**框架程序说明**

# 框架程序结构

由主程Afx.App和框架程序开发接口库Afx.PluginLib组成。主程序是运行时的EXE程序，对程序的界面、应用功能模块、应用资源配置、数据库连接等进行管理。接口库是应用模块模块开发的接口，应用功能模块按照接口库定义的接口构造类，根据系统的业务逻辑需要实现相应的功能，配置到界面配置文件里面，框架程序在启动时会自动配置模块，构造系统的用户界面。

主程Afx.App和开发接口库Afx.PluginLib一般保持稳定，不需要改动，特别的情况下在模块开发需要新的接口时，再修改Afx.App和Afx.PluginLib。

# 接口库

接口库是应用模块开发需要用到的库，包括应用模块功能类接口和其他辅助接口。

## 应用模块功能接口

应用模块由一个个类（功能类）组成，每个类继承自接口库的基类，实现相关的方法，完成相关的操作。如数据裁切提取，可以设计一个数据裁切提取的类，地图放大，可以设计一个地图放大的类。

命令的类别有以下几种：

1. Cmd类型：用于单击时实现一个操作，没有鼠标与地图的交互。
2. Tool类型：用于鼠标与地图交互，完成一些操作，如在地图上画矩形，然后按矩形范围查询等。
3. DockableWindow类型：停靠窗口类型的类，需要自定义一个控件，在控件内实现自已的业务逻辑，一般比较少用。

## 辅助类与接口

在开发应用模块功能时，还要实现与主地图的交互、获取数据资源、监听和处理主应用程序的事件等，这些操作要用到接口库里的辅助类，辅助类主要有：

1. MainApp：主应用程序的资源容器，封装了主应用程序提供给功能类的各类资源和接口。功能类在构造时，通过构建函数可以获取该对象。

主应用程序对象的主要属性和方法如下表，后续根据需要扩展。

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| AppPath | string | 应用程序路径 |
| CurrentMxdFile | string | 当前打开的MXD文档 |
| AppUI | IMainAppUI | 主应用界面接口 |
| MapControl | IMapControlDefault | 主地图控件 |
| TocControl | ITOCControlDefault | TOC(图层)控件 |

主应用程序对象提供了一些事件，功能类如果订阅了这些事件，在应用框架对应的事件发生时，也可以监听到，并执行一些处理操作，一般较少用到。主要事件如下表，后续根据需要扩展。

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| OnMapControlAfterDraw | IMapControlEvents2\_Ax\_OnAfterDrawEventHandler | 主地图绘制结束事件 |
| OnMapControlExtentUpdated | IMapControlEvents2\_Ax\_OnExtentUpdatedEventHandler | 主地图范围变化事件 |
| OnMapControlMapReplaced | IMapControlEvents2\_Ax\_OnMapReplacedEventHandler | 地图重新打开事件 |
| OnMapControlViewRefreshed | IMapControlEvents2\_Ax\_OnViewRefreshedEventHandler | 地图视图刷新事件 |

1. IMainAppUI：主应用界面接口，对主窗口的界面进行操作，主要的方法和属性如下表。后续根据需要扩展。

|  |  |  |
| --- | --- | --- |
| 名称 | 类型 | 说明 |
| AppCaption | string | 应用程序标题 |
| AppName | string | 应用程序名称 |
| AppForm | Form | 应用程序主窗体 |

# 功能模块开发示例

若需要通过鼠标与主地图交互（如在地图上用鼠标选择对象），继承Tool类，实现Tool类的方法。若不需要与主地图交互，继续Cmd类，实现 Cmd类的方法。若需要创建一个停靠窗口，需要继承DockableWindow，实现DockableWindow的方法。

以开发一个在主窗体中打开地图文档的功能类为例，代码及注释如下：

/// <summary>

/// 打开文档

/// </summary>

public class CommandOpenMxd : Afx.PluginLib.Cmd

{

//保存主应用程序对象

private Afx.PluginLib.MainApp hk;

public CommandOpenMxd()

{

}

#region ICommand 成员

/// <summary>

/// 操作

/// </summary>

public override string Operation

{

get { return "DEFAULT"; }

}

/// <summary>

/// 操作分组

/// </summary>

public override string OperationGroup

{

get { return "GenericCommands"; }

}

//功能类名称，注意与类一样

public override string Name

{

get { return "CommandOpenMxd"; }

}

public override string Category

{

get { return "GenericCommands"; }

}

//标题

public override string Caption

{

get { return "打开文档"; }

}

//提示

public override string Tooltip

{

get { return "打开文档"; }

}

public override bool Checked

{

get { return false; }

}

public override bool Enabled

{

get { return true; }

}

//对象创建，框架程序调用这个方法，会传入主应用对象，把主应用对象保存下来

public override void OnCreate(Afx.PluginLib.MainApp hook)

