Worklight Assigments

Free-form Class Activities

Overview

- These activities encourage students to exercise and consolidate their Worklight knowledge
- Some use as a starting point an application built by Roland Barcia, IBM Distinguished Engineer, CTO Mobile and WebSphere Foundation
- The application uses Ionic and AngularJS, no previous knowledge is required
- We start with a skeletal version of the application and incrementally exploit Worklight

Checkpoint 1

- Create a Worklight project, containing a Hybrid Application
- Add an Android Environment
- Deploy and run app in Browser Simulator
- Use Browser tools to show console
- Add to your application's initialisation
  - WL.Logger.debug("some message")
- Use non-build refresh to verify Logger output in console

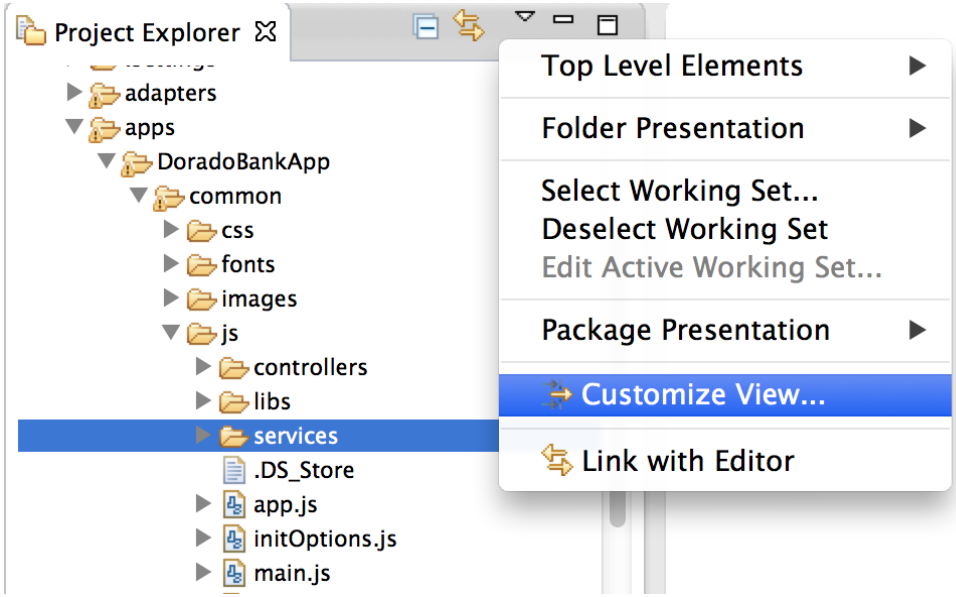
Outline

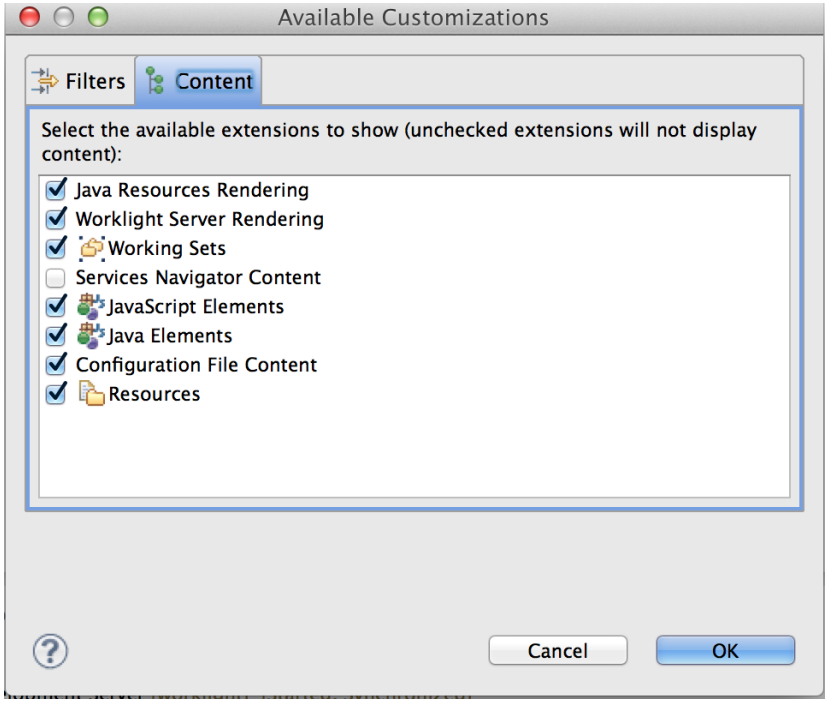
1. Deploy and Run the Skeleton
   - Understand the project structure
2. Worklight Client APIs
   - Display simple data items
3. Adapters
   - Build adapter, display master/detail
4. Authentication
   - Add adapter-based authentication
5. Connection APIs
   - Display connection status
6. Offline working
   - Cache master/detail data in JSONStore
7. Extra Credit: your own app
   - Choose a business domain, design and implement app

