

一等一科技

U-Office Force X Plugin 開發手冊

目錄

1. U-Office Force X Plugin 開發手冊	3
2. 環境準備	4
2.1 模組架構	4
2.2 下載 Sample	6
2.3 配置設定	10
2.4 開發環境	18
3. ★外掛欄位	21
3.1 簡介與實作	21
3.2 進階功能	31
3.3 跨欄位互動	46
4. ★外掛頁面	47
4.1 頁面架構	47
4.2 Web 外掛頁面	49
4.3 APP 外掛頁面	54
5. 共用元件	56
5.1 Web	56
5.2 App	80
6. Plugin 套件	85
6.1 安裝	85
6.2 Plugin API	86
6.3 頁面權限	88
6.4 頁面工具	89
7. 外掛模組部署	91
7.1 如何部署	91
7.2 模組更新	100
8. 升級注意事項	103
8.1 2407 → 2025-R1	103
8.2 2402 → 2407	106
8.3 2312 → 2402	108

1. U-Office Force X Plugin 開發手冊



外掛模組範例為 SPA (Single Page Application) 架構，其專案內容包含後端 API 與前端 UI。但主要的外掛內容兼容性仍以前端為主，後端可以自行抽換。

`.NET 8`

`nodeJS 18.20.3`

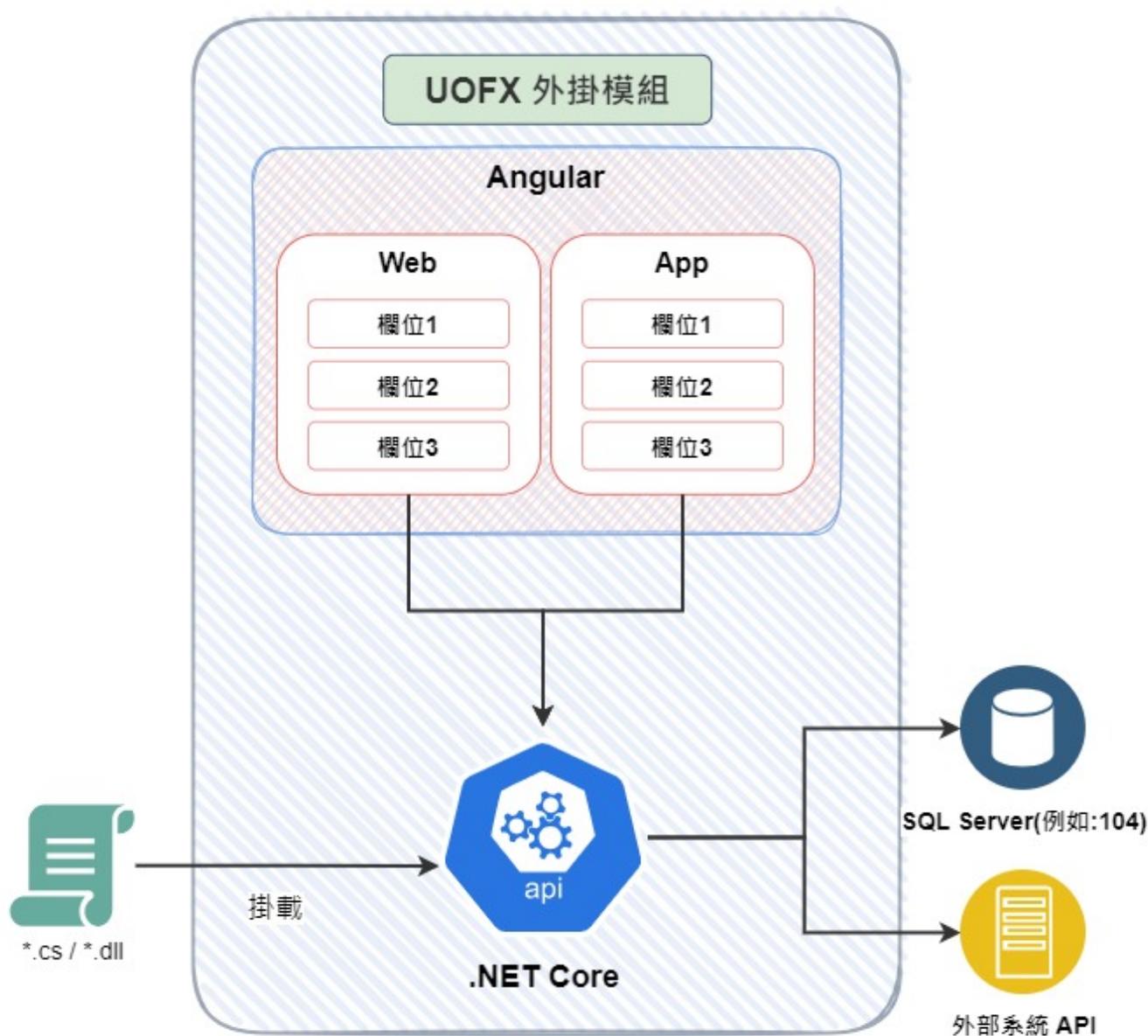
`Angular CLI 17.3.8`

前端 Web 所使用的 Framework 為 `Primeng 17.18.8`，而 App 則是 `ionic v8`。

2. 環境準備

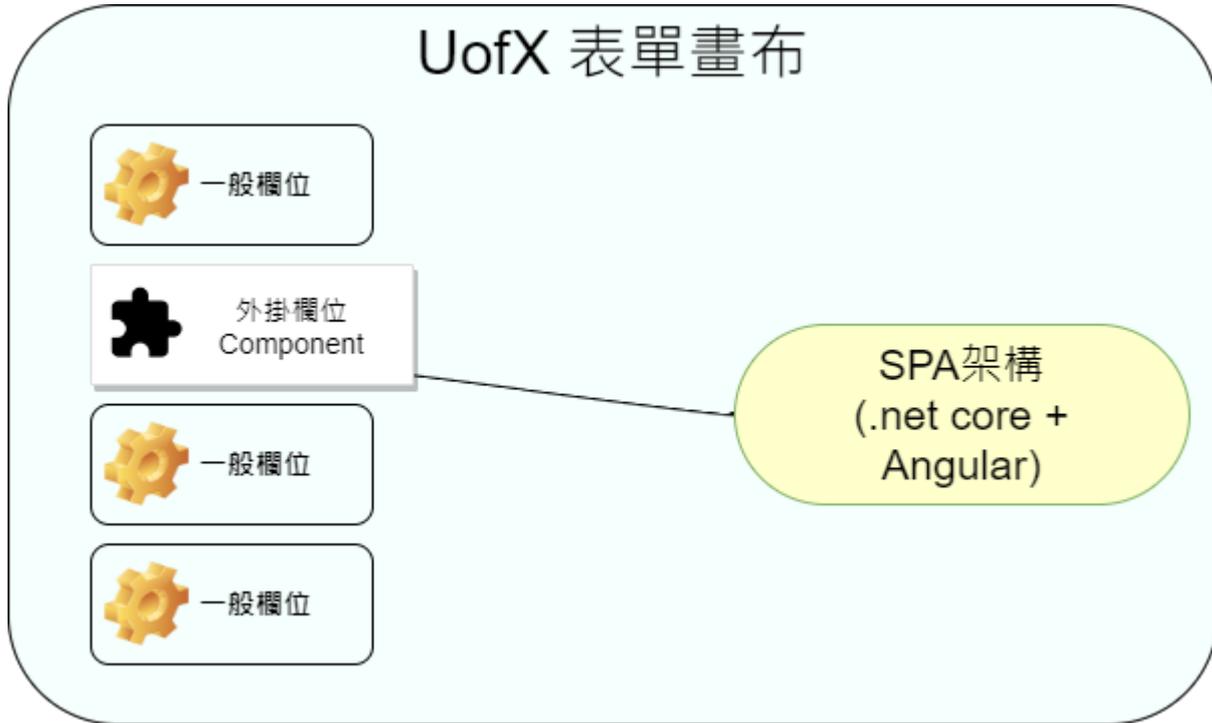
2.1 模組架構

外掛模組目前所提供的範例為 Visual Studio 上的前後端整合範本，透過 .NET 的 SPA Proxy 來做前後端一站服務，但主要的外掛功能仍是在前端上，所以後端 .NET 的部分可以依照自己的需求進行掛載或抽換，也可以當中間介面層去連接外部 API，並不會影響外掛本模組與 UOF X 相容性。



UOF X 外掛模組專案架構

外掛欄位是 BPM 模組表單欄位中其中一種類型，可透過 UOF X 的 BPM 模組功能掛載自製的欄位。



2.2 下載 Sample

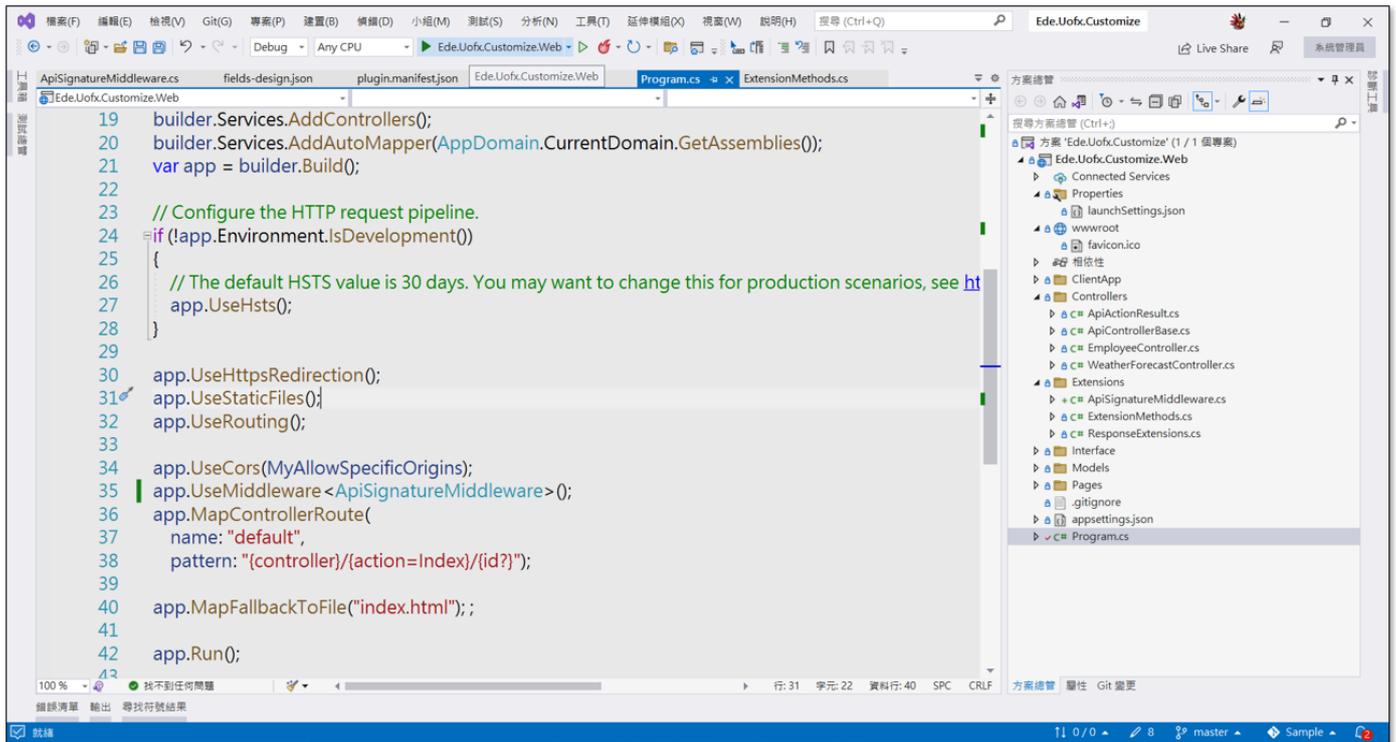
我們有在 Github 上提供基礎的 [範例程式](#) 上，直接架設後即可在 UOF X 掛入基礎的外掛欄位。範例中使用 .NET 專案搭配 SpaProxy，前端及後端都會啟動服務使用代理轉址的方式轉送請求。

使用 Visual Studio 啟動

 **Note**

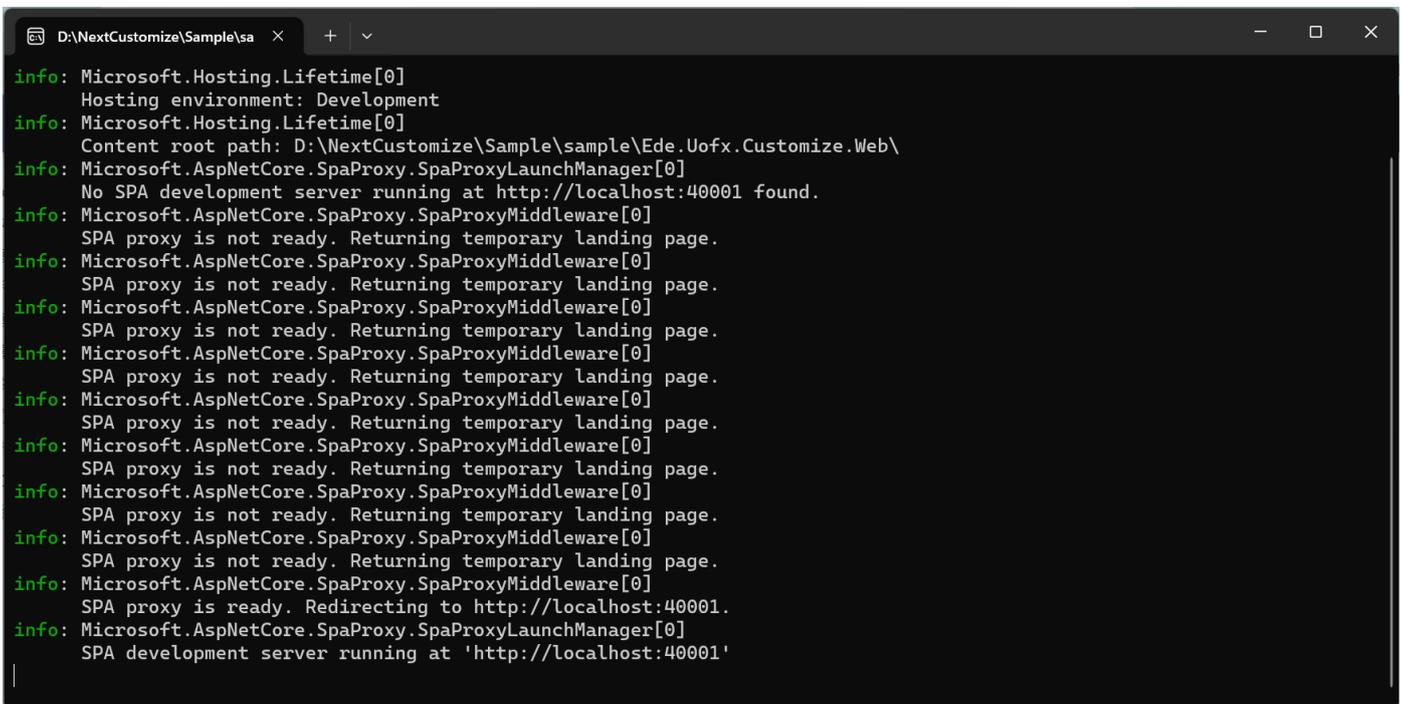
Visual Studio 建議安裝 2022 以上的版本。

1. 透過 Visual Studio 開啟 `sample/Ede.Uofx.Customize.sln`，透過執行按鈕或F5。



Visual Studio 啟動

2. 執行後會有兩個 console 程式執行，前端 `http://localhost:40001/` 和後端 web api 站台部分 `http://localhost:5246/`。



console webapi

```
node_modules_ionic_core_dist_esm_ion-backdrop_entry_js.js | -
| 3.24 kB |
node_modules_ionic_core_dist_esm_status-tap-211flad8_js.js | status-tap-211
flad8-js | 2.95 kB |
node_modules_rxjs_dist_esm_internal_operators_distinctUntilChanged_js-_55760_js | -
| 1.91 kB |
node_modules_rxjs_dist_esm_internal_operators_distinctUntilChanged_js-_55761_js | -
| 1.91 kB |
node_modules_rxjs_dist_esm_internal_operators_map_js.js | -
| 1.28 kB |
node_modules_rxjs_dist_esm_internal_observable_of_js.js | -
| 1.10 kB |
| - |
| - | 0 bytes |

Build at: 2024-02-02T01:41:37.251Z - Hash: dbec9873fb30b7ef - Time: 18128ms
** Angular Live Development Server is listening on localhost:40001, open your browser on http://localhost:40001/ **

✓ Compiled successfully.
```

console client

3. 執行成功將看到專案執行開啟的瀏覽器畫面中外掛欄位的设计畫面。



執行成功畫面

2.3 配置設定

外掛模組開發後於 UOF X 使用時需先調整 Config 內容，如此才能在 UOF X 正常更新內容，並且正常顯示表單欄位與選單。

JSON 設定檔位置固定不可任意更換，內容可參照於 @uofx/plugin 中所提供的 schema 進行撰寫，所需設定的屬性與類型使用 \$schema 設定後皆有說明可以參考。

相關 JSON 檔位置

plugin.manifest.json
plugin.versions.json
assets\configs\fields-design.json
assets\configs\fields-runtime.json assets\configs\routes.json

配置檔 **plugin.manifest.json**

- `schemaVersion`: 整個外掛模組的配置與設定案版本，若目前的版本與安裝 @uofx/plugin 後所提供的版本不一致，代表配置與設定檔結構有變更，需要進行調整。
- `name`: 外掛模組名稱。
- `code`: 必須在 Plugin 管理中為唯一值，僅能使用英文、數字和 `.`，且字串長度不可超過20。
- `description`: 外掛模組描述。
- `manufacturerCode`: 供應商代碼。
- `manufacturer`: 供應商名稱。
- `production`: 設定 `false` 會在 Plugin 管理中的更多下拉按鈕中出現「重載設定檔」，重載設定可以直接強制更新所有除了 `code` 以外的設定，不必透過標準的檢查更新流程來更新設定檔。

plugin.manifest.json

```
{
  "$schema": "../node_modules/@uofx/plugin/schema/plugin-manifest.schema.json",
  "schemaVersion": "104",
  "name": "一等一外掛模組範例",
  "code": "Ede.Sample",
  "description": "新一代 UOF X 外掛模組",
  "manufacturerCode": "Ede",
  "manufacturer": "一等一科技",
  "production": false
}
```

版本設定檔 `plugin.versions.json`

用來設定模組版本與 UOF X 相依性及變更說明，UOF X 會依照版本設定順序由上而下排序，所以版本需要由大到小進行設定。

- `version`: 模組版本。
- `minimumUOFXVersion`: 最小 UOF X 支援版號，共 3 碼，以下方範本為例，2.92.0 會執行並使用 2.0 版本程式，2.95.0 則會執行並使用 3.0 版本程式。
- `changelog`: 提供 `string` 陣列，條列式呈現。

plugin.versions.json

```
{
  "$schema": "../node_modules/@uofx/plugin/schema/plugin-versions.schema.json",
  "versions": [
    {
      "version": "3.0",
      "minimumUOFXVersion": "2.94.0",
      "changelog": [
        "新增了2個欄位",
        "修正了一些錯誤"
      ]
    },
    {
      "version": "2.0",
      "minimumUOFXVersion": "2.90.0",
      "changelog": [
        "本次更新除掉了幾條蟲"
      ]
    },
    {
      "version": "1.0",
      "minimumUOFXVersion": "2.89.1",
      "changelog": [
        "全新對話功能上線！"
      ]
    }
  ]
}
```

