[**anthemopensource**](https://anthemopensource.atlassian.net/wiki/display/~admin)

Technical Documentation

* [Introduction](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation" \l "TechnicalDocumentation-Introduction)
* [Config Annotations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConfigAnnotations)
* [View Config Annotations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewConfigAnnotations)
  + [Accordion](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Accordion)
  + [AccordionGroup](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-AccordionGroup)
  + [Button](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Button)
  + [ButtonGroup](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ButtonGroup)
  + [Calendar](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Calendar)
  + [CardDetail](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetail)
    - [CardDetail Children Components](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetailChildrenComponents)
    - [CardDetail.Body](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetail.Body)
    - [CardDetail.Footer](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetail.Footer)
    - [CardDetail.Header](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetail.Header)
    - [CardDetail.Tag](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetail.Tag)
  + [CardDetailsGrid](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CardDetailsGrid)
  + [CheckBox](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CheckBox)
  + [CheckBoxGroup](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CheckBoxGroup)
  + [ComboBox](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ComboBox)
  + [FieldValue](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-FieldValue)
  + [FileUpload](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-FileUpload)
  + [FilterButton](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-FilterButton)
  + [Form](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Form)
  + [GlobalHeader](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-GlobalHeader)
  + [GlobalFooter](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-GlobalFooter)
  + [GlobalNavMenu](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-GlobalNavMenu)
  + [Grid](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Grid)
    - [Controlling column data with @GridColumn](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Controllingcolumndatawith@GridColumn)
    - [Controlling row data with @GridRowBody](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Controllingrowdatawith@GridRowBody)
  + [Hints](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Hints)
  + [Image](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Image)
  + [InPlaceEdit](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-InPlaceEdit)
  + [InputDate](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-InputDate)
  + [Label](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Label)
  + [Link](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Link)
  + [LinkMenu](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-LinkMenu)
  + [Menu](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Menu)
  + [Modal](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Modal)
    - [Setting @Modal to be displayed by default](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Setting@Modaltobedisplayedbydefault)
  + [MultiGrid](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-MultiGrid)
  + [MultiSelect](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-MultiSelect)
  + [MultiSelectCard](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-MultiSelectCard)
  + [Page](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Page)
  + [PageFooter](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-PageFooter)
  + [PageHeader](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-PageHeader)
  + [PickList](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-PickList)
  + [Radio](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Radio)
  + [Section](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Section)
  + [StaticText](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-StaticText)
  + [SubHeader](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-SubHeader)
  + [TextArea](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-TextArea)
  + [TextBox](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-TextBox)
  + [Tile](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Tile)
  + [ViewRoot](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewRoot)
* [Core Config Annotations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CoreConfigAnnotations)
  + [ConceptId](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConceptId)
  + [Config](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Config)
    - [Configs](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Configs)
  + [ConfigNature](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConfigNature)
    - [Ignore](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Ignore)
  + [Domain](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Domain)
  + [DomainMeta](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-DomainMeta)
  + [Execution](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Execution)
  + [Initialize](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Initialize)
  + [MapsTo](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-MapsTo)
    - [Path](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Path)
    - [Type](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Type)
  + [Mode](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Mode)
  + [Model](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Model)
    - [Param](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Param)
    - [Text](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Text)
  + [ParamContext](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ParamContext)
  + [SearchNature](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-SearchNature)
    - [StartsWith](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-StartsWith)
  + [Repo](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Repo)
    - [rep\_mongodb](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-rep_mongodb)
    - [rep\_ws](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-rep_ws)
  + [Rule](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Rule)
  + [Values](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Values)
    - [Source](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Source)
    - [Examples](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Examples)
  + [ViewParamBehavior Component](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewParamBehaviorComponent)
  + [ViewStyle Component](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewStyleComponent)
* [Conditional Config Annotations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConditionalConfigAnnotations)
  + [State Event Handlers](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-StateEventHandlers)
  + [Framework Defined Conditional Annotations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-FrameworkDefinedConditionalAnnotations)
    - [AccessConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-AccessConditional)
    - [AccessConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-AccessConditionals)
    - [ActivateConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ActivateConditional)
    - [ActivateConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ActivateConditionals)
    - [ConfigConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConfigConditional)
    - [EnableConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-EnableConditional)
    - [EnableConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-EnableConditionals)
    - [ExpressionConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ExpressionConditional)
    - [ExpressionConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ExpressionConditionals)
    - [ValidateConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ValidateConditional)
    - [ValidateConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ValidateConditionals)
    - [ValuesConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ValuesConditional)
    - [ValuesConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ValuesConditionals)
* [View Components](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewComponents)
  + [Tooltip](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Tooltip)
* [User Interface Validation](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-UserInterfaceValidation)
* [UI Validation](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-UIValidation)
  + [JSR Annotations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-JSRAnnotations)
  + [Static Validations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-StaticValidations)
  + [Conditional Validations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConditionalValidations)
  + [Special Considerations](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-SpecialConsiderations)
  + [Custom Messages](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CustomMessages)
* [Layout Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-LayoutConfiguration)
* [Collection Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CollectionConfiguration)
* [Rule Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-RuleConfiguration)
* [Audit Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-AuditConfiguration)
* [Process Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ProcessConfiguration)
  + [Business Process Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-BusinessProcessConfiguration)
  + [Entity Lifecyle Management using BPM](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-EntityLifecyleManagementusingBPM)
  + [Associating a workflow with an entity](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Associatingaworkflowwithanentity)
  + [Stateless Business Function implementation using BPM](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-StatelessBusinessFunctionimplementationusingBPM)
  + [Function Handlers](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-FunctionHandlers)
  + [Predefined Function Handlers](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-PredefinedFunctionHandlers)
  + [Custom Function Handlers](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-CustomFunctionHandlers)
    - [Nimbus Search Filtering](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-NimbusSearchFiltering)
      * [Usage Examples](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-UsageExamples)
      * [Criteria](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Criteria)
      * [Projection](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Projection)
* [View Configuration](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewConfiguration)
  + [View Patterns](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewPatterns)
    - [Conceptual View](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ConceptualView)
    - [View Parameters](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewParameters)
    - [View Form](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewForm)
    - [View Grid](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewGrid)
    - [View Section](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewSection)
    - [View Card](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewCard)
    - [View Global Page](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewGlobalPage)
    - [View Work Page](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewWorkPage)
    - [View Tile](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-ViewTile)
* [Infrastructure](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-Infrastructure)

Technical documents regarding engineering and design of framework

**More to Come**

Please keep checking this section frequently for this space for more information.

**Introduction**

The following sections outlines the different options available while configuring an application.

**Config Annotations**

**View Config Annotations**

Table 1. View Components

| **Component** | **Description** | **Attributes** | **Notes** |
| --- | --- | --- | --- |
| Page | Used to create a view page | Type - Home,Details,Form,Static | Default value will be Home. |
| Tile | Used to create a view Tile | Size - XSmall,Small,Medium,Large | Default value will be Large. |
| Section | Used to create a view Section. A section is encapsulated in a tile | Size - XSmall,Small,Medium,Large | Default value will be Large. |
| Form | Used to create a view Form to capture user responses. A forn is encapsulated in a section |  |  |

**Accordion**

Table 2. Accordion Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Accordion | cannot be changed |
| cssClass | String |  | css class override where default is 'panel-default' |

Accordion.java

|  |
| --- |
| @Accordion  private VASectionA\_Questionnaire vaSectionA\_Questionnaire; |

Accordion is supported only in a form

**AccordionGroup**

Table 3. AccordionGroup Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | AccordionGroup | cannot be changed |
| cssClass | String | panel-default | css class override where default is 'panel-default' |

**Button**

Table 4. Button Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Button | cannot be changed |
| url | String |  | source url of the button element |
| b | String | $execute | b |
| method | String | GET | HTTP method that will be executed |
| imgSrc | String |  | source path/ url of the image |
| type | enum | Type.Plain | type of button |
| payload | String |  | payload |
| cssClass | String | btn btn-primary | class used to style the section |
| formReset | boolean | true | reset the form |

Styles on Button component is defined using Styles enum, rather than cssClass. The possible enum values for size are **PRIMARY, SECONDARY, DESTRUCTIVE, PLAIN (default)**. If a user wants an image button, then use imgSrc attribute with "Font-Awesome" image class name. In this case, user need not specify Type. The default Type (PLAIN) will be used.+ *Please read @Config for information regarding @Config annotation.*

Button.java

|  |
| --- |
| @Config(url="/pageAddEditGoal/tileEditGoal/sectionEditGoal/goalDetailsForm/\_nav?pageId=pageCarePlanSummary")  @Button(type = Button.Type.DESTRUCTIVE)  private String delete; |

**ButtonGroup**

Table 5. ButtonGroup Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | ButtonGroup | cannot be changed |
| cssClass | String | text-sm-center | class used to style the section |

CSS styles supported for ButtonGroup component are **inline, right, left, center (default)**.

ButtonGroup.java

|  |
| --- |
| @ButtonGroup(cssClass="text-sm-right")  private FormGoalsButtonGroup formGoalsButtonGroup; |

**Calendar**

@Calendar is the new datepicker component that replaces @InputDate. @Calendar allows user to deal with 3 kind of date configurations viz. date only, date and time, and time only.

Table 6. Calendar Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Calendar | cannot be changed |
| readOnly | boolean | false | hide/show the component |
| labelClass | String | anthem-label | class used to style the label |
| type | String | calendar | type of calendar |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |
| help | String |  | help text that will be displayed in tooltip component |
| hidden | boolean | false | hide/show component |
| showTime | boolean | false | hide/show time |
| timeOnly | boolean | false | show only time without date |
| hourFormat | String | "12" | hour format to display, can be "12" or "24" |

Calendar.java

|  |
| --- |
| @Path  @Calendar(postEventOnChange=true, showtime=true, hourFormat="24")  private LocalDate startDate;    @Path  @Calendar(postEventOnChange=true, timeOnly=true)  private LocalDateTime startDate; |

**CardDetail**

Table 7. CardDetail Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CardDetail | cannot be changed |
| cssClass | String | contentBox right-gutter bg-light mt-1 | class used to style the CardDetail |
| imgSrc | String |  | source path/ url of the image |
| editable | boolean | false | make the component editable |
| modelPath | String |  | specify the model path for CardDetail |
| title | String |  | specify the title of the CardDetail |
| draggable | boolean | false | determine of the CardDetailsGrid is draggable |

CardDetail.java

|  |
| --- |
| @CardDetail(title="Member Overview", cssClass="contentBox right-gutter bg-alternate mt-0")  private CardDetailMember cardDetailMember; |

**contentBox right-gutter bg-alternate mt-0** overrides the default cssClass specified for the CardDetail

**CardDetail Children Components**

CardDetail has several children components that nested level components can utilize to break the component into logical sections.

