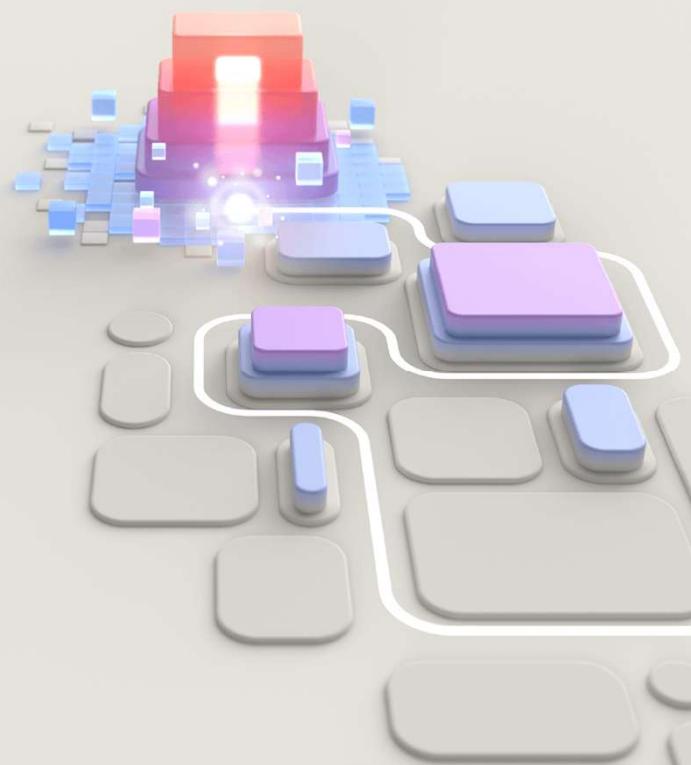




# Creating a Knowledge Mining Solution



© Copyright Microsoft Corporation. All rights reserved.

# Agenda

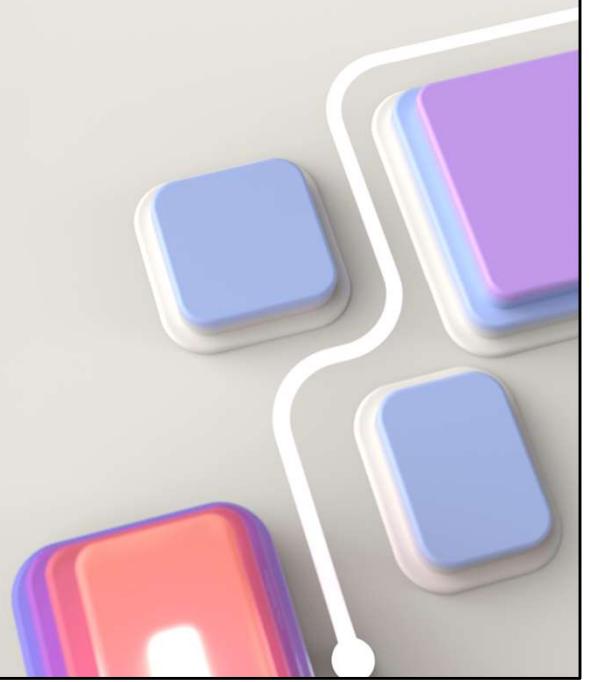
---

- Implementing an Intelligent Search Solution
- Developing Custom Skills for an Enrichment Pipeline
- Creating a Knowledge Store

© Copyright Microsoft Corporation. All rights reserved.

# Implementing an Intelligent Search Solution

© Copyright Microsoft Corporation. All rights reserved.



## Learning Objectives

After completing this module, you will be able to:

- 1 Create an Azure AI Search Solution
- 2 Implement a custom skill for Azure AI Search and integrate it into a skillset
- 3 Create a knowledge store with object, file, and table projections

© Copyright Microsoft Corporation. All rights reserved.

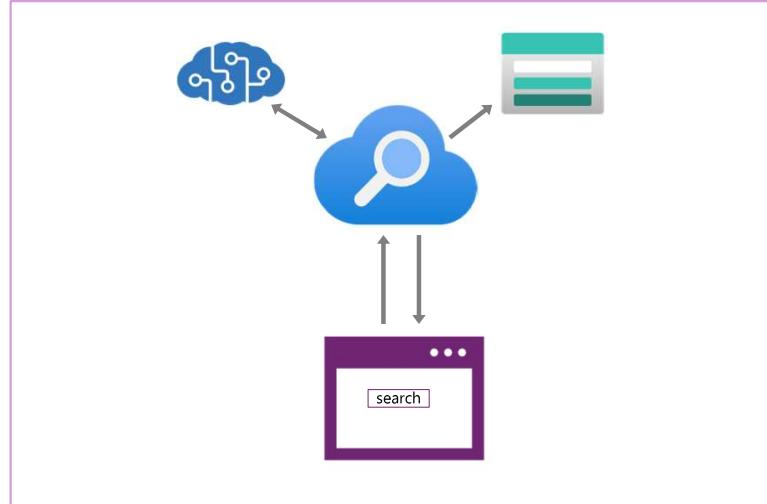
# Azure AI Search

## AI-Powered Knowledge Mining

- Index documents and data from a range of sources
- Use skills to enrich index data
- Store extracted insights in a knowledge store for analysis and integration

## Azure Resources:

- **Azure AI Search** for core indexing and querying
- **Azure AI Services** for index enrichment
- **Storage account** for knowledge store persistence



© Copyright Microsoft Corporation. All rights reserved.

Bilden handlar om **Azure AI Search** och hur den används för att göra AI-driven kunskapsutvinning.

### Huvudidé

Azure AI Search är en molnbaserad tjänst som hjälper till att indexera och söka i dokument och data från olika källor. Med hjälp av AI kan den berika indexet och lagra insikter för vidare analys.

### Vad den gör

#### 1. Indexerar data från olika källor

1. Dokument, databaser och andra strukturerade/ostrukturerade data kan bearbetas.

#### 2. Använder AI för att förbättra sökindex

1. Med hjälp av AI-tjänster (som språkförståelse och bildigenkänning) kan datan berikas innan den görs sökbar.

#### 3. Lagrar insikter för vidare analys

1. Information kan sparas i en **knowledge store** (t.ex. en lagringskonto i Azure).

### Azure-resurser i bilden

- **Azure AI Search** → Själva sökmotorn, som hanterar indexering och frågor.
- **Azure AI Services** → Ger AI-funktioner för att förstå och berika datan.
- **Storage account** → Används för att spara analyserad data.