`version` 的版本需和 `nginx` 的 `location /1_0/` 相對應，1_0 代表版本 1.0。以此類推，若是需要使用 2.0 版本則需要再新增 `location /2_0/` 來對應。

nginx.conf

```
http {
  # 其他設定省略...
```

```
upstream Remote {
    server localhost:40001;
}

# 主要設定
server{

    # 對外要使用的port
    listen 8888;

    location /1_0/ {
        proxy_pass http://Remote;
    }

    location / {
        proxy_pass http://Remote;
    }
}
```

💡 詳細設定請參考 [開發環境設定](#)

設定檔共用規則

- `icon` 設定相對路徑，如下範例中設定為 `assets/icons/u-plugin-form.svg`，於執行階段會依據版本組成 `http://plugin/1_0/assets/icons/u-plugin-form.svg`。

欄位設計檔 **fields-design.json**

`fields-design.json` 內的欄位代表在 BPM 表單設計時，左方可用的欄位類型，依照自訂的 `group` 進行分類。

- `fieldGroups.code` 與 `fields.group` 需對應。
- `fieldGroups.code` 須為唯一值。
- `fields.code` 與 **fields-runtime.json** 中唯一名稱需對應。
- `fields.icon` 請參考 [共用規則](#)。

fields-design.json

```
{
  "$schema": "../..../node_modules/@uofx/plugin/schema/fields-design.schema.json",
  "fieldGroups": [
    {
      "code": "group1",
      "name": "群組一"
    }
  ]
}
```

```

],
"fields": [
  {
    "group": "group1",
    "code": "sampleField",
    "name": "sample欄位",
    "icon": "assets/icons/u-plugin-form.svg",
    "sizeConfig": {
      "defaultCols": 2,
      "defaultRows": 1,
      "minCols": 1,
      "minRows": 1
    }
  },
  {
    "group": "group1",
    "code": "anotherField",
    "name": "進階欄位",
    "icon": "assets/icons/u-plugin-form.svg",
    "sizeConfig": {
      "defaultCols": 8,
      "defaultRows": 2,
      "minCols": 8,
      "minRows": 2
    }
  }
]
}

```

欄位執行設定檔 **fields-runtime.json**

fields-runtime.json 的設定為表單開啟後讀取的外掛欄位內容。

- sampleField 為提供上方 **fields-design.json** 中所設定的 fields.code 使用。
- exposedModule 與 moduleName 為 **webpack-exposes.config.js** 所自訂的名稱及檔案來源位置內的 moduleName，兩個參數與 web 中的對應，app 則與 app 的內容對應。

fields-runtime.json

```

{
  "$schema": "../..../node_modules/@uofx/plugin/schema/fields-runtime.schema.json",
  "sampleField": {
    "exposedModule": "./HelloWorld",
    "moduleName": "HelloWorldModule",
    "app": {
      "exposedModule": "./HelloWorld",
      "moduleName": "FieldHelloWorldAppModule"
    }
  }
}

```

```

}
},
"anotherField": {
  "exposedModule": "./AdvancedField",
  "moduleName": "AdvancedFieldModule",
  "app": {
    "exposedModule": "./AdvancedField",
    "moduleName": "FieldAdvancedAppModule"
  }
}
}
}
}

```

外掛頁面路由與選單設定 routes.json

- menu.icon 與 children.icon 請參考 [共用規則](#)。
- menu.name 為主選單名稱。
- funcId 為唯一值，admin、user 與 app 皆不可重複，用途請參考 [設定 Plugin Page 權限](#)。
- path 為路由路徑，設定內容請參考 [開始 Web 外掛頁面>路由設定](#)

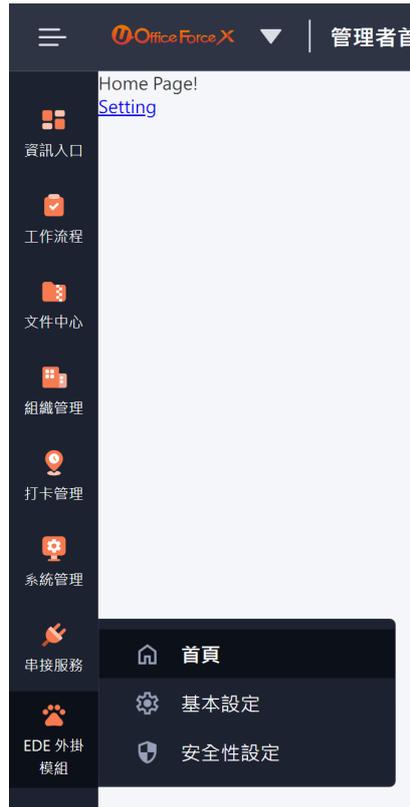
fields-runtime.json

```

{
  "$schema": "../../node_modules/@uofx/plugin/schema/routes.schema.json",
  "admin": {
    "menu": {
      "name": "EDE 外掛模組",
      "icon": "assets/icons/pets.png",
      "children": [
        {
          "funcId": "HOME",
          "name": "首頁",
          "icon": "assets/icons/home.png",
          "path": "home"
        },
        {
          "funcId": "SETTING",
          "name": "基本設定",
          "icon": "assets/icons/settings.png",
          "path": "setting"
        },
        {
          "funcId": "SETTING_SECURITY",
          "name": "安全性設定",
          "icon": "assets/icons/security.png",
          "path": "setting/security"
        }
      ]
    }
  }
}

```

```
]
}
},
"user": {
  "menu": {
    "name": "EDE 外掛模組",
    "icon": "assets/icons/pets.png",
    "children": [
      {
        "funclId": "LOBBY",
        "name": "大廳",
        "icon": "assets/icons/house.png",
        "path": "lobby"
      },
      {
        "funclId": "HOWTO",
        "name": "如何快速開始",
        "icon": "assets/icons/rocket_launch.png",
        "path": "howto"
      }
    ]
  }
},
"app": {
  "menu": {
    "name": "EDE 外掛模組",
    "icon": "assets/icons/pets.png",
    "children": [
      {
        "funclId": "LOBBY",
        "name": "大廳",
        "icon": "assets/icons/house.png",
        "path": "lobby"
      }
    ]
  }
}
}
```



管理者端主選單顯示結果



使用者端主選單顯示結果

功能列表

資訊入口



工作流程



外掛模組



手機端主選單顯示結果

2.4 開發環境

UOF X 掛載 Plugin 外掛欄位的方式是透過 BPM 表單設計時由左方的欄位類型清單拖曳外掛欄位進入表單畫布中，透過 nginx 的轉址服務讓 UOF X 的 BPM 表單服務能讀取本機建置的外掛欄位進行開發。



UOF X 外掛欄位執行架構

調整 nginx 的 config 檔案內容設定

透過 vscode 或是 notepad 記事本打開 nginx 的 config 檔案(nginx.conf)。

外掛模運行預設是 localhost:40001，透過調整 nginx 的轉址設定，讓本機執行外掛模組專案時，讓 UOF X 的 BPM 模組功能設定能讀取本機站台的 Plugin 內容；透過以下的設定內容，完成執行 nginx.exe，外掛站台將會執行在，例如：<http://本機host:8899/>。

nginx.conf

nginx conf 內容設定

```
events {
    worker_connections 1024;
}

http {
    include mime.types;
    default_type application/octet-stream;

    sendfile    on;
    keepalive_timeout 65;
```

```
upstream Remote {
    server localhost:40001;
}

# 主要設定
server{

    #對外要使用的port
    listen 8899;

    location /1_0/ {
        proxy_pass http://Remote/;
    }

    location / {
        proxy_pass http://Remote/;
    }
}
```

Plugin 新增主檔

設定完成可在 UOF X 中的串接服務的 Plugin 管理功能掛上外掛欄位模組，可參閱 [Plugin 管理](#)。

UOF X 在取得版本內容的方式是依照輸入的 host 後面直接加上 1_0 或 2_0，所以依照上方 nginx 的設定，host 最後面需要加 /，例如: <http://172.16.3.222/>，這樣執行時就會去取得 http://172.16.3.222/1_0/ 下的內容；如果部署在 IIS 方式設定那就不用加上 /，詳細請參考 [如何部署](#)。

新增Plugin



Plugin位置*

Plugin程式皆源於外部，因此會透過「Plugin位置」的設定，找到對應的外部程式來執行

請輸入網址 例：https://the.example.com

3. ★外掛欄位

3.1 簡介與實作

欄位檔案結構

基本的網站原始碼皆放置在 **Ede.Uofx.Customize.Web** 內。

```

├─ ClientApp (前端 Angular)
│  └─ src
│     └─ app
│        ├── mobile (放置手機版內容)
│        ├── service (基礎發送至後端API請求服務)
│        ├── shared (放置共用內容, EX: Props 模型)
│        ├── web (放置網頁版內容)
│        └─ app.module.ts
│  └─ assets
│     └─ configs
│        ├── fields-design.json (設定欄位樣式)
│        ├── fields-runtime.json (設定欄位執行時使用的程式)
│        └─ routes.json (設定各端的外掛頁面選單)
│  └─ plugin.manifest.json (模組包機處配置檔)
└─ webpack-exposes.config.js (設定前端 Module 建置後的別名)
└─ Controllers (後端 API)

```

- 實作 web 外掛欄位時可參考從 Sample 下載的專案檔內 src/app/web/hello-world 中的檔案結構，自行建立一樣的檔案結構內容，需有 design|props|write|print|view 這幾項目錄結構。
- 實作 app 外掛欄位可參考 src/app/mobile/hello-world 中檔案結構自行建立一樣的檔案結構內容，需有 design|props|write|print|view 這幾項目錄結構。

外掛欄位 MODULE.TS

外掛欄位實作時除了基本的模式之外也要設定外掛欄位本身的 **module.ts**(例如: hello-world.module.ts) 來定義外掛欄位所需的模組和對應讀取的外掛欄位 Component 設定。

hello-world.module.ts

```

@NgModule({
  imports: [
    CommonModule,
    FormsModule,
    ReactiveFormsModule,
    RouterModule.forChild([

```

```
{ path: "", redirectTo: "design", pathMatch: "full" },
{ path: "design", component: HelloWorldDesignComponent },
{ path: "props", component: HelloWorldPropsComponent },
{ path: "write", component: HelloWorldWriteComponent },
{ path: "view", component: HelloWorldWriteComponent },
{
  path: "app",
  loadChildren: () =>
    import("../mobile/hello-world/hello-world.module").then(
      (m) => m.FieldHelloWorldAppModule
    ),
},
]),
TranslateModule.forChild(),
...UOF_MODULES,
],
})
export class HelloWorldModule {
  static comp = {
    props: HelloWorldPropsComponent,
    design: HelloWorldWriteComponent,
    write: HelloWorldWriteComponent,
    view: HelloWorldWriteComponent,
    print: HelloWorldWriteComponent,
  };
}
```

Note

實作外掛欄位時可先以設計模式(design)、屬性設定(props)與填寫模式(write)這三種情境的外掛欄位來建立最基礎的 BPM 表單。

五種欄位模式

✓ Design 設計模式

當從 BPM 表單設計時拖曳進入表單畫布時所顯示的外掛欄位；對應檔案結構在專案 ClientApp 目錄 `src/app/mobile/hello-world/design` 中。



Design 設計模式

✓ Props 屬性設定

從外掛欄位編輯屬性(Properties)時設定的畫面；對應檔案結構在專案 ClientApp 目錄 `src/app/mobile/hello-world/props` 中。

有外掛欄位的 草稿 新草稿

欄位編排 → 流程設計 → 結案後流程 → 發佈設定 診斷欄位 複製欄位 預覽

基本 組織 自訂欄位 104hr欄位 104hr欄位

請假欄位

O365Dynamic

客戶資料

104hr欄位

請假欄位

sample欄位

設定 權限

代號* C003

欄位名稱* sample欄位

申請者 非必填

內容

顯示hello world訊息

欄位說明

寫些說明...

Props 屬性模式

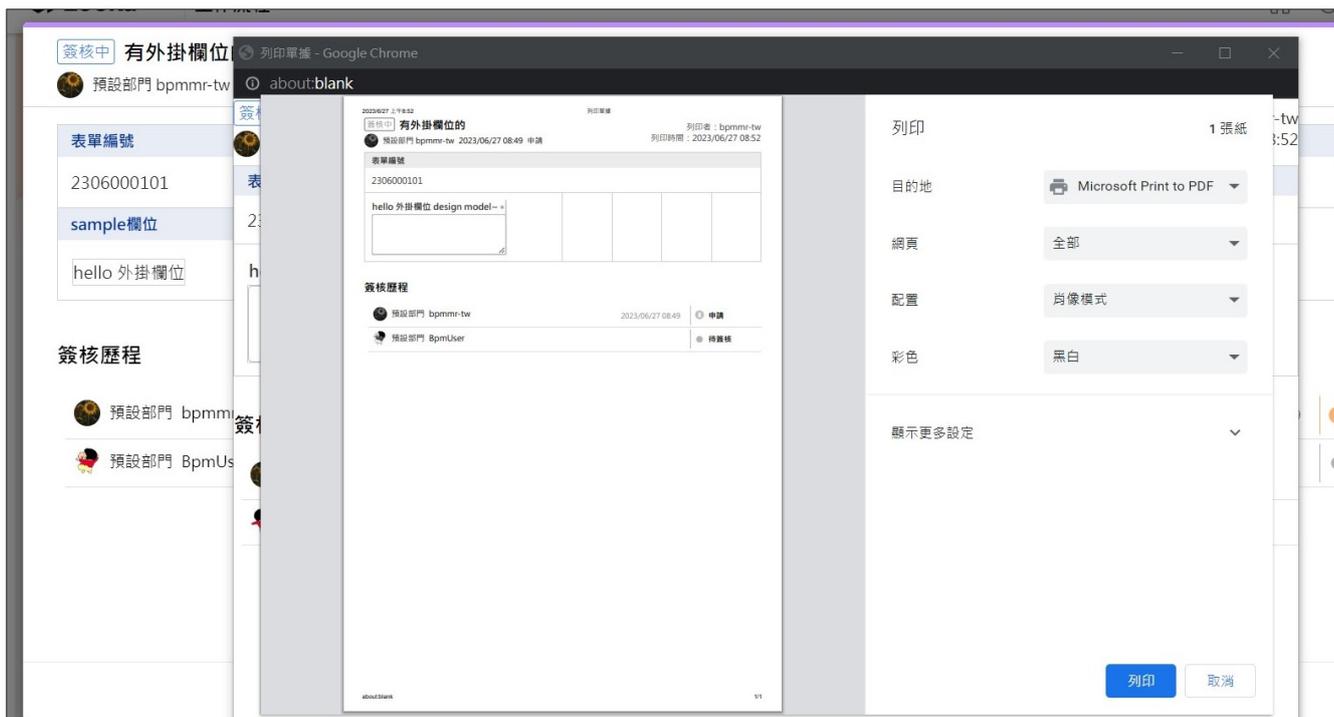
✓ Write 填寫模式

BPM 表單發佈後，使用者端申請表單或簽核時，填寫顯示的內容；對應檔案結構在專案 ClientApp 目錄 **src/app/mobile/hello-world/write** 中。

Write 填寫模式

✓ View 觀看 / Print 列印模式

- view(觀看)：BPM 表單申請或是簽核送出後，純觀看的模式，主要是為了欄位在觀看模式較為複雜時，可以額外特別設計，但也可不實作，直接使用 write 填寫模式作為觀看模式使用。
- print(列印)：BPM 表單觀看或簽核時，列印表單內容的模式，主要是為了欄位在觀看模式較為複雜時，可以額外特別設計，但也可不實作，直接使用 write 填寫模式作為觀看模式使用。



Print 列印模式

★開始實作

✓ Design Component

設計模式外掛欄位，當設計表單時從欄位列表拖曳進入表單畫布上顯示的樣式，需繼承 `BpmFwDesignComponent`，使用 `exProps` 接收傳入 component 的屬性值，可自行定義外掛欄位接收的屬性類型 `model`，例如：

`HelloExProps` 只定義了一個 `isShowHelloWorld` 屬性。

hello-world.design.component.ts

```
@Component({
  selector: "uofx-hello-world-design-component",
  templateUrl: "./hello-world.design.component.html",
  styleUrls: ["./hello-world.design.component.scss"],
})
export class HelloWorldDesignComponent extends BpmFwDesignComponent {
  @Input() exProps: HelloExProps;
}
```

hello.exprops-type.ts

```
export interface HelloExProps {
  isShowHelloWorld: boolean;
}
```

✓ Props Component

屬性設定模式的外掛欄位，透過這個外掛欄位可以自行定義表單使用上的屬性，例如單選鈕和下拉項目的選項值內容，需繼承 `BpmFwPropsComponent`。

hello-world.props.component.ts

```
@Component({
  selector: "uofx-hello-world-props-component",
  templateUrl: "./hello-world.props.component.html",
})
export class HelloWorldPropsComponent
  extends BpmFwPropsComponent
  implements OnInit
{
  form: FormGroup;
  @Input() exProps: HelloExProps;

  isShowHelloWorld: boolean;
  constructor(public fb: FormBuilder) {
    super(fb);
  }
}
```

```

ngOnInit(): void {
  this.initExProps();
}

initExProps() {
  if (!this.exProps) {
    // 初始化設定額外屬性
    this.exProps = {
      isShowHelloWorld: false,
    };
  } else {
    // 若已有存在的 exProps
    // 看是需要更新還是重設 value
  }
}
}
}

```

✓ Write Component

填寫模式的外掛欄位畫面 html 設定的部分,外層須有一個 `FormGroup` , 裡面在放上自己的 `FormControl` 欄位且可設定欄位的 `validator`。

- 使用 `uofx-form-field-name` 元件可與 UOF X 標題樣式一致 , `name` 與 `required` 變數薇 `BpmFwWriteComponent` 本身提供之變數。
- 使用 `fw-control` class 可以使框內內容撐開的空間與 UOF X 樣式一致。
- 使用 `fw-descr` class 可與 UOF X 標準欄位內的小字說明樣式一致。

💡 `fw-control` 與 `fw-descr` 樣式可參閱 [可用 CSS Class>欄位樣式](#)。

hello-world.write.component.html

```

<div>
  <uofx-form-field-name [name]="name" [required]="required">
  </uofx-form-field-name>
</div>
<div class="fw-control">
  <table [formGroup]="form">
    <tr>
      <!-- hello world 顯示歡迎訊息 -->
      <th colspan="3">
        <span *ngIf="exProps?.isShowHelloWorld ? true: false">hello 外掛欄位 write mode</span>
      </th>
    </tr>
    <tr>
      <td colspan="3">
        <ng-container *ngIf="editable; else elseTemplate5">

```

```

    <input pInputText FormControlName="message" />
    <uofx-form-error-tip [control]="form?.controls.message">
    </uofx-form-error-tip>
  </ng-container>
  <ng-template #elseTemplate5>
    <div class="textarea-content">{{ value?.message }}</div>
  </ng-template>
</td>
</tr>
</table>
<div class="fw-descr" *ngIf="showDescr && descr">
  {{ descr }}
</div>
</div>

```

填寫模式的外掛欄位需繼承 `BpmFwWriteComponent`，透過 `BpmFwWriteComponent` 可取得表單欄位的 `value`。

hello-world.write.component.ts

```

@Component({
  selector: "uofx-hello-world-write-component",
  templateUrl: "./hello-world.write.component.html",
})
export class HelloWorldWriteComponent extends BpmFwWriteComponent implements OnInit
{
  @Input() exProps: HelloExProps;

  form: FormGroup;

  initForm() {
    this.form = this.fb.group({
      message: this.value?.message || "",
    });
  }
}

