COMPREHENSIVE SEARCH ON I-SERIES MACHINE THROUGH EGL

Background

Prior to EGL, IBM modernization tools were not efficient to support a wide level comprehensive search on a database.

EGL not only overcomes these issues but comes with a wide array of built-in functionalities bundled with modern technologies like Web 2.0 and SOA.

Using EGL, we gracefully achieved our objective in a short span of time and rendered outputs in a robust grid-based paginated table.

Challenges

To create a comprehensive search, we had to face the below obstacles:

- Web 2.0 component based web-page
- Performance
- Lose Coupling & Code Abstraction
- SOA based solution

Solution

EGL is based on 3-tier architecture; we designed our application according to the same:

- UI layer
- Business logic layer
- Data access layer

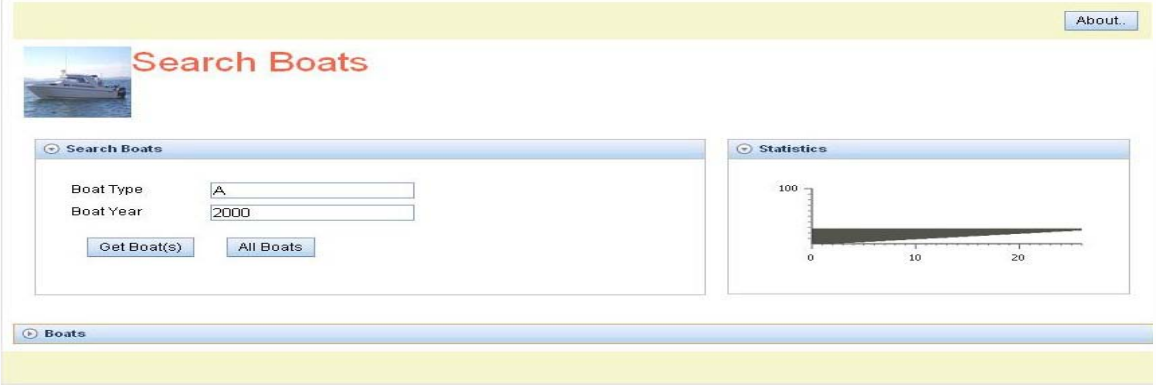
UI layer

Rich User Interface (**RUI**) provides a visual interface to design web pages visually through custom RUI widgets. We designed the interface using some of these widgets. Plus, we used a well-known JavaScript framework named DOJO to enrich interface.

Various search criteria based fields like boat type, feet & year fields are provided to retrieve relevant data.

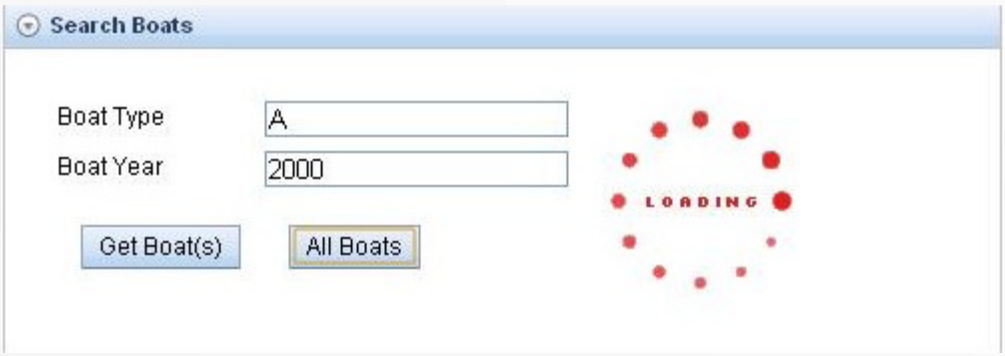
The results are displayed in a pagination enabled grid inside flexible panel layouts.

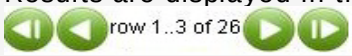
A snapshot is given below



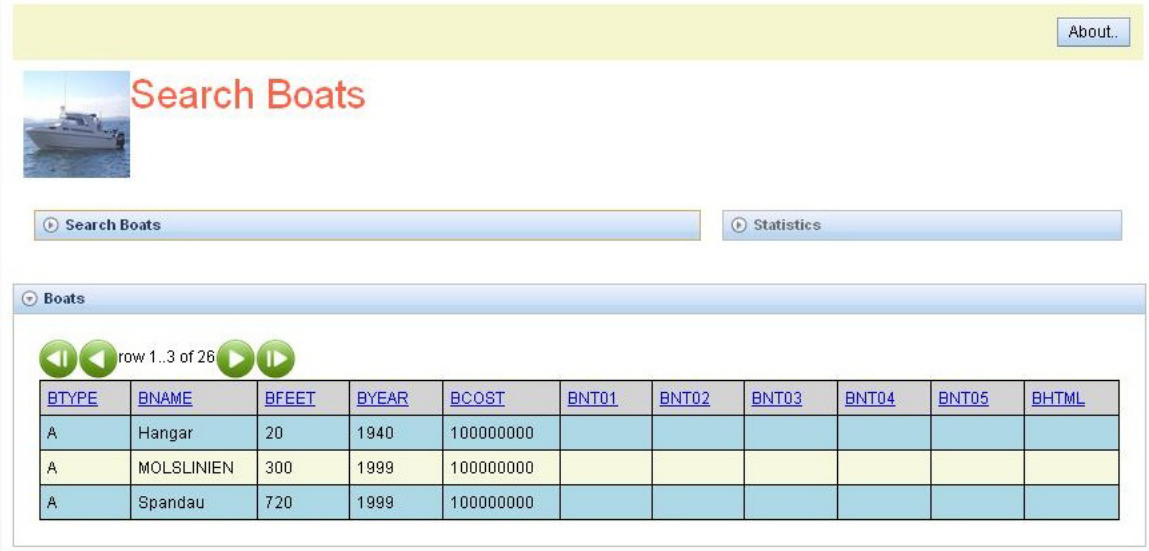
The pictured panel based layouts enable users to easily open or close them on will. One panel provides search functionality, where other is showing the result's statistics.

