Chatwoot 集成指南 v2.0

本文档为项目中的 Chatwoot 功能提供了一份全面的集成、使用与开发指南。

📋 目录

---

1. Chatwoot 概述

什么是 Chatwoot？

Chatwoot 是一个开源的客户沟通平台，为企业提供实时聊天解决方案，让您可以通过网站、App 等多种渠道与客户建立联系，提供实时的客户支持，提升用户满意度。

核心优势

- 🚀 **实时在线聊天**: 为网站提供现代化、易于集成的聊天窗口
- 🎯 **多渠道整合**: 统一管理来自网站、邮件、社交媒体等不同平台的客户对话
- ⚡ **自动化与效率**: 支持快捷回复、自动分配对话等功能，提升客服团队工作效率
- 🔒 **开源与私有化部署**: 可部署在自己的服务器上，完全掌控客户数据和系统架构
- 🔧 **丰富的 API**: 提供强大的客户端和服务端 API，可与现有业务系统深度集成

为什么选择 Chatwoot？

相比传统的客服系统，Chatwoot 具有以下独特优势：

- **成本效益**: 开源免费，无需支付高昂的 SaaS 订阅费用
- **数据安全**: 私有化部署，客户数据完全在自己掌控之下
- **高度可定制**: 开源架构允许根据业务需求进行深度定制
- **现代化体验**: 提供类似现代即时通讯工具的用户体验

---

2. 核心概念解读

为了更好地理解 Chatwoot 的配置和使用，以下是一些您会经常遇到的核心概念：

📥 Inbox (收件箱)

- **定义**: 处理来自特定来源的所有消息的地方，每个收件箱都独立于其他收件箱
- **类比**: 可以想象成一个部门的专用邮箱
- **示例**: 为"销售部"创建一个收件箱处理售前咨询，为"技术支持部"创建另一个收件箱处理售后问题

Channel (渠道)

- **定义**: 客户与您联系的具体途径
- **类型**: 网站聊天、电子邮件、Facebook Messenger、WhatsApp 等
- **关系**: 每个渠道都会关联到一个收件箱

Agent (客服/代理)

- **定义**: 团队中使用 Chatwoot 平台回复客户消息的成员
- **权限**: 需要将客服分配到特定收件箱，才能看到并回复该收件箱中的消息
- **移动办公**: 支持通过官方手机 App 随时随地处理客户咨询

Conversation (对话)

- **定义**: 客服与一位客户之间的一系列完整的消息交流
- **状态**: 开启、已解决、等待中等
- **连续性**: 支持跨设备、跨时间的对话历史记录

Contact (联系人)

- **定义**: 与您开始对话的客户或访客
- **识别**: 通过 `setUser` 方法将匿名访客转换为具名联系人
- **属性**: 可通过 `setCustomAttributes` 设置丰富的业务属性

---

3. 业务场景与价值

典型沟通流程

网站访客与客服的基础沟通流程

1. **访客发起对话**
    - 访客点击网站右下角的聊天图标
    - 输入问题并发送（如"你们的 PCB 加急服务最快多久？"）
2. **Chatwoot 创建对话**
    - 系统在对应收件箱中创建新对话
    - 对话状态设为"开启 (Open)"
3. **通知客服**
    - 分配到该收件箱的客服收到通知
    - 客服在后台看到新对话
4. **客服响应**

- 客服认领对话并回复
- 回复实时显示在访客的聊天窗口

5. **完成沟通**

- 双方继续沟通直到问题解决
- 客服将对话标记为"已解决"

业务场景示例

场景一：售前咨询（匿名访客）

访客: "你好，我正在看你们的 4层板 报价页面，如果我需要添加阻抗控制，费用怎么计算？"

客服看到的信息:

- 访客消息内容
- 当前浏览页面：`https://next-pcb.com/quote?layer=4`（通过 `setCustomAttributes` 传递）

客服回应: "您好！很高兴为您服务。4层板的阻抗控制是免费提供的。您可以在报价页面的'特殊工艺'选项中勾选'阻抗控制'，然后上传您的层压结构文件即可。"

价值: 客服能够结合访客的上下文提供精准服务，提高销售转化率。

场景二：售后支持（已登录用户）

用户（李经理）: "我的订单 PCB20231101-005 生产到哪个阶段了？"

客服看到的信息:

- 用户姓名：李经理
- 邮箱：li.jingli@example.com
- 自定义属性：
  - 用户ID：usr_12345
  - 公司名称：某某科技有限公司
  - 最近订单号：PCB20231101-005

