Handle UI State and Exceptions with Angular Data Binding

Paul D. Sheriff
PRESIDENT, PDSA, INC.

@pdsainc www.pdsa.com psheriff@pdsa.com

Goals

Create object literal for page state

Create constant for each mode

- List, Add, Edit

Create setUIState function

- Call to change page state

Learn to handle exceptions

Handle UI State

Staying Consistent

Going to handle UI state manually

Could have used Angular routing

Could have split out parts of page

Want to stay consistent with MVC page

Show use of ng-show and ng-hide

```
const pageMode = {
  LIST: 'List',
  EDIT: 'Edit',
  ADD: 'Add'
};
```

Add constant named pageMode

Used for which “mode” the page is in

- List
- Edit
- Add

Use to set “isVisible” properties

```
vm.uiState = {
  mode: pageMode.LIST,
  isDetailAreaVisible: false,
  isListAreaVisible: true,
  isSearchAreaVisible: true,
  isValid: true,
  messages: []
};
```

Create object literal for page state

Initialize page mode to “List”

Initialize all “isVisible” properties

- Data bound to HTML elements

Whether or not the model is valid

- Use to display messages

An array of messages to display

```
// Set appropriate UI state
function setUIState(state) {
  vm.uiState.mode = state;

  vm.uiState.isDetailAreaVisible =
    (state == pageMode.ADD || state == pageMode.EDIT);
  vm.uiState.isListAreaVisible =
    (state == pageMode.LIST);
  vm.uiState.isSearchAreaVisible =
    (state == pageMode.LIST);
}
```

Create setUIState() function

Pass a pageMode constant

Set the vm.uiState.mode property

Set “isVisible” properties based on mode

UI State Things to Do

Set isValid in handleException

Call setUIState from various functions

Create click events on HTML

Remove Razor code

Bind to ng-show and ng-hide

Demo

Handling UI State

Exception Handling

```
function handleException(error) {
  vm.uiState.isValid = false;
  var msg = {
    property: 'Error',
    message: ''
  };

  vm.uiState.messages = [];

  switch (error.status) {
    case 404:  // 'Not Found'
    case 500:  // 'Internal Error'
    default:
      msg.message = 'Status: ' +
          error.status + ' - Error Message: '
          + error.statusText;
      vm.uiState.messages.push(msg);

      break;
  }
}
```

Modify handleException function

Create messages object literal

- ‘property’ and ‘message’ properties

Clear messages array

Handle various HTTP status codes

Add error messages to uiState object

```
case 404:  // 'Not Found'
   msg.message = 'The product you were ' +
                 'requesting could not be found';
   vm.uiState.messages.push(msg);

   break;
```

404 Error

- If the requested product is not found

Modify message property

Add message object to messages array

```
case 500:  // 'Internal Error'
  msg.message =
    error.data.ExceptionMessage;
  vm.uiState.messages.push(msg);

  break;
```

500 Error

- Server throws an error

Get error message from server

- error.data.ExceptionMessage

Add message object to messages array

```
<ul>
  <li ng-repeat="msg in vm.uiState.messages">
    {{msg.message}}
  </li>
</ul>
```

Bind messages to <li> elements

Use ng-repeat

Display message property

Demo

Handling exceptions

Summary

Removed a lot of server-side rendering

Removed some jQuery/JavaScript

Learned about ng-hide and ng-show

Displayed exception messages

Coming up in the next module...

- Get rid of Razor code
- Get a single product
- Add, edit, and delete products