ProjectWise Web China

二次开发

目录

*本文档只涉及二次开发扩展的配置，关于原生功能的配置请参考 ProjectWise Web China 管理员手册。

1. 个性化配置

1.1 PWWeb China logo 以及产品名称配置

描述： 用户可根据自身需求个性化更改 PWWeb China 的 logo 和产品名称。分为登录页配置（图 1）以及功能页配置

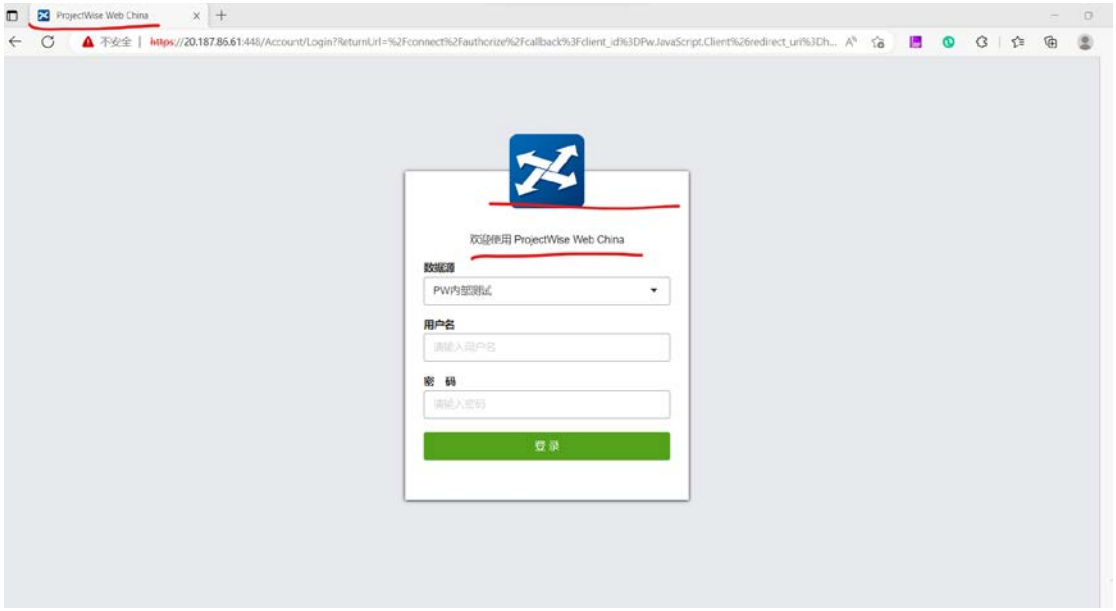


图 1

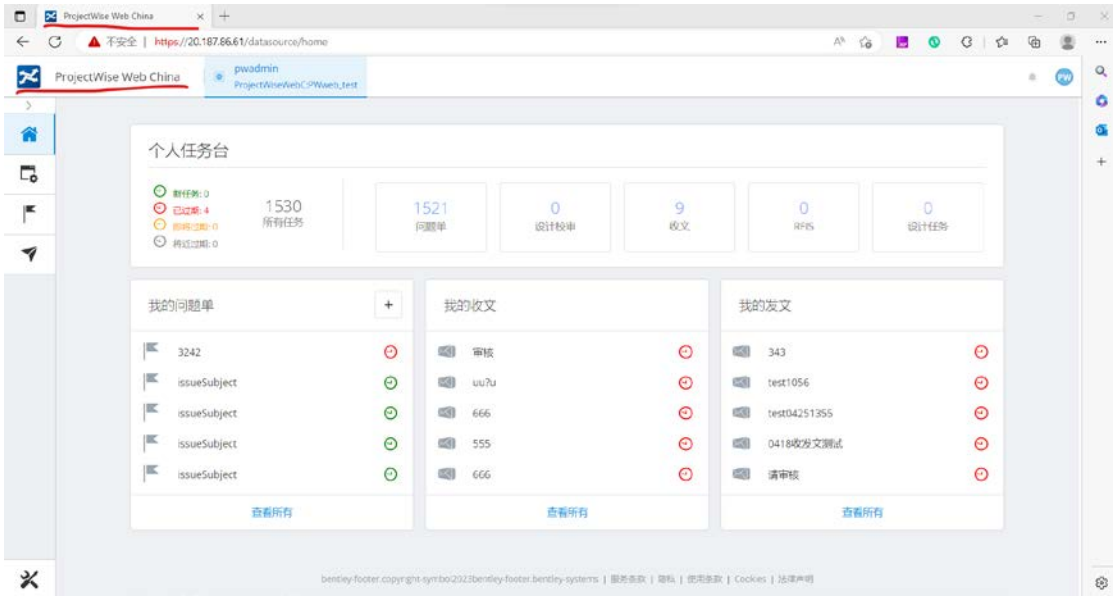


图 2

配置方法：

登录页配置：

图标配置在 IdentityServer/wwwroot 下更换 logo.svg 文件，PS：该文件必须命名为 logo.svg，图片格式只能是 svg 格式。

产品名称配置在 identityServer/ appsettings.json 下将 ProductName 更换为自定义名称。

产品配置：

图标配置在 PWWeb China 包中 images 目录下更换 logo.svg 文件，PS：该文件必须命名为 logo.svg，图片格式只能是 svg 格式。

产品名称配置在 PWweb 包 locales/zh/translations.json 下将 ProductName.ProjectWiseWebCN 的键值更换为自定义名称。

1.2 PWWeb-China CopyRight 配置

描述： 用户可根据自身需求个性化更改 PWWeb China 的 CopyRight。

配置方法：Copyright 配置在 PWWeb-China 包 locales/zh/translations.json 下将 footer.copyright-symbol 的键值更换为自定义名称。

