Graduate Research in Engineering and Technology (GRET)

Volume 1 | Issue 2 Article 16

October 2013

USER INTERFACE DESIGN FOR MANAGEMENT OF THE NETWORK ELEMENTS USING DOJO

SUDHARANI T. K
Dept of CSE, SIT, Tumkur, tksudharani88@gmail.com

SHANTHAKUMAR A. H
Dept of CSE, SIT, Tumkur, shanthakumara@sit.ac.in

USER INTERFACE DESIGN FOR MANAGEMENT OF THE NETWORK ELEMENTS USING DOJO

SUDHARANI T.K[1], SHANTHAKUMAR A.H[2]

[1]4[th]sem, M.Tech (CSE), [2]Asst professor, Dept of CSE, SIT, Tumkur
E-mail: tksudharani88@gmail.com, shanthakumara@sit.ac.in

Abstract- This paper focuses on the design of the user interface for managing network elements using dojo. Dojo is an open source java script toolkit which is used for constructing dynamic web user interfaces. Here we are designing the user interface for the management of the network elements in a optical management system .User can see the tree list of all the network elements and network element hierarchy and will be able to access on any network element and view the details and status of it and perform provisioning operations on it. It also provides various user forms and tabular representation of the network elements. Dojo is more efficient than the JavaScript and HTML in terms of security, cross platform support and providing dynamism. Dojo saves the time and scales with the development process, using web standards as its platform.

Keywords*: Equipment manager (EQM), Network element (NE),,Dojo, optical management System(OMS).*

I. INTRODUCTION

With rich Internet applications (RIA) rise ceaselessly, the function of all kinds of JavaScript development kit is enhancing. [5] Dojo is one of them. Dojo Toolkit is an open source modular JavaScript library (or more specifically JavaScript toolkit) designed to ease the rapid development of cross-platform, JavaScript or Ajax-based applications and websites. [5]

Dojo is a JavaScript framework which targets on the many needs of large-scale client-side web development. It provides modularity in code through dojo classes. Dojo gives desktop as well as mobile support and also provides rich UI by using Dijit widget. Dojo gives uniform access to browser APIs.

Dojo widgets are components comprising JavaScript Code, HTML markup, and CSS style declaration which
Provide cross-browser, interactive features: [5]

- Menus, tabs and tooltips
- Sortable tables, dynamic charts, and 2D vector Drawings
- Animated effects fade wipes and slides and provides custom animation effects
- Tree widgets that support drag and drop operations.
- Various forms for validating form input
- Calendar-based time selector, date selector and Clock.
- Core widgets
- Also for 3D.

II. BENEFITS OF DOJO

Dojo is more efficient than the HTML and JavaScript. HTML can create only static and plain pages so if we need dynamic pages then html is not useful. We need to write lot of code for making simple web page. Security features are not good in html. The disadvantage of java script is that not all browsers can recognize JavaScript. Using JavaScript limits the accessibility of a Website since some mobile devices, old browsers or screen readers do not recognize JavaScript. Some users turn off the JavaScript support in their browsers for security reasons. Dojo is supported in all major web browsers: Opera 9+, FF2+, Safari 3+, and IE6+.

JSP pages require double the disk space to hold the JSP page.
It requires more time when accessed for the first time as they are to be compiled on the server. Dojo's package system makes it easy to manage large-scale UI development projects and the build system layers on top to make the applications scream; all without code changes. Dojo also contains high-performance implementations of common utilities into its core, and the rebuild of Dojo for focuses heavily on performance and reduced code footprint It saves the time and scales with the development process, using web standards as its platform. It is the toolkit for experienced developers turn to for building high quality desktop and mobile web applications.

Dojo also provides a set of component library that can be used directly in any applications. We can use some of the core components as UI components, such as menus, tabs, tree component and so on. It provides abstraction layer to the programmer to develop powerful features very easily. Dojo base size is very small (26 KB) which take less bandwidth and less initial load time (based on required feature).

DojoX.Secure is a collection of tools for security, when working with untrusted data and code. Dojox.secure.dom provides a DOM facade that restricts access to a specified sub tree of the DOM. Dojox.secure.capability is a object-capability JavaScript validation. This is a validator to run before

evaluation to ensure that a script can't access or modify any objects outside of those specifically provided to it.