```

表單觸發儲存時，在前端驗證通過的情況下，外掛欄位可透過實作 `checkBeforeSubmit()` 再次做送出前的非同步檢查，`resolve(false)` 表示驗證失敗，可自行設計後續的錯誤處理與顯示。

hello-world.write.component.ts

```

/** 表單 submit 前的檢查 */
checkBeforeSubmit(): Promise<boolean> {
  return new Promise(resolve => {
    const value = this.form.value;
    console.log(value);
    resolve(true);
  });
}

```

```
});  
}
```

3.2 進階功能

取得欄位資訊

外掛欄位可以透過 [Plugin API](#) 來取得更多 UOF X 的使用者和公司資訊。

✓ 取得表單資訊

繼承 `BpmFwWriteComponent` 可以從外掛欄位的 `this.taskNodeInfo` 取得表單資訊。站點的各项資訊在申請狀態中無法取得。

參數名稱	型態	說明
<code>applicantId</code>	<code>string</code>	申請者 Id
<code>applicantDate</code>	<code>string</code>	申請時間
<code>nodeId</code>	<code>string</code>	站點 Node Id
<code>ownerId</code>	<code>string</code>	站點簽核者 id
<code>codeOfProcessSite</code>	<code>string</code>	站點代號
<code>variables</code>	<code>Array<BpmFwTaskVariableModel></code>	表單變數

BpmFwTaskVariableModel

參數名稱	型態	說明
<code>id</code>	<code>string</code>	Id
<code>code</code>	<code>string</code>	代碼
<code>name</code>	<code>string</code>	名稱
<code>value</code>	<code>string</code>	值

✓ 取得欄位資訊

- `id`: 欄位本身的 Id, 此資訊由系統自動產生。
- `code`: 欄位代號, 在表單維護中所設定的欄位代號。
- `name`: 欄位名稱, 在表單維護中所設定的欄位名稱。
- `descr`: 欄位說明, 在表單維護中所設定的欄位說明。
- `exProps`: 在屬性頁中所提供的 `exProps`。

- `editable`: 是否在表單維護中設定為可編輯。
- `required`: 是否在表單維護中設定為必填。
- `pluginSetting`: 目前僅提供 `entryHost` 可以取用，該值為站台主機位置。

上述欄位資料皆可直接使用 `this` 進行存取。

```
export class HelloWorldComponent extends BpmFwWriteComponent {
  ngOnInit() {
    // 在 console 中印出 Id、欄位代號與名稱
    console.log(this.id, this.code, this.name);
  }
}
```

設定欄位屬性

參數設定包含簽核、條件、欄位計算、主旨、匯出與搜尋條件，設定參數都需提供基礎的資訊，包含顯示名稱 `name` 與 `jsonPath`，`jsonPath` 的用途為對應到欄位的資料結構。

最早可設定的生命週期時間點為 `ngOnInit()`，透過繼承底層 `BpmFwPropsComponent` 後所提供的 `initPluginSettings` 來完成。

✓ 將所有參數同時設定

```
ngOnInit() {
  this.initPluginSettings({
    toBeNodes: [{ name: '職務代理人', jsonPath: 'agent'}],
    toBeConditions: [{ name: '類別名稱', jsonPath: 'categoryName', type: 'Text' }],
    toBeCalculates: [{ name: '總金額', jsonPath: 'total'}],
    toBeSubjects: [{ name: '總金額', jsonPath: 'total'}],
    toBeExports: [{ name: '總金額', jsonPath: 'total'}],
    searchContentJsonPath: 'productName'
  });
}
```

✓ 簽核站點

將可簽核的欄位作為簽核站點，通常為 `User Select` 選人元件，舉例來說，欄位中的職務代理人值存為 `value` 下的 `agent` 屬性，並使用 `agent` 做為簽核站點，設定範例如下：

```
ngOnInit() {
  this.initPluginSettings({
    toBeNodes: [{ name: '職務代理人', jsonPath: 'agent'}],
    ...
  });
}
```

設定完成後就能在設定流程時看到「來自外掛欄位」中出現該設定。

✓ 條件站點

若需要設定條件站點，則需要多提供 `type` 參數，因為設定條件時，會依照參考的目標值類型不同而有不同的條件選項，範例如下：

```
ngOnInit() {
  this.initPluginSettings({
    toBeConditions: [{ name: '類別名稱', jsonPath: 'categoryName', type: 'Text' }],
    ...
  });
}
```

```
});
}
```

type 總共有 4 種類型可以提供，分別為 Text (文字)、Numeric (數值)、Department (部門)、Employee (人員)。

✓ 欄位計算

這項設定是為了讓 UOF X 提供的標準欄位「欄位計算」可以選取到外掛欄位的 value。

比如說我們需要將總金額 total 在欄位計算中可以被設定選取作為運算中的變數，那麼就需要在 `initPluginSettings` 也同時設定 `toBeCalculates` 參數。

```
this.initPluginSettings({
  toBeCalculates: [{ name: '總金額', jsonPath: 'total'}],
  ...
});
```

✓ 表單主旨

比如需要將總金額 total 在欄位計算中可以被設定選取作為運算中的變數，那麼就需要在 `initPluginSettings` 也同時設定 `toBeSubjects` 參數。

```
this.initPluginSettings({
  toBeSubjects: { name: '總金額', jsonPath: 'total'},
  ...
});
```

前述有提到，可以設定隱藏欄位作為可用參數設定，我們就可以在 `write` 的元件中設定 `value.dateRange` 為特殊組合，結合開始日期和結束日期，顯示的結果可能類似於 `2023/06/08 9:00~2023/06/08 18:00`。

```
form: FormGroup;

constructor(private fb: FormBuilder) {
  this.form = this.fb.group({dateRange});
}

onDateChange() {
  // Get start date and end date...
  this.form.get('dateRange').setValue(...);
}
```

並在 `props` 元件中透過 `initPluginSettings` 設定。

```
this.initPluginSettings({
  toBeSubjects: { name: '日期區間', jsonPath: 'dateRange'},
```

```
...
});
```

✓ 表單匯出

由於 value 本身為巢狀結構，而 Excel 檔案為表格形式，須將所需匯出的資料攤平，轉成一般的文字或數值。

比如說我們在使用者端「表單查詢」的匯出功能，需要在匯出的 Excel 中包含總金額 total 欄位，那麼就需要在 `initPluginSettings` 設定 `toBeExports` 參數。

```
this.initPluginSettings({
  toBeExports: { name: '總金額', jsonPath: 'total'},
  ...
});
```

Note

可用的匯出參數設定與表單主旨一樣可以做成隱藏欄位唷！

✓ 表單搜尋條件

設定表單搜尋條件，可在使用者端的「表單查詢」中作為指定欄位內容被查詢，範例為設定查詢品項名稱 `productName`。

```
this.initPluginSettings({
  searchContentJsonPath: 'productName'
});
```

✓ 設定參數的建議方式

建議在 `ngOnInit` 的時間點做設定，並且判定 `exProps` 屬性還未被設定的狀態，這樣當 `props component` 每次被初始化的時候才不會重複一直執行。

Warning

但這種設計方式必須注意，當 `initPluginSettings` 有變更參數時，由於 `exProps` 已經有值，所以不會重新被初始化，這樣的寫法若需要重新初始化，需要將欄位重新拖入表單畫布中，等於是重新新增一次該欄位。

```
ngOnInit() {
  this.initExProps();
}

initExProps() {
  if (!this.exProps) {
```

```
// ...  
  
this.initPluginSettings({  
  toBeConditions:[...],  
  toBeNodes: [...]  
});  
} else {  
  // ...  
}  
}
```

 **Note**

在設定欄位可用參數時，除了可以存取自訂的可見欄位值，也可將所需要的值設定成隱藏狀態，就是實際上 value 會存，但是在申請或簽核時不可見。

送出前驗證

表單送出前的欄位驗證分為兩種，一是前端驗證，另一種是透過 API 等待結果決定是否可以送出。

✓ Import 相依

```
import { UofxFormFieldLogic } from '@uofx/web-components/form';
```

✓ 可用函式

函式名稱	說明
parentFormBinding	訂閱 parent form 的 status changes，送出時，一併顯示欄位內整張 form 的錯誤訊息
setSelfControlValue	放在 form.valueChanges 訂閱中，每次更新 selfControl 的值
checkValidators	放在 checkBeforeSubmit 中，如果是暫存就不需要驗證必填，且清除 form control error

✓ 設定前端驗證

可以透過「Angular」原本就支援的基本驗證或是使用 UOF X 元件的 [Validators 共用驗證](#) 元件，也可自訂欄位的驗證函式，設定在欄位初始化之後。

表單按下送出時，前端會先檢查 parentForm 是否通過驗證，而 selfControl 屬於 parentForm 的其中一個 control，所以當 selfControl 驗證不過的時候，會直接阻擋 parentForm 驗證。

advance-field.write.component.ts

```
initForm() {
  console.log('initForm', this.exProps);
  this.form = this.fb.group({
    'empNo': [this.value?.empNo ?? "", Validators.required],
    'mobile': [this.value?.mobile ?? "", [createMobileValidator(), Validators.minLength(10)]],
    'applyDate': [this.value?.applyDate, [Validators.required,
    createApplyDateValidator(this.exProps.checkDays)]]
  });

  // 表單送出時的前端驗證
  if (this.selfControl) {
    // 在此便可設定自己的驗證器
    this.selfControl.setValidators(validateSelf(this.form));
    this.selfControl.updateValueAndValidity();
  }
}
```

```
}
}
```

✓ 顯示欄位內的驗證失敗訊息

由於使用 `uofx-form-error-tip` 需要在 `control.dirty` 為 `true` 的情況下才會顯示，可以在 `ngOnInit()` 時呼叫 `parentFormBinding()`，這樣在驗證失敗時，會一併將自定義的表單 `markAsDirty`。

使用方式如下：

```
import { Component } from '@angular/core';
import { UofxFormFieldLogic } from '@uofx/web-components/form';

@Component({
  ...
})
export class LogicDemo {
  form: FormGroup;

  constructor(private logic: UofxFormFieldLogic)

  ngOnInit() {
    ...

    this.logic.parentFormBinding(this.parentForm, this.selfControl, this.form);
  }
}
```

✓ 設定對應的欄位值

當欄位資料異動時，更新 `selfControl`，讓跨欄位存取可拿到最新的資料。

```
import { Component } from '@angular/core';
import { UofxFormFieldLogic } from '@uofx/web-components/form';

@Component({
  ...
})
export class LogicDemo {
  form: FormGroup;

  constructor(private logic: UofxFormFieldLogic)

  ngOnInit() {
    ...

    this.formSubscription$ = this.form.valueChanges.subscribe(res => {
```

```

    this.formFieldLogic.setSelfControlValue(this.selfControl, this.form, res);
  });
}
}

```

✓ 設定後端驗證

表單本身的設計驗證分成兩段，確定前端驗證都通過後，才會在下個階段前進行後端 API 驗證。

`checkValidator` 設定按下表單下方按鈕時是否要檢查表單驗證。

在欄位填寫模式內實作 `checkBeforeSubmit()`，回傳結果 `Promise<boolean>`，`resolve(true)` 表示通過驗證，`resolve(false)` 表示未通過驗證，回傳 `false` 會阻擋表單送出。

advance-field.write.component.ts

```

/** 表單submit前要做的檢查 */
checkBeforeSubmit(checkValidator: boolean): Promise<boolean> {
  return new Promise(resolve => {
    const value = this.form.value;
    console.log(value);

    if (checkValidator) {
      // 如果要檢查表單驗證且表單 invalid，回傳 false 阻擋表單送出
      this.form.invalid ? resolve(false) : resolve(true);
    } else {
      // 表單不須驗證，直接回傳 true 暫存表單
      resolve(true);
    }
  });
}

```

✓ 在暫存狀態下，不需要驗證欄位資料

`checkBeforeSubmit` 時如果是暫存就不需驗證必填，會清除所有 `control error`。

```

import { Component } from '@angular/core';
import { UofxFFormFieldLogic } from '@uofx/web-components/form';

@Component({
  ...
})
export class LogicDemo {
  form: FormGroup;

  constructor(private logic: UofxFFormFieldLogic)

  checkBeforeSubmit() {

```

```
...  
  
this.formFieldLogic.checkValidators(checkValidator, this.selfControl, this.form);  
}  
}
```

串接遠端 API

✓ 建立自訂 Service

- 建立自訂的 API Service 需要繼承 `BasicApiService`，傳入要呼叫的 url。
- 範例中使用 `~/api` 開頭是因為 `BasicApiService` 會將覆蓋 `~` 波浪號的部分與 `serverUrl` 進行網址串接。

employee.service.ts

```
export class EmployeeService extends BasicApiService{
  /**
   * 取得合法的員工編號
   * @returns 合法的員工編號
   */
  getValidEmpNumber(){
    const url= '~/api/emp/validemp';
    // 呼叫之前需先設定serverUrl
    console.log(this.http.serverUrl);
    return this.http.get<string[]>(url);
  }
}
```

- 設定 API Service 的 `serverUrl` 時需要使用 `pluginSetting` 物件取得 `entryHost` 站台位址，此站台位址為 [Plugin 管理](#) 中您所輸入的 Plugin 位置。

advance-field.write.component.ts

```
// 呼叫 API 之前要設定 serverUrl 為外掛欄未站台位址
this.empService.serverUrl = this.pluginSetting?.entryHost;
// 呼叫 API 檢查是否是有效的帳號
this.empService.getValidEmpNumber().subscribe(res => {
  // 處理 res 回傳值...
});
```

API 身分驗證

✓ 外掛欄位 API 驗證流程

外掛欄位 API 驗證流程先透過自訂的 `httpHandler` 處理加上和 API 溝通的 `signature`，送給 API 處理比對通過後再回傳 API 的 `response` 給前端。



API 驗證流程

✓ 外掛欄位自訂的 `HttpHandler` 處理

外掛欄位前端可在 `NgModule` 中加入自訂的 `http` 控制呼叫 API 時處理 `Request header` 資訊在送給 API 後端驗證。

advanced-field.module.ts

```

@NgModule({
  imports :[
    CommonModule,
    FormsModule,
    ReactiveFormsModule
  ],
  providers: [
    UofxPluginApiService,
    { provide: BASIC_HTTP_HANDLER, useClass: EmployeeHttpHandler },
    BasicHttpClient,
    EmployeeService
  ]
})
  
```

`EmployeeHttpHandler` 處理呼叫後端 API 時在 `Request header` 一律加上和 API 驗證使用的簽章內容，透過前後端溝通好的 `key` 對時間用「`HMACSHA256`」產出簽證(signature)後再將資訊放到 `header` 上傳到 API 進行比對。

employee.handler.ts

```

/** 設定 Http Request Header */
private setSignatureHeader(): HttpHeaders {
  const message = new Date().toISOString();
  
```

```

const apiKey = 'SampleAdvanced';
const signature = CryptoJS.HmacSHA256(message, apiKey).toString();

return new HttpHeaders({
  'X-Timestamp': message,
  'X-Signature': signature,
});
}

```

✓ 外掛欄位 API Middleware 驗證

Http request 傳送過來 API 後端時，外掛欄位加上檢查 Http header 的「Middleware」，驗證外掛欄位傳送的簽章資訊是否合法。

Program.cs

```

app.UseRouting();

app.UseCors(MyAllowSpecificOrigins);
app.UseMiddleware<ApiSignatureMiddleware>();

```

ApiSignatureMiddleware.cs

```

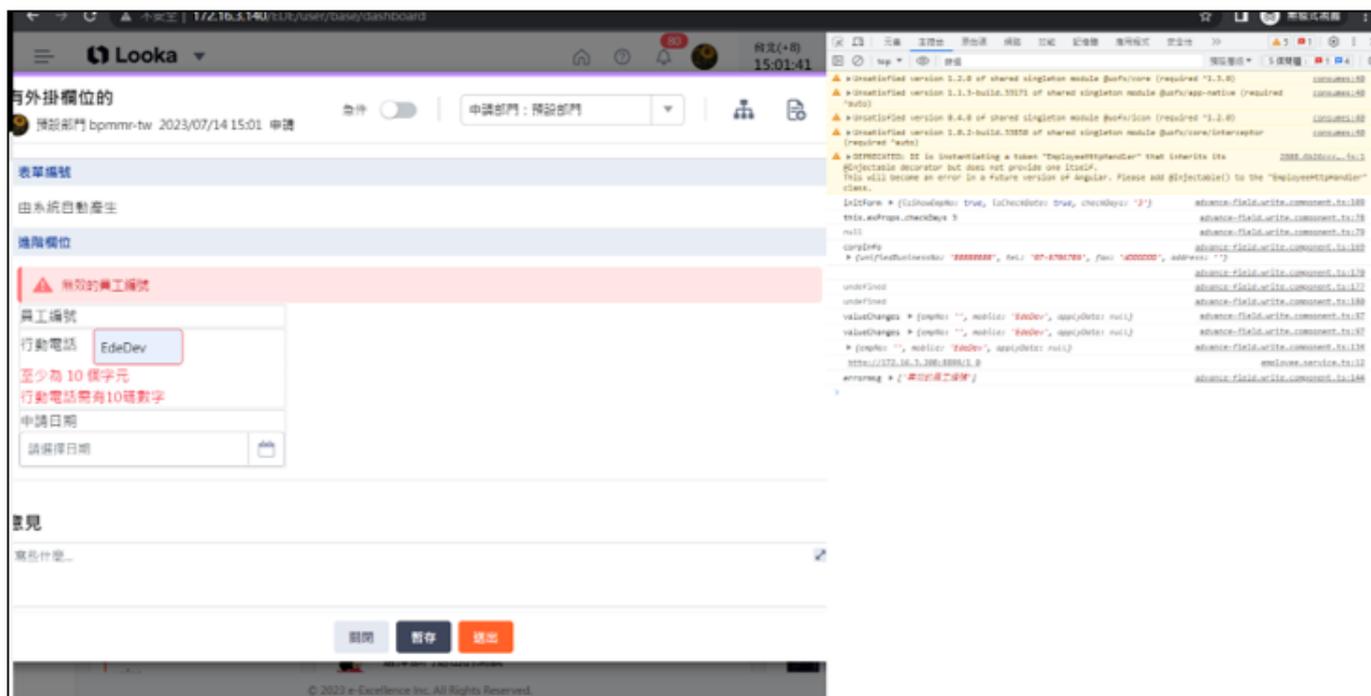
public async Task Invoke(HttpContext context)
{
  if( !context.Request.Headers.TryGetValue("X-Signature", out var signature))
  {
    context.Response.StatusCode = 401;
    await context.Response.WriteAsync("signature was not provided");
    return;
  }
  // 首先將它們依照指定格式組成字串後，
  // 接下來拿 API Key 作為 Key 使用 HMAC - SHA256 計算出 Signature
  var compareSignature = times.ToString().HMACSHA256(apikey);
  var requestSignature = signature;
  if (requestSignature != compareSignature)
  {
    context.Response.StatusCode = 401;
    await context.Response.WriteAsync("Signature verify fail.");
    return;
  }
}

