



adaptTo()

EUROPE'S LEADING AEM DEVELOPER CONFERENCE

27th – 29th SEPTEMBER 2021

SPA Editing with Sling to the rescue

Gabriel Walt, Hanish Bansal, Sharanya Vinod – Adobe

DEFINITION

The page doesn't reload when the visitor navigates.

IMPLIES

- The frontend consumes content as JSON from the backend.
- The frontend renders the view with JavaScript.

~~IMPLIES~~ DOES **NOT** NECESSARILY IMPLY

- Authors don't control the view as it's decoupled from the backend.
- Developers need to implement each view in the frontend.

Form-based editing

AUTHORS

- ⊕ Form-based content editing offers a guided process.
- ⊕ Authors can be faster and more systematic.
- ⊖ Authors cannot create content-specific layout.
- ⊖ Authors cannot optimize the content to the context.

DEVELOPERS

- ⊕ Content retrieval follows a strict content schema.
- ⊕ Developers can be faster and have more freedom.
- ⊖ Developers become a bottleneck in the creation process.

Demo: in-context editing

The demo works in both AEM Cloud Service and 6.5.

Page Model vs GraphQL

Both content API can be used together.

Use GraphQL JSON

- When the developer owns the layout, the developer knows best what properties are necessary to render the content.

Use Page Model JSON

- When the author owns the layout, the configured properties are expected to be considered by the rendering.

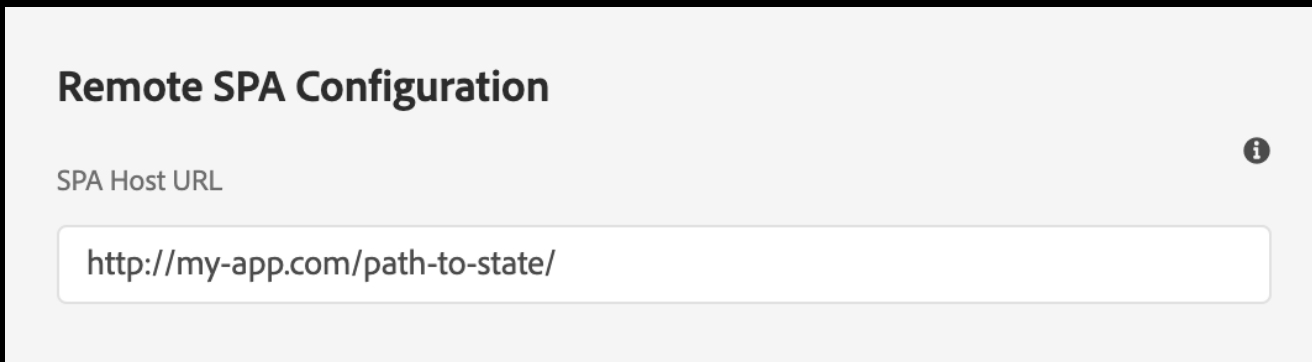
In-context editing

- 1. Authors can preview the content within the context.**
 - The editor loads the experience as it is delivered to the visitors.
- 2. Authors can optimize the content to the context.**
 - The editor recognizes the content items originating from its backend and allows to edit them.
- 3. Authors can insert content with layout.**
 - The editor offers building blocks (components) to create content and layout.

1. Load the experience

Currently done by the remote page component:

- Set the external URL of the state to load with a **page property**.

A screenshot of a configuration form titled 'Remote SPA Configuration'. The form has a light gray background. Below the title, there is a label 'SPA Host URL' on the left and a small circular information icon on the right. A text input field below the label contains the URL 'http://my-app.com/path-to-state/'.

1. Editor to load the experience

Currently done by the remote page component:

- Set the external URL of the state to load with a page property.
- Requires the SPA to provide an `asset-manifest.json`

```
my-app.com/asset-manifest.json
{
  - files: {
    static/js/0.chunk.js: "/static/js/0.chunk.js",
    static/js/0.chunk.js.map: "/static/js/0.chunk.js.map",
    main.js: "/static/js/main.chunk.js",
    main.js.map: "/static/js/main.chunk.js.map",
    runtime-main.js: "/static/js/bundle.js",
    runtime-main.js.map: "/static/js/bundle.js.map",
    index.html: "/index.html",
    static/media/icon-close.svg: "/static/media/icon-close.f227c489.svg",
    static/media/icon-loading.svg: "/static/media/icon-loading.200bc7ed.svg",
    static/media/wknd-logo-dk.svg: "/static/media/wknd-logo-dk.36593320.svg"
  },
  - entrypoints: [
    "static/js/bundle.js",
    "static/js/0.chunk.js",
    "static/js/main.chunk.js"
  ]
}
```


1. Editor to load the experience

Currently done by the remote page component:

- Set the external URL of the state to load with a page property.
- Requires the SPA to provide an `asset-manifest.json`
- Provided by github.com/adobe/aem-spa-project-core

1. Editor to load the experience

Currently done by the remote page component:

- Set the external URL of the state to load with a page property.
- Requires the SPA to provide an `asset-manifest.json`
- Provided by github.com/adobe/aem-spa-project-core
- Available in github.com/adobe/aem-project-archetype with option `frontendModule=react` or `angular`

2. Editor to recognize editable content

Information required to make a content item editable:

- Content ID → The Sling resource path
- Content type → The Sling resource type

♥ Thank you, Sling!

