

Exercise: Creating the Character Content Type

In this module we will cover the following topics:

- How to build a Content Type
- Review Field Types

Marketing has provided us with requirements to promote our Composable Heroes by creating a section of the site where each hero will have their own detail page and be listed on a gallery page.

In the next few exercises we will define the content type for a Composable Hero and build a hero detail page. There's no time to waste, let's get started!

Analyze the Page Design

When defining a new Content Type it's important to have the page design available (in photo / HTML format) so we can model the Content Type and define fields for the content elements.

Analyze the following pages and think of the properties that a superhero character would need. When analyzing pages we often look for as many representations of the same content as possible, and then we combine all these fields into one Content Type. In this exercise, let's take a look at these two pages and see what fields they contain. We will focus on the main page content and ignore the headers and footer. If you would like to learn more, the Content Modeling course covers the content type fields in more detail.



GOLDEN BASKET

Power: Running Fast



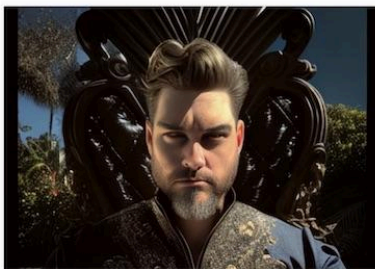
DEMO ALMIGHTY

Power: Power of the Gods



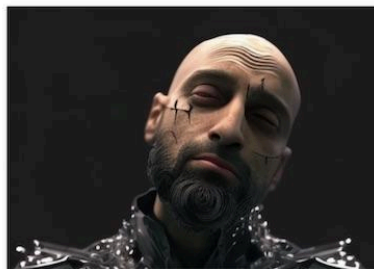
CYBER SURGE

Power: Super strength



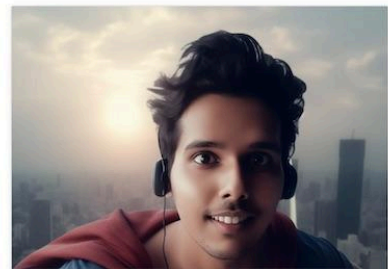
ROYAL SANCTUARY

Power: Telekinesis



CYBER SAVIOR

Power: Super strength



THE TECHNOMANCER

Power: Telekinesis



CYBER SURGE

Christine was busy helping a client when a group of hackers infiltrated the system. As she tried to fend them off, a stray voltage surge hit her, causing her to collapse. She woke up later in a hospital bed with various cybernetic implants. With her newfound abilities, including enhanced strength and hacking skills, she decided to use her powers for good. She kept her job as a Senior Product Manager while secretly fighting cybercrime as a cyborg-themed superhero.

Earth



Email : christine@email.com

Phone : 555-555-5555

Power : Super strength

Defining the Character Content Type for the Hero

In the previous section we analyzed the hero pages and thinking of what fields to use for defining the pages. In this section, we will create the Content Type for the Character. If you wish to copy / paste the code samples later, we would advise you to define the content type with the fields in this exercise. However, if you wish to use your own field definitions, that's fine, but you will also need to adjust the code samples accordingly.

Creating the Content Type

[Video](#)

Using the Content Type editor in Contentstack, create a new Content Type with fields for our Character content. While defining the fields, we will need to consider what content we want to show on the detail page.

After the exercise, your Content Type should look like this:



Character



Name

Default field



URL

Default field



Image



Description



Powers



Contact Info



Phone



Email



Home World

Steps for the exercise:


1. Create a new Content Type, named 'Character' for the composable heroes
2. Add the following fields:

Hero Name - Use the Title field


Every Content Type in Contentstack has a Title field. We can give it a different display name such as 'Name' or 'Character Name', but we cannot remove it.

URL

For the Path, use '/composable-heroes/' and then select the option to append the Title to the URL. This is the most common way to define the URL structure. For Content Types where we know the kind of content each page has (character) and that they will have detail pages, we decide to use '/composable-heroes/' as the URL path and then add the title to the end.