Step 1 – The Skeleton App

- Import and verify Data Project (add to Development Server)
  - http://localhost:10080/mwdserverstub/rest/accounts/accounts.json
- Import and verify App project
  - Login rbarcia/bl0wfish (in initial version, no credentials checked)
  - Observe
    - Navigation
    - Master/Details display for Accounts & Transactions
- Explore code:
  - Note: make services visible (next slide)
  - Routes, Views, Controllers, Services (see AngularJS next slide after that)
  - Consider: in a large project team what division of labour?
- Data display
  - How are Error Messages displayed?
  - How are lists of data (list of accounts or transactions) displayed?
- Debugging
  - Set breakpoint in code, examine values

Services Folder Contents





- Project Explorer View, select down arrow for context menu
- Customize View
- Unselect Services Navigator Content

AngularJS – an emerging framework

- jQuery alone not enough, need a framework
  - Dojo has dojox/app playing a similar role
- Key features (find these in the code)
  - Index.html: explicit pre-load of all .js files
  - app.js: routing from “page” to “page”, loads view html files, associates controller with page (if any)
  - services obtain data, async, so use promise
  - Bi-directional mapping between HTML and js data

```
<div>{{ account.name}}</div>
```

Step 2- Worklight Client APIs

- Choose one of unpopulated Views
- Amend to display data from WL.Client APIs
  - Language
  - Environment

Step 3 - Adapters

- Implement HTTP adapter procedures calling data service
  - getAccounts()
  - getTransactions(accountId)
- Test adapters in Development Environment
- Replace hard-coded values in client with calls to adapters (see idiom on next slide)
- Test client

Using results of an async call

- Promise idiom now pervasive in JavaScript
- All service functions here return a promise

```
busyIndicator.show();
fnReturningPromise().then(function (someData)
                        {
                              $scope.accounts = accounts;
                              $scope.$broadcast('scroll.refreshComplete');
                              busyIndicator.hide();
                              $scope.errorMsg = "";
                        },
                        function(error)
                        {
                              $scope.errorMsg = "Could Not Load Accounts";
                              $scope.$broadcast('scroll.refreshComplete');
                              busyIndicator.hide();
                        } );
```

Step 4 - Authentication

- Add adapter-based authentication methods to the adapter
- Configure authentication, apply to adapter methods
- Determine client behaviour if not authenticated
  - Examine network traffic in browser debug
- Add challenge handler and authentication code in client

Step 5 – Offline detection

- Display current connection status
- Display status change in response to connection events

Step 6 – Offline working

- Store data retrieved from Adapter in JSONStore
- Amend application to query JSONStore if it is started offline

Extra Credit

- Choose a business domain
  - Example: Health Care
- Sketch some use cases
  - Example: Request/View Lab Results for Patient
- Design Views / Data
- Design Adapters for notional payloads
- Implement
  - Use Dorado Bank as starting point