POLARION 2404

Widget SDK Documentation

Polarion 2404

About Siemens Digital Industries Software

Siemens Digital Industries Software is a leading global provider of product life cycle management (PLM) software and services with 7 million licensed seats and 71,000 customers worldwide. Headquartered in Plano, Texas, Siemens Digital Industries Software works collaboratively with companies to deliver open solutions that help them turn more ideas into successful products. For more information on Siemens Digital Industries Software products and services, visit https://www.siemens.com/plm

Support Center: https://www.support.sw.siemens.com

Send Feedback on Documentation: https://www.support.sw.siemens.com/doc_feedback_form

1 Java widget example

1.1 Introduction

This example will detail how to create a custom Java widget for **Pages**.

What can be extended:

- By developing your own widgets you can extend your widget library in **Pages**.

What will be shown in the example:

- How to create a custom widget. A *Work Record Report* widget will be implemented that calculates work records from items defined by its parent and link role.

1.2 Java API Workspace preparation

See the *Workspace preparation* section in the main Polarion SDK Guide.

1.3 Create a Project plugin

The best way to create a Project plugin is to import the provided example. (It contains all the dependencies needed to create a custom plugin.)

1.3.1 Import the example

> Info: Make sure that your plugin is compiled against **the Polarion version that you use**. This example contains a precompiled jar plugin. You can remove it before you start developing your own plugin based on this example. Eclipse ensures that the new jar plugin will be created against your source code and Polarion version.

1. Select **File** > **Import...**
2. In the dialog that appears, select **Existing Project into Workspace** in the **General** section and click **Next**.
3. Click **Browse**, and select the directory containing the examples. (Usually in `C:\Polarion\polarion\SDK\examples\`).
4. Submit it.
5. Select `com.polarion.example.widget` and click **Finish**.

1.4 Deploy to an installed Polarion

See the *Deployment to Installed Polarion* section in the main Polarion SDK guide.

1.5 Execute from your Workspace

See the *Execution from Workspace* section in the main Polarion SDK guide.

1.6 Configuration

After a successful deployment of the plug-in into Polarion the Work Record widget can be used in **Pages**.

> Info: A custom widget plugin is deployed using HiveMind. See /src/META-INF/hivemodule.xml for more details. Deployment configuration is located at /com.polarion.alm.ui/ui.jar/META-INF/richpages-hivemodule.xml.

2 Velocity widget example

2.1 Introduction

This example will detail how to create a custom, Velocity based, widget for **Pages**.

What can be extended:

- By developing your own widgets you can extend your widget library for **Pages**.

What will be shown in the example:

- How to create a custom Widget. A *Work Record Report* widget will be implemented that calculates work records from items defined by its parent and link role.

2.2 Deployment

2.2.1 Import the example

1. Use your favorite SVN client.
2. Commit the contents of `com.polarion.example.velocitywidget` (usually in `C:\Polarion\polarion\SDK\examples\`) to `.polarion/pages/widgets` in the repository.
3. The new widget can now be used from the widgets sidebar in the Page Designer (i.e. *a* **Page** in edit mode).

> Info: Velocity widgets can be deployed to the repository for both the Global and Project scopes. Widgets deployed for the Global scope will be visible to all Projects. If there are two widgets with the same ID deployed on both Global and Project scopes, then the widget in the Project scope will be used.

2.3 Tags

A widget's tags (or categories), that are specified as a comma separated list in the *tags* property in the *widget.properties* file, can be any String or localization key. For the localization keys of the standard tags see the javadoc for `com.polarion.alm.shared.api.model.rp.widget.RichPageWidget.getTags(SharedContext)`

3 Velocity Macro Example

3.1 Introduction

This example will detail how to deploy a custom macro for **Pages.**

What can be extended:

- By developing your own macros you can extend your macro library for **Pages**.

What will be shown in the example:

- How to create custom macro. A custom Info box will be implemented that displays information.

3.2 Deployment

3.2.1 Import the example

1. Use your favorite SVN client.
2. Commit the contents of `com.polarion.example.velocitymacro` (usually in `C:\Polarion\polarion\SDK\examples\`) to `.polarion/pages/scripts/velocity`. in the repository.
3. A new macro from the Script - Block widget and Script - Inline widget can now be used.

3.2.2 Usage

1. Place a **Script**, a **Block**, a **Script** or an **Inline** widget to your **Page** via the widgets sidebar.
2. Import macros using `#import("macros.vm").`
3. Use an infoBox macro like `#infoBox("your message").`

> Info: Polarion first searches for the included file in the Project scope. If it doesn't find it there, then it is taken from the Global scope.

4 Customize an existing widget within a Script widget

4.1 Introduction

This example will detail how to customize an existing Widget within a script widget using the **Copy Widget Source Code** operation.

What will be shown in the example:

- How to customize an existing **Multi Line Trend Chart** widget. (The number of work items, separated by category in a time axis will be displayed.)

4.2 Implementation

1. Insert a **Multi Line Trend Chart** widget into a **Page**.
2. Change title of the widget and click **Apply**.
   (So that all parameters are copied into the widget's source code. They are needed for the following steps.)
3. Click on the **Operation** menu button in the widget's header and select **Copy Widget Source Code**.
4. Insert a **Script Block** widget and paste the copied source code into its *Script* parameter in the opened **Parameters** sidebar. (You will see the basic widget definition with pre-filled parameters.)
5. Modify the source code to look like the code shown below. (The only changes should be the first line with the Velocity assignment and the content of the `<sub id="elements">` element.)

```
#set($projectId = "PROJECT_ID")
<div class="polarion-rp-widget-part" data-widget="com.polarion.multiSetTrendChart">
  <span class="polarion-rp-widget-parameters">
    <sub id="title">Work Items count separated by category</sub>
    <sub id="data">
      <sub id="elements">
        #set($categories = $transaction.categories().search().query("project.id:$projectId"))
        #foreach ($category in  $categories)
          <sub>
            <sub id="prototype">WorkItem</sub>
            <sub id="queryType">lucene</sub>
            <sub id="luceneQuery">categories.id:$category.fields.id.get</sub>
            <sub id="children">
              <sub id="name">$category.fields.name.get</sub><sub id="color"></sub>
            </sub>
          </sub>
        #end
      </sub>
    </sub>
    <sub id="timeAxis">
      <sub id="from">
        <sub id="relative">-30</sub>
      </sub>
      <sub id="to">
        <sub id="relative">0</sub>
      </sub>
      <sub id="scaleType">week</sub>
      <sub id="baseDate">1</sub>
    </sub>
  </span>
</div>
```

6. Change the `projectId` variable so that it points to your chosen project.
7. Click **Apply**.

> Info: There are two kinds of **Script** widgets - **Inline** and **Block**. **Inline** widgets can be used inside a paragraph because they don't break the text. **Block** widgets are used as separate components.

5 Extend Velocity context example

5.1 Introduction

This example will detail how to deploy and use your own HiveMind services or Java instances in scripts, widgets and macros.

What can be extended:

- By developing your own services and objects, you can extend the Velocity context used in **Pages**, widgets, macros and parameters that support it.

What will be shown in the example:

- How to extend the Velocity context. *Work Item Util* will be implemented and contributed as a Java instance that will count the time spent on a given Work Item. Then *Repository Util* will be implemented and contributed as a HiveMind service that provides a method for getting the last revision in the repository, and provide the *IRepositoryService*.

5.2 Java API workspace preparation

See the *Workspace preparation* section in the main Polarion SDK Guide.

5.3 Create a Project plugin

The best way to create a Project plugin is to import a provided example that contains all the dependencies needed to create your custom plugin.

5.3.1 Import the example

> Info: Make sure that your plugin is compiled against **the Polarion version that you use**. This example contains a precompiled jar plugin. You can remove it before you start developing the plugin based on this example. Eclipse ensures that the new jar plugin will be created against your source code and Polarion version.

1. Select **File** > **Import...**
2. In the dialog that appears, select **Existing Project into Workspace** in the **General** section and click **Next**.
3. Click **Browse** and select the directory of examples. (Usually in `C:\Polarion\polarion\SDK\examples\`).
4. Submit it.
5. Select `com.polarion.example.velocitycontext` and click **Finish**.

5.4 Deployment to Installed Polarion

See the *Deployment to Installed Polarion* section in the main Polarion SDK guide.

5.5 Execution from Workspace

See the *Execution from Workspace* section in the main Polarion SDK guide.