Dojo's class widget library is "Dijit". This framework enables rapid development of rich internet applications with a quality look and feel on modern browsers.

This paper focus on the use of the dojo for designing the user interface of the equipment manager which is a component in the optical management System (OMS).The Optical Management System (OMS) software centralizes multiple network management functions with one system. OMS provides the functionality to manage a wide variety of the network elements. A network element is usually defined as a manageable logical entity uniting one or more physical devices. It allows distributed devices to be managed using one management system in a unified way. NE will contain the shelves, shelves will contain the cards, and cards contain the ports. Ports are the physical interfaces of the pieces of the equipment that are located on cards. Ports can also be connectors. Equipment manager is the management of the network element and network element hierarchy.

OMS is a network management system that supports several management layers that can accommodate and grow with the customer's optical network. The Element management layer, or EML, provides the functionality that is needed to access any network elements (NEs) that are deployed in a customer network. The EML provides a single access point for communication with an NE. The network management layer, or NML, provides the functionality that is needed to commission, provision, and supervise the network that is deployed in a customer premise.[1] The Service management layer provides the functionality that is needed to commission, provision, and supervise a Virtual Private Network (VPN) that a customer deploys to its end users or to its customers. [1]

The Equipment Manager (EQM) is responsible for management of the equipment holder and equipment information.(e.g. shelves, slots, circuit packs, etc.) of a Network Element. [1] Shelves are telecom objects that can be made up of a certain number of slots of different widths in which cards can be stored. Cards are represented by rectangles that support the same base states as network elements. They can carry alarm and status icons like network elements. The equipment model represents the various manageable physical components of the network element (circuit Packs etc...). The Equipment holder represents an abstraction of Rack, Shelf, Slot and Sub slot.

The rest of the paper is organized as follows. Section III deals with the existing system. Section IV deals with proposed system (EQM user interface) and section V deals with comparison and results and section VI deals with the conclusion.

III. EXISTING SYSTEM.

ZIC for NE management. By ZIC we can open only 1 NE instance at a time. EQM can be opened if NE is not reachable also. Every NE has a ZIC. ZIC can be opened only if NE can be communicated. All the provisioning and we can see all the details of NE in one application in EQM. The ZIC Manager is the proxy for the connections towards managed NEs supporting GUI ZIC. This module supports ZIC NE GUI running into EML environment supplying services for user identification, NE connection, file system access and navigation between EML applications. ZIC was developed using JavaScript, HTML and applet.

IV.PROPOSED SYSTEM

The figure shows the EQM front end developed by using dojo. There are four parts in the graphical user interface which are tree, summary, graphical, grid.

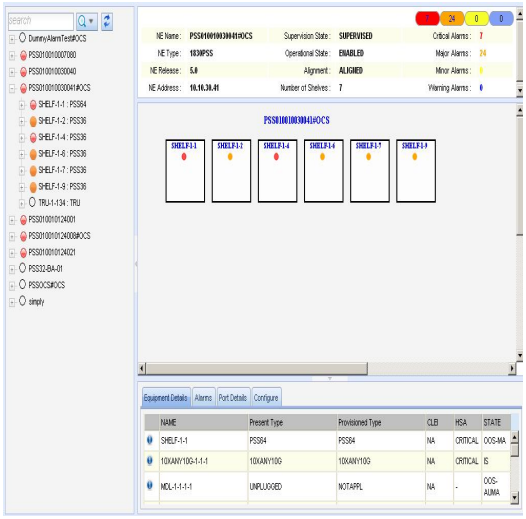

Fig 1: EQM user interface

Grid:

The Data Grid is the central component of many applications due to its effective and usable presentation of tabular data. Dojox.grid.DataGrid provides various forms and routines for validating form input widget and is also used for the construction of the grid. With minimal memory usage the Data Grid is able to consume and scroll through thousands of rows of data. The Data Grid also provides a way to control visual styling via CSS styles or CSS classes. The cell definition properties header Styles, cell Styles, and styles are strings of CSS style definitions (that must be terminated with a semi-colon) that are applied to only the header cell, only the content cells, and to all cells respectively.

