LLMProvider

LLMProvider

واجهة واحدة، 43 مزودًا — مع قواطع دوائر، وإعادة المحاولة، ومراقبة الصحة مدمجة.

الملخص

الموحدة `LLMProvider` عامة وقابلة لإعادة الاستخدام تحدد واجهة Go هو وحدة LLMProvider بالإضافة إلى أنماط المرونة الإنتاجية المحيطة بها — قاطع الدائرة، ومراقبة الصحة، وإعادة المحاولة مع التراجع التدريجي، والتحميل الكسول — وتوفر 43 تنفيذًا ملموسًا للمزودين خلف عقد واحد، مع محول عام متوافق مع OpenAI ونموذج اكتشاف نماذج صادق دون أي حلول احتياطية مشفرة مسبقًا.

الوصف المختصر

الواحدة `LLMProvider` قابلة لإعادة الاستخدام تعرض واجهة Go وحدة (`Complete`، `CompleteStream`، `HealthCheck`، `GetCapabilities`، `ValidateConfig`) بالإضافة إلى بدائيات تحمل الأخطاء — قاطع الدائرة، ومراقب الصحة، وإعادة المحاولة مع تراجع تدريجي مضاف إليه تذبذب زمني، والتحميل الكسول — عبر 43 محولًا للمزودين ومحول عام متوافق مع OpenAI. آمنة للخيوط.

الوصف الطويل

LLMProvider هي طبقة التجريد التي يحتاجها كل خدمة مستهلكة لـ LLM لكنها نادرًا ما تُبنى بشكل جيد — تلك البنية التحتية غير الجذابة التي تفصل بين العرض التوضيحي والنظام الذي ينجو من مواجهة حركة المرور الحقيقية. فهي تحدد واجهة واحدة محركة للقدرات — `Complete`، `CompleteStream`، `HealthCheck`، `GetCapabilities`، `ValidateConfig` — بحيث يستهدف كود التطبيق عقدًا واحدًا بالضبط بغض النظر عن أي من الـ 43 خلفية يجيب على الاستدعاء، ثم توفر التعزيزات التشغيلية التي تحول مكالمات المزودين الهشة إلى شيء يمكنك تشغيله في الإنتاج دون حبس أنفاسك. قاطع الدائرة ثلاثي الحالات (مغلق → مفتوح → نصف مفتوح) يغلف أي مزود حيث يُحسب الدفق الفارغ بشكل صحيح كفشل — بحيث ،بما في ذلك قناته البثية — بشكل شفاف يمكن لخلفية واحدة سيئة السلوك أن تفتح القاطع وتمنع انهيار الخدمة بأكملها؛ ويدير المركزي كل القواطع دفعة واحدة. مراقب الصحة القابل للتكوين يتابع `CircuitBreakerManager` المزودين باستمرار عبر حالات الصحة/التدهور/الخلل/المجهول بناءً على فحوصات العتبات والفترات، بحيث يُلاحظ التدهور بدلاً من اكتشافه عند حدوث انقطاع. منطق إعادة المحاولة يضيف التراجع الأسي مع التذبذب الزمني في الأعلى، متخذًا قرارات محركة للحالة — إعادة المحاولة للأخطاء التي تستحق أو سياق xxدون إهدار الدورات على أخطاء 4 ،(أعطال الشبكة العابرة ،429xx، 5) إعادة المحاولة

ملغى — مع تأخير محدد بحيث لا يمكن لعاصفة التراجع أن تخرج عن السيطرة. ونمط التحميل الكسول يؤجل بناء كل مزود حتى استخدامه الفعلي الأول — وهو خيار تصميم متعمد يجعل تسجيل جميع المزودين الـ 43 مجانيًا بشكل أساسي.

