("Please select an activity");
    return;
  }
  var file = inputFileField.files[0];
  startConnection(function () {
    var formData = new FormData();
    formData.append("inputFile", file);
...
```

```
var connection;
var connectionId;
function startConnection(onReady) {
  if (connection && connection.connectionState) {
  if (onReady) onReady();
    return;
  }
  connection = new signalR.HubConnectionBuilder()
    .withUrl("/api/signalr/designautomation")
    .build();
  connection.start().then(function () {
    connection.invoke("getConnectionId").then(function (id) {
    connectionId = id; // we'll need this...
    if (onReady) onReady();
    });
  });

  connection.on("downloadResult", function (url) {
    writeLog('<a href="' + url + '">Download result file here</a>');
  });

  connection.on("onComplete", function (message) {
    writeLog(message);
  });
}
```

機能設計～ Automation APIの実行結果を画面で確認できる



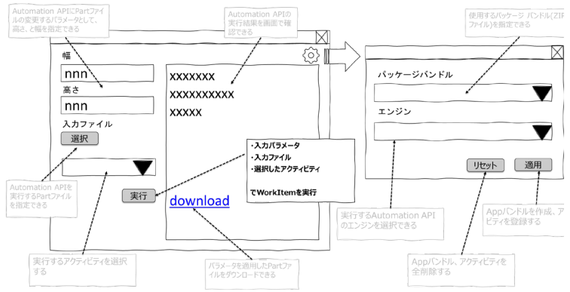
- WorkItemの状態を確認する
- 取得したWorkItemの状態をクライアントにプッシュ送信

```
        { "inputFile", inputFileArgument },
        { "inputJson",  inputJsonArgument },
        { "outputFile", outputFileArgument }


      }
    };
    WorkItemStatus workItemStatus = await _designAutomation.C
    MonitorWorkitem(oauth, browserConnectionId, workItemStatu
    return Ok(new { WorkItemId = workItemStatus.Id });
  }
```

- MonitorWorkitem

```
private async Task MonitorWorkitem(dynamic oauth, string browserConnectionId, WorkItem
{
  try
  {
    while (!workItemStatus.Status.IsDone())
    {
      await Task.Delay(TimeSpan.FromSeconds(2));
      workItemStatus = await _designAutomation.GetWorkitemStatusAsync(workItemStatu
      await _hubContext.Clients.Client(browserConnectionId).SendAsync("onComplete", w
    }
    using (var httpClient = new HttpClient())
    {
      byte[] bs = await httpClient.GetByteArrayAsync(workItemStatus.ReportUrl);
      string report = System.Text.Encoding.Default.GetString(bs);
      await _hubContext.Clients.Client(browserConnectionId).SendAsync("onComplete", re
    }

    if (workItemStatus.Status == Status.Success)
    {
      ObjectsApi objectsApi = new ObjectsApi();
      objectsApi.Configuration.AccessToken = oauth.access_token;

      ApiResponse<dynamic> res = await objectsApi.getS3DownloadURLAsyncWithHttpInfo
                    NickName.ToLower() + "-designautomation",
                    outputFileNameOSS, new Dictionary<string, object> {
                    { "minutesExpiration", 15.0 },
                    { "useCdn", true }
                    });
      await _hubContext.Clients.Client(browserConnectionId).SendAsync("downloadResult
      Console.WriteLine("Congrats!");
```

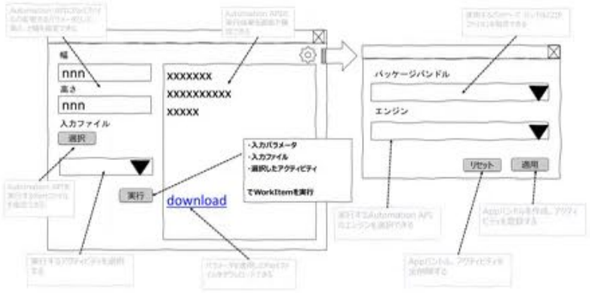
機能設計～パラメータを適用したPartファイルをダウンロードできる



- MonitorWorkitem

