ColumnType Class

window.almworks.structure.api.ColumnType

ColumnType class represents column type.

It needs to be subclassed for particular column type implementation.

Example

```
var api = window.almworks.structure.api;

var AwesomeColumnType = api.subClass('AwesomeColumnType', api.ColumnType, {
  createSwitchTypePreset: function(context) { return { key: 'com.acme.structure.awesome-column', params: {} };
},
  createAddColumnPresets: function(context) { return [
   { key: 'com.acme.structure.awesome-column', params: {} },
   { key: 'com.acme.structure.awesome-column', name: 'Awesome Column with a Twist', params: { twist: true } }
  ]; },
  createConfigurator: function(context, spec) { return new AwesomeColumnConfigurator({context: context, spec:
spec}); },
  createColumn: function(context, spec) { return new AwesomeColumn({context: context, spec: spec}); }
});

api.registerColumnType(new AwesomeColumnType(), 'com.acme.structure.awesome-column');
```

Methods

`createSwitchTypePreset(context)`

Returns default column specification to use when the user switches to this column type from another column type in the column configuration panel. May return `null` if the column type is unavailable.

`createAddColumnPresets(context)`

Returns an array of column presets (specifications) for this type to be offered to the user in the Add Column panel. May return an empty array if the column type is unavailable.

`createColumn(context, spec)`

Returns a new instance of `Column` subclass for the specified column specification. May return `null` if the specification is invalid, the column type is unavailable, etc.

`createConfigurator(context, spec)`

Returns a new instance of `ColumnConfigurator` subclass for the specified column specification. May return `null` if the specification is invalid, the column type is unavailable, etc.

`getPresetMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this column type to create presets. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createSwitchTypePreset()` or `createAddColumnPresets()` is called. See Requesting and Using Metadata for more information. By default returns `null`, which means that no metadata is needed.

`getColumnMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this column type to create `Column` instances. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createColumn()` is called, and also will be available to the created `Column` instance via `this.context`. See Requesting and Using Metadata for more information. By default returns `null`, which means that no metadata is needed.

`getConfigMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this column type to create `ColumnConfigurator` instances. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createConfigurator()` is called, and also will be available to the created `ColumnConfigurator` instance via `this.context`. See Requesting and Using Metadata for more information. By default returns `null`, which means that no metadata is needed.

`getMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this column type. Unless the AJAX requests fail, the metadata will be available through `context.getMetadata(key)` when `createSwitchTypePreset()`, `createAddColumnPresets()`, `createColumn()`, or `createConfigurator()` is called, and also will be available to the created `Column` and `ColumnConfigurator` instances via `this.context`. See Requesting and Using Metadata for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    somedata: {                         // metadata key
      url: baseUrl + '/some/data/url', // request URL
      cacheable: true,                  // if the response for this URL can be reused for other cacheable
requests
      extract: function(response) {    // response to the AJAX request
        return response.property || 1; // the actual value for context.getMetadata('somedata')
      }
    },
    otherdata: {
      url: baseUrl + '/other/data/url',
      cacheable: true
    }
  };
}
```