5.6 Configuration

After a successful deployment of the plug-in into Polarion, you can start using *Work Item Util* and *Repository Util* within your Velocity code on **Pages**:

- `$workItemUtil` - Work Item Util
- `$repositoryUtil` - Repository Util

5.6.1 Work Item Util usage example

Create a page with, for example, a Script Block widget. Use the following as the script's content:

```
$workItemUtil.printTimeSpent($transaction.workItems().getBy().ids("PROJECT_ID", "WORKITEM_ID"))
```

Replace `PROJECT_ID` and `WORKITEM_ID` with Project and Work Item IDs that have work records.

> Info: The Velocity Context plugin is deployed using HiveMind. See /src/META-INF/hivemodule.xml for details.

5.7 Create an Inline widget

To enable an **Inline** widget, create a `widget.properties` file in the target widget's folder and add the fillowing line:

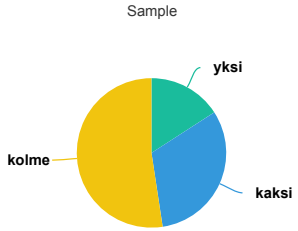
```
inline=true
```

6 Small Examples

6.1 Create a custom Highcharts chart

This example details how to use the Script Block widget to generate a custom Highcharts chart using Velocity and the Polarion Rendering API.

```
#set($html = $renderingContext.createHtmlFragmentBuilder())

#set($chart = $renderingContext.createChartBuilder())
#set($pie = $chart.pie().name("Number of demands"))

$!pie.addValue("yksi", 1).null
$!pie.addValue("kaksi", 2).null
$!pie.addValue("kolme", $math.toDouble("3.3")).null

$!chart.build().title().text("Sample").null
$!chart.build().plotOptions().pie().addRawAttribute("tooltip", "{pointFormat: '<b>{point.y}</b>'}").null

$!chart.build().render($html, 300, $renderingContext.columnWidth).null

$html
```



6.2 Set raw attributes for a Highcharts chart

This example shows how to set raw attributes for these charts.

```
#set($html = $renderingContext.createHtmlFragmentBuilder())
#set($builder = $renderingContext.createChartBuilder().build())

$!builder.title().text("Chart").null
$!builder.chart().type().line().null

$!builder.addRawAttribute("noData","{style: {fontWeight: 'bold', fontSize: '15px', color: '#FF0000'}}").null

$!builder.render($html, 300, $renderingContext.columnWidth).null

$html
```

The Highchart is now configured and you can use Velocity to pull data from Polarion to populate the chart.

6.3 Render a set of objects as a table

The following code examples show how to use the **Script Block** widget to render a table that displays object properties. The object set is defined by a query.

EXAMPLE 1:

```
<table class="polarion-rpw-table-content">
  <tbody>
    <tr class="polarion-rpw-table-header-row">
      <th>Item</th>
      <th>Status</th>
      <th>Parent</th>
    </tr>
#foreach($w in $transaction.workItems.search.query("project.id:drivepilot"))
    <tr class="polarion-rpw-table-content-row">
      <td>$w.render.withLinks.withTitle</td>
      <td>$w.fields.status.render</td>
```

```
      <td>
        #foreach($l in $w.fields.linkedWorkItems.direct)
          #if($l.fields.role.get.id == "parent")
            $l.fields.workItem.render.withTitle.withLinks
          #end
        #end
      </td>
    </tr>
#end
  </tbody>
</table>
```

The code above renders the following table:

Item	Status	Parent
SDK-2 - Child Requirement	Open	SDK-1 - Parent Requirement
SDK-1 - Parent Requirement	Open	

EXAMPLE 2:

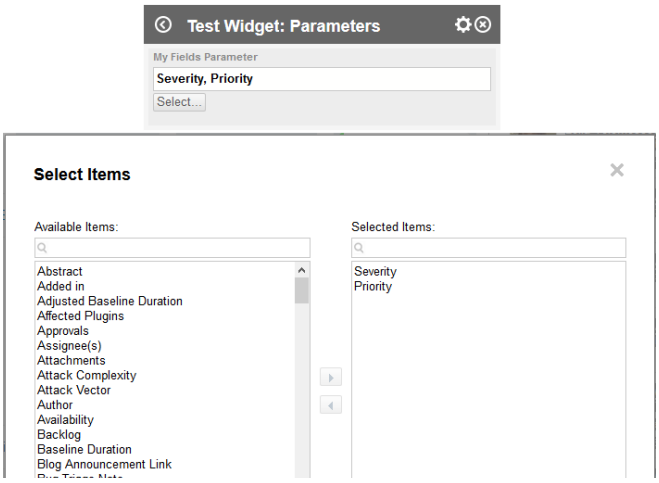
```
<table class="polarion-rpw-table-content">
  <tbody>
    <tr class="polarion-rpw-table-header-row">
      <th>Plan #</th>
      <th># of Work Items at Start</th>
      <th># of WIs at End</th>
    </tr>

#set($counter=1)

#foreach($plan in $transaction.plans().search().limit(19))
  #if($plan.fields().startedOn().get() && $plan.fields().finishedOn().get() && $plan.fields().name().get().contains("Team"))
    #set($revisionStarted = $transaction.revisions().getBy().dateOrAfter($plan.fields().startedOn().get()).fields().id().get())
    #set($planAtStart = $transaction.plans().getBy().revision($revisionStarted).path($plan.getReference().toPath()))
    #set($revisionFinished = $transaction.revisions().getBy().dateOrAfter($plan.fields().finishedOn().get()).fields().id().get())
    #set($planAtEnd = $transaction.plans().getBy().revision($revisionFinished).path($plan.getReference().toPath()))
    #if(!$planAtStart.isUnresolvable() && !$planAtEnd.isUnresolvable())

    <tr class="polarion-rpw-table-content-row">
      <td>$counter</td>
      <td>$planAtStart.items().size()</td>
      <td>$planAtEnd.items().size()</td>
    </tr>

    #set($counter=$math.add($counter, 1))
    #end
  #end
#end

  </tbody>
</table>
```

6.4 Widget parameter for selecting fields

This example shows how to use the **Field selector** widget parameter and how to define which fields are selected by default.

This is the content of widget's **parameters.vm** file:

```
#set($myFields = $objectFactory.newSet())

#set($void = $myFields.add("priority"))
#set($void = $myFields.add("severity"))

$parameters.put("myFieldsParameter", $factory.fields("My Fields Parameter").fields($myFields).build())
```



6.5 Get a historical object by date

The following code example details how to get a respective revision for a given date. (Followed by how to use that revision to get the historical data of an object.)

Example displays **number of work items** linked to plans when **started** and **finished**.

```
<table class="polarion-rpw-table-content">
  <tbody>
    <tr class="polarion-rpw-table-header-row">
      <th>Plan #</th>
      <th># of Work Items at Start</th>
      <th># of WIs at End</th>
    </tr>

#foreach($plan in $transaction.plans().search())
  #set($revisionStarted = $transaction.revisions().getBy().dateOrAfter($plan.fields().startedOn().get()).fields().id().get())
  #set($planAtStart = $transaction.plans().getBy().revision($revisionStarted).path($plan.getReference().toPath()))
  #set($revisionFinished = $transaction.revisions().getBy().dateOrAfter($plan.fields().finishedOn().get()).fields().id().get())
  #set($planAtEnd = $transaction.plans().getBy().revision($revisionFinished).path($plan.getReference().toPath()))

    <tr class="polarion-rpw-table-content-row">
      <td>$plan.fields().name().get()</td>
      <td>$planAtStart.items().size()</td>
      <td>$planAtEnd.items().size()</td>
    </tr>

#end

  </tbody>
</table>
```

Getting the **historical data** for a revision:

```
<table class="polarion-rpw-table-content">
  <tbody>
    <tr class="polarion-rpw-table-header-row">
      <th>Plan #</th>
      <th># of Work Items at Start</th>
      <th># of WIs at End</th>
    </tr>

#set($counter=1)

#foreach($plan in $transaction.plans().search().limit(19))
  #if($plan.fields().startedOn().get() && $plan.fields().finishedOn().get() && $plan.fields().name().get().contains("Team"))
    #set($revisionStarted = $transaction.revisions().getBy().dateOrAfter($plan.fields().startedOn().get()).fields().id().get())
    #set($planAtStart = $transaction.plans().getBy().revision($revisionStarted).path($plan.getReference().toPath()))
    #set($revisionFinished = $transaction.revisions().getBy().dateOrAfter($plan.fields().finishedOn().get()).fields().id().get())
    #set($planAtEnd = $transaction.plans().getBy().revision($revisionFinished).path($plan.getReference().toPath()))
    #if(!$planAtStart.isUnresolvable() && !$planAtEnd.isUnresolvable())

    <tr class="polarion-rpw-table-content-row">
      <td>$counter</td>
      <td>$planAtStart.items().size()</td>
      <td>$planAtEnd.items().size()</td>
    </tr>

    #set($counter=$math.add($counter, 1))
    #end
  #end
#end

  </tbody>
</table>
```

6.6 How to use Page Parameters in Velocity and queries

6.6.1 String parameters

The examples below refer to a page parameter with "Name" as the **ID**, "String" as the **Type** and "John Doe" as the **Value**.