The dojox.grid.DataGrid is a powerful component. By using it we can render large data sets and we can create complex layouts of rows and sub-rows for our data. Without having to click controls it allows users to intuitively scroll through long lists of information. When a large set of data needed to be displayed, we would use a method called paging: only a small subset (perhaps 25-50 records) of the entire set would be displayed. Using buttons or other control the user would navigate through "pages" of data.

The DataGrid takes a slightly different approach called virtual scrolling: a user only needs to scroll to navigate. The reason it is called virtual scrolling is that only a small subset of the data is ever rendered at one time although it may seem as though it is a long list of records. Grid provides all the NE details like alarms, ports, etc.

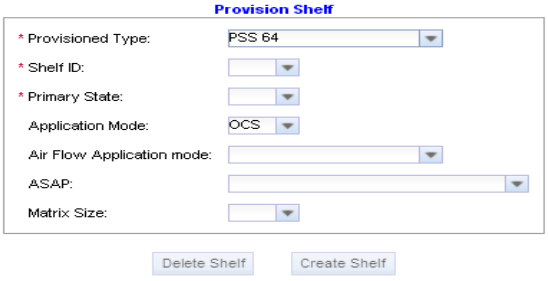

Fig 2: User form

Grid displays Equipment details, port details, alarm and configure tab to create and manage NE and NE related aspects by providing various user forms.

Layout management is the process of actively controlling layout after a page has loaded, and responding to and propagating events, which then drive layout on the page. In Dijit, layout management is accomplished by specialized layout widgets. These widgets act as a container for one or more content areas or child widgets, and to control the sizing and display of those children to implement the layout, we will be using three widget classes from Dijit: dijit.layout.BorderContainer, dijit.layout.TabContainer and dijit.layout.ContentPane.

	NAME	Present Type	Provisioned Type	CLEI	HSA	STATE
	SHELF-1-1	PSS64	PSS64	NA	CRITICAL	OOS-MA
	10XANY10G-1-1-1	10XANY10G	10XANY10G	NA	CRITICAL	IS
	MDL-1-1-1-1	UNPLUGGED	NOTAPPL	NA	-	OOS-AUMA
	XFP-1-1-1-2	SR111G1AU	SR111G1AU	NA	CRITICAL	OOS-MA
	XFP-1-1-1-3	SR111G1AU	SR111G1AU	NA	CRITICAL	OOS-MA
	XFP-1-1-1-4	SR111G1AU	SR111G1AU	NA	CRITICAL	OOS-MA
	XFP-1-1-1-5	SR111G1AU	SR111G1AU	NA	-	IS
	XFP-1-1-1-6	SR111G1AU	SR111G1AU	NA	-	IS
	XFP-1-1-1-7	SR111G1AU	SR111G1AU	NA	-	IS
	XFP-1-1-1-8	SR111G1AU	SR111G1AU	NA	-	IS
	XFP-1-1-1-9	SR111G1AU	SR111G1AU	NA	-	IS
	XFP-1-1-1-10	SR111G1AU	SR111G1AU	NA	-	IS

Fig 3: Tabular presentation

The above figure shows the tabular representation of the details of the shelf and cards that are present on a particular network element. Shelf or card name, shelf type, primary state, highest severity alarm (HSA) present on it.

Tree:

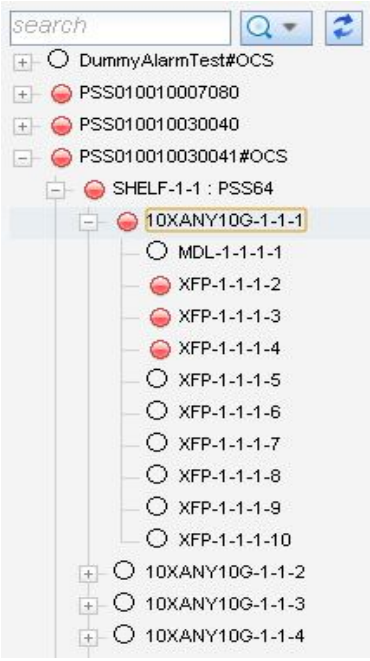

Fig 4: Dojo Tree

Tree is developed by using dojo's Dijit.Tree widget. The tree in User Interfaces help sort out long, hierarchical lists. Helps in connecting the tree to any dojo.store, like xml with or without a single root item, and with various ways to express parent/child relationships. Nest items to an arbitrary depth. Each branch is independently expandable. Setup a global handler for when a user clicks particular nodes. Tree will automatically reflect changes made to the underlying data store (when connected to the data store through the ObjestStoremodel or TreeStoremodel or ForestStoreModel).[2]

