ADF Code Corner

85. af:query component field validation

twitter.com/adfcodecorner

Abstract:

The af:query component is used in Oracle ADF to declaratively build search forms based on ADF Business Component View Criterias. The af:query component is not configurable to a great extend, and input field validation is one of the areas that cannot be configured declaratively. This article explains how developers can use the af:query component query listener to evaluate dependent fields. The example compares two input date fields, ensuring that the start date is lower than the end date.

Author: Frank Nimphius, Oracle Corporation
twitter.com/fnimphiu
06-JUL-2011

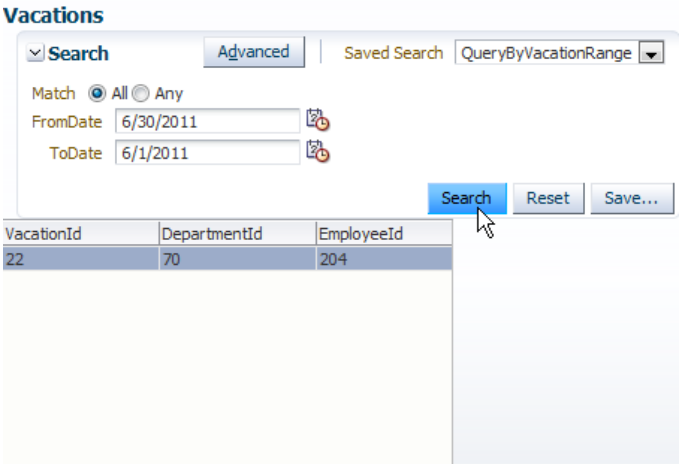
Oracle ADF Code Corner is a loose blog-style series of how-to documents that provide solutions to real world coding problems.

Disclaimer: All samples are provided as is with no guarantee for future upgrades or error correction. No support can be given through Oracle customer support.

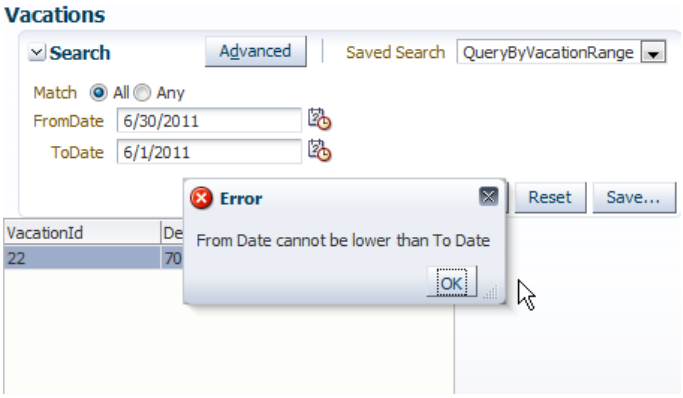
Please post questions or report problems related to the samples in this series on the OTN forum for Oracle JDeveloper: http://forums.oracle.com/forums/forum.jspa?forumID=83

Introduction

The example has two date fields defined for specifying a date range to search vacation requests. To find a request, it has to be within the FromDate and ToDate range.

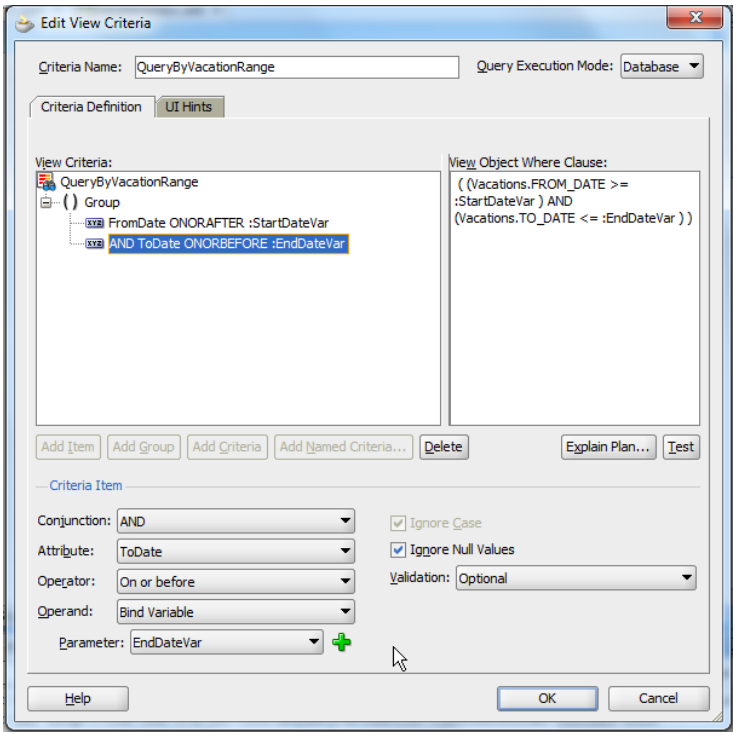


However, no search is performed if the user specifies invalid date criteria, for example when the ToDate value is lower than the FromDate. In this case an error message is displayed as shown in the image below.