CardDetailChildren.java

|  |
| --- |
| @Model  @Getter @Setter  public static class MyCard {        @CardDetail.Header      MyCardDetailTag myCardTag;        @CardDetail.Body      MyCardDetailBody myCardBody;  }    @Model  @Getter @Setter  public static class MyCardDetailTag { ... }    @Model  @Getter @Setter  public static class MyCardDetailBody { ... } |

**CardDetail.Body**

See the available CardDetail children components below:

Table 8. CardDetail.Body Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CardDetailsBody | cannot be changed |
| cssClass | String |  | class used to style the component |

**CardDetail.Footer**

Table 9. CardDetail.Footer Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CardDetailsFooter | cannot be changed |
| cssClass | String |  | class used to style the component |

**CardDetail.Header**

Table 10. CardDetail.Header Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CardDetailsHeader | cannot be changed |
| cssClass | String |  | class used to style the component |

**CardDetail.Tag**

Table 11. CardDetail.Tag Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CardDetailsTag | cannot be changed |
| cssClass | String |  | class used to style the component |

**CardDetailsGrid**

Table 12. CardDetailsGrid Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CardDetailsGrid | cannot be changed |
| editUrl | String |  |  |
| draggable | boolean | false | determine of the CardDetailsGrid is draggable |
| onLoad | boolean | false | specify the behaviour of CardDetailsGrid during page load |

CardDetailsGrid.java

|  |
| --- |
| @CardDetailsGrid(onLoad=true, draggable=true)  @Config(url="/pageHealthConcerns/tileHealthConcerns/sectionConcerns/concernsCards.m/\_process?fn=\_set&amp;url=/p/cmcase/\_search?fn=query&amp;where=cmcase.id.eq('<!/.m/id!>')&amp;project=/healthProblemsEnclosed")  @Path(linked=false)  private List<CardDetailConcerns> concernsCards; |

**onLoad=true** and **draggable=true** will override the default false values for onLoad and draggable respectively.  
*For information regarding @Config, please read View Configuration Annotations section*

**CheckBox**

Table 13. CheckBox Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CheckBox | cannot be changed |
| cssClass | String |  | class used to style the CheckBox |
| labelClass | String | anthem-label | class used to style the CheckBox label |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |
| help | String |  | help text that will be displayed in tooltip component |

CheckBox.java

|  |
| --- |
| @CheckBox(postEventOnChange=true)  private boolean admin; |

**postEventOnChange=true** will override the default false value of postEventOnChange.

**CheckBoxGroup**

CheckBoxGroup can be used for multi-select checkboxes.

Table 14. CheckBoxGroup Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | CheckBoxGroup | cannot be changed |
| cssClass | String |  | class used to style the CheckBox |
| labelClass | String | anthem-label | class used to style the CheckBox label |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  |  |
| help | String |  | help text that will be displayed in tooltip component |

CheckBoxGroup.java

|  |
| --- |
| @CheckBoxGroup(postEventOnChange=true)  private String[] days; |

**postEventOnChange=true** will override the default false value of postEventOnChange. **DataType** for checkboxgroup attribute should be **String[]** to hold multiple values.

**ComboBox**

Table 15. ComboBox Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | ComboBox | cannot be changed |
| readOnly | boolean | false | make the component readonly |
| labelClass | String | anthem-label | class used to style the label |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |
| help | String |  | help text that will be displayed in tooltip component |

ComboBox.java

|  |
| --- |
| @Path  @Model.Param.Values(url="~/client/orgname/staticCodeValue/\_search?fn=lookup&amp;where=staticCodeValue.paramCode.eq('/goalCategory')")  @ComboBox  private String goalCategory; |

Please refer @Model in the core config annotations section.

**FieldValue**

A simple field which holds a value. Has the ability to become an in place edit field with several configurable properties.

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | FieldValue | cannot be changed |
| cols | String | 4 | the number of columns to display |
| cssClass | String |  | class used to style the component |
| datePattern | String |  | date format to display on ui - use LocalDate/ LocalDateTime/ Date/ ZonedDateTime as datatypes. String will not work. |
| iconField | String |  | adds a class 'iconField' and the value of iconField to the css class for this component |
| imgSrc | String |  | source path/url of the image |
| inPlaceEdit | boolean | false | whether or not to configure this field as able to be edited in place |
| inPlaceEditType | String |  | set the Type of the in place edit component |
| placeholder | String |  | placeholder text to be displayed when the state of this param is null |
| showName | boolean | true | whether or not to show the label for this component |
| value | Enum (Type) | Type.Field | classification of the link useful for determining functional behavior |
| url | String |  | source url to operate on |

cols supported for FieldValue component in a CardDetail component are **1, 2, 3, 4 (default)**.

FieldValue.java

|  |
| --- |
| @MapsTo.Type(CMGoal.class)  @Getter @Setter  public static class VCHeaderGoal {      @FieldValue(showName=false, cols="2")      @MapsTo.Path("/description")      private String description;        @FieldValue(showName=false, iconField="date")      @MapsTo.Path("/targetDate")      private LocalDate targetDate;        @FieldValue(showName=false, iconField="planned")      @MapsTo.Path("/status")      private String goalStatus;        @FieldValue(showName=false, datePattern="MM/dd/yyyy")      @MapsTo.Path("/startDate")      private LocalDate startDate;  } |

**FileUpload**

Table 16. FileUpload Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | FileUpload | Cannot be changed |
| type | String | .pdf,.png | comma separated list of file types user is allowed to upload |

FileUpload.java

|  |
| --- |
| @FileUpload(type=".png,.pdf")  private String fileUpload; |

The above configuration allows only ".png" and ".pdf" to be uploaded by the users. If they try to upload other file types, an error is thrown.

**FilterButton**

A collection of buttons to operate with a configurable filter

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | FilterButton | cannot be changed |
| b | String | $execute | behavior to use |
| cssClass | String | btn btn-primary | class used to style the component |
| imgSrc | String |  | source path/url of the image |
| method | String | GET | HTTP method that will be executed |
| url | String |  | source url to operate on |

FilterButton.java

|  |
| --- |
| @FilterButton  private VFbAllTasks vFbAllTasks;    @Model @Getter @Setter  public static class VFbAllTasks {        @Config(url="/vpHome/vtTasks/vsTasks/tasks.m/\_process?fn=\_set&amp;url=/p/task/\_search?fn=query&amp;where=task.status.eq('Open')")      @Button(cssClass="btn btn-secondary btn-badge")      private String vbOpenTasks;  } |

Decorate complex classes with @FilterButton to define behavior for child elements (e.g. vbOpenTasks in the example above.)

**Form**

Table 17. Form Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Form | cannot be changed |
| submitUrl | String |  | url that will be hit after submit |
| b | String |  | b |
| cssClass | String |  | class used to style the section |
| submitButton | boolean | true | form will be submitted |
| navLink | String |  | navigation link |

CSS styles supported for Form component are **oneColumn, twoColumn, threeColumn, fourColumn, questionGroup, sixColumn, inline**.

Form.java

|  |
| --- |
| @Form(cssClass="twoColumn")  @Path(linked=false, state=State.External)  private FormGoalDetails goalDetailsForm; |

**GlobalHeader**

@GlobalHeader component allows a user to configure the main header of an application.

Table 18. GlobalHeader Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | Global-Header | Y | cannot be changed |
| cssClass | String |  | N | class used to style the component |

GlobalHeader.java

|  |
| --- |
| @GlobalHeader()  private VSHomeHeader vsHomeHeader; |

VSHomeHeader.java

|  |
| --- |
| @Paragraph  @PageHeader(Property.APPTITLE)  @Label(value = "Long Term Services &amp; Supports (LTSS)")  private String productName; |

**GlobalFooter**

GlobalFooter component lets you configure the sections in the Footer.

Table 19. GlobalFooter Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | GlobalFooter | Y | cannot be changed |

Table 20. GlobalFooter Properties - FooterProperty

| **Name** | **Occurance** | **Description** |
| --- | --- | --- |
| DISCLAIMER | 1 | Disclaimer text to be shown in the Footer |
| LINK | \* | Links to sitemap, privacy, terms of use etc. Can have multiple occurances. |
| SSLCERT | 1 | SSL Cert link |

GlobalFooter.java

|  |
| --- |
| @GlobalFooter()  private VSHomeFooter vsHomeFooter;    @Paragraph  @FooterProperty(Property.DISCLAIMER)  @Label(value = "© 2005 - 2017 copyright of Anthem Insurance Companies, Inc. Serving Colorado, Connecticut, Georgia,\n"          + "        Indiana, Kentucky, Maine, Missouri (excluding 30 counties in the Kansas City area), Nevada, New Hampshire, Ohio,\n"          + "        Virginia (excluding the Northern Virginia suburbs of Washington, D.C.), and Wisconsin.")  private String appDisclaimer;    @Link(url = "#")  @FooterProperty(Property.LINK)  @Label(value = "Site Map")  private String siteMap; |

**GlobalNavMenu**

@GlobalNavMenu component lets you configure the navigation bar that sits besides the global header. It can display logged in user’s organization(s), menus that have one or more sub menu links, and individual link. A user can configure these items based on the requirement per application.

Table 21. GlobalNavMenu Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | Global-Nav-Menu | Y | cannot be changed |
| cssClass | String |  | N | class used to style the component |

GlobalNavMenu.java

|  |
| --- |
| @GlobalNavMenu()  private VSHomeMenu vsHomeMenu; |

VSHomeMenu.java

|  |
| --- |
| @ComboBox  @Model.Param.Values(value = ListOrg.class)  private String organizationName;    @Link(cssClass = "menuLink", url = "#")  @Label("this is link menu without submenus")  private String individualLink1;    @Menu  @Label("this is for menu1")  private MenuHomeSettingsLinks1 menuHomeSettingsLinks1;    @Model  @Getter  @Setter  public static class MenuHomeSettingsLinks1 {    @Link(cssClass = "menuLink", url = "#")  @Label(value = "Sub Menu 1")  private String subMenu1;    @Link(cssClass = "menuLink", url = "#")  @Label(value = "Sub Menu 2")  private String subMenu2;      } |

**Grid**

The @Grid component can be used to display tabular data within an HTML table. As the table itself can be configured in a variety of ways, so too can the @Grid component.

Table 22. Grid Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Grid | cannot be changed |
| isTransient | boolean | false | specify the behaviour of CardDetailsGrid during page load |
| expandableRows | boolean | false | whether or not expandable rows should be enabled |
| onLoad | boolean | false | specify the behaviour of CardDetailsGrid during page load |
| rowSelection | boolean | false | determine if row selection is enabled on the Grid |
| pageSize | String | 10 | specify the page size for the Grid |
| pagination | boolean | true | determine if pagination is enabled on the Grid |
| postButton | boolean | false | determine if post Button is enabled on the Grid |
| postButtonUrl | String |  | specify the url of post button for the Grid |
| postButtonTargetPath | String |  | Target path for the multi select rows on the grid. The payload for the selections will be created based on this path. |
| postButtonAlias | String |  | Button name alias for the multiselect action on the grid |
| postButtonLabel | String |  | Button label to be used on the post button |
| postEventOnChange | boolean | false | post the value to the server onblur |
| url | String |  | specify url for the Grid |

**Configuring the @Grid component**

In the example below, we can see a typical structure for using @Grid.

|  |
| --- |
| @Grid  private List<PetLineItem> pets; |

**Configuring the @Grid component with multiple row selection**

In the example below, we can see an example of configuring a grid that should allow the user to select multiple rows and then perform some action with those selections.

|  |
| --- |
| // Provide the postButtonUrl based on the config.  @Grid(rowSelection=true,          postButtonUrl="/view/page/tile/section/remove",          postButton=true,          postButtonTargetPath="temp\_ids",          postButtonAlias="Remove")  @Path("/members")  private List<VGGroupView> vgGroupList;    @Configs({      @Config(url="/page/tile/section/vsRemoveList/\_replace"),      @Config(url="/page/tile/section/vgGroupList/<!col!>/.m/\_process?fn=\_setByRule&amp;rule=updategroup", col="<!/vsRemoveList/temp\_ids!>"),      @Config(url="<!#this!>/../vgGroupList.m/\_process?fn=\_set&amp;url=/p/group/\_search?fn=query&amp;project=/members")  })  private String remove;    private VSRemoveList vsRemoveList;    @Getter @Setter  public static class VSRemoveList {      private List<String> temp\_ids;  } |

**Controlling column data with @GridColumn**

The @GridColumn component has been introduced to provide control over the individual params displayed on the @Grid. @GridColumn will be recognized when used within the *generic class* of a collection that has been annotated with @Grid.