ينفذ الواجهة OpenAI متوافق مع `generic` توفر الوحدة 43 حزمة مزود ملموسة بالإضافة إلى محول مع SSE وبث Bearer، مصادقة — `/v1/chat/completions` نقطة نهاية أي الكاملة ضد بحيث يصبح المزود الذي ليس لديه حزمة مخصصة مواطنًا من الدرجة — `[DONE]` معالجة صحيحة لـ الخاص به. تُحل بيانات الاعتماد في مكان واحد بالضبط URL الأولى بمجرد توجيه المحول إلى مما يغلق فئة الأخطاء الكاملة التي، (`ApiKey_<Provider>` باستخدام اصطلاح صارم، `apikeys`) تقول "المفتاح المشفر يجتاز مجموعة الاختبارات، لكن المفتاح الحقيقي لم يُربط أبدًا، ويتعطل المنتج في الإنتاج" من مصدرها. اكتشاف النماذج متعمد وصارم في صدقه: فهو يستعلم واجهات برمجة التطبيقات للمزودين الفعليين خلف ذاكرة تخزين مؤقتة ذات مدة حياة، و— وفقًا للحوكمة — تم حذف طبقة الحل أي LLMProvider الاحتياطي المشفرة مسبقًا بالكامل. عندما يفشل الاكتشاف الفعلي، لا تُرجع بدلاً من كتالوج قديم، بحيث لا يُعطى المتصل مطلقًا معرّف نموذج يبدو صالحًا ثم لا يمكن شيء استدعاؤه. كل جزء من هذا البناء آمن للخيوط للاستخدام المتزامن.

المحتوى

لماذا أنشأناها

في بيئة الإنتاج — حيث تفرض مقدمو الخدمة قيودًا على LLM تفشل المكالمات البسيطة لـ المعدلات، أو تتدهور الخدمة، أو تتوقف تمامًا، وقد يتسبب مزود واحد سيئ في إسقاط الخدمة بأكملها. LLMProvider تتغير فهارس النماذج، وتقدم القوائم المشفرة مسبقًا معرفات لم تعد صالحة. تجمع بين الواجهة ومرونة الأداء والاكتشاف الصادق بحيث يرث كل مستهلك القدرة على التحمل والموثوقية تلقائيًا دون جهد إضافي.

لماذا تُعد نقلة نوعية

إلى خطوة واحدة فقط — إما تنفيذ واجهة واحدة، أو توجيه المحول "LLM تختزل عملية "دمج مزود العام إلى نقطة نهاية — ثم تغلف هذا المزود تلقائيًا وشفافًا في قواطع الدوائر ومراقبة الصحة وإعادة المحاولة مع تأخير متذبذب. تتوقف المرونة عن كونها شيئًا يعيد كل فريق اختراعه (بشكل سيئ، وتحت ضغط المواعيد النهائية، بعد أول انقطاع) لتصبح السلوك الافتراضي للمكتبة عبر جميع مزودي الخدمة البالغ عددهم ٤٣ مزودًا. تُكتب هندسة الموثوقية مرة واحدة، وتُختبر بدقة، ويرثها الجميع مجانًا بمجرد استيرادها.

ما الجديد في الابتكار

- **واجهة واحدة تدرك القدرات** — دمج الاستكمال والبث ومراقبة الصحة والتحقق من القدرات والتكوين في عقد واحد تلتزم به جميع الخلفيات بنفس الطريقة.
- **التغليف الشفاف لقواطع الدوائر — بما في ذلك البث.** `CompleteStream` يحمي القاطع قناة — وليس مجرد طلب/استجابة فحسب، ويعامل البث الفارغ على أنه الفشل الحقيقي الذي يمثله مع إشعارات المستمعين للآمنة من التعطل وخارج القفل.
- **٤٣ حزمة مزودات + محول عام متوافق مع OpenAI** — تبقى الحزم المخصصة خفيفة، وأي `/v1/chat/completions` محول يعمل على الفور بمجرد توجيه بائع غير مدرج يتحدث إليه.