{

if (hook != null)

{

this.hk = hook;

//这里添加其它初始化代码

}

}

//单击主窗体的菜单按钮的操作：在这里实现打开地图文档的功能

public override void OnClick()

{

try

{

OpenFileDialog openMxdFileDialog = new OpenFileDialog();

openMxdFileDialog.Title = "请选择要打开的地图文档";

openMxdFileDialog.Filter = "mxd Files (\*.mxd)|\*.mxd";

openMxdFileDialog.RestoreDirectory = true;

openMxdFileDialog.Multiselect = false;

if (openMxdFileDialog.ShowDialog() == DialogResult.OK)

{

string mxdfilename;

mxdfilename = openMxdFileDialog.FileName;

//调用框架程序的MapControl打开mxd文档

this.hk.MapControl.LoadMxFile(mxdfilename, null, null);

this.hk.CurrentMxdFile = openMxdFileDialog.FileName;

//地图比例尺

}

}

catch

{

}

}

#endregion

}

# 界面配置

系统界面结构如下图：



系统界面配置文件为GeoDbSysUI.xml，配置说明如下：

<?xml version="1.0" encoding="utf-8" ?>

<AppUi>

<!--dll 功能模块配置，一个dll对应一个，功能模块要和exe放在一起-->

<PluginFiles>

<dll>DBApp.OverviewMap.dll</dll>

<dll>DBApp.GenericCommands.dll</dll>

<dll>DBApp.MapNavCommands.dll</dll>

<dll>DBApp.MapLayerOperation.dll</dll>

<dll>DBApp.PlatformAbout.dll</dll>

</PluginFiles>

<!-- defaultStyle 是界面显示风格，不动就可以了 -->

<UI defaultStyle="Office2007Blue">

<!-- title 系统启动时的显示标题 dbRootTitle数据库目录的根标题 -->

<App title="基础地理信息数据库系统" dbRootTitle="数据库目录"></App>

<!-- SystemMenu 配置大的系统按钮下面的子菜单 -->

<SystemMenu name="buttonItemSystem">

<!-- MenuItem 子菜单按钮配置，name是菜单命令名称，与dll里面的类的名称一样。 image菜单显示的图标文件，png格式，放在debug\data\image下面 buttonStyle：显示样式，图片和文字同时显示，如果只显示文字就是Text beginGroup:是否开始分组，或如果配置为1，表示开始分组，有分割线 -->

<MenuItem name="CommandOpenMxd" text="打开" image="" imagePosition="left" buttonStyle="ImageAndText" beginGroup="0"></MenuItem>

<MenuItem name="CommandNewMxd" text="新建" image="" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</SystemMenu>

<!-- QuickToolbar 快速工具条 配置大的系统按钮旁边的小图标的子菜单 -->

<QuickToolbar>

<MenuItem name="CommandNewMxd" text="" image="Open.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="CommandOpenMxd" text="" image="Options.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</QuickToolbar>

<!-- MenuTabs 配置主菜单下的tab 一个MenuTab对应一个 菜单面板 showTabPanels表示显示tab下的面板，双击可切换显示和隐藏 -->

<MenuTabs showTabPanels="1">

<MenuTab name="menuTabWorkspace" text="工作空间" isExisting="0">

<!-- MenuBar 是MenuTab 下的一个一个的框-->

<MenuBar name="menuBarWorkspace" text="地图文档操作" isExisting="0">

<MenuItem name="CommandNewMxd" text="新建文档" image="bb.png" imagePosition="top" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="CommandOpenMxd" text="打开文档" image="bb.png" imagePosition="top" buttonStyle="ImageAndText"></MenuItem>

</MenuBar>

</MenuTab>

<MenuTab name="menuTabMapView" text="地图视图" isExisting="0">

<MenuBar name="menuBarMapNav" text="地图浏览" isExisting="0">

<MenuItem name="CommandAddData" text="加载数据" image="bb.png" imagePosition="top" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="CommandFullExtent" text="全图" image="bb.png" imagePosition="top" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="ToolZoomOut" text="缩小" image="bb.png" imagePosition="top" buttonStyle="ImageAndText"></MenuItem>

</MenuBar>

</MenuTab>

<MenuTab name="menuTabQuery" text="查询浏览">

</MenuTab>

<MenuTab name="menuTabHelp" text="帮助" isExisting="0">

<MenuBar name="menuBarHelp" text="系统帮助">

<MenuItem name="CommandAbout" text="关于" image="bb.png" imagePosition="top" buttonStyle="ImageAndText" exists="1"></MenuItem>