Table 23. GridColumn Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | GridColumn | cannot be changed |
| datePattern | String |  | date format to display on ui - use LocalDate/ LocalDateTime/ Date/ ZonedDateTime as datatypes. String will not work. |
| hidden | boolean | false | Setting this to true will hide the column in the grid |
| filterMode | String | equals | Mode to filter data within the column. It is defaulted to 'equals'.Current filter modes supported by the Grid component are listed as enum values - equals,contains,endsWith,in |
| filter | boolean | false | Setting this to true will enable a explicit filter component(Input TextBox) on the UI |
| filterValue | String |  | Default filterValue for a particular column.Can be used without setting filter attribute to restrict user from providing any other filter value. |
| placeholder | String |  | placeholder text to be displayed when the state of this param is null |
| sortable | boolean | false | Enables sorting on the column |
| sortAs | String | DEFAULT | supports (DEFAULT, NUMBER, TEXT). DEFAULT would sort the column as a string, NUMBER would sort the column as a numeric where the actual datatype could be String (e.g. id field), TEXT would again sort the column as default (string) |

**Configuring the @GridColumn component**

In the example below, we can see a typical structure for using @GridColumn.

|  |
| --- |
| @Grid  private List<PetLineItem> pets;    @MapsTo.Type(Pet.class)  @Getter @Setter  public static class PetLineItem {        @GridColumn      @MapsTo.Path      private String petName;        @GridColumn(datePattern="MM/dd/yyyy")      @MapsTo.Path      private LocalDate startDate;  } |

In this scenario, PetLineItem is the *generic class* of our collection pets and petName represents the column entry of the data displayed in the HTML table.

This example does not really showcase the usage of what @GridColumn can do, but is more intended to show a structure of what a typical object might look like. More examples will be provided below to show scenarios where we can leverage it to provide control over the column.

@GridColumn is not required to be present in order to render the param on the UI. Adding it simply supplies the user with more control over the column, if desired.

**Hiding a Column from being displayed**

Consider the scenario where we want to keep a param in the *generic class* of a collection decorated with @Grid, but we do not want that param to be displayed as a column in the HTML table.

The following example will show us how to achieve this:

|  |
| --- |
| @MapsTo.Type(Pet.class)  @Getter @Setter  public static class PetLineItem {        @GridColumn(hidden = true)      @MapsTo.Path      private String id;        @GridColumn      @MapsTo.Path      private String petName;  } |

In this scenario, only petName will be displayed in the rendered HTML table. The id param is hidden as a result of hidden = true.

**Controlling row data with @GridRowBody**

The @GridRowBody component has been introduced to render additional data within an HTML table, outside of the standard table row content. When decorating a param within the *generic class* of a @Grid, the UI will render an "expandable row" that can be toggled to display additional data.

This functionality is intended to be used when:

1. The data to be displayed in the "expanded row" is viewed as non-vital data.
2. There are too many columns to display within a table’s row.

The field expandableRows of @Grid should be set to true when using a @GridRowBody. e.g. @Grid(expandableRows = true).

Table 24. GridRowBody Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | GridRowBody | Cannot be changed |
| cssClass | String |  | Defines a css class for the "expanded row" container |

**Which elements are able to be placed within @GridRowBody**

@GridRowBody should always be used on a *complex object* (an object that has nested objects). In this way, the UI framework is able to discern that the param decorated with @GridRowBody is a container of sorts that is expected to render other components within it.

The framework treats @GridRowBody in a similar fashion to @Section. Consequently, the list of available components that are able to be rendered within the @GridRowBody are the same as those that can be rendered by @Section:

* @Button
* @ButtonGroup
* @CardDetail
* @CardDetailGrid
* @ComboBox
* @Form
* @Grid
* @GridContainer
* @Link
* @Menu
* @Paragraph
* @StaticText
* @TextBox

**Configuring the @GridRowBody component**

Consider the following scenario where the need is to treat some data as non-vital data:

|  |
| --- |
| @MapsTo.Type(Pet.class)  @Getter @Setter  public static class PetLineItem {        @GridColumn      @MapsTo.Path      private String petName;        @GridRowBody      private ExpandedRowContent expandedRowContent;        @Model @Getter @Setter      public static class ExpandedRowContent {            @CardDetail          private CardDetails cardDetails;      }        @Model @Getter @Setter      public static class CardDetails {            @CardDetail.Body          private CardBody cardBody;      }        @Model @Getter @Setter      public static class CardBody {            @FieldValue          @MapsTo.Path("/id")          private String id;      }  } |

This is a bit of a larger example since we have used the @CardDetail component, but in this scenario we are displaying a table with a single column for petName. Since we have decorated expandedRowContent with @GridRowBody, we will display an "expanded row" containing the idfield displayed as a @FieldValue component.

**Hints**

Table 25. Hints Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | enum | AlignOptions.Inherit | Align optinons of Hints |

Hints.java

|  |
| --- |
| @Link(url="/#/h/cmdashboard/vpDashboard", imgSrc="anthem-rev.svg")  @Hints(AlignOptions.Left)  @PageHeader(Property.LOGO)  private String linkHomeLogo; |

The possible Property enum values for value are **INHERIT, LEFT, RIGHT, CENTER**

**Image**

An image component used for displaying images.

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Image | cannot be changed |
| imgSrc | String |  | source path/ url of the image |

Image.java

|  |
| --- |
| @Image(imgSrc="/resources/img.png")  private String title; |

**InPlaceEdit**

A component which allows for in-line editing of a particular field.

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | InPlaceEdit | cannot be changed |
| type | String | text | the type of field being configured. Supported types are **Text**, **Textarea**, and **ComboBox** |

**InputDate**

Table 26. InputDate Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | InputDate | cannot be changed |
| readOnly | boolean | false | hide/show the component |
| labelClass | String | anthem-label | class used to style the label |
| type | String | date | type of input date |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |
| help | String |  | help text that will be displayed in tooltip component |

InputDate.java

|  |
| --- |
| @Path  @InputDate private LocalDate startDate; |

**Label**

Table 27. Label Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | String |  | Content Text |
| localeLanguageTag | String | en-US | Locale |
| helpText | String |  | HelpText content whereever valid\* |

Label.java

|  |
| --- |
| @Label("First Name")  private String firstName;    @Label(value="Last Name")  private String lastName;    @Label(value = "Select 3" , helpText = "Please select atleast 3 cities")  @CheckBoxGroup  private String[] city;    @Tile  @Label("Dashboard")  private VTDashboard vtDashboard;    @Page  @Label(" ")  private VPDashboard vpDashboardWithEmptyLabel;    @Label(value="Test Label C in English", helpText="some tooltip text here C")  @Label(value="Test Label A in French", localeLanguageTag="fr")  private String staticText;    @Textbox  private String addressline; |

@Label can be used on all View annotations like @Page, @Tile, @TextBox etc wherever content is necessary. Below is the list of components which can use @Label for content text. Multiple @Label annotations can be provided for managing content for different locales.

* Page
* Tile
* Accordion
* CardDetails
* SubHeaders
* Link
* FieldValue
* Button
* Action dropdown
* Filter Button
* CheckBox \*
* CheckBox Group \*
* ComboBox \*
* Grid, Grid Column
* InputDate \*
* Modal
* MultiSelectCard \*
* Radio \*
* TextBox \*
* TextArea

\* View Components that can display help text as a tooltip.

If a label is not provided in the config for above UI components , then the code value of the param is displayed. Example : In the above snippet of code for the param "adddressline", since @Label has not been provided, the UI would display a TextBox with addressLine as the label value.

**Link**

A hyperlink component used for navigation or user interaction of text.

Links can currently be used under the following components:

* CardDetail
* Menu
* Section

Table 28. Link Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| altText | String |  | alternate text to be used |
| b | String | $executeAnd$nav | behavior to use |
| cssClass | String |  | class used to style the component |
| imgSrc | String |  | source path/ url of the image |
| method | String | GET | HTTP method that will be executed |
| rel | String |  | the HTML rel attribute |
| target | String |  | the HTML target attribute |
| url | String |  | source url to operate on |
| value | Enum (Type) | Type.Default | classification of the link useful for determining functional behavior |

**Type.java**

The possible enum values for value are: **DEFAULT, EXTERNAL, MENU**.

Link.java

|  |
| --- |
| // UI Navigation to page vpHome under domain petclinic  @Link(url="/h/petclinic/vpHome")  private String linkToHome;    // Executes a request against \_subscribe, and sets the image  @Link(url="/notifications/\_subscribe:{id}/\_process", b="$executeAnd$configAnd$nav" , method="POST")  private String subscribeToEmailNotifications;    // Creates an external link, as in a link navigating outside of the context of the Nimbus framework.  @Link(url="[https://www.mywebsite.com"](https://www.mywebsite.com/), value = Link.Type.EXTERNAL, target="\_new", rel="nofollow")  private String myWebsiteLink; |

**LinkMenu**

LinkMenu component is used for displaying links for a grid row. LinkMenu groups the actions for a grid row and displays it as a drop down menu. Link Menu contains @Link components. Variables can be passed in the Link URLs using {} syntax (Example below).

Table 29. LinkMenu Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | LinkMenu | Y | cannot be changed |
| cssClass | String | false | N | custom style class |

LinkMenu.java

|  |
| --- |
| @LinkMenu  private VLMCaseItemLinks vlmCaseItemLinks;    @Model  @Getter @Setter  private static class VLMCaseItemLinks {      @Configs({ @Config(url = "/p/cmcaseview:<!/.m/id!>/\_get?b=$execute"),          @Config(url = "/p/cmcaseview:<!/.m/id!>/\_nav?pageId=pageCaseInfo")      })      @Link(imgSrc = "task.svg")      @Label(value = "View Case")      private String viewCase;        @Configs({          @Config(url = "/p/caseassignmentwithgridview/\_new?fn=\_initEntity&amp;target=/vpAssignmentTask/vtTaskDetails/vsShowMycases/associatedCaseId&amp;json=<!/.m/id!>")      })      @Link(imgSrc = "task.svg")      @Label(value = "Assign Case Owner")      private String assignCase;        @Link(value=Type.EXTERNAL,target="\_blank",imgSrc = "task.svg" , url="[http://va10dwviss323.us.ad.wellpoint.com:81/CUEDEV\_Services/RxShowDocument.aspx?userid=AC633468&amp;documentid={documentKey}"](http://va10dwviss323.us.ad.wellpoint.com:81/CUEDEV_Services/RxShowDocument.aspx?userid=AC633468&amp;documentid=%7bdocumentKey%7d))      @MapsTo.Path(linked = false)      private String viewDoc;  } |

**Menu**

A section intended to contain other navigation components.

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String |  | cannot be changed |
| cssClass | String |  | class used to style the component |
| value | Enum (Type) | Type.CONTEXT | classification of the link useful for determining functional behavior |

**Type.java**

The possible enum values for value are: **CONTEXT**.

Menu.java

|  |
| --- |
| // Defines a menu with elements home and signup  @Menu  private SideMenu sideMenu;    @Model @Getter @Setter  public static class SideMenu {      @Link(url="#")      private String home;        @Link(url="#")      private String signup;  } |

Decorate complex classes with @Menu to define behavior for child elements (e.g. home and signup in the example above.)