To access the instance:

`$pageParameters.Name` => StringParameterImpl: Name, John Doe

To obtain the value:

`$pageParameters.Name.value` => John Doe

To obtain the value usable in SQL queries:

`$pageParameters.Name.toSql` => John Doe

To obtain the value usable in Lucene queries:

`$pageParameters.Name.toLucene` => "John Doe"

6.6.2 Integer parameters

Examples below refer to a page parameter "Year" as the **ID**, "Integer" as the **Type** and "2014" as the **Value**.
To access the instance:
`$pageParameters.Year` => IntegerParameterImpl: Year, 2014

To obtain the value:
`$pageParameters.Year.value` => 2014

To obtain the value usable in SQL queries:
`$pageParameters.Year.toSql` => 2014

To obtain the value usable in Lucene queries:
`$pageParameters.Year.toLucene` => 2014

6.6.3 Boolean parameters

Examples below refer to a page parameter with "Advanced" as the **ID**, "Boolean" as the **Type** and "Yes" as the **Value**.
To access the instance:
`$pageParameters.Advanced` => BooleanParameterImpl: Advanced, true

To obtain the value:
`$pageParameters.Advanced.value` => true

To obtain the value usable in SQL queries:
`$pageParameters.Advanced.toSql` => true

To obtain the value usable in Lucene queries:
`$pageParameters.Advanced.toLucene` => true

6.6.4 Enumeration Parameters

Examples below refer to a page parameter with "reqtype" as the **ID**, "Enumeration" as the **Type** and "Functional" and "Business" as the **Values**.
To access the instance:
`$pageParameters.reqtype` => EnumParameterImpl: Requirement Types, [functional, business]

To obtain the list of values:
`$pageParameters.reqtype.values` => [ProxyEnumOption id:functional, ProxyEnumOption id:business]
Returned list is usable in #foreach loop. List is returned also if the enumeration is not multi-valued.

To obtain the first value (or the sole value for single-value enumeration):
`$pageParameters.reqtype.singleValue` => ProxyEnumOption id:functional

To obtain the value usable in SQL queries:
`$pageParameters.reqtype.toSql` => ('functional', 'business')
Usage example: `WHERE C_TYPE IN $pageParameters.reqtype.toSql`

To obtain the value usable in Lucene queries:
`$pageParameters.reqtype.toLucene` => (functional business)
Usage example: `reqtype:$pageParameters.reqtype.toLucene`

6.6.5 Custom Enumeration Parameters

Examples below refer to a page parameter with "color" as the **ID**, "Custom Enumeration" as the **Type**, "Red", "Green" and "Blue" as the **Names** and "#f00", "#0f0", "#00f" as the **Values**.
To access the instance:
$pageParameters.color => com.polarion.alm.shared.api.model.rp.parameter.impl.enumeration.CustomEnumParameterImpl@4a67d91b

To obtain the list of values:
$pageParameters.color.values => [#f00, #0f0, #00f]
Returned list is usable in #foreach loop. List is returned also if the enumeration is not multi-valued.

To obtain the first value (or the sole value for single-value enumeration):
$pageParameters.color.singleValue => #f00

To obtain the list of names:
$pageParameters.color.names => [Red, Green, Blue]
Returned list is usable in #foreach loop. List is returned also if the enumeration is not multi-valued.

To obtain the first name (or the sole name for single-value enumeration):
$pageParameters.color.singleName => Red

6.6.6 Date Parameters

Examples below refer to a page parameter with "From" as the **ID**, "Date" as the **Type** and "March 20th 2015" as the **Absolute Date**.
To access the instance:
`$pageParameters.From` => DateParameterImpl: From, Fri Mar 20 00:00:00 CET 2015

To obtain the value:
`$pageParameters.From.value` => Fri Mar 20 00:00:00 CET 2015

To obtain the value usable in SQL queries:
`$pageParameters.From.toSql` => 2015-03-20

To obtain the value usable in Lucene queries:
`$pageParameters.From.toLucene` => 20150320

6.6.7 Usage in "Lucene + Velocity" and "SQL + Velocity" queries

Method calls `toSql()` and `toLucene()` are automatically added when **Page Parameters** are used in "Lucene + Velocity" and "SQL + Velocity" queries. So given a page parameter with "Year" as the **ID**, you can use the query for all items resolved in that year: `resolvedOn:[${pageParameters.Year}0101 TO ${pageParameters.Year}1231]`

For more information see the Javadoc for methods defined by the `com.polarion.alm.shared.api.model.rp.parameter.ParameterUsableInQuery` interface.

7 Page Script

The Page Script is executed before widgets and can be used to:

- Initialize variables that can then be used by widgets.
- Add page parameters that widget parameters can be bound to.

The Page Script can use the same Velocity context variables as widgets with following exceptions:

- **parameters** is not available.
- **pageContext** is available. (It's the holder for the context data.)
- **factory** is available for creating the scripted page parameters.
- **scriptedPageParameters** is available for adding the scripted page parameters.

The results of executing the Page Script can be displayed using the **Show Results** action in the sidebar toolbar. (Useful for debugging.)
The errors and warnings logged by Velocity can be displayed using the **Show Problems** action in the sidebar menu. (Displayed only if problems are reported.)

7.1 Page Context

Page Context is the holder of the variables prepared by the **Page Script** and then accessed by scripted widgets. It can be modified only by the **Page Script**.

The rendering of widgets runs concurrently in some cases, so the variables placed in the **Page Context** must be thread-safe objects if they are accessed by more than 1 widget on the page.

API objects returned from a transaction like **WorkItem** or **Document** should not be stored to the **Page Context**, because they are only usable inside the given transaction and the rendering of widgets is sometimes executed in different transactions. Instead, store references like **WorkItemReference** that you get from the *WorkItem.getReference()* method (and later in the widget). Use the `$reference.get($transaction)` to get the *WorkItem* again.

7.1.1 Page Context Example

The following **Page Script** is an example of how to prepare common parts of SQL queries that can then be used by multiple widgets to display the same data in different ways:

```
#set($uncoveredReqWhere = $objectFactory.newStringBuilder)
$!uncoveredReqWhere.append("and C_TYPE in $pageParameters.reqtypes.toSql ${esc.n}").null

#if(!$pageParameters.version.values.empty)
    $!uncoveredReqWhere.append("and CUSTOM_FIELD.FK_WORKITEM = WORKITEM.C_PK ${esc.n}").null
    $!uncoveredReqWhere.append("and CUSTOM_FIELD.C_NAME = 'targetVersion' ${esc.n}").null
    $!uncoveredReqWhere.append("and CUSTOM_FIELD.C_STRING_VALUE = '$pageParameters.version.singleValue().id()' ${esc.n}").null
#end

#if(!$pageParameters.status.values.empty)
    $!uncoveredReqWhere.append("and C_STATUS IN $pageParameters.status.toSql ${esc.n}").null
#end

#set($coveredReqWhere = $objectFactory.newStringBuilder)
$!coveredReqWhere.append($uncoveredReqWhere).null

#set($testQuery = $objectFactory.newStringBuilder)
$!testQuery.append("select TEST.C_PK from WORKITEM TEST, STRUCT_WORKITEM_LINKEDWORKITEMS LINK ${esc.n}").null
$!testQuery.append("    where LINK.FK_WORKITEM = WORKITEM.C_PK ${esc.n}").null
$!testQuery.append("    and LINK.FK_P_WORKITEM = TEST.C_PK ${esc.n}").null
$!testQuery.append("    and LINK.C_ROLE = '$pageParameters.verifiesLinkRole.toSql'").null

$!uncoveredReqWhere.append("and not exists ($!testQuery)").null
$!coveredReqWhere.append("and exists ($!testQuery)").null

$!pageContext.put("uncoveredReqWhere", "$uncoveredReqWhere")
$!pageContext.put("coveredReqWhere", "$coveredReqWhere")

#if(!$pageParameters.version.values.empty)
    $!pageContext.put("joinCustomField", ",CF_WORKITEM CUSTOM_FIELD").null
#end

## show the values in the result for debugging
<pre>
uncoveredReqWhere:

$uncoveredReqWhere

coveredReqWhere:
```

```
$coveredReqWhere
</pre>
```

