Seneca
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

BTI425

Web Programming for Apps and Services

HTML Form patterns

The purpose of this note is to provide guidance about the typical HTML Form patterns that are used in apps. This guidance will work for React apps and for Angular apps.

How to handle common use cases

When designing an interaction of any kind, we must use HTML Forms technology. It doesn't matter whether the task is data entry, or sending a message, or "liking" something, or posting a video, the task is done with forms.

It is also important to know and understand the kind of interaction. There really are only five kinds:

1. Get all (or get some, filtered)
2. Get one
3. Add new
4. Edit existing
5. Delete item

All interactions can be covered by one of these kinds. This realization also applies to related / associated data scenarios.

The following sections briefly show how to implement these interactions. You can use the guidance to write the logic in your React or Angular components.

> *Make sure that you use the React or Angular specific guidance when designing and coding your forms. The guidance in this note is general, and you must build upon it with React or Angular forms techniques.*

> *Note: The guidance is influenced by common data-entry form scenarios.*
> *To implement some of the tasks named above (e.g. "liking" something, posting a video, etc.), the general approach and logic will be the same, but you'll probably do something a bit different and more appropriate for the task.*

Get all

- aka "List"
- URL is typically a collection name
- result of the HTTP GET is an array usually
- table
- each row can have links for details, edit, delete
- page has "add new" link

Get one

- aka "Detail"
- URL is typically a collection name plus an identifier
- result of the GET is an object usually
- description list (dl>dt+dd)
- can have a link for edit and delete
- can have a link for back (or the get all collection)

Add new

- aka "Create"
- URL is typically a collection name
- two-step process, 1) data entry form, then submit, and 2) result page/view
- result of the POST is an object, as stored by the web service
- causes a redirect to a result page/view, "Detail" (above)
- this is known as the "PRG" pattern - Post, Redirect, Get

Edit existing

- aka “Edit”
- URL is typically a collection name plus an identifier
- two-step process, 1) data entry form, then submit, and 2) result page/view
- result of the PUT is an object, as stored by the web service
- causes a redirect to a result page/view, “Detail”

Delete item

- aka “Delete”
- URL is typically a collection name plus an identifier
- two-step process, 1) data form, then submit, and 2) result page/view
- the data form can be the content of the “Get one” (“Detail”) component above
- result of the DELETE is unimportant (we don’t want to leak info)
- causes a redirect to the “List” page/view
