COI Demo Overview

The COI Demo, when finished, will allow clinical researchers to apply eligibility criteria to patient records returned in their request. The demographic, SDTM and Drug Ontology (DO) trees allow the specification of criteria into either inclusion or exclusion tables. Each table has support for linking parameters together. Eligibility criteria specifications can will be represented as sparql queries and can be executed over limited patient datasets.

The COI Demo is implemented using ICEfaces JSF components and third party Java API’s. A detailed description of each component and its relationships with other components is listed below.

(Note: the naming conventions associated with this guide are based on those present in the “COI Demo Model” image)

Inclusion and Exclusion Tables

Each of the inclusion and exclusion tables are implemented in JSF code using an ice:dataTable tag and have an arraylist of TableItems in the TablesBean as the iterative model bound to it.

Each TableItem has fields for the item’s ID, criteria, constraints, a Boolean flag whether it should be added to the link caused when the link button is pressed, the link ID of the link it belongs to (if applicable) and its query variable in the latest sparql query generated.

Each row of both tables is populated by some of the fields of the TableItem corresponding to the row. Each row also includes a delete button to remove unwanted criteria.

The underlying TablesBean also manages all links between criteria.

SDTM and DO Trees

Each of the tree structures is implemented in JSF code using a standard ice:tree tag with a command link as the iterative child component of the ice:treenode tag and an actionlistener to detect expansion events for lazy loading.

The tree model (SDTMTreeModel or DOTreeModel) for the tree structure is based on results returned from the underlying tree model builder (SDTMModelBuilder or DOModelBuilder) and is expanded when an unloaded node is expanded, identified using the tree’s actionListener.

The underlying model builder contains Jena models of the N3 ontologies the trees are based on and is able to query the ontologies to find child elements of expanded nodes.

When the command link of an associated child node is selected, the link’s label is stored. In the case of a link being clicked in the Drug Ontology Tree, a query is also performed of the inherent properties of the drug selected and any results are placed in the property section. When the “apply” button of the tree is clicked, the label of the last node clicked is added to the currently selected inclusion/exclusion table with a text field for its constraints.

Dynamic Interface Component

The dynamically created tree is implemented in the JSF code through binding a PanelGrid created programmatically to a PanelGrid tag.

The PanelGrid created programmatically is also modified programmatically to give the appearance of expanding/contracting nodes. The model for the PanelGrid component is expanded in a similar manner to the SDTM and DO tree models, however, much more information must be provided about each node. Information about each nodes element form or child element forms as well as default values and relationships to other nodes is passed in to the underlying model builder from the ontology and a complex, though likely improvable, algorithm creates element moulds to return to the underlying model. The model is then able to construct the appropriate iceFaces elements, attach them to the appropriate nodes and manage their display in the PanelGrid.

TreeNodes are simulated through an extension of the HtmlCommandButton class that contains additional parameters for expansion and loaded state, a collection of child components and a reference to the parent element.