**Modal**

Table 30. Modal Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | Modal | Y | Cannot be changed. |
| closable | boolean | false | Y | Specify whether the modal window can be closed. |
| context | ParamContext | @ParamContext(enabled = true, visible = false) | N | Specify the default contextual properties. |
| cssClass | String |  | N | Override default class. |
| title | String |  | N | Title for the Modal window. |
| type | Type | Type.dialog | Y | Types of modal can be a dialog or a slider. |
| width | String | medium | Y | Width of the modal window |

width supported for Modal component are **small, large, medium (default)**.

Modal.java

|  |
| --- |
| @Modal  private VMMyTasks vmMyTasks; |

Modal is supported only in a tile

**Setting @Modal to be displayed by default**

Modal makes use of **@ParamContext** to by default render it’s visible property to false during initialization. We can override this functionality by providing our own unique **@ParamContext** values as a parameter of **@Modal**.

public static class SampleView {

@Modal(context = @ParamContext(enabled=true, visible=true))

private MyModal myModal;

public static class MyModal { ... }

}

In this example, we are creating a modal *myModal* who’s visible property will be true on initialization, and hence will be displayed when the corresponding page is rendered.

**MultiGrid**

Table 31. MultiGrid Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | MultiGrid | cannot be changed |
| level | String | 0 | class used to style the picklist label |
| header | String | test | make the component readonly |
| cssClass | String | question-header | class used to style the picklist |
| postEventOnChange | boolean | false | post the value to the server onblur |

**MultiSelect**

Table 32. MultiSelect Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | MultiSelect | cannot be changed |
| labelClass | String | anthem-label | class used to style the picklist label |
| postEventOnChange | boolean | false | post the value to the server onblur |
| help | String |  | help text that will be displayed in tooltip component |

MultiSelect.java

|  |
| --- |
| @MultiSelect  @Model.Param.Values(url="Anthem/fep/icr/p/clientusergroup/\_search?fn=lookup&amp;projection.mapsTo=code:id,label:displayName")  @MapsTo.Path(value="/userGroups", colElemPath="/ownerId")  private List<String> userGroups; |

**MultiSelectCard**

Table 33. MultiSelectCard Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | MultiSelectCard | cannot be changed |

MultiSelectCard.java

|  |
| --- |
| @MultiSelectCard @NotNull  @Model.Param.Values(url="~/client/orgname/cmprogram/\_search?fn=lookup&amp;projection.mapsTo=code:id,label:programName")  @Path(value="/programReferred", colElemPath="/programName")  private List<String> cmProgramList; |

@NotNull indicates that the component cannot be null.

**Page**

Table 34. Page Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | Page | Y | Cannot be changed. |
| title | String |  | N | Specify the page title to be displayed on top of the page. |
| imgSrc | String |  |  | class name for the image next to page title. |
| cssClass | String |  | Y | Custom class used to style the page. |
| defaultPage | boolean | false | Y | Determine if the page will be the default page. Default page loads first in a flow. |
| route | String |  | N | Specify route of the page. |

The possible enum values for type are **HOME, DETAILS, FORM, STATIC**

Page.java

|  |
| --- |
| @Page(route="cmcaseR/assignmenttaskview", defaultPage=true)  private Page\_AssignmentTask pageAssignmentTask; |

**defaultPage=true** will override the default false value of defaultPage.

**PageFooter**

Table 35. PageFooter Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | enum | Property.Default | property of PageFooter |

PageFooter.java

|  |
| --- |
| @PageFooter(Property.VERSION)  private String appVersion; |

The possible Property enum values for value are **DEFAULT, TOU, VERSION, COPYRIGHT, SSLCERT, PRIVACY**

**PageHeader**

Table 36. PageHeader Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | enum | Property.Default | property of PageHeader |

PageHeader.java

|  |
| --- |
| @PageHeader(Property.USERNAME)  private String fullName; |

The possible Property enum values for value are **DEFAULT, LOGO, APPTITLE, SUBTITLE, USERNAME, USERROLE, SUBHEADER, MENU**

**PickList**

Table 37. PickList Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | PickList | cannot be changed |
| labelClass | String | anthem-label | class used to style the picklist label |
| readOnly | boolean | false | make the component readonly |
| cssClass | String |  | class used to style the picklist |
| postEventOnChange | boolean | false | post the value to the server onblur |
| sourceHeader | String | SourceList | specify the source header for the picklist |
| targetHeader | String | TargetList | specify the target header for the picklist |
| help | String |  | help text that will be displayed in tooltip component |

**Radio**

Table 38. Radio Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | Radio | cannot be changed |
| labelClass | String | anthem-label | class used to style the radio label |
| level | String | 0 | determine the level of radio group |
| cssClass | String |  | class used to style the radio |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |
| help | String |  | help text that will be displayed in tooltip component |

Radio.java

|  |
| --- |
| @Model.Param.Values(url="~/client/orgname/staticCodeValue/\_search?fn=lookup&amp;where=staticCodeValue.paramCode.eq('/thisIsParentQuestion14')")  @MapsTo.Path()  @Radio(postEventOnChange=true, controlId = "27")  private String thisIsParentQuestion14; |

**postEventOnChange=true** will override the default false value of postEventOnChange.  
**controlId** is used because in this particular case, there is a question number (27) that is associated with an assessment.  
Please refer Core Config Annotations section for @MapsTo

**Section**

Table 39. Section Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | enum | Type.Default | section type |
| alias | String | Section | cannot be changed |
| cssClass | String |  | class used to style the section |

Section.java

|  |
| --- |
| @Section  @Path(linked=false)  private SectionEditGoal sectionEditGoal; |

The possible enum values for size are **HEADER, LEFTBAR, DEFAULT**. CssClass is optional.

**StaticText**

Directly outputs a raw html content/text value to the view. May use this to bind unsafe HTML directly onto the page. Use wisely.

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | StaticText | cannot be changed |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |

StaticText.java

|  |
| --- |
| @StaticText  private String description; |

**SubHeader**

Unresolved directive in TechnicalDocumentation/pages/viewConfigAnnotations.adoc - include::viewConfigAnnotations/subHeader.adoc[]

**TextArea**

A text input component that allows for a specified number of rows.

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | TextArea | cannot be changed |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |
| help | String |  | help text that will be displayed in tooltip component |
| hidden | boolean | false | hide/show the component |
| labelClass | String | anthem-label | class used to style the label |
| postEventOnChange | boolean | false | post the value to the server onblur |
| readOnly | boolean | false | make the component readonly |
| rows | String | 5 | type of input |
| type | String | textarea | type of input |

TextArea.java

|  |
| --- |
| @TextArea  private String description; |

**TextBox**

Table 40. TextBox Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | TextBox | cannot be changed |
| hidden | boolean | false | hide/show the component |
| readOnly | boolean | false | make the component readonly |
| help | String |  | help text that will be displayed in tooltip component |
| labelClass | String | anthem-label | class used to style the label |
| type | String | text | type of input |
| postEventOnChange | boolean | false | post the value to the server onblur |
| controlId | String |  | currently not in use except for assessments where this signifies the question number |

TextBox.java

|  |
| --- |
| @TextBox(hidden=true)  private String goalId; |

In the above example, the default false value of hidden property will be overriden and the textbox will be hidden. Similarly, other properties can be overriden by specifying the properties and its values along with @Textbox.

**Tile**

Table 41. Tile Attributes

| **Name** | **Type** | **Default** | **Required** | **Description** |
| --- | --- | --- | --- | --- |
| alias | String | Tile | Y | Cannot be changed. |
| imgSrc | String |  | N | Source path/ url of the image. |
| title | String |  | N | Title displayed on top of the tile. |
| size | enum | Size.Large | Y | Size of the tile width on the page. Size.XSmall - 25% width Size.Small - 33% width Size.Medium - 50% width Size.Large - 100% width |

The stle of Tile component is determined by value of the Size attribute. Default is **Size.LARGE**.

Tile.java

|  |
| --- |
| @Tile(size=Tile.Size.Large)  private TileEditGoal tileEditGoal; |

The possible enum values for size are **XSmall, Small, Medium, Large**

**ViewRoot**

Table 42. ViewRoot Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | root | cannot be changed |
| layout | String | "" | determine layout of root |

ViewRoot.java

|  |
| --- |
| @Domain(value = "cmcaseview", includeListeners={ListenerType.websocket}, lifecycle="cmcaseview")  @MapsTo.Type(CMCase.class)  @Repo(value=Database.rep\_none, cache=Cache.rep\_device)  @ViewRoot(layout = "caseoverviewlayout")  @Getter @Setter  public class VRCmCase {  } |

layout of caseoverviewlayout is leveraged here.

**Core Config Annotations**

**ConceptId**

Table 43. ConceptId Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | String | "" | possible value for conceptId |

ConceptId.java

|  |
| --- |
| @TextBox(postEventOnChange = true)  @ConceptId(value = "IOT1.1.1")  @Label(value = "If Other, provide reason")  private String otherReason; |

**Config**

Table 44. Config Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| url | String |  |  |

Config.java

|  |
| --- |
| @Config(url="/pageAddEditGoal/tileEditGoal/sectionEditGoal/goalDetailsForm/\_nav?pageId=pageCarePlanSummary")  @Button(type = Button.Type.PLAIN)  private String cancel; |

A class with @Config annotation is used to perform an action on button click. In most cases, the action is to retrieve values via HTTP Rest calls from database (MongoDB), and display on the web page.

In the example shown above, when the button is clicked, the control will be navigated to the specified url.  
*nav* is the http call for navigation.

The possible Actions are: -

* get for HTTP GET
* new for HTTP post
* update for HTTP update
* delete for HTTP delete
* search for searching
* nav for navigation
* process for custom process/ work-flow definitions

**Configs**

Table 45. Configs Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| values | Config[] |  |  |

Configs.java

|  |
| --- |
| @Configs({  @Config(url="~/client/org\_name/\_update"),  @Config(url="~/client/org\_name/\_process?fn=\_set&amp;url=/p/cmcase/\_search?fn=query&amp;where=cmcase.patientReferred.firstName.eq('<!/.m/patientReferred/firstName!>').and(cmcase.patientReferred.lastName.eq('<!/.m/patientReferred/lastName!>'))"),  @Config(url="~/client/org\_name/\_nav?pageId=vpAdvancedCaseSearch")          })  @Button(type=Button.Type.PRIMARY, formReset=false)  private String submit; |

@Configs is a collection of configuration calls. Multiple calls to the database can be made simultaneously using @Configs, where each call will be represented by @Config.

As shown in the example, when the submit button is clicked, three http calls that are being made viz. update, process, and navigation.

**ConfigNature**

**Ignore**

Framework persists the data objects of a class in database using @Repo by serializing the class and associating a version number that is called seriaVersionUID. However, if we do not want the framwework to serialize for time being, we can use @Ignore component of ConfigNature class. The following example shows that: -

StartsWith.java

|  |
| --- |
| @Domain(value="patient", includeListeners={ListenerType.persistence})  @Repo(value=Database.rep\_mongodb, cache=Cache.rep\_device)  @ToString  public class Patient extends IdString {        @Ignore      private static final long serialVersionUID = 1L;        } |

**Domain**

Core Config configuration @Domain annotation persists data.

Core config @Domain will always be followed by @Repo that will specify the way data is persisted.

**includeListeners={ListenerType.persistence, ListenerType.update}** of @Domain specifies that the data will be persisted.  
**value=Database.rep\_mongodb** of @Repo specifies that a class with @Domain annotation will use MongoDb for persistence.

