**List Child Pages As A Dropdown**

This Java servlet named **DropDownServlet** retrieves a list of items from a specified path in the repository and exposes them as a datasource for a dynamic dropdown in Adobe Experience Manager (AEM).

The **doGet** method in the **DropDownServlet** class handles HTTP GET requests sent to the servlet. When a client, such as a web browser, makes a GET request to the URL mapped to this servlet, the **doGet** method is invoked. In this method, the servlet performs its logic, which in this case involves dynamically generating the options for a dropdown menu, and then sends the response back to the client.

**protected** **void** doGet(SlingHttpServletRequest request,

SlingHttpServletResponse response)

ResourceResolverFactory**:**

This is a Sling service that you can use to create new ResourceResolver instances. This is useful for cases outside of servlets, like background jobs or OSGi services.

Once a **Resource** object is obtained, we can access its properties using the **getValueMap()** method to retrieve the page title.

Once we have the **Resource** object, we can access various properties associated with the resource. In the example, we access the **jcr:title** property using **getValueMap().get("jcr:title", String.class)**. This retrieves the title of the page represented by the resource.

Now, let's relate these concepts to the example path **/apps/mysite/components/1-digiquad-blueheader**:

**ResourceResolver**:

If we want to access or manipulate the resource located at this path, we would use a **ResourceResolver** to resolve the path and obtain the corresponding **Resource** object.

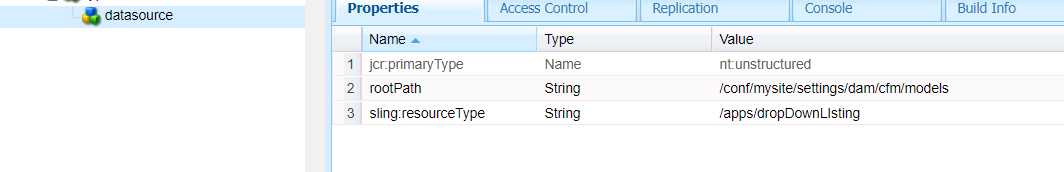
**Resource**:

Once resolved, the **Resource** object would represent the content and metadata of the component located at **/apps/mysite/components/1-digiquad-blueheader**. This **Resource** may contain properties such as **sling:resourceType**, **jcr:title**, and others, which describe the component.

Getchild():

Getchild method used to take the child properties

String rootPath = pathResource.getChild("datasource").getValueMap().get("rootPath",String.**class**);



The purpose of the code block is to create a list of key-value pairs where the key is the name of a resource and the value is the same as the key. This can be used to populate a drop-down menu with resource names as the options.

list.forEach(res -> {

//ValueMap valueMap = res.getValueMap();

String title = res.getName();

dropDownList.add(**new** KeyValue(title, title));

});

**dropDownList** contains **KeyValue** objects. If you pass this list directly to a **DataSource**, it might not work properly in an AEM component that expects a **DataSource** backed by **Resource** objects.

By converting each **KeyValue** object into a **ValueMapResource**, you're providing a resource-like representation of your data, which can be more seamlessly integrated into AEM components:

@SuppressWarnings("unchecked")

DataSource ds =

**new** SimpleDataSource(

**new** TransformIterator(

dropDownList.iterator(),

input -> {

KeyValue keyValue = (KeyValue) input;

ValueMap vm = **new** ValueMapDecorator(**new** HashMap<>());

vm.put("value", keyValue.key);

vm.put("text", keyValue.value);

**return** **new** ValueMapResource(

resourceResolver, **new** ResourceMetadata(),

JcrConstants.***NT\_UNSTRUCTURED***, vm);

}));

------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------