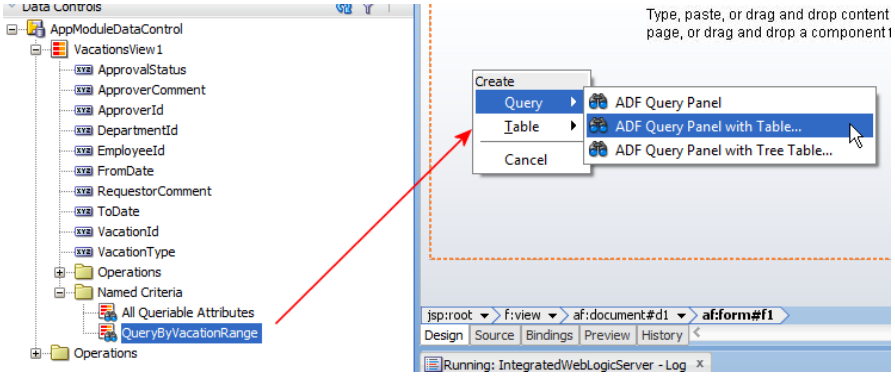
Implementing this solution

To implement this solution, a View Criteria is defined on the *Vacations* View Object. The criteria uses bind variables for the ADF query form to pass the **FromDate** and the **ToDate** attribute values.



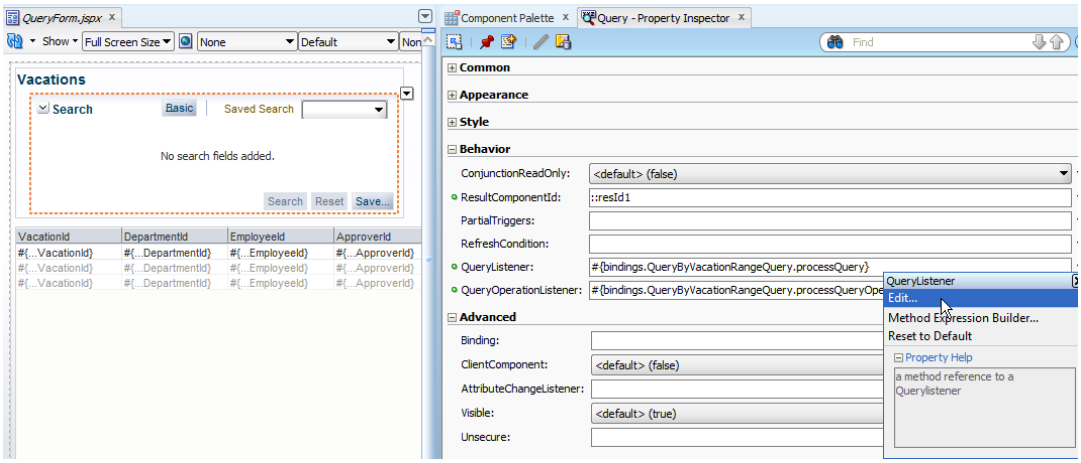
In the design time, the View Criteria is dragged from the Oracle JDeveloper Application Navigator to the page and dropped as **Query | ADF Query Panel with Table**.

The `af:query` component that represents the search form does reflect basic constraints defined on the View Object but doesn't expose complex Java method or Groovy validation. For this, you need to apply the validation on the client component as shown next.



The `af:query` component that gets created to render the search form has a pre-configured **QueryListener** pointing to an ADF search binding. **Copy** the existing string to the clipboard.

As shown in the image below, click the **arrow icon** next to the **QueryListener** property and choose **Edit** from the context menu to create or reference a managed bean to define a custom query listener method.



