MCP'ye Giriş

Model Context Protocol (MCP) nedir, hangi N×M entegrasyon problemini çözer, host /client/server rolleri, ve bir server'ın sunabileceği tools/resources/prompts primitive'leri.

Orta 18 dk

Son güncelleme: 5 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/introduction-to-mcp

---

MCP'ye Giriş

"Tools and Function Calling" mekanizmayı anlatmıştı: model bir fonksiyon call'ı ister, ve bir uygulama onu çalıştırır. O dersin açık bıraktığı şey, bir uygulamanın bir modele TUTARLI bir tool, veri ve prompt kümesini gerçekten nasıl sunduğuydu -- özellikle tool'lar ve veri birçok farklı yerde (bir dosya sistemi, bir şirket veritabanı, üçüncü taraf bir API) yaşadığında ve uygulama birden fazla AI modeli ya da sağlayıcıyla konuştuğunda. **Model Context Protocol (MCP)**, bu soruya açık ve standartlaştırılmış bir cevaptır.

MCP Nedir?

MCP, bir AI uygulamasının dış tool'lara, veri kaynaklarına ve prompt şablonlarına nasıl bağlanacağını standartlaştıran açık bir protokoldür. Her uygulamanın tool'ları tarif etme ve çağırma yolunu kendi başına icat etmesi yerine ("Tools and Function Calling"ın uygulamaya özel bıraktığı gibi), MCP bunları sunmak için tek, tutarlı bir biçim, neyin mevcut olduğunu keşfetmek için tek, tutarlı bir yol, ve onu çağırmak için tek, tutarlı bir yol tanımlar -- karşı tarafta hangi AI modelinin olduğundan ya da tool'un hangi programlama dilinde yazıldığından bağımsız olarak. Önceki dersteki tool-calling loop'un yerini almaz; onun *sunucu tarafını* standartlaştırır, böylece aynı tool implementasyonu herhangi bir MCP-uyumlu uygulama tarafından yeniden kullanılabilir.

Neden Var?

Paylaşılan bir standart olmadan, *M* farklı AI uygulamasını *N* farklı tool ve veri kaynağına bağlamak, çok sayıda ayrı, özel entegrasyon gerektirir -- bir sağlayıcının tool-calling biçimiyle kurulmuş bir uygulama, bir başkasıyla çalışması için yeniden yazılmalıdır, ve her yeni veri kaynağı, onu kullanmak isteyen her uygulama için ısmarlama bağlantı kodu gerektirir. MCP'nin amacı, her AI uygulaması ile her tool/veri kaynağı arasında ayrı özel entegrasyonlar yazma ihtiyacını azaltarak entegrasyon mimarisini standartlaştırmaktır: bir

tool ya da veri kaynağı, bir MCP server olarak BİR KEZ implemente edilir, ve herhangi bir MCP-uyumlu uygulama, değiştirilmeden ona bağlanabilir. Bunu kısaca "$M \times N \rightarrow M + N$" diye özetlemek yaygındır -- ama bunun kesin bir matematiksel entegrasyon-maliyeti formülü değil, fikri hızlıca akılda tutmaya yarayan basitleştirilmiş bir zihinsel model olduğunu unutmayın. Bu, hem "LLM Yetenekleri ve Sınırlamaları" hem de "Tools and Function Calling" in birlikte kurduğu şeye doğrudan, pratik bir cevaptır: modellerin güncel bilgiye ulaşmanın ve gerçek eylemler almanın güvenilir bir yoluna ihtiyacı vardır, ve bu erişim her uygulama ve her tool için yeniden inşa edilmek yerine yeniden kullanılabilir olmalıdır.

Host, Client ve Server

MCP, her kurulumda aynı kalan üç rol tanımlar:

- **Host** -- kişinin gerçekten etkileşime girdiği AI uygulaması (bir chat uygulaması, bir IDE, özel bir agent). Host, LLM ile konuşmaktan ve MCP'nin ne zaman kullanılacağına karar vermekten sorumludur.
- **Client** -- host'un içinde yaşar ve tam olarak tek bir server'a birebir bağlantı sürdürür. Üç farklı server'a ulaşması gereken bir host, içeride her bağlantı için bir tane olmak üzere üç client çalıştırır.
- **Server** -- tool'ları, veriyi ya da prompt şablonlarını, bağlanan herhangi bir client'a sunan ayrı bir programdır. Bir server, hangi host'la konuştuğunu ya da host'un hangi LLM'i kullandığını bilmez ya da umursamaz -- yalnızca protokolü konuşur.