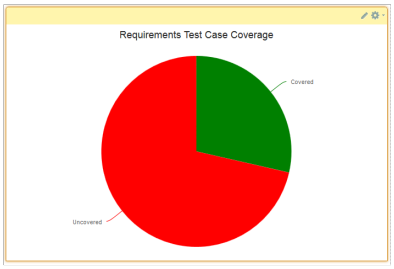
The example was developed for the **Drive Pilot** demo project and uses the following page parameters:

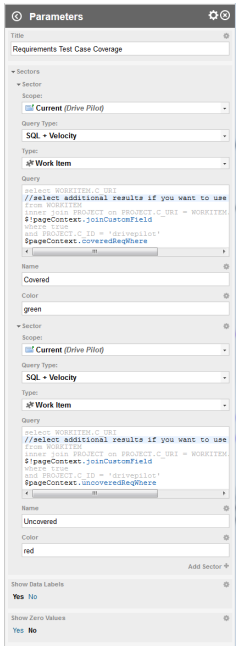
- **reqtypes** - Requirement Item Types - Multi-enumeration parameter for the *workitem-type* enumeration.
- **version** - Version - Enumeration parameter for Plans with query: template.id:release.
- **status** - Status - Enumeration parameter for the *status* enumeration.

The results of this page script:

```
uncoveredReqWhere:

and C_TYPE in ('systemRequirement', 'softwareRequirement', 'mechanicalRequirement', 'electricalRequirement')
and CUSTOM_FIELD.FK_WORKITEM = WORKITEM.C_PK
and CUSTOM_FIELD.C_NAME = 'targetVersion'
and CUSTOM_FIELD.C_STRING_VALUE = 'Version_2_0'
and C_STATUS IN ('approved')
and not exists (select TEST.C_PK from WORKITEM TEST, STRUCT_WORKITEM_LINKEDWORKITEMS LINK
    where LINK.FK_WORKITEM = WORKITEM.C_PK
    and LINK.FK_P_WORKITEM = TEST.C_PK
    and LINK.C_ROLE = 'verifies')

coveredReqWhere:

and C_TYPE in ('systemRequirement', 'softwareRequirement', 'mechanicalRequirement', 'electricalRequirement')
and CUSTOM_FIELD.FK_WORKITEM = WORKITEM.C_PK
and CUSTOM_FIELD.C_NAME = 'targetVersion'
and CUSTOM_FIELD.C_STRING_VALUE = 'Version_2_0'
and C_STATUS IN ('approved')
and exists (select TEST.C_PK from WORKITEM TEST, STRUCT_WORKITEM_LINKEDWORKITEMS LINK
    where LINK.FK_WORKITEM = WORKITEM.C_PK
    and LINK.FK_P_WORKITEM = TEST.C_PK
    and LINK.C_ROLE = 'verifies')
```

An example widget that uses the variables. The parameters of the widget are shown on the right side:





7.2 Scripted Page Parameters

Page Script can add additional page parameters on top of the page parameters defined in the **Page Parameters** sidebar. **Page Script** cannot add a page parameter with the same **ID** as a parameter already defined in the sidebar and an error will be logged in the **Page Script** error log if it tries to.

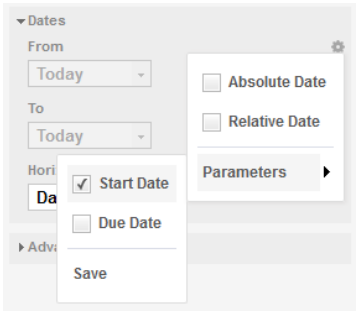
7.2.1 Scripted Page Parameters example

The **Page Script** below shows how to add 2 date type **Page Parameters** with values taken from **Plan Start Date** and **Due Date** fields.
The example is meant to be used in a **Plan** report.

```
#set($startDate = $plan.fields.startDate.getIfCan)
#set($startDateBuilder = $factory.date("Start Date"))
#if ($!startDate)
    $!startDateBuilder.absolute($!startDate)
#end
$!scriptedPageParameters.put("startDate", $startDateBuilder.build())

#set($dueDate = $plan.fields.dueDate.getIfCan)
#set($dueDateBuilder = $factory.date("Due Date"))
#if ($!dueDate)
```

```
    $!dueDateBuilder.absolute($!dueDate)
#end
$!scriptedPageParameters.put("dueDate", $dueDateBuilder.build())
```

These parameters can be then used for any date parameters in widgets, for example in chart widgets as shown below:



8 Dependencies between widget parameters

It is possible to define dependencies between widget parameters. When a report designer changes the value of a widget parameter that is marked as '**dependency source**', any other parameters that are marked as '**dependency target**' can be reconfigured accordingly.

8.1 Example

A widget has two parameters:

- The first is the object selector parameter where a user can select a Work Item.
- The second is an enumeration parameter for selecting link roles.

The second parameter should provide the possibility to select a link role from a Project when *defined in the Work Item.*
A complete implementation of this example is available in the Work Records (Java and Velocity) widget examples .

8.1.1 Java widget

Definition of Parameters:

```
parameters.put(PARAMETER_USER_STORY, factory.objectSelector("User
Story").allowedPrototypes(PrototypeEnum.WorkItem).dependencySource(true).build());
parameters.put(PARAMETER_LINK_ROLE, factory.enumeration("Link Role", "workitem-link-role").dependencyTarget(true).build());
```

For implementing a dependency method:
`com.polarion.alm.shared.api.model.rp.widget.RichPageWidget.processParameterDependencies(RichPageWidgetDependenciesContext)` should be overridden in a widget class:

```
ObjectSelectorParameter parameter = context.parameter(WorkRecordReportWidget.PARAMETER_USER_STORY);
WorkItem workItem = (WorkItem) parameter.value();
Scope scope = (workItem == null) ? null : workItem.getReference().scope();

EnumParameter parameter = context.parameter(WorkRecordReportWidget.PARAMETER_LINK_ROLE);
parameter.set().scope(scope);
```

8.1.2 Velocity widget

Definition of Parameters:

```
$parameters.put("userStory", $factory.objectSelector("User Story").allowedPrototypes(["WorkItem"]).dependencySource(true).build())
$parameters.put("linkRole", $factory.enumeration("Link Role", "workitem-link-role").dependencyTarget(true).build())
```

Dependency implementation is defined in the **`dependencies.vm`** file.

```
#set($userStoryParameter = $parameters.userStory)
#set($linkRoleParameter = $parameters.linkRole)
#set($workItem = $userStoryParameter.value)

#if (!$workItem)
    #set($scope = $null)
#else
    #set($scope = $workItem.getReference().scope())
#end

$linkRoleParameter.set().scope($scope)
```

9 Actions in widgets

It is possible to create a button in a widget that triggers a server side action when clicked. (For example, a button that starts a **Plan**.)

9.1 How it works

A rendered widget contains elements with the following special attribute (`RichPageWidget.ATTRIBUTE_ACTION_ID`) defined.
When a user clicks on it, a request is sent to the server and `RichPageWidget.executeAction(RichPageWidgetActionContext)` is triggered.

A widget developer can handle the event by implementing `executeAction` in the widget.

Note: The standard action handling persists some changes, so its good practice to request a user confirmation up front.
(See `RichPageWidget.ATTRIBUTE_CONFIRM_*` keys.)

A widget developer can also define whether the page should be refreshed after the action executes, or if only the widget will be re-rendered.
This can be done by calling `RichPageWidgetActionContext.refresh(boolean)`.

It's important to understand that there can be several widgets of the same kind on the same page, so identifying exactly what should be done should be based on the `ATTRIBUTE_ACTION_ID` value and the widget parameters.

