

# ChatVPet Process 设置教程

本教程将指导你如何在 **VPet.Plugin.ChatVPet** 的设置窗口 (winSetting) 中正确填写 API 信息、Embedding 信息及其他高级参数，让 ChatVPet 能够正常调用大语言模型对话。

## 目录

1. 打开设置窗口
2. 基本设置 — API 配置 - 使用 OpenAI API - 使用 DeepSeek API
3. 基本设置 — Embedding 配置 - 什么是 Embedding - 使用 OpenAI Embedding - 使用 DeepSeek Embedding
4. 基本设置 — 其他参数
5. 其他设置 — 语音输入与高级参数
6. 知识库设置
7. 数据库预览
8. 常见问题

## 1. 打开设置窗口

在 VPet 主界面右键桌宠 **ChatVPet** 设置 即可打开设置窗口。

## 2. 基本设置 — API 配置

设置窗口默认停留在 **基本设置** 标签页，需要填写以下字段才能让 ChatVPet 正常工作：

ChatVPet 使用 OpenAI 兼容API调用方法. 大部分AI服务商和本地AI部署器均支持 OpenAI API调用格式.

字段	说明
---	---
<b>API URL</b>	聊天接口地址，建议以 /v1 结尾
<b>API Key</b>	用于鉴权的密钥
<b>Web 代理</b>	可选 HTTP/SOCKS 代理地址，国内访问 OpenAI 等限制模型商时使用. 无需可不填
<b>Model</b>	使用的语言模型名称，可下拉选择或手动输入
<b>初始化文本</b>	系统提示词 System Prompt 不会被遗忘，字越多越贵
<b>温度</b>	回复随机性，范围 0.1 - 2，值越小越稳定，值越大越随机，默认 0.6
<b>单次最大花费</b>	单次请求允许消耗的最大 Token 数（在“其他设置”中设置，参见第 5 节）

**\*\*提示\*\***：初始化文本中 `{Name}` 占位符会被替换为桌宠的实际名称，可在初始化文本中使用，例如：  
``你是一只桌宠，你的名字叫{Name}``

### 注意: 模型需要支持 **ToolCall (工具调用)** 功能

目前已知 OpenAI, DeepSeek, Claude, Gemini, Gemma, Qwen, Glm, gpt-oss 等模型支持工具调用, 建议在使用前先查查相关文档

本教程使用2种API提供商作为示例教程, 你也可以使用其他API或本地模型(例如LM studio)

## 2.1 使用 **OpenAI API**

1. 前往 [OpenAI 平台](#) 注册账号并充值。
2. 进入 **API Keys** 页面, 点击 **Create new secret key** 生成密钥 (以 sk- 开头)。
3. 在设置窗口中填写: - **API URL** `https://api.openai.com/v1/` - **API Key**: 粘贴你的 sk-xxxxxxx 密钥 - **Model**: 推荐 gpt-4o-mini (性价比高) 或 gpt-4o

若在中国大陆访问, 需在 **\*\*Web 代理\*\*** 中填写代理地址, 或使用中转

### 常用模型参考

模型	特点
---	---
gpt-5.4-nano	便宜
gpt-5.4-mini	贵
gpt-5.4	很贵

## 2.2 使用 **DeepSeek API**

[DeepSeek](#) 提供与 OpenAI 接口完全兼容的 API, 在中文对话场景表现优秀, 价格也更具竞争力。

1. 前往 [DeepSeek 开放平台](#) 注册账号并充值。
2. 进入 **API Keys** 页面, 创建并复制你的 API Key (以 sk- 开头)。
3. 在设置窗口中填写: - **API URL** `https://api.deepseek.com/v1` - **API Key**: 粘贴你的 DeepSeek API Key - **Model** `deepseek-chat` (通用对话) 或 `deepseek-reasoner` (推理增强)

DeepSeek API 服务器在国内可直接访问, **\*\*Web 代理\*\*** 字段通常留空即可。

### 常用模型参考

模型	特点
---	---
deepseek-chat	通用对话, 速度快, 中文能力强
deepseek-reasoner	链式推理, 适合逻辑密集型任务

## 2.3 使用本地部署模型

除了使用云端 API, 您也可以在本地通过 **LM Studio** 或 **llama.cpp** 部署大语言模型, 实现离线运行、数据

隐私保护和成本控制。

## 方案一（LM Studio（有图形界面））

**LM Studio** 是一款跨平台桌面应用（支持 Windows、macOS、Linux），提供了直观的图形界面，能够轻松下载、加载和管理各类 GGUF 格式的本地模型。

部署步骤：

### 1. 下载并安装 LM Studio

访问 <https://lmstudio.ai/> 下载对应操作系统的安装包，完成安装。

### 2. 下载模型

打开 LM Studio 进入 **Model Search** 标签页，搜索并下载一个支持 Tool Call 的模型。建议选择 **8B** 或以上参数量的模型（例如 qwen3.5-35b-a3b、gemma-4-26b-a4b、glm-4.7-flash）。更小的模型容易出现格式错误。

