DBFIELD

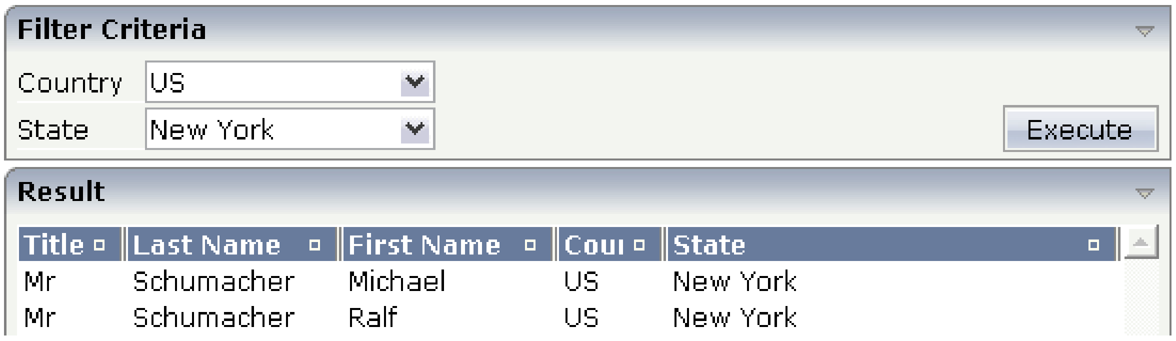
The DBFIELD control represents a filter criterion of a database query. It provides for a value help that is read from the database, a convenient way to append the filter criterion to the SELECT statement, and the ability to reflect a "to-one" dependency between filter criteria.

This chapter covers the following topics:

- Example
- Properties

Example

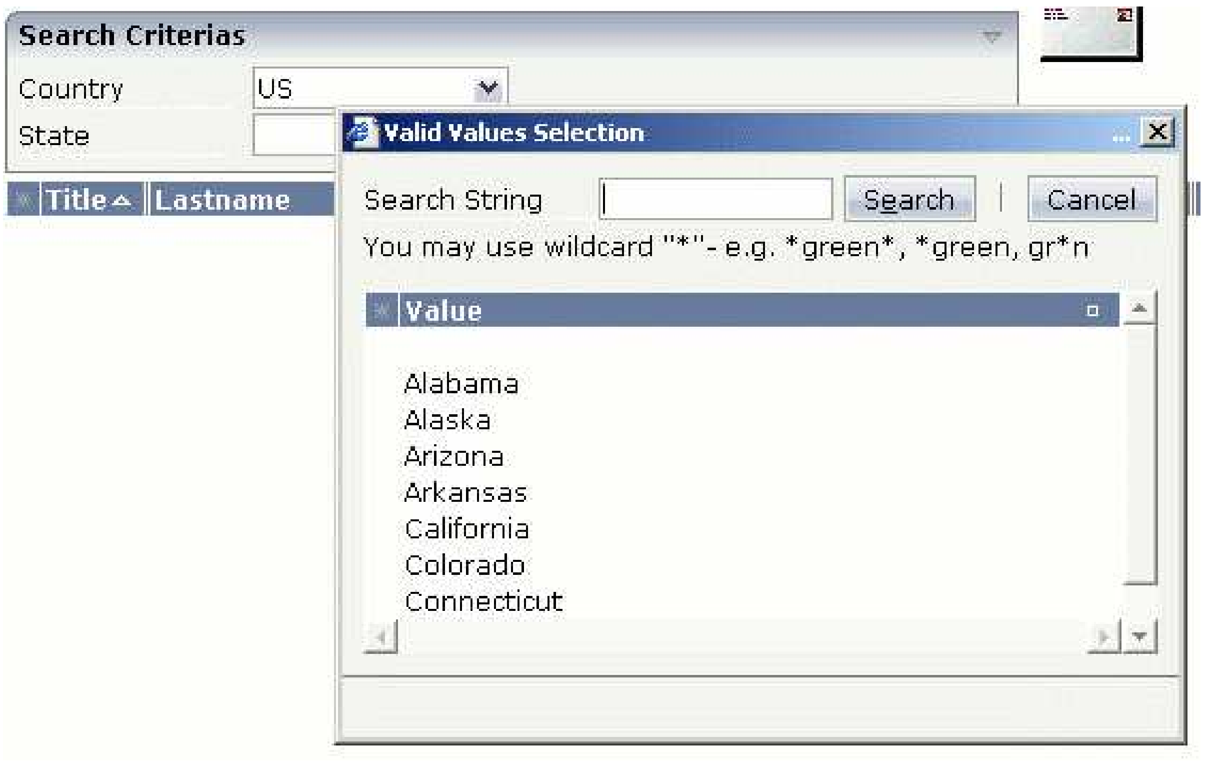
The following image shows an example in which two DBFIELD controls are used for the filter criteria "Country" and "State" within a simple business partner report. Both fields provide for value help. The field "State" is defined to be dependent on "Country". As a consequence, the list of valid values for "State" is country-specific. The result shows partners that reside in state "New York" of the United States of America.

