

Content Sources

Here's a breakdown of **content sources**, **CMS**, and **content modules** based on the provided context:

Content Sources

A **content source** refers to the platform, database, or service where your content is stored. These sources can be headless CMS tools (e.g., Contentful, Sanity), databases (e.g., PostgreSQL), local files (e.g., Markdown), or even non-traditional services (e.g., Figma).

Key Characteristics of Content Sources

- **Two-Way Syncing:** Enables loading and modifying content dynamically via the **Content Source Interface (CSI)**. This ensures changes in your content source (e.g., CMS) reflect in the Visual Editor and vice versa.
- **Types of Content Sources Supported by CSI:**
 - **API-Based CMS:** Such as Contentful, Sanity, or Contentstack.
 - **Databases:** External or internal databases like PostgreSQL or PlanetScale.
 - **File-Based Content:** Markdown files or other stored assets.
 - **Non-Traditional Services:** For example, using Figma as a content repository.

Example Integration with a Content Source:

Here's an example where Contentful is added as a content source:

```
1  // stackbit.config.js
2  import { ContentfulContentSource } from "@stackbit/cms-
    contentful";
3
4  export default {
5    contentSources: [
6      new ContentfulContentSource({
7        spaceId: process.env.CONTENTFUL_SPACE_ID,
8        environment: process.env.CONTENTFUL_ENVIRONMENT,
9        previewToken: process.env.CONTENTFUL_PREVIEW_TOKEN,
10       accessToken: process.env.CONTENTFUL_MANAGEMENT_TOKEN,
11     }),
12   ],
13 };;
```

The above demonstrates how the Visual Editor interacts with a configured content source like Contentful. Content sync happens through APIs or other mechanisms defined in the source module.

CMS (Content Management Systems)

A **Content Management System (CMS)** is a software platform that facilitates creating, managing, and modifying digital content. Headless CMS platforms are particularly relevant for modern web development, as they provide **API-first content delivery** without being tied to a frontend.

Key Points About CMS:

- **Headless CMS:** Unlike traditional CMS systems (e.g., WordPress), API-based CMS platforms such as Contentful, Contentstack, and Sanity allow developers to retrieve content purely through APIs, making them ideal for integration with tools like Visual Editor.
- **CRUD Operations via CSI:** With CSI, CMS platforms allow seamless interaction (create, read, update, delete) between Visual Editor and the CMS.
- **Use Case:** CMS platforms store structured content (e.g., articles, events, product descriptions) that can easily be queried and presented on websites, applications, or any digital interface.

Examples of Supported CMS:

- **Officially Supported:** Contentful, Contentstack, Sanity, DatoCMS.
- **Experimental Support:** Strapi, Shopify, Figma.

CMS Workflow in Visual Editor:

1. Install the corresponding SDK or API client for the desired CMS (e.g., `@stackbit/cms-contentful` for Contentful).
2. Configure the content source for Visual Editor, specifying API keys and other parameters.
3. View and edit the CMS data directly in the Visual Editor, with two-way syncing in real-time.

By using the CMS, Visual Editor becomes a direct interface for content management.

Content Modules

A **content module** represents a custom integration or plugin that connects a content source (like a CMS, database, or file system) with the Visual Editor. Modules implement the **Content Source Interface (CSI)** to enable real-time interaction and data management.

Key Characteristics of Content Modules:

- Written as **JavaScript classes** that conform to the **Content Source Interface**.
- Provide methods like `loadContent`, `saveContent`, `deleteContent`, and `loadSchema` to enable interaction with the source.
- These modules must be configured in `stackbit.config.js` for Visual Editor to load and manage data from the content source.

Custom Modules and BYOS (Bring Your Own Source):

- Even if a content source is not officially supported, you can build your own module.
- Essential Requirements for Building Modules:
 - Your content source should support read/write data via its API or local access.
 - Implement the methods defined by the CSI API (e.g., CRUD operations, uploading assets).

Example of Multiple Content Modules:

Below is an example showcasing two content sources being used together:

```
1  // stackbit.config.js
2  import { ContentfulContentSource } from "@stackbit/cms-contentful";
3  import { SanityContentSource } from "@stackbit/cms-sanity";
4
5  export default {
6    contentSources: [
7      new ContentfulContentSource({
8        spaceId: process.env.CONTENTFUL_SPACE_ID,
9        environment: process.env.CONTENTFUL_ENVIRONMENT,
10       previewToken: process.env.CONTENTFUL_PREVIEW_TOKEN,
11       accessToken: process.env.CONTENTFUL_MANAGEMENT_TOKEN,
12     }),
13     new SanityContentSource({
14       projectId: process.env.SANITY_PROJECT_ID,
15       dataset: process.env.SANITY_DATASET,
16       token: process.env.SANITY_TOKEN,
```

```
17     } ),  
18     ],  
19     } ;
```

This example demonstrates connecting multiple CMSES in one project, allowing for simultaneous management of data from both services in the Visual Editor.

Visual Relationships Between the Concepts

Term	Definition	Example
Content Source	A storage platform for structured/unstructured content that can sync with the Visual Editor.	Contentful, Sanity, PostgreSQL, Markdown
CMS (Headless CMS)	A backend service or API-first platform for managing and delivering content.	Contentful, Contentstack, Sanity
Content Module	A JavaScript class that connects a content source to the Visual Editor through the CSI API.	<code>ContentfulContentSource</code> or a custom module

Practical Benefits of Using CSI, CMS, and Content Modules

1. **Unified Interface:** Bring multiple content sources (CMS, databases, files) into one interface for seamless management.
2. **Two-Way Sync:** Ensure changes propagate between the Visual Editor and the content source instantly.
3. **Scalable Architecture:** Integrate new sources or migrate from one CMS to another with minimal overhead.
4. **Customizability:** Use custom content modules to support any source—not just officially supported CMS platforms.

In summary:

- **Content Sources** are the platforms where your content resides.
- **CMS** represents a specific type of content source focused on managing structured data.
- **Content Modules** act as the bridge between the content sources and the Visual Editor, enabling real-time operations and data management.

This ecosystem ensures flexibility, scalability, and efficient workflows for content management and delivery.