1.3 PWWeb-China itwinViewer 个性化配置

描述： 用户可根据自身需求个性化更改 PWWeb China 的 itwinViewer 中的工具栏、背景色等。

配置方法：在 PWWeb-China 包下 configuration.json 下去配置 urlConfig 字段，其配置如下：

''urlConfig'': {

''backgroundColor'': ''steelBlue'', //背景色

```
  "backgroundMap": false,  //背景地图是否显示
  "hideStatusBar": false,    //是否显示状态工具
  "hideBaseTools": false,    //是否显示基础的工具
  "sky": false,              // 是否显示天空盒子
  "baseToolConfig": "0x00007FFF"
 },
```

baseToolConfig 中对应的是工具栏中各个功能求和，对应的 16 进制的枚举值如下：

```
enum ToolbarStatus {
 eToolbarRoamingTool  = 0x00000001,  //路径漫游
 eToolbarMeasureTool  = 0x00000002,  //测量工具
 eToolbarSectionTool  = 0x00000004,  //剖切工具
 eToolbarHideSectionTool  = 0x00000008,  //影藏剖切
 eToolbarIsolateSelectionTool = 0x00000010,  //隔离选择集
 eToolbarClearHideIsolateEmphasizeElementsTool  =  0x00000020,//清除隐藏和隔离状态
 eToolbarBackgroundColorsTool  = 0x00000040,  //背景色
 eToolbarViewAttributesTool  = 0x00000080, //视图属性
 eToolbarSearchElementTool  = 0x00000100,  //构件搜索
 eToolbarTreeItemTool  = 0x00000200,  //树结构
 eToolbarDisplaySystemItemTool  = 0x00000400, 按属性展示
 eToolbarViewSelectorItemTool  = 0x00000800,  //视图
```

```
  eToolbarAnnotationItemTool      = 0x00001000, //批注
  eToolbarDesignReviewIssueItemsTool = 0x00002000, //问题单
  eToolbarTimelineToolItemTool = 0x00004000 //工程进度
}
```

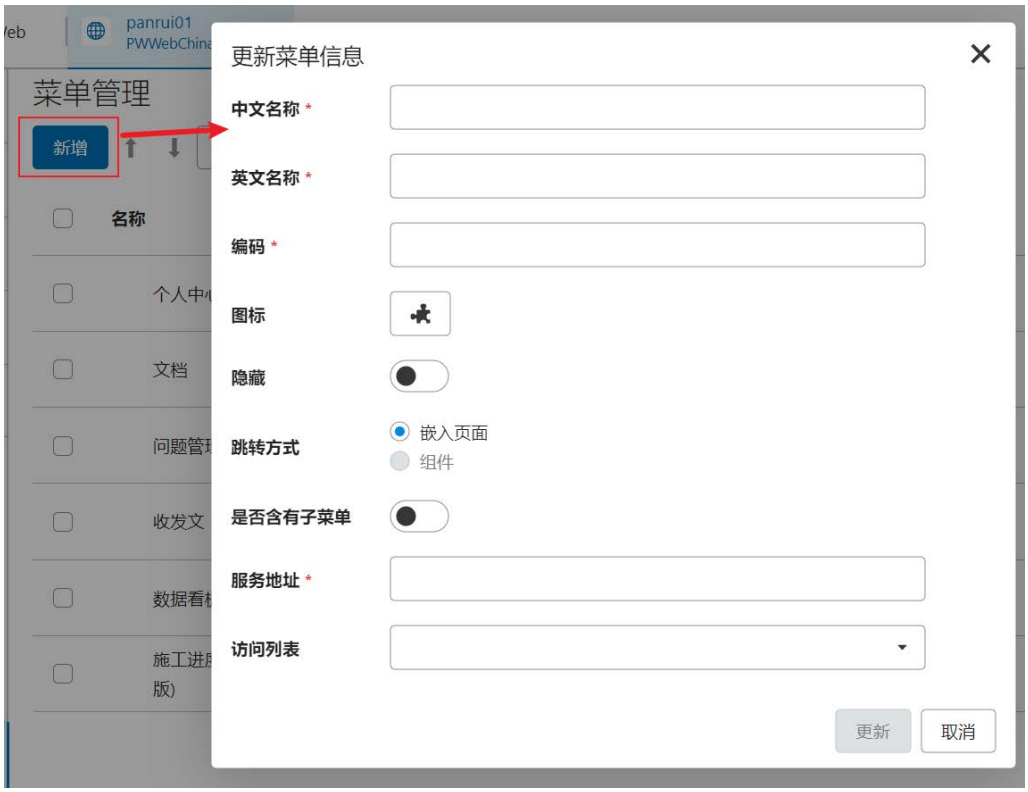
2. 基于 PWWeb-China 前端二次开发

1. 侧边栏功能二次开发（Iframe）

描述：可对 PWWeb China 现有的侧边栏菜单按照数据源进行扩展。

配置方法：

1. 使用管理员分组或受限制管理员分组的账号登录对应的数据源；
2. 登录成功后，点击侧边栏右下角“管理员”；
3. 点击“菜单管理”；
4. 点击“新增”进行侧边栏菜单扩展配置；