### Illustrationen

På höger sida visas en översiktlig arkitektur:

- Ett moln (Azure AI Search) kopplas till en AI-tjänst (hjärnan), en datakälla (ikon med rader) och en sökfunktion (webbläsare).
- Pilarna visar hur data flödar mellan komponenterna.

### Sammanfattning

Azure AI Search är en kraftfull lösning för att indexera, söka och analysera data med hjälp av AI. Det gör det enklare att snabbt hitta relevant information i stora datamängder.

# Core Components of a AI Search Solution



## Data Source

The data store to be searched:

- Blob storage container
- SQL Database
- Cosmos DB

You can also push JSON documents directly into an index

## Skillset

Defines an enrichment pipeline of AI skills to enhance data during indexing:

- Built-in AI skills
- Custom skills

## Indexer

Maps data source fields and skillset outputs to index fields

- Running the indexer builds the index

## Index

Searchable collection of JSON documents containing extracted and enriched fields

© Copyright Microsoft Corporation. All rights reserved.

Självklart! Här är en pedagogisk förklaring av bilden, steg för steg:

Tänk dig att du vill bygga en **smart sökmotor** som kan söka igenom och förstå information, exempelvis dokument, databaser eller lagrade filer. För att göra detta på ett effektivt sätt behöver vi flera komponenter som samverkar.

### 1. Data Source – Var finns datan?

Det första vi behöver är själva **datan** som ska göras sökbar. Den kan komma från:

- **Blob Storage** (filer och dokument lagrade i molnet)
- **SQL-databaser** (strukturerad data)
- **Cosmos DB** (NoSQL-databas)

Alternativt kan vi skicka **JSON-dokument** direkt till sökmotorn för indexering.

### 2. Skillset – Förbättra datan med AI

Datan i sin ursprungliga form kanske inte är perfekt för sökning. Här kommer **AI-funktioner** in i bilden!

- Vi kan använda **inbyggda AI-funktioner** för att t.ex. extrahera text från bilder, analysera sentiment i text eller översätta språk.

- Vi kan även skapa **egna anpassade AI-funktioner** för att bearbeta och förbättra datan efter behov.

Tänk på detta som att ge datan "superkrafter" innan vi gör den sökbar!

### 3. Indexer – Bygga sökindexet

Nu behöver vi en **indexer**, som fungerar som en bro mellan datakällan och sökmotorn.

- Den tar information från **datakällan** och de berikade fälten från **Skillset** och kopplar ihop dem med rätt fält i indexet.

- När vi **kör indexern** byggs hela sökindexet upp automatiskt.

**Detta steg gör att datan blir strukturerad och redo att sökas!**

### 4. Index – Den sökbara databasen

Slutligen hamnar all information i ett **index**, vilket är själva kärnan i sökmotorn.

- Indexet består av **JSON-dokument** där varje dokument innehåller de extraherade och berikade fälten.

- När en användare söker efter något, letar sökmotorn snabbt igenom indexet och hittar relevant information.

Tänk på indexet som en smart katalog där all information är organiserad och lätt att hitta!

#### Sammanfattande liknelse

Om vi jämför detta med att bygga en bokkatalog i ett bibliotek:

1.**Data Source** – Böckerna som ska registreras.

2.**Skillset** – AI läser och analyserar böckerna för att skapa bättre beskrivningar.

3.**Indexer** – Registrerar böckerna och skapar ett smart system där vi kan söka efter rätt information.

4.**Index** – En färdig katalog där vi enkelt kan hitta böcker utifrån nyckelord och ämnen.

Med denna process kan vi bygga en **snabb och intelligent sökmotor** för stora datamängder! 

## Vad är ett skillset?

Ett **skillset** i Azure Cognitive Search är en samling av **skills** (färdigheter) som används för att analysera och berika data under indexeringen.

```

• {
  • "name": "my-skillset",
  • "skills": [
    • {
      • "@odata.type": "#Microsoft.Skills.Text.KeyPhraseExtractionSkill",
      • "name": "ExtractKeyPhrases",
      • "description": "Extracts key phrases from text",
      • "context": "/document/content",
      • "outputs": [
        • {
          • "name": "keyPhrases",
          • "targetName": "key_phrases"
        • }
      • ]
    • }
  • ]
}
  
```

© Copyright Microsoft Corporation. All rights reserved.

### Vad är ett Skillset?

Ett **skillset** i Azure Cognitive Search är en samling av **skills** (färdigheter) som används för att analysera och berika data under indexeringen.

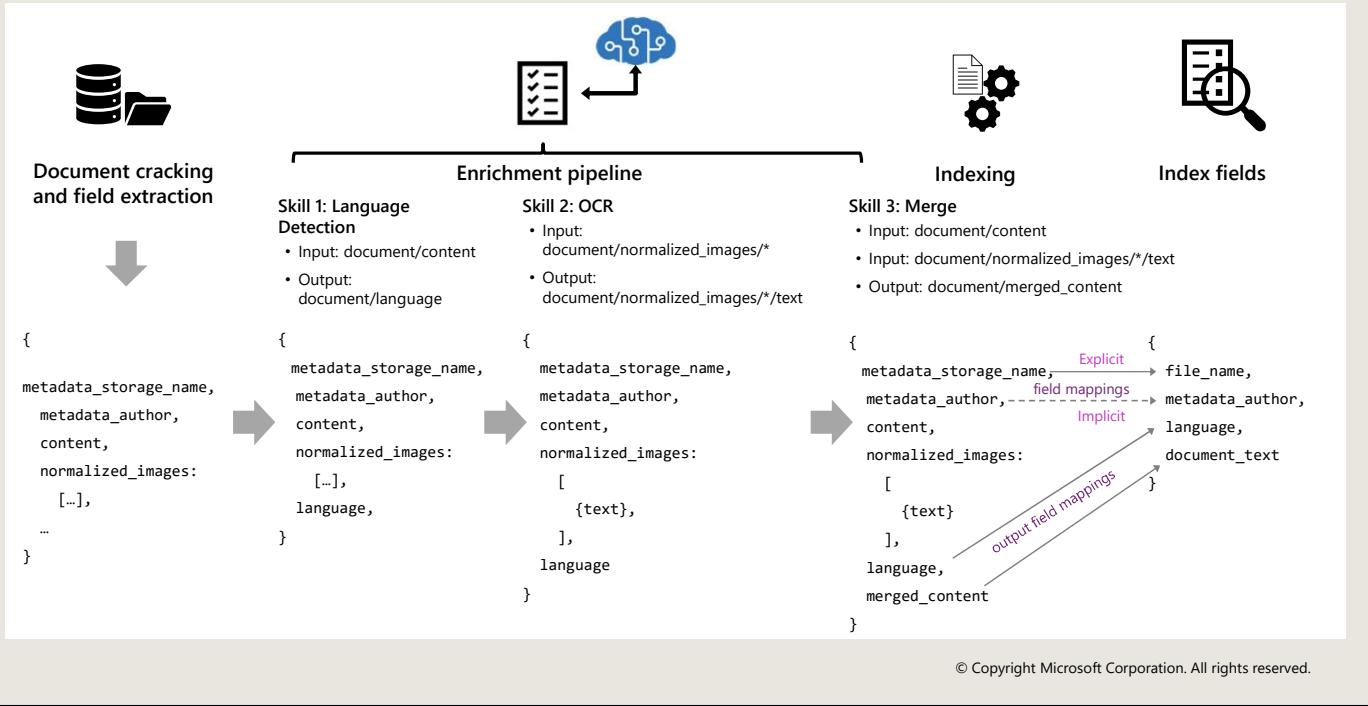
Exempel: Du kan använda ett **skillset** för att extrahera nyckelfraser, identifiera språk eller känna igen ansikten i bilder.