客服回应: "李经理您好！您的订单 PCB20231101-005 目前已经完成钻孔工序，正在进行沉铜电镀，预计明天可以进入线路制作环节。"

价值: 无需反复询问用户身份和订单号，提供专业高效的服务体验。

聊天历史的连续性

实现原理

通过 `setUser` API 中的 `identifier` 字段实现用户身份的永久标识。只要使用同一个唯一的 `identifier`，Chatwoot 就会自动拉取所有关联的历史对话。

我们的实现

使用用户数据库中的唯一 ID (`user.id`) 作为 `identifier`：

```
setUser({
  identifier: user.id, // 使用用户数据库 ID 作为永久标识
  name: user.display_name,
  email: user.email,
});
```

效果

用户在不同设备、不同时间登录网站，都能看到完整的聊天历史记录，提供无缝的沟通体验。

移动端办公支持

Chatwoot 提供功能完善的官方手机 App，支持 iOS 和 Android 平台。

主要功能:

- 接收新消息的实时推送通知
- 直接在 App 中回复对话
- 管理对话状态、分配任务、添加标签
- 查看完整的客户资料和自定义属性

下载地址:

- **iOS**: Apple App Store
- **Android**: Google Play Store

---

4. 技术架构设计

核心设计思想

我们采用基于 **React Context** 和 **自定义 Hook** 的现代化集成方案，核心思想是**"一次加载，全局共享"**。

设计优势

- **避免重复加载**: 通过全局 `ChatwootProvider` 一次性加载 SDK，避免页面切换时重复请求
- **状态集中管理**: 所有 Chatwoot 相关状态由 `ChatwootProvider` 统一管理
- **简化业务逻辑**: 业务组件通过 `useChatwootOptimized` Hook 轻松获取状态和 API

架构组成

1. 环境配置 (`.env.local`)

```
NEXT_PUBLIC_CHATWOOT_BASE_URL="https://chat.yourdomain.com"
NEXT_PUBLIC_CHATWOOT_WEBSITE_TOKEN="your_website_token"
```

2. SDK 加载器 (`lib/chatwoot-optimized.ts`)

负责动态创建 `<script>` 标签，异步加载 Chatwoot 的 `sdk.js`，并设置全局 `window.chatwootSettings`。

3. 全局提供者 (`components/ChatwootProvider.tsx`)

React Context Provider 组件，包裹整个应用根布局，负责：

- 调用 SDK 加载器
- 监听加载状态
- 向下层组件提供 Chatwoot 状态和 API

4. 自定义 Hook (`lib/hooks/useChatwootOptimized.ts`)

业务开发中最常使用的工具，提供：

- `chatwootAPI` 实例调用官方 SDK 方法
- `isReady` 和 `hasError` 状态判断
- 封装好的 `setUser` 和 `setCustomAttributes` 便捷函数

5. UI 组件 (`components/ChatwootWidgetOptimized.tsx`)

客户端组件，负责：

- 显示加载提示
- 提供友好的错误信息
- 渲染聊天窗口

---

5. 首次配置指南

目标

获取以下两个关键信息用于项目配置：

- **Base URL** (服务地址)
- **Website Token** (网站渠道令牌)

步骤 1：登录并创建网站渠道

1. **登录 Chatwoot 账户**
    - 官方云服务：`https://app.chatwoot.com`
    - 私有化部署：访问您的服务器地址（这个地址就是 Base URL）
2. **创建收件箱**
    - 导航至 **设置 (Settings)** > **收件箱 (Inboxes)**
    - 点击 **"添加收件箱 (Add Inbox)"**
3. **选择渠道类型**
    - 选择 **"网站 (Website)"**

步骤 2：填写网站信息

1. **填写渠道详情**
   - **Website Name**: 为网站渠道命名，如 `speedxPCB 项目官网`
   - **Website Domain**: 填写网站域名（开发环境可填 `http://localhost:3000`）
   - 点击 **"创建下一步 (Create Next)"**
2. **添加客服代理**
   - 选择至少一位客服人员处理来自网站的咨询
   - 点击 **"添加代理并完成 (Add agents and finish)"**

步骤 3：获取凭证

完成配置后，Chatwoot 会展示嵌入代码。从中提取关键信息：

```
window.chatwootSDK.run({
  websiteToken: 'a1b2c3d4e5f6g7h8i9j0', // Website Token
  baseUrl: 'https://app.chatwoot.com'       // Base URL
})
```

请妥善保存这两个值，下一步将在项目中使用。

---

6. 开发使用指南

步骤 1：配置环境变量

在项目根目录下的 `.env.local` 文件中添加：

```
NEXT_PUBLIC_CHATWOOT_BASE_URL="https://app.chatwoot.com"
NEXT_PUBLIC_CHATWOOT_WEBSITE_TOKEN="a1b2c3d4e5f6g7h8i9j0"
```

