zkInfo 系统架构图与数据流图

📋 目录

---

1 完整调用链路图

```
graph TB
    subgraph "客户端层"
        C1[MCP Inspector]
        C2[mcp-router-v3]
    end

    subgraph "zkInfo 接口层"
        I1["GET /sse/endpoint"]
        I2["POST /mcp/serviceName/message"]
    end

    subgraph "zkInfo 服务层"
        S1[SseController]
        S2[McpMessageController]
        S3[McpSessionManager]
        S4[EndpointResolver]
        S5[VirtualProjectRegistrationService]
        S6[McpExecutorService]
    end

    subgraph "数据存储层"
        D1[Redis
会话存储]
        D2[Nacos Config
工具配置]
        D3[MySQL
元数据]
    end

    subgraph "外部服务"
        E1[Dubbo Provider
实际服务]
    end

    C1 --> I1
    C2 --> I1
    C1 --> I2
    C2 --> I2

    I1 --> S1
    I2 --> S2

    S1 --> S3
    S2 --> S3
    S2 --> S4
    S4 --> S5
    S2 --> S6

    S3 --> D1
    S5 --> D2
    S4 --> D3
    S6 --> E1

    style C1 fill:#e1f5ff
    style C2 fill:#e1f5ff
    style I1 fill:#fff4e1
    style I2 fill:#fff4e1
    style S1 fill:#e8f5e9
    style S2 fill:#e8f5e9
    style S3 fill:#e8f5e9
    style S4 fill:#e8f5e9
    style S5 fill:#e8f5e9
    style S6 fill:#e8f5e9
    style D1 fill:#fff3e0
    style D2 fill:#fff3e0
    style D3 fill:#fff3e0
    style E1 fill:#fce4ec
```

---

2 完整数据流转图

```
flowchart LR
    subgraph "数据源"
        ZK[ZooKeeper
Provider 节点]
        ZKMeta[ZooKeeper
元数据路径]
    end

    subgraph "zkInfo 处理"
        Load[数据加载
ZooKeeperBootstrapService]
        Filter[白名单过滤
InterfaceWhitelistService]
        Parse[参数解析
DubboServiceMethodService]
```

```
        Gen[工具生成
EnhancedMcpToolGenerator]
        Reg[服务注册
NacosMcpRegistrationService]
    end

    subgraph "数据存储"
        DB[(MySQL
元数据持久化)]
        NacosConfig[Nacos Config
工具配置]
        NacosReg[Nacos Registry
服务注册]
    end

    subgraph "外部调用"
        Client[MCP 客户端]
        Executor[Dubbo 泛化调用]
        Provider[Dubbo Provider]
    end

    ZK -->|1. 读取 Provider| Load
    ZKMeta -->|2. 读取元数据| Parse

    Load -->|3. 应用过滤| Filter
    Filter -->|4. 持久化| DB

    DB -->|5. 查询参数| Parse
    Parse -->|6. 生成工具| Gen

    Gen -->|7. 发布配置| NacosConfig
    Gen -->|8. 注册服务| NacosReg

    Client -->|9. 建立连接| Reg
    Client -->|10. 调用工具| Executor
    Executor -->|11. 泛化调用| Provider
    Provider -->|12. 返回结果| Executor
    Executor -->|13. 返回结果| Client

    style ZK fill:#e3f2fd
    style ZKMeta fill:#e3f2fd
    style Load fill:#fff3e0
    style Filter fill:#fff3e0
    style Parse fill:#f3e5f5
    style Gen fill:#f3e5f5
    style Reg fill:#e8f5e9
    style DB fill:#fff9c4
    style NacosConfig fill:#fff9c4
    style NacosReg fill:#fff9c4
    style Client fill:#e1f5ff
    style Executor fill:#fce4ec
    style Provider fill:#fce4ec
```

3 MCP 消息处理模块

```
flowchart TD
    Start[接收 MCP 消息] --> Parse[解析 JSON-RPC 请求]
    Parse --> Method{判断方法类型}

    Method -->|initialize| Init[处理初始化]
    Method -->|tools/list| ToolsList[处理工具列表]
    Method -->|tools/call| ToolCall[处理工具调用]
    Method -->|resources/list| ResourcesList[处理资源列表]
    Method -->|prompts/list| PromptsList[处理提示列表]
    Method -->|notifications/initialized| Notify[处理通知]

    Init --> InitResp[返回初始化响应]

    ToolsList --> Resolve[解析端点]
    Resolve --> GetTools[获取工具列表]
    GetTools --> ToolsResp[返回工具列表]

    ToolCall --> ParseTool[解析工具名称]
    ParseTool --> ExtractArgs[提取参数]
    ExtractArgs --> Execute[执行工具调用]
    Execute --> Dubbo[泛化调用 Dubbo]
    Dubbo --> ToolResp[返回调用结果]

    ResourcesList --> ResourcesResp[返回资源列表]
    PromptsList --> PromptsResp[返回提示列表]
    Notify --> NotifyResp[返回 202 Accepted]

    InitResp --> SSE[通过 SSE 发送响应]
    ToolsResp --> SSE
    ToolResp --> SSE
    ResourcesResp --> SSE
    PromptsResp --> SSE
    NotifyResp --> HTTP[直接 HTTP 响应]

    style Parse fill:#e3f2fd
    style Method fill:#fff3e0
    style Execute fill:#e8f5e9
    style SSE fill:#f3e5f5
```