5. 在“更新菜单信息”弹出框，填写菜单的基本信息；

1. 中文名称：平台中文环境下菜单的显示名称；

2. 英文名称：平台英文环境下菜单的显示名称；

3. 编码：编码必须唯一，最好使用英文；

4. 图标：可以从本地上传 svg 图片，如不设置图标，会使用默认图标；

5. 隐藏：该菜单是否隐藏，如设置隐藏则访问平台时，不会显示该菜单。

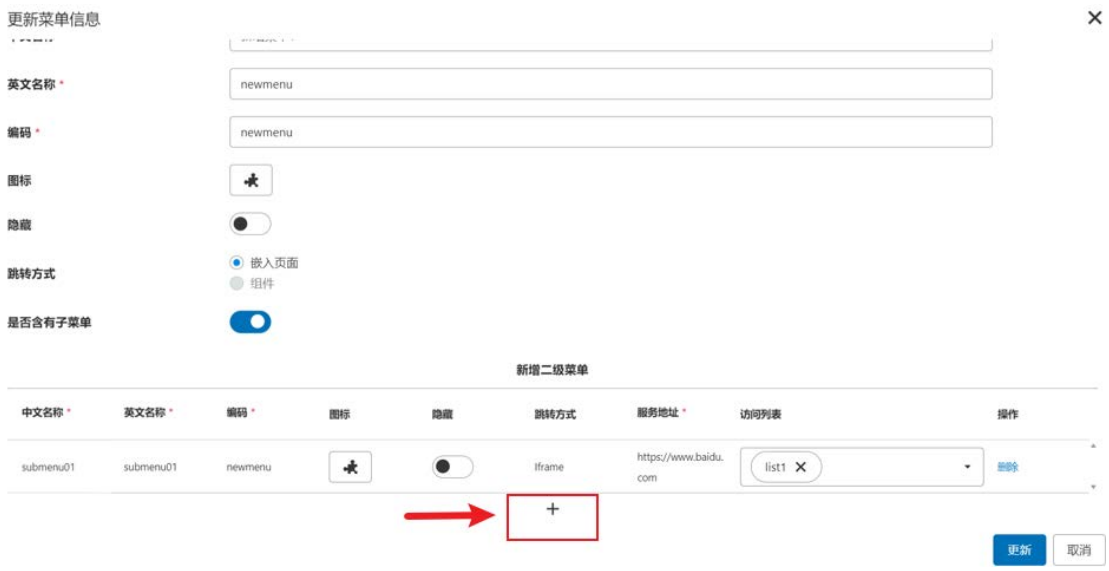
6. 跳转方式：默认选择嵌入页面；

如不含有子菜单，则服务地址为必填项，填写二次开发扩展功能的地址信息。

如需要控制菜单的访问权限，则在访问列表里选择可以访问该菜单的用户列表。如不做任何选择，则代表该菜单不受访问权限控制，数据源下有效的 PW 用户都可访问。

7. 如果一级菜单下需包含子菜单，则在“更新菜单信息”页面，打开“是否含有子菜单”选项（打开之后，一级菜单的“服务地址”将默认不生效）。点击下方“+”，按照上述相同要求填写二级菜单的基本信息，点击更新。

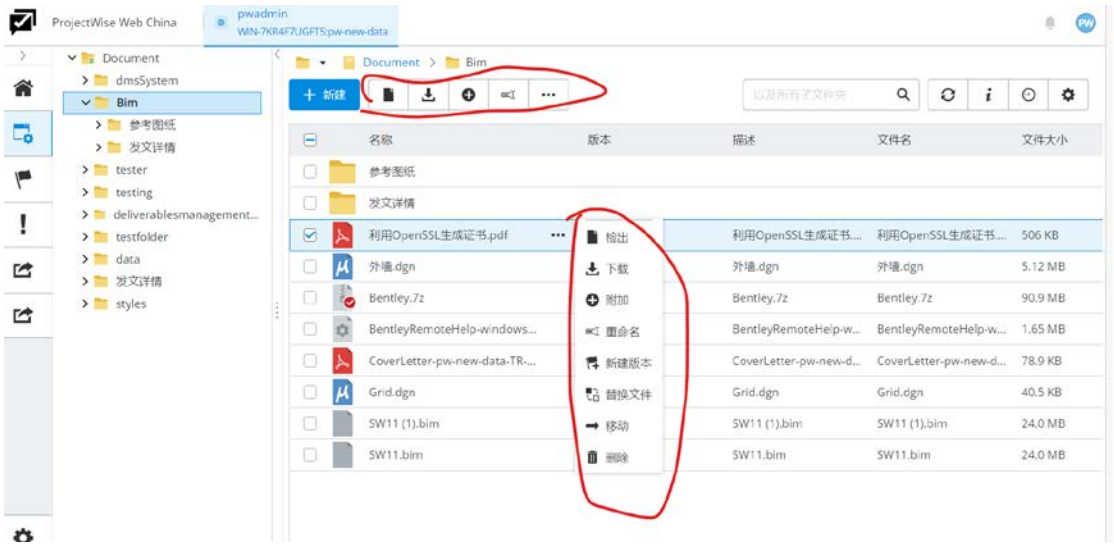
目前只能创建两级菜单。

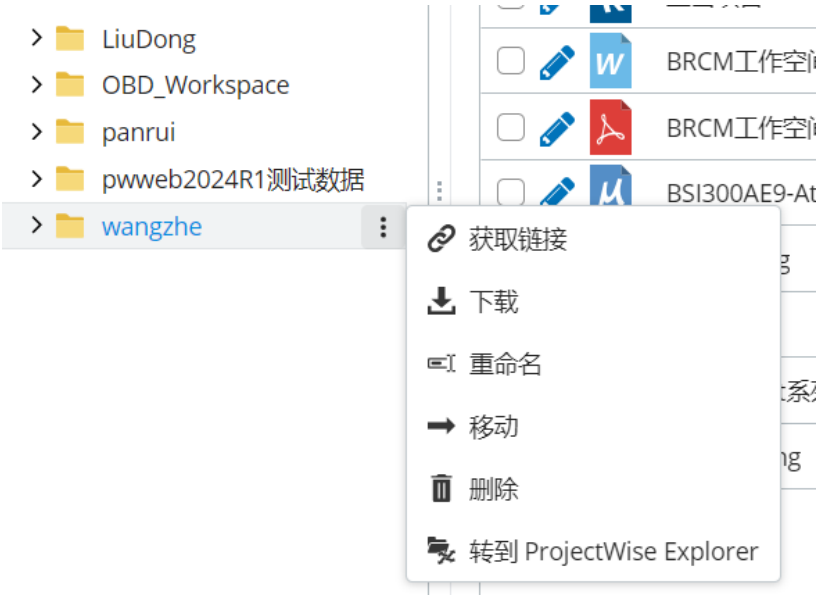


8. 对菜单也可以进行排序、删除、修改、设置默认路由操作。

2.4 文档模块关于文档文件夹右键功能二次扩展

描述：可在文档功能下对于文件夹和文档进行二次开发，可以增加二次扩展功能。





实现方法：

第三方：

1 插件功能实现：

1.1 格式：

文档或文件夹被操作的 plugin 的格式应该遵循如下 json，且当前不允许使用 hooks。

```
{
  title: string,  //操作的名字
  icon: React.JSX,    //显示的图标
  onClick: () => {}        //点击所触发的事件,
  disabled: boolean,      //是否被禁用
  hidden: boolean,        //是否被隐藏
};
```

1.2 参数：

为了第三方开发方便，对于在第三方可能会用到的一些一些组件或者参数，PWWeb-China 方将会以参数的形式传给第三方，目前 PWWeb-China 为第三方提供了：

参数名称	描述	用法
items	所选择的元素数组	
primaryModal	弹框方法	primaryModal.open（modal）:打开弹框，参数 modal 为弹框的内容的 JSX primaryModal.close():关闭弹框

以下为 plugin 实现的示例：

```
import React, { useState } from 'react';

import { Svg3D, SvgAddCircular, SvgImport } from '@itwin/itwinui-icons-react';

import { Modal, ModalContent, ModalButtonBar, Button } from '@itwin/itwinui-react'

import './test.css';

const PluginOne = (items, primaryModal) => {

 return {

  title: '导入到档案系统',

  icon: <SvgImport />,

  onClick: () => {
```

```
      primaryModal.open(
        <Modal
          isOpen={true}
          title="导入到档案系统"
        >
          <ModalContent>
            {items.name}导入到档案系统
          </ModalContent>
          <ModalButtonBar>
            <Button onClick={() => {primaryModal.close() }}>确认</Button>
          </ModalButtonBar>
        </Modal>
      )
    },
    disabled: false,
    hidden: false,
  };
};
export const config = {
  componentOne: (items, primaryModal) => PluginOne(items, primaryModal),
};
```

