duckcreek technologies

Exercise: Product Definition OptionLists

We saw an option list in an earlier lab, but this goes into more detail and shows some of the rich features available.

Java-Based Labs

Make the following changes for the WorkersComp Full Quote Transaction

Custom List Design Approach

In this exercise we'll use the `OptionsMap` class to retrieve a static (xml-based) list and hand back a subset of that as our custom list.

While it may not be used in every AgencyPortal implementation, it is not uncommon to take the approach of using Static lists as the basis for Custom list, where the custom list implementation provides business logic that might combine different Static lists in different combinations, and/or (as in this example) filter the list(s) in some way.

In general terms, the basis for the list data could be acquired from any source, so instead of a Static list via `OptionsMap` you might use a SOAP call or a JDBC database call.

Specification for the new Custom List

We're going to implement a custom list to the "Billing and Contact Information" (formerly "Billing/Contact Information") > "Billing/Audit Information" > "Audit Frequency" field. We simply want to omit any of the current list's values that start with the letter 'Q' (eliminating the Quarterly option).

Don't take the example too literally, but it's conceivable that a business rule would specify that Quarterly Audits are only available/necessary based on the particular coverages and limits or rate classes or location states chosen by the user. It would be a small step to add the appropriate business logic for conditional filtering.

duckcreek technologies

Implementing the new Custom List

1. In the TDF definition of the Audit Frequency field, disable the existing reference to the `audit` optionlist

   That is, comment out the specified optionList element, rather than removing it; we'll want to reference it when we write the new code

2. Create a new Java class

   - Package (create it):

     `com.sdktraining.workerscomp.codelists`

   - Class:

     `AuditFrequencyNoQ`

     - Implement the proper Interface as described in the presentation materials for this module

- The `getDefaultValue()` method can be used to specify a default value for the user

  - Java enables us to optionally implement business logic to determine the default value (change it conditionally based on context)

  - We have no spec for dynamic determination of the default value, so the default implementation is all we need…

```
return null;
```

- The `generate()` method is used to produce the optionlist (the set of data value + display text pairs)

  - As described in the presentation material, account for the `existingListToUpdate` parm, insuring that your method return value includes any values already in that list (if any)

  - The `OptionsMap` class exposes the `getSingleList()` static method which takes an instance of `OptionList.Key` – use the information in the commented-out (prior) TDF optionlist element to fill the `Key`

- Iterate over the items in the static `audit` list and conditionally add them to your `returnList` per the specification (above)

3. Configure the "Audit Frequency" field to use the new custom list implementation

Advanced Exercise:

- What happens if you create a local `OptionList` instance and add the pared down `audit` values to it (and return it), instead of accounting for the `existingListToUpdate` parm passed in?