### 3. 启动本地服务器

- 在左侧边栏点击 **Developer** 标签页。 - 选择 **Load Model**，选择已下载的模型，点击 **Start Server** 启动服务。 - 默认 API 地址为 `http://localhost:1234/v1`

### 4. 在 ChatVPet 中填写配置

参考以下信息填写设置窗口： - **API URL**：`http://localhost:1234/v1` - **API Key**：任意非空字符串即可（如 local） - **Model**：填写 LM Studio 中实际加载的模型名称（可在 Server 界面查看，如 glm-4.7-flash） - 其余字段（如温度、代理等）根据实际需求填写

**\*\*注意\*\*** LM Studio 的 `/v1/chat/completions`` 端点完全兼容 OpenAI API 格式，原生支持 **\*\*Tool Call**（函数调用）功能。只要模型本身具备 Tool Call 能力，ChatVPet 即可正常调用。

## 方案二（llama.cpp（命令行，性能更优））

**llama.cpp** 是一个高性能的 C/C++ 推理框架，支持 CPU 和 GPU 加速（CUDA、Metal、Vulkan 等），适合追求极致性能或需要在服务器端部署的用户。许多流行的本地运行工具（如 Ollama、LM Studio）底层都基于它构建。

部署步骤：

### 1. 获取 llama.cpp

[Install pre-built version of llama.cpp](#)

### 2. 下载模型

从 Hugging Face 等平台下载 GGUF 格式的模型文件。（例如 qwen3.5-35b-a3b、gemma-4-26b-a4b、glm-4.7-flash）。更小的模型容易出现格式错误。

### 3. 启动 API 服务器

使用内置的 llama-server 启动 OpenAI 兼容的 API 服务：

```
bash ./llama-server -m /path/to/model.gguf \ --host 0.0.0.0 \ --port 8080 \ -c 4096 \ --jinja --m:模型文件路径 --host / --port:监听地址和端口 --c:上下文长度（需与模型匹配） --jinja 启用 Jinja 模板解析，这是 Tool Call 正常工作的关键参数
```

### 4. 在 ChatVPet 中填写配置

- **API URL** `http://localhost:8080/v1` - **API Key** : 任意非空字符串 - **Model** : 填写模型名称 (可先访问 `http://localhost:8080/v1/models` 查看)

**\*\*提示\*\*** `llama.cpp` 的 API 服务完全兼容 OpenAI 格式, 原生支持 Tool Call。若需要 GPU 加速, 请根据您的硬件在编译时启用相应的后端 `NVIDIA` 使用 `CUDA` `Apple Silicon` 使用 `Metal`

自己部署模型选型的重要提醒：小模型可能导致 **Tool Call** 格式错误

并非所有本地模型都能完美支持 **Tool Call**。许多本地模型的智能程度还不足以准确识别何时需要调用工具, 以及如何按照 ChatVPet 要求的 JSON 格式输出调用指令。

- **9B** 以下的模型容易出现以下问题：
  - 输出内容中混杂了非标准 JSON 格式的工具调用描述, 而不是 ChatVPet 预期的 `tool_calls` 字段
  - 模型完全忽略工具定义, 直接以纯文本形式回答
  - 输出的 JSON 结构不完整或缺少必要字段 (如 `name` 或 `arguments`) , 导致解析失败
  - 被量化的模型因精度损失, 工具调用成功率大幅下降

因此, 建议：

本地部署时优先选择 **8B** 或以上参数量的模型, 如 `Qwen2.5-7B-Instruct` `Llama-3.1-8B-Instruct` `Mistral-7B-Instruct` 等, 这些模型对 Tool Call 的支持经过广泛验证, 相对成熟。

### 3. 基本设置 — Embedding 配置

#### 3.1 什么是 Embedding

ChatVPet 使用 向量嵌入 **Embedding** 技术为知识库、工具库和聊天记录建立语义索引, 从而在每次对话时智能检索最相关的内容, 而不是把全部内容塞进提示词——这样可以大幅节省 Token 消耗。

字段	说明
---	---
<b>Embedding URL</b>	Embedding 接口地址, 留空则使用主 API URL
<b>Embedding Key</b>	Embedding 接口密钥, 留空则自动使用主 API Key
<b>Embedding Model</b>	使用的 Embedding 模型, 默认 <code>text-embedding-3-small</code>

**\*\*注意\*\*** : 更换 Embedding 模型或 API 后, 请前往 **\*\*数据库预览\*\*** 标签页点击 **\*\*清除向量缓存\*\***, 以确保旧缓存失效, 重新生成向量。

#### 3.2 使用 OpenAI Embedding

若主 API 已填写 OpenAI 信息 `Embedding` 字段全部留空即可——程序会自动继承主 API 的 URL 和 Key

并使用默认模型 `text-embedding-3-small`

如需单独指定：

- **Embedding URL** `https://api.openai.com/v1`
- **Embedding Key**：与主 API Key 相同
- **Embedding Model** `text-embedding-3-small`或 `text-embedding-3-large`

### 3.3 使用由 LBGAME 提供的 `bge-m3` 模型

因为感觉这种Embedding模型比较小众,也正好还有多余的显卡服务器没用上,就部署了个显卡服务器放了个模型 未来可能会产生变化,不对持续性服务负责.