 **URL - Properties**

Basic | **Advanced**


URL Pattern 

/composable-hero/

☒ Title

☐ Date and Title

☐ Month and Title

☐ Custom 

Enter custom URL

☐ Disable

Image (File)

Contentstack has an [Images API](#) and supports uploading any media file, including images, up to 700 MB. For images, there are additional properties such as

min/max on width or height, so the CMS ensures the images used in a field meet the minimum size requirements.

The screenshot displays the CMS configuration interface for an 'Image' field. On the left, a sidebar lists fields: 'Image' (selected), 'Description' (with a {T} icon), 'Powers' (with a list icon and a 'Multiple' button), 'Contact Info' (with a folder icon), 'Phone' (with an A icon), 'Email' (with an A icon), and 'Home World' (with a G icon). The main panel shows the 'File - Properties' configuration for the 'Image' field. It has two tabs: 'Basic' and 'Advanced' (which is active). Under 'Options', there are four checkboxes: 'Mandatory' (unchecked), 'Multiple' (unchecked), 'Allow Images Only' (checked), and 'Non-localizable' (unchecked). To the right, the 'File Size Limit (MB)' is set from 'Min' to 'Max', with a note: 'Each time you add a new file type, press 'Enter.''. Below this, a note states: 'If enabled, editing this field is restricted in localized entries. The field will use the value of the master-language entry in all localized entries.' Under 'Image Dimension Validation', there are two rows: 'Width' (checked) with a 'Min' dropdown set to '300', and 'Height' (unchecked) with a 'Min' dropdown set to 'Min (px)'.

Description (JSON RTE)

The JSON RTE field stores the content in JSON and the frontend webapp uses a [JSON RTE Serializer](#) after publishing to transform the JSON to HTML. There is also an [Automation Hub connector that can transform the RTE JSON to HTML](#).

The JSON RTE supports custom plugins and you can use the [App SDK](#) and extend the Rich Text Area with your own customizations. Examples in the Marketplace include a word count extension and a Cloudinary extension.

Powers (select field, add the powers)

Powers

Multiple

Contact Info

Phone

Email

Home World

SEO

Meta Title

Meta Description

Meta Keywords

Enable Search Ind

Select - Properties

BasicAdvanced

Display Name*

Powers

Unique ID* ?

powers

Instruction Value ?

Enter value

Help Text ?

Enter value

Selection Type ?

Multiple Choices

Limit for Multiple Choices ?

Min

Max

Display Type ?

Dropdown

Choice Data Type ?

☒ text

☐ number

Add Choices ?

☐ Enable key-value

Enter values here

+ Add

Enter values (choices), each in a separate line, and click 'Add'.

Flying

Running Fast

Contact Info (Group field, email, phone)

Group fields help us organize content fields and in the UI the editor can collapse / open the group. They're also helpful when using the [Visibility Rules](#) in Content Types


Home world (Reference field)

The Home world has an image and some additional properties, so we use a Reference Field and define a Content Type for the Home World instead of using a select field or defining the fields here.

Reference fields should be used when the content you want to include either already exists as a standalone entry (or part of one), or will be used by multiple content entries (either of the same type or not)

SEO

Add a Global Field and choose the SEO Global Field

 **Global - Properties**

Basic

Advanced

Display Name*

SEO

Help Text ?

Enter value

Unique ID* ?

seo

Instruction Value ?

Enter value

// //

Select Global Field

SEO x

All Done!

Congrats - you are done with the Content Type definition.

3. Now, let's try out the new Character Content Type. Create a new Entry with the Character Content Type and fill in the content and publish to Development

Content Type Best Practices

URL field: Best practices/patterns. Use the `/path/:title` for most content types, where relevant. This maps to the Route in the web application and is also important for SEO. Choose this wisely.

Image field: Consider using the height/width limitations

Text fields: Consider any validations

Modular Blocks: Think of this as a 'design system' of components the user can choose from. They have the flexibility to add it if they want, or not. Blocks can also be added at a later time to extend the content type. This can be done in confidence that things will not break if adjustments are made.

If you're using a modern framework such as NextJS or Nuxt, then we can think of having a NextJS Component responsible for rendering the content of one Modular Block. In this way, the content and design are atomic and can be re-used across multiple pages.

