Technical Test

Implementation details

This deck provides:

Location of deployed implementation:

<happy to create instance on request>

Location of code:

https://github.com/elendil-src/JSONPublisher/tree/master/src/main

Known defects:

1) a search term containing internationalised characters (e.g Rôle) does not match valid terms; anglicised filter term (e.g Role) does match internationalised terms. Problem is only on Amazon Linux; solution works correctly on Windows10

Follow slides identify key decisions made during implementation.

Brief

- In a language of your choice create an application which uses the attached JSON Graph as a data store.

- The application must provide the following capabilities and be suitable for other developers to enhance and improve upon in the near and distant future.
  - A display of all the product titles
  - The ability to search for a product based on product title

- Host and make your application available in a cloud provider of your choice.

Functional Requirements

1. As a Librarian, I wish to view all Product Titles in the Catalogue in order to understand extent of the catalogue and data available.

    a. When a Product is displayed I need to view Display ProductTitle, ProductId, and ProductForm
    b. Products to be ordered in ascending order by ProductId

2. As an Librarian I wish to view all Products whose title partially matches a given term in order to find a specific Product

    a. **Given** a matching partial term **when** performing a search **then** all Products whose titles that partially match the partial term should be visible.

    b. **Given** a non-matching partial term **when** performing a search **then** no should be visible.

    c. **Given** a matching partial term **when** performing a search **then** all Products whose titles that partially match (ignoring case) the partial term should be visible.

    d. **Given** a matching partial term **when** performing a search **then** all Products whose titles that partially match (ignoring leading and trailing spaces) the partial term should be visible.

    e. **Given** a matching partial term **when** performing a search **then** all Products whose titles that partially match (ignoring accents, diacritics and similar) the partial term should be visible.

    f. **Given** an empty partial term **when** performing a search **then** all Products should be visible.

Solution Related Decisions

1) Architecture: used web application architecture to satisfy viewing at a different location
2) Use Java/JSP/Tomcat/Linux as target language & stack: in common use, I have some understanding & it is supported by cloud providers
3) Use AWS Elastic Beanstalk with pre-configured platform of Tomcat & Java: free & simple to setup and administer
4) Use IntelliJ IDEA community as IDE: common, free and more able of IDEs
5) Use Maven to build and deploy to local Tomcat instance to accelerate development
6) Source control: Use Git (local and Github to make source available)
7) Design approach: scaling and security are not important; support/extensibility of design and implementation are.
8) Security: keep simple - infrastructure: use root account on AWS; HTTP not HTTPS: no user authentication; secure JSP servlets
9) Use Model-View-Controller: provide separation of concerns and a known design pattern for other developers to understand and extend from.
10) Implement a user session to support simultaneous users; but a single model
11) ProductModel class is responsible for complexity of business domain including complex searching and matching. Validates JSON & product data integrity. Use DOM API to traverse tree to retrieve data; simpler than streaming (and binding cannot be used);
12) Usability: pages/views should appear be intuitive and coherent, but not professional. Target browser/device is Microsoft Edge/1024x768. Other modern browsers should work.
13) Java Server Pages: keep simple and focus on scriptlets (not JSTL)
14) Error handling approach: keep simple (JSP error page, exceptions etc, validation JSON as a lazy reader)

Not used due to lack of time: Automated Unit testing; automated functional testing; used of JSTL instead of JSP; more elaborate user interface ; Javascript controls that could display Products in more user friendly way; AWS EB CLI for deployment; explore patterns that decouple ProductModel from Product POJO.