We can allow nodes to be dragged and dropped through the Dojo DnD API. .Drag and drop onto the tree, which updates the data store indirectly. The tree displays the list of all the NE's, which are arranged in the alphabetical order, so that it is easy for the user to search for a particular NE. Each NE, shelf or sub slot is equipped with the alarm, which indicates a fault or failure in the network. Alarms consist in messages emitted by network resources that have detected problems or failures in the network. Malfunctions are ranked into five severity levels:

–Critical indicates when it is no longer possible to provide the service requested.

– Major.

– Minor.

– Warning which means no impact is reported upon the quality of service offered.

– Indeterminate if the severity level cannot be defined by the resource which raises the alarm. [1]
There is option to refresh the tree. When clicked, we can open the details of the node. Expanding the tree node at multiple levels by recursively getting the children of the nodes. Focusing to the first node after freshly searching. In the search we have provided a drop down box to search for the NE, equipment or all. We have the refresh button to refresh the tree contents. The hierarchy is that Network element will contain shelves. Shelves will contain slots. Slots will contain sub slots which in turn will contain cards. Card can be a switch or hardware device which is used for communication purpose.

Tree has the following three components:
1) Tree
The Tree widget itself is merely a view of the data. It's in charge of displaying the data and handling user events only. The Tree is a black-box in the sense that the developer generally won't be dealing with individual nodes of the Tree. Rather, there are just onClick () notifications, which refer to the item that was clicked. Item is usually an item in the dojo.data store that the tree is connected to. Tree has an idea of a currently selected item, such as the currently opened folder in a mail program.[4]

2) Model
The real power comes in the tree model, which represents the hierarchical data that the tree will display. Tree can interface to any class implementing the model API, but typically either the TreeStoreModel or ForestStoreModel are used, both of which themselves interface with the powerful dojo.data API[4]

3) Data Stores
Usually the model interfaces with a dojo.data store. There can be many different types of stores, such as stores that execute on the client vs. stores that pass through to the server, stores that work from XML vs. stores that work from JSON, stores that load data as it's needed or stores that load all the data on initialization, etc. All the stores, though, have the same API, so they can be connected to with either TreeStoreModel or ForestStoreModel, depending on whether there is a single or multiple top level item in the store. [4]

4) Relationship
From the simplest point of view, the information flows like this:
Data Store –> Model –> Tree.
Dijit's modal Dialog Box simulates a regular GUI dialog box. The contents can be arbitrary HTML, but are most often a form or a short paragraph. The user can close the dialog box without acting by clicking on the X button in the top-right corner. Dialog sizes itself to be just big enough to show its contents. If the contents are too large to fit in the viewport, Dialog uses a scrollbar to scroll its contents.
Using Dojo to create tree menu is very simple, even if a complex tree, it also can create completely:
• Link Tree to any dojo.data store, with or without root item.
• Nested items to arbitrary depth, each branch can be independently expandable.
• Using different icons for different branches or leaves.
• Installing a global handler for response to users' click or double-click on a specific node.
• Tree can automatically reflect its change to data store. (when using TreeStoreModel or ForestStoreModel to connect to data store).
• Allow node to be dragged through the Dojo DnD API.
• Dragging on the tree, will update data store directly.

Summary:

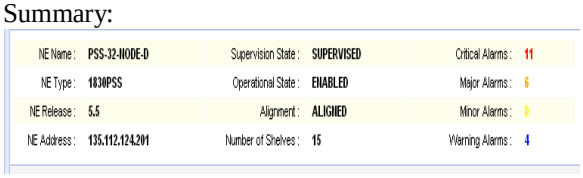

Fig 5: Summary window

Summary will display the details of the NE like NE supervision state, Operational state, NE name, NE address, NE type, NE release. It will display the alarms of the NE, alignment, number of shelves on the NE, critical alarms, major alarms, minor alarms, warning alarms.NE will be equipped with the alarms like critical, major, minor and warning.