```

如何 Debug

✓ 透過瀏覽器開發人員工具

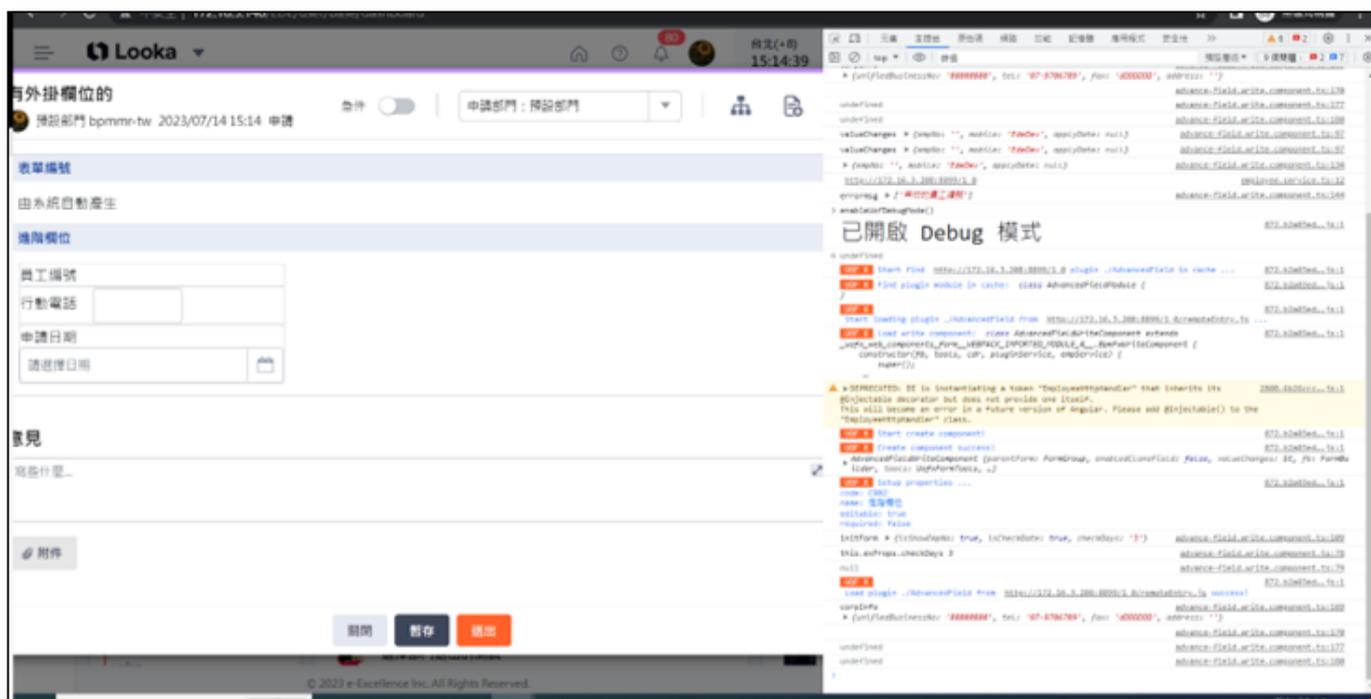
可打開瀏覽器的開發人員工具(F12)，透過主控台的頁籤功能中觀察前端是否已有錯誤訊息，也可在外掛欄位的*.ts檔案中寫 `console.log()`，偵測錯誤原因。



Chrome 開發人員工具

✓ 啟用外掛欄位偵錯模式

由開發人員工具的主控台中可輸入指令 `enableUofDebugMode()`，可查看 UOF X 載入外掛欄位時成功或失敗相關訊息內容。



啟用外掛欄位偵錯模式

3.3 跨欄位互動

取得其他欄位值

在欄位互動中，可以透過欄位的「代號」取得其他欄位的值。

取值需要主動觸發，無法透過訂閱的形式完成，取值使用 `getTargetFieldValue`，回傳為 `Promise<any>` 型別，以下範例為取得其他欄位代號為 `C003` 的內容。

```
onGetValueClick() {  
  this.getTargetFieldValue('C003').then(res => {  
    this.fieldValue = res;  
  });  
}
```

或是使用 `await` 的方式。

```
async onGetValueClick() {  
  this.fieldValue = await this.getTargetFieldValue('C003');  
}
```

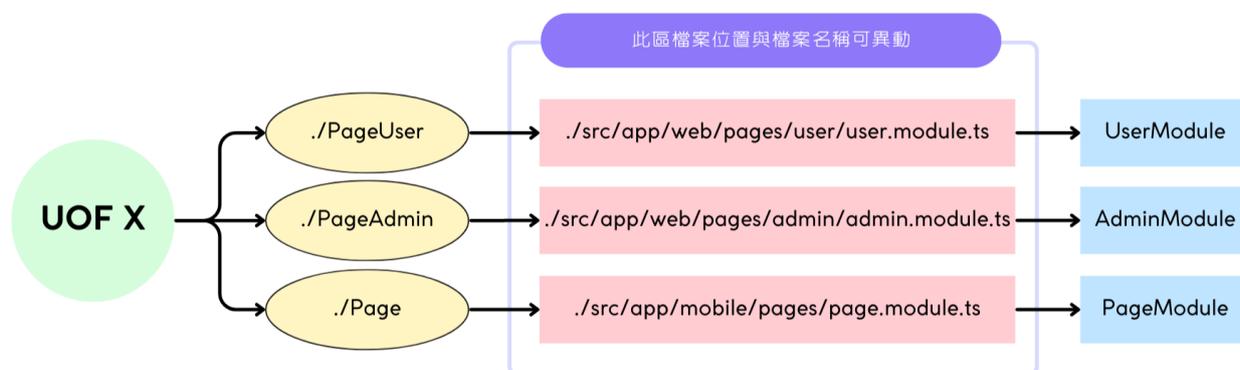
欄位的回傳值會依照不同欄位而回傳不同類型，可能為 `string`、`boolean` ... 等，也可能為純物件；除了標準欄位外，當然也可以取得其他外掛欄位的值。

Note

目前並沒有提供方法訂閱目標欄位內容變更。

4. ★外掛頁面

4.1 頁面架構



外掛頁面於 `webpack-exposes.config.js` 內的設定為單一 `module` 進入點，所以管理者、使用者與手機端共只有 3 個進入點、3 份 `module` 檔案，且名稱固定，僅在專案內檔案位置可自行定義。

webpack-exposes.config.js

```
const exposes = {
  web: {
    // ./PageAdmin 名稱固定，為管理者端進入點
    './PageAdmin': './src/app/web/pages/admin/admin.module.ts',
    // ./PageUser 名稱固定，為使用者端進入點
    './PageUser': './src/app/web/pages/user/user.module.ts',
  },
  app: {
    // ./Page 名稱固定，為手機端進入點
    './Page': './src/app/mobile/pages/page.module.ts',
  }
};
```

檔案位置可以任意搬動，也可修改檔案名稱，但檔案內的 `module` 名稱不可更動 - 管理者端須為 `AdminModule` - 使用者端須為 `UserModule` - 手機端為 `PageModule`

進入點設置方式

以 `UserModule` 為例，在空路由的情況下，即 `/user/plugin/[codePath]` 會自動導至 `lobby` 大廳頁面。

user.module.ts

```
import { LobbyPage } from "../lobby.page";
import { NgModule } from "@angular/core";
```

```
import { RouterModule } from "@angular/router";

@NgModule({
  imports: [
    RouterModule.forChild([
      { path: "", redirectTo: "lobby", pathMatch: "full" },
      { path: "lobby", component: LobbyPage },
    ]),
  ],
  declarations: [LobbyPage],
})
export class UserModule {}
```

4.2 Web 外掛頁面

新增外掛頁面

1. 新增檔案 **lobby.page.ts**。
2. 設定 `@UofxPluginAuthorize()`，並指定唯一的 `functionId`。

lobby.page.ts

```
import { Component } from "@angular/core";
import { UofxPluginAuthorize } from "@uofx/plugin";

@UofxPluginAuthorize({ functionId: "LOBBY" })
@Component({
  selector: "uofx-web-lobby",
  template: `<h1>Lobby</h1>`,
})
export class LobbyPage {}
```

設定路由

管理者端新增在 **admin.module.ts**，使用者端則在 **user.module.ts**，範例中為設定空路徑導向 lobby，lobby 會載入 `LobbyPage`。

user.module.ts

```
import { LobbyPage } from "../lobby.page";
import { NgModule } from "@angular/core";
import { RouterModule } from "@angular/router";

@NgModule({
  imports: [
    RouterModule.forChild([
      { path: "", redirectTo: "lobby", pathMatch: "full" },
      { path: "lobby", component: LobbyPage },
    ]),
  ],
  declarations: [LobbyPage],
})
export class UserModule {}
```

✓ 使用 lazy load module

新增其他 `module` 與 `page`，如同一般在使用 Angular lazy module 的方法，下方為範例新增一支 `options.module.ts`。

user.module.ts

```
imports: [
  RouterModule.forChild([
    { path: "", redirectTo: "lobby", pathMatch: "full" },
    { path: "lobby", component: LobbyPage },
    {
      path: "options",
      loadChildren: () => import("./options.module").then((m) => m.OptionsModule),
    },
  ]),
],
declarations: [LobbyPage],
})
export class UserModule {}
```

⚡ Danger

目前暫不支援使用 standalone component 並設定 loadComponent 載入，此方法會導致 Page 驗證失效。除非所設定的 Page 不需要通過身份驗證皆可查看。

設定 routes.json

在 `routes.json` 中設定的 `functionId`，於 Plugin 功能管理中設定可用的特定使用者，則特定使用者權限中即包含該 `functionId`，便可正確進入該相同於 `@UofxPluginAuthorize` 所設定的功能頁面。

以下方 menu 設定方式為例，在 Plugin 功能管理中設定選單「大廳」為 user1 可以使用，則 user1 可以看到選單「EDE 外掛模組」/「大廳」，點選大廳會導向 `/user/plugin/{code}/lobby`，lobby 路徑於 `user.module.ts` 指定為讀取 `LobbyPage`，此時 `LobbyPage` 所設定的 `functionId` 為 `LOBBY` 則可進入使用。

即 `LobbyPage` 若因設定錯誤 `functionId` 為 `LOBBY_ONE`，那該功能會無法進入使用。

routes.json

```
{
  "user": {
    "menu": {
      "name": "EDE 外掛模組",
      "icon": "assets/icons/pets.png",
      "children": [
        {
```

```

    "funcId": "LOBBY",
    "name": "大廳",
    "icon": "assets/icons/house.png",
    "path": "lobby"
  }
]
}
}
}
}

```

✓ 設定 Icon

可參考 [配置與設定>設定檔共用規則](#)

ICON 設計規範

此規範是為了符合 UOF X 風格，若不需要風格一致，可忽略。

使用者端

主選單 子選單

項目	規格
大小	22px
主色	#FFC2AF
輔色	#FFC2AF
邊界留白	2~4px
項目	規格
大小	22px
主色	#999999
輔色	-
邊界留白	2~4px

管理者端

主選單 子選單

項目	規格
大小	22px
主色	#FFC2AF
輔色	#FFC2AF
邊界留白	2~4px
項目	規格
大小	22px
主色	#97A1B7
輔色	-
邊界留白	2~4px

✓ 設定 Path

Path 設定為相對路徑，路徑設定方式皆與原 Angular router 設定方式相同。

單層設定 多層設定

```
"path": "setting"
```

```
"path": "setting/security"
```

為避免各個獨立的 Plugin 衝突，所以皆有保留路徑。

- 使用者端: /user/plugin/{codePath}
- 管理者端: /admin/plugin/{codePath}

codePath 為由 PluginCode 將 . 移除，並全部變成小寫轉換而成，如下：

- Ede.Sample.Plugin > /user/plugin/edesampleplugin
- Ede.HRSystem -> /user/plugin/edehrsystem

如上方所設定之範例，連結組合為 /admin/plugin/edesampleplugin/setting 與 /admin/plugin/edesampleplugin/setting/security。

 因路徑名稱較長，且相對路徑使用不便，可改用 [Plugin 套件/頁面工具](#)

4.3 APP 外掛頁面

新增外掛頁面

新增方式基本與網頁端相同，以下謹說明不同之處。

Module 為了符合 UOF X 底層所使用的 IonicFramework，所以為了符合手機端操作方式需要在 Module 內引用。

```
page.module.ts

import { NgModule } from '@angular/core';
import { IonicModule } from '@ionic/angular';

@NgModule({
  imports: [
    IonicModule,
    ...
  ]
})
export class PageModule { }
```

設定路由

手機端新增在 `page.module.ts`。

設定 routes.json

權限配置方式與網頁端相同，於 UOF X 中管理的位置也一樣。

僅手機端的主選單與網頁端不同，因設計版面關係，menu 下未提供 icon 可設置，只提供子選單的設定。

```
routes.json

{
  "app": {
    "menu": {
      "name": "EDE 外掛模組",
      "children": [
        {
          "funcId": "LOBBY",
          "name": "大廳",
          "icon": "assets/icons/house.png",
          "path": "lobby"
        }
      ]
    }
  }
}
```

```
}  
}
```

✓ 設定 Icon

可參考 [配置與設定>設定檔共用規則](#)

ICON 設計規範

此規範是為了符合 UOF X 風格，若不需要風格一致，可忽略。

項目	規格
大小	24px
主色	#FF6128
輔色	-
邊界留白	2~4px

✓ 設定 Path

手機端的保留路徑: /home/plugin/{codePath}。

5. 共用元件

5.1 Web

Button 按鈕

使用與 UOF X 內同款的按鈕元件。

✓ Import 相依

```
import { UofxButtonModule } from '@uofx/web-components/button';
```

✓ 使用方式



```
<uofx-button [mode]="u-btn-primary">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Primary</span>
</uofx-button>
```

```
<uofx-button [mode]="u-btn-second">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Second</span>
</uofx-button>
```

```
<uofx-button [mode]="u-btn-third">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Third</span>
</uofx-button>
```

```
<uofx-button [mode]="u-btn-negative">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Negative</span>
</uofx-button>
```

```
<uofx-button [mode]="u-btn-cancel">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Cancel</span>
</uofx-button>
```

```
<uofx-button [mode]="u-btn-upload">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
```

```

<span>Upload</span>
</uofx-button>

<uofx-button [mode]="u-btn-empty">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Text</span>
</uofx-button>

<uofx-button [mode]="u-btn-empty" [underline]="true">
  <span>underline</span>
</uofx-button>

```

✓ Outline



將 `uofx-button` 屬性 `outlined` 為 `true`。

```

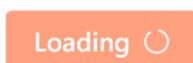
<uofx-button [mode]="u-btn-primary" [outlined]="true">
  <uofx-icon uofName="mi-add" uofSize="15"></uofx-icon>
  <span>Outlined</span>
</uofx-button>

<uofx-button [mode]="u-btn-third" [outlined]="true">
  <uofx-icon uofName="u-selector" uofColor="gb-500" uofSize="20"></uofx-icon>
</uofx-button>

<uofx-button [mode]="u-btn-third" [outlined]="true">
  <uofx-icon uofName="mi-add" uofColor="gb-500" uofSize="20"></uofx-icon>
</uofx-button>

```

✓ Loading



可以透過設定 `uofx-button` 屬性 `loading` 為 `true` 來讓按鈕顯示轉圈載入中的樣式。

```

<uofx-button
  [mode]="u-btn-primary"
  [loading]="loading"
  (click)="onLoadClick()">
  <span>Loading</span>
</uofx-button>

```

```
import { Component } from '@angular/core';

@Component({
  selector: 'button-loading-demo',
  templateUrl: './button-loading-demo.html'
})
export class ButtonLoadingDemo {
  loading = false;

  onLoadClick() {
    this.loading = true;

    setTimeout(() => {
      this.loading = false
    }, 2000);
  }
}
```

✓ Size



預設為上圖右方標準尺寸，若需要較小的按鈕，將 `uofx-button` 屬性 `smallSize` 設定為 `true`。

```
<uofx-button [mode]="u-btn-second" [smallSize]="true">
  <span>Small</span>
</uofx-button>
<uofx-button [mode]="u-btn-second">
  <span>Normal</span>
</uofx-button>
```

✓ Disabled



將 `uofx-button` 的 `disabled` 屬性設定成 `true` 就能變成禁用狀態，且游標會顯示禁止符號。

```
<uofx-button [disabled]="true">
  <uofx-icon uofName="mi-add" uofSize="15" uofColor="or-600"></uofx-icon>
  <span>disabled</span>
</uofx-button>
```

✓ Round

純 Icon 圓形按鈕。



```
<uofx-rounded-button  
  [iconName]="mi-more-horiz"  
  [iconColor]="gb-500"  
  [iconSize]="20">  
</uofx-rounded-button>
```

Error Message 錯誤訊息

UOF X 所提供的錯誤訊息包含「純紅色文字錯誤訊息」與「紅底色條列式錯誤訊息方塊」。

兩種錯誤提示都在 `UofxFormModule` 內。

✓ Import 相依

```
import { UofxFormModule } from '@uofx/web-components/form';
```

✓ 純紅色文字錯誤訊息

使用方式是提供一個 `control` 給元件，`control` 可以接受 `FormControl` 或 `AbstractControl` 類型，而訊息出現的條件為 `control.dirty` 為 `true` 且有 `control.errors` 的狀況下。

```
<input pInputText class="width-100" formControlName="textboxCtrl">
<uofx-form-error-tip [control]="form.controls.textboxCtrl"></uofx-form-error-tip>
```

```
this.form = this.fb.group({
  textboxCtrl: [null, Validators.required]
});
```

可用的預設錯誤訊息

說明	顯示條件 (hasError)	置換預設訊息 (properties)
必填	<code>required</code>	<code>requiredText</code>
上傳檔案必填	<code>fileRequired</code>	<code>fileRequiredText</code>
輸入重複	<code>duplicate</code>	<code>duplicateText</code>
資料不存在	<code>notExist</code>	
格式不正確	<code>email</code> 、 <code>pattern</code>	<code>formatIncorrectText</code>
超出範圍	<code>min</code> 、 <code>max</code>	<code>rangeText</code>
最大最小長度限制	<code>minLength</code> 、 <code>maxLength</code>	<code>minLengthText</code> 、 <code>maxLengthText</code>
只允許字母及數字	<code>onlyDigitsOrLetters</code>	<code>onlyDigitsOrLettersText</code>
比較值內容不同	<code>notEqual</code>	<code>notEqualText</code>

修改錯誤訊息內容

舉例來說，可以在需要提示必填的情況下，將預設文字改為「身分證必填」，可以設定 `requiredText`，如此一來在 `uofx-form-error-tip` 內偵測到 `control.hasError('required')` 時就會顯示改訊息了。

```
<uofx-form-error-tip
  [control]="textboxCtrl"
  [requiredText]="身分證必填">
</uofx-form-error-tip>
```

✓ 紅底色條列式錯誤訊息方塊

錯誤訊息方塊為條列式呈現，所以接收的參數為陣列形式。

屬性名稱	說明
<code>showClose</code>	是否顯示右方的 [x] 關閉按鈕
<code>showHeader</code>	是否顯示錯誤訊息區塊標題
<code>textEllipsis</code>	是否限制內容字數

```
transErrorCodes = ['輸入的訊息內容格式不正確'];
```

```
<uofx-error-block
  [uofErrorI18nKeys]="transErrorCodes"
  [showClose]="true"
  [showHeader]="true"
  [textEllipsis]="true">
</uofx-error-block>
```

Dialog 開窗

開啟在 UOF X 內置的視窗。

✓ Import 相依

```
import { UofxDialogModule } from '@uofx/web-components/dialog';
```

✓ 基礎使用方式

快速開啟提示視窗。

```
import { UofxDialogController } from '@uofx/web-components/dialog';

constructor(private dialogCtrl: UofxDialogController) {}

this.dialogCtrl.alert({
  titleKey: '標題',
  contentKey: '次標題',
  message: '其他訊息'
}).afterClose.subscribe(res => {
  this.message = 'alert closed result is :: ' + res;
})
```