<MenuItem name="CommandHelp" text="帮助" image="bb.png" imagePosition="top" buttonStyle="ImageAndText" exists="1"></MenuItem>

</MenuBar>

</MenuTab>

</MenuTabs>

<!-- 停靠窗口-->

<DockableWindowsDefs>

<MenuItem name="DockWinOverviewMap" text="缩略图" image="bb.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</DockableWindowsDefs>

<!-- 所有右键菜单 注意右键菜单分组的type是与类型（1个或多个）绑定的 -->

<ContextMenuDefs>

<!-- 右键菜单集合 TocObjType 表示绑定到TOC -->

<ContextMenuGroups type="TocObjType">

<!-- 右键菜单 TOC\_DataFrame 表示绑定到TOC的dataframe -->

<Group name="contextMenuDataFrame" type="TOC\_DataFrame" isExists="0" caption="DataFrame右键菜单">

<MenuItem name="CommandShowAllLayers" text="显示所有图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="CommandHideAllLayers" text="隐藏所有图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</Group>

<!-- 右键菜单 TOC\_DataFrame 表示绑定到TOC的GroupLayer -->

<Group name="contextMenuGroupLayer" type="TOC\_GroupLayer" isExists="0" caption="GroupLayer右键菜单">

<MenuItem name="CommandRemoveLayer" text="移除图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</Group>

<!-- 右键菜单 TOC\_DataFrame 表示绑定到TOC的FeatureLayer -->

<Group name="contextMenuFeatureLayer" type="TOC\_FeatureLayer" isExists="0" caption="FeatureLayer右键菜单">

<MenuItem name="CommandZoomToLayer" text="缩放至图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</Group>

<Group name="contextMenuRasterLayer" type="TOC\_RasterLayer" isExists="0" caption="RasterLayer右键菜单">

<MenuItem name="CommandRemoveLayer" text="移除图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="CommandZoomToLayer" text="缩放至图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText" beginGroup="1" ></MenuItem>

</Group>

<Group name="contextMenuRasterLayer" type="TOC\_MosaicLayer" isExists="0" caption="MosaicLayer右键菜单">

<MenuItem name="CommandRemoveLayer" text="移除图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

<MenuItem name="CommandZoomToLayer" text="缩放至图层" image="x2.png" imagePosition="left" buttonStyle="ImageAndText" beginGroup="1"></MenuItem>

</Group>

<Group name="contextMenuLegend" type="TOC\_LegendClass" isExists="0" caption="图例右键菜单">

<MenuItem name="CommandLegendColor" text="颜色" image="x2.png" imagePosition="left" buttonStyle="ImageAndText"></MenuItem>

</Group>

<!-- 右键菜单 还要按需扩展 -->

</ContextMenuGroups>

<!-- 右键菜单集合 DbObjType表示绑定到目录树 -->

<ContextMenuGroups type="DbObjType">

<!-- 数据库根 -->

<Group name="contextMenuDbRoot" type="DbCatalogRoot" isExists="0" caption="数据库目录根右键菜单">

</Group>

<!-- 右键菜单 绑定到目录 -->

<Group name="contextMenuDbCatalog" type="DbCatalog" isExists="0" caption="数据库目录右键菜单">

</Group>

<!--GDB数据库和对象注册相关 -->

<Group name="contextMenuGdb" type="Gdb" isExists="0" caption="Gdb数据库右键菜单">

</Group>

<Group name="contextMenuGdbVirtualDs" type="GdbVirtualDataset" isExists="0" caption="GDB虚拟数据集右键菜单">

</Group>

<Group name="contextMenuGdbFeatureDs" type="GdbFeatureDs" isExists="0" caption="GDB要素数据集右键菜单">

</Group>

<Group name="contextMenuGdbMosaicDs" type="GdbMosaicDs" isExists="0" caption="GDB镶嵌数据集右键菜单">

</Group>

</ContextMenuGroups>

</ContextMenuDefs>

</UI>

</AppUi>

# 数据配置

系统所用到的资源放在Debug\data\下面，各文件夹如下，以后可以按需增加。应用功能模块可以根据MainApp里的应用的相对路径来获取这些资源。

|  |  |
| --- | --- |
| 文件夹名称 | 说明 |
| CoordinateSystems | 坐标系统文件（Prj文件）的文件夹 |
| images | 系统用到的图标资源文件夹 |
| MapDoc | 地图文档文件夹，里面的default.mxd是框架程序启动时默认会加载的文档 |
| OverviewMapDoc | 缩略图文件夹，里面的overview.mxd是系统启动时缩略图模块默认会加载的文档 |