#### Förklaring av koden:

- **name** → Namnet på skillsetet (*my-skillset*).
- **skills** → En lista med olika analysfärdigheter (*skills*).
- **KeyPhraseExtractionSkill** → En inbyggd färdighet som extraherar nyckelord ur text.
- **context** → Anger vilket fält i dokumentet som analyseras (här: /document/content).
- **outputs** → Definierar var resultatet ska lagras (*key\_phrases*).

Denna kod går att använda i **Azure Cognitive Search** för att automatiskt identifiera viktiga nyckelord från textdata och spara dem i indexet.

# How an Enrichment Pipeline Works



Här kommer en pedagogisk genomgång av hur en **enrichment pipeline** fungerar, steg för steg!

## Vad gör en enrichment pipeline?

Den hjälper till att förbättra och berika data innan den indexeras för sökning.

1. Först **extraheras** data från dokument.

2. Sedan **förbättras** den med AI-tjänster som språkigenkänning och OCR.

3. Slutligen **indexeras** datan så att den blir sökbar.

## Steg 1: Extrahera data från dokument

Inledningsvis hämtas metadata och innehåll från ett dokument. Det kan vara en PDF, bild eller textfil.

### Exempel på extraherad data:

- { "metadata\_storage\_name": "file1.pdf", "metadata\_author": "John Doe", "content": "Some text content...", "normalized\_images": [...] } **metadata\_storage\_name** → Filens namn

- **metadata\_author** → Författare

- **content** → Dokumentets textinnehåll

- **normalized\_images** → Eventuella bilder i dokumentet

## Steg 2: Förbättra data med AI-tjänster (Enrichment Pipeline)

### Språkigenkänning (Skill 1: Language Detection)

- Tjänsten analyserar dokumentets text och identifierar vilket språk det är skrivet på.

- **Utdata:** Ett nytt fält **"language"** läggs till.

### Efter språkigenkänning:

```
{ "metadata_storage_name": "file1.pdf", "metadata_author": "John Doe", "content": "Some text content...", "normalized_images": [...] }, "language": "en" }
```

**OCR – Extrahera text från bilder (Skill 2: OCR)**

Om dokumentet innehåller bilder, körs **Optical Character Recognition (OCR)** för att läsa av texten.

- Tjänsten letar efter text i **normalized\_images** och lägger till den i en ny struktur.

### Efter OCR:

```
{ "metadata_storage_name": "file1.pdf", "metadata_author": "John Doe", "content": "Some text content...", "normalized_images": [ { "text": "Extracted text from image" } ], "language": "en" }
```

**Steg 3: Slå ihop all data (Skill 3):**

## Merge)

Nu sammanfogas all text (från **content** och **OCR-extraherad text**) till ett fält kallat **merged\_content**.

### Efter sammanslagning:

- { "metadata\_storage\_name": "file1.pdf", "metadata\_author": "John Doe", "content": "Some text content...", "normalized\_images": [ { "text": "Extracted text from image" } ], "language": "en", "merged\_content": "Some text content... Extracted text from image" } **merged\_content** → Innehåller både dokumenttexten och den OCR-extraherade texten.

### Steg 4: Indexering – Gör datan sökbar

Nu behöver vi mappa fälten i vårt dokument till de sökbara fälten i vårt index.

**Inkommande fält Mappas till indexfält** metadata\_storage\_name file\_name metadata\_author metadata\_author language language merged\_content document\_text **Slutresultat: Indexerat dokument**

Nu ser datan ut så här i indexet:

- { "file\_name": "file1.pdf", "metadata\_author": "John Doe", "language": "en", "document\_text": "Some text content... Extracted text from image" } **file\_name** → Dokumentets namn

- **metadata\_author** → Författare

- **language** → Upptäckt språk

- **document\_text** → All text i dokumentet (inklusive OCR-extraherad text)

### Slutsats

Denna pipeline säkerställer att vi inte bara indexerar rådata, utan att vi **förbättrar** den genom att:

- Identifiera språk
- Extrahera text från bilder
- Slå ihop all relevant text
- Göra det sökbart i en indexerad databas

På så sätt får vi en **bättre och mer kraftfull sökfunktion** som kan hantera både text och bilder! 

## Demo – Create an Azure Cognitive Search Solution



**Create an indexing solution**

**Modify an indexing solution**

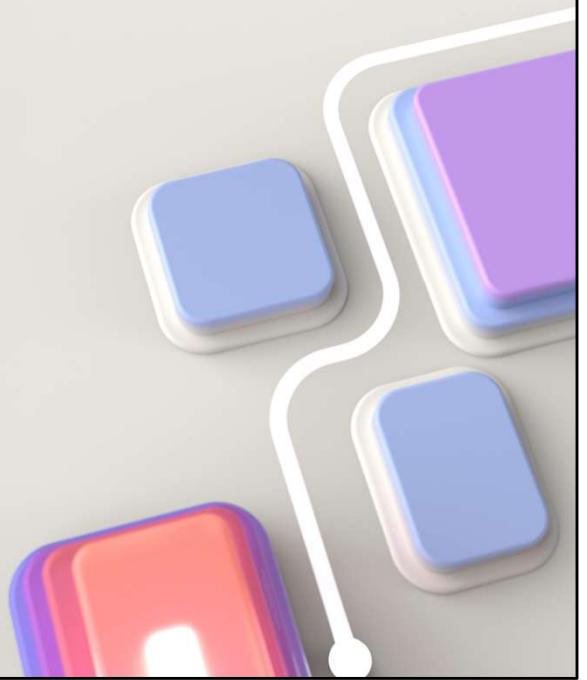
**Query an index from a client application**

© Copyright Microsoft Corporation. All rights reserved.