- **Embedding URL** `https://lolisbr.exlb.net/v1`
- **Embedding Key** `sk-Lolis-provides-everyone-with-free-embeddings-Lolis-is-great`
- **Embedding Model** `text-embedding-bge-m3`

注:服务器不收集用户信息,但是为了节约性能开销,会缓存 文本->向量 以避免重复计算

### 3.4 使用本地部署

参见 2.3, 多下一个 **BAAI/bge-m3** 模型即可

## 4. 基本设置 — 其他参数

字段	说明
---	---
累计花费	显示迄今为止消耗的 Token 总数, 只读
<b>Token 显示</b>	是否在聊天记录中显示每次消耗的 Token 数量
提交内容	是否将聊天记录提交给 LBGAME 以改进 ChatVPet

## 5. 其他设置 — 语音输入与高级参数

切换到 其他设置 标签页可配置语音输入及各类上限。

### 语音输入 `Azure AI Speech`

ChatVPet 支持通过 Azure 认知服务实现语音输入, 需先在 [Azure 门户](#) 创建 **Speech** 资源。

字段	说明
---	---
启用语音输入	开关，启用后聊天界面会显示麦克风按钮
语音密钥	Azure Speech 资源的订阅密钥
语音区域	Azure 资源所在区域，例如 eastasia[]westus
语音语言	识别语言代码，例如 zh-CN[]en-US[] <a href="#">查看支持列表</a>

### 对话与记忆参数

字段	默认值	说明
---	---	---
最大聊天记录	20	单次对话注入上下文的最大历史条数
最大工具库	10	每次对话最多召回的工具条数，越多越消耗 Token
最大知识库	10	每次对话最多召回的知识库条数，越多越消耗 Token
输出最大花费	4000	单次请求允许的最大 Token 数，建议不超过模型上限；
最大回合	5	工具调用允许的最大循环轮次（防止死循环）

### 历史压缩与日记参数

ChatVPet 会将超长历史压缩为日记，以节省 Token 并实现长期记忆。

字段	默认值	说明
---	---	---
压缩触发条数	24	聊天记录超过此数时自动触发压缩，设为 0 禁用
压缩保留最近数	10	压缩时保留最近几条对话不参与摘要
日记衰减率	0.03	每轮对话后未被命中的日记权重衰减比例（0.03 = 3%）
最大日记注入数	10	每次对话最多向系统提示注入的日记条数

## 6. 知识库设置

切换到 知识库设置 标签页，可在文本框中逐行添加自定义知识：

```
{name}喜欢吃草莓蛋糕
{hostname}喜欢吃草莓
{hostname}不喜欢上班, 因为老板老是让他加班
```

规则：- 每行一条知识 - 知识不会直接全部塞入提示词，而是由 Embedding 算法按相关度动态召回。 - 可以描述桌宠的人设、日常偏好、特定领域知识等。 - 支持使用 {name}指桌宠名字, {hostname}指玩家名字 - 重启游戏后生效

## 7. 数据库预览

切换到 数据库预览 标签页，可查看当前所有知识库、工具库和聊天记录。

- **搜索**：输入关键词后点击 **搜索**，进行文本精确匹配查找。
  - **向量搜索**：输入自然语言后点击 **向量搜索**，通过语义相似度排序，效果更好。
  - **清除向量缓存**：更换 Embedding API 或模型后，点击此按钮清除旧向量，强制重新计算。
  - **删除聊天记录**：在“聊天记录”子标签中右键选中条目，选择 **删除** 可移除特定历史对话。
- 

## 8. 常见问题

**Q: 填写 API 信息后保存 ChatVPet 没有反应？**

A: 请检查：

1. API URL 是否正确。
2. API Key 是否有效且未过期。
3. 若使用 OpenAI 在中国大陆需配置 **Web 代理**
4. 检查账户余额是否充足。

**Q: Embedding 是否必须配置？**

A: 是必须的. 若不知道填什么, 可以参考 **3.3**

**Q: 更换了 Embedding 模型，搜索结果变差了？**

A: 不同模型生成的向量不兼容。请在 **数据库预览** 页面点击 **清除向量缓存**，等待下次对话时重新生成向量。

**Q: 如何降低 Token 消耗？**

A: 可以：

- 减小 **最大聊天记录**、**最大工具库**、**最大知识库** 的值。
- 缩短 **初始化文本**
- 选择费率更低的模型