✓ 自定義視窗內容

自訂視窗使用 `pDialog`，套用下列 `template` 可以呈現和 UOF X 一樣的視窗樣式。

```
<p-dialog #pDialog [(visible)]= "visible">
  <ng-template pTemplate="header">
    <uofx-dialog-header [title]="標題"></uofx-dialog-header>
  </ng-template>
  <ng-template pTemplate="content">
    <uofx-dialog-body>
      <div> ... </div>
    </uofx-dialog-body>
  </ng-template>
  <ng-template pTemplate="footer">
    <!-- 如果 uofx-dialog-footer 放置超過3顆按鈕，class 要加上 'buttons-opposite' -->
    <uofx-dialog-footer>
      <uofx-button [mode]="u-btn-primary" (click)="onSubmitClick()">Submit</uofx-button>
    </uofx-dialog-footer>
  </ng-template>
</p-dialog>
```

視窗需要繼承類別 `UofxDialog`，內含一些預設屬性與函式可以使用。

使用 `this.close()` 關閉視窗事件，也可在括號中放入數據回傳至原始元件，例如 `this.close(data)`。

```
import { UofxDialog } from '@uofx/web-components/dialog';

@Component({
  ...
})
export class NewDialog extends UofxDialog {

  ngOnInit() {
    this.id = this.params.id;
    this.type = this.params.type;
  }

  onSubmitClick() {
    // 送出...

    // 關閉視窗
    this.close();
  }
}
```

Note

`NewDialog` 不用像以往一樣特別加入到 `entryComponents` 即可使用。

開啟視窗的方式

開啟視窗時可以在 `params` 中放入要傳遞進視窗的參數。

如果關閉視窗時有回傳資料(`this.close(data)`)，會在 `afterClose` 後的 `res` 收到回傳資料。

```
import { NewDialog } from './dialog.component';

onOpenDialogClick() {
  this.dialogCtrl.create({
    component: NewDialog,
    params: { id: 'H001' }
  }).afterClose.subscribe(res => {
    if (res) ...
  });
};
```

開啟自訂視窗的方法

函式名稱	大小
<code>create</code>	開啟基本一般視窗
<code>createFullScreen</code>	填滿瀏覽器可視區域
<code>createFlexibleScreen</code>	填滿瀏覽器可視區域 90%

自訂視窗的可用參數

參數類型為 `UofxDialogOptions`。

名稱	用途
<code>component</code>	要開啟的元件
<code>params</code>	要傳入的參數
<code>size</code>	開窗大小，可設定 <code>xsmall</code> 、 <code>small</code> 、 <code>middle</code> 、 <code>large</code> 、 <code>xlarge</code> 、 <code>2xlarge</code> ，預設是 <code>large</code>
<code>noPadding</code>	設定開窗內容是否需要 <code>padding</code>
<code>showCloseBtn</code>	是否顯示右上角關閉鈕 (預設顯示)
<code>showMaximizeBtn</code>	是否顯示最大化按鈕 (預設不顯示)
<code>showHeader</code>	是否要顯示header區塊 (預設顯示)

User Select 選人

這個元件可以使用快篩搜尋，也可以點開視窗進行多種分類查詢，其中包含部門人員、部門、職稱、職務、部門+職稱、部門+職務和部門主管，也可以新增「我的常用」。

✓ Import 相依

```
import { UofxUserSelectModule } from '@uofx/web-components/user-select';
```

✓ 使用方式

設定可編輯的選人元件基本需要提供公司別和進階選人時所顯示的類別，設定和取得選人結果的方式是當作 form control 來使用。

```
import { UofxUserSetItemType, UofxUserSetModel } from '@uofx/web-components/user-select';
```

```
/** 公司別 */
corpId = Settings.UserInfo.corpId;
types: Array<UofxUserSetItemType> = [UofxUserSetItemType.DeptEmployee];
agent: Array<UofxUserSetModel> = [...];
```

```
<uofx-user-select
  [corpId]="corpId"
  [types]="types"
  [(ngModel)]="agent">
</uofx-user-select>
```

以下為 UofxUserSetItemType 中所包含的可選類別。

```
export enum UofxUserSetItemType {
  /** 人員 */
  Empl = 0,
  /** 部門 */
  Dept = 1,
  /** 職級 */
  JobTitle = 2,
  /** 職務 */
  JobFunc = 3,
  /** 部門+職級 */
  DeptJobTitle = 4,
  /** 部門+職務 */
  DeptJobFunc = 5,
  /** 部門主管 */
  DeptSupervisor = 6,
  /** 部門+人員 */
}
```

```
DeptEmployee = 7,
/** 我的常用 */
Commonuse = 99,
}
```

可用參數

參數名稱	說明
types	選人元件類型，不指定則表示全部類型
disabled	是否為允許編輯
multiple	是否為多選模式，預設為 true
expandToUser	選取後的結果是否展成人
allowFiltering	是否允許快篩，預設為 true
allowChoiceCompany	是否允許選擇公司
placeholder	要顯示在元件上的未填寫狀態提示文字，預設為 請選擇

✓ 使用純觀看模式

選人還有另外提供純觀看模式，因為不需要操作，所以僅提供要顯示的人員資訊就能完成。

```
import { UofxUserSetModel } from '@uofx/web-components/user-select';
```

```
selectedUserSet: Array<UofxUserSetModel> = [...];
```

```
<uofx-user-select-view [userSet]="selectedUserSet"></uofx-user-select-view>
```

可用參數

參數名稱	說明
userSet	傳入的資料
subUserSet	代理人資料
displayMode	顯示模式
displayCount	搭配 displayMode 為 more 模式，限定顯示的數量
showAvatar	是否顯示大頭照
uofSize	大頭照尺寸

顯示模式

模式名稱	說明
scroll	卷軸(用於表單檢視)
more	顯示 displayCount 設定的數量，其餘顯示其他幾個
default	預設只有一筆
expand	全部顯示:逗號串接(用於表單列印、退簽dialog、取回dialog)
vertical	全部顯示:垂直、沒有分隔號(、用於模擬流程)

Validators 共用驗證

UofxValidators 提供常用的驗證器，用於驗證輸入資料的正確性。這些驗證器會根據定義的規則進行檢查，並回傳 ValidationResult。

✓ Import 相依

```
import { UofxValidators } from '@uofx/web-components/form';
```

✓ 方法

名稱	說明
phoneNumber	手機號碼：0912345678 or 0912-345678 or 0912-345-678
idCardNumber	身分證字號 或 外籍居留證
telephone	市內電話：中間須包含 - 符號，ex: 07-9706789
unifiedBusinessNo	統一編號：8位數字，ex: 22311556
notAllowedSpaceString	必填時不允許純空格字串

✓ 使用方式

```
this.form = this.fb.group({  
  'id': id,  
  'code': ['', UofxValidators.notAllowedSpaceString]  
});
```

可用 CSS Class

✓ 基礎樣式

常用

class 名稱	樣式內容
height-100	height: 100%;
width-100	width: 100%;
span[uofx-required]:after	color: #f3162a; font-size: 1.4rem; font-weight:600; display: inline-block; content: "*"; font-family: SimSun; line-height: 1.5; vertical-align: middle;

顯示模式 (DISPLAY)

class 名稱	樣式內容
d-none	display: none;
d-block	display: block;
d-inline-block	display: inline-block;
d-flex	display: flex;
d-inline-flex	display: inline-flex;
justify-content-start	justify-content: flex-start;
justify-content-end	justify-content: flex-end;
justify-content-center	justify-content: flex-center;
justify-content-between	justify-content: flex-between;
justify-content-around	justify-content: flex-around;
align-items-start	align-items: flex-start;
align-items-end	align-items: flex-end;
align-items-center	align-items: flex-center;
align-items-baseline	align-items: flex-baseline;
align-items-stretch	align-items: flex-stretch;
align-content-start	align-content: flex-start;
align-content-end	align-content: flex-end;
align-content-center	align-content: flex-center;
align-content-between	align-content: flex-between;
align-content-around	align-content: flex-around;
align-content-stretch	align-content: flex-stretch;
align-self-auto	align-self: auto;
align-self-start	align-self: start;
align-self-end	align-self: end;

class 名稱	樣式內容
<code>align-self-center</code>	<code>align-self: center;</code>
<code>align-self-baseline</code>	<code>align-self: baseline;</code>
<code>align-self-stretch</code>	<code>align-self: stretch;</code>
<code>flex-center</code>	<code>display: flex;</code> <code>align-items: center;</code>
<code>flex-center-between</code>	<code>display: flex;</code> <code>align-items: center;</code> <code>align-content: space-between;</code>
<code>flex-nowrap</code>	<code>flex: stretch;</code>
<code>flex-1</code>	<code>flex: 1;</code>
<code>flex-2</code>	<code>flex: 2;</code>
<code>flex-row</code>	<code>flex-direction: row;</code>
<code>flex-row-reverse</code>	<code>flex-direction: row-reverse;</code>
<code>flex-column</code>	<code>flex-direction: column;</code>
<code>flex-column-reverse</code>	<code>flex-direction: column-reverse;</code>
<code>float-right</code>	<code>float: right;</code>
<code>float-left</code>	<code>float: left;</code>
<code>float-inline-start</code>	<code>float: inline-start ;</code>
<code>float-inline-end</code>	<code>float: inline-end ;</code>

文字 (FONTS & TEXT)

文字分為依比例和固定式的，命名方式為 `text-12` 與 `text-12-fixed`，每一個文字大小都會有一組對應的。

class 名稱	文字大小	固定式 (fixed)
<code>text-12</code>	1.2rem;	12px
<code>text-13</code>	1.3rem;	13px
<code>text-14</code>	1.4rem;	14px
<code>text-15</code>	1.5rem;	15px
<code>text-16</code>	1.6rem;	16px
<code>text-17</code>	1.7rem;	17px
<code>text-18</code>	1.8rem;	18px
<code>text-20</code>	2.0rem;	20px
<code>text-25</code>	2.5rem;	25px
<code>text-30</code>	3.0rem;	30px
<code>text-35</code>	3.5rem;	35px
<code>text-40</code>	4.0rem;	40px
<code>text-45</code>	4.5rem;	45px
<code>text-50</code>	4.0rem;	50px

class 名稱	樣式內容
<code>textarea-content</code>	<code>white-space: pre-line;</code>
<code>text-wrap</code>	<code>white-space: normal;</code>
<code>text-nowrap</code>	<code>white-space: nowrap;</code>
<code>text-underline</code>	<code>text-decoration: underline;</code>
<code>text-left</code>	<code>text-align: left;</code>
<code>text-right</code>	<code>text-align: right;</code>
<code>text-center</code>	<code>text-align: center;</code>
<code>font-weight-600</code>	<code>font-weight: 600;</code>
<code>text-transform-none</code>	<code>text-transform: none;</code>
<code>text-break</code>	<code>word-break: break-all;</code>

物件外間距 (MARGIN)

僅列出 `margin` 和 `margin-top`，剩下的 `margin-bottom`、`margin-right` 和 `margin-left` 以此類推。

class 名稱	樣式內容
<code>margin-s</code>	<code>margin: 4px;</code>
<code>margin</code>	<code>margin: 8px;</code>
<code>margin-2x</code>	<code>margin: 16px;</code>
<code>margin-3x</code>	<code>margin: 24px;</code>
<code>margin-4x</code>	<code>margin: 32px;</code>
<code>margin-5x</code>	<code>margin: 40px;</code>
<code>margin-6x</code>	<code>margin: 48px;</code>
<code>margin-7x</code>	<code>margin: 56px;</code>
<code>margin-8x</code>	<code>margin: 64px;</code>
<code>margin-top-s</code>	<code>margin-top: 4px;</code>
<code>margin-top</code>	<code>margin-top: 8px;</code>
<code>margin-top-2x</code>	<code>margin-top: 16px;</code>
<code>margin-top-3x</code>	<code>margin-top: 24px;</code>
<code>margin-top-4x</code>	<code>margin-top: 32px;</code>
<code>margin-top-5x</code>	<code>margin-top: 40px;</code>
<code>margin-top-6x</code>	<code>margin-top: 48px;</code>
<code>margin-top-7x</code>	<code>margin-top: 56px;</code>
<code>margin-top-8x</code>	<code>margin-top: 64px;</code>

`margin-horizontal` 和 `margin-vertical` 也是同上列的模式。

class 名稱	樣式內容
<code>margin-horizontal</code>	<code>margin-left: 8px;</code> <code>margin-right: 8px;</code>
<code>margin-vertical</code>	<code>margin-top: 8px;</code> <code>margin-bottom: 8px;</code>

其他間距設定

class 名稱	樣式內容
<code>no-margin</code>	<code>margin: 0 !important;</code>
<code>no-margin-horizontal</code>	<code>margin-left: 0;</code> <code>margin-right: 0;</code>
<code>no-margin-vertical</code>	<code>margin-top: 0;</code> <code>margin-bottom: 0;</code>
<code>margin-auto</code>	<code>margin: auto;</code>
<code>margin-horizontal-auto</code>	<code>margin-left: auto;</code> <code>margin-right: auto;</code>
<code>margin-vertical-auto</code>	<code>margin-top: auto;</code> <code>margin-bottom: auto;</code>

物件內間距 (PADDING)

padding 的設定大致上 margin 相同，只是少了 auto 的部分，不過下列也會將 padding 可用的設定列出。一樣僅列出 padding 和 padding-top，剩下的 padding-bottom、padding-right 和 padding-left 以此類推。

class 名稱	樣式內容
padding-s	padding: 4px;
padding	padding: 8px;
padding-2x	padding: 16px;
padding-3x	padding: 24px;
padding-4x	padding: 32px;
padding-5x	padding: 40px;
padding-6x	padding: 48px;
padding-7x	padding: 56px;
padding-8x	padding: 64px;
padding-top-s	padding-top: 4px;
padding-top	padding-top: 8px;
padding-top-2x	padding-top: 16px;
padding-top-3x	padding-top: 24px;
padding-top-4x	padding-top: 32px;
padding-top-5x	padding-top: 40px;
padding-top-6x	padding-top: 48px;
padding-top-7x	padding-top: 56px;
padding-top-8x	padding-top: 64px;

`padding-horizontal` 和 `padding-vertical` 也是同上列的模式。

class 名稱	樣式內容
<code>padding-horizontal</code>	<code>padding-left: 8px;</code> <code>padding-right: 8px;</code>
<code>padding-vertical</code>	<code>padding-top: 8px;</code> <code>padding-bottom: 8px;</code>

其他間距設定

class 名稱	樣式內容
<code>no-padding</code>	<code>padding: 0 !important;</code>
<code>no-padding-horizontal</code>	<code>padding-left: 0;</code> <code>padding-right: 0;</code>
<code>no-padding-vertical</code>	<code>padding-top: 0;</code> <code>padding-bottom: 0;</code>

✓ 欄位樣式

class 名稱	樣式內容
<code>fw-control</code>	<code>padding: 12px;</code> <code>word-break: break-all;</code> <code>white-space: pre-line;</code> <code>font-size: 1.6rem;</code>
<code>fw-descr</code>	<code>margin-top: 4px;</code> <code>word-break: break-all;</code> <code>font-size: 1.3rem;color: #2578fa;</code>

✓ row col 樣式

bootstrap grid system 用於排版和對齊列與欄，主要用 flexbox 建立，並支援自適應樣式(RWD)。

每一列總共有 12 欄，可任意的組合。col-x 代表著要使用的欄數 (例如：col-3 三欄、col-6 六欄)

```
<div class="container">
  <div class="row">
    <div class="col-3"> column1 </div>
    <div class="col-3"> column2 </div>
    <div class="col-3"> column3 </div>
    <div class="col-3"> column4 </div>
```

```
</div>

<div class="row">
  <div class="col-2"> column1 </div>
  <div class="col-4"> column2 </div>
  <div class="col-6"> column3 </div>
</div>

<div class="row">
  <div class="col-12"> column1 </div>
</div>
</div>
```

5.2 App

Error Message 錯誤訊息

UOF X 所提供的錯誤訊息包含「純紅色文字錯誤訊息」與「紅底色條列式錯誤訊息方塊」。

✓ Import 相依

```
import { UofxErrorBlockModule, UofxErrorTipModule } from '@uofx/app-components/form';
```

✓ 純紅色文字錯誤訊息

使用方式是提供一個 control 給元件，control 可以接受 FormControl 或 AbstractControl 類型，而訊息出現的條件為 control.dirty 為 true 且有 control.errors 的狀況下。

```
<input pInputText class="width-100" formControlName="textboxCtrl">
<uofx-error-tip [control]="form.controls.textboxCtrl"></uofx-error-tip>
```

```
this.form = this.fb.group({
  textboxCtrl: [null, Validators.required]
});
```

可用的預設錯誤訊息

說明	顯示條件 (hasError)	置換預設訊息 (properties)
必填	required	requiredText
輸入重複	duplicate	duplicateText
資料不存在	notExist	
格式不正確	email、pattern	formatIncorrectText
超出範圍	min、max	rangeText
最大最小長度限制	minLength、maxLength	minLengthText、maxLengthText
只允許字母及數字	onlyDigitsOrLetters	onlyDigitsOrLettersText
比較值內容不同	notEqual	notEqualText

修改錯誤訊息內容

舉例來說，可以在需要提示必填的情況下，將預設文字改為「身分證必填」，可以設定 requiredText，如此一來在 uofx-error-tip 內偵測到 control.hasError('required') 時就會顯示改訊息了。

```
<uofx-error-tip  
  [control]="textboxCtrl"  
  [requiredText]="身分證必填">  
</uofx-error-tip>
```

✓ 紅底色條列式錯誤訊息方塊

錯誤訊息方塊為條列式呈現，所以接收的參數為陣列形式。

- `errorMessage` 顯示錯誤訊息內容

```
transErrorCodes = ['輸入的訊息內容格式不正確'];
```

```
<uofx-error-block [errorMessage]="transErrorCodes" >  
</uofx-error-block>
```

User Select 選人

這個元件可以使用快篩搜尋，也可以點開視窗進行多種分類查詢，其中包含部門人員、部門、職稱、職務、部門+職稱、部門+職務和部門主管，也可以新增「我的常用」。

✓ Import 相依

```
import { UofxUserSelectModule, UofxUserSetModel } from '@uofx/app-components/user-select';
```

✓ 使用方式

設定可編輯的選人元件基本需要提供公司別和進階選人時所顯示的類別，設定和取得選人結果的方式是當作 form control 來使用。

```
import { UofxUserSetItemType, UofxUserSetModel } from '@uofx/app-components/user-select';
```

```
/** 公司別 */
corpId = Settings.UserInfo.corpId;
types: Array<UofxUserSetItemType> = [UofxUserSetItemType.JobTitle];
agent: Array<UofxUserSetModel> = [...];
```

```
<uofx-user-select
  [corpId]="corpId"
  [types]="types"
  [(ngModel)]="agent">
</uofx-user-select>
```

以下為 UofxUserSetItemType 中所包含的可選類別。

```
export enum UofxUserSetItemType {
  /** 人員 */
  Empl = 0,
  /** 部門 */
  Dept = 1,
  /** 簽核層級 */
  JobTitle = 2,
  /** 職務 */
  JobFunc = 3,
  /** 部門+職級 */
  DeptJobTitle = 4,
  /** 部門+職務 */
  DeptJobFunc = 5,
  /** 部門主管 */
  DeptSupervisor = 6,
  /** 部門+人員 */
}
```

```
DeptEmployee = 7,
/** 我的常用 */
Commuse = 98
}
```

