森亿智能
SYNYI·AI

中国智慧医院整体解决方案提供商

通用运行时插件化方案

SYNYI . AI

021-58342202 contact@synyi.com

1.背景介绍

项目定制化带来的痛点——以SSO为例

森亿智能 SYNYI·AI

- 带来大量的代码分支，难以维护和升级
- 版本迭代发布会变得十分复杂和混乱
- 代码当中会夹杂大量和核心业务无关的定制业务或是外部依赖的代码，使整个项目变得臃肿
- 定制化需求的开发人员需要了解主程序代码，不利于项目管理

现有的项目定制扩展方案——编译时扩展

森亿智能 SYNYI·AI

通过在项目编译时的开关控制启用的模块，来实现项目定制化的需求

优点：

- 实施侧的交付成本降低
- 可以一定程度上避免不兼容问题
- 某些场景下性能优异

缺点：

- 对编译-打包工具链的要求过高
- 集成难度大
- 研发侧交付成本高

运行时插件扩展方案

森亿智能 SYNYI·AI

- 插件项目可以独立开发，保持主程序代码洁净
- 定制化项目不影响项目正常迭代发布
- 集成成本低，便于推广
- 前后端方案联动，达成统一
- 基于面向接口开发思维，插件开发者可以快速上手

森亿智能
SYNYI·AI

2.方案介绍

森亿运行时插件方案

后端（C#）

- 动态链接库
- RPC（规划中）

前端

- Iframe & WebComponent
- Native
- 注入

插件模式

森亿智能 SYNYI·AI

扩展

通过在**业务代码**中提供钩子，来对预设计的业务模型进行扩展。

通过在**框架上**提供扩展点，来实现预设计之外的定制化功能。

修改

通过修改现有的代码逻辑行为，来达到定制的需求。

森亿智能
SYNYI·AI

3.案例展示

案例——第三方鉴权（CA/钉钉/企业微信/友商SSO）

- 背景：现场通常会对接不同的第三方鉴权服务作为上游，全部集成在SSO中会使SSO变得十分臃肿。

- 实现：通过在标准鉴权流程中添加第三方鉴权流程并提供对应钩子，来加载插件中实现的第三方鉴权。

属于对预设计的业务流程进行扩展的插件类型

案例——第三方鉴权（CA/钉钉/企业微信/友商SSO）

- 背景：现场通常会对接不同的第三方鉴权服务作为上游，全部集成在SSO中会使SSO变得十分臃肿。

- 实现：通过在标准鉴权流程中添加第三方鉴权流程并提供对应钩子，来加载插件中实现的第三方鉴权。

属于对预设计的业务流程进行扩展的插件类型

案例——MDM员工同步

森亿智能 SYNYI·AI

- 背景：通过订阅MDM字典变动事件，来实现MDM员工字典与SSO用户信息的同步。同时提供订阅MDM员工字典上的属性字段，来实现MDM属性与SSO角色自动映射，并同步绑定的功能。

- 实现：通过插件在SSO宿主程序上新开线程来订阅MDM事件，并请求注入SSO的用户管理服务来同步用户和角色信息。

属于利用框架来实现预设计以外的定制化功能

案例——福建省立内外网互通

森亿智能 SYNYI·AI

- 背景：福建省立现场需要通过在外网能够通过钉钉登录到SSO

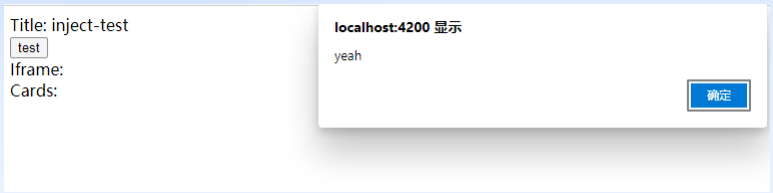
  福建省立的复杂网络环境具有以下特点：

  - 应用所在服务器无法访问钉钉服务器
  - 防火墙会将非200请求拦截

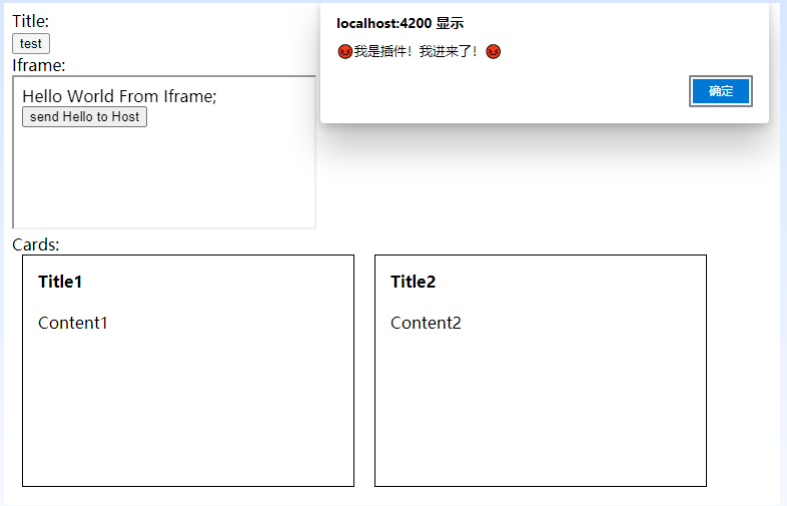
  导致处于的SSO无法按传统方式进行接入。
- 实现：
  - 在前置机部署一个SSO核心，称为外网SSO。
  - 使用插件在内网SSO增加可以使用username或id查询用户信息的接口。
  - 使用插件在外网SSO的DI树上注入改写的UserManager类，重写GetUserByID和GetUserByName为使用HTTP方式从内网SSO获取用户信息。