Domain.java

|  |
| --- |
| @Domain(value="cmcase", includeListeners={ListenerType.persistence, ListenerType.update})  @Repo(value=Database.rep\_mongodb, cache=Cache.rep\_device)  public class CMCase extends IdString {  } |

Please read @Repo for mroe information regarding @Repo annotation.

**DomainMeta**

Similar to ViewStyle component, DomainMeta component is used to define an ANNOTATION\_TYPE level annotation that is used with few view config annotations, as follows: -

* @ConceptId

|  |
| --- |
| @Retention(RetentionPolicy.RUNTIME)      @Target(value={ElementType.ANNOTATION\_TYPE})      @Inherited      public @interface DomainMeta {        } |

Here is an example: -

|  |
| --- |
| @Retention(RetentionPolicy.RUNTIME)      @Target({ElementType.FIELD})      @DomainMeta      public @interface ConceptId {          String value() default "";      } |

**Execution**

Execution is inherited annotation for @Config and @KeyValue. It is not currently being directly used, but is there for hieracrchial purposes.

More documentation will be added here if Execution expands or is directly used.

**Initialize**

Table 46. Initialize Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String | initialize | cannot be changed |

Initialize.java

|  |
| --- |
| @Section  @Initialize  @Config(url="/vpAdvancedMemberSearch/vtAdvancedMemberSearch/vsMemberSearchCriteria/vfAdvMemberSearch/\_process?fn=\_setByRule&amp;rule=updateadvmbrsearchcriteria")  private VSMemberSearchCriteria vsMemberSearchCriteria; |

**MapsTo**

**Path**

Table 47. Path Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | String |  | specify value of Path |
| colElemPath | String | DEFAULT\_COL\_ELEM\_PATH | specifies column element path |
| state | State | State.Internal | specifie path state |
| linked | boolean | true | determine if the path has links |
| cache | Cache | Cache.rep\_device | specifies cache type |

Path.java

|  |
| --- |
| @Path()  private Long version; |

**Type**

Table 48. Type Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | class<?> |  |  |
| silent | boolean | true | determines if Type is silent |

Type.java

|  |
| --- |
| @MapsTo.Type(CMCase.class)  public static class SectionEditGoal  {      } |

If it is not mapped *(@Mapped)*, an exception will be thrown.  
If no exception is thrown, defaults to silent.

**Mode**

Table 49. Mode Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | enum | Options.INHERIT | OPtions of Mode |

The possible Property enum values for value are **INHERIT, READONLY, HIDDEN**

**Model**

**Param**

Table 50. Param Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | Class<? extends Source> | Source.class | specify value of Path |
| url | String | staticCodeValue | specifies url |

Param.java

|  |
| --- |
| @Model.Param.Values(url="~/client/orgname/staticCodeValue/\_search?fn=lookup&amp;where=staticCodeValue.paramCode.eq('/thisIsParentQuestion14')")  @Radio(postEventOnChange=true, controlId = "27")  private String thisIsParentQuestion14; |

The example will retrieve the value of Radio button from the specified url.  
**staticCodeValue** is a collection in MongoDB that will be searched and retrieved using @Model annotation. The search criteria is specified using param code *thisIsParentQuestion14*

**Text**

Table 51. Text Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| label | String |  |  |

**ParamContext**

**@ParamContext** is used to set the contextual properties of a field during the *OnStateLoad* event (e.g. *visible*, *enabled*).

Table 52. ParamContext Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| visible | boolean |  | Whether or not the decorated target should be visible. |
| enabled | boolean |  | Whether or not the decorated target should be enabled. |

The intent of **@ParamContext** is to be able to decorate fields to define default contextual behavior. For example:

SampleView.java

|  |
| --- |
| public static class SampleView {       @TextBox     @ParamContext(enabled=false, visible=false)     private String myTextBox;  } |

In this scenario we have configured the contextual values for *enabled* and *visible* to be **false** for *myTextBox*. These values will be set during the *OnStateLoad* event and *myTextBox* consequently will not be enabled or visible when the corresponding page is rendered.

**@ParamContext** can also be defined on annotations. In these scenarios when a field is decorated with that annotation, then the handler for **@ParamContext** will execute. This may be useful when building a framework extension.

**SearchNature**

**StartsWith**

This component is ued to validate a field. The wilcard attribute determines the validation criteria for a field. This is a server side component.

Table 53. StartsWith Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| wildCard | String | \* | determines the search criteria |

StartsWith.java

|  |
| --- |
| @NotNull  @StartsWith  @Label(value = "First Name")  private String firstName; |

The example will always search the first name that starts with anything, represented with the default value. A specific search criteria can be specified using wildcard attribute of @StartsWith.

**Repo**

Table 54. Repo Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| alias | String |  | cannot be changed |
| value | enum |  | specify value of Path |
| cache | enum | Cache.rep\_device | specifies cache type |
| autoSave | boolean | true | determine if the autosave is enabled |

The possible enum values for value are **DATABASE.rep\_none, DATABASE.rep\_rdbms, DATABASE.rep\_mongodb, DATABASE.rep\_ws**  
The possible enum values for cache are **CACHE.rep\_none, CACHE.rep\_device**

@Repo is used to determine where the data will be persisted. It is always used along with @Domain.

**rep\_mongodb**

The following example shows how data is persisted/ retrieved using MongoDB as a source.

Repo.java

|  |
| --- |
| @Domain(value="cmassessment", includeListeners={ListenerType.persistence})  @Repo(alias="cmassessment",value=Database.rep\_mongodb, cache=Cache.rep\_device)  @Getter @Setter  public class CMAssessment extends IdString {  } |

**rep\_ws**

Databse values can be persisted/ retrieved not only using internal MongoDb as a source, we can now use an external web service for the same purpose. All we have to do is create a REST call to a web service that will provide or store the data. Following is an example: -

ExtClient.java

|  |
| --- |
| @Domain(value="ext\_client")  @Repo(value=Database.rep\_ws, cache=Cache.rep\_device)  @Getter @Setter @ToString(callSuper=true)  public class ExtClient {    } |

Notice **@Repo(value=Database.rep\_ws)** in the code above. This indicates that it will make use of an external web service call rather than a MongoDb call for ExtClient data objects' persistence/ retrieval purposes.

The advantage of this feature is that now we do not rely on internal source only. This gives us better flexibility and maintainability

**Rule**

@Rule allows its decorated field a mechanism for triggering one or more rule definitions during its OnStateLoad and OnStateChange events.

SampleRuleEntity.java

|  |
| --- |
| @Domain(value="sample\_rule\_entity", includeListeners={ListenerType.persistence})  @Repo(Database.rep\_mongodb)  @Getter @Setter  public class SampleRuleEntity {        // Execute the rule at "rules/sample\_increment" during the OnStateLoad and      // OnStateChange events of ruleParam.      @Rule("rules/sample\_increment")      private String rule\_param;  } |

By default, the framework provides support for firing all rules for a given domain entity. That is, for the **SampleRuleEntity.java** above we might have a rule file defined as **sample\_rule\_entity.drl** which will be automatically fired by naming convention.

For cases where additional configuration for other rules is needed, @Rule can be used.

Table 55. Rule Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | String[] |  | rule file path(s) to execute. |

**Values**

Values provides a mechanism for populating a fields *values* property. This can be used by a number of components to perform such functions as: define a set of selections for radio buttons and checkboxes, or populating a dropdown list.

Table 56. Values Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | Class<? extends Source> | EMPTY | Source implementation that returns a list of **ParamValues** |
| url | String |  | url endpoint that returns a list of **ParamValues** |

*value*'s default value is **EMPTY**, which is a simple Source implementation that returns an empty list of **ParamValues**.

**Source**

The Source is a simple abstraction for providing a contract between implementations to provide data to the framework.

Source is exclusively used for **@Values**.

Source.java

|  |
| --- |
| public static interface Source {      public List<ParamValue> getValues(String paramCode);  } |

We can use this to define several different types of values providers. A simple static Source implementation is shown below:

SampleStaticSource.java

|  |
| --- |
| public class SampleStaticSource implements Source {      public List<ParamValue> getValues(String paramCode) {          List<ParamValue> values = new ArrayList();          values.add(new ParamValue("sample.value.1", "Sample Value 1"));          return values;      }  } |

**Examples**

Using a static Source implementation to define a set of values

|  |
| --- |
| @Values(SampleStaticSource.class)  @CheckBoxGroup  private String petTypes; |

In this example, all of the values retrieved from SampleStaticSource.getValues will be displayed as a collection of checkboxes.

Using a url-based Source implementation to define a set of values

|  |
| --- |
| @Values(url="CLIENT\_ID/ORG/p/staticCodeValue/\_search?fn=lookup&amp;where=staticCodeValue.paramCode.eq('/petType')")  @CheckBoxGroup  private String petTypes; |

In this example, all of the values retrieved from the url defined in **@Values** will be displayed as a collection of checkboxes.

**ViewParamBehavior Component**

Similar to ViewStyle component, ViewParamBehavior component is used to define an ANNOTATION\_TYPE level annotation that is used with few view config annotations, as follows: -

* @Hints
* @Initialize
* @Mode
* @PageHeader
* @PageFooter

|  |
| --- |
| @Retention(RetentionPolicy.RUNTIME)      @Target(value={ElementType.ANNOTATION\_TYPE})      @Inherited      public @interface ViewParamBehavior {        } |

Here is an example: -

|  |
| --- |
| @Retention(RetentionPolicy.RUNTIME)      @Target({ElementType.FIELD})      @ViewParamBehavior      public @interface Initialize {          String alias() default "initialize";      } |

**ViewStyle Component**

ViewParamBehavior component is used to define an ANNOTATION\_TYPE level annotation that is used with all most of the annotations described below.

|  |
| --- |
| @Retention(RetentionPolicy.RUNTIME)      @Target(value={ElementType.ANNOTATION\_TYPE})      @Inherited      public @interface ViewStyle {        } |

The above code is the definition if ViewStyle component. As you can see, ElementType.ANNOTATION\_TYPE determines that this annotation can only be used in other annotations.

Here is an example: -

|  |
| --- |
| @Retention(RetentionPolicy.RUNTIME)      @Target({ElementType.TYPE})      @ViewStyle      public @interface ViewRoot {          String alias() default "root";          String layout() default "";      } |

**Conditional Config Annotations**

This section covers the *conditional behavior* that can be applied to params. In this context, *conditional behavior* refers to performing a set of actions based on a given condition. The condition is evaluated by using the powerful capabilities of *SpEL (Spring Expression Language)*.

Typically this *conditional behavior* is evaluated from the context of the *conditionally decorated param* (the param that is annotated with a conditional annotation), meaning that the condition will inspect the *decorated param* object to infer if the condition should be true or false.

The following example shows what a conditional configuration might look like:

|  |
| --- |
| @SampleConditional(when = "state == 'YES'")  @Radio(postEventOnChange = true)  @Values(YesNo.class)  private String question; |

The when condition is effectively stating, when question.getState().equals("YES"). In this case, since question is the *decorated param*, we are evaluating from the context of question. This is why state is understood in the *SpEL* condition state == 'YES'.

In this scenario @SampleConditional is not a real annotation understood by the framework, but it illustrates the idea that when this particular *condition* is true, we should perform some sort of behavior. We defer that logic to *State Event Handlers*.

**State Event Handlers**

The role of a *State Event Handler* is to instruct the framework on how to process a conditional annotation. Therefore, it is responsible for maintaining the logic to be applied when a conditional annotation is present on a param. *State Event Handlers* are registered with the framework through the use of @EventHandler on startup as Spring beans. The logic is then executed during specific event hooks defined within the framework.

**Framework Defined Conditional Annotations**

**AccessConditional**

@AccessConditional allows to control the access of the specific area/functionality within the application based on User’s assigned role(s) and permission(s).

