Döcu Content

Navigation

Access to Ids, Controls, Pages, Charts...

Introduction:
Döcu Content is a content management system. It is a compilation of charts, reports (JasperReports, JFreeChart, and Google Chart). It comes complete with Admin, Log, Maintenance, and Bulletin pages to create a user experience based on security settings set forth by JavaAgent, JavaBean, and XML code. Use this navigation tutorial document to leverage Döcu Content benefits.... do stay tuned for future versions of the App.

Disclaimer:
Information contained in the following is presented as is. This tutorial assumes you have basic Lotus Notes Configuration knowledge.

Load Lotus Notes Domino Administrator
First order of business is to load Lotus Notes Domino Administrator, you can do this right from your Domino Designer Environement by going to File + Launch Domino Administrator. Once there, you will need to go the Files Tab to view your list of databases. Provided you have already copied the Döcu Content App into your ***root*** folder, (1) select that dev folder, (2) right-click on the App (3) and choose Sign.

Döcu Content Login:
Use the following userID to login to Döcu Content: **AlienHouse** If you need to use your own account userID, please follow instructions in the FileBin.nsf documention. For now here are some scrteenshot on what to expect as you navigate the App; areas of interest are added for your convenience.

Figure 1
If your userID does not work, please note you will remain on the login page. You can change this by adding an error page, or making available a page validator that loads an error against incorrect Ids right on the Login Form.

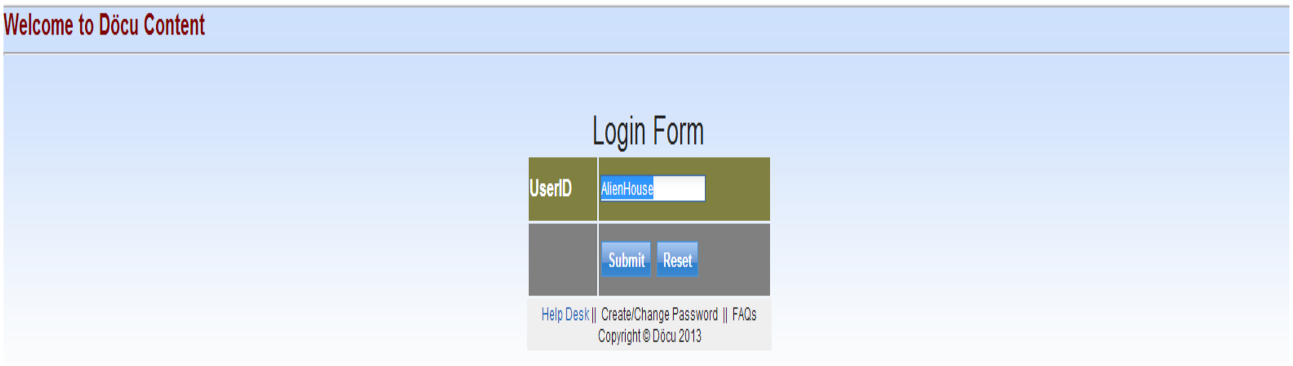
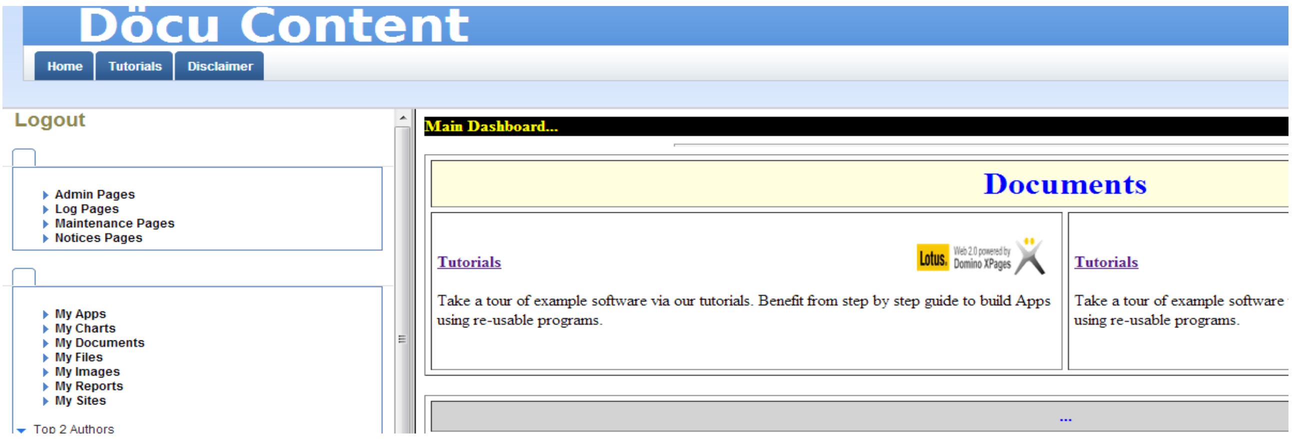