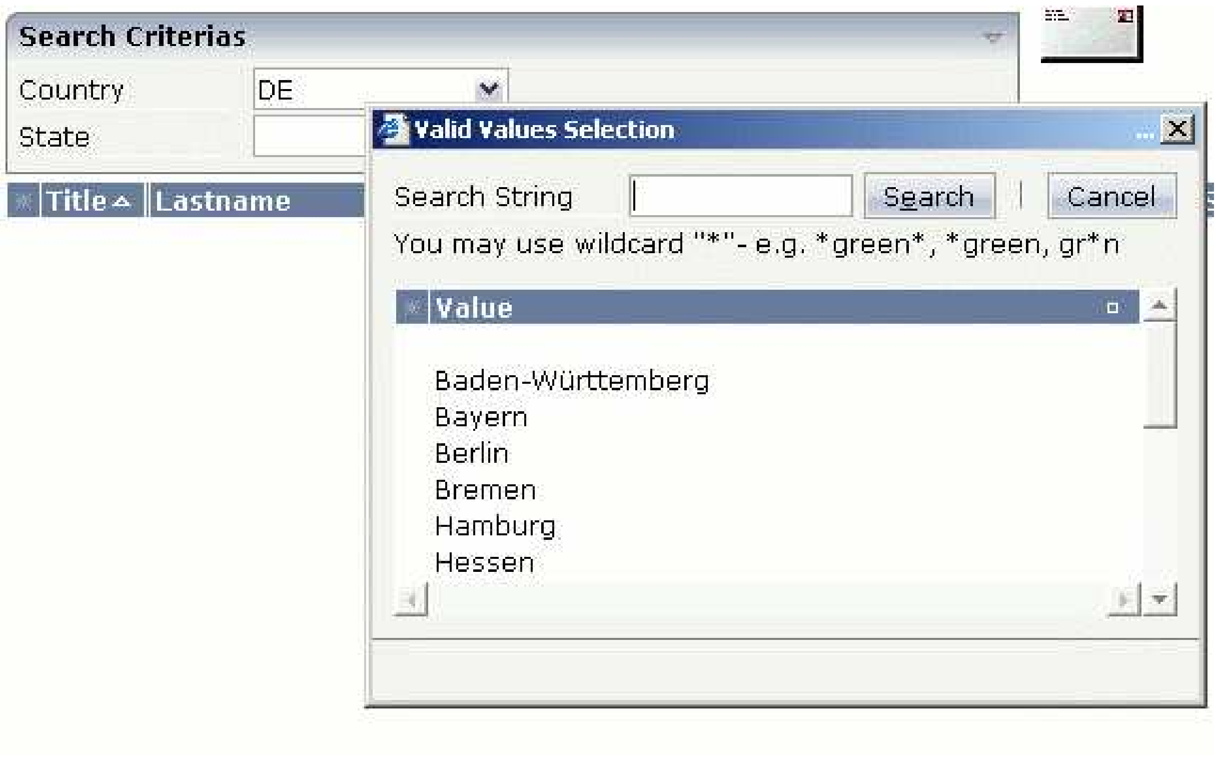


Country	US
State	New York

Title	Last Name	First Name	Cou	State
Mr	Schumacher	Michael	US	New York
Mr	Schumacher	Ralf	US	New York

The following screenshots demonstrate the dependency between country and state. The first pop-up shows the valid states if country is set to "US". The second pop-up shows the valid states in Germany.