如果需要自定义多个 button 操作，可在同一 tsx 下定义多个，导出多个对象，也可在不同的 tsx 下定义分多个文件导出。PS：最终导出的时候请将开发好的组件放入 config 的对象中。PWWeb-China 将会便利该对象中的值。

2 打包

定义好 plugin 后就就可以将其进行打包，打包前配置好 webpack，在 webpack.config.js 中配置方式如下：

2.1 配置入口：指示 webpack 使用哪个模块作为构建内部。其中 fileName 表示：输出的文件名称，path 指所构建模块的路径。

```
entry: {
 fileName: 'path ',
},
```

2.2 配置出口：

```
output: {
 filename: '[name].js',     //以入口配置的名字为 name 在 dist 下
  libraryTarget: 'commonjs', //以 export 形式暴露给第三方
  },
```

2.3 配置共享依赖：第三方在文件中排除依赖，去依赖 pwweb-china 的 node-modules 的依赖。目前 pwweb-china 暴露给第三方共享的依赖有 react、react-dom、@itwin/itwinui-icons-react、@itwin/itwinui-react。

第三方的配置方法如下：

```
   externals: {
 react: 'react',
```

```
    "react-dom": "react-dom" ,
    "@itwin/itwinui-icons-react": "@itwin/itwinui-icons-react" ,
    "@itwin/itwinui-react": "@itwin/itwinui-react",
  },
```

2.4 源代码转换：对于 css/ts 等对应使用的预处理文件配置（根据项目需求配置对应的预处理器）。示例如下：

```
module: {
  rules: [
    {
      test: /\.(jsx|jsx|tsx|ts)$/,
      use: ['babel-loader'],
      exclude: /node_modules/,
    },
    {
      test: /\.css$/,
      use: ['style-loader', 'css-loader'],
      exclude: /node_modules/,
    },
  ],
 },
```

关于插件的 webpack 配置已完成，其余配置按照项目需求自行配置。

2.5 打包：在 webpack 配置结束后对开发好的插件进行打包(npm run build)。

3 部署

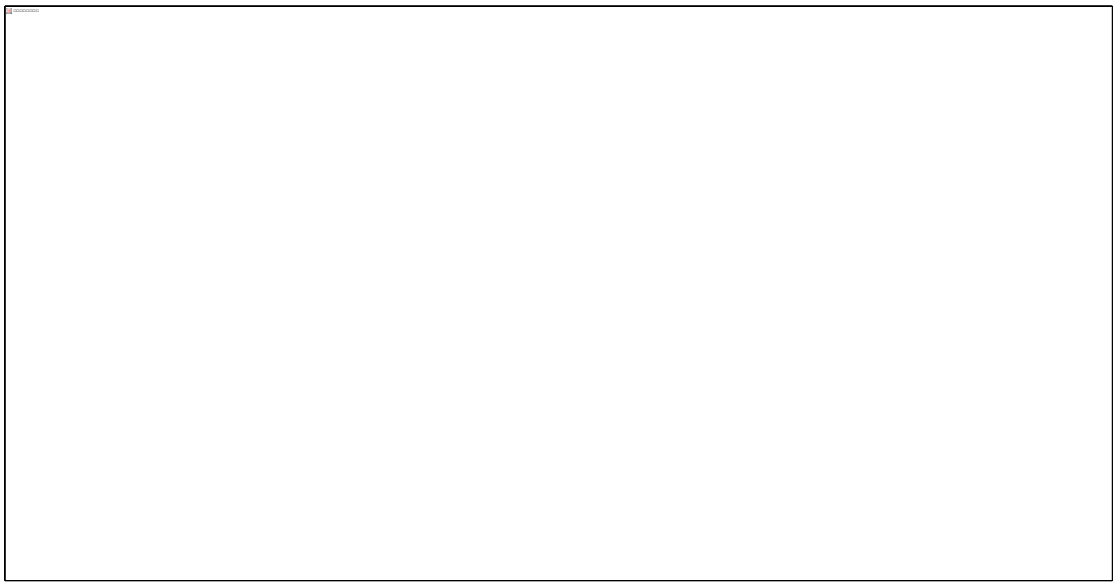
3.1 第三方打包完成的 dist 包部署在 IIS、nginx…等服务器上。

3.2 PWWeb-China 端：在 PWWeb-China 管理员—>文档管理—>功能管理下进行配置。

中文名称和英文名称不可填写，组件名称对应第三方对应导出组件的名称，服务地址对应的是第三方打包的路径。例如：

2.5 文档模块关于文档文件夹信息面板二次扩展

描述：可在文档功能下对于文件夹和文档展示的信息面板进行二次开发，如图圈出模块是可扩展的。



实现方法：

第三方：

1 插件功能实现：

1.1 格式：

文档或文件夹的信息面板的 plugin 的格式应该遵循如下 json，且当前不允许使用 hooks。

```
{
  tab: string,
  label:string,    //tab 栏显示的名字
  content: (React.JSX),      //需要展示的内容
};
```

1.2 参数：

为了第三方开发方便，对于在第三方可能会用到的一些一些组件或者参数，PWWeb-China 方将会以参数的形式传给第三方，目前 PWWeb-China 为第三方提供了：

参数名称	描述	用法
items	所选择的元素（只限于选择单个元素）	
PanelProperty	属性展示组件	<PanelProperty label={attribute.name} value={attribute.value.toString()} key={index} />、 label 为属性名；value 为属性值

以下为 plugin 实现的示例（PS：最终导出的时候请将开发好的组件放入 config 的对象中。PWWeb-China 将会便利该对象中的值。）：

```
import React from 'react';

import './test.css';

const PluginOne = async (items, PanelProperty) => {

 return {

  tab: '自定义属性',

  label: '用户属性',

  content: (

   <div>属性 1：{items.desc}</div>

  )

 };
```

```
};

const PluginTwo = (items, PanelProperty) => {
  const attributes = [
    {
      name: '测试 1',
      value: '测试 1 值'
    },
    {
      name: '测试 2',
      value: '测试 2 值'
    }
  ];
  return {
    tab: '自定义属性 1',
    label: '客户属性',
    content: (
      <>
        {
          attributes.map((attribute, index) => (
            <PanelProperty label={attribute.name} value={attribute.value.toString()}
key={index} />
```

```
          ))
        }
      </>
    )
  };
}

export const config = {
  infoOne: (items, PanelProperty) => PluginOne(items, PanelProperty),
  infoTwo: (items, PanelProperty) => PluginTwo(items, PanelProperty),
};
```