> ⚠ **重要提醒**: 修改 `.env.local` 后必须重启开发服务器才能生效！

步骤 2：全局布局配置

`ChatwootProvider` 和 `ChatwootWidgetOptimized` 已配置在 `app/layout.tsx` 中：

```
// app/layout.tsx
import { ChatwootProvider } from '@/components/ChatwootProvider';
import { ChatwootWidgetOptimized } from '@/components/ChatwootWidgetOptimized';

export default function RootLayout({ children }) {
  return (
    <html lang="zh-CN">
      <body>
```

```
          <ChatwootProvider>
            {children}
            <ChatwootWidgetOptimized />
          </ChatwootProvider>
        </body>
      </html>
    );
  }
```

步骤 3：在组件中使用

场景一：用户登录后同步信息

```
'use client';

import { useEffect } from 'react';
import { useChatwootOptimized } from '@/lib/hooks/useChatwootOptimized';
import { useUserStore } from '@/lib/stores/user-store';

export default function ChatwootUserSyncer() {
  const { isReady, setUser, setCustomAttributes } = useChatwootOptimized();
  const { user } = useUserStore();

  useEffect(() => {
    if (isReady && user) {
      // 识别用户核心身份
      setUser({
        identifier: user.id,
        name: user.display_name,
        email: user.email,
        avatar_url: user.avatar_url,
      });

      // 设置业务属性
      setCustomAttributes({
        '用户ID': user.id,
        '公司': user.company_name,
        '电话': user.phone_number,
        '最近登录IP': user.last_sign_in_ip,
      });
    }
  }, [isReady, user, setUser, setCustomAttributes]);

  return null;
}
```

场景二：主动打开聊天窗口

```
'use client';

import { useChatwootOptimized } from '@/lib/hooks/useChatwootOptimized';
import { Button } from '@/components/ui/button';
import { MessageCircle } from 'lucide-react';

export default function ContactSupportButton() {
  const { isReady, chatwootAPI } = useChatwootOptimized();

  const handleOpenChat = () => {
    if (isReady && chatwootAPI) {
      chatwootAPI.toggle('open');
    } else {
      alert('在线聊天功能暂不可用，请稍后重试。');
    }
  };

  return (
    <Button onClick={handleOpenChat} disabled={!isReady} variant="outline">
      <MessageCircle className="mr-2 h-4 w-4" />
      联系技术支持
    </Button>
  );
}
```

---

7. 高级功能与扩展

事件监听

监听 Chatwoot SDK 派发的事件来响应聊天状态变化：

```
useEffect(() => {
  if (!isReady) return;

  const onChatwootEvent = (event) => {
    console.log('Chatwoot event received:', event.detail);
  };

  window.addEventListener('chatwoot:on-message', onChatwootEvent);
  window.addEventListener('chatwoot:on-conversation-status-change',
onChatwootEvent);

  return () => {
    window.removeEventListener('chatwoot:on-message', onChatwootEvent);
    window.removeEventListener('chatwoot:on-conversation-status-change',
onChatwootEvent);
  };
}, [isReady]);
```

自定义外观

主题定制

通过 CSS 变量自定义聊天窗口外观：

```
:root {
  --chatwoot-primary-color: #1f93ff;
  --chatwoot-secondary-color: #f0f0f0;
  --chatwoot-font-family: 'Inter', sans-serif;
}
```

位置调整

```
.woot-widget-holder {
  bottom: 20px !important;
  right: 20px !important;
}
```

多语言支持

设置界面语言

```
useEffect(() => {
  if (isReady && chatwootAPI) {
    chatwootAPI.setLocale('zh-CN'); // 设置为中文
  }
}, [isReady, chatwootAPI]);
```

安全考虑

内容安全策略 (CSP)

如果您的网站使用了 CSP，需要添加以下规则：

```
<meta http-equiv="Content-Security-Policy" content="
  script-src 'self' http://www.leodennis.top:3000;
  connect-src 'self' http://www.leodennis.top:3000 ws://www.leodennis.top:3000;
  frame-src http://www.leodennis.top:3000;
">
```

数据隐私

- 避免在 `setCustomAttributes` 中传递敏感信息（密码、API 密钥等）
- 定期审查传递给 Chatwoot 的用户数据
- 确保符合 GDPR、CCPA 等数据保护法规

---

8. 故障排查与最佳实践

常见问题排查

1. 聊天窗口无法加载

检查清单:

