

# أدوات aiXplain المستخدمة في المشروع

## نظرة عامة

يستخدم مشروع 6 Policy Navigator Agent أنواع من أدوات aiXplain بالإضافة إلى 4 وكلاء (Agents) تم إنشاؤهم على المنصة. إليك شرح مفصل لكل منها:

## 1 أدوات aiXplain المستخدمة

### 1. LLM (Large Language Model) - GPT-4o-mini 🤖

الموقع في الكود: demo/app\_faiss.py (السطر 101-137)

الوظيفة:

- قراءة وفهم المستندات المسترجعة من قاعدة البيانات
- توليد إجابات بلغة طبيعية بناءً على السياق
- تلخيص المعلومات من مصادر متعددة
- الإجابة على الأسئلة بطريقة ذكية ومفهومة

كيف يعمل:

```
# استدعاء النموذج من aiXplain
model = ModelFactory.get('6646261c6eb563165658bbb1') # GPT-4o-mini

# إرسال السؤال مع السياق
response = model.run(prompt)

# استخراج الإجابة
answer = response.data
```

مثال على الاستخدام:

- المدخل: سؤال المستخدم + مستندات ذات صلة
- المخرج: إجابة واضحة ومفصلة بلغة طبيعية
- التكلفة: ~\$0.000115 لكل استعلام (رخيص جداً)

## 2. Text Embedding Model

الموقع في الكود: `src/data/faiss_vector_store.py` (السطر 20-30)

الوظيفة:

- تحويل النصوص إلى أرقام (vectors) يمكن للكمبيوتر فهمها
- البحث الدلالي - إيجاد نصوص متشابهة في المعنى وليس فقط الكلمات
- فهرسة المستندات في قاعدة البيانات الشعاعية

كيف يعمل:

```
from sentence_transformers import SentenceTransformer

# تحميل نموذج التضمين
model = SentenceTransformer('all-MiniLM-L6-v2')

# تحويل النص إلى vector
embedding = model.encode("نص المستند")

# النتيجة: [0.123, -0.456, 0.789, ...] (384 رقم)
```

مثال:

- "Clean Air Act" → [0.12, -0.34, 0.56, ...]
- "Air Quality Regulations" → [0.15, -0.31, 0.58, ...]
- هذان النصان قريبان في الفضاء الشعاعي لأنهما متشابهان في المعنى

## 3. Custom Python Tools (أدوات مخصصة)

تم إنشاء 4 أدوات Python مخصصة:

## أ Document Processor Tool

الملف: src/tools/document\_processor.py

### الوظيفة:

- قراءة ملفات XML (مثل CFR - Code of Federal Regulations)
- استخراج الأقسام والعناوين من المستندات
- تنظيف البيانات وتحضيرها للفهرسة

### مثال:

```
processor = DocumentProcessor()
sections = processor.extract_cfr_sections("cfr_title40.xml")
# النتيجة: قائمة من الأقسام مع العناوين والمحتوى
```

## ب Federal Register API Tool

الملف: src/tools/federal\_register\_tool.py

### الوظيفة:

- الاتصال ب Federal Register API (سجل القوانين الفيدرالية)
- البحث عن القوانين الجديدة والتحديثات
- استرجاع معلومات عن الأوامر التنفيذية والقواعد

### مثال:

```
fr_tool = FederalRegisterTool()
results = fr_tool.search_documents("EPA air quality", limit=10)
# النتيجة: قائمة من القوانين المتعلقة بجودة الهواء
```

## ج URL Scraper Tool

الملف: src/tools/url\_scraper\_tool.py

### الوظيفة:

- استخراج المحتوى من المواقع الحكومية
- تنظيف HTML والحصول على النص فقط
- فهرسة المحتوى في قاعدة البيانات

مثال:

```
scraper = URLScraperTool()
result = scraper.scrape_url("https://www.epa.gov/clean-air-act")
# النتيجة: نص نظيف من الموقع جاهز للفهرسة
```

#### د CourtListener Tool

الملف: src/tools/courtlistener\_tool.py

الوظيفة:

- البحث في قضايا المحاكم الأمريكية
- التحقق من الطعون القانونية على القوانين
- استرجاع تفاصيل القضايا والأحكام

مثال:

```
cl_tool = CourtListenerTool()
cases = cl_tool.check_regulation_challenges("Clean Air Act", "Section 111")
# النتيجة: قائمة القضايا التي طعنت في هذا القانون
```

#### 4. Vector Database (FAISS)

الملف: src/data/faiss\_vector\_store.py

الوظيفة:

- تخزين المستندات على شكل vectors
- البحث السريع عن أقرب المستندات للسؤال
- استرجاع السياق للإجابة على الأسئلة

## كيف يعمل:

```
# إضافة مستندات
vector_store.add_documents([
    {"content": "نص المستند", "metadata": {"title": "عنوان"}}
])

# البحث
results = vector_store.search("سؤال المستخدم", top_k=3)
# النتيجة: أقرب 3 مستندات للسؤال
```

## المزايا:

- **سريع جداً** - يبحث في آلاف المستندات في ميلي ثانية
- **دقيق** - يفهم المعنى وليس فقط الكلمات
- **مستقر** - يعمل على Windows بدون مشاكل

## 5. Pipeline as Tool (خط أنابيب)

**الموقع:** تم دمجها في Agent Manager

## الوظيفة:

- **ربط عدة أدوات** في سلسلة واحدة
- **تمرير البيانات** من أداة لأخرى تلقائياً
- **تنسيق العمليات** المعقدة