可用參數

參數名稱	說明
types	選人元件類型，不指定則表示全部類型
disabled	是否為允許編輯
expandToUser	選取後的結果是否展成人
placeholder	要顯示在元件上的未填寫狀態提示文字，預設為 請選擇

✓ 使用純觀看模式

選人還有另外提供純觀看模式，因為不需要操作，所以僅提供要顯示的人員資訊就能完成。

```
import { UofxUserSetModel } from '@uofx/app-components/user-select';
```

```
selectedUserSet: Array<UofxUserSetModel> = [...];
```

```
<uofx-user-select-view [userSet]="selectedUserSet"></uofx-user-select-view>
```

Date Picker 日期

UOF X 透過 ion-modal 與 ion-datetime 的互動來實作日期元件

✓ Import 相依

```
import { UofxDatePickerModule } from '@uofx/app-components/form';
```

✓ 基礎使用方式

```
selectdate = new Date();
```

```
<uofx-date-picker [(ngModel)]="selectdate"></uofx-date-picker>
```

可用參數

參數名稱	說明
presentation	選擇日期 date、選擇日期時間 date-time，預設為 date
displayFormat	日期格式，預設為 yyyy/MM/dd
min	日期可選最小值
max	日期可選最大值
disabled	是否為允許編輯

6. Plugin 套件

6.1 安裝

安裝 **Plugin** 套件

於外掛模組的 Angular 專案中，執行下列指令進行安裝。

```
npm install @uofx/plugin
```

6.2 Plugin API

Import 相依

advanced-field.module.ts

```
import { UofxPluginApiService } from "@uofx/plugin/api"

@NgModule({
  imports: [ ... ]
  providers: [
    UofxPluginApiService
  ]
})
export class AdvancedFieldModule {
  ...
}
```

使用方式

如此便可在外掛欄位元件中取得。

advanced-field.write.component.ts

```
import { UofxPluginApiService } from "@uofx/plugin/api"

@Component({
  ...
})
export class AdvancedFieldWriteComponent ... {
  constructor(private pluginService: UofxPluginApiService) {
    this.pluginService ...
  }
}
```

可用方法

- getUserInfo(): 取得員工資訊。
- getCorpInfo(): 取得公司資訊。

advance-field.write.component.ts

```
loadInfo() {
  // 呼叫 API 取得員工和公司相關資訊
  zip(
```

```
    this.pluginService.getCorpInfo().toPromise(),
    this.pluginService.getUserInfo(this.taskNodeInfo.applicantId).toPromise()
  ).subscribe({
    next: ([corpInfo, emplInfo]) => {
      if(emplInfo.employeeNumber){
        // 取得員工編號
        this.empNo = emplInfo.employeeNumber;
      }
      console.log(this.empNo);
    },
    complete: () => {
      console.log(this.empNo);
      // 讓畫面更新
      this.cdr.detectChanges();
    }
  });
}
```

6.3 頁面權限

權限設定為用來限制可存取的使用者。

Import 相依

```
import { UofxPluginAuthorize } from '@uofx/plugin';
```

使用方式

可自訂 `functionId` 名稱。

```
@UofxPluginAuthorize({ functionId: "LOBBY" })  
export class LobbyPage {}
```

6.4 頁面工具

Import 相依

```
import { UofxPluginPage } from '@uofx/plugin';
```

使用方式

頁面類別繼承底層類型 `UofxPluginPage` 即有對應的變數可用。

```
import { UofxPluginPage } from '@uofx/plugin'
@Component({
  ...
})
export class SettingPage extends UofxPluginPage implements OnInit {
  ...
}
```

取得 Entry Host

取得 Entry Host 設定為 API Service 的 Server Url。

```
import { UofxPluginPage } from '@uofx/plugin'
@Component({
  selector: 'uofx-web-setting',
  template: `...`
})
export class SettingPage extends UofxPluginPage implements OnInit {
  constructor(private apiService: APIService) { super(); }

  ngOnInit() {
    console.log(this.pluginSetting.entryHost);
    this.apiService.serverUrl = this.entryHost;
  }
}
```

取得路徑

透過取得的路徑可在 `routerLink` 上組合使用，使用此方法方便直接指定對應的路徑，不必擔心相對路徑問題。

```
import { UofxPluginPage } from '@uofx/plugin'
@Component({
  selector: 'uofx-web-setting',
```

```

template: `
  <uofx-breadcrumb [items]="breadCrumbItems"> </uofx-breadcrumb>
  `
})
export class SettingPage extends UofxPluginPage implements OnInit {
  /** 麵包屑清單 */
  breadCrumbItems: Array<MenuItem> = [
    { label: '大廳', routerLink: [this.baseUrl.admin] },
    { label: '內容置中' },
  ];

  ngOnInit() {
    console.log(this.baseUrl.adminPlugin);
    // 輸出: /admin/plugin/edesampleplugin
  }
}

```

✓ Base URI 可用屬性

名稱	說明	目標位置
admin	管理者大廳	/admin
adminPlugin	管理者目標外掛模組根路徑	/admin/plugin/{codePath}
user	使用者大廳	/user
userPlugin	使用者目標外掛模組根路徑	/user/plugin/{codePath}
appPlugin	App目標外掛模組根路徑	/home/plugin/{codePath}

7. 外掛模組部署

7.1 如何部署

從專案執行發佈

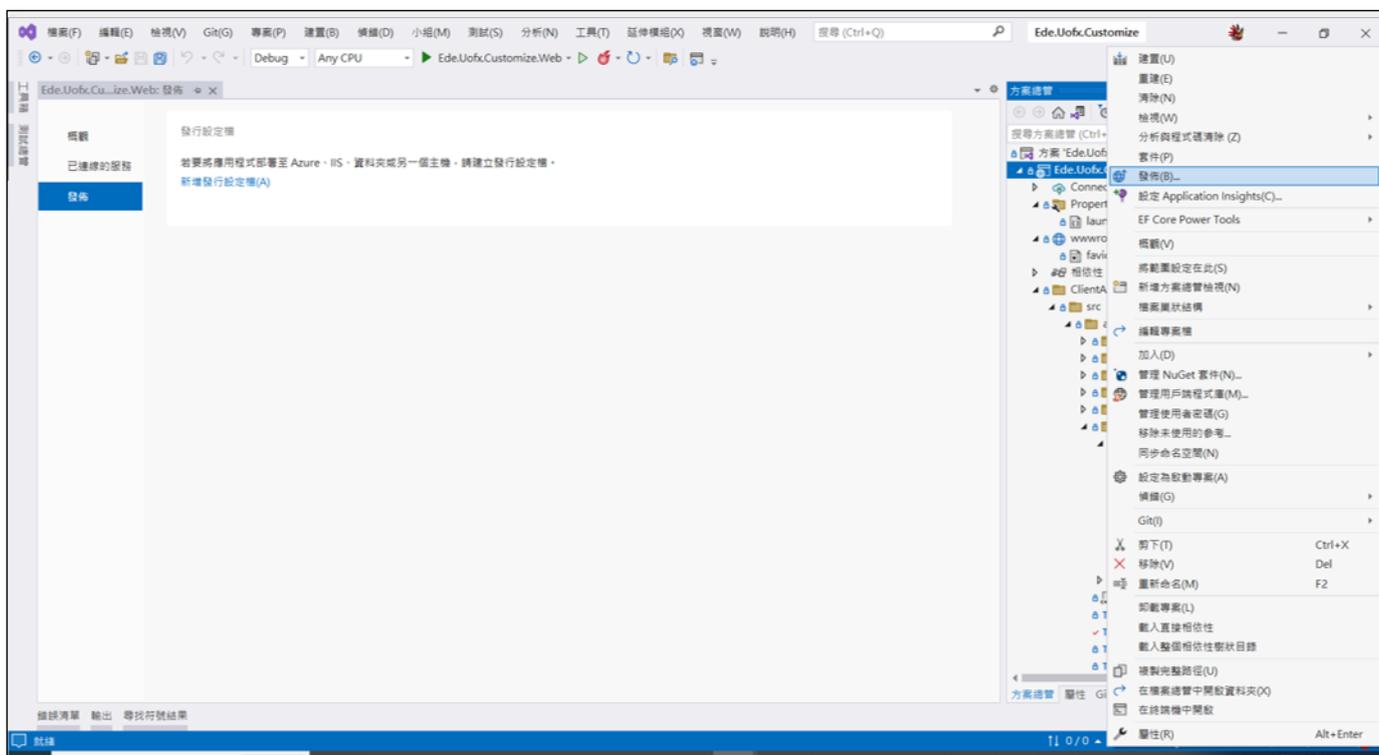
- 發佈專案之前先注意調整 `webpack-exposes.config.js` 的內容，設定讀取外掛欄位設定檔的位置。

`webpack-exposes.config.js`

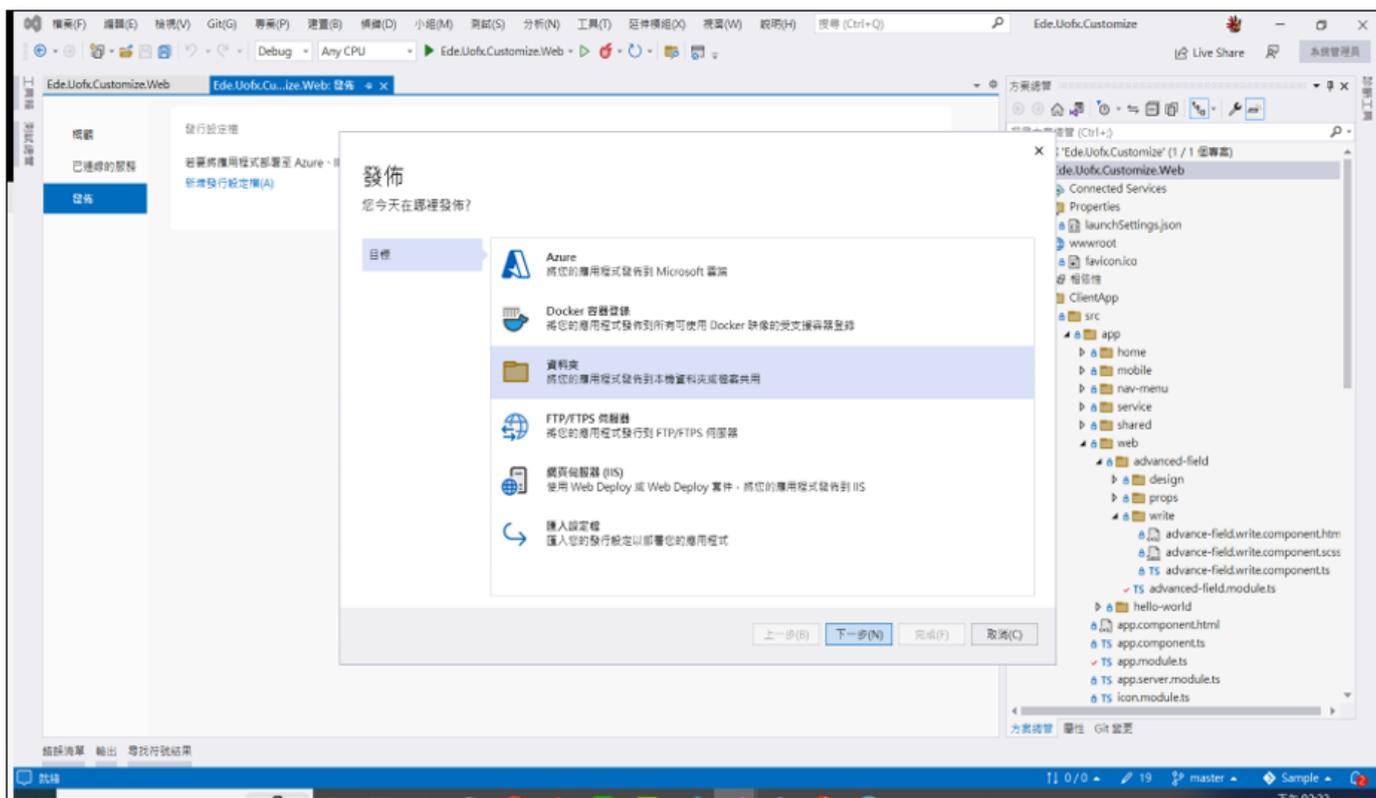
```
const exposes = {  
  // 設定要開放外部使用欄位和頁面  
  web: {  
    "/HelloWorld": "./src/app/web/hello-world/hello-world.module.ts",  
    "/AdvancedField": "./src/app/web/advanced-field/advanced-field.module.ts",  
  },  
  app: {  
    "/HelloWorld": "./src/app/mobile/hello-world/hello-world.module.ts",  
    "/AdvancedField": "./src/app/mobile/advanced-field/advanced-field.module.ts",  
  },  
}
```

},
};