```
private async Task MonitorWorkitem(dynamic oauth, string browserConnectionId, WorkItem
{
    try
    {
        while (!workItemStatus.Status.IsDone())
        {
            await Task.Delay(TimeSpan.FromSeconds(2));
            workItemStatus = await _designAutomation.GetWorkitemStatusAsync(workItemStatu
            await _hubContext.Clients.Client(browserConnectionId).SendAsync("onComplete", wo
        }
        using (var httpClient = new HttpClient())
        {
            byte[] bs = await httpClient.GetByteArrayAsync(workItemStatus.ReportUrl);
            string report = System.Text.Encoding.Default.GetString(bs);
            await _hubContext.Clients.Client(browserConnectionId).SendAsync("onComplete", rep
        }

        if (workItemStatus.Status == Status.Success)
        {
            ObjectsApi objectsApi = new ObjectsApi();
            objectsApi.Configuration.AccessToken = oauth.access_token;

            ApiResponse<dynamic> res = await objectsApi.getS3DownloadURLAsyncWithHttpInfo
                                NickName.ToLower() + "-designautomation",
                                outputFileNameOSS, new Dictionary<string, object> {
                                { "minutesExpiration", 15.0 },
                                { "useCdn", true }
                                });
            await _hubContext.Clients.Client(browserConnectionId).SendAsync("downloadResult
            Console.WriteLine("Congrats!");
```

機能設計～ Automation APIの実行結果を画面で確認できる



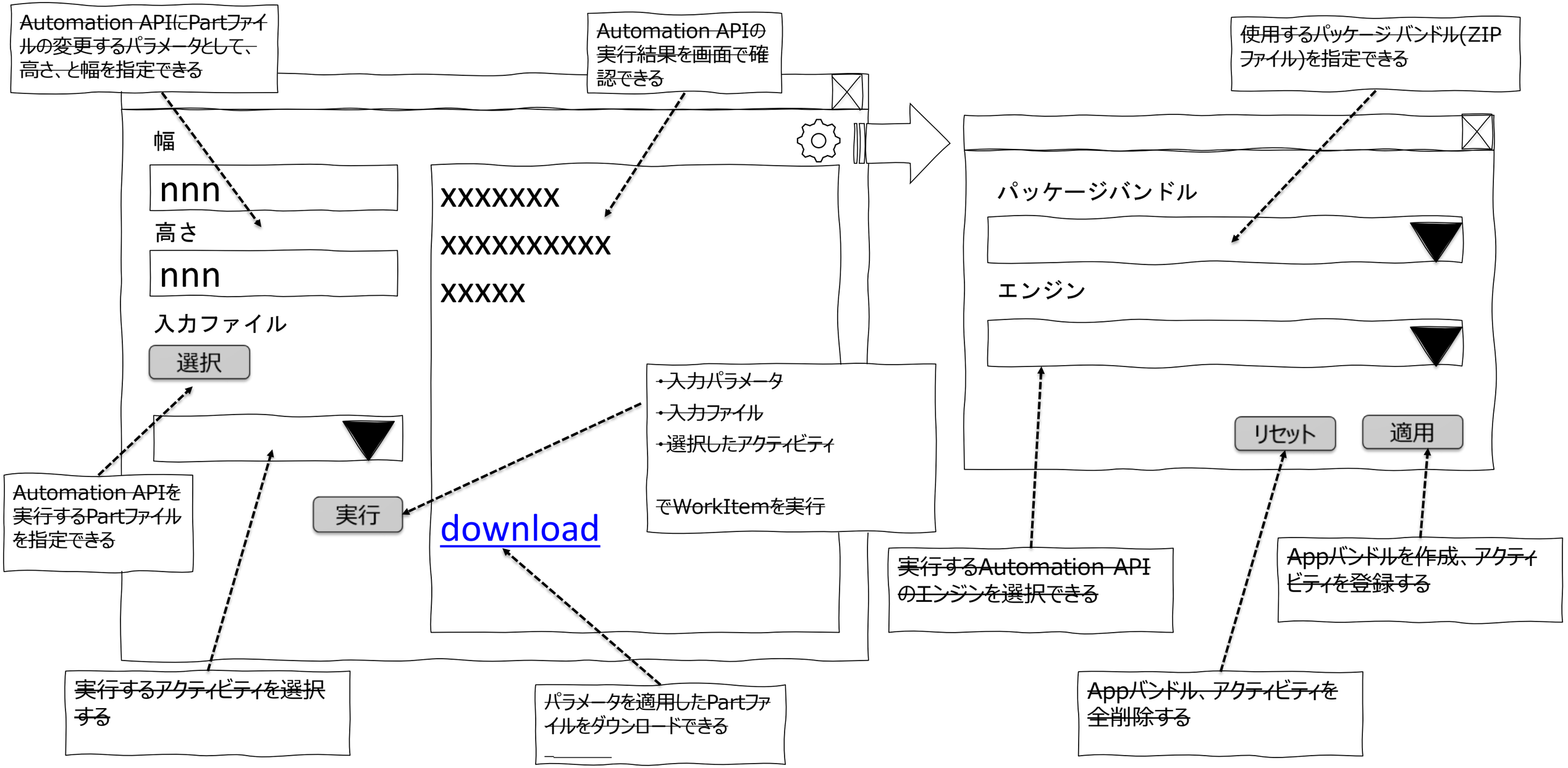
- WorkItemの状態を確認する
- 取得したWorkItemの状態をクライアントにプッシュ送信