## مثال:

سؤال المستخدم

- **FAISS Search** (استرجاع مستندات)
- **LLM Processing** (توليد إجابة)
- **Format Response** (تنسيق النتيجة)
- إرجاع الإجابة

## 6. Code Interpreter (مفسر الأكواد) 🖥️

الموقع: مدمج في Agent Manager

الوظيفة:

- تنفيذ أكواد Python ديناميكياً
- معالجة البيانات وتحليلها
- إجراء حسابات معقدة

مثال استخدام:

```
# حساب إحصائيات من البيانات
stats = vector_store.get_collection_stats()
# النتيجة: عدد المستندات، الحجم، إلخ
```

## 2 الوكلاء (Agents) على منصة aiXplain

تم إنشاء 4 وكلاء على منصة aiXplain:

### 1. Team Agent (الوكيل المنسق) 🎯

ID: 6905048fa1a609715ed913cc

الوظيفة:

- تنسيق العمل بين جميع الوكلاء الآخرين
- اتخاذ القرارات - أي وكيل يجب استخدامه؟
- دمج النتائج من وكلاء متعددة
- إدارة سير العمل الكامل

مثال على القرار:

سؤال: "Has Section 230 been challenged in court?"

Team Agent يقرر:

1. استخدم CourtListener Agent هذا سؤال قانوني → استخدم
2. أيضاً RAG Agent قد نحتاج سياق → استخدم
3. دمج النتائج وإرجاع إجابة شاملة

## 2. RAG Agent (وكيل استرجاع المستندات)

ID: 6905048c56dba9504302685f

### الوظيفة:

- البحث في المستندات المفهرسة
- استرجاع السياق ذي الصلة
- تصفية النتائج وترتيبها حسب الأهمية

### التعليمات (Instructions):

You are a specialized RAG agent for policy documents.  
Your role is to search and retrieve relevant sections  
from government regulations and policies.

## 3. API Agent (وكيل الـ API)

ID: 6905048d56dba95043026860

### الوظيفة:

- الاتصال بـ Federal Register API
- استرجاع القوانين الجديدة والتحديثات
- التحقق من حالة القوانين

### التعليمات:

You **are** an API integration agent.  
Use Federal Register API **to fetch** latest regulations  
**and** policy updates.

## 4. Scraper Agent (وكيل الاستخراج)

ID: 6905048ea1a609715ed913cb

### الوظيفة:

- استخراج المحتوى من المواقع الحكومية
- معالجة HTML وتنظيف البيانات
- فهرسة المحتوى الجديد

### التعليمات:

You are **a** web scraping agent.  
Extract **content** **from** government websites  
and prepare it for indexing.

## 3 كيف تعمل الأدوات معاً؟

### سيناريو كامل:

المستخدم يسأل: "What are EPA's air quality standards?"

### الخطوات:

1. Flask App يستقبل السؤال

```
@app.route('/api/query', methods=['POST'])
```



## 2. FAISS Vector Store يبحث عن مستندات ذات صلة

```
results = vector_store.search(query, top_k=3)
```

## 3. Embedding Model يحول السؤال إلى vector

```
query_embedding = model.encode(query)
```

## 4. FAISS يجد أقرب المستندات

```
distances, indices = index.search(query_embedding, k=3)
```

## 5. LLM (GPT-4o-mini) يقرأ المستندات ويولد إجابة

```
model = ModelFactory.get('6646261c6eb563165658bbb1')  
answer = model.run(prompt_with_context)
```

## 6. Flask يرجع الإجابة للمستخدم

```
return jsonify({'answer': answer})
```

---

## 4 ملخص الأدوات

الأداة	النوع	الوظيفة الرئيسية	التكلفة
GPT-4o-mini	LLM	توليد إجابات ذكية	query/\$0.0001
Embedding Model	ML Model	تحويل النصوص لـ vectors	مجاني
FAISS	Vector DB	تخزين وبحث سريع	مجاني
Document Processor	Custom Tool	معالجة XML/TXT	مجاني
Federal Register API	API Tool	استرجاع القوانين	مجاني
URL Scraper	Custom Tool	استخراج المواقع	مجاني
CourtListener	API Tool	البحث في القضايا	مجاني
Team Agent	aiXplain Agent	تنسيق الوكلاء	حسب الاستخدام
RAG Agent	aiXplain Agent	استرجاع المستندات	حسب الاستخدام
API Agent	aiXplain Agent	اتصالات API	حسب الاستخدام
Scraper Agent	aiXplain Agent	استخراج الويب	حسب الاستخدام

## 5 الفرق بين الأدوات والوكلاء

### الأدوات (Tools):

- وظيفة واحدة محددة
- لا تتخذ قرارات
- تُستدعى مباشرة
- مثال: `document_processor.extract_cfr_sections()`

### الوكلاء (Agents):







- يمكنهم اتخاذ قرارات
- يستخدمون أدوات متعددة

- يفهمون السياق
  - مثال: Team Agent يقرر أي أداة يستخدم
- 

## 6 الخلاصة

---

المشروع يستخدم:

- LLM (GPT-4o-mini) 1 
- Embedding Model (all-MiniLM-L6-v2) 1 
- Vector Database (FAISS) 1 
- Custom Python Tools 4 
- aiXplain Agents 4 
- External APIs (Federal Register, CourtListener) 2 

المجموع: 13 مكون يعملون معاً لإنشاء نظام RAG متقدم! 🚀