- **سلطة اعتماد واحدة (apikeys)** — مكان واحد فقط يقرأ متغيرات البيئة `ApiKey_<Provider>`، والمنتج "المعطل" بدلًا من مجرد التحذير منه. مما يلغي بشكل هيكلي عدم التطابق بين "الاختبارات الناجحة والمنتج".
- **اكتشاف صادق للنماذج (دون احتياطي مشفر)** — واجهات برمجة التطبيقات الحية للمزودين ولا تقدم أبدًا فهرسًا قديمًا أو `nil`، خلف ذاكرة تخزين مؤقتة ذات مدة صلاحية؛ عند الفشل، تُرجع مُختلقًا يمنح معرفات غير قابلة للاستدعاء.
- **التهيئة الكسولة باستخدام `sync.Once`** — يُؤجل البناء إلى أول استخدام، لذا فإن تسجيل جميع المزودات البالغ عددها ٤٣ مزودًا لا يكلف شيئًا تقريبًا حتى يتم استدعاء أحدها فعليًا.
- **مكدس تحدي متعدد اللغات ومقاوم للخداع** — أداة تشغيل حقيقية تختبر سلوك قواطع الدوائر والصحة وإعادة المحاولة عبر خمس لغات، محكومة باختبار الطفرات المزدوج (يجب أن يخرج الكود غير المتحور بصفر؛ ويجب أن يفرض الطفر المحقون الخروج برقم ٩٩)، لذا فإن المجموعة الناجحة تعني سلوكًا يعمل بشكل مؤكد.

أكبر التحديات التقنية وكيف تغلبنا عليها

- **فشل المزودات المتتالي.** يجب ألا يسحب مزود واحد غير مستقر خدمة بأكملها معه. تم حل هذه المشكلة باستخدام قاطع دوائر ثلاثي الحالات (مغلق → مفتوح → نصف مفتوح) يغلف أي مزود بشكل شفاف، ويفتح عند الفشل المستمر، ويتحقق من التعافي في الحالة نصف المفتوحة وبثه `CircuitBreakerManager`، ويتم تنسيقه مركزيًا بواسطة.
- **الأخطاء العابرة وحدود المعدلات.** — تم حلها باستخدام تأخير أسي حساس للحالة مع تذبذب بحيث — `min(InitialDelay·Multiplier^(n-1), MaxDelay) ± jitter` تنتشر محاولات إعادة المحاولة بدلًا من التزامن في قطيع مدوٍ. يعيد المحاولة بالضبط لما يجب إعادة المحاولة له (٤٢٩، ٥٠٠، ٥٠٢، ٥٠٣، ٥٠٤، وأخطاء الشبكة)، ويرفض إهدار المحاولات على سياق ملغى أو أي خطأ من فئة ٤xx آخر.
- **التوسع مع العديد من المزودات المسجلة وغير المستخدمة.** مع تسجيل ٤٣ مزودًا ولكن استخدام عدد قليل منها فقط في أي خدمة معينة، فإن البناء المتسرع سيكون هدرًا خالصًا. تم حل هذه المشكلة بالتهيئة الكسولة المحروسة بواسطة `sync.Once`، بحيث لا تدفع المزودات التي تستدعيها فعليًا سوى تكلفة إعدادها.
- **منح معرفات نماذج غير صالحة.** تم حل هذه المشكلة بحذف طبقة الاحتياطي المكتشفة مسبقًا وعدم إرجاع أي شيء عند فشل الاكتشاف المباشر — بالإضافة (CONST-036 وفقًا لـ) بالكامل إلى نسخة دفاعية عند الإرجاع بحيث لا يتمكن المتصل من تعديل ذاكرة التخزين المؤقت أو التنافس مع قارئ آخر. تُفرض الصدق هيكليًا، وليس بالاتفاق.
- **البث وصحة التزامن.** يتمثل نمط الفشل الخفي في حدوث تعطل بين قفل القاطع واستدعاءات المستمعين. تم حل هذه المشكلة بالتقاط لقطات من المستمعين وإخطارهم خارج القفل ضمن مهلة زمنية مدتها ٥ ثوانٍ، وفك القفل قبل الإخطار عند إعادة الضبط — مع بناء كل مكون للاستخدام المتزامن واختباره بواسطة مجموعة `race-`.