Bu ayrım, "bir kez yaz, her yerde kullan" özelliğinin çalışmasını sağlayan şeydir: bir şirketin dahili ticket sistemini sunmak için kurulmuş bir server, sonunda onu çağıracak belirli chat uygulaması, IDE ya da LLM hakkında hiçbir şey bilmesine gerek duymaz.

Bir Server Ne Sunar: Tool'lar, Resource'lar ve Prompt'lar

Bir MCP server üç tür primitive sunabilir, ama belirli bir server yalnızca birini sunmakta serbesttir:

- **Tools** -- tam olarak "Tools and Function Calling"da anlatılan anlamda çalıştırılabilir fonksiyonlar: bir isim, bir açıklama, ve bir parameter schema, bir eylem ya da hesaplama gerçekleştirmek ve bir sonuç döndürmek için çağrılır.
- **Resources** -- host uygulamasının çekebileceği, belirli bir dosyanın içeriği ya da bir veritabanı kaydı gibi, bir URI ile tanımlanan okunabilir veri. Bir resource çalıştırılmaz, OKUNUR -- bir eylem gerçekleştirmek yerine bilgi sağlar.
- **Prompts** -- bir server'ın sunabileceği yeniden kullanılabilir prompt şablonları, böylece yaygın ya da karmaşık prompting kalıplarının, onları kullanmak isteyen her host uygulaması içinde tekrar tekrar yazılması gerekmez.

Bu kursun "TypeScript ile MCP Sunucusu Oluşturma" dersindeki uygulamalı örneği özellikle tool'lara odaklanıyor, çünkü bunlar "Tools and Function Calling"e en doğrudan şekilde bağlanıyor -- ama gerçek bir server, üç primitive türünü birleştirmekte serbesttir.

MCP ve Tool-Calling Döngüsü

MCP, önceki dersteki tool-calling loop'u değiştirmez; onun iki adımını standartlaştırır. O döngünün 1. adımı -- uygulamanın hangi tool'ların mevcut olduğunu öğrenmesi -- MCP terimleriyle, client'ın bağlı bir server'dan tool'larını listelemesini istemesi ve server'ın her tool'un adını, açıklamasını ve parameter schema'sını, hepsi tek, paylaşılan bir biçimi izleyerek döndürmesi haline gelir. 4. adım -- uygulamanın gerçekten bir fonksiyon çalıştırması -- client'ın aynı bağlantı üzerinden server'a yapılandırılmış bir call göndermesi ve server'ın gerçek kodu çalıştırıp sonucu döndürmesi haline gelir. Arada olan her şey (modelin bir tool'u çağırıp çağırmayacağına ve nasıl çağıracağına karar vermesi) değişmez ve hâlâ tam olarak "Tool-Calling Döngüsü"nde anlatıldığı gibi gerçekleşir -- MCP, tool'ların NASIL keşfedildiğini ve çağrıldığını standartlaştırır, modelin onları nasıl kullanmaya karar verdiğini değil.

Best Practices

- Bir MCP server'ı, bire bir entegrasyon değil, yeniden kullanılabilir altyapı olarak düşünün -- onu belirli bir host uygulamasının ihtiyaçlarına göre değil, altta yatan veri kaynağının ya da sistemin gerçekten neler yapabildiğine göre tasarlayın (bkz. "Neden Var?").
- Hata ayıklarken host/client/server ayrımını aklınızda tutun: bir client her zaman tek bir server'la konuşur, bu yüzden birden fazla server'a bağlı bir host, aslında paralel olarak birden fazla bağımsız client bağlantısı çalıştırıyordur (bkz. "Host, Client ve Server").
- İş için doğru primitive'i seçin -- modelin yalnızca okuması gereken veri, aynı veriyi döndüren bir tool'a değil, bir resource'a aittir, çünkü bir tool çalıştırılabilir bir eylemi ima eder (bkz. "Bir Server Ne Sunar: Tool'lar, Resource'lar ve Prompt'lar").