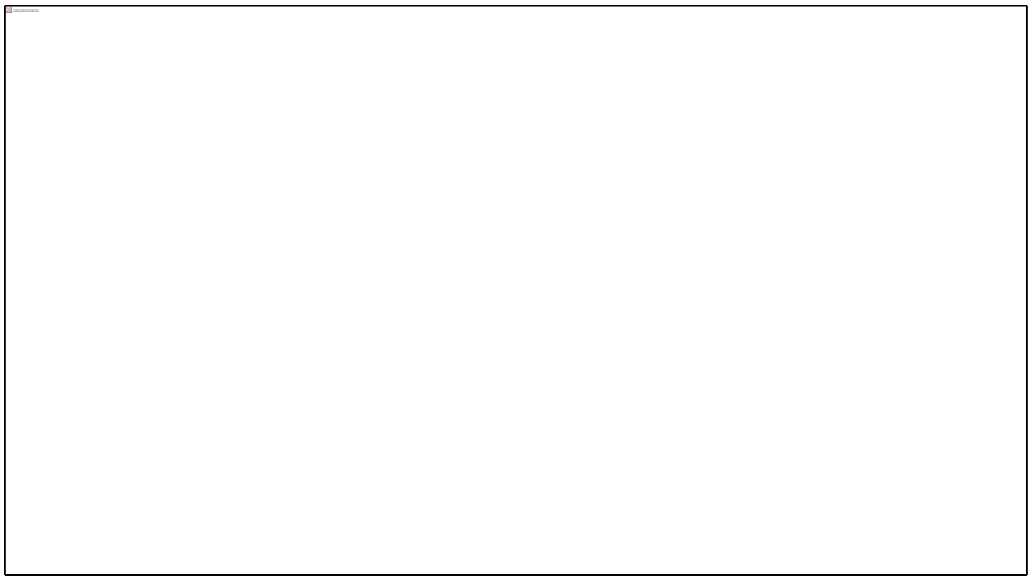
2 打包

打包配置方法可参考 2.2.

3 部署

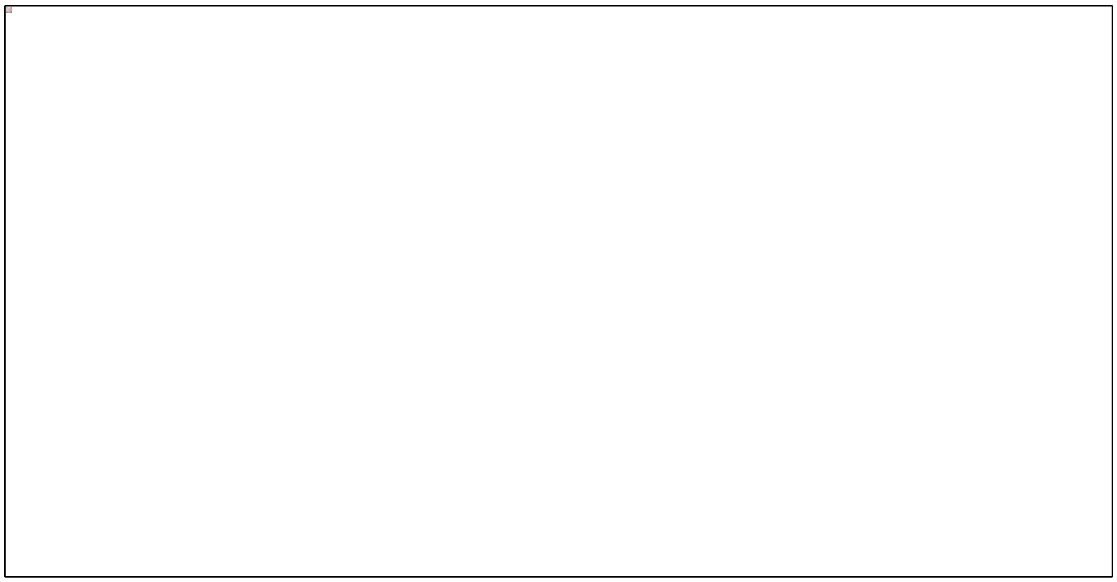
3.1 第三方打包完成的 dist 包部署在 IIS、nginx…等服务器上。

3.2 PWWeb-China 端：在 PWWeb-China 管理员—>文档管理—>功能管理，信息面板下进行配置插件的远程地址。其中组件名称对应第三方对应导出组件的名称。服务地址对应的是第三方打包的路径。



2.6 模型展示（iTwinViewer）模块关于右键菜单的二次扩展

描述：可在 ItwinViewer 下在构建上点击右键，弹出可操作菜单，菜单可由第三方自定义



实现方法:

第三方:

1 插件功能实现:

1.1 格式:

itwinViewer 右键菜单的 plugin 的格式应该遵循如下 json，

```
{
  id: string,
  item: {
   label:string,
   execute: (info, position, renderToString, existFunction
) => {
  } // 点击触发的事件，PW 方回传递相关参数予第三方
 };
```

1.2 参数：

为了第三方开发方便，对于在第三方可能会用到的一些一些组件或者参数，PWWeb-China 方将会以参数的形式传给第三方，目前 PWWeb-China 为在组件最顶层为第三方提供了：

参数名称	描述	用法
IModelApp	iTwin 操作相关模型需要用户接口，也可从 @itwin/core-frontend 引入	
t	翻译所需要的函数	t(“xxx”)

