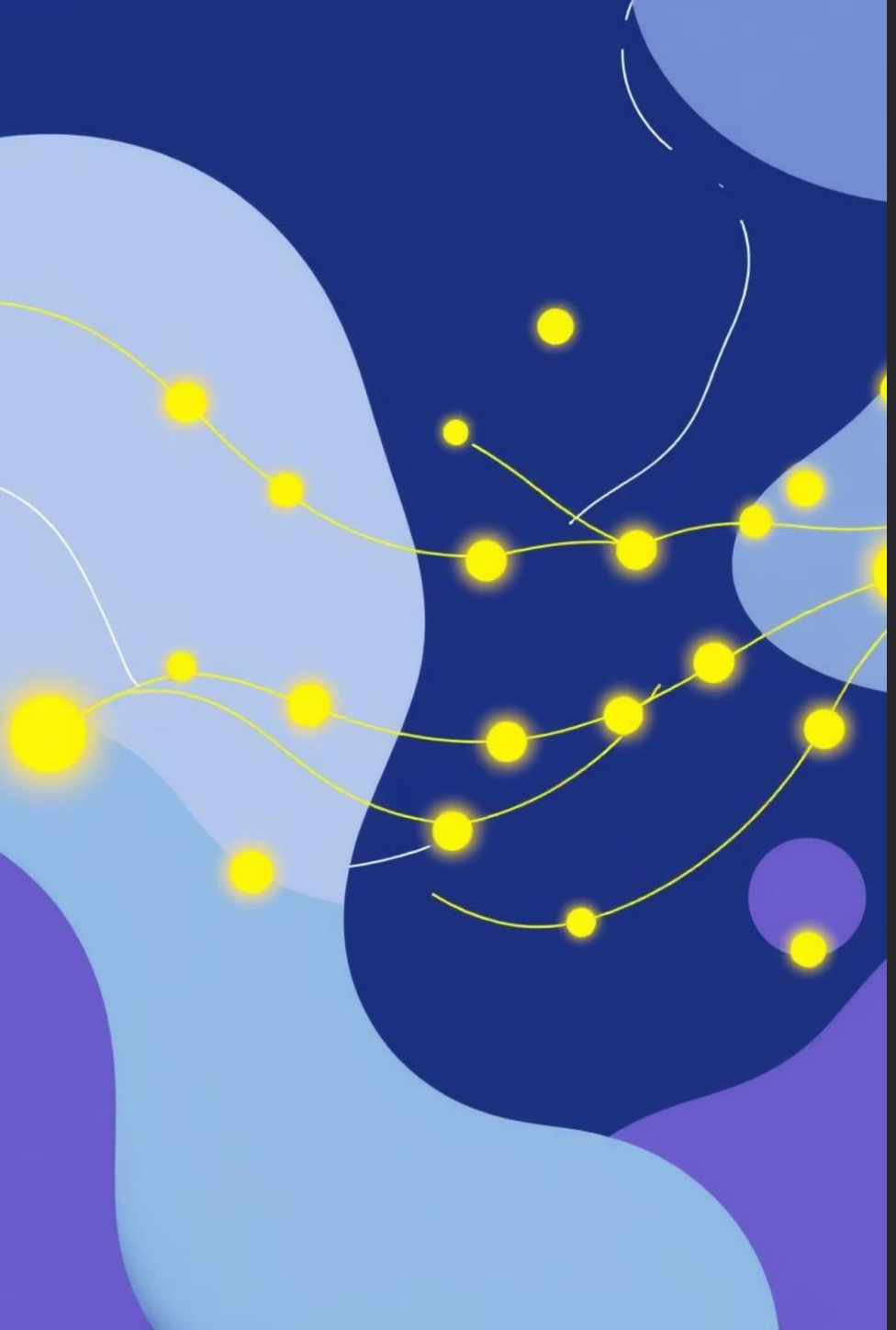
lab04

مقدم العرض

م /محمود اغا

الاستاذ م /مالك المصنف

mahmoodaghe@gmail.com



مقارنة بين أنظمة قواعد البيانات المدعومة في Django

نستكشف اليوم كيف يتعامل Django، الإطار الشهير لتطوير الويب، مع قواعد البيانات المختلفة، وكيف يمكن للمطورين الاستفادة من هذه الإمكانات لبناء تطبيقات قوية ومرنة.

أنظمة قواعد البيانات المدعومة رسميًا في) Djangoحتى (2025

MariaDB

تفرع من MySQL، يوفر أداءً ممتازًا وتوافقًا كبيرًا مع MySQL، مع تحسينات في الأمان والميزات.

PostgreSQL

خيار قوي ومفضل للمشاريع الكبيرة والمعقدة، يتميز بالموثوقية والمرونة ودعم الميزات المتقدمة.

