duckcreek technologies

Exercise: DTR/IPDTR

Exercise 1: (DTR) Make Additional Company Info required if Years In Business is less than '5'

In the previous XML DTR lab exercises, you created behaviors that operated for all instances of a particular transaction. In this lab, you will create a behavior that operates conditionally, based upon the value of a field stored in the PreConditions data associated with the work item. Specifically, you are going to store the value for the Date Business Started field on the Agency/Applicant Information page and make the Additional Company Info field required if the (computed) Years In Business value is less than 5.

1. Add uniqueId for Date Business Started

   In the workerscompCommons pageLibrary, find the Date Business Started field. Note this fieldElement does not include a uniqueId. Set the uniqueId to "DateBusinessStarted".

2. Add Behavior to workersCompBehavior.xml

   Add the following to the workerscompBehavior.xml file, using the appropriate <for> spec to alter the Additional Company Info field:

```
<behavior>

   <do action="alter">
      <property name="required">true</property>
   </do>

   <where>
   <preCondition name="DateBusinessStarted">!</preCondition>
   </where>

   <for>*/<page_id>/<pageElement_id>/<AddtionalCompanyInfo_id></for>

</behavior>
```

   The "!" in the precondition indicates we're for anything but an empty value. The behavior as written will be in effect if anything is entered in the Date Business Started field.

3. Initial Test

   Deploy the update and refresh the page. The Additional Company Info should always be required, regardless of the DateBusinessStarted value.

4. Add YearsInBusiness computed PreCondition

5. In the WorkersCompPreConditions class, add the following method:

```
public Object getFieldValue( String preConditionFieldName, String defaultValueIfNotPresent,
WhereClause whereClause ) {

return super.getFieldValue(preConditionFieldName, defaultValueIfNotPresent, whereClause);
}
```

This method will be called each time a precondition needs to be evaluated, for example, when the page is being rendered. The default return value for this method should always be the call to super.getFieldValue as shown. We will override this method to compute the YearsInBusiness value.

Add an "if" statement to check for a preConditionFieldName of "YearsInBusiness". For this field only, call super.getFieldValue to get the value of the DateBusinessStarted field. Use the APDate class to convert this to a date. Use the DateMath class to calculate the difference in years between this and the current date. Return the result as a String.

6. Update the behavior to reference YearsInBusiness

   Change the behavior you previously added to include the YearsInBusiness precondition, checking for less than five years.

```
<where>
<preCondition name="DateBusinessStarted">!</preCondition>
<preCondition name="YearsInBusiness">lt(5)</preCondition>
</where>
```

7. Final Test

   Deploy the updates. Enter a Date Business Started more than five years ago. Verify the Additional Company Info field is not required. Change the Date Business Started value to within the last five years. Verify the Additional Company Info field is required.

   Try removing the reference to the DateBusinessStarted precondition in the behavior. Deploy the update and retest as described above, entering dates more than and less than five years ago. How does the behavior change? Try testing this altered behavior with a new workitem.

Exercise 2: (IPDTR) Make Additional Company Info required if Participating Plan is No

NOTE: Prior to working on this lab, restore the solution for Lab 1 to a functional state (the "Final Test" was to temporarily break the behavior).

duckcreek technologies

In the prior lab, we used "plain" DTR, where the precondition data input is on a *prior* page, different than the behavior target as specified in the <for> clause.

In this lab, we'll use IPDTR: Intra-Page DTR, where the precondition data input is on the *same* page as the target specified in the behavior for clause.

This lab will effectively create an "OR" behavior vs. exercise 1- if either condition specified in the behaviors for Lab 1 or Lab 2 is true, the Additional Company Info will be altered to required. The field will not be required *only* if both behaviors' preconditions evaluate to false.

Initial Setup:

1. Copy the behavior from Exercise 1.

2. Change the <where> clause to reference the Participating field (in the same section as Additional Company Info).

3. Set the preconditions value expression to the data value for "No" selection in the Participating field.

Test the new precondition:

1. Reset the Date Business Started (1st page) to 5+ years before the current date (in effect, disabling the behavior from Lab 1).
2. In the Policy Information page, set the Participating field to "Yes". Verify the Additional Company Info is NOT required.
3. Set the Participating field to "No". Verify the Additional Company Info IS required.

Did the tests fail? If so, repeat the test, but Save the page each time you change the Participating field selection. The required setting of the Additional Company Info field should be reflecting the Participating field selection after the page is saved.

IPDTR requires an additional setting so that a change to the field referenced by the behavior preconditions clause will have immediate impact on other fields in the same page, without requiring an explicit Save (submit of page data).