Example:

|  |
| --- |
| @AccessConditional(whenAuthorities="?[#this == 'entity\_assign'].empty", p=Permission.HIDDEN)  private Section\_EntityAssignment vsEntityAssignment; |

Above configuration would hide the entity assignment section for user(s) who do not have access to "entity\_assign" action.

Table 57. AccessConditional Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| whenAuthorities | String |  | SpEL (spring expression language) to check for user permissions **Recommended** |
| whenRoles | String |  | SpEL (spring expression language) to check for user roles |
| containsAuthorites | String[] |  | Simple contains check for user permissions. (for more complex conditions, use *whenAuthorities*) |
| containsRoles | String[] |  | Simple contains check for user roles. (for more complex conditions, use *whenRoles*) |

***whenRoles*** and ***containsRoles*** should be used with caution, since roles may be added/changed more frequently in the application and in that event, you would need to update the entity configurations.

More Examples:

|  |
| --- |
| // vbEntityCreate will be hidden for user(s) who do not have access to "entity\_create"  @AccessConditional(whenAuthorities="?[#this == 'entity\_create'].empty", p=Permission.HIDDEN)  private Button\_EntityCreate vbEntityCreate;    //vsEntityEdit will be hidden for user(s) who do not have access to "entity\_create" or "entity\_edit"  @AccessConditional(whenAuthorities="?[#this == 'entity\_create'].empty &amp;&amp; ?[#this == 'entity\_edit'].empty", p=Permission.READ)  private Section\_EntityEdit vsEntityEdit; |

**AccessConditionals**

@AccessConditionals is a collection of @AccessConditional calls.

Table 58. AccessConditionals Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | AccessConditional[] |  |  |

AccessConditionals.java

|  |
| --- |
| @AccessConditionals({      @AccessConditional(whenAuthorities="!?[#this == 'entity\_management'].empty || !?[#this == 'entity\_create'].empty", p=Permission.READ)      @AccessConditional(whenAuthorities="?[#this == 'entity\_management'].empty &amp;&amp; ?[#this == 'entity\_create'].empty", p=Permission.HIDDEN)  })  private String entity\_accessConditionals; |

**ActivateConditional**

@ActivateConditionals is a collection of @ActivateConditional calls.

Table 59. ActivateConditionals Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | ActivateConditional[] |  |  |

ActivateConditionals.java

|  |
| --- |
| @ActivateConditionals({      @ActivateConditional(when="state == 'A'", targetPath="/../q3Level1"),      @ActivateConditional(when="state == 'B'", targetPath="/../q3Level2")  })  private String q3;  private SampleCoreNestedEntity q3Level1;  private SampleCoreNestedEntity q3Level2; |

**ActivateConditionals**

@ActivateConditionals is a collection of @ActivateConditional calls.

Table 60. ActivateConditionals Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | ActivateConditional[] |  |  |

ActivateConditionals.java

|  |
| --- |
| @ActivateConditionals({      @ActivateConditional(when="state == 'A'", targetPath="/../q3Level1"),      @ActivateConditional(when="state == 'B'", targetPath="/../q3Level2")  })  private String q3;  private SampleCoreNestedEntity q3Level1;  private SampleCoreNestedEntity q3Level2; |

**ConfigConditional**

@ConfigConditional is an extension capability provided by the framework. The annotation is used to conditionally execute @Config calls based on a *SpEL* based condition. This annotation can be triggered for multiple events. Framework provides default event handling for this annotation on StateChange.

Table 61. ConfigConditional Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| when | String |  | SpEL based condition to be evaluated relative to param’s state on which this annotation is declared. |
| config | Config[] | emptyString | Array of @Configs |

ConfigConditional.java

|  |
| --- |
| @ConfigConditional(      when = "state == 'Completed'", config = {      @Config(url="<!#this!>/../state/\_update?rawpayload=\"Closed\""),      @Config(url="/p/dashboard/\_get")  })  private String status; |

In the above example , whenever there is statechange of status and the status is changed to **Completed**, the Configs will be executed.

**EnableConditional**

@EnableConditional is an extension capability provided by the framework. The annotation is used to conditionally activate/deactivate the param based on a SpEL condition. This annotation can be triggered for multiple events. Framework provides default event handling for this annotation on StateChange and StateLoad. The difference between this annotation and @ActivateConditional is this annotation only affects the "enabled" state whereas @ActivateConditional affects both "enabled" and "visible" state.

Table 62. EnableConditional Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| when | String |  | SpEL based condition to be evaluated relative to param’s state on which this annotation is declared. |
| targetPath | String[] |  | Path of param to enable when condition is satisfied relative to param on which this annotation is declared |
| disableWhen | String |  | SpEL based condition on which param would be disabled. If value is not overridden, then the negation of EnableConditional#when() would be used |

EnableConditional.java

|  |
| --- |
| @EnableConditional(when="state == 'hooli'", targetPath="../enable\_p2")  private String enable\_p1;    private String enable\_p2; |

**EnableConditionals**

@EnableConditionals is a collection of @EnableConditional calls.

**ExpressionConditional**

@ExpressionConditional is a very versatile conditional annotation that can be used to apply any logic that is able to be crafted using *SpEL*expressions.

The framework provides default event handling for this annotation during the events:

* OnStateLoad
* OnStateChange

Table 63. ExpressionConditional Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| then | String |  | SpEL based condition to be evaluated relative to param’s state when the *when* attribute evaluates to *true*. |
| when | String |  | SpEL based condition to be evaluated relative to param’s state on which this annotation is declared. |

**Configuring @ExpressionConditional**

@ExpressionConditional works by first evaluating the when attribute by means of a *SpEL* condition. When the when condition is true, the framework will next execute the *SpEL* expression contained in the then attribute.

Consider the following example where we conditionally set the state on load:

|  |
| --- |
| @ExpressionConditional(when="null == state &amp;&amp; onLoad()", then="setState(T(java.time.LocalDate).now())")  private LocalDate initDateOnLoad; |

Recall that the when expression is evaluated from the context of the param that @ExpressionConditional decorates (initDateOnLoad). In this scenario, the when expression evaluates to true when the param is intially loaded and it’s state is null.

The then condition uses the SpEL language to set the value of LocalDate.now() into initDateOnLoad.

The observant will notice that @ExpressionConditional is so versatile in that it could be used to perform the same logic achieved by several other framework defined *conditional annotations*. Ideally it should not be used when another *conditional annotation* is available to achieve the same behavor.

**Available Param Methods**

Given that @ExpressionConditional is extremely versatile, misuse of the annotation could result in unexpected behavior from within the framework. As such, the access to the context param in the then condition has been restricted to provide access to a subset of it’s methods.

The methods available to be used can be identified by reviewing EntityStateHolder.

**ExpressionConditionals**

@ExpressionConditionals is a collection of @ExpressionConditional calls.

@ExpressionConditionals should be used when multiple @ExpressionConditional calls are needed to decorate a single param. See the following example:

|  |
| --- |
| @ExpressionConditionals({      @ExpressionConditional(when="null != state", then="findStateByPath('/../acceptedBy').setState(state)"),      @ExpressionConditional(when="null != state", then="findStateByPath('/../signedForBy').setState(state)")  })  @TextBox(postEventOnChange = true)  private LocalDate name;    @TextBox  private String acceptedBy;    @TextBox  private String signedForBy; |

In this scenario, when name is not null, the framework will set the value of name into acceptedBy and signedForBy.

**ValidateConditional**

@ValidateConditional is used to conditionally set validations that should appear for a param based on a *SpEL* condition. This annotation can be triggered for multiple conditions, if necessary.

The framework provides default event handling for this annotation on StateChange.

Table 64. ValidateConditional Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| scope | ValidationScope | ValidationScope.SIBLING | The scope for the param layer at which validations will be applied |
| targetGroup | Class<? extends ValidationGroup> |  | Specifies which validation group should be applied |
| when | String |  | Configures whether or not the first truthy condition should be exclusive. If true, then only the first truthy condition will be executed. If false then all truthy conditions will be executed. the default value is true. |

**Configuring Conditional by Group**

@ValidateConditional works by first evaluating the when attribute by means of a *SpEL* condition. When the when condition is true, the framework will attempt to identify a subset of params with Validation constraints and apply validation logic to those params. See the following example:

ValidateConditional Sample 1

|  |
| --- |
| @ValidateConditional(when = "state == 'dog'", targetGroup = GROUP\_1.class)  @TextBox(postEventOnChange = true)  private String petType;    @NotNull  @TextBox  private String petName;    @NotNull(groups = { GROUP\_1.class })  @TextBox  private String dogFood;    @NotNull(groups = { GROUP\_2.class })  @TextBox  private String catFood; |

**ValidateConditional Sample 1 - Results**

In the above example, assuming the state of petType is "dog", the validations that will be applied are: petName and dogFood.

* petName - There are no groups associated with petName, hence it is seen as a *static validation* and always applied.
* dogFood - GROUP\_1.class is present within the @NotNull’s `groups attribute, and this is matching petType’s `@ValidateConditional’s `targetGroup attribute(GROUP\_1.class) and is applied.

The validations that will not be applied are: catFood.

* *catFood* - While *catFood* has an entry in @NotNull’s `groups attribute(GROUP\_2.class), it is not matching @ValidateConditional’s `targetGroup attribute(GROUP\_1.class).

**Specifying a Validation Group**

As seen in the previous example, there may be many params decorated with *javax.validation.constraints*. The framework needs to uniquely identify which validations should be applied when the when condition is **true**. To handle this, the groups attribute will be used as it is supplied by of all *javax.validation.constraints* annotation classes. The final subset of params where validations will be applied will be composed of only those whos groups attribute contains the @ValidateConditional targetGroup attribute.

@ValidateConditional’s `targetGroup parameter is simply a marker interface of type ValidationGroup to be used for identification purposes by the framework to identify the subset of params.

Use [@ValidateConditionals](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/_validateconditionals) when:

* Different groups should be applied based on multiple *when* conditions
* Multiple groups should be applied for the same *when* condition

**Validation Group Identity Classes**

In the previous example, we used targetGroup = GROUP\_1.class. GROUP\_1.class is an identity class that implements the ValidationGroupinterface and is used by the framework to identify *javax.validation.constraints* annotatons that should be applied.

Any implementation that implements the *ValidationGroup* interface may be used as a group identity class. For convenience, a set of identify class implementations have been defined within @ValidateConditional as GROUP\_**X**.class, where 0 <= **X** <= 29.

If additional marker classes are needed, simply create a new implementation of ValidationGroup and use that class in the targetGroupattribute as well as the corresponding param’s *javax.validation.constraints* annotation.

**Controlling Scope**

Given that the framework uses groups to identify which params should have validations applied, it needs a way to also limit the number of params that will be considered during the processing phase. For example, should a @ValidateConditional trigger **all** defined params in the application that have a particular targetGroup? While necessary, such an operation may be costly.

The framework has provided the ability to define a ValidationScope as part of @ValidateConditional’s `scope to provide control over these scenarios. The following scopes are available:

Table 65. @ValidateConditional Scopes

| **Name** | **Description** |
| --- | --- |
| **SIBLING** | Applies validations to sibling params relative to the current param on which this annotation is defined. |
| **SIBLING\_NESTED** | Applies validations to sibling params relative to the current param on which this annotation is defined. Also recursively traverses each of the previous param’s nested params (or children) and applies validations. |

**ValidateConditionals**

@ValidateConditionals is a collection of @ValidateConditional calls.