Have a look at the XML layout definition:

```
<rowarea name="Search Criteria">
    <itr>
        <label name="Country" width="60">
        </label>
        <dbfield valueprop="dBFieldCountry" querycolumn="COUNTRY" datasource="addressdb"
                 valuehelptable="COUNTRY" valuehelpcolumn="ID">
        </dbfield>
    </itr>
    <itr>
        <label name="State" width="60">
        </label>
        <dbfield valueprop="dBFieldState" querycolumn="STATE" datasource="addressdb"
                 valuehelptable="STATE" valuehelpcolumn="ID" valuehelpcolumncond="COUNTRY">
        </dbfield>
        <hdist width="100%">
        </hdist>
        <button name="Execute" method="onExecute">
        </button>
    </itr>
</rowarea>
<rowarea name="Result" height="140">
    <itr height="100%">
        <textgridsss2 griddataprop="result" rowcount="5" width="100%">
            <column name="Title" property="TITLE" width="50">
            </column>
            <column name="Last Name" property="LASTNAME" width="100">
            </column>
            <column name="First Name" property="FIRSTNAME" width="100">
            </column>
            <column name="Country" property="COUNTRY" width="50">
            </column>
            <column name="State" property="STATE" width="100%">
            </column>
        </textgridsss2>
    </itr>
</rowarea>
```

The corresponding adapter code is:

```
// This class is a generated one.

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;

import com.softwareag.cis.server.Adapter;
import com.softwareag.cis.server.IDynamicAccess;
import com.softwareag.cis.server.util.DBFIELDInfo;
import com.softwareag.cis.server.util.DBQUERYDataObject;
import com.softwareag.cis.server.util.DBQUERYInfo;
import com.softwareag.cis.server.util.DBTEXTGRIDCollection;
import com.softwareag.cis.server.util.DBTEXTGRIDLine;
import com.softwareag.cis.server.util.DBUtil;
import com.softwareag.cis.server.util.DelegateError;
import com.softwareag.cis.server.util.IDBCondition;
import com.softwareag.cis.server.util.IDBQUERYConnectionProvider;
import com.softwareag.cis.server.util.IDBQUERYContextMenuRequestListener;
import com.softwareag.cis.server.util.IDBQUERYGeneratePDFRequestListener;
import com.softwareag.cis.server.util.IDBQUERYOptimizer;
import com.softwareag.cis.server.util.MENUNODEInfo;
import com.softwareag.cis.server.util.TREECollection;

public class DBFIELD_Adapter
    extends Adapter
    implements  IDBQUERYConnectionProvider, IDBDemoAdapter
{
    private Connection m_connection;

    // property >dBFieldCountry<
    DBFIELDInfo m_dBFieldCountry = new DBFIELDInfo(this);
    public DBFIELDInfo getDBFieldCountry() { return m_dBFieldCountry; }

    // property >dBFieldState<
    DBFIELDInfo m_dBFieldState = new DBFIELDInfo(this, m_dBFieldCountry);
    public DBFIELDInfo getDBFieldState() { return m_dBFieldState; }

    // property >result<
    DBTEXTGRIDCollection m_result = new DBTEXTGRIDCollection();
    public DBTEXTGRIDCollection getResult() { return m_result; }
    public void setResult(DBTEXTGRIDCollection value) { m_result = value; }
...
    // ------------------------------------------------------------------------
    // inner classes
    // ------------------------------------------------------------------------
    /** class used for a simple connection management. */
    public class ConnectionProvider
        implements IDBQUERYConnectionProvider
    {
        public Connection getDBConnection(String datasource)
        {
            try
            {
                Class.forName("org.hsqldb.jdbcDriver");
                return DriverManager.getConnection("jdbc:hsqldb:hsql://localhost", "sa", "");
            }
            catch (Exception exc)
            {
                throw new DelegateError(exc);
            }
        }
    }
...
    public void onExecute()
    {
        try
        {
            StringBuffer sb = new StringBuffer();
            sb.append("SELECT * FROM BUSINESSPARTNER INNER JOIN ADDRESS ON BUSINESSPARTNER.ID =
                                     ADDRESS.BUSINESSPARTNERID");
            DBUtil.addToQuery(sb, new IDBCondition[] { m_dBFieldCountry, m_dBFieldState}, true);
            String dataSource = m_dBFieldCountry.getDataSource();
            Connection con = getDBConnection(dataSource);
            ResultSet rs = con.createStatement().executeQuery(sb.toString());
            m_result.initWithResultSet(rs);
        }
        catch (Exception exc)
        {
            throw new DelegateError(exc);
        }
   }
...
}
```