To determine the additional setting required for IPDTR support, refer back to the presentation or add'l info in the DTR module, or existing pages that use IPDTR (generally: Policy Effective + Expiration Dates in any transaction type). Test the change and verify the Additional Company Info indicator is updated as soon as the Participating field selection is changed, without requiring a page submit.

duckcreek technologies

Exercise 3 (Advanced): Make Additional Company Info NOT required if Controlling State is Massachusetts

In Exercises 1 & 2, two different behaviors were setting the Additional Company Info field to required based on different conditions, effectively creating a logical OR- the required flag is changed if either behaviors' precondition evaluates to true.

In this Lab, we'll define an alter behavior that conflicts with labs 1 & 2, then resolve the conflict…

1. Copy the behavior from Exercise 2
2. Change the property value in the do…alter to false

```
<do action="alter">
      <property name="required">false</property>
</do>
```

3. Change the <where> clause to reference the Controlling State field; set the preconditions value expression to MA (Massachusetts)

```
<where>
      <preCondition name="CommlPolicy.ControllingStateProvCd">MA</preCondition>
</where>
```

4. Be sure to create the hotfield declaration for Controlling State field so that IPDTR works in-page.

Test the new behavior in combination with the preconditions for labs 1 & 2; record the required setting of the Additional Company Info field for each test condition…

Test Cases:

Condition	Results
Date Business Started < 5 yrs, Participating = No, Controlling State != MA	
Date Business Started 5+ yrs, Participating = No, Controlling State != MA	
Date Business Started < 5 yrs, Participating = Yes, Controlling State != MA	
Date Business Started < 5 yrs, Participating = No, Controlling State = MA	

Does the required setting of Additional Company Info match your expectations?

duckcreek technologies

Try re-ordering the behaviors- in the TDF XML, put the new behavior from this lab 3 above the behaviors for labs 1 & 2 and repeat the above tests. The required setting of Additional Company Info should be inconsistent with prior testing for at least some Test Cases.

The fundamental problem here is that the "spec" is incomplete: the conflicting behaviors (labs 1 & 2 vs lab 3) must either be mutually exclusive (that, is the preconditions for labs 1 or 2 will never be true if lab 3 precondition is true) or the behaviors must account for each others' precondition values in order to get a reliable runtime result- that is, the business requirements must clarify the *precedence* of the different preconditions that are impacting the required setting of Additional Company Info: either…

1. YearsInBusiness or Participating is more important than (overrides/supersedes) Controlling State (in terms of determining whether Additional Company Info should be required)

OR

2. Controlling State is more important than (overrides/supersedes) YearsInBusiness or Participating.

In this case, the conflicting behavior preconditions are NOT mutually exclusive- all of input fields are unconditionally included, and there are no behaviors or validations that change or restrict the range of choices available to the user.

Because the Additional Company Info defaults to not required (as defined in TDF fieldElement, independent of any behaviors), this suggests that having the Controlling State "reset"/restore the required setting to false should take precedence over YearsInBusiness or Participating. To be absolutely certain, we must check w/ business operators (or, the original source of spec or business requirements) / PM / BA, since they are most likely not aware of implementation details and DTR runtime characteristics.

Change the behaviors from labs 1 & 2 to allow the behavior for lab 3 to take precedence- in a sense, labs 1 & 2 behaviors will "defer" to the lab 3 precondition by including a check for Controlling State != MA. Repeat the test conditions above. Re-order the behaviors w/ lab 3 behavior above & below labs 1 & 2 behaviors. The required setting of Additional Company Info should now be consistent, regardless of the order of the behaviors. In this case, do we still need the behavior from lab 3?

In this lab, we've seen the effects of the following runtime DTR subsystem characteristics:

- The precedence of behaviors is determined by rules related to explicit include, exclude or alter. This is covered in detail the training material.

- The order of behavior evaluation is *undefined*, and not related to their position in the behavior XML file relative to each other.

- For conflicting alters *that do not account for each others' preconditions* (as we had w/ labs 1 & 2 vs. the original implementation of lab 3), the "last one" wins, *but* exactly which alter behavior will be "last" (in terms of order of evaluation) is *indeterminate* as noted above *and may be different on different platforms.*

As seen with this lab, the above should not be a problem unless the business requirements / spec are incomplete; because behavior order is indeterminate, conflicting alters with preconditions that are not mutually exclusive *must* account for each others' preconditions in order to guarantee consistent runtime behavior impact on all platforms.

There is some potential for similar issues when different behaviors specify conflicting include vs. exclude specs for the same transaction elements. While the precedence of conflicting include vs. exclude is more consistent and predictable than conflicting alters, there may be cases where these include and exclude behaviors must also account for each others' preconditions, when the business requirements conflict w/ well-defined DTR include vs. exclude precedence.