@ValidateConditionals should be used when multiple @ValidateConditional calls are needed to decorate a single param. See the following example:

ValidateConditionals Sample 1

|  |
| --- |
| @ValidateConditionals({      @ValidateConditional(when = "state == 'dog'", targetGroup = GROUP\_1.class),      @ValidateConditional(when = "state == 'cat'", targetGroup = GROUP\_2.class)  })  @TextBox(postEventOnChange = true)  private String petType;    @NotNull  @TextBox  private String petName;    @NotNull(groups = { GROUP\_1.class })  @TextBox  private String dogFood;    @NotNull(groups = { GROUP\_2.class })  @TextBox  private String catFood; |

We will examine two scenarios using Sample 1 above. One scenario where petType has the state "dog" and one where it has the state "cat".

**ValidateConditionals Sample 1 - Results 1**

The state of *petType* is "dog".

The params that have validations that will be applied are: petName and dogFood.

* petName - There are no groups associated with petName, hence it is seen as a *static validation* and always applied.
* dogFood - GROUP\_1.class is present within the @NotNull’s `groups attribute, and this is matching petType’s `@ValidateConditional’s `targetGroup attribute(GROUP\_1.class) and is applied.

The params that have validations that will not be applied are: catFood.

* catFood - While catFood has an entry in @NotNull’s `groups attribute(GROUP\_2.class), it is not matching @ValidateConditional’s `targetGroup attribute(GROUP\_1.class).

**ValidateConditionals Sample 1 - Results 2**

The state of *petType* is "cat".

The params that have validations that will be applied are: petName and catFood.

* petName - There are no groups associated with petName, hence it is seen as a *static validation* and always applied.
* catFood - GROUP\_2.class is present within the @NotNull’s `groups attribute, and this is matching petType’s `@ValidateConditional’s `targetGroup attribute(GROUP\_2.class) and is applied.

The params that have validations that will not be applied are: dogFood.

* dogFood - While dogFood has an entry in @NotNull’s `groups attribute(GROUP\_1.class), it is not matching @ValidateConditional’s `targetGroup attribute(GROUP\_2.class).

**ValuesConditional**

@ValuesConditional is an extension capability provided by the framework. The annotation is used to conditionally set the @Valuesconfiguration for a dependent field based on a SpEL condition. This annotation can be triggered for multiple events. Framework provides default event handling for this annotation on StateChange and StateLoad.

Table 66. ValuesConditional Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| target | String |  | The target path relative to the this annotated field to update. |
| condition | Condition[] |  | An array of conditional metadata that is responsible for determining whether or not the target field will be updated. |
| resetOnChange | boolean | true | When **true** and the associated *when* expression evaluates to **true**, the *state* of *target* will be set to null. When **false** and the associated *when* expression evaluates to **true**, the *state* of *target* will be set to null only when the previously existing *target*'s *state* does not exist in the *then*'s **@Values** property. Otherwise, the existing *state* will be preserved. |
| exclusive | boolean | true | Configures whether or not the first truthy condition should be exclusive. If true, then only the first truthy condition will be executed. If false then all truthy conditions will be executed. the default value is true. |

Table 67. Condition Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| when | String |  | SpEL based condition to be evaluated relative to param’s state on which this annotation is declared. |
| then | Values |  | Values configuration to be applied to the param identified by the target path when this condition’s when clause is found to be true. |

Consider the following sample defined Values:

SampleValues.java

|  |
| --- |
| // Sample Values Implementations  public static class SRC\_FOODS\_ALL implements Source {      @Override      public List<ParamValue> getValues(String paramCode) {          final List<ParamValue> values = new ArrayList<>();          values.add(new ParamValue("Generic Food 1", "Generic Food 1"));          values.add(new ParamValue("Generic Food 2", "Generic Food 2"));          return values;      }  }  public static class SRC\_FOODS\_DOG implements Source {      @Override      public List<ParamValue> getValues(String paramCode) {          final List<ParamValue> values = new ArrayList<>();          values.add(new ParamValue("Dog Food 1", "Dog Food 1"));          return values;      }  }  public static class SRC\_FOODS\_CAT implements Source {      @Override      public List<ParamValue> getValues(String paramCode) {          final List<ParamValue> values = new ArrayList<>();          values.add(new ParamValue("Cat Food 1", "Cat Food 1"));          return values;      }  } |

Given a defined set of Values that we can assign using the @Values annotation, we can explicitly define conditions to set a dependent field’s values.

ValuesConditional.java

|  |
| --- |
| @Model  @Getter @Setter  public static class StatusForm {        @ValuesConditional(target = "../petFoodSelection", condition = {              @Condition(when = "state=='dog'", then = @Values(SRC\_FOODS\_DOG.class)),              @Condition(when = "state=='cat'", then = @Values(SRC\_FOODS\_CAT.class)),          }      )      @TextBox(postEventOnChange = true)      private String petType;        @Radio      @Values(SRC\_FOODS\_ALL.class)      private String petFoodSelection;  } |

In this example, *petType* is the field and *petFoodSelection* is the dependent field. We set *petFoodSelection* to contain the defaults ["Generic Food 1", "Generic Food 2"] initially and conditionally define those values when *petType*'s state is "dog" or "cat".

When the state of *petType* is "dog", then the Values for *petFoodSelection* will be ["Dog Food 1"]. Similarly when the state of *petType* is "cat", then the Values for *petFoodSelection* will be ["Cat Food 1"].

Conceptually speaking, we are **pushing** the updates of Values to the dependent field whenever the state of the annotated field loads or is changed.

**ValuesConditionals**

@ValuesConditionals is a collection of @ValuesConditional calls.

The framework has the capability for a field to define ValuesConditional to affect multiple dependent fields.

Table 68. ValuesConditionals Attributes

| **Name** | **Type** | **Default** | **Description** |
| --- | --- | --- | --- |
| value | ValuesConditional[] |  |  |

ValuesConditionals.java

|  |
| --- |
| @Model  @Getter @Setter  public static class StatusForm {        @ValuesConditionals({          @ValuesConditional(target = "../petFoodSelection", condition = {                  @Condition(when = "state=='dog'", then = @Values(SRC\_FOODS\_DOG.class)),                  @Condition(when = "state=='cat'", then = @Values(SRC\_FOODS\_CAT.class)),              }          ),          @ValuesConditional(target = "../petFoodSelection2", condition = {                  @Condition(when = "state=='dog'", then = @Values(SRC\_FOODS\_DOG.class)),              }          )      })      @TextBox(postEventOnChange = true)      private String petType;        @Radio      @Values(SRC\_FOODS\_ALL.class)      private String petFoodSelection;        @Radio      private String petFoodSelection2;  } |

In this example, we have two dependent fields: *petFoodSelection* and *petFoodSelection2*. Note that nothing has changed for the *petFoodSelection*field from the previous example, but we have set a @ValuesConditional for the dependent field *petFoodSelection2* that when the state of *status* is "dog", the Values of *petFoodSelection2* will be ["Dog Food 1"].

**View Components**

**Tooltip**

This is a new custom component that is added in the framework to provide additional information about all the input fields that are used within the application. The input fields are the fields that require user selection like textbox, textarea, combobox, checkbox, input date etc. In order to create a tooltip component for a particular field, all that is required is to add a help attribute in the field configuration. Please look at the following example: -

Tooltip.java

|  |
| --- |
| @Radio(postEventOnChange = true, help="This field requires a value for pet type")  private String petType; |

The tooltip component will be created for this field and will display the specified value in the help field.

**User Interface Validation**

**UI Validation**

**JSR Annotations**

Support has been added for JSR 303 type annotations. All annotations inheriting from *javax.validation.Constraint* will be picked up and added to a configured parameter’s configuration.

Specific support within the UI has been added for the following javax.validation annotations

Table 69. Supported Constraint Annotations

| **Name** | **Description** |
| --- | --- |
| @NotNull | Requires that a field should not have a null value. |
| @Pattern | Requires that a field follow a specific regex pattern. |
| @Min | Requires that a minimum value should be provided. |
| @Max | Requires that a maximum value should be provided. |
| @Size | Requires that a specific range should be provided. Useful for component groups. |

**Static Validations**

*Static validations* are validations that will always be applied when rendered on the UI. All *javax.validation.constraints* annotations are equipped with a *groups* attribute of type *Class<? extends Payload>[]*. When *groups* is empty (this is the default behavior), the framework treats the validation as a *static validation*.

**Conditional Validations**

*Conditional validations* are validations that will only be applied on the UI when a particular condition has been met. An annotation from*javax.validation.constraints* is understood as a *conditional validation* when the *groups* attribute is not empty.

*Conditional validations* can be activated by the use of [@ValidateConditional](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Technical+Documentation#TechnicalDocumentation-validateconditional).

If *groups* has a non-empty array value, the validation will NOT be applied unless it is conditionally activated.

**Special Considerations**

**@Size**

* When used in conjunction with **@CheckBoxGroup**, **@Size** will require the given number (*min*) of checkboxes to be selected or will limit the given number (*max*) of checkboxes that are able to be selected.

**Custom Messages**

The framework has the ability to pass custom validation messages from within JSR Annotations to the UI.

Repo.java

|  |
| --- |
| @NotNull(message = "First name should not be empty.")  private String status; |

Here, if the UI field 'status' is left blank by the user, "First name should not be empty" message would be displayed. @NotNull will just display the default message.

**Layout Configuration**

As you can see in the screenshot above, the layout of the application is divided into 3 sections viz. Header, Footer, and Leftbar. Please read the code below to understand the way the sections has been created.

Repo.java

|  |
| --- |
| public static class VPHome {    @Section(Type.HEADER)  private VSHomeHeader vsHomeHeader;    @Section(Type.LEFTBAR)  private VSHomeLeftBar vsHomeLeftBar;    @Section(Type.FOOTER)  private VSHomeFooter vsHomeFooter;        } |

@Section(Type.***value***) determines the section on the page. This code shows three of the *type* values for @Section. For more values, please refer to the @Section annotation within View Config Annotations.

**Collection Configuration**

Collection configuration is introduced in the framework to eliminate the creation and use of different functional handlers for adding, editing and deleting elements from/ to a collection. This structure now gives us the ability to work on any of these functions on any of the elements from a collection at any given time.

This configuration is explained using a use case of a Car. A user might want to see a Car home page that has list of cars. User may be able to add, edit, or delete cars and also view properties of car. The first step is to create a core domain entity.

CarEntity.java

|  |
| --- |
| @Domain(value="Car", includeListeners={ListenerType.persistence})  @Repo(Database.rep\_mongodb)  @Getter @Setter  public class CarEntity extends IdString {        private String attr\_String;        private int attr\_int;      private Integer attr\_Integer;        private CarNestedEntity attr\_CarNestedEntity;        private List<CarNestedEntity> attr\_list\_2\_CarNestedEntity;        } |

This configuration provides us with the flexibility such that we can define a custom set of car features to be displayed. The features are defined in the nested entity as shown in the code below: -

CarNestedEntity.java

|  |
| --- |
| @Model  @Getter @Setter  public class CarNestedEntity {        private String car\_name;  } |

The following class defines the home page that will show a list of cars. Notice the way in which the nested entities are mapped to its property.