The custom query listener method, `onQuery(QueryEvent queryEvent)` in this example, has access to the query parameters added by the application user through the `af:query` component search form. Though in this example there are only two query form fields, you can have many more. Each of the fields is accessible by the name of the underlying View Object attribute. The listener code below accesses the two date fields and compares the date values.

```
import java.util.List;
import javax.el.ELContext;
import javax.el.ExpressionFactory;
import javax.el.MethodExpression;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import oracle.adf.view.rich.event.QueryEvent;
import oracle.adf.view.rich.model.AttributeCriterion;
import oracle.adf.view.rich.model.AttributeDescriptor;
import oracle.adf.view.rich.model.ConjunctionCriterion;
import oracle.adf.view.rich.model.Criterion;
import oracle.adf.view.rich.model.QueryDescriptor;

public class VacationsQueryListener {
  public VacationsQueryListener() {}

  //method referenced from the QueryListener property
  public void onQuery(QueryEvent queryEvent) {
    //variable to hold the start and end date
    oracle.jbo.domain.Date startDate = null;
    oracle.jbo.domain.Date endDate = null;

    //the af:component describes the query using a QueryDescriptor that
    //is accessible from the query event. Criterions are used to define
    //the query fields
    QueryDescriptor qdesc =
```

```
                (QueryDescriptor)queryEvent.getDescriptor();
    ConjunctionCriterion conCrit = qdesc.getConjunctionCriterion();

    //access the list of search fields
    List<Criterion> criterionList = conCrit.getCriterionList();

    //iterate over the attributes to find FromDate and ToDate
    for (Criterion criterion : criterionList) {
       AttributeDescriptor attrDescriptor =
                ((AttributeCriterion)criterion).getAttribute();
       if (attrDescriptor.getName().equalsIgnoreCase("FromDate")) {
         startDate = (oracle.jbo.domain.Date)
                   ((AttributeCriterion)criterion).getValues().get(0);
       } else {
       if (attrDescriptor.getName().equalsIgnoreCase("ToDate")) {
          endDate = (oracle.jbo.domain.Date)
                   ((AttributeCriterion)criterion).getValues().get(0);
         }
      }
    }
    //startDate must be lower than end date
    if ((startDate != null && endDate != null) &&
        (startDate.compareTo(endDate) > -1)) {
       FacesContext fctx = FacesContext.getCurrentInstance();
       fctx.addMessage("VacationQueryComponent",
       new FacesMessage(FacesMessage.SEVERITY_ERROR,
                        "From Date cannot be lower than To Date",
                        "From Date cannot be lower than To Date"));
       //immediately render response if ToDate is lower than FromDate
       fctx.renderResponse();
     } else {

       //only if the date fields are enetered correctly, execute the
       //QueryListener search binding using a copy of the original EL
       //string added by JDeveloper upon drag and drop
       invokeMethodExpression(
        "#{bindings.QueryByVacationRangeQuery.processQuery}",
        queryEvent);
     }

  }
}

//helper method to execute the QueryListener EL
private void invokeMethodExpression(String expr,
                                    QueryEvent queryEvent) {
  FacesContext fctx = FacesContext.getCurrentInstance();
  ELContext elContext = fctx.getELContext();
```

```
  ExpressionFactory eFactory =
            fctx.getApplication().getExpressionFactory();
  MethodExpression mexpr =
            eFactory.createMethodExpression(
                      elContext, expr, Object.class,
                      new Class[] { QueryEvent.class });
        mexpr.invoke(elContext, new Object[] { queryEvent });
  }
}
```

Running the Sample

The Oracle JDeveloper workspace for this example can be downloaded as sample #85 from the ADF Code Corner website

http://www.oracle.com/technetwork/developer-tools/adf/learnmore/index-101235.html

You need to configure the database connection to use the Oracle HR schema and also run the **hrExtension.sal** script contained in the **afQueryValidation | sql** folder. The script creates the VACATIONS and VACATION_TYPES tables and populates them with data rows.

From Oracle JDeveloper 11.1.1.5, run the JSPX page and query the single row that is created by the script. You can then try and select a lower ToDate than StartDate in which case an error message is shown.

RELATED DOCOMENTATION

⊠	Jobinesh - Retrieving ViewCriteria from a custom queryListener method http://jobinesh.blogspot.com/2011/03/retrieving-viewcriteria-from-custom.html
⊠	Jobinseh - Programmatically resetting the <af:query> and search result table http://jobinesh.blogspot.com/2011/04/programmatically-resetting-and-search.html
⊠	QueryDescriptor http://download.oracle.com/docs/cd/E21764_01/apirefs.1111/e10684/oracle/adf/view/rich/model/QueryDescriptor.html
⊠	af:queryComponent http://download.oracle.com/docs/cd/E21764_01/apirefs.1111/e12419/tagdoc/af_query.html