Action handling is only available for Java Rich Page widgets.
An updatable version of the model objects can be received by `ModelObjectBase.getUpdatable(WriteTransaction)`

9.2 Example

```
private void renderAction(@NotNull HtmlContentBuilder htmlContentBuilder, @NotNull DurationValue total) {
    HtmlTagBuilder a = htmlContentBuilder.tag().a();
    a.append().text("Update time spent in the user story");
    a.attributes().byName(RichPageWidget.ATTRIBUTE_ACTION_ID, total.toString());
    a.attributes().byName(RichPageWidget.ATTRIBUTE_CONFIRM_TITLE, "Update time spent");
    a.attributes().byName(RichPageWidget.ATTRIBUTE_CONFIRM_TEXT, "Do you want to update time spent?");
}

public void executeAction(@NotNull RichPageWidgetActionContext context) {
    ObjectSelectorParameter parameter = context.parameter(PARAMETER_USER_STORY);
    WorkItem workItem = (WorkItem) parameter.value();
    if (workItem == null) {
        return;
    }
    Duration value;
    try {
        value = new Duration(context.actionId());
    } catch (Exception e) {
        return;
    }
    UpdatableWorkItem updatableWorkItem = workItem.getUpdatable(context.transaction());
    updatableWorkItem.fields().timeSpent().set(value);
    updatableWorkItem.save();
    context.refresh(true);
}
```

Complete implementation of this example is available in the Java Work Records widget example.

10 Velocity context reference

10.1 Available everywhere

10.1.1 Polarion objects

- **transaction**
  - com.polarion.alm.shared.api.transaction.ReadOnlyTransaction
- **localization**
  - com.polarion.alm.shared.api.utils.SharedLocalization
- **me**
  - java.lang.String
  - current user ID (shortcut for `$transaction.context.currentUserId`)
- **objectFactory**
  - com.polarion.alm.shared.api.utils.velocity.ObjectFactory

10.1.2 Polarion services (from Polarion Open Java API)

- **trackerService**
  - com.polarion.alm.tracker.ITrackerService
- **projectService**
  - com.polarion.alm.projects.IProjectService
- **securityService**
  - com.polarion.platform.security.ISecurityService
- **platformService**
  - com.polarion.platform.IPlatformService
- **testManagementService**
  - com.polarion.alm.tracker.ITestManagementService

10.1.3 Velocity Generic Tools

- **date**
  - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/DateTool.html
- **math**
  - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/MathTool.html
- **number**
  - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/NumberTool.html
- **esc**
  - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/EscapeTool.html
- **alternator**

    - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/AlternatorTool.html
- **lists**
    - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/ListTool.html
- **sorter**
    - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/SortTool.html
- **mill**
    - http://velocity.apache.org/tools/2.0/apidocs/org/apache/velocity/tools/generic/IteratorTool.html

10.2 Available in Velocity widgets and Page Script

All of the following are also available in "Lucene + Velocity" and "SQL + Velocity" queries.

10.2.1 Velocity Widget Objects

- **renderingContext**
    - com.polarion.alm.shared.api.model.rp.widget.RichPageRenderingContext in Page Script
    - com.polarion.alm.shared.api.model.rp.widget.RichPageWidgetRenderingContext in other cases
- **widgetContext**
    - Synonym for renderingContext.
    - Not available in **Page Script**.
- **pageContext**
    - java.util.Map<Object, Object>
    - Not available in **details.vm** and **parameters.vm**.
- **parameters**
    - com.polarion.alm.shared.api.utils.collections.ReadOnlyStrictMap<String, ? extends com.polarion.alm.shared.api.model.rp.parameter.RichPageParameter>
    - Not available in **details.vm** and **Page Script**.
- **pageParameters**
    - com.polarion.alm.shared.api.utils.collections.ReadOnlyStrictMap<String, ? extends com.polarion.alm.shared.api.model.rp.parameter.RichPageParameter>
    - Not available in **details.vm** and **parameters.vm**.
- **scriptedPageParameters**
    - com.polarion.alm.shared.api.utils.collections.StrictMap<String, ? extends com.polarion.alm.shared.api.model.rp.parameter.RichPageParameter>
    - Available only in **Page Script**.
- **urlParameters**
    - com.polarion.alm.shared.api.utils.collections.ReadOnlyStrictMap<String, String>
    - Not available in **details.vm** and **parameters.vm**.
- **page**
    - current page: com.polarion.alm.shared.api.model.rp.RichPage
    - Available for stand-alone pages. Not available in **details.vm** and **parameters.vm**.
- **plan**
    - current Plan object: com.polarion.alm.shared.api.model.plan.Plan
    - Available only on **Plans** that use Rich Page technology.
- **testRun**
    - current TestRun object: com.polarion.alm.shared.api.model.tr.TestRun
    - Available only on **Test Runs** that use Rich Page technology.
- **object**
    - current object: com.polarion.alm.shared.api.model.ModelObject
    - Reference to a **Plan** or **Page** (dependent on context).
- **factory**
    - com.polarion.alm.shared.api.model.rp.parameter.ParameterFactory
    - Available in **parameters.vm** and **Page Script**.

10.3 Accessing Derived Fields

Derived Field accessors in the Polarion Rendering API are prefixed with an underscore. The Velocity parser cannot deal with such methods, but will accept the method name without a leading underscore or parentheses, i.e., `$transaction.userGroups.getBy.id('someId').fields.users.size` works while `$transaction.userGroups.getBy.id('someId').fields._users.size` does not.

11 Examples that use ParameterFactory in Velocity

11.1 Adding scripted page parameters in Page Script

```
$!scriptedPageParameters.put("myString", $factory.string("My String").value("my value").build())
$!scriptedPageParameters.put("myInteger", $factory.integer("My Integer").value(3).build())
$!scriptedPageParameters.put("myBoolean", $factory.bool("My Boolean").value(true).build())
$!scriptedPageParameters.put("mySingleSeverity", $factory.enumeration("My Single Severity", "severity").singleValue("major").build())
$!scriptedPageParameters.put("myMultiSeverity", $factory.enumeration("My Multi Severity", "severity").allowMultipleValues(true).values(["major", "critical"]).build())
$!scriptedPageParameters.put("myRelativeDate", $factory.date("My Relative Date").relative(-12).build())
$!scriptedPageParameters.put("myAbsoluteDate", $factory.date("My Absolute Date").absolute($date.toDate("yyyy-MM-dd", "2015-01-02")).build())
$!scriptedPageParameters.put("myPlan", $factory.objectSelector("My Plan").allowedPrototypes(["Plan"]).objectPath("drivepilot/Version_1_0").build())
```

The other parameter types cannot be used as page parameters.

11.2 Adding widget parameters in parameters.vm

All of the above mentioned examples can be used. Just replace *$!scriptedPageParameters* with *$!parameters*

Additional parameter type examples:

$!parameters.put("myScript", $factory.script("My Script").value("${esc.d}me").build())
$!parameters.put("myTimeAxis", $factory.timeAxis("My Time Axis").fromRelative(-28).toRelative(0).scaleWeek().build())
$!parameters.put("myFields", $factory.fields("My Fields").fields($mySetOfFields).build())
$!parameters.put("myQuery", $factory.luceneQuery("My Query").scope().global().prototype().workItem().subtype("defect").value("HAS_VALUE:resolution").build())

Example combining *DataSetParameter* with *FieldsParameter* and *SortingParameter*. (The way they are used in the Table widget.)

#set($myColumns = $factory.fields("My Columns").build())
#set($mySorting = $factory.sorting("My Sorting").build())
$!parameters.put("myDataSet", $factory.dataSet("My Data Set").add("myColumns", $myColumns).add("mySorting", $mySorting).build())

Example of a custom enumeration parameter:

$parameters.put("teams", $factory.customEnum("Teams").addEnumItem("mainDev","Main Development Team").addEnumItem("mainQa","Main QA Team").addEnumItem("extDev", "External Dev Team").addEnumItem("extQa","External QA Team").allowMultipleValues(true).values(["mainDev","mainQa"]).build())

12 Predefined Velocity macros

There is a set of predefined Velocity macros that can be used in Script widgets.

Plain Text box

```
#message("some plain text")
```

> some plain text

Information Message box

```
#info("something")
```

> Info: something

Warning Message box

```
#warning("something")
```

> Warning: something

Error Message box

```
#error("something")
```

> Error: something

Load CSS

```
#loadCss("http://mywebsite.com/myCSS.css")
```

Load CSS from Widget Resources

```
#loadWidgetCss("resources/myCSS.css")
```

Load Java Script File

```
#loadJs("http://mywebsite.com/myJS.js")
```

Load Java Script File from Widget Resources

```
#loadWidgetJs("resources/myJS.js")
```

13 Enable the Work Item Properties sidebar in a custom widget

To enable the **Work Item Properties** sidebar:

1. Set up the initial java script configuration to associate part or all of the widget with the **Work Item Properties** sidebar. This should be done at the beginning of the rendering process and is accomplished by using `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguration`
2. Set up the configuration to have the **Work Item Properties** sidebar open *on click* during the rendering of each individual item. This is done by adding html attributes to the elements where the item is rendered by using `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguredContainer`

An example is provided in the Polarion SDK examples under `com.polarion.example.widget`.