VPCarHome.java

|  |
| --- |
| @MapsTo.Type(CarEntity.class)  @Getter @Setter  public class VPCarHome {        @Tile(title="Car Home", size=Tile.Size.Large)      private TileBlue tile;        @MapsTo.Type(CarEntity.class)      @Getter @Setter      public static class TileBlue {            // add new Nested Entity          @Configs(              @Config(url="/page\_red/tile/vt\_attached\_convertedNestedEntity/\_get?fn=param&amp;expr=assignMapsTo('../.m/attr\_list\_2\_CarNestedEntity')")          )          private String addButton;            // view/edit mapped collection to core          @Path(value="/attr\_list\_2\_CarNestedEntity")          private List<Section\_ConvertedNestedEntity> vm\_attached\_convertedList;        }        @MapsTo.Type(CarNestedEntity.class)      @Getter @Setter      public static class Section\_ConvertedNestedEntity {            // assigning collection element of core to Form          @Configs(              @Config(url="/page\_red/tile/vt\_attached\_convertedNestedEntity/\_get?fn=param&amp;expr=assignMapsTo('/.d/<!#this!>/../.m')")          )          private String editButton;            @Path("/car\_name")          private String vm\_car\_name;      }    } |

As you can see, when you click on addButton, the config url will follow the domain model path until the form view where a user can save the new car (nested entity), and then switch from view config to core config using .m to assign a new car to attr\_list\_2\_CarNestedEntity list. The Section\_ConvertedNestedEntity is a nested entity that is used to view/ edit collection that is mapped to core, using edit button and vm\_car\_name respectively. The edit button that has a config url that will be resolved in a similar way to that of add button, however, as you can see, there is a .d that signifies the mapping to the core domain model i.e. it will point to the root which is CarEntity.

VPCarDetails.java

|  |
| --- |
| @MapsTo.Type(CarEntity.class)  @Getter @Setter  public class VPCarDetails {        @Tile(title="Car Details", size=Tile.Size.Large)      private TileRed tile;        @MapsTo.Type(CarEntity.class)      @Getter @Setter      public static class TileRed {            @Path(value="/attr\_list\_2\_CarNestedEntity", nature=Nature.TransientColElem)          private Form\_ConvertedNestedEntity vt\_attached\_convertedNestedEntity;      }        @MapsTo.Type(CarNestedEntity.class)      @Getter @Setter      public static class Form\_ConvertedNestedEntity {            @Configs(              @Config(url="<!#this!>/../\_update")          )          private String saveButton;            @Configs({              @Config(url="<!#this!>/../.m/\_delete"),              @Config(url="<!#this!>/../\_get?fn=param&amp;expr=unassignMapsTo()")          })          private String deleteButton;            @Path("/car\_name")          private String vt\_car\_name;      }    } |

This is the car details page that will be displayed when user clicks on add/ edit button from the Home page. This is used to create another nested entity, however, please notice **nature=Nature.TransientColElem**. This property is used to determine that Form\_ConvertedNestedEntity have transient relationship with the Car (core entity). Form\_ConvertedNestedEntity will display the car property (car name) and two buttons viz. save and delete to perform add/edit and delete functions respectively.

Hence, by using the collection configuration, we can eliminate the use of add/edit/delete custom functional handlers to perform add/edit/delete functions respectively.

**Rule Configuration**

Framework comes with support for Drools Rules (<https://www.drools.org/>) engine out of the box. The rules can be defined at multiple levels.

**Where can rules be configured?**

* **Entity Level**: Rules defined at the entity level gets triggered everytime the entity is updated. Typically business rules that manage the entity state would be defined at this level. The framework would pick any rule with the extension <domainalias>.drl and associate with the entity updates. The rule would have access to the entity domain that it is being executed.
* **Function Handler Level**: Business rules that can span across multiple entities or needs to be controlled based on business flow rather than everytime entity is updated can be configured using Function handler. Please refer to the [Function Handler documentation](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Documentation.html#_function_handlers) for more information on configuring these rules.
* **Attribute Level**: Activation and deactivation at attribute level based on certain conditions can be handled using @ActivateConditional annotation. Please refer to the [ActivateConditional documentation](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Documentation.html#_activateconditional) for more information.
* **Configuration Level:** Individual fields can be annotated with **@Rule** so that a particular rules file(s) will be executed. Please refer to the [Rule documentation](https://anthemopensource.atlassian.net/wiki/spaces/OSS/pages/156959159/Documentation.html#_rule) for more information.

**Audit Configuration**

**Framework has the ability to create an Audit Configuration for leaf parameters to generate audit history on state change of the annotated parameter.**

When a leafParam is annotated with @Audit, based on the event handler defined for the annotation, the audit history is created for that particular leafParam. By default, the @Audit functionality is invoked for OnStateChange event handlers.

Usage : @Audit(AuditHistory.class)

Car.java

|  |
| --- |
| @Audit(SampleCarAuditEntry.class)  private String audit\_String;    @Audit(SampleCarAuditEntry.class)  private Integer audit\_Integer;    @Audit(SampleCarAuditEntry.class)  private String unmapped\_String; |

VPCar.java

|  |
| --- |
| @Audit(SampleCarAuditEntry.class)  @Path  private String audit\_String; // mapped and marked with Audit on both view &amp; core    @Audit(SampleViewCarAuditEntry.class)  private String unmapped\_String; // unmapped and marked with Audit on a quad with persistent core    @Path  private Integer audit\_Integer; // mapped view to a core which is annotated with Audit |

As you can see @Audit can be annotated on both view and core entities.

**Process Configuration**

**Business Process Configuration**

Activiti BPM engine (<https://www.activiti.org/>) is currently integrated out of the box with the framework. All standard BPM functions are available for creating business processes. Business process can be defined to manage business entity lifecycle and for creation of stateless processes that executes complex business functions.

**Entity Lifecyle Management using BPM**

The framework provides the ability to back a business or view entity using a workflow. There might be scenarios where an entity needs to traverse through a series of business steps. Having a BPM defined through standard BPMN construct can help with configuring the business steps and will also provide a visual depiction of the bsuiness configuration.

**Associating a workflow with an entity**

Lets take the example domain named UmCaseFlow with an alias "umcaseview".

UmCaseFlow.java

|  |
| --- |
| @Domain(value="umcaseview", includeListeners={ListenerType.websocket}, lifecycle ="umcaseview")  @MapsTo.Type(UMCase.class)  @Repo(value=Database.rep\_none)//default cache is repo\_session  @Getter @Setter  public class UmCaseFlow {        @Page(route="umCaseR", defaultPage=true)      private PageUMCaseCreateMember pageCreateMember;        @Page(route="umCaseR")      private PageUMCaseCreateProvider pageCreateProvider;        @Page(route="umCaseR")      private PageUMCaseCreateCaseInfo pageCreateCaseInfo;    } |

@Domain Annotation takes a "lifecyle" attribute. The attribute value represents the process id of the BPMN backing the entity. The bpmn execution has access to the ProcesEngineContext through the key

**Stateless Business Function implementation using BPM**

BPMN provides a strong visual and expression syntax for configuring business processes. It might not be always associated with an entity lifecyle. If there is a need to execute a statless BPM as a function.

1. Steps for configuring a BPMN process as a function
   * Create a BPMN process with a unique process id
   * Define the config url with action as **\_process**,**fn** as **\_bpm** and **processId** as the name of the process id. Example **@Config(url="/p/patient:<!/.m/id!>/\_process?fn=\_bpm&processId=createcaseforpatient")**

**Function Handlers**

**Function Handlers** are an abstraction within the framework to execute a common set of instructions for a given **Action**.

Given an action, a particular function handler can be executed by specifying a value for the query parameter **fn**.

Example configuration for a \_set function handler

|  |
| --- |
| @Config(url="/p/patient:<!/.m/id!>/\_process?fn=\_set") |

In the example above, ***process*** is the **Action** and ***set*** is the Function Handler we have configured to execute as a part of this **@Config**configuration.

**Predefined Function Handlers**

There are several default function handlers defined within the core framework to handle common framework instructions, such as setting the state of a parameter (***set***) or adding a parameter value into a collection (***add***).

The following function handlers are provided out-of-the-box:

Table 70. Predefined Function Handlers

| **Handler** | **Action** | **Description** |
| --- | --- | --- |
| \_param | \_get |  |
| \_default | \_nav |  |
| \_initEntity | \_new |  |
| \_add | \_process | Adds parameter value to a collection. |
| \_addCollection | \_process |  |
| \_bpm | \_process | Invokes a statless bpm process. |
| \_eval | \_process |  |
| \_set | \_process | Sets parameter value. |
| \_setByRule | \_process | Set parameter value through a rule. |
| \_example | \_search |  |
| \_lookup | \_search |  |
| \_query | \_search |  |

**Custom Function Handlers**

The framework provides the ability to users to define custom function handlers.

To create a custom function handler, create a **@Bean** to be injected in the spring context which implements **FunctionHandler** class from the framework. Below are two samples for creating custom handlers.

PetClinicExtensionConfig.java

|  |
| --- |
| @Configuration  public class PetClinicExtensionConfig {      @Bean(name="default.\_process$execute?fn=\_custom")      public CustomFunctionHandler<?,?> setCustomFunctionHandler(BeanResolverStrategy beanResolver) {          return new CustomFunctionHandler<>(beanResolver);      }      @Bean(name="petclinic.\_process$execute?fn=\_setByRule")      public SetByRuleFunctionHandler<?,?> setByRuleFunctionHandler() {          return new SetByRuleFunctionHandler<>();      }  } |

Custom implementation example:

CustomFunctionHandler.java

|  |
| --- |
| public class CustomFunctionHandler<T,S> implements FunctionHandler<T,S> {      public CustomFunctionHandler(BeanResolverStrategy beanResolver) {}      @Override      public S execute(ExecutionContext eCtx, Param<T> actionParameter) {          /\*\*           \* Add custom function handler logic here           \*\*/          return null;      }  } |

Usage: **@Config(url="/vpOwners/vtOwners/vsSearchOwnerCriteria/vfSearchOwnerCriteria/vbgSearchOwner/\\_process?fn=\\_custom")**

**Notes**

* Naming convention for the bean name should to be followed for the custom function handler to be recognized by the framework - **<default>.\\_process$execute?fn**
* <default> is used for the default implementation of framework. It can be overwritten by specific implementations of the function handler by replacing <default> with ex. petclinic

**Nimbus Search Filtering**

Nimbus uses query dsl syntax to construct filtering queries. The framework uses groovy parser to parse the criteria and replace with the required domain name from the library.

**Usage Examples**

Lets take the example domain named Test with an alias "testdsl".

Test.java

|  |
| --- |
| @Getter @Setter  @Domain("testdsl")  @Repo(Database.rep\_mongodb)  @Execution.Input.Default  @Execution.Output.Default  @ToString  public class Test extends IdString{      @Ignore      private static final long serialVersionUID = 1L;        private String name;      private Integer age;      private LocalDate startDate;  } |

**Criteria**

If the requirement is to search for all records that have age between 18 and 21, the criteria will be as below

&criteria=testdsl.age.between(1,11)

The &criteria above denotes a query parameter that will be part of a service task in the BPMN as shown in the below example.

If the requirement is to search for records those have names beginning with 'a' and age between 18 and 21, the criteria is as below

&criteria=testdsl.age.between(18,21).and(testdsl.name.startsWith('a'))

If the requirement is to search for records those have names beginning with 'a' or age between 18 and 21, the criteria is as below

&criteria=testdsl.age.between(18,21).or(testdsl.name.startsWith('a'))

If the requirement is to search for records that have a start date as today’s date, the criteria is as below

&criteria=testdsl.startDate.eq(todaydate)

Please note that 'todaydate' is a keyword that is used by framework’s groovy parser to parse it to actual date.

**Projection**

The framework provides for support for certain projections. If the requirement is to fetch the count of all records

If the requirement is to get the count of records that have a start date as today’s date, the criteria is as below

&criteria=testdsl.startDate.eq(todaydate)&projection=count