这份讲座由 IBM 开放平台的 David Jones-Gilardi 主讲，主要演示了如何使用开源工具 **LangFlow** 快速构建 AI 智能体（Agent）并将其集成到应用程序中。

以下是讲座的核心内容总结：

1. 核心工具：LangFlow

- 简介：LangFlow 是一个用于构建 RAG（检索增强生成）和 Agentic（代理）工作流的开源可视化 IDE。它拥有超过 14.4k 的 GitHub Stars，特点是低代码/无代码的拖拽式界面。
- 模型无关性：LangFlow 不限制模型提供商。演示中展示了它可以连接 OpenAI、IBM WatsonX 以及本地模型（通过 Ollama 运行 Llama 3 或 Gwen 3）。
- **MCP 原生支持**：LangFlow 原生支持 **模型上下文协议 (MCP)**，允许开发者轻松集成外部工具（如 GitHub 只读工具）。

2. 演示一：从零构建基础 Agent

- 构建过程：David 展示了通过拖拽 Input（输入）、Agent（智能体）和 Output（输出）组件，在几秒钟内搭建一个基本的聊天机器人。
- 工具调用能力：
  - 为了展示 Agent 如何获取训练数据之外的信息，他添加了一个 **URL 抓取工具**。
  - 案例：当用户询问“200 美元兑换印度卢比是多少”时，模型通过工具获取实时汇率并给出准确答案，而不是依靠过时的训练数据。
- **游乐场 (Playground)**：在未连接外部应用前，开发者可以在 LangFlow 内置的 Playground 中直接与 Agent 对话进行测试，查看中间步骤（如工具调用过程）。

3. 演示二：集成到 "Pixel Ticker" 应用

- 应用架构：演示了一个名为 "Pixel Ticker" 的复古风格 Next.js 应用程序。
- **智能路由 (Smart Router)**：
  - App 不直接连接单一 Agent，而是先通过一个“智能路由器”（由本地小模型如 Ollama/LM Studio 驱动）。
  - 工作原理：路由器分析用户意图，将请求分发给专门的 Agent（例如：关于股票的问题发给“金融 Agent”，关于太空的问题发给“通用/天文学 Agent”）。

    - **结构化输出**：不同的 Agent 返回特定格式的 JSON 数据，前端 App 根据数据类型渲染不同的 UI 组件（如图表或文本）。

4. 关键技术细节与最佳实践

- **Session ID (会话 ID)**：在 API 调用中传递 `session_id` 至关重要，它确保 Agent 能够记住对话历史（Context），从而实现连续对话。
- **API 集成**：LangFlow 提供现成的代码片段（Python, JS/TS, cURL）。开发者只需设置 API Key 即可通过 API 调用构建好的 Flow。
- **开发效率**：David 提到该应用的原型约 1 小时完成，完善用了约 3 天。他还使用了 IBM 的 AI 编码助手 "IBM Bob" 来辅助编写前端代码。

5. 总结

LangFlow 提供了一种灵活、可视化的方式来构建复杂的 AI 工作流。它支持本地和云端模型，允许通过 MCP 集成各种工具，并能轻松地通过 API 将智能体能力嵌入到现代 Web 应用中。