2019 BIBFRAME Workshop - Stockholm, Sweden

https://ld4p.github.io/bf-workshop-2019/lde-edit-rdf/

Sidestepping the graph - Sinopia Linked Data Editor's approach for editing RDF

Background

The Sinopia's public facing linked data editor, available at https://sinopia.io/, constructs forms for creating and editing RDF based on resource templates properties defined in the Library of Congress derived Profiles. The editor's use of a more modern Javascript React user interface library coupled with the Redux library for application-state management allows for the dynamic creation of valid RDF triples that are then saved through an API call to the Linked Data Platform Trellis. This approach simplified the implementation of the editor by eliminating the need for complex SPARQL statements for querying and updating a RDF triplestore.

Profiles and Resource Templates

Sinopia generates HTML forms for creating and editing linked data based on extending of the Library of Congresses Profiles used in the BIBFRAME Editor and Profile Editor projects. Profiles, as implemented in the BIBFRAME Editor, are JSON files that contain one or more resource templates.

Resource Templates

Each resource template includes an identifier, information on who created the resource template, when it was created, a description, a URI used to create a triple of the RDF type, and a list of property templates.

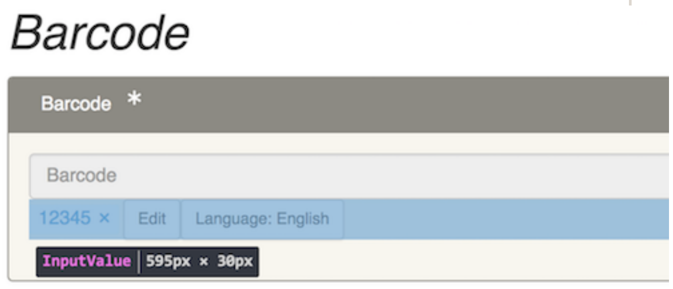
```
"resourceTemplates": [
      {
        "propertyTemplates": [
          {
            "mandatory": "false",
            "repeatable": "true",
            "type": "literal",
            "propertyURI": "http://id.loc.gov/ontologies/bflc/cataloge
            "propertyLabel": "Your cataloger ID",
            "resourceTemplates": [],
            "valueConstraint": {
              "valueTemplateRefs": [],
              "useValuesFrom": [],
              "defaults": []
            }
          }
        ]
    }
}
```

React

An open-source project sponsored by Facebook, React is a Javascript module for building user interfaces. Early on, Sinopia adopted React as a way to dynamically generate the HTML elements for creating and editing linked data.

Components

Most of the React components in Sinopia are pure functions that either generate HTML elements, css classes, and behavior or provide a collection-level container for other React components. For example the `InputValue` component, pictured below



The source code for this component is available at https://github.com/LD4P/sinopia_editor/blob/master/src/components/

In this code snippet from that file, shows the `InputValue` component being defined as a `const` with an important data structure `props` that are properties of the component. The next two lines set two constants, `isLiteral` and `label` that are themselves one-line functions that return conditional values when called by the component. Similarly, the `const handleEditClick` wraps two function calls that change the language and remove an item.

```
const InputValue = (props) => {
  const isLiteral = typeof props.item.content !== 'undefined'
  const label = isLiteral ? props.item.content : props.item.uri

  const handleEditClick = () => {
    props.handleEdit(label, props.item.lang)
    props.removeItem(props.reduxPath)
  }
```

Next these functions are tied and rendered in HTML with the **return** statement below:

```
  return (<div id="userInput">
    <div
      className="rbt-token rbt-token-removeable">
      {label}
      <button
        onClick={() => props.removeItem(props.reduxPath)}
        className="close rbt-close rbt-token-remove-button">
        <span aria-hidden="true">×</span>
      </button>
    </div>
    <button
```

```
        id="editItem"
        onClick={handleEditClick}
        className="btn btn-sm btn-literal btn-default">
        Edit
      </button>
      { isLiteral ? (<LanguageButton reduxPath={props.reduxPath}/>)
    </div>)
  }
```

Redux

Thinking about the transformation of the Profiles with Resource Templates being transformed into a client-side editor but we still need a way associate any changes made by the cataloger in the Linked Data Editor made with the values in React components so that we can do such things as generation of RDF validations, and updating the backend Sinopia Server.

To capture the current data of the application's React components and to build …

State to RDF (and back again)

Quick `sinopia:hasTemplate` solution