Both properties `dBFieldCountry` and `dBFieldState` are of type `DBFIELDInfo` (from the package `com.softwareag.cis.server.util`). The `DBFIELDInfo` implements (like all DB controls) the interface `IDBCondition` (`com.softwareag.cis.server.util`). With class `DBUtil` (`com.softwareag.cis.server.util`), you can append the values of the filter criteria to the SELECT statement in a convenient way. See the JavaDoc documentation of class `DBUtil` for details.

The `DBFIELDInfo` class does not open a database connection on its own (same to all DB controls). The embedding adapter provides for an implementation of interface `IDBQUERYConnectionProvider` (`com.softwareag.cis.server.util`) when creating a `DBFIELDInfo` object. The interface method `getDBConnection` is called once - at the first time the `DBFIELDInfo` accesses the database. There are no updates (insert/update/delete) done with this connection. As the `DBFIELDInfo` does not open the connection, it does not care about closing the connection.

You see that the object `DBFIELDInfo` `dBFieldCountry` is passed in the constructor of `DBFIELDInfo` `dBFieldState`. With this, you define `DBFIELDINfo` `dBFieldState` depending on `dBFieldCountry`. The list of valid states only shows items that belongs to the country actually set.

For displaying the result, the class `DBTEXTGRIDCollection` (from the package `com.softwareag.cis.server.util`) is used. This class extends `TEXTGRIDCollection` by the ability to initialise the collection with a result set (`java.sql.ResultSet`). For each line of the result set, it creates an object of class `DBTEXTGRIDLine` (package `com.softwareag.cis.server.util`). Class `DBTEXTGRIDLine` implements the interface `IDynamicAccess`. With this, "normal" text grid controls can be used to visualize the data of the `DBTEXTGRIDCollection`.

Properties

Basic			
valueprop	Property that returns a DBFIELDInfo-instance. This instance provides for the value help read from database as well as for a convenient way to append the filter value to query string.	Obligatory	
querycolumn	Name of the column in the query to that the filter criteria is belongs to. This column may differ from the value help table/column (properties VALUEHELPTABLE, VALUEHELPCOLUMN). This name is used to build a SQL string in method "toSQLString".	Obligatory	
datasource	Logical identifier of the data source to use. This name is passed to the connection provider in method "IDBQUERYConnectionProvider.getDBConnection".	Obligatory	
valuehelptable	Name of the table from there the list of valid values can be read.	Obligatory	
valuehelpcolumn	Name of the column from there the list of valid values can be read.	Optional	
valuehelpcolumndescr	Name of a column where an additional description of the valid values is stored. The name must identify a column inside the "value help table" (property VALUEHELPTABLE).	Optional	
valuehelpcolumncond	Name of the column inside the "value help table" (property VALUEHELPTABLE) that defines the "to-one" dependency to another DBFIELD control.	Optional	
width	Width of DBFIELD in pixels or as percentage value.	Optional	
length	Width of DBFIELD in amount of characters. WIDTH and LENGTH should not be used together.	Optional	

datatype	By default, the DBFIELD is managing its content as a string. By the DATATYPE property, force the type of the data that is represented. As a consequence the DBFIELD is checking the data during input (e.g. if the DATATYPE is "int", it is not allowed to enter alphabetic characters) and adds a logic to transfer the data into various output formats (e.g. if the DATATYPE is "date", the date is formatted into the right date format).	Optional	int float date
flush	Flushing behaviour, please view "Common Rules" for details	Optional	screen server
displayonly	If set to "true", the DBFIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Explicit Alignment	Optional	
valign	Explicit Alignment	Optional	
colspan	Number of columns occupied by this control.	Optional	
rowspan	Number of rows occupied by this control.	Optional	
fieldstyle	Explicit style information passed to the DBFIELD. Example: if you want the text inside the DBFIELD to be right aligned, define "text-align: right".	Optional	
helpid	Identifier that is used for building the URL of the online help page. Please refer to "Online Help Management" for details.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	