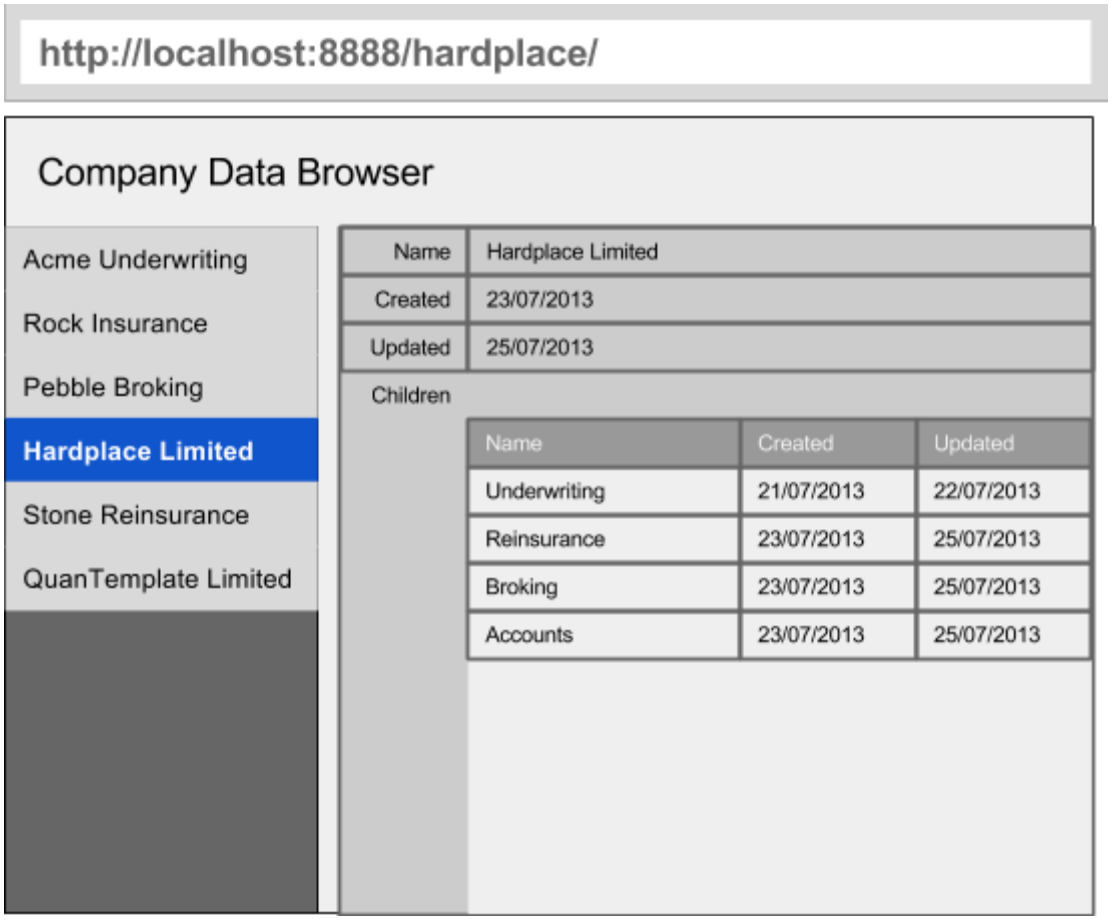
JavaScript Middleware Exercise

Purpose

The aim of this exercise is to test your ability to work with client-side data and your ability to work with client-side APIs (e.g. pushState and history management APIs) to successfully implement a very simple single-page application for browsing company data.

