**问题1.2**

**摘要**

这篇论文研究了“混搭”（Mashup）应用程序开发的方法。混搭是一种将多个服务聚合在一起的方法，每个服务都有自己的功能，最终形成一个具有新功能的服务。虽然这种方法为数据和服务消费者提供了新的、更广泛的机会，但开发过程仍然要求用户不仅要会编写代码，还要懂得如何使用各种Web API。

论文的目标是分析混搭工具的优缺点。作者识别了通用混搭应用的行为和特征，并根据这些关键方面对混搭工具进行了分析。研究认为，这种分析很重要，因为它可以推动未来在这个新兴领域的贡献。在这个领域中，许多研究和应用领域（如数据库、用户与机器的交互等）可以相互结合。

**介绍**

探讨了Web 2.0中Mashup框架的发展及其工具的分析。以下是论文的主要内容：

**背景**

* **Web 2.0目标**：简化创建、使用、描述、分享和重用网络资源。
* **技术发展**：包括博客、社交网络等，推动了Web API的普及（如Google Maps、Amazon、YouTube）。
* **数据源**：RSS和ATOM等数据源的兴起，提供了丰富的数据供应用程序使用。

**Mashup框架**

* **定义**：Mashup是一种将多个服务聚合在一起以创建新服务的应用开发方法。
* **应用**：通过重用和组合现有服务（如搜索引擎、数据等）创建新应用。
* **示例**：包括地图Mashup（在地图上绘制数据）、多媒体内容Mashup（使用YouTube、Flickr等）、电子商务服务Mashup（如Amazon、Yahoo购物）以及数据源Mashup（如RSS、ATOM订阅）。

**挑战与工具**

* **挑战**：开发Mashup应用需要编程知识和对各种Web API的熟悉。
* **工具发展**：为了解决这一问题，一些工具被开发出来，以支持编程知识有限的用户进行Mashup应用开发。

**研究目标**

* **分析工具的优缺点**：研究Mashup工具的行为和特性，重点在于数据集成。
* **关键问题**：工具如何处理数据、进行哪些处理、Mashup的结果是什么等。

**Mashup应用程序的组成和实现**

通过一个示例应用程序来说明。该应用程序允许用户从新闻提供商（如CNN国际）选择新闻，并在地图上显示新闻的位置。为了实现这个应用程序，需要使用三种类型的组件服务：（i）地图服务，（ii）新闻服务，（iii）RSS-GeoRSS转换服务。

**关键组件：**

1. **数据调解层（Data Mediation Level）**：

* 处理来自多个异构数据源的数据访问和集成。
* 需要解决结构和语义上的多样性问题，例如不同名称或数据类型的相同元素。
* 涉及数据的转换、过滤、格式转换和组合，以实现数据流的集成。

1. **过程调节层（Process Mediation Level）**：

* 定义应用程序之间的编排和交互。
* 关注服务的行为聚合和远程数据服务的连接。
* 需要使用特定语言（如Bite和Swashup）来描述交互和组合模型。

1. **展示层（Presentation Level）**：

* 提供用户界面以与用户交互。
* 包括使用HTML、Ajax和JavaScript等技术来显示结果。
* 需要处理客户端和服务器端Mashup的整合问题，尤其是跨域问题。

**实现挑战：**

* **数据整合**：处理不同数据格式（如XML、JSON）和语义差异。
* **服务组合**：需要适应交互式和异步进程的建模。
* **用户界面**：需要解决跨域访问问题，可能需要服务器端解决方案。

**工具分析**

Mashup 工具如何帮助用户获取和整合异构数据，以创建 Mashup 应用程序。以下是关键点的总结：

**背景与挑战**

* 创建一个 News Mashup 应用需要编程技能，包括调用 API、实现屏幕抓取技术，以及理解数据结构以进行数据中介。
* Mashup 工具提供了一些功能来解决这些挑战，帮助用户搜索数据源、进行数据转换和创建数据流。

**分析维度**

论文从八个维度分析了 Mashup 工具如何解决数据中介问题：

1. **APIs**

* APIs 是应用程序提供的接口，允许用户与其他程序、应用或网站进行数据交互。
* 通过 REST/SOAP 等标准协议，用户可以从提供者（如 Microsoft、Google）获取内容。
* APIs 还可以访问非 URL 可寻址的资源，如企业数据。

1. **内部数据模型**

* Mashup 工具使用内部数据模型来统一表示数据。
* XML-based 模型：常用于 Web 数据，适合处理半结构化数据。
* Object-based 模型：使用面向对象的方式，需要程序员定义对象结构。