Reference fields: When re-using content with many entries, we can use a Reference Field and connect the Entries. We will need to add an additional query parameter to the Content Delivery Queries so that the referenced Entry fields will also be returned. One example of using a reference field is on a homepage or landing page where we want to also show a listing of articles, products in a promo, etc.

However, be mindful when implementing Reference fields as they can be easily overused. When modeling content, it's best to keep the relationship as flat as possible and only use reference fields when referring to another type of content that is mostly independent of the model it is being referenced in. If content fields are only used by the current entry, then it's better to include them in the content type rather than using a reference field. Reference field over-use is the most common content modeling pitfalls and increases the complexity of the system for end users.

Create the NextJS Component for the superhero detail page

Next step is to render the Character Entries on a superhero detail page. This process involves the following steps:

1. Create a new NextJS Component and import the Contentstack libraries
2. Use the HTML Provided and add the code to render the Contentstack Character Content Type fields
3. Run locally and preview the page

Now that we have defined our superheroes using the Character Content Type and added some sample content, it's time to create a detail page to display them on our website. The HTML file for the superhero detail page is provided below. Using Contentstack's Content Delivery API and Next.js, we can create the code to render this page on our website.

Programming the Character Detail Page

1. Create a new NextJS Component for the detail page in a folder 'composable-heroes' and using the name [post].tsx
2. Analyze the code for the Detail page and notice how we render text fields, reference fields and group fields. [The code is available on Github here.](#)
3. Examine the Helper/index.js file and [review the code for getting a Single Entry \(Character Content Type\)](#). This function will use the existing method 'GetEntryByUrl' in the /contentstack-sdk/index.js file to query for the Entry content.

```
export const getComposableHeroSingleRes = async (entryUrl) => {
  const response = await Stack.getEntryByUrl({
    contentTypeUid: "character",
    entryUrl,
    referenceFieldPath: ["home_world"],
    jsonRtePath: ["description"],
  });

  liveEdit && addEditableTags(response[0], "character", true);
  return response[0];
};
```

Content Delivery JS SDK API Ref

Examples using the Content Delivery JS SDK in a NextJS Service

4. Call the Character Query function from the NextJS `composable-heroes/[post].tsx` file

```
export async function getServerSideProps({ params }: any) {
  try {
    const posts = await getComposableHeroSingleRes(`/composable-heroes/${params.post}`);
    if (!posts) throw new Error('404');

    return {
      props: {
        pageUrl: `/composable-heroes/${params.post}`,
        superHeroPost: posts,
      },
    };
  } catch (error) {
    console.error(error);
    return { notFound: true };
  }
}
```

Code Solution for Composable Hero Detail Page on Github

Field rendering syntax:

Text fields - Rendering the title field

`{postData?.title}`

Group fields

`{postData?.contact_info?.email ?`

Email : `{postData?.contact_info?.email}`

Images

```

<img
  className='superHero-detail-img img-fluid mb-5'
  src={postData?.image.url}
  alt={postData?.image?.filename}
  {...postData?.image.$?.url as {}}
/>

```

Reference Field

```

{
  postData?.home_world?.map((homeWorld: {
    title: string | undefined;
    image: { url: string | undefined; $: { url: {} }; }; filename: string; };
  ),
    indx: {}) => (
    <div key={indx.toString()} className="mb-3">
      {homeWorld?.title ? <p><strong>{homeWorld?.title}</strong></p> : ''}
      {homeWorld?.image?.url ?
        <img
          className='superHero-logo-img img-fluid mb-3'
          src={homeWorld?.image?.url}
          alt={homeWorld?.image?.filename}
        />
        : ''}
      <hr />
    </div>

```

6. Confirm your code with the Composable Detail Page solution

Final Result - Composable Hero Detail Page



CYBER SURGE

Christine was busy helping a client when a group of hackers infiltrated the system. As she tried to fend them off, a stray voltage surge hit her, causing her to collapse. She woke up later in a hospital bed with various cybernetic implants. With her newfound abilities, including enhanced strength and hacking skills, she decided to use her powers for good. She kept her job as a Senior Product Manager while secretly fighting cybercrime as a cyborg-themed superhero.

Earth



Email : christine@email.com

Phone : 555-555-5555

Power : Super strength