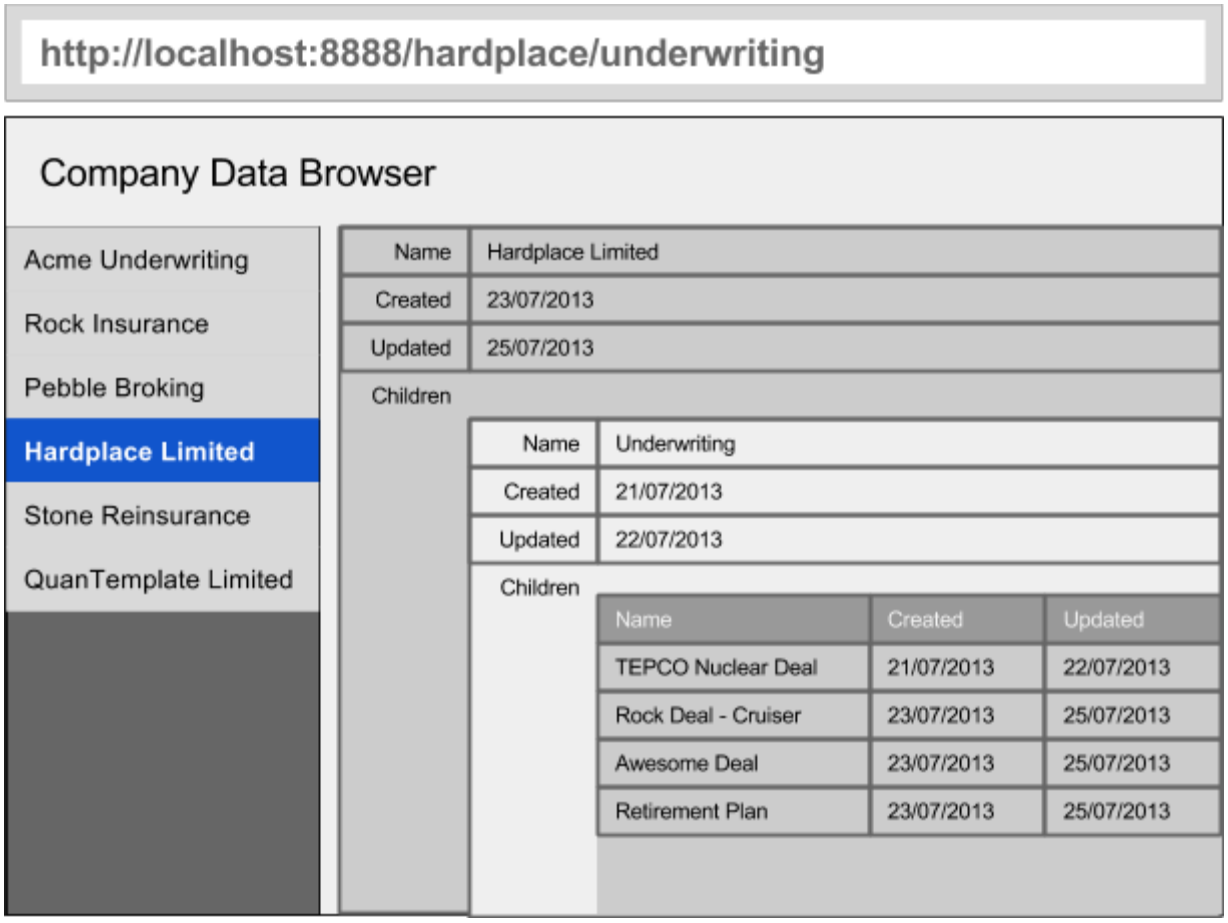
Overview

The application is required to display hierarchically structured information about insurance companies. The following figure gives an overview of the required application:



The left panel, is for selecting the organisation whose data is to be browsed. The right hand panel shows the root document for that organisation and some metadata about creation and update times.

The ‘Children’ table shows child documents belonging to that organisation. Clicking on a row in the table should replace the table with a view of that document and it’s children. As follows:



As navigation proceeds the URL should be updated to reflect the location within the document structure. For example, in the preceding diagram, we've navigated to the underwriting document, so the URL reflects "hardplace/underwriting".

The Underwriting document has its own children and those rows should in turn be clickable, replacing the table section with a full document as before, and further impacting the URL, e.g. clicking on 'TEPCO Nuclear Deal' would navigate to 'hardplace/underwriting/tepco' etc.

Clicking on a parent document will navigate back to the view of that document, impacting the URL accordingly.

Implementation Notes

Data

- The data is provided to the application as a single, large JSON file, here. The data within should be loaded into a set of Backbone Models and Collections as required at application startup.
- For simplicity all views of the data need only be readonly.
- No sorting or ordering of data is necessary, it is fine for data to be ordered as it in the

JSON.

Navigation

- You must NOT use Backbone or any other library's controller mechanism, the purpose of this exercise is to write your own pushState history controller.
- The back button of the browser should function properly.
- Navigation should be progressive, the entire page should not re-draw on each navigation event. Instead, only those views that are additionally required should be added to the page, and any no-longer required views should be removed.

Target

- You only need to target Google Chrome, there is no-need to worry about cross-browser compatibility.
- The submitted solution should be a single HTML page or a directory structure that can be statically loaded by Chrome.

3rd Party Libraries & Tools

- You may use Backbone (with exception of its Router and History classes), Underscore, jQuery and that's its.

Language

The solution should be implemented in JavaScript, HTML5 and CSS. You may use CoffeeScript and LESS during development if it makes things easier, if you do, please also provide the source code.

Assessment

You will NOT be assessed on the aesthetic quality of your work, although it has to be clean and functional enough to assess.

You will be assessed on the structure of your code, how well it solves the problem as specified and of course whether it works or not at all. It will also be assessed on how easy it is to understand and use as the nature of middleware is that other developers necessarily build on top of it.

Questions

If you have any questions whilst carrying out this test, please send them to the following email address.

andre.brantom@quantemplate.com

We'll do our best to point you in the right direction.