1. 在 visual studio 的外掛欄位專案(Ede.Uofx.Customize.Web)中，從右鍵選單執行發佈的選項，選擇發佈到資料夾在執行下一步。

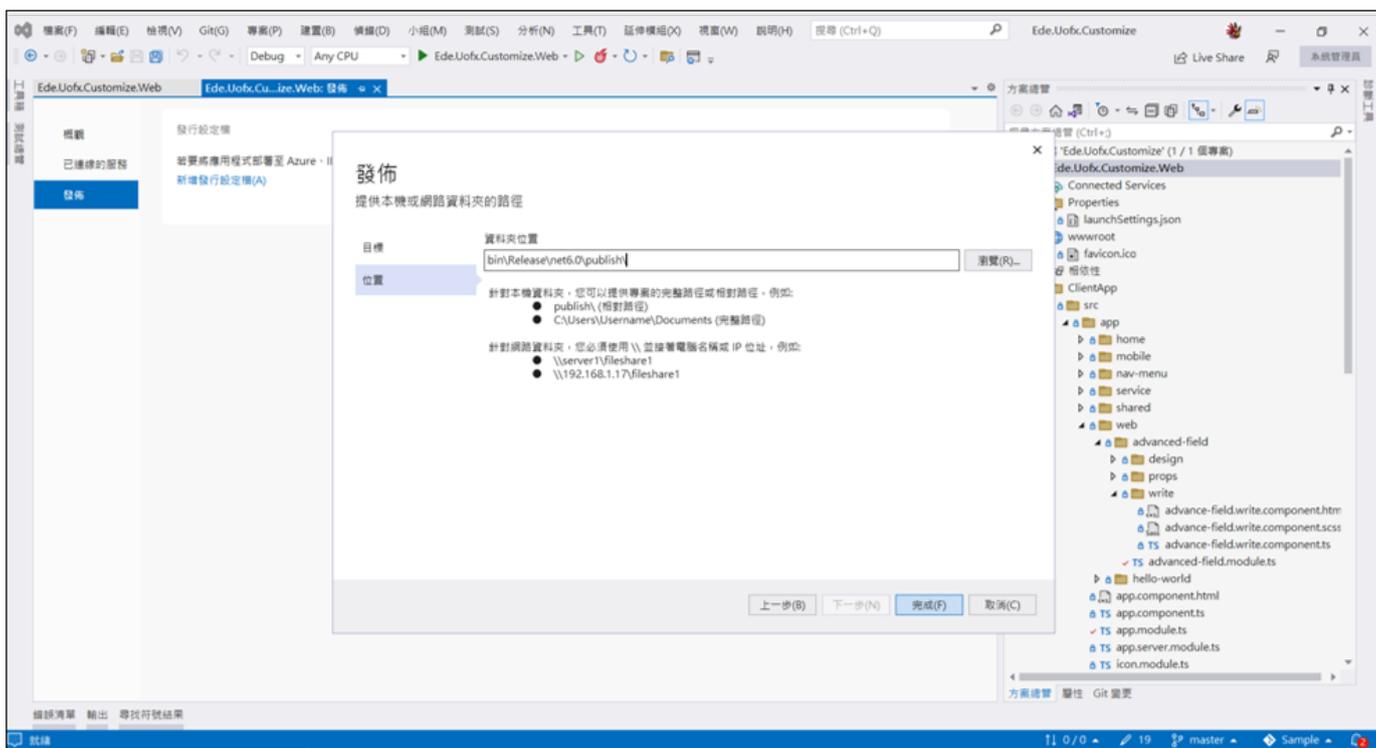


visual studio發佈



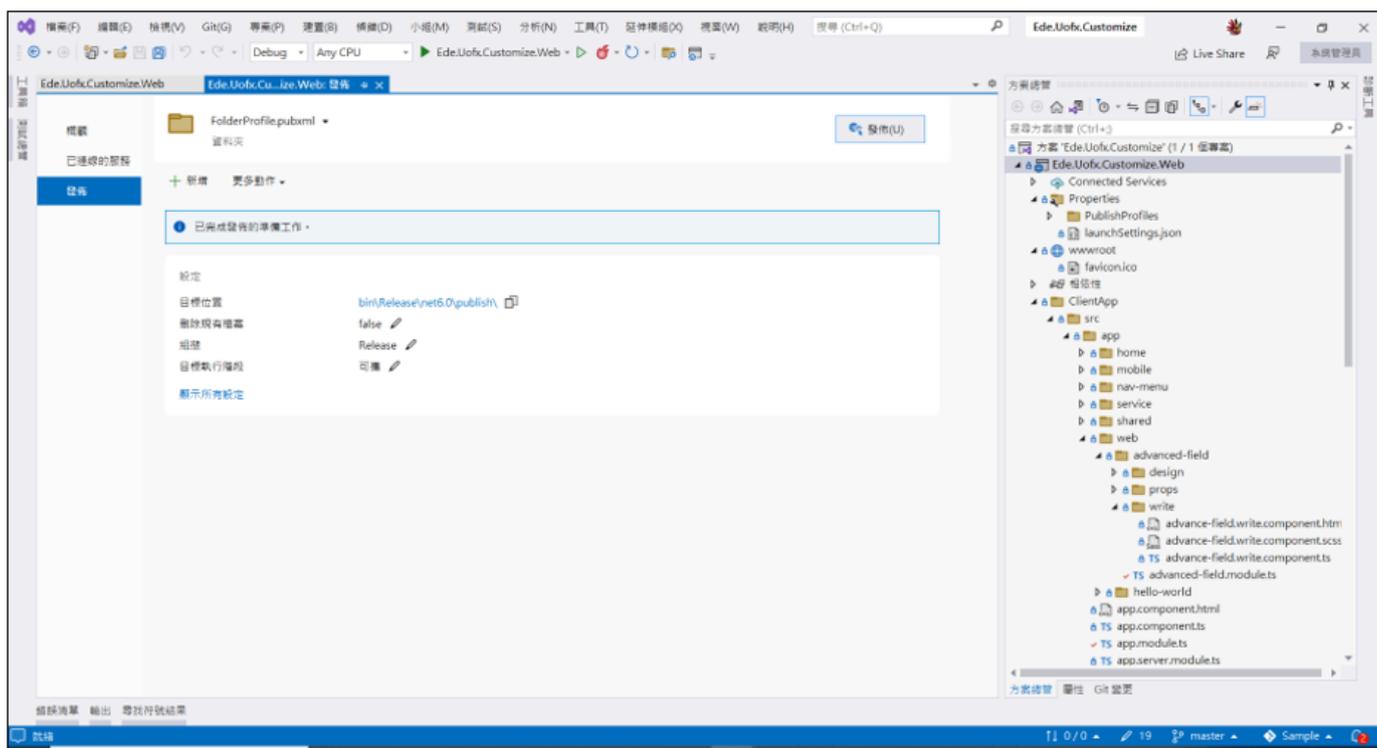
發佈到資料夾

2. 設定發佈資料夾的位置後，點選完成，最後在選擇發佈。

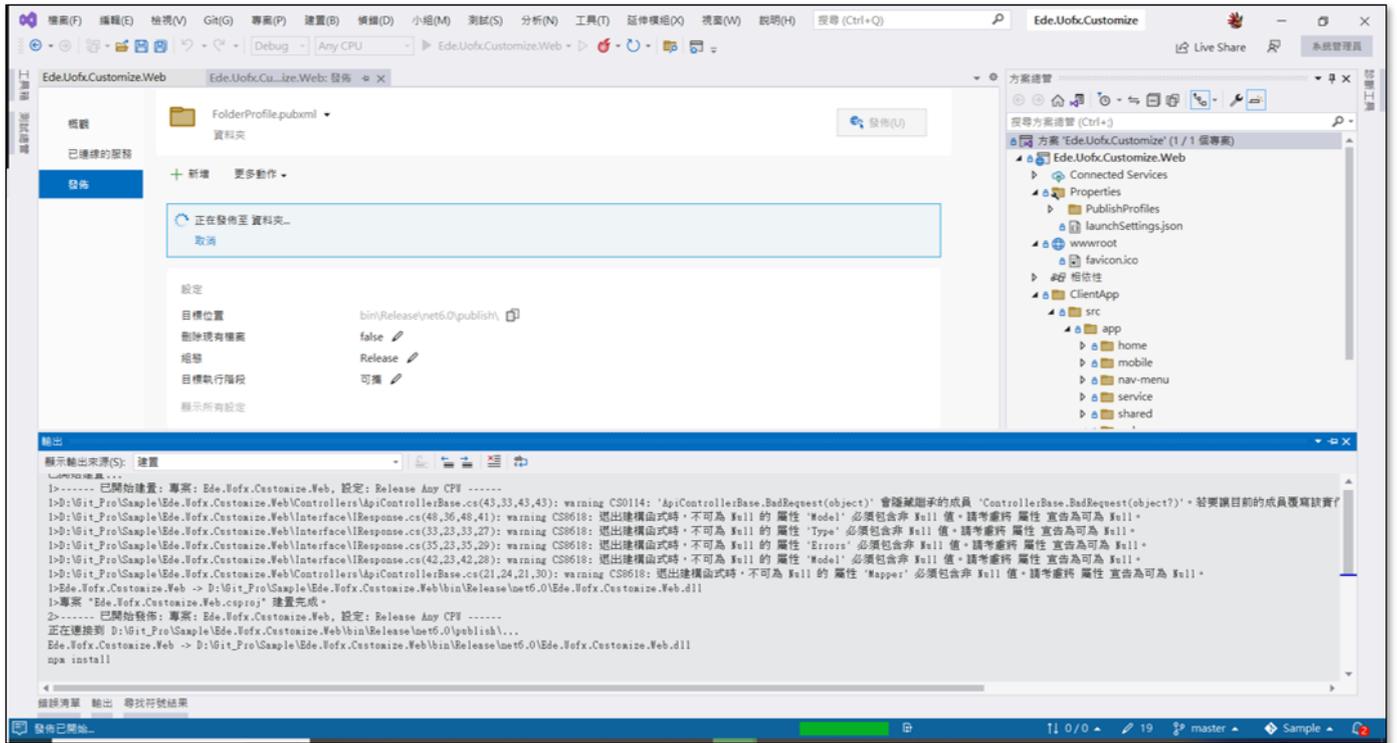


選擇資料夾

3. 執行發佈。

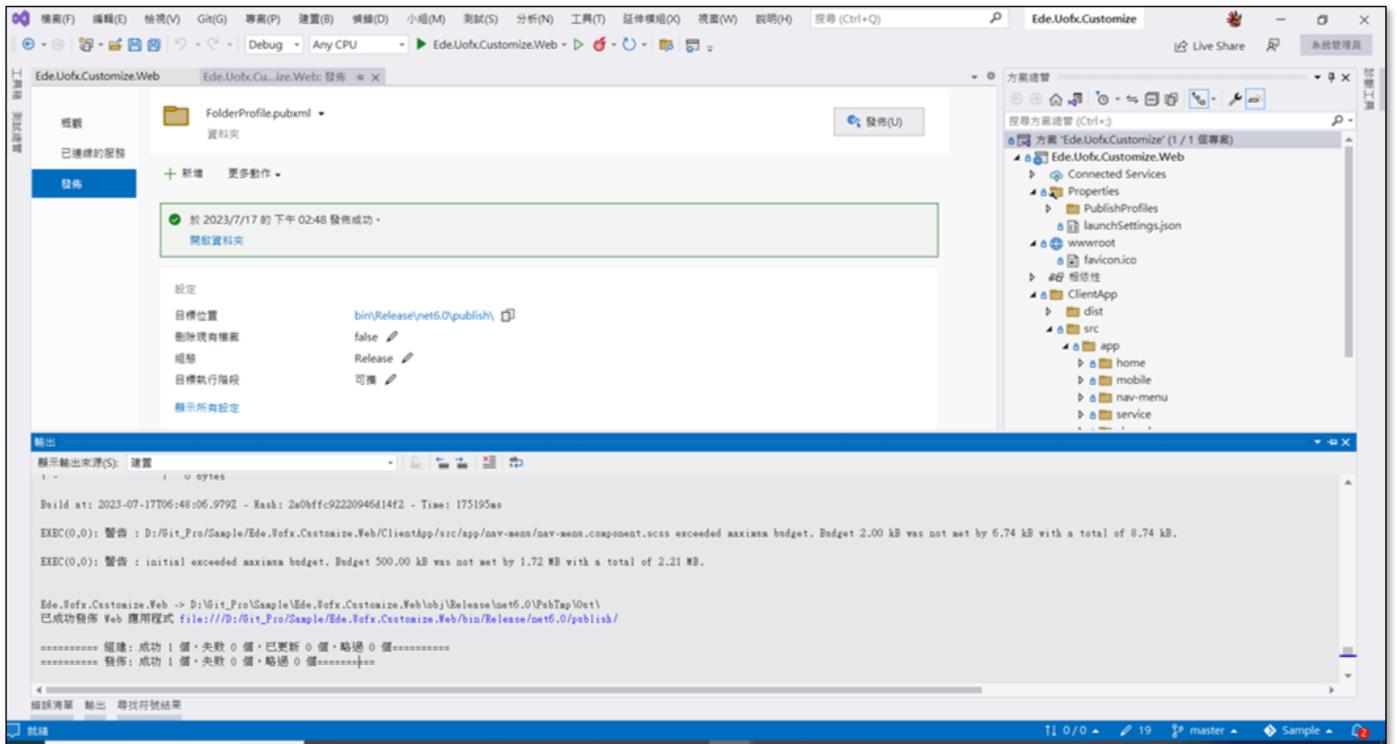


外掛欄位發佈1



外掛欄位發佈2

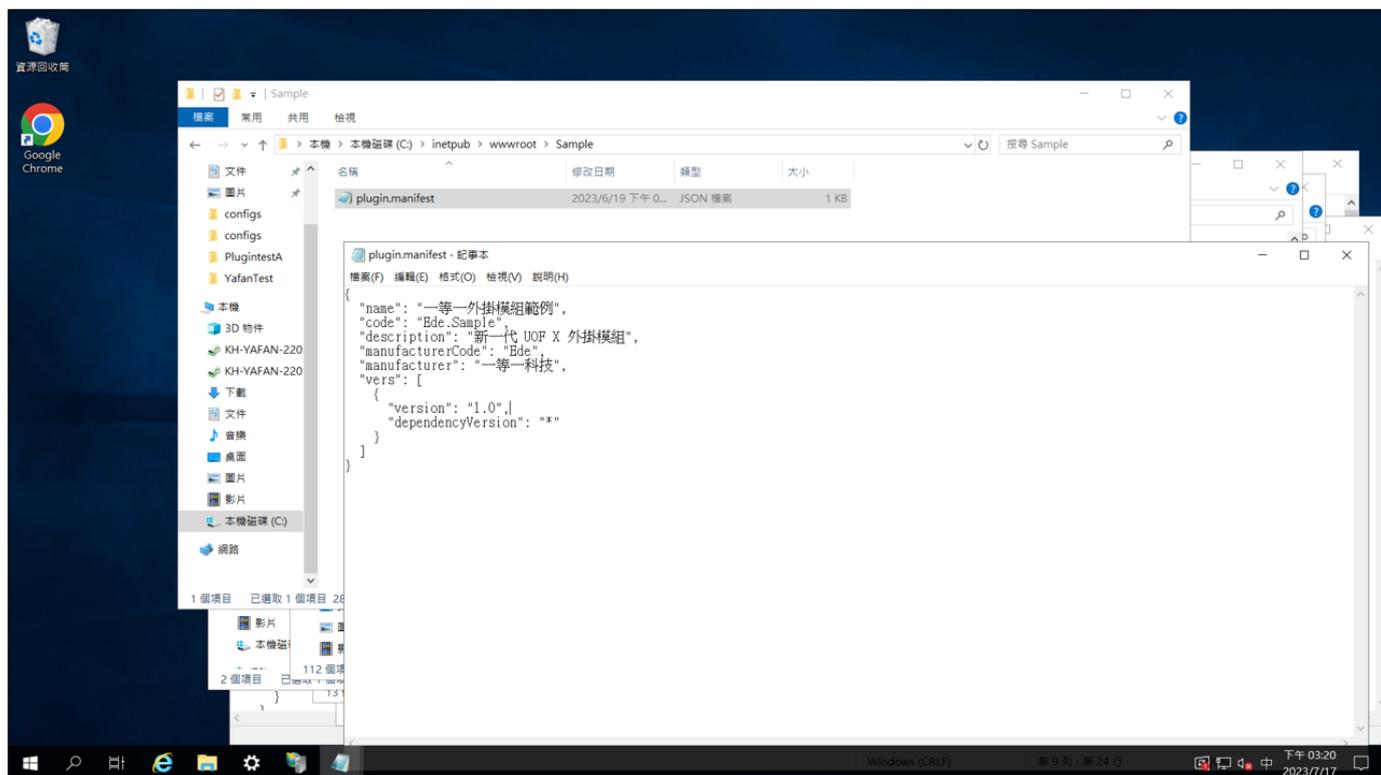
4. 發佈成功。



發佈成功

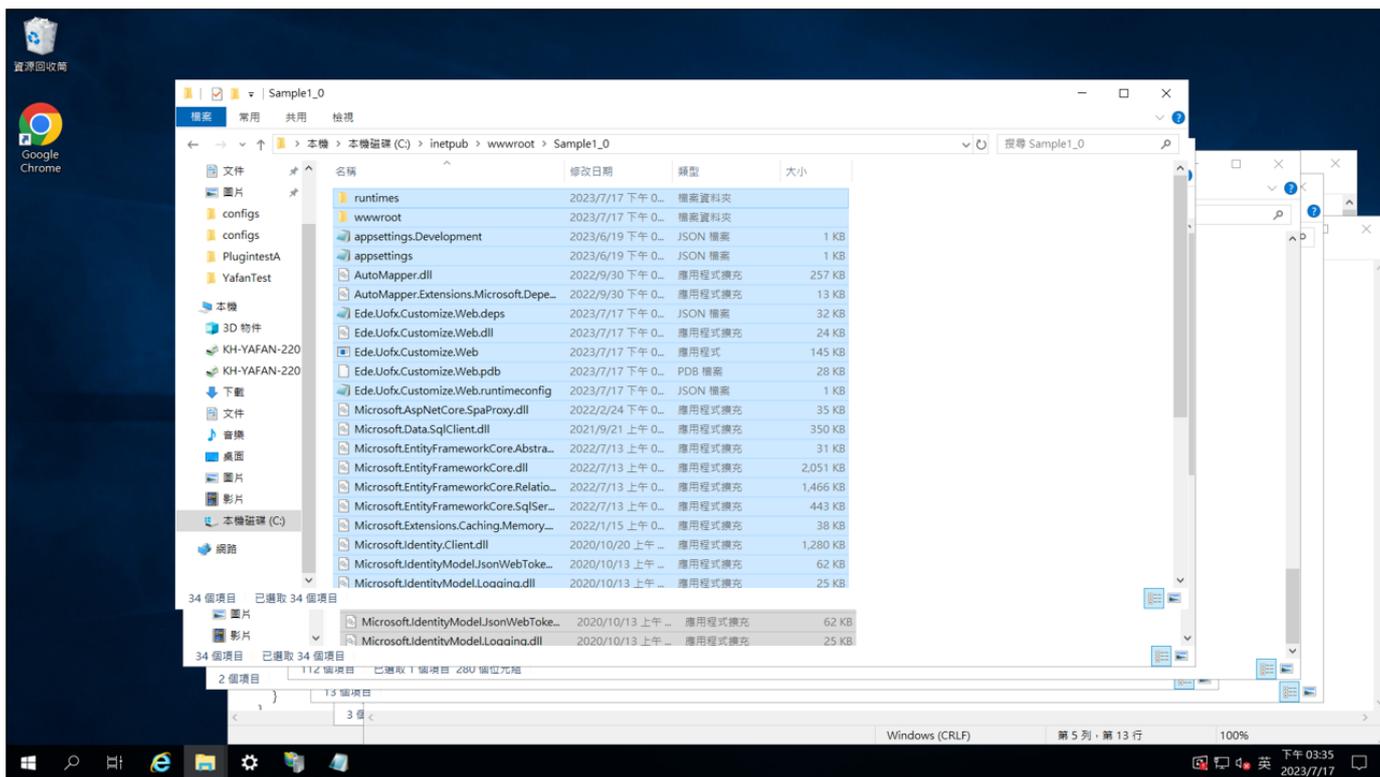
在 IIS 中建立外掛模組站台

1. 建立要在 IIS 中放置外掛模組程式站台的目錄(假設命名為 Sample, C:\inetpub\wwwroot\Sample), 將發佈後的 **plugin.manifest.json** 與 **plugin.versions.json** 檔案複製在這個路徑下, 檔案內容的設定可以參考 [配置與設定](#)。



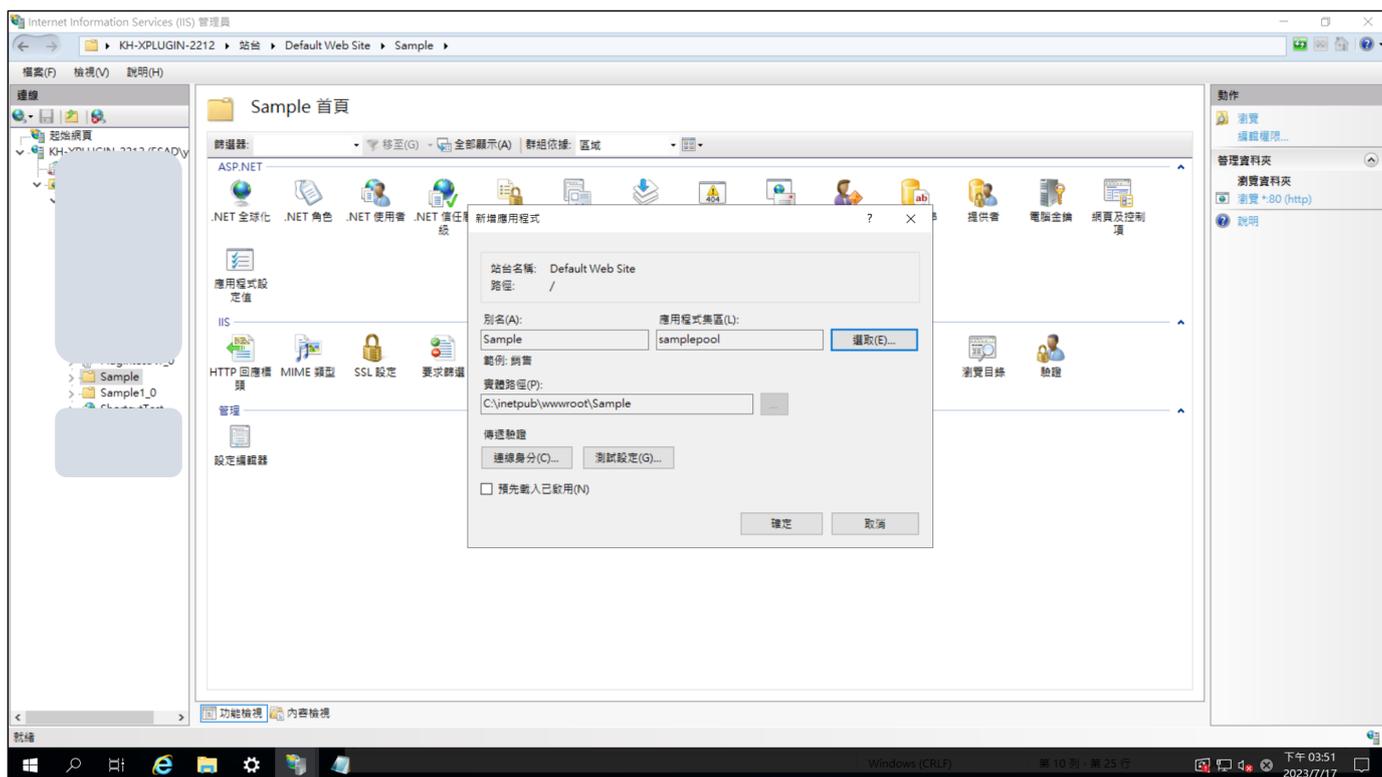
建立外掛模組站台根目錄

2. 建立要在 IIS 中與步驟1相同且加上版號的目錄, 例如版本 1.0 就是 Sample1_0, 目錄名稱後的版號需要和 **plugin.versions.json** 內容中的 **version** 設定 1.0 相對應, 所以如果版號是 2.0, 則目錄名稱需命名為 Sample2_0 以此類推, 接著將外掛模組發的所有目錄與檔案都複製到這個 Sample1_0 目錄下。



建立外掛模組站台對應版本目錄

- 將目錄 Sample 和 Sample1_0 轉換成 Web 應用程式目錄，轉換完成後可透過直接瀏覽外掛模組站台的 `remoteEntry.js` 確認是否有成功執行，例如：http://172.16.3.123/Sample1_0/remoteEntry.js。



