

ColumnConfigurator Class

window.almworks.structure.api.ColumnConfigurator

ColumnConfigurator class encapsulates everything related to column type configuration.

It needs to be subclassed for a particular column type implementation and passed as return value in [ColumnType.createConfigurator\(\)](#) method.

Example

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

var MyColumnConfigurator = api.subClass('MyColumnConfigurator', api.ColumnConfigurator, {
  getDefaultColumnName: function() { return 'My Column'; }
  getOptions: function() {
    return [ new MyOption1({configurator: this}), new MyOption2({configurator: this}) ];
  }
});
```

Required Methods

You have to override the following methods in the subclass.

getColumnTypeName()

Returns column type name, used in the column configuration panel.

getDefaultColumnName()

Returns default column name.

Other Methods

These methods may be optionally overridden.

init(options)

Optional initializer.

getGroupKey()

Return column preset's group key. See [registerColumnGroup\(\)](#) for reference.

getMetadataRequests()

Returns a JavaScript object specifying the metadata needed by this configurator to set up the UI. See [Requesting and Using Metadata](#) for more information. By default returns null, which means that no metadata is needed.

Example

```

getMetadataRequests: function() {
  return {
    somedata: {
      url: baseUrl + '/some/data/url', // 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
    }
  };
}

```

getOptions()

Returns array of column type options. Each option should be a subclass of [ColumnOption Class](#).

Example

```

getOptions: function() {
  return [ new MyOption1({configurator: this}), new MyOption2({configurator: this}) ];
}

```