---

4 McpSessionManager 核心节点

```
graph TB
    subgraph "会话注册层"
        A1[registerSseEmitter]
        A2[创建 SessionMeta]
        A3[存储到 Redis]
    end
    subgraph "会话获取层"
        B1[getSseEmitter]
        B2[从 Redis 获取]
        B3[验证会话有效性]
        B4[返回 SseEmitter]
    end

    subgraph "会话管理层"
        C1[getEndpointForSession]
        C2[getServiceName]
        C3[更新会话信息]
        C4[移除会话]
    end

    subgraph "Redis 存储层"
        D1[SessionRedisRepository]
        D2[存储会话元数据]
        D3[设置 TTL]
        D4[查询会话信息]
    end

    subgraph "清理任务层"
        E1[SessionCleanupService]
        E2[定时扫描]
        E3[查找过期会话]
        E4[清理过期会话]
    end

    A1 --> A2
    A2 --> A3
    A3 --> D1
    D1 --> D2
    D2 --> D3
    B1 --> B2
    B2 --> D4
    D4 --> B3
    B3 --> B4
    C1 --> D4
    C2 --> D4
    C3 --> D2
    C4 --> D1
    E1 --> E2
    E2 --> E3
    E3 --> D4
    E4 --> D1

    style A1 fill:#e3f2fd
    style B1 fill:#fff3e0
    style C1 fill:#e8f5e9
    style D1 fill:#f3e5f5
    style E1 fill:#fce4ec
```

5 McpMessageController 核心节点

```
graph TB
    subgraph "请求接收层"
        A1[handleMessage]
        A2[handleMessageWithPath]
        A3[解析 JSON-RPC 请求]
    end

    subgraph "端点解析层"
        B1[EndpointResolver]
        B2[从参数获取]
        B3[从会话获取]
        B4[从缓存获取]
    end

    subgraph "方法路由层"
        C1[handleInitialize]
        C2[handleToolsList]
        C3[handleToolCall]
        C4[handleResourcesList]
        C5[handlePromptsList]
    end

    subgraph "工具调用层"
        D1[McpExecutorService]
        D2[解析工具名称]
        D3[提取参数]
        D4[泛化调用 Dubbo]
    end

    subgraph "响应发送层"
        E1[通过 SSE 发送]
        E2[直接 HTTP 响应]
        E3[构建 JSON-RPC 响应]
    end

    A1 --> A3
    A2 --> A3
    A3 --> B1
    B1 --> B2
    B2 --> B3
    B3 --> B4
    B4 --> C1
    B4 --> C2
    B4 --> C3
    B4 --> C4
    B4 --> C5
    C3 --> D1
    D1 --> D2
    D2 --> D3
    D3 --> D4
    C1 --> E1
    C2 --> E1
    C3 --> E1
    C4 --> E1
    C5 --> E1
    E1 --> E3
    E2 --> E3

    style A1 fill:#e3f2fd
    style B1 fill:#fff3e0
    style C1 fill:#e8f5e9
    style D1 fill:#f3e5f5
    style E1 fill:#fce4ec
```

6 McpMessageController 数据流

```
flowchart TD
    Request[接收请求] --> Parse[解析 JSON-RPC]
    Parse --> Resolve[解析端点]
    Resolve --> Route{路由方法}

    Route -->|initialize| Init[处理初始化]
    Route -->|tools/list| ToolsList[处理工具列表]
    Route -->|tools/call| ToolCall[处理工具调用]
    Route -->|resources/list| ResourcesList[处理资源列表]
    Route -->|prompts/list| PromptsList[处理提示列表]

    ToolCall --> Exec[执行工具调用]
    Exec --> Dubbo[泛化调用 Dubbo]
    Dubbo --> Result[获取结果]

    Init --> Response[构建响应]
    ToolsList --> Response
    Result --> Response
    ResourcesList --> Response
    PromptsList --> Response

    Response --> SSE{是否有 SSE?}
    SSE -->|是| SendSSE[通过 SSE 发送]
    SSE -->|否| SendHTTP[直接 HTTP 响应]

    style Request fill:#e3f2fd
    style Parse fill:#fff3e0
    style Route fill:#e8f5e9
    style Exec fill:#f3e5f5
    style Response fill:#fce4ec
```

文档版本: 1.0.0
最后更新: 2025-01-XX
维护者: zkInfo Team

注意: 本文档使用 Mermaid 图表语法，可以在支持 Mermaid 的 Markdown 查看器中渲染（如 GitHub、GitLab、VS Code 等）。