# 🚀 Gradio多功能工具平台 - 模块化版本

## 📋 项目概述

这是一个采用**分离式模块化架构**的Gradio应用，类似ComfyUI的设计理念。每个功能模块完全独立开发和维护，通过配置文件统一管理，实现高度的可扩展性和维护性。

## 🏗️ 架构特点

### 🧩 模块化设计

* **完全分离**：每个功能模块独立存在，互不干扰
* **配置驱动**：通过config.py统一管理所有配置和分类
* **动态加载**：模块可独立更新而不影响其他功能
* **扩展友好**：新增功能只需添加模块文件即可

### 📁 项目结构

├── app\_modular.py # 🎯 主应用程序（模块加载器和界面组织者）  
├── config.py # ⚙️ 配置管理文件（类似ComfyUI的节点定义）  
├── modules/ # 📦 功能模块目录  
│ ├── \_\_init\_\_.py # 📋 模块包初始化  
│ ├── ai\_image.py # 🎨 AI图像处理模块  
│ ├── ai\_chat.py # 💬 AI对话模块  
│ ├── ai\_video.py # 🎥 AI视频处理模块  
│ ├── image\_tools.py # 🖼️ 图像工具模块  
│ ├── video\_tools.py # 🎬 视频工具模块  
│ └── navigation.py # 🧭 平台导航模块  
├── start\_modular.sh # 🚀 模块化版本启动脚本  
├── requirements.txt # 📋 依赖包列表  
└── README\_模块化版本.md # 📖 本文档

## 🔧 功能模块

### 🤖 AI工具类

* **🎨 AI图像**：老照片修复、图片上色、高清放大
* **💬 AI对话**：智能问答、创意辅助、知识咨询
* **🎥 AI视频**：视频增强、文本生成视频、视频稳定化

### 🖼️ 图像工具类

* **📦 基础处理**：图片压缩、格式转换
* **✨ 图像增强**：参数调整、滤镜效果
* **📏 尺寸调整**：智能缩放、比例控制

### 🎬 视频工具类

* **🔧 基础处理**：格式转换、视频压缩
* **✂️ 视频编辑**：视频剪辑、添加水印
* **🎵 音频处理**：音频提取、格式转换

### 🧭 平台导航类

* **📂 分类导航**：开发工具、设计工具、AI工具、学习资源
* **🔗 自定义链接**：个人常用网站管理
* **🌐 网站嵌入**：iframe方式直接访问外部网站

## 🚀 快速开始

### 环境要求

* Python 3.8+
* pip包管理器

### 安装依赖

pip install -r requirements.txt

### 启动应用

#### 方法1：使用启动脚本（推荐）

bash start\_modular.sh

#### 方法2：直接运行

python3 app\_modular.py

### 访问应用

* 本地访问：http://localhost:7860
* 网络访问：http://0.0.0.0:7860

## ⚙️ 配置说明

### config.py 配置文件

#### 应用基本配置

APP\_CONFIG = {  
 "title": "🚀 AI多功能工具平台",  
 "description": "应用描述HTML",  
 "theme": "soft",  
 "css\_file": None  
}

#### 模块分类配置

MODULE\_CATEGORIES = {  
 "ai\_tools": {  
 "name": "🤖 AI工具",  
 "description": "人工智能驱动的各种处理工具",  
 "icon": "🤖",  
 "color": "#3b82f6"  
 },  
 # ... 其他分类  
}

#### 功能模块配置

每个模块都有详细的功能定义，包括输入输出参数、描述信息等。

## 🔧 开发指南

### 添加新模块

#### 1. 创建模块文件

在modules/目录下创建新的.py文件：

# modules/new\_module.py  
import gradio as gr  
  
class NewModuleProcessor:  
 def \_\_init\_\_(self):  
 self.name = "新模块"  
 self.description = "新功能描述"  
   
 def process\_function(self, input\_data):  
 # 处理逻辑  
 return result, status  
  
def create\_new\_module\_interface():  
 processor = NewModuleProcessor()  
   
 with gr.Tab("🆕 新模块"):  
 # 界面定义  
 pass  
  
\_\_all\_\_ = ["NewModuleProcessor", "create\_new\_module\_interface"]

#### 2. 更新配置文件

在config.py中添加模块配置：

MODULE\_CATEGORIES["new\_category"] = {  
 "name": "🆕 新分类",  
 "description": "新功能分类",  
 "icon": "🆕",  
 "color": "#10b981"  
}

#### 3. 注册模块接口

在app\_modular.py中注册新模块：

from modules.new\_module import create\_new\_module\_interface  
  
self.module\_interfaces["new\_module"] = create\_new\_module\_interface

#### 4. 更新模块导入

在modules/\_\_init\_\_.py中添加导入：

from . import new\_module  
\_\_all\_\_.append("new\_module")

### 模块开发规范

#### 必需组件

1. **处理类**：实现核心功能逻辑
2. **界面函数**：创建Gradio界面
3. **导出声明**：使用\_\_all\_\_声明导出接口

#### 代码风格

* 使用类型提示
* 完整的错误处理
* 详细的文档字符串
* 统一的命名规范

#### 界面规范

* 使用Tab布局组织功能
* 提供清晰的状态反馈
* 包含使用说明和示例
* 响应式设计适配

## 📊 系统监控

应用内置系统监控功能，可查看： - CPU和内存使用情况 - 磁盘空间状态 - 模块加载状态 - 实时性能指标

访问”ℹ️ 系统信息”标签页查看详细信息。

## 🛡️ 安全说明

### 文件安全

* 严格的文件类型检查
* 文件大小限制
* 路径遍历防护

### 处理安全

* 超时控制机制
* 资源使用限制
* 异常隔离处理

### 网络安全

* 输入验证和过滤
* XSS防护
* CSRF保护

## 🔄 版本管理

### 当前版本：v1.0.0 - 模块化架构版

* ✅ 完整的模块化架构实现
* ✅ 配置驱动的功能管理
* ✅ 6个核心功能模块
* ✅ 系统监控和状态显示

### 升级路径

1. **v1.1.0**：增加更多AI功能模块
2. **v1.2.0**：用户管理和权限系统
3. **v2.0.0**：插件市场和云端同步

## 🤝 贡献指南

### 报告问题

1. 检查现有Issue
2. 提供详细的复现步骤
3. 包含系统环境信息
4. 附上错误日志

### 提交功能

1. Fork项目仓库
2. 创建功能分支
3. 实现新功能模块
4. 添加测试用例
5. 提交Pull Request

### 代码审查

* 遵循代码规范
* 完整的测试覆盖
* 详细的文档说明
* 向后兼容性

## 📞 技术支持

### 联系方式

* 📧 邮箱：support@example.com
* 💬 问题讨论：GitHub Issues
* 📖 文档更新：Wiki页面

### 常见问题

**Q: 如何添加新的AI功能？** A: 在modules/目录创建新模块，实现处理类和界面函数，然后在配置文件中注册。

**Q: 模块间如何共享数据？** A: 可以通过配置文件定义共享配置，或使用事件机制进行模块间通信。

**Q: 如何自定义界面样式？** A: 修改config.py中的CUSTOM\_CSS配置，或在模块中使用自定义CSS。

## 📄 许可证

本项目采用MIT许可证，详见LICENSE文件。

## 🙏 致谢

感谢所有贡献者和用户的支持！

**🌟 如果这个项目对您有帮助，请给个Star支持！**