Instructor demo – can use the exercises here: [mslearn-knowledge-mining/Instructions/Exercises at main · MicrosoftLearning/mslearn-knowledge-mining \(github.com\)](https://github.com/MicrosoftLearning/mslearn-knowledge-mining)

# Create a custom skill for Azure AI Search

© Copyright Microsoft Corporation. All rights reserved.



# Introduction to Custom Skills

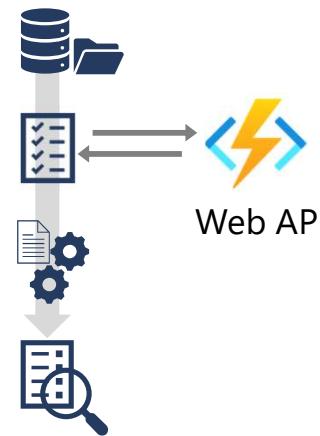
**When built-in skills don't provide what you need...**

**Create a custom skill, for example:**

- Integrate Document Intelligence
- Consume an Azure Machine Learning model
- Any other custom logic

**Custom skills are implemented as Web APIs**

- Commonly Azure Functions



© Copyright Microsoft Corporation. All rights reserved.

## Vad är Custom Skills?

Azure AI Search har många **inbyggda AI-funktioner** (t.ex. OCR, språkigenkänning), men ibland räcker de inte till. Om du behöver något mer avancerat kan du skapa **egna anpassade skills**, kallade **Custom Skills**.

## Hur fungerar Custom Skills?

Custom Skills fungerar som en **Web API-tjänst** som kopplas in i **enrichment pipeline**.

1. **Datan hämtas från en lagringskälla** (exempelvis en databas eller blob storage).

2. **Enrichment Pipeline** kallar en **Custom Skill** via ett Web API.

3. **Custom Skill bearbetar data**, exempelvis:

1. **Integrator dokumentanalys** (t.ex. extrahära nyckelinformation)
2. **Använder en maskininlärningsmodell** (t.ex. sentimentanalys eller klassificering)
3. **Anpassar data med egen logik**

4. **Resultatet skickas tillbaka till pipelinen** och används i indexet.

## Hur implementeras Custom Skills?

- Custom Skills är Web APIs

- Vanligtvis byggs de med **Azure Functions**

- De kan också vara andra REST- eller serverlösningar som tar emot en begäran, bearbetar den och returnerar ett svar.

## Vad visar bilden?

- **Datakälla (överst)** – Här hämtas dokument och data.

- **Enrichment Pipeline (mitten)** – Skickar data till en **Custom Skill** via ett Web API (Azure Function).

- **Indexering (nedre delen)** – Data lagras och görs sökbar.

## Slutsats

Custom Skills ger extra kraft till Azure AI Search genom att låta dig lägga till **valfri anpassad logik**.

Perfekt om du vill:

- Analysera komplexa dokument
- Koppla in maskininlärningsmodeller

Anpassa bearbetningarna efter egna behov  
Så om standardfunktionerna inte räcker – **bygg din egen!** 

# Custom Skill Interfaces

## Input Schema

```
{  
  "values": [  
    {  
      "recordId": "<unique_identifier>",  
      "data": {  
        "<input1_name>": "<input1_value>",  
        "<input2_name>": "<input2_value>",  
        ...  
      }  
    },  
    {  
      "recordId": "<unique_identifier>",  
      "data": {  
        "<input1_name>": "<input1_value>",  
        "<input2_name>": "<input2_value>",  
        ...  
      }  
    },  
    ...  
  ]  
}
```

## Output Schema

```
{  
  "values": [  
    {  
      "recordId": "<unique_identifier_from_input>",  
      "data": {  
        "<output1_name>": "<output1_value>",  
        ...  
      },  
      "errors": [...],  
      "warnings": [...]  
    },  
    {  
      "recordId": "<unique_identifier_from_input>",  
      "data": {  
        "<output1_name>": "<output1_value>",  
        ...  
      },  
      "errors": [...],  
      "warnings": [...]  
    },  
    ...  
  ]  
}
```

This is a *property bag* of values – it can be a single value or a complex JSON structure

© Copyright Microsoft Corporation. All rights reserved.

## Custom Skill Interfaces i Azure AI Search

### Vad handlar detta om?

När du skapar en **Custom Skill** i Azure AI Search kommunicerar din tjänst med **enrichment pipelinen** via JSON-meddelanden.

- **Input Schema** – Hur data skickas till din Custom Skill.
- **Output Schema** – Hur din Custom Skill skickar tillbaka data.

### Hur ser en Custom Skill-begäran ut? (Input Schema)?

Detta är JSON-strukturen som skickas till en Custom Skill:

#### Exempel på input JSON:

```
{  
  "values": [  
    {  
      "recordId": "<unique_identifier>",  
      "data": {  
        "<input1_name>": "<input1_value>",  
        "<input2_name>": "<input2_value>"  
      }  
    },  
    {  
      "recordId": "<unique_identifier>",  
      "data": {  
        "<input1_name>": "<input1_value>",  
        "<input2_name>": "<input2_value>"  
      }  
    },  
    ...  
  ]  
}
```

### Vad betyder detta?

- "values" → En lista med flera dataposter (records) som skickas samtidigt.

- "recordId" → En unik identifierare för varje post.
- "data" → Innehåller alla inparametrar som din Custom Skill behöver.

**Exempel på JSON-data för en översättningsfunktion:**

```
"data": {
  "text": "Hello, how are you?",
  "language": "en"
}
```

**Hur ser Custom Skill-svaret ut? (Output Schema)?**

När Custom Skill har bearbetat datan returnerar den ett JSON-svar i detta format:

**Exempel på output JSON:**

```
{
  "values": [
    {
      "recordId": "<unique_identifier_from_input>",
      "data": {
        "<output1_name>": "<output1_value>"
      },
      "errors": [...],
      "warnings": [...]
    },
    {
      "recordId": "<unique_identifier_from_input>",
      "data": {
        "<output1_name>": "<output1_value>"
      },
      "errors": [...],
      "warnings": [...]
    }
  ]
}
```

**Vad betyder detta?**

- "values" → En lista med resultat för varje input-post.
- "recordId" → Samma ID som i input, så att Azure AI Search vet vilken post som hör till vilken.
- "data" → Innehåller resultatet från Custom Skill.
- "errors" → Eventuella fel.
- "warnings" → Varningar om något gick snett men ändå kunde slutföras.