13.1 The HTML attributes configured on a Work Item so that the Work Items sidebar is displayed

- Path to the item (for Work Items: project id/work item id):
  `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguredContainer.Attributes.referencePath()`

- Css class for styling before clicking:
  `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguredContainer.Attributes.clickableCssClass()`
- Css class for highlighting the item once the sidebar is open:
  `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguredContainer.Attributes.highlightingCssClass()`
- (Optional) Attribute for associating the list of fields displayed in the sidebar for a group of Work Items:
  `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguredContainer.Attributes.fieldsConfigurationKey().`
  See *"13.2 Showing Different Fields for Different Groups of Work Items"*. for more details.

They can be accessed from `com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguredContainer.attributes();`

13.2 Show different fields for different groups of Work Items

This can be done by providing an initial configuration using
`com.polarion.alm.shared.api.model.rp.widget.PropertiesSidebarConfiguration.fieldsConfiguration(Map<String, List<String>>).`

Example:

For the Work Items Board Widget (that has a structure with **Swimlanes** that contain **User Stories** and **Cards** that contain **Implementation** tasks or **Issues**), you might want to show a different set of fields in the sidebar:

For example:

- Status, severity and priority for **Swimlane** elements.
- Status and author for **Card** elements.

To do so, during the initial configuration, you would have:

```
...
 Map<String, List<String>> configurationMap = new HashMap<>();
 List<String> swimlaneFields = Arrays.asList(new String[] { "status", "severity", "priority" });
 configurationMap.put("swimlane", swimlaneFields);
 List<String> cardFields = Arrays.asList(new String[] { "status", "author" });
 configurationMap.put("card", cardFields);
context.propertiesSidebarConfiguration().fieldsConfiguration(configurationMap).addTo(builder);
..
```

And later during the rendering of **Cards**:

```
...
configuredContainer.openSidebarOnClick(tr).reference(wiReference).fieldsConfigurationKey("card"); << When clicking on this item, the sidebar will display "status", "author".
...
```

Or during the rendering of **Swimlanes**:

```
...
configuredContainer.openSidebarOnClick(tr).reference(wiReference).fieldsConfigurationKey("swimlane"); << When clicking on this item, the sidebar will display "status", "severity", "priority".
```

Note: The different field lists can be read from different widget field parameters.

14 Customize the Kanban Board

A **Kanban Board is** made up of Work Item **Cards** grouped by status columns.

You can customize the appearance of the **Cards** by adding custom velocity snippets. You can change the color of the card, remove the status border color, show different Work Item fields, re-arrange the fields already rendered and *more*. The custom Velocity scripts will be rendederd instead of the default Java scripts.
Default Velocity snippets are included with the widget under the **Board Script** and **Card Script** parameters and are a great starting point.

(The sample scripts mimic the java rendered output out of the box.)

First, set the **Enable Customization** widget parameter to `"Yes".` The **Board Script** and **Card Script** parameters appear below.

The **Board Script** is inserted at the beginning of the board and should always be used to add custom css styles and/or Java script common for all your **Cards**.

(This avoids having to duplicate them for each **Card**.)

> Warning: The styles inserted in this script are applied to the whole page.

The **Card Script** defines the content that appears within the cards. The **Card Script** Velocity script option allows for the following additional objects:

- **$workItem** - The Work Item rendered in the card.
- **$columnIndex** - The column that the card is rendered on.

> Info: The Work Item Properties sidebar and the ability to drag and drop are still available when customizations are enabled.

You can always revert to using the internal Java rendering by setting **Enable Customization** to `"No".`

Here are a few examples on how to customize the look and content of your Kanban board's **Cards**:

14.1 Examples

14.1.1 How to change the card color

1. Open a page containing a Kanban board in edit mode.

2. Open the **Kanban Board: Parameters** sidebar.

3. Click on **Advanced** at the bottom and click **Yes** under **Enable Customizations**.

4. Click on **Board Script**. (For a full screen view of the script hit **F11**. Hit **Esc** to exit full screen view.)

5. Change the background color of the card by adding `"background-color: #d0d2e0;"` to `".kanbanWorkItemDiv"`.

6. Change the gradient of a long title by changing the `"background"` property in `".kanbanWorkItemTitleGradientDiv"` to `"background: linear-gradient(to bottom, rgba(255, 255, 255, 0), #ccffff) repeat scroll 0 0 rgba(0, 0, 0, 0);"`

7. Enable the **Work Item Properties Sidebar** by adding:

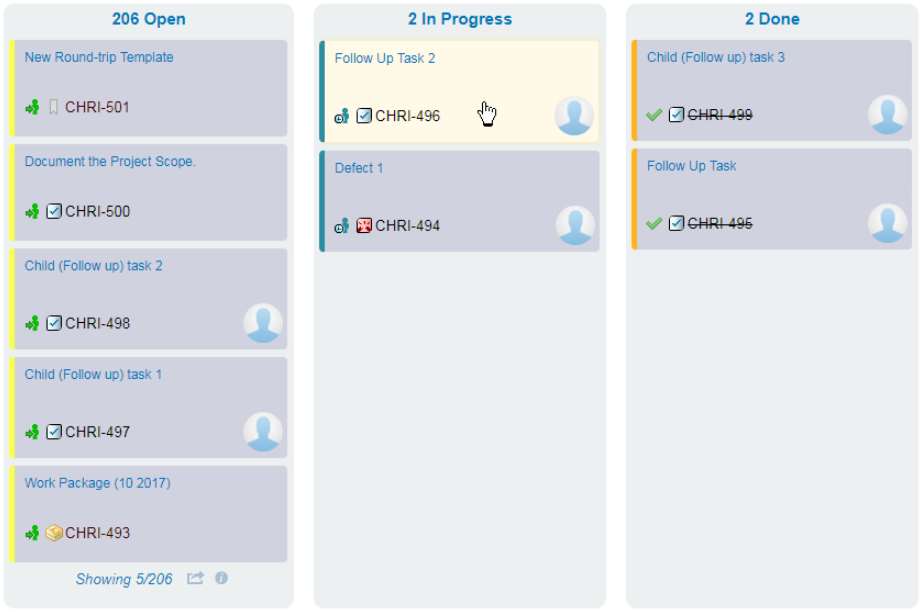
```
.polarion-sidebar-highlight .kanbanWorkItemDiv {
 background-color: #fff9e8;
 }
```

The only changes to the default "`Board Script`" are the **bold** lines in the example below:

```
...
.kanbanWorkItemDiv {
  padding: 7px 3px 3px 7px;
  box-sizing: border-box;
  border-left: 5px solid;
  border-right: 1px solid #EEEEEE;
  border-top: 1px solid #EEEEEE;
  border-bottom: 1px solid #EEEEEE;
  border-radius: 5px;
  position: relative;
  background-color: #d0d2e0;
}
...
.kanbanWorkItemTitleGradientDiv {
  bottom: 0px;
  height: 12px;
  left: 0;
  position: absolute;
  width: 100%;
  display: inline-block;
  background: linear-gradient(to bottom, rgba(255, 255, 255, 0), #d0d2e0) repeat scroll 0 0 rgba(0, 0, 0, 0);
  filter: progid:DXImageTransform.Microsoft.gradient(startColorstr='#00FFFFFF', endColorstr='#FFFFFFFF', GradientType=0);
 }
...

.polarion-sidebar-highlight .kanbanWorkItemDiv {
    background-color: #fff9e8;
}

...
```

8. Click **Apply**.

The color of the cards will change to the following:



(No changes are required to the default **Card Script**.)

14.1.2 How to add a Work Item's Tree Structure link

You can customize what content appears within the cards in **Card Script**.

1. Open a page containing a Kanban board in edit mode.

2. Open the **Kanban Board: Parameters** sidebar.

3. Click on **Advanced** at the bottom and click **Yes** under **Enable Customizations**.

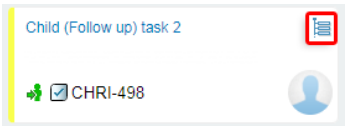
4. Click on **Card Script**. (For a full screen view of the script hit **F11**. Hit **Esc** to exit full screen view.)

