

図5 「発注点を下回っている商品は？」がデータベースに届くまで

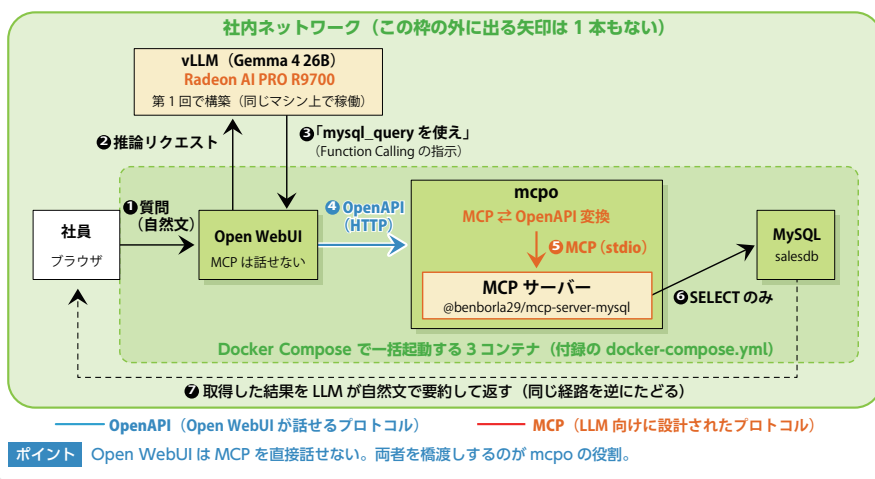


表3 カスタムモデル「在庫管理」

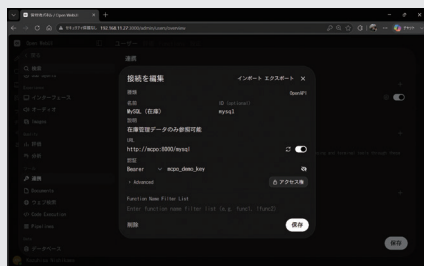
項目	内容
モデル名	在庫管理
モデルID	inventory-assistant
ベースモデル	接続しているLLM
Tools	先ほど登録したMySQL（在庫） Tool Serverを有効化

完全版はこちら▶



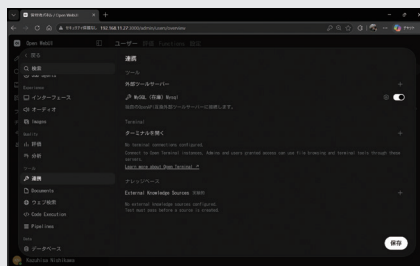
bashの抜粋

```
curl -X POST -H "Authorization: Bearer (APIキー)" -H "Content-Type: application/json" \
-d '{"sql": "SELECT product_name, stock_qty, reorder_point FROM ... WHERE stock_qty < reorder_point"}' \
http://localhost:8000/mysql/mysql_query
# => [{"product_name": "ラミネーター", "stock_qty": 14, "reorder_point": 23}, ...]
```



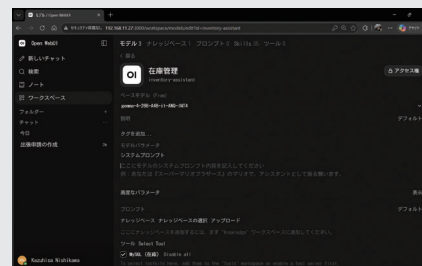
OpenAPIツールサーバーの登録フォーム

管理者パネル > 設定 > 連携 画面。OpenAPIツールサーバーの追加フォーム



Tool Server「MySQL(在庫)」の登録完了

Tool Serverとして登録が完了し、一覧に「MySQL(在庫)」が表示された状態



「在庫管理」モデルにMySQLツールを有効化

新しいモデル作成画面。「在庫管理」にMySQL（在庫） Toolを有効化しているところ

APIとはそのままでは会話できない。両者の間を変換するのが「mcpo」の役割だ。mcpoはOpen WebUIチームが開発するプロキシで、stdio（標準入出力）で動くMCPサーバーを、通常のHTTP・OpenAPI形式のAPIサーバーに変換する。Open WebUIはMCPを直接は話さず、OpenAPI形式の「Tool Server」を読み込む設計のため、両者を橋渡しする役割を担う。試しにcurlで直接たたいてみると、在庫が発注点を下回っている商品がJSONで返ってくる（上の**bashの抜粋**参照。完全版は**Web**に掲載）。この時点でMCP経由のDBアクセスは完成。あとはOpen WebUIから使えるようにするだけだ。

●準備4: Tool Serverとして登録し、カスタムモデルを作る

Open WebUIにTool Serverとして登録するには、Open WebUIの管理者パネル（個人アカウント設定とは別画面）から、mcpoが公開しているOpenAPIエンドポイントを登録する。

このURLの登録・保存はサーバー側で処理されるが、画面

上の「接続を確認」ボタンはブラウザから直接このURLを叩く。ブラウザはmcpoというホスト名をDocker内部のDNSとして解決できないため、ここだけエラーになりやすい（保存自体は通る）。気になる場合は、作業しているPCのhostsファイルに「127.0.0.1 mcpo」を追加しておく（要管理者権限）、ブラウザからも同じ名前で解決できるようになる。

なおカスタムモデルの作成は、前回のコンテンツのRAG編でQ&A専用モデルを作ったのと同じ要領で、DB連携済みのカスタムモデルを作成する。ひと通り準備ができれば実際に試してみよう（下記**テスト例**参照）。

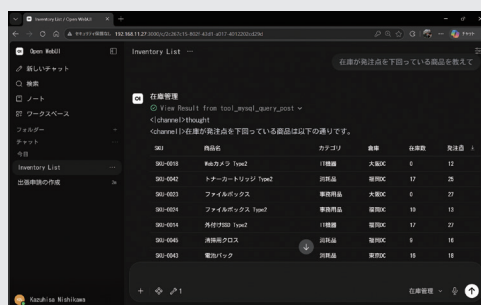
一連の仕組みは、ドキュメントの検索ではなく、その場でSQLを組み立てて実行している点がRAG・Skillとの決定的な違いだ。データが更新されれば、次に聞いたときには最新の値が返る。「在庫が発注点を下回っている商品を教えて」と質問してから回答が出るまでの時間は約7秒だった。初回はテーブル構造を確認する問い合わせも挟んでおり、スキーマを把握したあとの会話ならさらに短縮できる。

テスト例



質問「在庫一覧」

質問への回答として、全商品一覧が表示された



質問「在庫が発注点を下回っている商品を教えて」

質問への回答。ツール呼び出しの様子と結果一覧として、発注点割れの商品がSQLで抽出され表示された

Tool Server（ツールサーバー）とは：Open WebUIの用語で、OpenAPI形式で公開されたAPIサーバーを「モデルが呼び出せる道具箱」として登録する仕組み。一度登録すれば、その道具箱に入っているツール（今回ならmysql_query）を、任意のモデルやカスタムモデルから有効・無効を切り替えて使い回せる。