**Exempel på en översatt text i JSON-format:**

```
"data": {
  "translated_text": "Hej, hur mår du?"
}
```

**Slutsats**

1. **Input** → Azure AI Search skickar data till din Custom Skill via JSON.

2. **Processing** → Din Custom Skill bearbetar datan, exempelvis genom att köra en ML-modell eller analysera text.

3. **Output** → Din Custom Skill skickar tillbaka ett JSON-svar med resultaten.

Detta gör det enkelt att integrera **egna AI-funktioner** i Azure AI Search! 

# Adding a Custom Skill to a Skillset

## Add a Custom.WebApiSkill to the skillset

### Specify the URI to your web API endpoint

- Optionally add parameters and headers

### Set the context to specify at which point in the document hierarchy the skill should be called

### Assign input values

- Usually from existing document fields

### Store output in a new field

- Optionally, specify a target field name (otherwise the output name is used)

```
{  
  "skills": [  
    ...  
    {  
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",  
      "description": "<custom skill description>",  
      "uri": "https://<web_api_endpoint>?<params>",  
      "httpHeaders": {  
        "<header_name>": "<header_value>"  
      },  
      "context": "/document/<where_to_apply_skill>",  
      "inputs": [  
        {  
          "name": "<input1_name>",  
          "source": "/document/<path_to_input_field>"  
        }  
      ],  
      "outputs": [  
        {  
          "name": "<output1_name>",  
          "targetName": "<optional_field_name>"  
        }  
      ]  
    }  
  ]  
}
```

## Adding a Custom Skill to a Skillset

### Vad handlar detta om?

I Azure AI Search kan du lägga till en **Custom Skill** genom att använda **Custom.WebApiSkill** i ett Skillset. Detta gör det möjligt att ansluta en extern **Web API-tjänst** för att berika data innan den indexeras.

### Hur fungerar det?

#### 1. Ange API-adressen (URI)

1. URl:n pekar på din Web API-endpoint.
2. Du kan lägga till **parametrar och headers** vid behov.

#### 2. Ange "context"

1. Definierar var i dokumentet denna skill ska användas.

#### 3. Tilldela indata ("inputs")

1. Vanligtvis kommer indata från dokumentfält som redan finns i datan.

#### 4. Lagra utdata ("outputs")

1. Resultatet kan lagras i ett nytt fält.
2. Du kan ange ett specifikt **target field name**, annars används det namn som skickas ut.

### JSON-exempel för att lägga till en Custom WebApiSkill

```
{  
  "skills": [  
    ...  
    {  
      "@odata.type": "#Microsoft.Skills.Custom.WebApiSkill",  
      "description": "",  
      "uri": "https://<web_api_endpoint>?",  
      "httpHeaders": {  
        "<header_name>": "<header_value>"  
      },  
      "context": "/document/<where_to_apply_skill>",  
      "inputs": [  
        ...  
      ]  
    }  
  ]  
}
```

```
{
  "name": "<input1_name>",
  "source": "/document/<path_to_input_field>"
}
],
"outputs": [
{
  "name": "<output1_name>",
  "targetName": "<optional_field_name>"
}
]
}
]
```

### Vad betyder detta?

- "skills" → En lista över de skills som används i pipelinen.
- "@odata.type" → Anger att det är en **Web API-baserad Custom Skill**.
- "description" → Beskriver vad skillen gör.
- "uri" → Adressen till API:t som anropas.
- "httpHeaders" → Eventuella headers som skickas med API-anropet.
- "context" → Bestämmer var i dokumentet skillen ska tillämpas.
- "inputs" → En lista över fält som skickas till API:t.
  - "name" → Namn på fältet.
  - "source" → Var i dokumentet detta fält hämtas från.
- "outputs" → Definierar vilka resultat API:t returnerar.
  - "name" → Namn på utdatafältet.
  - "targetName" → (Valfritt) Namn på det fält där resultatet ska lagras.

### Exempel på hur detta kan användas

Om du har en **Custom Skill** som analyserar sentiment i text kan input vara:

```
"inputs": [
{
  "name": "text",
  "source": "/document/content"
}]
```

Och output kan vara:

```
"outputs": [
{
  "name": "sentiment_score",
  "targetName": "analyzed_sentiment"
}]
```

### Slutsats

Genom att lägga till en **Custom Web API Skill** kan du ansluta externa AI-modeller eller specialiserad logik till Azure AI Search och förbättra indexeringen. 

## Exercise – Create a Custom Skill for Azure AI Search



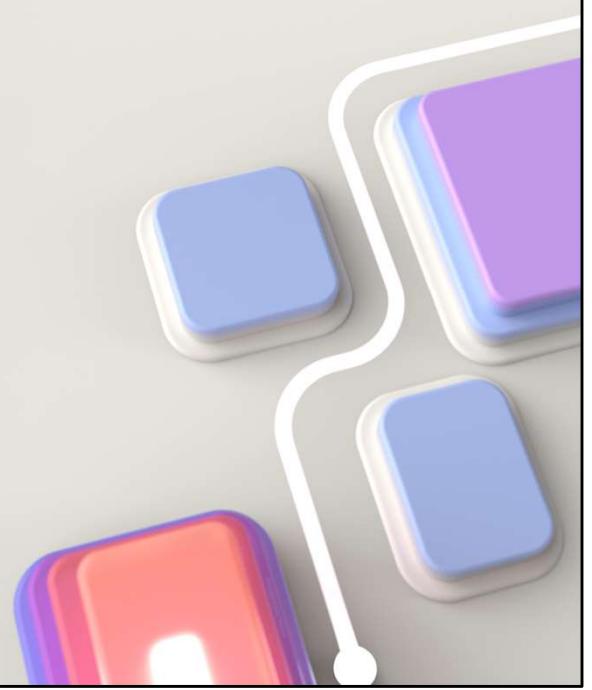
Use an Azure Function to implement a custom skill

Integrate a custom skill into a skillset

© Copyright Microsoft Corporation. All rights reserved.

# Creating a Knowledge Store

© Copyright Microsoft Corporation. All rights reserved.