5. Paste the **bold** table text below under the `<div class="kanbanWorkItemDiv" style="border-left-color: $wiStatusColor;">` tag.

6. Click **Apply**.

```
...
<div class="kanbanWorkItemDiv" style="border-left-color: $wiStatusColor;">

 <table width="100%">
 <tbody>
  <tr>
   <td  class="kanbanWorkItemTitleDiv">
    <div class="kanbanWorkItemTitleGradientDiv"></div>
    <a href="$wiLink" target="_top">$workItem.fields().title().render()</a>
   </td>
   #set($treeLink =
$transaction.context().createPortalLink().project("$wiProjectId").workItems().query("id:$workItem.getReference().id()").toEncodedRelativeUrl())
   #set($suffix = '&tree_depth=2&tab=tree')
   <td style="vertical-align: top;">
    <a href="$treeLink$suffix"  target="_blank">
     <img  style="float: right;"  title="View Item in a Tree Structure" src="/polarion/ria/images/details/treetable.gif"/>
    </a>
   </td>
   </tr>
       </tbody>
</table>

 <table class="kanbanWorkItemTable">
...
```

6. A tree icon with a link to the card's Work Item in **Tree** view is added to the card.



15 Customize content for Quick Create dialog

The **Quick Create** dialog is available via a **Create...** button added to Polarion's Navigation Header. Its content comes preconfigured, but it is possible to customize the artifacts your users can create with it.

The default configuration includes most of the features currently available for this dialog and can be used as an initial template to create your own configuration.

You can download the default configuration page archive from Polarion Help (Configure Quick Create) and customize it based on your needs. Import the customized page to Polarion and define the path to it as the source for the **Quick Create** dialog.

The **Quick Create** dialog is not available for the Project Group, Baseline, or Administration contexts.

Here we will introduce to the features we included in the default **Quick Create** configuration. You can also find a detailed description of the used APIs in our JavaDoc: `JsActionsBuilder`.
`[POLARION_HOME]/polarion/sdk/doc/javadoc-rendering/com/polarion/alm/shared/api/utils/js/JsActionsBuilder.html`

All actions available in **Quick Create** can be divided into context-independent and context-dependent actions.

15.1 Context-independent actions

These actions are available in the **Quick Create** dialog no matter where users are in the Polarion UI.

All actions are divided into three columns: **Work Item**, **Document** and **Other**.

15.1.1 Work Item

The **Work Item** column contains list of all **Work Item Types** configured for the current project.

Each line is clickable, with a tooltip containing the **Work Item Type Description** (taken from **Administration** - **Work Items** - **Types**), or just the **Work Item Type** label (if the **Description** is not defined), which is visible on mouse over. The current configuration allows you to disable actions which are denied to users based on permissions.

A new **Work Item** form for the appropriate type is opened when users click on the item. (The **Quick Create** dialog is then closed.)

In the project context, users can create **Work Items** of every configured **Work Item** Type in the project defined by `jsBuilder.createWorkItem()` with the type sent on input:

```
<div class="polarion-quick-create-column">
#foreach($type in $types)
 #if($transaction.workItems().can().inScope().forProject($projectId).createWorkItemOfType($type.id))
   #set($itemDisabled="")
   #set($tooltip = "" )
 #else
   #set($itemDisabled="polarion-quick-create-column-item-disabled")
   #set($tooltip=$loc.getString('js.actions.createWorkItem.no.permission'))
 #end
 <div class="polarion-quick-create-column-item $itemDisabled" onclick='$jsBuilder.createWorkItem().type($type.id)'>
   $type.render().withCustomTooltip($tooltip)
 </div>
 #end
</div>
```

In the Global context, users can call the **Create Work Item** dialog where a **Project** and **Type** must be selected to proceed with creation.

The same `jsBuilder.createWorkItem()` is used without a type specification on input:

```
<div class="polarion-quick-create-column-item" onclick='$jsBuilder.createWorkItem()'>

   <img class="polarion-quick-create-column-item-img" src="/polarion/icons/default/actions/new_workitem.gif">
        $loc.getString('navigation.quickCreateDialogCreateWorkItem')

</div>
```

15.1.2 Document

The **Document** column in the project context contains:

- Import Word, invoked by calling `jsBuilder.importWordDocument()`
- RegIF Document, invoked by calling `jsBuilder.importReqIfDocument()`
- Create a **Document** of each configured **Document Type** in the current project, invoked by calling `jsBuilder.createDocument()` with the type sent on input.

Each line is clickable, with a tooltip with the **Document Type's Description** (taken from **Administration** - **Documents & Pages** - **Document Types**), or just with a **Document Type** label (if the **Description** is not defined), which is visible on mouse over. The current configuration allows you to disable actions denied to users based on permissions.

The **Create New Document** dialog is opened by clicking on the line with a required **Document Type**, where this **Document Type** is predefined, as well as a **Space** if users are in a **Space** context in Polarion at the time they invoke the **Create Document** action via **Quick Create** dialog:

```
<div class="polarion-quick-create-column">
       #if(!$transaction.documents().can().inScope().forProject($projectId).create())
           #set($itemDisabled="polarion-quick-create-column-item-disabled")
           #set($tooltip=$loc.getString('js.actions.createDocument.no.permission'))
       #end
       <div class="polarion-quick-create-column-item $itemDisabled"
       onclick='$jsBuilder.importWordDocument()' title="$tooltip">
          <img class="polarion-quick-create-column-item-img"
src="/polarion/ria/images/dle/operations/actionMsWordRoundtrip16.svg">
          $loc.getString('navigation.quickCreateDialogCustomImportWord')
       </div>
       <div class="polarion-quick-create-column-item $itemDisabled"
       onclick='$jsBuilder.importReqIfDocument()' title="$tooltip">
           <img class="polarion-quick-create-column-item-img" src="/polarion/ria/images/control/Reqif.png">
           $loc.getString('navigation.quickCreateDialogCustomImportReqIF')
       </div>
       <div class="polarion-quick-create-column-item-separator"></div>
           #foreach($type in $types)
               <div class="polarion-quick-create-column-item polarion-ellipsis $itemDisabled"
               onclick='$jsBuilder.createDocument().type($type.id)'>
                  $type.render().withCustomTooltip("$tooltip")
               </div>
           #end
</div>
```

In the Global context, users cannot work with **Documents**. (That's why the default configuration just contains a message rendered to users with this information.):

```
<div style="display: block; padding: 6px;">$loc.getString('navigation.quickCreateDialogDocumentsCannotCreate')</div>
```

15.1.3 Other

For both the Global and Project contexts, users can create Rich Pages and Spaces.

Separate actions are rendered for **LiveReport** and **Info Pages.**

`jsBuilder.createRichPage()` is used to create a **LiveReport Page**.

`jsBuilder.createRichPage().liveReport(false)` is used to create an **Info Page**.

The difference is just what page template is used. (The same principle we have for the **Create New Artifact** dialog available from **Space Index Page**.)

`jsBuilder.createSpace()` is used to create a **Space**.

For all actions the current user context is taken into consideration:
If a user is in any **Space** context in Polarion, it is used as the target **Space** for a new Rich Page or as the Parent **Space** for a new **Space**.

```
<div class="polarion-quick-create-column-item $itemDisabledPage"
onclick='$jsBuilder.createRichPage()' title="$tooltipReport">
  <img class="polarion-quick-create-column-item-img" src="/polarion/ria/images/topicIconsSmallDark/richpage.png">
  $loc.getString('dialog.createNewArtifact.newReport.title')
</div>
<div class="polarion-quick-create-column-item $itemDisabledPage"
onclick='$jsBuilder.createRichPage().liveReport(false)' title="$tooltipInfo">
  <img class="polarion-quick-create-column-item-img" src='/polarion/ria/images/topicIconsSmallDark/richpage.png'>
  $loc.getString('dialog.createNewArtifact.newTextPage.title')
</div>
<div class="polarion-quick-create-column-item $itemDisabledPage"
onclick='$jsBuilder.createSpace()' title="$tooltip">
  <img class="polarion-quick-create-column-item-img" src="/polarion/ria/images/topicIconsSmallDark/space.png">
  $loc.getString('definition.space')
```

```
</div>
```

