

# 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

<code>renderingParameters.getFieldValue()</code>	returns current row's field value
<code>renderingParameters.getIssueId()</code>	returns current row's issue id
<code>renderingParameters.getEscapedFieldValue()</code>	returns current row's field value escaped

### Example

```
getCellValueHtml: function(rp) {
    return '<span class="acme-field">' + rp.getEscapedFieldValue('com.acme.structure:awesome-field-data') + '</span>';
}
```

### 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.getFieldValue()	returns current row's field value
renderingParameters.getIssueId()	returns current row's issue id
renderingParameters.getEscapedFieldValue()	returns current row's field value escaped

## getRequiredFields()

Returns array of data fields required to display column. Those data must be provided by pluggable data providers in Java code.

### Example

```
getRequiredFields: function() {
    return [ 'com.acme.structure:awesome-field-data', 'jira:raw:status' ];
}
```

The fields below can be provided by Structure, so you don't need to write your own data providers.

Format	Example	Description
jira:html:<fieldID>	jira:html:assignee	The HTML representation of an issue field value, as seen on the issue page or in the Issue Navigator.
jira:raw:<fieldID>	jira:raw:issuekey jira:raw:project	The "raw", unformatted representation of an issue field. This can be a string or numeric ID of the referenced entity. The following fields are supported: <ul style="list-style-type: none"> <li>• summary</li> <li>• issuekey</li> <li>• issuetype</li> <li>• resolution</li> <li>• status</li> <li>• project</li> <li>• priority</li> <li>• assignee</li> <li>• reporter</li> </ul>
jira:permission:<permission>	jira:permission:edit	Returns "1" if the user has the given permission for the issue, "0" otherwise.
jira:icon:<fieldID>	jira:icon:project	The icon for the referenced entity, as an HTML fragment. The following fields are supported: <ul style="list-style-type: none"> <li>• issuetype</li> <li>• priority</li> <li>• status</li> <li>• assignee</li> <li>• reporter</li> <li>• project</li> </ul>
structure:sum:<fieldID>	structure:sum:timespent structure:sum:votes structure:sum:customfield_10153	The sum of a numeric or duration field over the issue's subtree in the structure. The value is formatted according to the user's locale.

## 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
    }
  };
}
```