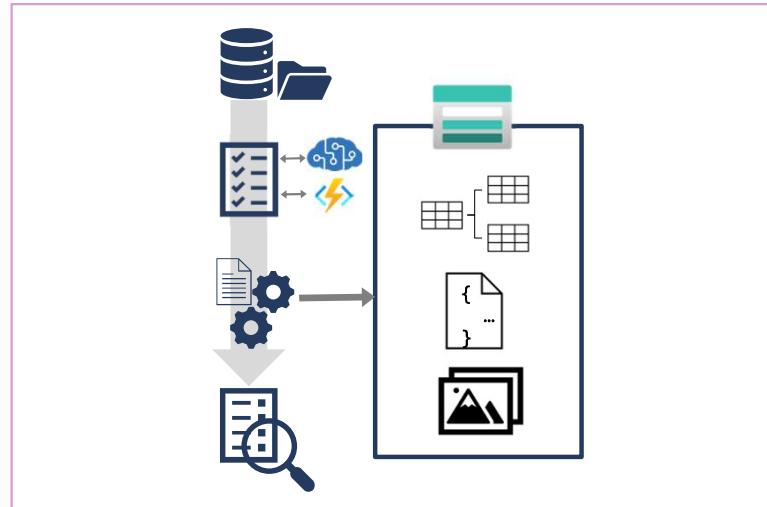
# What is a Knowledge Store?

Persisted insights extracted by indexing process

Stored as *projections* in Azure Storage

- **Tables:** Relational tables with keys for joining
- **Objects:** JSON structures of document fields
- **Files:** Extracted images saved in JPG format

Used for analysis or integration into data processing workflows



© Copyright Microsoft Corporation. All rights reserved.

Vad är en Knowledge Store?

En Knowledge Store är en lagringslösning i Azure AI Search där insikter som extraheras under indexeringsprocessen sparas och organiseras för vidare analys eller integration.

Vad gör en Knowledge Store?

- Samlar in **berikad data** från indexeringsprocessen.
- Strukturera och lagrar den i **Azure Storage**.
- Gör information **tillgänglig för analys och vidare bearbetning**.

Hur lagras datan?

En Knowledge Store sparar data i tre olika format:

**1 Tabeller (Tables)**

- Relationsdatabaser med **nycklar** för att koppla ihop data.
- Används när du behöver **strukturerad data** som kan **frågas** via SQL.

Exempel:

keyphrase\_id keyphrase document\_id 1 "Azure AI" 1001 2 "Machine Learning" 1002 **2 JSON-objekt (Objects)**

- **Strukturerad JSON-data** för att lagra dokumentfält.
- Flexibelt och lätt att bearbeta i moderna dataplattformar.

Exempel på JSON-struktur:

```
{
  "document_id": 1001,
  "title": "AI in Azure",
  "summary": "Azure AI provides powerful tools for machine learning."
}
```

**3 Filer (Files)**

- Extraherade bilder och dokument sparar i JPG eller andra format.
- Perfekt för **bildanalys, OCR och vidare processering**.

Hur används en Knowledge Store?

- Dataanalys** – Gör insikter sökbara och användbara.
- Maskininlärning** – Träna modeller med strukturerad data.
- Integration** – Koppla data till andra system (BI-verktyg, databaser).

#### Slutsats

En **Knowledge Store** gör det möjligt att **strukturera, spara och analysera** AI-berikad data i Azure AI Search. Den är flexibel och kan lagra information som **tabeller, JSON-objekt och filer**, vilket gör den användbar för **avancerade analys- och integrationslösningar**. 

#### Tips för Notepad:

- JSON-strukturen är skriven som **ren text** för att undvika radbrytningsproblem.
- Slå på **Radbrytning** i Notepad (**Visa → Radbrytning**) för bättre läsbarhet.

# Using the Shaper Skill for Projections

## Restructure fields to simplify projections

- Create a JSON object with the fields you want to persist
- Use sourceContext and inputs to map primitives to well-formed JSON objects

```
{  
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",  
  "name": "define-projection",  
  "description": "Prepare projection fields",  
  "context": "/document",  
  "inputs": [  
    {  
      "name": "url",  
      "source": "/document/url"  
    },  
    {  
      "name": "sentiment",  
      "source": "/document/sentiment"  
    },  
    {  
      "name": "key_phrases",  
      "source": null,  
      "sourceContext": "/document/merged_content/keyphrases/*",  
      "inputs": [  
        {  
          "name": "phrase",  
          "source": "/document/merged_content/keyphrases/*"  
        }  
      ]  
    }  
  ],  
  "outputs": [  
    {  
      "name": "output",  
      "targetName": "projection"  
    }  
  ]  
}
```

## Using the Shaper Skill for Projections

### Vad handlar detta om?

I Azure AI Search används **Shaper Skill** för att **omstrukturera fält** och förenkla projektioner. Detta gör att du kan skapa ett välförmerat **JSON-objekt** med de fält du vill spara.

### Hur fungerar det?

1. Skapa ett **JSON-objekt** med de fält som ska behållas.
2. Använd **sourceContext** och **inputs** för att mappa fält till en strukturerad JSON-struktur.
3. Samla information från flera källor och kombinera den i ett enda objekt.

### JSON-exempel för att använda Shaper Skill

```
{  
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",  
  "name": "define-projection",  
  "description": "Prepare projection fields",  
  "context": "/document",  
  "inputs": [  
    {  
      "name": "url",  
      "source": "/document/url"  
    },  
    {  
      "name": "sentiment",  
      "source": "/document/sentiment"  
    },  
    {  
      "name": "key_phrases",  
      "source": null,  
      "sourceContext": "/document/merged_content/keyphrases/",  
      "inputs": [  
        {  
          "name": "phrase",  
          "source": "/document/merged_content/keyphrases/*"  
        }  
      ]  
    }  
  ],  
  "outputs": [  
    {  
      "name": "output",  
      "targetName": "projection"  
    }  
  ]  
}
```

```

"source": "/document/merged_content/keyphrases/"
}
]
}
],
"outputs": [
{
"name": "output",
"targetName": "projection"
}
]
}

```

### Vad betyder detta?

- "@odata.type" → Anger att det är en **Shaper Skill**.
- "name" → Namn på denna skill i pipelinen.
- "description" → Beskriver vad den gör.
- "context" → Bestämmer var i dokumentet denna skill ska appliceras.
- "inputs" → Lista över fält som ska inkluderas i projektionen.
  - **Exempel:**
    - "url" hämtas direkt från /document/url.
    - "sentiment" hämtas från /document/sentiment.
    - "key\_phrases" är mer avancerat – den hämtar flera **nyckelord** från merged\_content/keyphrases/\*.
- "outputs" → Bestämmer var det projicerade resultatet ska lagras.
  - "targetName": "projection" innebär att det lagras i ett nytt fält kallat "projection".