In the summary dojox.data.xmlstore is used. XmlStore is a store provided by Dojo that is contained in the DojoX project. XmlStore provides read and write interface to basic XML data. XML is a common data interchange format and a store that can work with fairly generic XML documents is useful. The store is designed so that when performing read and saves we can over-ride certain functions to get specific behaviors to occur. Javax.xml.bind.annotation defines annotations for customizing Java program elements to XML Schema mapping. HttpServletRequest is the interface to provide request information for HTTP servlets.

The dojo.data APIs that are implemented by the xmlstore are dojo.data.api.Read, dojo.data.api.Write and dojo.data.api.Identity. We can connect the xmlstore to dijit.form.combobox and dojox.grid.datagrid.The store is designed so that it can read generic XML and present back nodes as dojo.data items.
Graphical:
Graphical window displays the graphical view of the

NE, shelf, slot or sub slot when clicked on the name in the tree. Here we use the xml annotations to set the number of the shelves, slot width, slot type, image name, slot height, width.

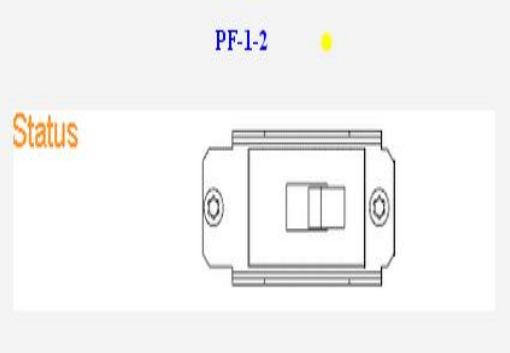

Fig 6: Graphical view of a card

Dojox.gfx is a cross platform vector graphics API. GFX helps to isolate application from the many native vector graphics implementation differences across all modern Browsers by detecting the best graphics engine implementation for the Browser that the application is running on, and providing API's that are same across the various implementations

Development environment:

Configuration Management – Clear Case
Build Tool Ant
IDE Eclipse
Java Version 1.5
Data Base ORACLE
ORM Hibernate
GUI MS-GUI Framework
Unit Testing JUnit

V. COMPARISION AND RESULTS

The figure shows the performance of the dojo vs. jQuery. jQuery is a lightweight JavaScript library. The purpose of jQuery is to make it much easier to use JavaScript on the website.

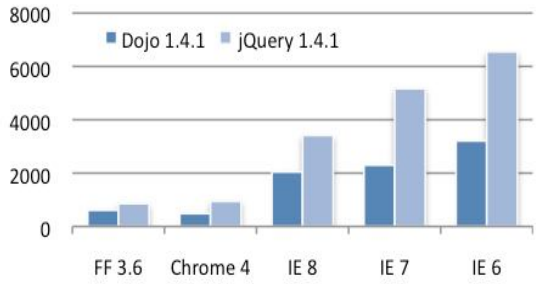

Fig 7: performance of dojo vs. jQuery.

Here smaller is better. Dojo provides the fastest possible way to do each of the tasks in the benchmark. That is it provides the fastest implementation. Here the average of the three separate runs of the taskspeed benchmark is considered in comparing the latest versions of both Dojo and jQuery. Taskspeed is explicitly designed to capture common-case web development tasks

VI. CONCLUSION

This paper has shown how to use dojo for constructing dynamic web user interfaces. Dojo gives desktop as well as mobile support and also provides rich user interface by using Dijit widget. The Dojo Mobile package provides a number of widgets that can be used to build web-based applications for mobile devices such as iPhone, Android, or BlackBerry.

REFERENCES

[1] User_guide_18_mar_2010_eml9.1.1_user_Guide

[2] Dojo: The Definitive Guide.O'REILLY, Matthew A. Russell.

[3] Bill Keese, Nikolai Onken, Marcus Reimann. dijit.Tree.

http://docs.dojocampus.org/dijit/Tree , 2010-10

[4] The dynamic Retrieval tree menu based on dojo, XiaoXiao Zhang, Yan Cao, Xiangwei Mu, 2011 International Conference of Information Technology, Computer Engineering and Management Sciences.

[5] Data validation based on dojo. 2011 International Conference of Information Technology, Computer Engineering and Management Sciences. Bingyan Chen, Yan Cao, Xiangwei Mu.

◈◈◈