Once required values are filled in the given fields a progress bar will appear to show the search status something like 'the results are being fetched, please wait'.



Results are displayed in the below pagination enabled grid, where these icons 

help the user to move forward or backward through the result pages.



BTYPE	BNAME	BFEET	BYEAR	BCOST	BNT01	BNT02	BNT03	BNT04	BNT05	BHTML
A	Hangar	20	1940	100000000						
A	MOLSLINIEN	300	1999	100000000						
A	Spandau	720	1999	100000000						

Business Logic Layer

This layer hold logic to draw/display RUI widgets, settings widget-properties, assigning values to widgets, controlling & managing user actions (events) to page components & so on.

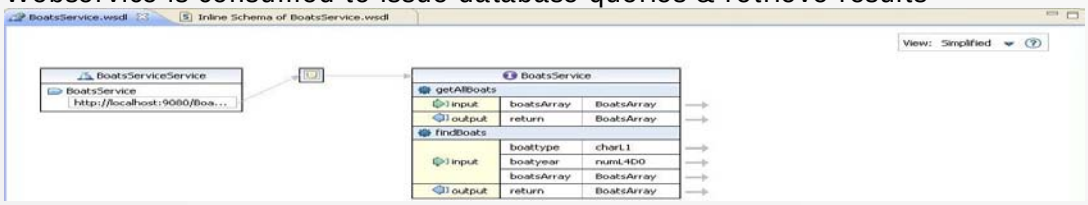
Code creating grid is partially given in the below snapshot.

```
columns = [
        new com.ibm.egl.rui.widgets.GridColumn{name = "BTYPE", displayName = "BTYPE", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BNAME", displayName = "BNAME", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BFEET", displayName = "BFEET", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BYEAR", displayName = "BYEAR", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BCOST", displayName = "BCOST", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BNT01", displayName = "BNT01", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BNT02", displayName = "BNT02", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BNT03", displayName = "BNT03", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BNT04", displayName = "BNT04", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BNT05", displayName = "BNT05", width=120},
        new com.ibm.egl.rui.widgets.GridColumn{name = "BHTML", displayName = "BHTML", width=120}],
    visibleRows = 3,  //How many rows in view at a time
    data = []
    };
```

RUI handler code is pictured in the below snapshot

```
handler BoatsDetails type RUIhandler{initialUI =[  mainBox
            ], onConstructionFunction = initialization, cssFile = "css/BoatsRUI.css"}
    mainBox com.ibm.egl.rui.widgets.Box{padding = 20, children =[ LeftPadBox, CenterContentBox,RightPadBox
            ], columns = 1  };
```

Webservice is consumed to issue database queries & retrieve results



It's very easy to call a webservice in EGL, like we did in the below code

```
call BService.findBoats(btype, byear, allBoats)
    returning to findBoatsResult
    onException showExp;
```

Once the values are retrieved, these will be assigned directly to grid without any type of complexity.

Data access layer

This essential layer holds business logic, it set up database connections, querying database, retrieving results, assigning results to display widgets and so on.

EGL provides very easy way to retrieve database records, snapshot shows below

```
service BoatsService

    // Variable Declarations
    //inputboats Boats;

     function getAllBoats( boatsArray Boats[] in ) returns (Boats[])
            try
                get boatsArray with
                    #sql{

            onException (e  SQLException)
                SysLib.writeStdout( "Error inside Service" + e.message );
            end

            if (size(boatsArray)>0 )
                return (boatsArray);
            else
                return(null);
            end
        end
```

Conclusion

EGL enabled us to create a robust & comprehensive search application for an iSeries machine. It demanded less efforts & time to meet challenges swiftly and helped us to stand in the diversified marketplace.

© Copyright IBM Corporation 2010
IBM Global Services
Route 100
Somers, NY 10589
U.S.A.
Produced in the United States of America
08-10
All Rights Reserved

IBM, the IBM logo, ibm.com, Lotus®, Rational®, Tivoli®, DB2® and WebSphere® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at ibm.com/legal/copytrade.shtml Other company, product and service names may be trademarks or service marks of others. The information contained in this documentation is provided for informational purposes only. While efforts were made to verify the completeness and accuracy of the information contained in this documentation, it is provided “as is” without warranty of any kind, express or implied. In addition, this information is based on IBM’s current product plans and strategy, which are subject to change by IBM without notice. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this documentation or any other documentation. Nothing contained in this documentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM (or its suppliers or licensors), or altering the terms and conditions of the applicable license agreement governing the use of IBM software. This document illustrates how one organization uses IBM products. Many factors have contributed to the results and benefits described; IBM does not guarantee comparable results elsewhere.