2. Editor to recognize editable content

In the HTML DOM rendered by the SPA:

- Resource path: `data-cq-data-path`
- Resource type: `data-cq-resource-type`

```
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title>My Single Page App</title>
    <link rel="stylesheet" href="/style.css" type="text/css">
    <script src="/script.js" defer></script>
  </head>
  <body>
    <div class="cmp-container" data-cq-data-path="/content/my-app/path-to-state/jcr:content/root" data-cq-resource-type="my-app/components/container">
      <div class="cmp-title" data-cq-data-path="/content/my-app/path-to-state/jcr:content/root/title" data-cq-resource-type="my-app/components/title">
        <h1>Welcome!</h1>
      </div>
      <div class="cmp-text" data-cq-data-path="/content/my-app/path-to-state/jcr:content/root/text" data-cq-resource-type="my-app/components/text">
        <p>Hello World</p>
      </div>
    </div>
  </body>
</html>
```

2. Editor to recognize editable content

In the JSON retrieved, this information is required too:

my-app.com/path-to-state.model.json

```
{
  title: "My Single Page App",
  :type: "my-app/components/page",
  - :items: {
    - root: {
      :type: "my-app/components/container",
      - :items: {
        - title: {
          title: "Welcome!",
          type: "h1",
          :type: "my-app/components/text"
        },
        - text: {
          text: "Hello World",
          :type: "my-app/components/text"
        }
      },
    },
    - :itemsOrder: [
      "title",
      "text"
    ]
  },
  - :itemsOrder: [
    "root"
  ]
}
```

→ Resource path: /content/my-app/path-to-state/jcr:content
Resource type: my-app/components/**page**

→ Resource path: /content/my-app/path-to-state/jcr:content/**root**
Resource type: my-app/components/**container**

→ Resource path: /content/my-app/path-to-state/jcr:content/root/**title**
Resource type: my-app/components/**title**

→ Resource path: /content/my-app/path-to-state/jcr:content/root/**text**
Resource type: my-app/components/**text**

3. Editor to offer content building blocks

Map the JS component to the resource types:

```
import { withMappable } from "@adobe/aem-react-editable-components";
```

```
const TextRender = ({ cqPath, text }) => {  
  return (<div className="cmp-text"><p>{text}</p></div>);  
};
```

```
export const Text = withMappable(TextRender,  
  { resourceType: "my-app/components/text", ... });
```

3. Editor to offer content building blocks

Delegate the layout to a container component:

```
import { ResponsiveGrid } from "@adobe/aem-react-editable-components";
```

```
export const Container = withMappable(ResponsiveGrid,  
  { resourceType: "my-app/components/container", ... });
```

```
<Container  
  pagePath="/content/my-app/path-to-state"  
  itemPath="root"/>
```

3. Editor to offer content building blocks

The layout is passed to the JS ResponsiveGrid component as CSS classes in the JSON:

```
my-app.com/path-to-state.model.json

{
  title: "My Single Page App",
  :type: "my-app/components/page",
  - :items: {
    - root: {
      :type: "my-app/components/container",
      gridColumnNames: "aem-Grid aem-Grid--12 aem-Grid--default--12",
      - columnClassNames: {
        title: "aem-GridColumn aem-GridColumn--default--4",
        text: "aem-GridColumn aem-GridColumn--default--8"
      },
      - :items: {
        - title: {
          title: "Welcome!",
          type: "h1",
          :type: "my-app/components/text"
        },
      },
    },
  },
}
```


What's new?

- **Remote page component**

Open an existing SPA from the AEM Page Component

- Support for Angular, in addition to React
- Capability to open child states directly

- **Virtual component support**

The content structure can also be driven by the SPA.

These two new capabilities are sometimes referred to as “SPA Editor 2.0”.

Demo: the code setup

The code samples work in both AEM Cloud Service and 6.5.

Get started right away!

Try the remote SPA tutorial:

→ bit.ly/remote-spa-tutorial

SPA Editor overview:

→ bit.ly/spa-editor-overview