Oracle

تستخدم في بيئات الشركات الكبيرة، وتوفر قدرات متقدمة لإدارة البيانات والتحكم فيها.

MySQL

قاعدة بيانات شائعة ومعروفة بسهولتها وسرعتها، مناسبة للعديد من التطبيقات ومواقع الويب.

دعم إضافي

توفر مكتبات الطرف الثالث دعماً لأنظمة أخرى مثل MongoDBوMSSQL، مما يوسع خيارات المطورين.

SQLite

خيار خفيف ومناسب لتطبيقات التطوير والاختبار، لا يتطلب خادمًا منفصلاً.

لماذا تدعم Djangoهذه الأنظمة تحديدًا؟

- الشهرة والانتشار :هذه القواعد البيانات هي الأكثر استخدامًا في المشاريع الواقعية، مما يضمن توافر الدعم والموارد للمطورين.
- التوافق مع Django ORM:تتكامل هذه الأنظمة بسلاسة مع نظام التعيين العلائقي للكائنات (ORM)الخاص بـ Django، مما يبسط عملية التفاعل مع البيانات.
- الاستقرار والأداء :توفر هذه القواعد البيانات أداءً مستقرًا وتكاملاً قويًا مع إطار عمل Django، مما يضمن تطبيقات موثوقة وعالية الأداء.
- دعم الميزات المتقدمة :تدعم هذه الأنظمة ميزات مهمة مثل الاتصالات المستمرة وإدارة المعاملات، وهي ضرورية للتطبيقات المعقدة.

كيف تتعامل Djangoمع قواعد البيانات؟

التحويل عبر ORM

يستخدم Django ORM (Object-Relational Mapper)لتحويل نماذج Pythonإلى جداول قواعد بيانات، مما يتيح التفاعل مع البيانات باستخدام كائنات Pythonبدلاً من SQLمباشرة.

إدارة الاتصالات التلقائية

يدير Djangoالاتصالات بقاعدة البيانات تلقائيًا .يمكن التحكم في عمر الاتصال باستخدام خاصية CONN_MAX_AGE، مما يقلل من زمن الاستجابة.

دعم الاتصالات المستمرة

تدعم Djangoالاتصالات المستمرة لتقليل حمل إعادة الاتصال، مما يحسن الأداء بشكل كبير في التطبيقات عالية التحميل.

معالجة الأخطاء الذكية

في حال حدوث أخطاء في الاتصال، تقوم Djangoبإغلاق الاتصالات التالفة وفتح اتصالات جديدة تلقائيًا، مما يضمن استمرارية الخدمة.

أمثلة إعداد الاتصال بقواعد بيانات مختلفة في settings.py

لتكوين Djangoللاتصال بقواعد البيانات المختلفة، يجب تعديل ملف settings.pyبإضافة الإعدادات التالية في قسم DATABASES:

PostgreSQL

```
DATABASES = {    'default': {        'ENGINE': 'django.db.backends.postgresql',
'NAME': 'mydatabase',        'USER': 'mydatabaseuser',        'PASSWORD':
'mypassword',        'HOST': 'localhost',        'PORT': '',    }}
```

Oracle

```
DATABASES = {    'default': {        'ENGINE': 'django.db.backends.oracle',
'NAME': 'mydatabase',        'USER': 'mydatabaseuser',        'PASSWORD':
'mypassword',        'HOST': 'localhost',        'PORT': '',    }}
```

MariaDB / MySQL

```
DATABASES = {    'default': {        'ENGINE': 'django.db.backends.mysql',
'NAME': 'mydatabase',        'USER': 'mydatabaseuser',        'PASSWORD':
'mypassword',        'HOST': 'localhost',        'PORT': '',    }}
```

SQLite

```
DATABASES = {    'default': {        'ENGINE': 'django.db.backends.sqlite3',
'NAME': BASE_DIR / 'db.sqlite3',    }}
```

الخلاصة والخطوات التالية

لقد استعرضنا قدرة Djangoالفائقة على التعامل مع أنظمة قواعد البيانات المتعددة، مما يوفر للمطورين مرونة لا مثيل لها في بناء تطبيقاتهم.

مرونة الاتصال

Djangoيدعم مجموعة واسعة من قواعد البيانات، مما يتيح لك اختيار الأنسب لمشروعك.

قوة ORM

نظام ORMيبسط التفاعل مع قواعد البيانات ويجعل الكود أكثر قابلية للقراءة والصيانة.

الخطوات التالية

- تجربة الاتصال بقواعد بيانات مختلفة.
- التعمق في إمكانيات Django ORMالمتقدمة.
- استكشاف مكتبات الطرف الثالث لدمج قواعد بيانات إضافية.