Cours 6

Introduction au Model Context Protocol

Anthropic Academy — Resume detaille

Yoann Boulch | 01/04/2026

1. Qu'est-ce que le Model Context Protocol

■ MCP est un protocole ouvert qui standardise les connexions entre les applications IA et les services externes.

Le Model Context Protocol (MCP) est une initiative Anthropic qui fournit une interface standardisée pour connecter les modèles d'IA à des services externes, des bases de données et des outils. Plutôt que d'écrire du code personnalisé pour chaque intégration, MCP propose un protocole unifié.

Caractéristiques clés:

- Protocole ouvert et standardisé
- Réutilisable entre applications
- Type-safe grâce aux schémas JSON
- Support pour Python, TypeScript et autres langages

2. Pourquoi utiliser MCP

✓ MCP élimine la nécessité de code personnalisé répétitif et crée un écosystème réutilisable.

Avant MCP, intégrer un nouveau service externe nécessitait du code spécifique à chaque application et à chaque service. MCP change cela en fournissant une interface standardisée.

Avantages principaux:

- Réutilisabilité: un serveur MCP peut servir plusieurs applications
- Maintenabilité: une seule implémentation à maintenir
- Écosystème: partager et découvrir des serveurs MCP
- Sécurité: permissions granulaires et auditable

3. Architecture MCP

Flux d'architecture MCP

L'architecture MCP suit un modèle client-serveur où l'application cliente communique avec un serveur MCP qui interagit avec des services externes.

Composants:

- **Application:** le client utilisant IA (ChatGPT, Claude, etc.)
- **Client MCP:** interface vers le serveur MCP
- **Serveur MCP:** implémente les outils, ressources, et prompts
- **Service Externe:** base de données, API, système de fichiers

4. Les trois primitives MCP

■ Outils (contrôlés par le modèle), Ressources (contrôlées par l'app), Prompts (contrôlés par l'utilisateur)

Primitive	Qui contrôle	Description
Tools	Modèle IA	Fonctions que le modèle peut invoquer
Resources	Application	Données que l'app expose
Prompts	Utilisateur	Instructions pré-formatées

5. Construction de serveurs MCP avec Python

■ Utilisez FastMCP SDK avec des décorateurs pour créer facilement des serveurs MCP

La SDK FastMCP pour Python rend la création de serveurs MCP simple avec des décorateurs et des type hints.

Exemple basique:

```
from mcp.server.fastmcp import FastMCP mcp = FastMCP("mon_serveur") @mcp.tool() def mon_outil(param: str) -> str: """Description de l'outil""" return f"Résultat: {param}" if __name__ == "__main__": mcp.run()
```

6. Définir des outils MCP

■ Les outils utilisent le décorateur @mcp.tool() avec schéma JSON auto-généré

Le décorateur @mcp.tool() transforme une fonction Python en un outil MCP. Les type hints génèrent automatiquement le schéma JSON.

Exemple: gestion de documents:

```
@mcp.tool() def lire_document(chemin: str) -> str: """Lit le contenu d'un document""" with open(chemin, 'r') as f: return f.read() @mcp.tool() def editer_document(chemin: str, contenu: str) -> bool: """Modifie un document""" with open(chemin, 'w') as f: f.write(contenu) return True
```

7. Définir des ressources

■ Les ressources exposent des données statiques avec URIs et types MIME

Les ressources permettent à l'application de fournir des données au serveur MCP. Elles peuvent être statiques ou templées.

Types de ressources:

- Ressources directes: URIs statiques comme 'file:///config.json'
- Ressources templées: URIs paramétrées comme 'file:///{id}/data'
- MIME types: application/json, text/plain, etc.

8. Définir des prompts

■ Les prompts sont des instructions pré-formatées exposées par le serveur

```
@mcp.prompt() def analyser_code(langage: str) -> str: """Prompt pour analyser du code""" return
f"""Analysez le code {langage} fourni. Vérifiez: erreurs, performance, sécurité."""
```

9. Inspecteur serveur MCP

■ L'inspecteur basé navigateur facilite le test et le débogage des serveurs MCP

L'inspecteur MCP fournit une interface web pour tester les outils, ressources et prompts d'un serveur MCP.

Fonctionnalités:

- Tester les outils avec des paramètres
- Inspecter les schémas générés
- Déboguer les erreurs en temps réel
- Visualiser les ressources exposées

10. Implémenter des clients MCP

■ Les clients MCP connectent les applications aux serveurs MCP

Un client MCP gère la communication avec le serveur, appelle les outils et accède aux ressources.

Responsabilités du client:

- Découvrir les outils disponibles
- Envoyer les requêtes d'exécution
- Gérer les réponses asynchrones
- Accéder aux ressources du serveur

11. Quand utiliser chaque primitive

■ Choisissez la primitive selon qui contrôle l'action

Situation	Primitive	Exemple
Le modèle doit agir	Tools	Lire un fichier
L'app partage des données	Resources	Configuration
L'utilisateur donne instructions	Prompts	Template de requête

12. Résumé et bonnes pratiques

■ MCP crée un écosystème standardisé pour les intégrations IA

Le Model Context Protocol simplifie l'intégration des services externes avec les applications IA. En utilisant ses trois primitives de manière cohérente, vous créerez des systèmes maintenables et réutilisables.