المحتوى

مجموعة الأدوات التقنية

- **Go (1.25.3)** — تم اختياره لقدرته المتميزة على التوازي، والثنائيات الثابتة، ومكتبته القياسية القوية؛ حيث يحمل الوحدة، والواجهة، وكل أدوات المرونة، وجميع المحولات البالغ عددها ٤٣.
- **`net/http` (مكتبة قياسية)** — بروتوكول HTTP خالٍ من التبعيات عن قصد: يدعم عملاء واستدعاءات الاكتشاف المباشرة، فلا توجد OpenAI، المزودين لكل مزود، والمحول المتوافق مع طبقة نقل خارجية تحتاج إلى تدقيق أو تصحيح.

- **logrus** — تسجيل منظم ومدرك للمستويات حيث يحتاج المشغلون إلى وضوح الرؤية: داخل تحولات حالة قاطع الدائرة ومسار الاكتشاف.
- **testify** — يدير مجموعة الاختبارات، والأهم من ذلك، تثبيت الفروع المتحولة الذي يجعل نجاح التشغيل ذا معنى حقيقي.
- **yaml.v3** — يحلل حزم التدويل والإعدادات بتنسيق يبقى قابلاً للتحرير البشري.
- **`digital.vasic.models`** — الأنواع المشتركة `LLMRequest` / `LLMResponse` / `ProviderCapabilities`، محفوظة في مكان واحد بحيث يتحدث كل محول بنفس المفردات (تبعية تشغيل موثقة).
- **حزم داخلية** — `circuit`، `health`، `retry`، `apikeys`، `discovery`، `providers/` (٤٣ + `generic`)، و`i18n` بائعًا: سطح المرونة والتكامل مقسم إلى وحدات صغيرة قابلة للاختبار بشكل مستقل بدلاً من كونه كتلة واحدة ضخمة.
- **`.env` + `~/api_keys.sh`** (اتفاقية **`ApiKey_<Provider>`**) — مصدر واحد لا لبس فيه للحقيقة المتعلقة بالمصادقة، بحيث يتم توصيل المفاتيح بنفس الطريقة في الاختبارات وفي الإنتاج.
- العمود — **مشغل التحدي + (`-race -p 1`) للاختبار المتزامن** **Makefile** **مجموعة أوامر** الفقري لمكافحة الخداع: يكتشف كاشف السباق صحة التوازي، بينما يضرب مشغل التحدي السلوك الحقيقي عبر الفوضى، وهجمات الحرمان من الخدمة، والتوسع، والضغط، واكتشاف الخدمات المباشرة، وسيناريوهات عدم التعليق.

ملاحظات الحالة والشفافية

- **الحالة: تجريبية.** وحدة قابلة لإعادة الاستخدام ومستقلة؛ مستودع GitHub عام.
- **الرخصة: لم تُحدد بعد.** يشير `doc.go` غير متسقة — بينما يوجد ملف MIT إلى رخصة LICENSE بنمط رخصة Apache-2.0 — تحقق قبل النشر.
- LLMsVerifier هو المصدر الرئيسي الموثوق به لكاتالوج النماذج القياسي. يبدو ملف البيان `helix-deps.yaml` (يعلن `deps: []` قديمًا بينما تذكر الوثائق وجود تبعية لـ `digital.vasic.models`) ؛ "المستوى الثاني" (models.dev) في الاكتشاف مخطط له كعنصر نائب، وليس نشطًا بعد.

مستوى الأولوية: للبنية التحتية — وحدة قابلة لإعادة الاستخدام LLM رئيسي (مجموعة Helix-ومستقلة). يأتي في المرتبة التالية بعد HelixTrack.