在点击事件 execute 中传入了以下参数：

参数名称	描述	用法

Info	右键选中的构件的信息，目前暂时是单选，多选后续会补上。其类型参照 itwin 中的 HitDetail	
position	其类型参数为：{ x: number, y: number }。 当前点击焦点的位置，如果第三方需要开发二次页面，可根据改参数进行定位计算	参照示例 PluginOne
renderToString	其类型为(element: ReactElement) => string。 主要用与将 ReactElement 转化为 string，注入到 itwin 中	参照示例 PluginOne
existFunction	将现有部分功能可添加到右键二次开发的按钮中直接调用。 existFunction: { createIssue: () => Promise<void>，//创建问题单 IssuePanel: () => Promise<void>，//问题单面板 annotation: () => Promise<void>，//批注 view: () => Promise<void>， // 视图 displayBySystem: () => Promise<void>, //按照属性显示	参照示例 PluginTwo

tree: () => Promise<void>, //树结构

SearchEle: () => Promise<void>

}, //构建搜索

ITwinRpcInterface	已在 PW 注册完成的 RPC，主要用于获取当前元素的 ECId 的子元素以及通过资源 id 获取元素的 ECId. 包括: getIdListByCodeList（_token: IModelRpcProps,_codeList: string[]）=》获取 Userlabel 列表对应的所有 Id 列表，包含子元素； getIdListByResIdList(_token: IModelRpcProps,_codeList: string[]) =》获取 ResId 列表对应的所有 Id 列表，包括族元素； getChildElements(_token:IModelRpcProps,_idList: string[]) =》 获取所有 ID 的子元素	参考示例 PluginHighLight

以下为 plugin 实现的示例，从点击出现弹框，展示构建信息、引用 PW 现有功能到右键菜单、右键高亮元素三个方面对 PW 的中 iTwinViewer 进行右键菜单的扩充（PS：最终导出的时候请将开发好的组件放入 config 的对象中。PWWeb-China 将会便利该对象中的值。）:

```
import React from 'react';

import './test.css';

import { ColorDef } from "@itwin/core-common";

import { EmphasizeElements } from "@itwin/core-frontend";



const PluginOne = (IModelApp, t) => {
```

```
const MenuEle = (info) => {
  return (
    <div className="circle">
      <div>模型信息</div>
      <br />
      <br />
      <div>
        {info._iModel._iModelId}
      </div>
      <br />
      <br />
      <div>
        {info._iModel._name}
      </div>
      <br />
      <br />
      <div>
        {info.sourceId}
      </div>

    </div>
```

```
    );
  }
  return {
    id: "test1",
    iconRight: "icon-devbeloper",
    item: {
      label: "获取构件信息",
      execute: (info, position, renderToString, existFunction) => {
        const parentDiv = document.createElement("div");
        const staticElement = renderToString(MenuEle(info));
        parentDiv.innerHTML = `${staticElement}`;
        IModelApp.uiAdmin.showHTMLElement(parentDiv, position,
          { x: -125, y: -125 },
          () => {
            IModelApp.uiAdmin.hideHTMLElement();
            console.log("cancel");
          })
      }
    }
  };
};
```

```
const PluginTwo = (IModelApp, t) => {
  return {
    id: "annotate",
    iconRight: "icon-devbeloper",
    item: {
      label: "批注",
      execute: async (info, position, renderToString, existFunction) => {
        existFunction.annotation()
      }
    }
  };
}
const PluginHighLight = (IModelApp, t) => {
  async function getChildren(ecIds, view, ITwinRpcInterface) {
    const imodel = view.iModel;
    const prop = imodel.getRpcProps();
    const ids = await ITwinRpcInterface.getClient().getChildElements(prop, ecIds);
    return ids;
  }
  async function hilightElements(vp, eleIds, _applyZoom) {
```

```
if (!vp) return;
const vf = vp.viewFlags;
vp.changeBackgroundMapProps({
  transparency: 0.01,
})
vp.viewFlags = vf;
vp.synchWithView({noSaveInUndo: false,animateFrustumChange:false});
const provider = EmphasizeElements.getOrCreate(vp);
provider.clearEmphasizedElements(vp);
provider.clearOverriddenElements(vp);
provider.overrideElements(
  eleIds,
  vp,
  ColorDef.red,
  0,
  true
);
provider.wantEmphasis = true;
provider.emphasizeElements(eleIds, vp, undefined, false);
if (_applyZoom) {
  const viewChangeOpts:any = {};
```

```
    viewChangeOpts.animateFrustumChange = true;
    vp.zoomToElements(eleIds, { ...viewChangeOpts }).catch((error) => {
      console.error(error);
    });
  }
}
return {
  id: "HighLight",
  iconRight: "icon-devbeloper",
  item: {
    label: "高亮到元素",
    execute: async(info, position, renderToString, existFunction, ITwinRpcInterface) => {
      const vp = IModelApp.viewManager.selectedView;
      if (vp) {
        const idArray = [info.sourceId];
        console.log(idArray);
        const allIds = await getChildren(idArray, vp.view, ITwinRpcInterface);
        await hilightElements(vp, allIds, true);
      }
    }
  }
```

```
    };
  }

  export const config = {
    componentOne: (IModelApp, t) => PluginOne(IModelApp, t),
    componentTwo: (IModelApp, t) => PluginTwo(IModelApp, t),
    componentThree: (IModelApp, t) => PluginHighLight(IModelApp, t),
  };
```

2 打包

打包配置方法可参考 2.2.

3 部署

3.1 第三方打包完成的 dist 包部署在 IIS、nginx…等服务器上。

3.2 PWWeb-China 端：在 PWWeb-China 包下 config/extension.json 配置 information 中配置插件的远程地址。其中 name 对应第三方对应导出组件的名称。pluginUrl 对应的是第三方打包的路径。

例：

```
{
  " iTwinRightMenu
": [
    {
      "name": "infoOne",
```

```
        "pluginUrl": "http://10.232.178.15:9001/information.js"
      },
      {
        "name": "infoTwo",
        "pluginUrl": "http://10.232.178.15:9001/information.js"
      }
    ]

}
```

2.7 模型展示（iTwinViewer）模块关于工具栏的二次扩展

描述： 对 itwinViewer 现有的工具栏的功能进行扩展。如下图所圈部分进行二次扩展

前端配置修改:

前端部署 config 目录下，extension.json 文件 iTwinExtensionProvider 节点下添加

如下示例内容:

```
//config file /config/extension.json
{
    "itemAction": [
    ],
    "information": [
    ],
    "iTwinExtensionProvider": [{
        "jsUrl": "http://localhost:3006/dist/index.js",
        "manifestUrl": "http://localhost:3006/package.json",
        "registerHost": "localhost:3006"
    },{
        "jsUrl": "http://localhost:3007/dist/index.js",
        "manifestUrl": "http://localhost:3007/package.json",
        "registerHost": "localhost:3007"
    }]
}
```

扩展组件实现方法:

