Column Class

window.almworks.structure.api.Column

A subclass of Column class represents column objects of a specific type.Columns need to be subclassed for a particular column type implementation. You can override methods while subclassing to modify the default behavior.

Example

```
var api = window.almworks.structure.api;
var MyColumn = api.subClass('MyColumn', api.Column, {
  init: function() {
    ...
  },
  getCellViewHtml: function() {
    return '<div> ... </div>';
  }
});
```

Properties

`context`

Contains context information about where the column is used. See The Column Context for more information.

`spec`

Contains column specification object. Specification object is serialized as a part of the overall view specification and stored on the server and in the browser's local storage. See Column Specifications for more information.

Methods

`init(options)`

Initializer method.

`getCellValueHtml(renderingParameters)`

Returns HTML that is displayed in the grid cell for a specific issue. The HTML should contain the value provided by this column. Structure will also wrap the value in decorative elements – this could be overridden by providing `getCellViewHtml()` method.

Parameters

renderingParameters.getAttributeValue()	returns current row's attribute value
renderingParameters.getRowId()	returns current row's id
renderingParameters.getItemId()	return current row's item id

Example

```
var Template = require('almworks/util/Template');
var cellTemplate = new Template('<span class="acme-field">{awesomefield}</span>');
getCellValueHtml: function(rp) {
  return cellTemplate.renderHtml({ awesomefield: rp.getAttributeFieldValue({id: 'com.acme.awesome-data',
format: 'text'}) });
}
```

`getCellViewHtml(renderingParameters)`

Returns customized HTML that is displayed in the grid cell for a specific issue. By default, calls `getCellValueHtml()` and wraps the retrieved value into the default Structure style. Can be overridden to allow higher degree of control over the cell appearance.

Parameters

renderingParameters.getAttributeValue()	returns current row's attribute value
renderingParameters.getRowId()	returns current row's id
renderingParameters.getItemId()	return current row's item id

`collectRequiredAttributes(attributeSet)`

Lets column request attributes that are needed for rendering. The attributes are provided on the server side by AttributeLoaderProvider.

Parameters

attributeSet.requireAttribute(attributeSpec, forestSpec)	Method for collecting required attributes. Parameters are: • `attributeSpec` is the attribute specification object • `forestSpec` is the forest specification for the forest, from which attribute should be loaded **(optional)**

About Attribute Specs

AttributeSpec defines the attribute and format to be loaded. See Loading Attribute Values for more information on attributes.

Some of the attributes are shown below. You can also define your own attribute, calculate it on the server side and request from your column.

About Forest Spec

Forest specification is optional. When used, it allows you to get attribute value from a different forest – however, it must be related to the forest being displayed, otherwise it will not have the same rows.

For example, you can specify a forest specification with some transformation to display values from there in the untransformed forest. There are also two special values for `forestSpec`:

- `'displayed'` is the default value, meaning "use the forest that is being displayed"
- `'unfiltered'` means "use the same forest, but remove all filters that are coming at the end of transformation chain"

Example

```
collectRequiredAttributes: function(attributeSet) {
  attributeSet.requireAttribute({id: 'key', format: 'text'});
  attributeSet.requireAttribute({
    id: 'sum',
    format: 'number',
    params: {
      id: 'customfield',
      format: 'number',
      params: {
        fieldId: 10010
      }
    }
  }, 'unfiltered');
  attributeSet.requireAttribute({id: 'com.mycompany.work-stats', format: 'json'});
}
```

Some of the attributes provided by Structure:

Attribute Spec	Example	Description
`{id: <jira-field-id>, format: 'html'}`		The HTML representation of a JIRA issue field value, as seen on the issue page or in the Issue Navigator. Structure allows non-issue items also have these values. `<jira-field-id>` is the common name for the JIRA's standard field id. This attribute does not load custom fields.
`{id: 'customfield', format: 'html', params: { fieldId: <field-numeric-id> }}`		HTML representation of a custom field value.

`{id: 'project', format: 'id'}`		Project ID for the issues. The `id` format means either a string or a number, depending on what is being used for identifying the object.
`{id: 'editable', format: 'boolean'}`		Boolean value telling whether the item can be edited by the user.

See also CoreAttributeSpecs for examples of bundled attributes.

`getDefaultName()`

Must return default column name, assigned when user adds column of specified type to the structure view. Returns empty string by default.

Example

```
getDefaultName: function() { return 'My Column'; }
```

`isResizable()`

Returns whether the column is resizable or not. Returns `true` by default.

Example

```
isResizable: function() { return false; }
```

`canShrinkWhenNoSpace()`

Returns whether column can shrink beyond minimum size if there's not enough space on the screen. Returns `false` by default.

Example

```
canShrinkWhenNoSpace: function() { return true; }
```

`isAutoSizeAllowed()`

Returns if the column should be auto-resized to fit its contents. Returns `false` by default.

Example

```
isAutoSizeAllowed: function() { return true; }
```

`getMinWidth()`

Returns minimum width of the column in pixels. Returns 27 by default.

Example

```
getMinWidth: function() { return 100; }
```

`getDefaultWidth()`

Returns default width of the column in pixels. Returns 120 by default.

Example

```
getDefaultWidth: function() { return 100; }
```

`getHeaderCellHtml()`

Returns HTML that will be used in the grid header. By default returns cell with column name in default Structure style.

Example

```
getHeaderCellHtml: function() { return '<div>' + this.name + '</div>'; }
```

`getMetadataRequests()`

Returns a JavaScript object specifying the metadata needed by this column to render the values. See Requesting and Using Metadata for more information. By default returns `null`, which means that no metadata is needed.

Example

```
getMetadataRequests: function() {
  return {
    status: {
      url: baseUrl + '/rest/api/2/status',
      cacheable: true
    }
  };
}
```

`getSortAttribute()`

Returns attribute specification for sorting when the user clicks on the header. If `null` is returned (the default), the clicking this column header does not result in added sorting transformation.

`isSortDescendingByDefault()`

If returns `true` the initial direction of the sorting will be descending.