轉換成 IIS 應用程式

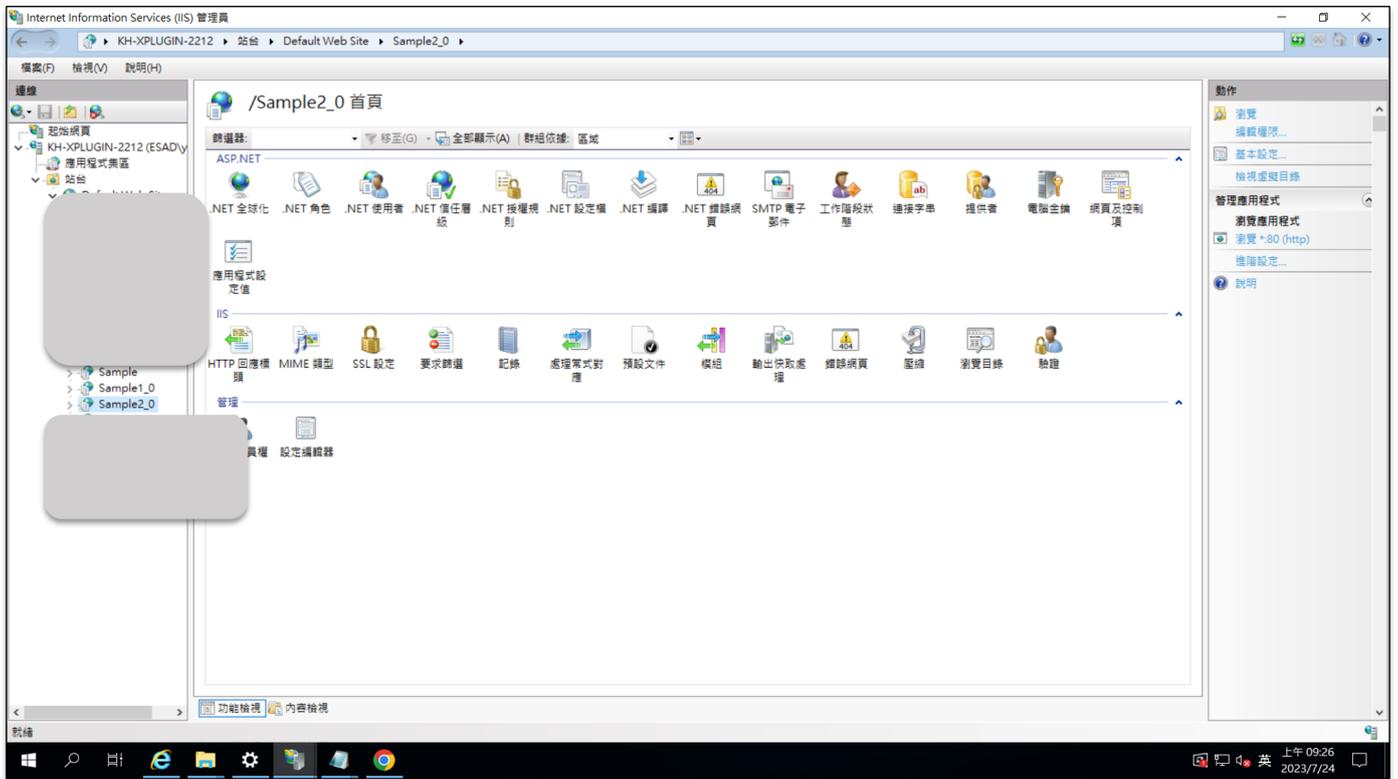
4. 可以檢查以下幾個位置是否能正常讀取

- ✔ <http://host/Sample/plugin.manifest.json>
- ✔ <http://host/Sample/plugin.versions.json>
- ✔ http://host/Sample1_0/remoteEntry.js
- ✔ http://host/Sample1_0/assets/configs/field-design.json
- ✔ http://host/Sample1_0/assets/configs/field-runtime.json

7.2 模組更新

外掛模組更新或是卸載都是透過 **Plugin 管理** 功能進行處理。

- 外掛模組更新版本時，需再重新發佈一次外掛模組網站，並於 IIS 站台上建置新版本的外掛模組站台。



IIS 新版本外掛站台建立

- 調整 Sample 站台內 **plugin.versions.json** 內容加上新版 2.0 的設定。
- UOF X 會依照這份設定檔的最小支援版本(minimumUOFXVersion)自動切換。

plugin.versions.json

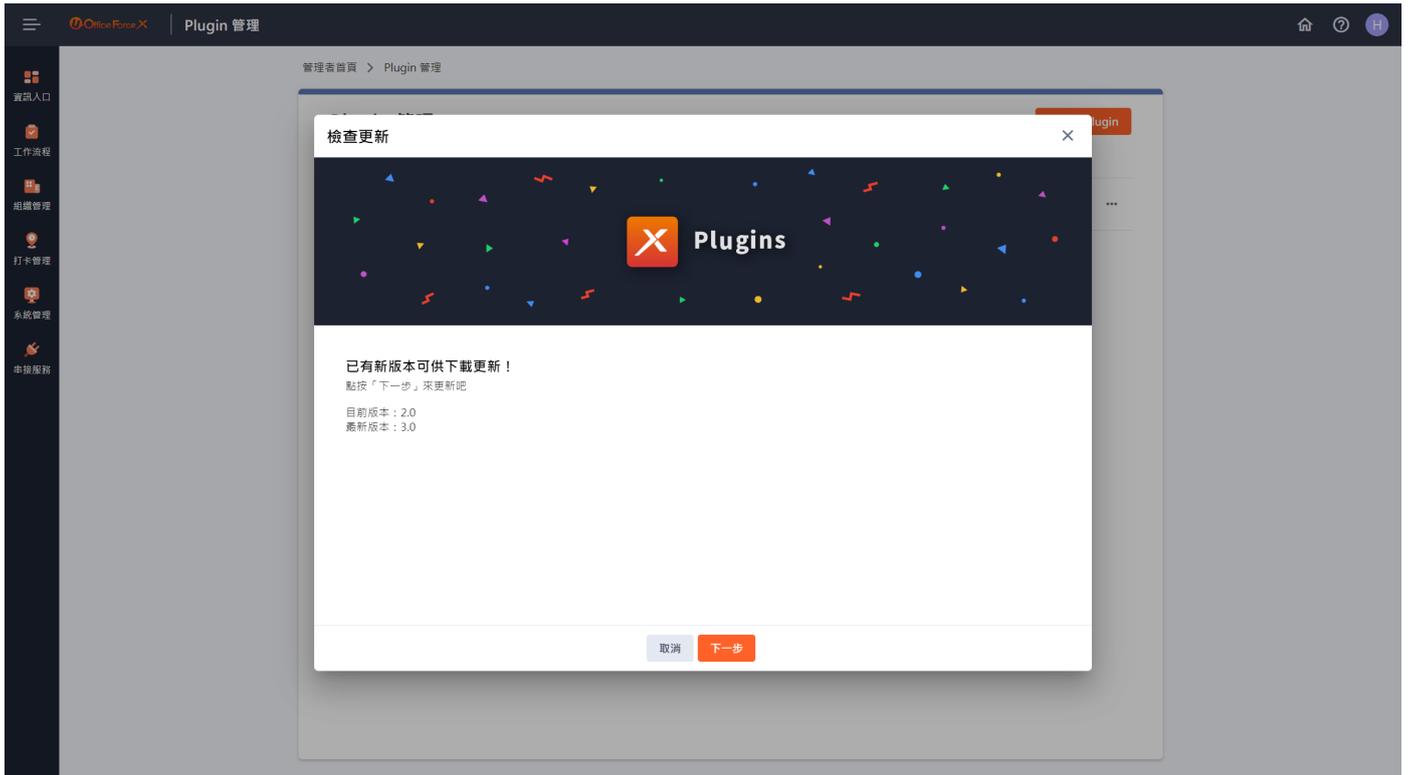
```
{
  "$schema": "../node_modules/@uofx/plugin/schema/plugin-versions.schema.json",
  "versions": [
    {
      "version": "2.0",
      "minimumUOFXVersion": "2.94.0",
      "changelog": [
        "本次更新除掉了幾條蟲"
      ]
    },
    {
      "version": "1.0",
```

```

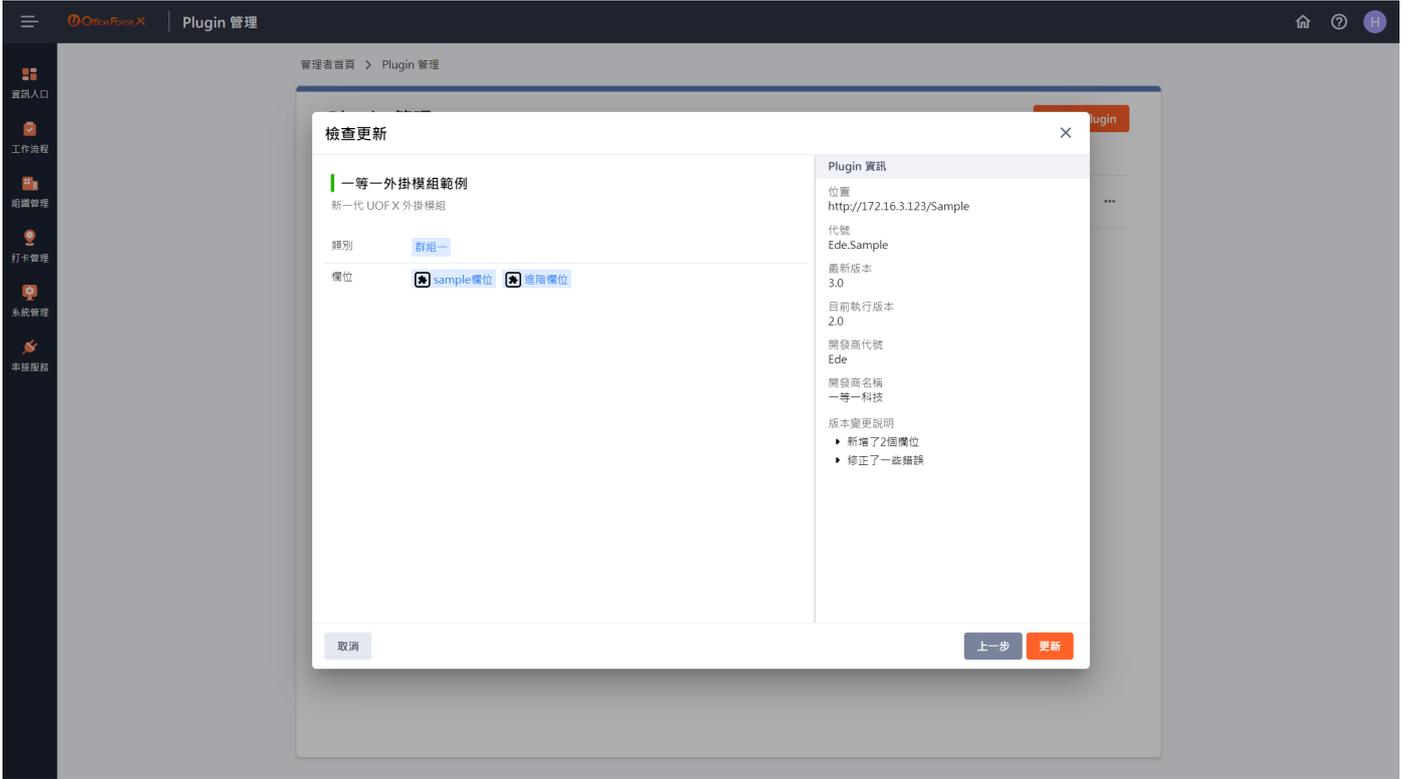
"minimumUOFXVersion": "2.89.1",
"changelog": [
  "全新對話功能上線！"
]
}
]
}

```

- 設定完成可由 Plugin 管理功能對 Sample 外掛模組進行更新 [Plugin 檢查更新](#)。



Plugin 外掛模組更新



Plugin 外掛模組更新

8. 升級注意事項

8.1 2407 → 2025-R1

本次 UOF X 由 2407 升級至 2025-R1 注意事項如下：

破壞性異動

 NodeJS 升級至 18.20.3

 Angular 升級至 17.3.11

Danger

全系統下載原始檔時，產生的連結需判斷權限。

外部站點原先從前端所取得使用的 `/api/desktop/v1.0/base/file/Download`，因為改為需要權限判斷，所以需要將其取代改為 `/api/desktop/v1.0/base/file/ext/Download` 才能正常使用。

WEB

 primeng 升級至 17.18.8

 gcpdfviewer 升級至 7.1.2

```
npm install primeng@17.18.8 @grapecity/gcpdfviewer@7.1.2
```

APP

 ionic v8

調整 UofxDatPickerModule import 路徑

```
-import { UofxDatPickerModule } from '@uofx/app-components/form';  
+import { UofxDatPickerModule } from '@uofx/app-components/date-picker';
```

調整 uofx-error-tip selector

```
-<uofx-error-tip>...</uofx-error-tip>  
+<uofx-form-error-tip>...</uofx-form-error-tip>
```

調整 uofx-user-select-view 屬性

```
<uofx-user-select-view  
- [uofUserSet]="users"
```

```
+ [userSet]="users"
</uofx-user-select-view>
```

外掛欄位 IonicModule

```
imports: [
- IonicModule
+ IonicModule.forRoot()
]
```

新功能

WEB

APP

新增提供 UofxFormTools 工具

```
import { UofxFormTools } from '@uofx/app-components/form';

export class Page {
  constructor(private uofTool: UofxFormTools) {}
}
```

 詳細使用方式請參考 [UOF X APP 元件>Reactive Form 工具包](#)

表單欄位提供與整張表單驗證勾稽方法

同 Web 版本提供 `UofxFormFieldLogic` 服務，並有函式 `parentFormBinding`、`setSelfControlValue` 與 `checkValidators` 可以使用。

 詳細使用方式請參考 [外掛欄位進階>表單送出前驗證欄位](#)

安裝 UOF X 新版套件包

套件名稱	版本
@uofx/app-components	3.8.1
@uofx/app-native	1.1.4
@uofx/core	3.4.1
@uofx/error-code	2.6.0
@uofx/icon	2.5.0
@uofx/plugin	4.0.6
@uofx/web-components	4.8.1

可複製下方指令，直接執行。

```
npm install @uofx/app-components@3.8.1 @uofx/app-native@1.1.4 @uofx/core@3.4.1 @uofx/error-code@2.6.0 @uofx/icon@2.5.0 @uofx/plugin@4.0.6 @uofx/web-components@4.8.1
```

升級部署流程

1. 請先調整專案並重新部署
2. 由於配置與設定檔有變更，請先更新 UOF x
3. 才至 UOF X 中操作檢查更新

 若不慎先使用了檢查更新

1. 若有「重載設定檔」可以使用，則直接操作。
2. 若沒有，請重新從 步驟1，加入新的版號與站台，再次操作 步驟3 即可。

8.2 2402 → 2407

本次 UOF X 由 2402 升級至 2407 注意事項如下：

新功能

✓ Web

提供一些全新的函式將複雜的邏輯包裝！

目前僅 Web 有提供 `UofxFormFieldLogic` 使用，App 的部分目前提供範例在 [Github Sample - mobile/advance-field.component.ts](#)，如有需要，請先自行做調整唷！

- 顯示欄位內的驗證失敗訊息
- 設定對應的欄位值
- 在暫存狀態下，不需要驗證欄位資料

DIALOG

多提供一個屬性 `showMaximizeBtn`，說明可參考 [Dialog 開窗](#)。

提供檔案上傳內容 `UOFXFILEMODEL` 類型

在跨欄位取得檔案上傳元件結果時，可以引用 `UofxFileModel` 來定義結果類型。

```
import { UofxFileModel } from '@uofx/app-components/file-helper';
```

破壞性異動

✓ Web

更換日期元件

日期底層元件改用 `flatpickr`，所以專案中需要安裝該元件。

```
npm install flatpickr@4.6.13
```

安裝 UOF X 新版套件包

套件名稱	版本
@uofx/app-components	2.12.0
@uofx/app-native	1.1.3
@uofx/core	2.11.0
@uofx/error-code	1.11.0
@uofx/icon	1.9.0
@uofx/plugin	2.0.0
@uofx/web-components	3.8.4

可複製下方指令，直接執行。

```
npm install @uofx/app-components@2.12.0 @uofx/app-native@1.1.3 @uofx/core@2.11.0 @uofx/error-code@1.11.0 @uofx/icon@1.9.0 @uofx/plugin@2.0.0 @uofx/web-components@3.8.4
```

升級部署流程

1. 請先調整專案並重新部署
2. 由於配置與設定檔有變更，請先更新 UOF x
3. 才至 UOF X 中操作檢查更新

 若不慎先使用了檢查更新

1. 若有「重載設定檔」可以使用，則直接操作。
2. 若沒有，請重新從 步驟1，加入新的版號與站台，再次操作 步驟3 即可。

8.3 2312 → 2402

本次 UOF X 由 2312 升級至 2402 注意事項如下：

新功能

- 升級 PrimeNG 至 16.9.1

```
npm install primeng@16.9.1
```

破壞性異動

- 套件包由 EJs Framework 改為 PrimeNG 16.9.1。

使用指令移除 EJs 所有套件。

```
npm uninstall @syncfusion/ej2 @syncfusion/ej2-angular-base @syncfusion/ej2-angular-buttons
@syncfusion/ej2-angular-calendars @syncfusion/ej2-angular-dropdowns @syncfusion/ej2-angular-grids
@syncfusion/ej2-angular-inputs @syncfusion/ej2-angular-lists @syncfusion/ej2-angular-maps @syncfusion/
ej2-angular-navigations @syncfusion/ej2-angular-notifications @syncfusion/ej2-angular-popups
@syncfusion/ej2-angular-progressbar @syncfusion/ej2-angular-richtexteditor @syncfusion/ej2-angular-
splitbuttons @syncfusion/ej2-base
```

安裝套件包。

```
npm install @uofx/app-components@2.7.0 @uofx/core@2.7.0 @uofx/error-code@1.6.0 @uofx/icon@1.5.0
@uofx/plugin@2.0.0 @uofx/web-components@3.3.1
```

- **@uofx/plugin-api** 更名為 **@uofx/plugin**，並將 api 功能收至 @uofx/plugin 內，所以請先移除 @uofx/plugin-api 後安裝 @uofx/plugin，所有原先有使用 @uofx/plugin+api 的路徑須改為 @uofx/plugin/api。
- 請參考我們所提供 [sample](#) 複製並覆蓋 **webpack.config.js**。
- 將所有設定檔加上對應的 \$schema 位置（請務必先安裝新版 @uofx/plugin 套件包），請參考 [配置與設定](#)。
- 調整 **plugin.manifest.json**
- 新增可設定 \$schema。
- 新增可設定 schemaVersion，本次出版為 "94"。
- 移除 vers，版本設定移至 **plugin.versions.json** 內。

plugin.manifest.json

```
{
  "$schema": "../node_modules/@uofx/plugin/schema/plugin-manifest.schema.json",
  "schemaVersion": "94",
}
```

```

"name": "一等一外掛模組範例",
"code": "Ede.Sample",
"description": "新一代 UOF X 外掛模組",
"manufacturerCode": "Ede",
"manufacturer": "一等一科技",
"production": false
}

```

- 新增 **plugin.versions.json**，修改了原先的相依特性，可以使用多版本以及設定變更說明。

```

plugin.versions.json
{
  "$schema": "../node_modules/@uofx/plugin/schema/plugin-versions.schema.json",
  "versions": [
    {
      "version": "3.0",
      "minimumUOFXVersion": "2.94.0",
      "changelog": [
        "新增了2個欄位",
        "修正了一些錯誤"
      ]
    },
    {
      "version": "2.0",
      "minimumUOFXVersion": "2.90.0",
      "changelog": [
        "本次更新除掉了幾條蟲"
      ]
    },
    {
      "version": "1.0",
      "minimumUOFXVersion": "2.89.1",
      "changelog": [
        "全新對話功能上線！"
      ]
    }
  ]
}

```

- **angular.json** 中需要額外匯出檔案，所以要在 `architect.build.options.assets` 中新增 **src/plugin.versions.json**。

升級步驟

1. 請先調整專案並重新部署
2. 由於配置與設定檔有變更，請先更新 UOF x
3. 才至 UOF X 中操作檢查更新

 若不慎先使用了檢查更新

1. 若有「重載設定檔」可以使用，則直接操作。
2. 若沒有，請重新從 步驟1，加入新的版號與站台，再次操作 步驟3 即可。

相依套件包

套件名稱	版本
@uofx/app-components	2.7.0
@uofx/app-native	1.1.3
@uofx/core	2.7.0
@uofx/error-code	1.6.0
@uofx/icon	1.5.0
@uofx/plugin	2.0.0
@uofx/web-components	3.3.0