```
        { "inputFile", inputFileArgument },
        { "inputJson",  inputJsonArgument },
        { "outputFile", outputFileArgument }


    }
};
WorkItemStatus workItemStatus = await _designAutomation.Cr
MonitorWorkitem(oauth, browserConnectionId, workItemStatu
return Ok(new { WorkItemId = workItemStatus.Id });
}
```

- MonitorWorkitem

```
private async Task MonitorWorkitem(dynamic oauth, string browserConnectionId, WorkItem
{
  try
  {
    while (!workItemStatus.Status.IsDone())
    {
      await Task.Delay(TimeSpan.FromSeconds(2));
      workItemStatus = await _designAutomation.GetWorkitemStatusAsync(workItemStat
      await _hubContext.Clients.Client(browserConnectionId).SendAsync("onComplete", w
    }
    using (var httpClient = new HttpClient())
    {
      byte[] bs = await httpClient.GetByteArrayAsync(workItemStatus.ReportUrl);
      string report = System.Text.Encoding.Default.GetString(bs);
      await _hubContext.Clients.Client(browserConnectionId).SendAsync("onComplete", re
    }

    if (workItemStatus.Status == Status.Success)
    {
      ObjectsApi objectsApi = new ObjectsApi();
      objectsApi.Configuration.AccessToken = oauth.access_token;