Yaygın Hatalar

- **MCP'nin belirli bir AI modeli ya da yeni bir tool-calling biçimi olduğunu varsaymak.** "MCP Nedir?" bölümünün açıkladığı gibi, MCP, mevcut tool-calling kavramlarının ("Tools and Function Calling"dan) nasıl keşfedildiğini ve çağrıldığını standartlaştıran bir protokoldür -- tool use'un altında yatan fikrin yerini almaz ya da onunla yarışmaz.
- **"Server"ı istemci-yüzlü bir web sunucusu anlamına geliyor olarak görmek.** MCP terimleriyle, "Host, Client ve Server"ın tanımladığı gibi, bir server, tool'ları, resource'ları ya da prompt'ları bir client'a sunan HERHANGİ BİR programdır -- küçük, yerel bir process kadar kolay, uzak bir web servisi de olabilir.
- **Her data kaynağı için bir server yerine, her host uygulaması için bir server inşa etmek.** "Neden Var?" MCP'nin tüm değerinin, bir server'ın host'lar arasında yeniden kullanılabilir olmasından geldiğini gösterdi -- tek bir host'un tuhaflıklarına göre tasarlamak, bu yeniden kullanılabilirliği çöpe atar.

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- MCP (Model Context Protocol), AI uygulamalarının dış tool'lara, veriye ve prompt şablonlarına nasıl bağlanacağını standartlaştıran açık bir protokoldür.
- Var oluş amacı, her uygulamanın her tool'a özel olarak bağlanma ihtiyacını azaltacak şekilde entegrasyon mimarisini standartlaştırmaktır -- kısaca "$M \times N \rightarrow M + N$" diye özetlenen (kesin bir formül değil, basitleştirilmiş bir zihinsel model) bir fikir: bir veri kaynağını bir kez server olarak implemente edin, herhangi bir uyumlu host'tan kullanın.
- Üç rol sabit kalır: **host** (AI uygulaması), **client** (host'un içinde, her server bağlantısı için bir tane), ve **server** (belirli bir host'tan bağımsız olarak işlevsellik sunar).
- Bir server **tools** (eylemler, "Tools and Function Calling"daki gibi), **resources** (okunabilir veri), ve **prompts** (yeniden kullanılabilir şablonlar) sunabilir.
- MCP, modelin bir tool'u nasıl kullanmaya karar verdiğini değiştirmeden, tool-calling loop'un iki uygulama-tarafı adımını -- tool keşfi ve çağrısını -- standartlaştırır.

Cheat Sheet

- MCP = AI uygulamaları için tool/veri/prompt erişimini standartlaştıran açık protokol.
- Çözülen problem: M x N özel entegrasyon -> M + N (bir kez yaz, her yerde kullan).
- Host = AI uygulaması. Client = host içinde, her server için bir tane. Server = işlevselliği sunar.
- Primitive'ler: tools (eylemler), resources (okunabilir veri), prompts (yeniden kullanılabilir şablonlar).

Terimler Sözlüğü

- **Model Context Protocol (MCP):** AI uygulamalarının dış tool'lara, veri kaynaklarına ve prompt şablonlarına nasıl bağlanacağını standartlaştıran açık bir protokol.
- **Host:** bir kişinin etkileşime girdiği AI uygulaması; LLM ile konuşmaktan ve MCP'nin ne zaman kullanılacağına karar vermekten sorumludur.
- **Client:** bir host'un içinde, tek bir MCP server'a birebir bağlantı sürdüren bileşen.
- **Server:** tool'ları, resource'ları ya da prompt'ları bağlanan herhangi bir client'a sunan bağımsız bir program.
- **Resource (MCP):** bir server'ın sunduğu, bir URI ile tanımlanan, eylem gerçekleştirmek yerine bilgi sağlayan okunabilir veri.
- **Prompt (MCP):** bir server'ın bağlanan client'lara sunabileceği yeniden kullanılabilir bir prompt şablonu.