创建 Manifest

```
// package.json
{
```

```
  "name": "my-new-extension",
  "version": "0.0.1",
  "main": "./dist/index.js",
  "type": "module",
  "activationEvents": [
    "onStartup"
  ]
}
```

添加需要的 dependencies

```
  // package.json
  "dependencies": {
    "@itwin/core-extension": "^3.2.0"
  },
  "devDependencies": {
    "typescript": "~4.4.0",
    "@itwin/build-tools": "^3.2.0",
  },
```

package.json 中添加

```
// package.json
  "scripts": {
```

```
    "build": "node esbuild.js"
  }
```

添加新的 tsconfig.json 文件

```
// tsconfig.json
{
  "extends": "./node_modules/@itwin/build-tools/tsconfig-base.json",
  "include": ["./*.ts", "./*.tsx"]
}
```

添加 esbuild.js 文件

```
// esbuild.js
import { NodeModulesPolyfillPlugin } from "@esbuild-plugins/node-modules-polyfill";
import { NodeGlobalsPolyfillPlugin } from "@esbuild-plugins/node-globals-polyfill";
import path from "path";
import esbuild from "esbuild";
import { fileURLToPath } from "url";
import { argv } from "process";
```

```
const dir = path.dirname(fileURLToPath(import.meta.url)).replace(/\\/g, "/");

const arg = argv.length > 2 ? argv[2] : undefined;


esbuild
  .build({
    entryPoints: ["src/index.ts"],
    bundle: true,
    minify: true,
    define: { global: "window", __dirname: `"${dir}"` },
    outfile: "dist/index.js",
    plugins: [new NodeGlobalsPolyfillPlugin(), new
NodeModulesPolyfillPlugin()],
    format: "esm",
  })
  .catch(() => process.exit(1));
```

创建 Extension

Src 目录下添加 tool.ts 文件,并添加代码

```
// src/tool.ts
```

```
import { PrimitiveTool } from "@itwin/core-extension";


export class ExtensionTool extends PrimitiveTool {
  public static override hidden = false;
  public static override toolId = "ExtensionTool";
  public static override namespace = "Extensions";
  public static override iconSpec = "icon-select-single";
  public async onRestartTool(): Promise<void> {
    return this.exitTool();
  }
  public override async run(): Promise<boolean> {
    console.log("Extension tool clicked!");
    return super.run();
  }
}
```

Src 目录下添加并实现 index.ts 文件

```
// src/index.ts


import { registerTool } from "@itwin/core-extension";
```

```
import { ExtensionTool } from "./tool";


export default function main() {
   console.log("Hello from Extension!");
   void registerTool(ExtensionTool);
   console.log("Tool Registered");
}
```

编译后目录结构如下

```
my-itwin-extension
|   package.json
|   esbuild.js
|   tsconfig.json
|
└──src
|   |   index.ts
|   |   tool.ts
|
└──dist
    |   index.js
```

Pwweb 前端目录下添加 iTwinExtensionProvider 相关配置

//config file /config/extension.json

{

"itemAction": [

```
  ],
  "information": [
  ],
  "iTwinExtensionProvider": [{
    "jsUrl": "http://localhost:3006/dist/index.js",
    "manifestUrl": "http://localhost:3006/package.json",
    "registerHost": "localhost:3006"
  },{
    "jsUrl": "http://localhost:3007/dist/index.js",
    "manifestUrl": "http://localhost:3007/package.json",
    "registerHost": "localhost:3007"
  }]
}
```

9. iTwinViewer 的引用

描述：可以将 iTwinViewer 的模块单独出来，在 PW 其余模块或者第三方进行引用。

实现方式: 第三方或者是 PW 其余模块可以通过 iframe 的形式去嵌入 iTwinViewer，具体的链接形式如下：

http://localhost:3000/datasource/designreviews/view?fld=${fld}&file=${res.serverpath}&backgroundColor=steelBlue&backgroundMap=false&hideStatusBar=true&hideBaseTools=false&baseToolConfig=0&sky=false&hideHeaderBar=true

上述链接中 http://localhost:3000 表示 PWWeb-China 的部署地址；fld=${fld} 表示 bim 文件在 PWWeb-China 中的 guid，file=${res.serverpath} 表示 bim 文件在 server 中的地址，这个地址的获取可以通过 CNServer 请求接口 /api/DesignReview/GetPreviewUrl?fileGuid="xxx"，来获取到地址，其中 token 可以通过 PWWeb-China 的 iframe 中的 url 参数 AccessToken 值对应的是请求接口所需要的 token；backgroundColor=steelBlue 表示背景色；backgroundMap=false 表示背景地图；hideStatusBar=true 表示是否隐藏状态栏；hideBaseTools=false 表示是否隐藏工具栏；baseToolConfig=0 表示工具栏需要打开的那工具对应的数字，具体定义参考 1.4 中 PWWeb-China itwinViewer 个性化配置中 ToolbarStatus 定义对应值；sky=false 表示是否打开天空盒子；hideHeaderBar=true 表示是否隐藏 PWWeb-China 上方的状态栏。

10. 基于第三方嵌入 PWWeb-China 功能

描述：第三方的 web 端可以嵌入 **PWWeb-China** 所具备的功能（问题单/文档（以及定位高亮到的文档）/设计校审/收发文）。

实现方式：对于第三方想要嵌入 **PWWeb-China** 某些功能，可以在第三方入口用 **iframe** 形式链接到 **PWWeb-China** 的功能模块。第三方可通过 API 接口的方式获取到 PWWeb-China 的认证 Token。

4.1 获取认证 Token

如果想使用域账号不输入密码即可登录获取到 token，需要配置代理账号。

在第三方后端服务器或者前端服务中调用

①：CNServer 中 api/Login/Initialize; 请求方式为 Get;

②：调取 Identity 端服务接口 Account/LoginPassword?datasource=` +

datasource + '&userName=' + userName + '&passWord=' + password

其中 datasource 表数据源，userName 表用户名，passWord 表密码（加密后），如果为域账号，可以不填密码为空即可。请求方式 Get；（如果要使用 pw 部署中的 nginx 代理，其代理为//）

③调取 Identity 端服务接口
Token/GetToken?UserName=${userName}&DataSource=${datasource}&Password=${passWord}&GrantType=password`

其中 DataSource 表数据源，UserName 表用户名，Password 表密码（加密后），如果为域账号，可以不填密码为空即可。其余均为固定参数。请求方式 Get。