      ApiResponse<dynamic> res = await objectsApi.getS3DownloadURLAsyncWithHttpInfo
                    NickName.ToLower() + "-designautomation",
                    outputFileNameOSS, new Dictionary<string, object> {
                    { "minutesExpiration", 15.0 },
                    { "useCdn", true }
                    });
      await _hubContext.Clients.Client(browserConnectionId).SendAsync("downloadResult
      Console.WriteLine("Congrats!");
```

サンプルアプリケーションのGUIと機能

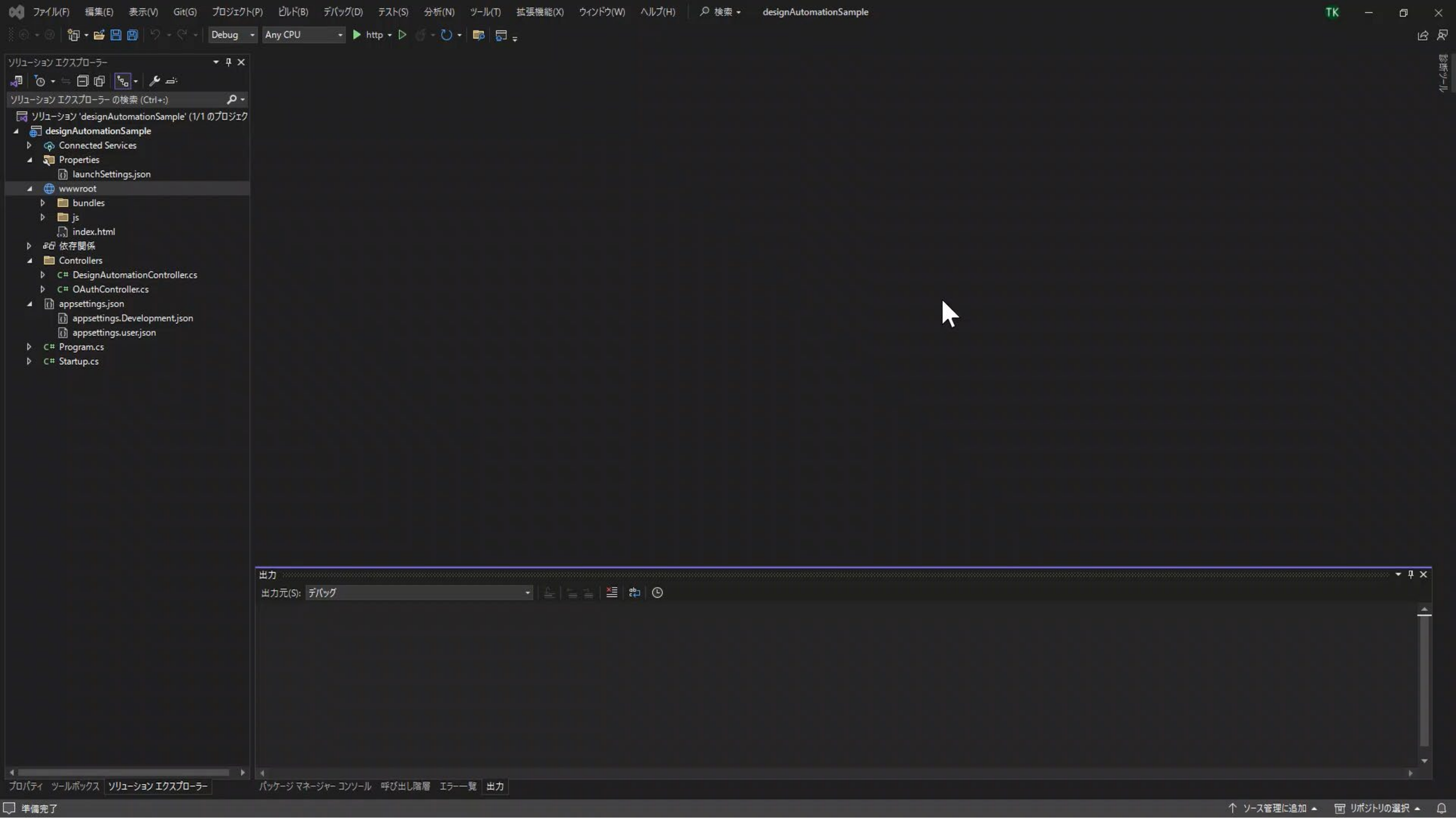


サンプルアプリケーションの要件

- ~~独自のWebアプリケーション~~
- ~~実行するAutomation APIのエンジンを選択できる~~
- ~~使用するパッケージ バンドル(ZIPファイル)を指定できる~~

⋮ ~~Automation APIの仕組み上必要な機能がある？~~

- ~~Automation APIを実行するPartファイルを指定できる~~
- ~~Automation APIにPartファイルの変更するパラメータとして、高さ、と幅を指定できる~~
- ~~Automation APIの実行結果を画面で確認できる~~
- ~~パラメータを適用したPartファイルをダウンロードできる~~


ファイル(F) 編集(E) 表示(V) Git(G) プロジェクト(P) ビルド(B) デバッグ(D) テスト(S) 分析(N) ツール(T) 拡張機能(X) ウィンドウ(W) ヘルプ(H)
検索
designAutomationSample
TK
Debug
Any CPU
http
ソリューション エクスプローラー
ソリューション エクスプローラー の検索 (Ctrl+;)
ソリューション 'designAutomationSample' (1/1 のプロジェク
designAutomationSample
Connected Services
Properties
launchSettings.json
wwwroot
bundles
js
index.html
依存関係
Controllers
DesignAutomationController.cs
OAuthController.cs
appsettings.json
appsettings.Development.json
appsettings.user.json
Program.cs
Startup.cs
出力
出力元(S): デバッグ
プロパティ ツールボックス ソリューション エクスプローラー
パッケージ マネージャー コンソール 呼び出し階層 エラー一覧 出力
準備完了
ソース管理に追加
リポジトリの選択

最後に…

- 本セッションでは、チュートリアルの内容を要件、機能にブレイクダウンして説明を進めました。チュートリアルの作業、説明の順序とは若干の差異があることに留意ください。

- Automation APIの概要、仕組み、注意点については、AutoCAD Automation APS Online Trainingの公開動画を参照ください。
  - AutoCAD Automationを題材にしたトレーニングですが、Automation APIの仕組みや、留意点 等はほぼ共通です。

参考ブログ記事：APS Online Training：AutoCAD Automation 収録公開