### Exempel på hur detta kan användas

Om du analyserar ett dokument och vill sammanfatta URL, sentiment och nyckelord i ett enda JSON-objekt, kan resultatet se ut så här:

```

"projection": {
"url": "https://example.com",
"sentiment": "positive",
"key_phrases": [
"Azure AI",
"Machine Learning",
>Data Analysis"
]
}

```

### Slutsats

- **Shaper Skill** används för att **strukturera om data** och göra den enklare att indexera.
- Den låter dig **kombinera olika fält** och skapa en mer organiserad JSON-output.
- Perfekt för att skapa **välstrukturerade projektioner** från analysresultat! 🎉

# Using the Shaper Skill for Projections

- **Hur fungerar Shaper Skill?**
  - När data extraheras (t.ex. nyckelord, sentiment, ansiktsigenkänning) är den ofta **ostrukturerad**.
  - *Shaper Skill låter dig **gruppera, organisera och strukturera** data i en **logisk tabell- eller objektstruktur**.*
  - Den skapar en **samling av records** med **unika ID:n** som kan lagras i en Knowledge Store för vidare analys.

## Före Shaper Skill:

```
{  
  "email": {  
    "content": "Hej, vi har ett problem med  
vår server!",  
    "sentiment": "Negative",  
    "sender": "support@company.com",  
    "keywords": ["problem", "server"]  
  }  
}
```

## Efter Shaper Skill:

```
[  
  {  
    "id": "1",  
    "sender": "support@company.com",  
    "sentiment": "Negative",  
    "keywords": ["problem", "server"]  
  }  
]
```

## Using the Shaper Skill for Projections

### Vad handlar detta om?

I Azure AI Search används **Shaper Skill** för att **omstrukturera fält** och förenkla projektioner. Detta gör att du kan skapa ett välförmerat **JSON-objekt** med de fält du vill spara.

### Hur fungerar det?

1. Skapa ett **JSON-objekt** med de fält som ska behållas.
2. Använd **sourceContext** och **inputs** för att mappa fält till en strukturerad JSON-struktur.
3. Samla information från flera källor och kombinera den i ett enda objekt.

### JSON-exempel för att använda Shaper Skill

```
{  
  "@odata.type": "#Microsoft.Skills.Util.ShaperSkill",  
  "name": "define-projection",  
  "description": "Prepare projection fields",  
  "context": "/document",  
  "inputs": [  
    {  
      "name": "url",  
      "source": "/document/url"  
    },  
    {  
      "name": "sentiment",  
      "source": "/document/sentiment"  
    },  
    {  
      "name": "key_phrases",  
      "source": null,  
      "sourceContext": "/document/merged_content/keyphrases/",  
      "inputs": [  
        {  
          "name": "phrase",  
          "source": null  
        }  
      ]  
    }  
  ]  
}
```

```

"source": "/document/merged_content/keyphrases/"
}
]
}
],
"outputs": [
{
"name": "output",
"targetName": "projection"
}
]
}

```

### Vad betyder detta?

- "@odata.type" → Anger att det är en **Shaper Skill**.
- "name" → Namn på denna skill i pipelinen.
- "description" → Beskriver vad den gör.
- "context" → Bestämmer var i dokumentet denna skill ska appliceras.
- "inputs" → Lista över fält som ska inkluderas i projektionen.
  - **Exempel:**
    - "url" hämtas direkt från /document/url.
    - "sentiment" hämtas från /document/sentiment.
    - "key\_phrases" är mer avancerat – den hämtar flera **nyckelord** från merged\_content/keyphrases/\*.
- "outputs" → Bestämmer var det projicerade resultatet ska lagras.
  - "targetName": "projection" innebär att det lagras i ett nytt fält kallat "projection".

### Exempel på hur detta kan användas

Om du analyserar ett dokument och vill sammanfatta URL, sentiment och nyckelord i ett enda JSON-objekt, kan resultatet se ut så här:

```

"projection": {
"url": "https://example.com",
"sentiment": "positive",
"key_phrases": [
"Azure AI",
"Machine Learning",
>Data Analysis"
]
}

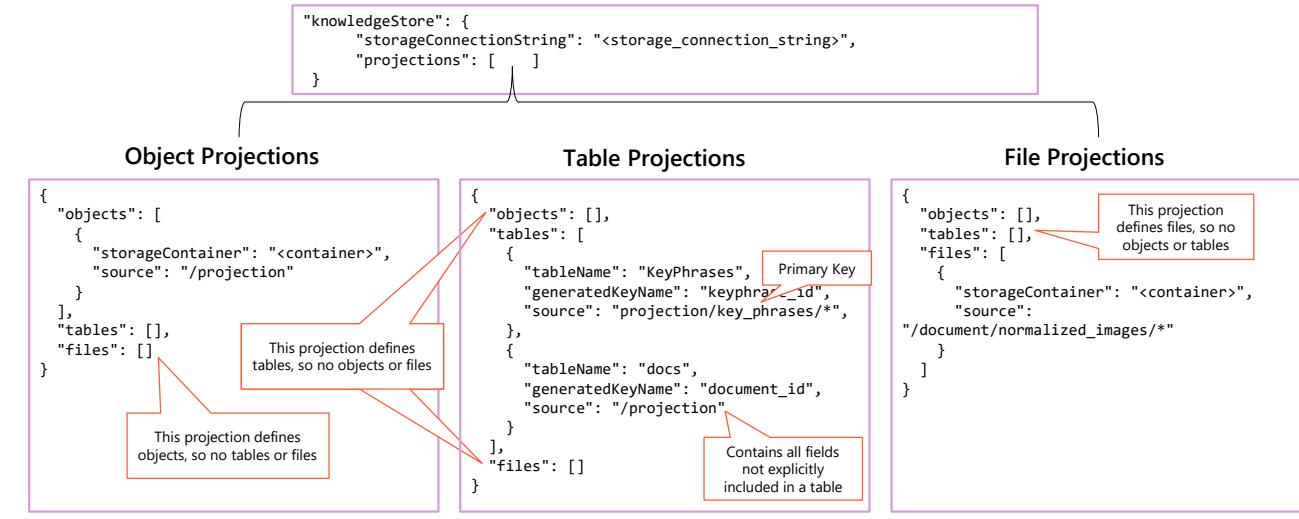
```

### Slutsats

- **Shaper Skill** används för att **strukturera om data** och göra den enklare att indexera.
- Den låter dig **kombinera olika fält** och skapa en mer organiserad JSON-output.
- Perfekt för att skapa **välstrukturerade projektioner** från analysresultat! 🎉

# Implementing a Knowledge Store

## Knowledge Store and Projections are defined in the Skillset



© Copyright Microsoft Corporation. All rights reserved.