以上接口均为链式调用，初始化完成后执行用户名密码登录，用户名密码登录完成后执行获取 token。

4.2 在第三方系统嵌入 PWWeb

其 URL 定义如下：

"https://xx.xx.xx:xxx/datasource/documents?accessToken=" + token;

黄色标注分别对应 PWWeb-China 包部署的地址/模块的名字（问题单 issues/文档 docments/设计校审 designreview/收发文 transmittals）/通过 Api 获取的 token。

如果第三方系统想定位高亮到具体的文档，URL 定义方式如下：

"http://xx.xx.xx:xxx/pwlink/datasource/documents?accessToken=" + token + '&datasource=xxx&objectId=xxx&objectType=xxx'

黄色标注分别对应 PWWeb-China 包部署的地址 /通过 Api 获取的 token/数据源名称（机器名!!数据源名称）/文档或者文件夹的 guid/文件类型（文档 doc/文件夹 folder）。

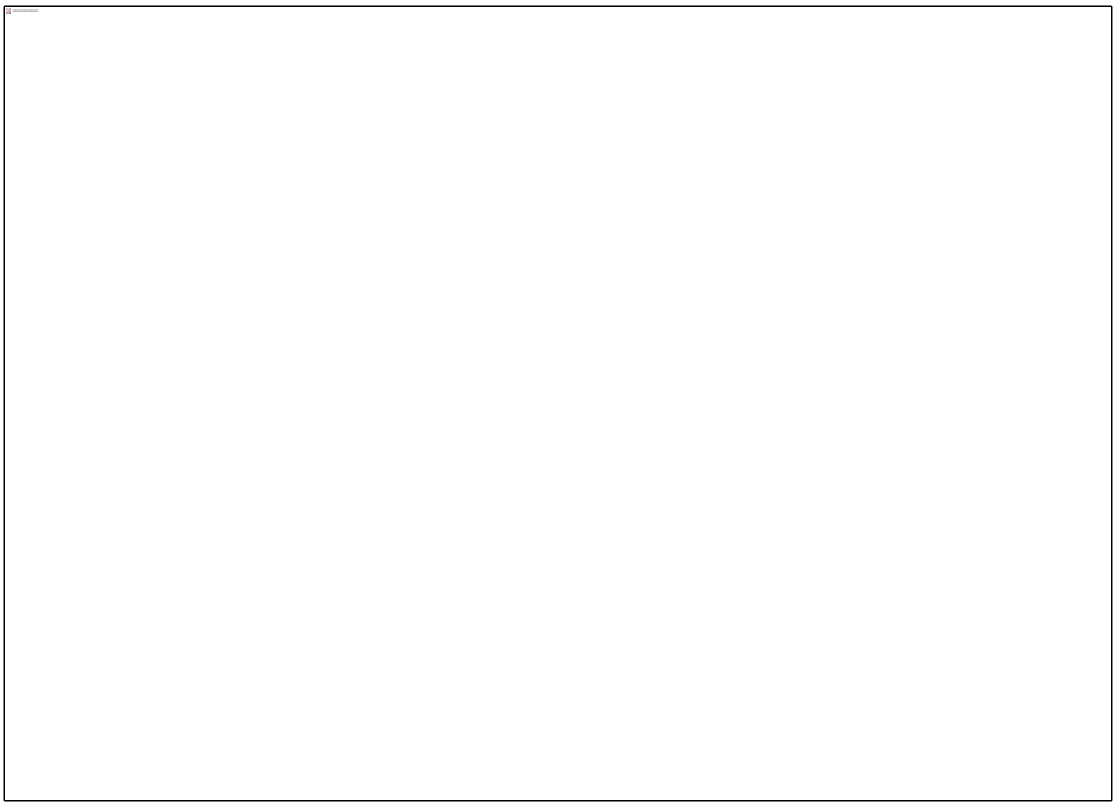
用户名密码的加密程序如下：

5、PWWeb-China 密钥动态配置

使用工具生成动态密钥对，此处我们使用一个公开的在线工具做为示例：

在线生成密钥对 https://www.metools.info/code/c80.html

密钥长度 2048，密钥格式 PKCS#1



拷贝私钥内容到 private_key.pem 文件中，拷贝公钥到 public_key.pem 文件中。

1. 替换 IdentityServer 安装目录下 key 目录下的 private_key.pem 和 public_key.pem 文件

2. 替换 ProjectwiseCNServer 安装目录下的 private_key.pem 文件

密钥更换完成。

使用其它工具，如 OPENSSL，生成密钥替换方法类似。

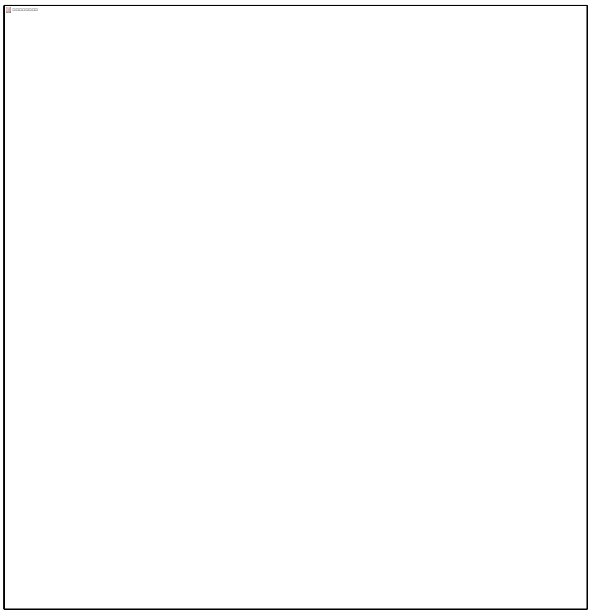
6、代理账号密码以及部署密码加密

1. ProjectwiseCNServer 部署目录下 appsetting.json 文件中如下节点

```
"Login": {
  "delegate": "",
  "delegateP": ""
}
```

Delegate 中配置 PW 用户名，delegateP 中配置密码，密码使用 DeployPasswordCrypto 压缩包中的 DeployCrypto.exe 加密。

注意：这里的 PW 账号要添加至管理员组，对该账号勾选 Enable as delegate user



1. dmskrnl.cfg 中添加[Trusted Servers]，将 nginx 前端服务器添加进信任列表。

   如下图所示，localCumputer 和 localCumputerName 名称可以自定义

   localCumputer=192.168.139.253

   localCumputerName=WIN-UCU50NMUJGO