1. **数据映射**

* 数据映射用于指定外部数据源与内部数据模型之间的对应关系。
* 可以是手动、半自动或自动的映射。当前工具大多支持手动或半自动映射。

1. **数据流操作符**

* 这些操作符用于对数据结构或内容进行操作，如重构数据模式、过滤、合并数据集等。

1. **数据源刷新**

* 数据源的刷新策略包括拉取（pull）和推送（push）策略。
* 当前 Mashup 工具主要使用拉取策略，用户可以设置刷新频率。

1. **输出**

* Mashup 的输出可以是可视化的，也可以导出为其他数据格式以供进一步处理。

1. **可扩展性**

* 工具的可扩展性体现在支持用户定义功能的能力上，可以是嵌入式或外部调用的形式。

1. **共享**

* 共享功能涉及 Mashup 应用的隐私和安全。
* 涉及共享对象、共享方式（如只读、读写）和共享对象的用户群体。

**举例分析**

1. **Damia (IBM)**

* **功能**: 用于聚合和转换数据流，支持REST和SOAP服务，以及本地文件（如Excel、CSV、XML）。
* **数据处理**: 将所有数据源转换为XML格式的序列元组。
* **共享策略**: 支持完全共享或部分共享Mashup输出，只有只读权限。

1. **Yahoo Pipes**

* **功能**: 通过模块组合来聚合和操作数据流，支持REST服务。
* **数据处理**: 内部使用RSS数据模型，提供自动和半自动的数据映射。
* **共享策略**: 支持完全共享或部分共享，代码和输出可以共享。

1. **Popfly (Microsoft)**

* **功能**: 通过连接块来创建Mashup，支持REST和SOAP服务。
* **数据处理**: 使用对象为基础的数据模型，手动指定数据映射。
* **共享策略**: 支持完全共享或不共享。

1. **Google Mashup Editor (GME)**

* **功能**: 使用HTML、JavaScript等技术构建Mashup，支持RSS和Atom。
* **数据处理**: 使用基于Atom的Google Data feed，无需显式映射。
* **共享策略**: 支持完全共享、部分共享和不共享，允许指定访问权限。

1. **Exhibit**

* **功能**: 用于创建包含动态可视化的网页，支持多种数据格式。
* **数据处理**: 基于JSON的图模型，自动生成数据映射。
* **共享策略**: 不提供共享支持，用户需自行上传网页。

1. **Apatar**

* **功能**: 集成桌面数据与网络数据，支持多种数据库和Web API。
* **数据处理**: 使用对象为基础的数据模型，用户定义数据转换。
* **共享策略**: 仅支持完全共享或不共享。

1. **MashMaker**

* **功能**: 直接在网页上编辑和操作数据，使用RDF模式。
* **数据处理**: 提供用户熟悉的操作（如复制和粘贴），通过RDF描述提取数据。
* **共享策略**: 支持与所有用户或不共享，无法管理特定用户组。

**Mashup 工具的优缺点以及可能的改进方向**

1. **数据访问**：Mashup 工具主要用于处理网络数据，这既是优势也是劣势。优势在于可以访问和管理仅在网上可用的数据（如 RSS 源），劣势在于无法访问和使用用户桌面上的数据，这限制了数据清理和操作的能力。
2. **协议支持**：这些工具支持 REST 和 SOAP 协议，这是因为这些协议的流行和实用性。
3. **数据模型**：大多数工具内部使用 XML 数据模型，这是因为网络数据通常以 XML 格式暴露。另一个常见的数据模型是基于对象的，虽然更灵活，但需要更多编程。
4. **数据操作**：工具提供的数据操作功能有限，通常根据工具的主要目标设计。对于非专业用户来说，这些操作器不易使用，且工具的表达能力有限，无法实现复杂查询。
5. **数据刷新策略**：没有工具实现数据推送策略，所有工具都使用拉取策略，这主要是为了防止服务器过载。拉取间隔设置为全局策略，限制了高刷新频率的数据处理。
6. **扩展性和重用**：尽管 Mashup 工具具有扩展性，可以开发新操作器和数据模式，但大多数工具不支持重用已创建的 Mashup。这限制了不同工具之间的协作。
7. **数据共享和安全性**：目前对数据共享和安全性的关注较少。安全问题需要考虑，因为通信问题可能导致过多请求，造成服务器过载。
8. **服务器端应用**：所有工具都是服务器端应用，意味着 Mashup 和数据都托管在工具提供者的服务器上，用户无法完全控制应用的可靠性。
9. **性能和可扩展性**：没有工具提供关于性能和可扩展性的分析信息，无法评估系统处理增长数据和用户请求的能力。
10. **用户友好性**：尽管工具面向“非专家”用户，但通常需要编程知识。没有工具能让用户在无需编程或低编程投入的情况下构建 Mashup。