## Implementing a Knowledge Store

### Vad handlar detta om?

I Azure AI Search kan du använda en **Knowledge Store** för att lagra och projicera berikad data i olika format. Projektioner definieras i **Skillset** och kan struktureras i:

- **Objektprojektor** (JSON-objekt)
- **Tabellprojektor** (strukturera data i tabeller)
- **Filprojektor** (lagring av filer)

### Definiera en Knowledge Store

```
{  
    "knowledgeStore": {  
        "storageConnectionString": "<storage_connection_string>",  
        "projections": [ ]  
    }  
}
```

- "storageConnectionString" → Anger var datan ska lagras.
- "projections" → Lista över de olika projektionerna (objekt, tabeller, filer).

### Objektprojektor (Object Projections)

```
{  
    "objects": [  
        {  
            "storageContainer": "",  
            "source": "/projection"  
        }  
    ],  
    "tables": [],  
    "files": []  
}
```

- Definierar **objekt** men **inte** tabeller eller filer.

- "storageContainer" → Var objekt ska lagras.
- "source" → Fältet eller sökvägen i pipelinen där data hämtas.

### Tabellprojektorer (Table Projections)

```
{
  "objects": [],
  "tables": [
    {
      "tableName": "KeyPhrases",
      "generatedKeyName": "keyphrase_id",
      "source": "projection/key_phrases/*"
    },
    {
      "tableName": "docs",
      "generatedKeyName": "document_id",
      "source": "/projection"
    }
  ],
  "files": []
}
```

- **Definierar tabeller** men **inte** objekt eller filer.
- "tableName" → Namnet på tabellen i Knowledge Store.
- "generatedKeyName" → Unik primärnyckel för varje post.
- "source" → Varifrån data hämtas.

### Filprojektorer (File Projections)

```
{
  "objects": [],
  "tables": [],
  "files": [
    {
      "storageContainer": "",
      "source": "/document/normalized_images/*"
    }
  ]
}
```

- **Definierar filer** men **inte** objekt eller tabeller.
- "storageContainer" → Var filerna ska lagras.
- "source" → Anger vilka filer som ska lagras.

### Slutsats

En Knowledge Store hjälper dig att lagra berikad data i olika format:

- Objektpunktion** → JSON-data för flexibel användning.
- Tabellprojektor** → Strukturerad data för enklare analys.
- Filprojektor** → Lagring av bilder eller andra filer.

## Extended interactive exercises – Create a Knowledge Store with Azure AI Search



<https://aka.ms/km-ai-lp>

© Copyright Microsoft Corporation. All rights reserved.

Labs to complete on students own time. These indicated exercises are recommended for them to complete on Learn. <https://aka.ms/km-ai-lp>

# Knowledge check



1 You want to enrich an index by extracting any geographical locations mentioned in the source data. Which built-in skill should you use?

- Entity Recognition
- Key Phrase Extraction
- Language Detection

2 You have implemented a custom skill as an Azure function. How can you include the custom skill in your indexing process?

- Add a Merge skill to the skillset to combine output from built-in skills with your custom skill.
- Add a WebApiSkill to a skillset, referencing the Azure function's URI
- Add a Shaper skill to the skillset to create a collection of records with unique IDs generated by your custom

3 You want to create a knowledge store that contains JSON representations of the extracted data. What kind of projection should you define?

- File
- Object
- Table

© Copyright Microsoft Corporation. All rights reserved.

1 a

2 b

3 c

Bilden visar en "Knowledge check" med tre flervalsfrågor relaterade till Azure Cognitive Search och dess indexeringsfunktioner. Här är en genomgång av varje fråga med både rätt och fel svar:

Fråga 1:

"You want to enrich an index by extracting any geographical locations mentioned in the source data. Which built-in skill should you use?"

- Rätt svar: *Entity Recognition*
  - Denna inbyggda färdighet kan extrahera enheter såsom geografiska platser från text.
- Fel svar:
  - *Key Phrase Extraction* → Identifierar nyckelfraser men extraherar inte specifikt geografiska namn.
  - *Language Detection* → Identifierar vilket språk texten är skriven på men har inget att göra med geografiska enheter.

Fråga 2:

"You have implemented a custom skill as an Azure function. How can you include the custom skill in your indexing process?"

- Rätt svar: *Add a WebApiSkill to a skillset, referencing the Azure function's URI*
  - *WebApiSkill* gör det möjligt att integrera en extern tjänst (t.ex. en Azure Function) i indexeringsprocessen.
- Fel svar:
  - *Add a Merge skill to the skillset to combine output from built-in skills with your custom skill.* → *Merge skill* används för att slå ihop data från olika källor, men det kopplar inte in en extern tjänst.
  - *Add a Shaper skill to the skillset to create a collection of records with unique IDs generated by your custom skill.* → *Shaper skill* används för att omorganisera och strukturera data men används inte för att koppla in externa API:er.

**Fråga 3:**

"You want to create a knowledge store that contains JSON representations of the extracted data. What kind of projection should you define?"

- **Rätt svar:** *Table*

- Tabellen är rätt val för att lagra strukturerad data (JSON-representationer) i en organiserad form.

- **Fel svar:**

- *File* → Detta skulle lagra data som en fil snarare än en strukturerad tabell.
  - *Object* → Objekt kan vara en bra representation av JSON men är inte det rätta valet för en kunskapsdatabas i det här sammanhanget.

Sammanfattning:

• Fråga 1 handlar om att extrahera geografiska namn – *Entity Recognition* är rätt.

• Fråga 2 gäller att koppla in en Azure Function i indexeringsprocessen – *WebApiSkill* är rätt.

• Fråga 3 handlar om att skapa en kunskapsdatabas med JSON-data – *Table* är rätt.

*Table* är det bästa valet i frågan eftersom det ger en **strukturerad, sökbar och filtrerbar representation** av den **extraherade datan**.

*Object* skulle göra det svårare att analysera och hantera informationen, eftersom all data skulle ligga som JSON utan tydlig uppdelning i kolumner.

## Learning Path Recap

In this learning path, we learned how to:

Create an Azure AI Search Solution

Implement a custom skill for Azure AI Search and integrate it into a skillset

Create a knowledge store with object, file, and table projections

© Copyright Microsoft Corporation. All rights reserved.



© Copyright Microsoft Corporation. All rights reserved.