- [ ] `.env.local` 文件中的环境变量是否正确设置？
- [ ] 修改环境变量后是否重启了开发服务器？
- [ ] 网络是否能访问 Chatwoot 服务器？
- [ ] 浏览器是否安装了广告拦截插件？

调试步骤:

1. 打开浏览器开发者工具，查看 Console 和 Network 标签页
2. 检查是否有 404 错误（无法加载 sdk.js）
3. 检查是否有 CORS 错误
4. 尝试直接访问 `[BASE_URL]/sdk.js` 确认文件可访问

2. 用户信息无法同步

可能原因:

- `setUser` 调用时机过早（SDK 未就绪）
- `identifier` 值不唯一或为空
- 网络请求失败

解决方案:

```
useEffect(() => {
  // 确保 SDK 就绪且用户信息存在
  if (isReady && user && user.id) {
    setUser({
      identifier: String(user.id), // 确保是字符串类型
      name: user.display_name || 'Unknown User',
      email: user.email || '',
    });
  }
}, [isReady, user]);
```

3. 自定义属性不显示

检查要点:

- 属性值是否为字符串类型
- 属性名是否包含特殊字符
- 是否在 `setUser` 之后调用 `setCustomAttributes`

最佳实践

性能优化

- **懒加载**: SDK 采用异步加载，不阻塞页面渲染
- **缓存策略**: 利用浏览器缓存减少重复请求
- **条件加载**: 仅在需要时加载聊天功能

代码组织

- **统一管理**: 将用户同步逻辑集中在专用组件中
- **错误边界**: 使用 React Error Boundary 包裹 Chatwoot 组件
- **类型安全**: 为 Chatwoot API 添加 TypeScript 类型定义

用户体验

- **加载状态**: 显示加载指示器，避免用户困惑
- **错误处理**: 提供友好的错误提示和重试机制
- **响应式设计**: 确保在移动设备上的良好体验

数据管理

- **适度使用**: 只传递对客服有价值的信息
- **数据验证**: 确保传递的数据格式正确
- **隐私保护**: 避免传递敏感信息

---

9. API 快速参考

环境变量

变量名	描述	示例
`NEXT_PUBLIC_CHATWOOT_BASE_URL`	Chatwoot 服务器地址	`https://app.chatwoot.com`
`NEXT_PUBLIC_CHATWOOT_WEBSITE_TOKEN`	网站渠道令牌	`a1b2c3d4e5f6g7h8i9j0`

Hook API

`useChatwootOptimized()`

```
const {
  isReady,        // boolean: SDK 是否就绪
  hasError,       // boolean: 是否有加载错误
  chatwootAPI,    // object: Chatwoot API 实例
```

```
  setUser,        // function: 设置用户信息
  setCustomAttributes, // function: 设置自定义属性
} = useChatwootOptimized();
```

核心方法

setUser(userInfo)

```
setUser({
  identifier: string,    // 必需：用户唯一标识
  name?: string,         // 可选：用户姓名
  email?: string,        // 可选：用户邮箱
  avatar_url?: string,   // 可选：头像 URL
});
```

setCustomAttributes(attributes)

```
setCustomAttributes({
  '属性名1': '属性值1',
  '属性名2': '属性值2',
  // ... 更多属性
});
```

chatwootAPI.toggle(action)

```
chatwootAPI.toggle('open');   // 打开聊天窗口
chatwootAPI.toggle('close');  // 关闭聊天窗口
chatwootAPI.toggle();         // 切换聊天窗口状态
```

chatwootAPI.setLocale(locale)

```
chatwootAPI.setLocale('zh-CN'); // 设置为中文
chatwootAPI.setLocale('en');    // 设置为英文
```

事件监听

事件名	描述	触发时机
chatwoot:on-message	收到新消息	客服或用户发送消息时
chatwoot:on-conversation-status-change	对话状态变化	对话被打开、关闭或解决时

事件名	描述	触发时机
`chatwoot:on-unread-message-count-changed`	未读消息数变化	未读消息数量发生变化时

常用 CSS 选择器

选择器	描述
`.woot-widget-holder`	聊天窗口容器
`.woot-widget-bubble`	聊天气泡按钮
`.woot-widget-holder iframe`	聊天窗口 iframe

相关文档

- Chatwoot 官方文档
- Chatwoot JavaScript SDK
- 项目故障排查指南

贡献与支持

如果您在使用过程中遇到问题或有改进建议，请：

1. 查阅本文档的故障排查部分
2. 检查 Chatwoot 官方文档
3. 在项目中创建 Issue 或联系开发团队

最后更新：2024年12月