If users are in the Project context, the creation of **Plans**, **Test Runs** and **Collections** are available by using: `jsBuilder.createPlan()`, `jsBuilder.createTestRun()`, `jsBuilder.createCollection()` accordingly:

```
<div class="polarion-quick-create-column-item-separator"></div>
  #set($tooltipPlan="")
  #if(!$transaction.plans().can().inScope().forProject($projectId).create())
     #set($itemDisabledPlan="polarion-quick-create-column-item-disabled")
     #set($tooltipPlan=$loc.getString('js.actions.createPlan.no.permission'))
  #end
  <div class="polarion-quick-create-column-item $itemDisabledPlan" onclick='$jsBuilder.createPlan()'
title="$tooltipPlan">
     <i class="fa fa-bar-chart-o polarion-font-icon-16 polarion-quick-create-column-item-img" ></i>
     $loc.getString('definition.plan')
  </div>

  #set($tooltipTr="")
  #if(!$transaction.testRuns().can().inScope().forProject($projectId).create())
     #set($itemDisabledTr="polarion-quick-create-column-item-disabled")
     #set($tooltipTr=$loc.getString('js.actions.createTestRun.no.permission'))
  #end
  <div class="polarion-quick-create-column-item $itemDisabledTr" onclick='$jsBuilder.createTestRun()'
title="$tooltipTr">
     <img class="polarion-quick-create-column-item-img" src="/polarion/ria/images/topicIconsSmallDark/testruns.svg">
     $loc.getString('definition.testRun')
  </div>

  #set($tooltipColl="")
  #if(!$transaction.baselineCollections().can().inScope().forProject($projectId).create())
     #set($itemDisabledColl="polarion-quick-create-column-item-disabled")
     #set($tooltipColl=$loc.getString('js.actions.createCollection.no.permission'))
  #end
  <div class="polarion-quick-create-column-item $itemDisabledColl" onclick='$jsBuilder.createCollection()'
title="$tooltipColl">
     <img class="polarion-quick-create-column-item-img"
src="/polarion/ria/images/topicIconsSmallDark/collectionsTopic.svg">
     $loc.getString('definition.collection')
  </div>
  #end
</div>
```

15.2 Context-dependent actions

You can provide the current user's context to velocity code using: `$renderingContext.getDisplayedReference()`.

Based on this information, it is possible to render actions in **Quick Create** only if the user is defined in the Polarion context.

In the default configuration we have examples for actions available when a user selects a **Work Item** or is in the **LiveDoc Document** context.

15.2.1 For Selected Item

For the "**Selected Work Item**" condition, the following actions are available: **Create Sibling Work Item**, **Create Child Work Item**.

Your users can create a sibling or child **Work Item** for the selected **Work Item** when working in the following places:

- When they select a single **Work Item** that belongs to a **Document** from **Table**/ **Tree** views in the **Work Item Tracker**.
- When they select a single **Work Item** in **Table**/**Tree** view while working in a **Document**.
- If they are working with a **Document Work Item** in the **Work Item form (editor)**.
- If the selected **Work Item** is not a **Referenced Work Item**.
  (The current API creates a linked **Work Item** only in its primary **Document**. The action is disabled for **Referenced Work Items**.)
- If the **Document** is not a historical revision.

```
#if($selectedWorkItems.size() == 1)
    #set($selectedWorkItemReference=$selectedWorkItems.iterator().next())
#elseif($displayedReference && $displayedReference.prototype().name()=="WorkItem")
    #set($selectedWorkItemReference=$displayedReference)
    #set($isFullscreenWorkItem=true)
#end

#if($selectedWorkItemReference)
    #set($selectedWorkItem=$selectedWorkItemReference.get($transaction))
    #set($document=$selectedWorkItem.fields().module().getIfCan())
    #set($isHistoricalRevision=$displayedReference.requestedRevision())
    #set($elementIsInternal=$selectedObject.items().isInternal($selectedWorkItem))

#if($document && ("home"!=$renderingContext.urlParameters().get("tab") || $isFullscreenWorkItem) && !$isHistoricalRevision &&
($elementIsInternal==true || "!elementIsInternal"==""))
    #set($documentAllowedTypes=$document.fields().allowedWITypes().toArrayList())
```

If all conditions are met, users see the following context actions under a separate "**For Selected Item**" heading:

Create Sibling Work Item and **Create Child Work Item**.

As a result, items are created as linked items to the selected **Work Item** in the same **Document** context.

Users can choose what type the result the **Work Item** has. In the default configuration example there is different rendering logic:

- If just a single **Work Item Type** is defined in **Document Work Item presentation**, the context action button covers an entire line.

- If up to four **Work Type Types** are defined in **Document Work Item presentation**, then small buttons with the **Work Item** type icons and extended tooltip are rendered.

- The default configuration does not render more than the first four **Work Item** types defined in **Document Work Item presentation**.

You can update the configuration as required. (That's only an example.)

```
  <div class="polarion-quick-create-separator-with-icons">
      <img src="/polarion/ria/images/actions/slibing_wi.gif">Sibling
  </div>
```

```
 #if($documentAllowedTypes.size() > 1)
           #foreach($type in $documentAllowedTypes)
               #if($transaction.workItems().can().inScope().forProject($projectId).createWorkItemOfType($type.id))
                   #set($itemDisabled="")
                   #set($tooltip="")
                   #set($customTooltip=" ${type.label} (in this Document) - $type.fields.description.getIfCan() ")
               #else
                   #set($itemDisabled="polarion-quick-create-column-item-disabled")
                   #set($tooltip=$loc.getString('js.actions.createWorkItem.no.permission'))
                   #set($customTooltip=$loc.getString('js.actions.createWorkItem.no.permission'))
               #end
               #if($velocityCount < 5)
                   <div class="polarion-quick-create-context-aware-buttons $itemDisabled"
                       onclick='$jsBuilder.createLinkedWorkItem().type($type.id).linkedType("sibling")'>
                       $type.render().withText(false).withCustomTooltip($customTooltip)
                   </div>
               #end
           #end
       #elseif($documentAllowedTypes.size() == 1)
           #foreach($type in $documentAllowedTypes)
               #if($transaction.workItems().can().inScope().forProject($projectId).createWorkItemOfType($type.id))
                   #set($itemDisabled="")
                   #set($tooltip="")
               #else
                   #set($itemDisabled="polarion-quick-create-column-item-disabled")
                   #set($tooltip=$loc.getString('js.actions.createWorkItem.no.permission'))
               #end
               <div class="polarion-quick-create-context-aware-button $itemDisabled"
               onclick='$jsBuilder.createLinkedWorkItem().type($type.id).linkedType("sibling")'>
                   $type.render().withCustomTooltip($tooltip)
               </div>
           #end
       #end

       <div class="polarion-quick-create-separator-with-icons">
           <img src="/polarion/ria/images/actions/linking_wi.gif">Child
       </div>
       ... the same piece of velocity code as the "Sibling" above
```

15.2.2 For this Document

If users are in any **Document** context (no matter what view is used: **Document Editor**, **Table** or **Tree**), additional actions are available in the **Quick Create** dialog under a separate "**For this Document**" heading:

- **Create Document Baseline** for the current Document (by `jsBuilder.createBaseline()`)
- **Reuse current Document** (by `jsBuilder.duplicate()`)
- **Branch current Document** (by `jsBuilder.branchDocument()`)

Check for current user's context is: `#if($renderingContext.getDisplayedReference().prototype().name()=="Document").`

```
<div class="polarion-quick-create-context-aware-button $itemDisabled"
     onclick='$jsBuilder.createBaseline()' title="$tooltip">
         <img class="polarion-quick-create-context-document-img" src="/polarion/ria/images/dle/create_baseline.png">
         $loc.getString('navigation.quickCreateDialogCustomDocumentBaseline')
</div>
<div class="polarion-quick-create-context-aware-button $itemDisabled"
   onclick='$jsBuilder.duplicate()' title="$tooltip">
       <img class="polarion-quick-create-context-document-img" src="/polarion/ria/images/dle/operations/reuse.png">
        $loc.getString('navigation.quickCreateDialogCustomDocumentReuse')
</div>
<div class="polarion-quick-create-context-aware-button $itemDisabled"
    onclick='$jsBuilder.branchDocument()' title="$tooltip">
       <img src="/polarion/ria/images/dle/operations/branch.png">
        $loc.getString('navigation.quickCreateDialogCustomDocumentBranch')
</div>
```

16 F.A.Q

Q: How do I get the **ID** of the current project?

A: In Velocity: `$page.reference.projectId`

Q: What to do when HTML table exported to PDF loses styles and/or there are no columns?

A: PDF export requires that HTML tags <tr>, <th> and <td> are closed with </tr>, </th> and </td>.