属于修改现有实现方式，来实现定制化的需求

前端Demo



Without Plugin Service



With Plugin Service

API

```
<div>Iframe:</div>
<div class="slot" #slot1></div>
```

```
async ngAfterViewInit(): Promise<void> {
  const plugins = this.pluginService.getSlotPlugin( slotid: 'slot1');
  if (plugins.length < 1) return;
  const ref = await plugins[0]?.render(this.slot1);

  ref.on('hello', (arg) => {
    alert( message: 'Hello From Frame!' + JSON.stringify( value: arg));
    ref.send('world', { yeah: 'yeah' });
  });
}
```

渲染扩展Iframe/WebComponent并与其通信

```
this.pluginService.executeSubscribedComponentCallback( componentType: TestComponent,  componentInstance: this);
```

注入脚本并修改现有组件或是监听路由

```
1  pluginHelper.subscribeComponent("TestComponent", (instance) => {
2    instance.title = sessionStorage.getItem( key: "hint");
3    instance.onClick = () => {
4      alert( message: "😡我是插件！我进来了！😡");
5    };
6  });
7
8  pluginHelper.subscribeRoute("/plugin/gz-sso", (route) => {
9    sessionStorage.setItem( key: "hint",  value: route.queryParams.hint);
10   return true;
11 });
12
```

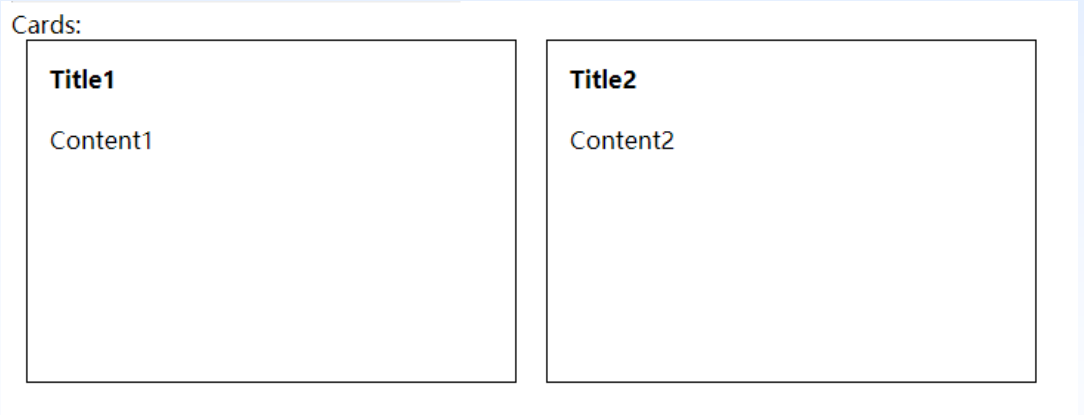
API

```ts
this.cards = this.pluginService
  .getSlotPlugin( slotId: 'test-card')
  .filter( predicate: (x) => x.type === 'native' && x.nativeOptions)
  .map((x) => x.nativeOptions);
```

```html
<ng-container *ngFor="let card of cards">
  <app-card [title]="card.title" [content]="card.content"></app-card>
</ng-container>
```

```json
{
  "type": "native",
  "slotId": "test-card",
  "nativeOptions": {
    "title": "Title1",
    "content": "Content1"
  }
},
{
  "type": "native",
  "slotId": "test-card",
  "nativeOptions": {
    "title": "Title2",
    "content": "Content2"
  }
}
```

通过配置来渲染预提供的组件



现有的缺陷

森亿智能 SYNYI·AI

- DLL插件和宿主程序在同一进程内运行，可能会因为插件导致宿主程序无法正常运行。
  - 插件也需要进行严格的Review
- DLL插件依赖处理的特点，容易出现由于不兼容而产生的种种问题。
  - 宿主程序和插件开发时要严格遵循面向接口开发的规范和原则
- DLL插件不建议使用.Net Core框架以外的语言来编写。
- 前端脚本注入方案受浏览器限制暂不能进行沙箱化处理。
  - 同样需要进行严格的Review

未来的规划 – 后端RFC插件框架

森亿智能 SYNYI·AI

- 基于RFC来实现插件和宿主之间的通信
- 完全意义上的隔离，可以避免插件对宿主造成不良影响
- 可以使用任意语言来编写

- 对基础设施要求高
- 可以实现的功能有限，不如DLL插件灵活

推广路标规划

森亿智能 SYNYI·AI

SSO运行时插件
化方案落地

8月

9月

通用运行时插件
化方案草案发布

通用运行时插件化方案在单
病种及BANK项目完成落地

11月

明年1月

通用开发套件（前端NPM
包，后端Nuget包）发布

召开宣讲会进行宣传，并为
协助业务线进行需求分析和
方案落地

通用开发套件进入标准迭代流程
维护文档
完善场景范例和相关实践博客

明年3月

森亿智能
SYNYI·AI

中国智慧医院整体解决方案提供商

THANK YOU
Q&A

SYNYI . AI

021-58342202 contact@synyi.com