Figure 2
Main page loads when Login is successful, navigate through pages via collapsible items @ left. Please note, if your userID is not in the specific view and you have not yet modified the specific Java class files, some sections will not be visible. You can modify code at your leizure to enhance this behaviour and remove the need to modify files



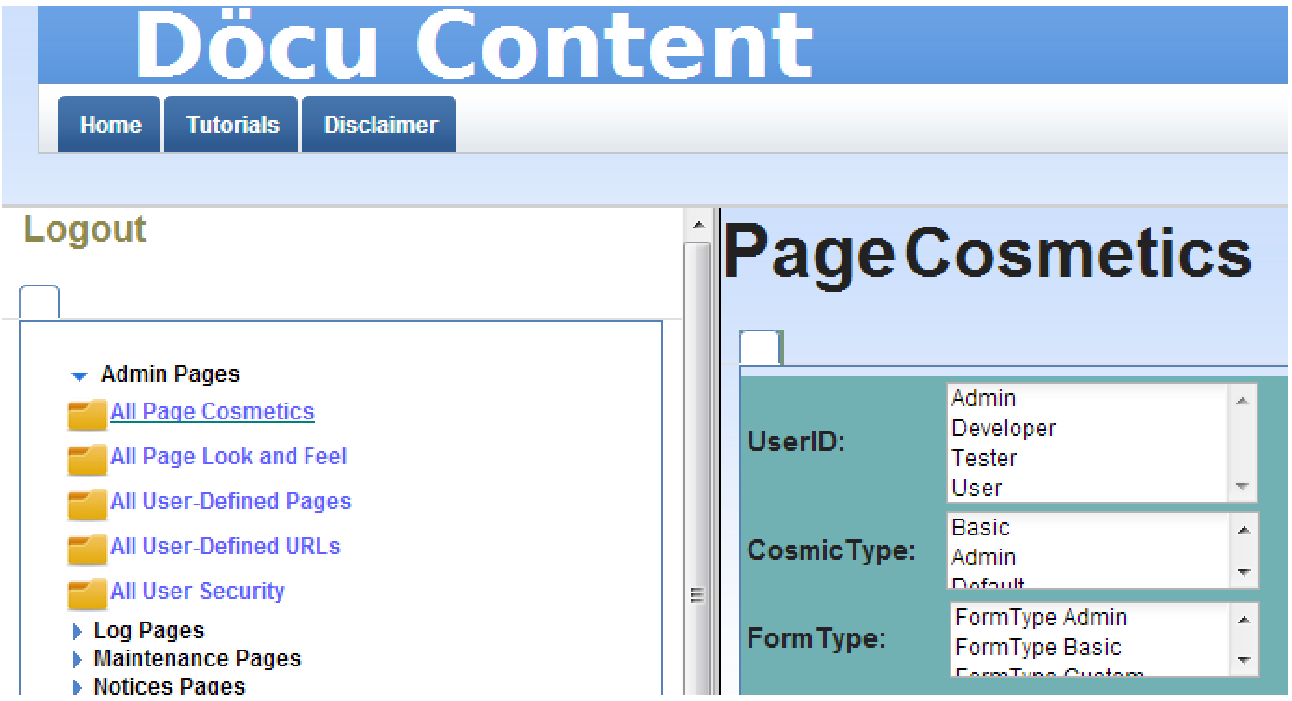
Take a tour of example software via our tutorials. Benefit from step by step guide to build Apps using re-usable programs.

Take a tour of example software using re-usable programs.

TIP:

Only the folder icons are based on security settings in code, links will need modification to hide/show based on view information in the NSF back-end. See below figures for hints on how to change this, but first here is a look of the folders/links expanded.

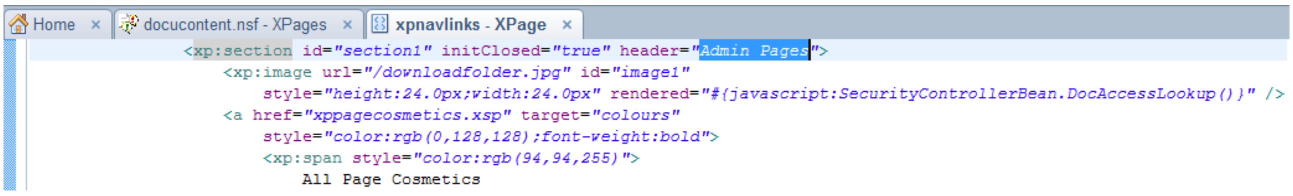
Figure 3
Page Cosmetics by the way is used by Administrators to add access for specific users, this section is based on their access to see the the current page, see associating screenshots for more



UserID:	Admin Developer Tester User
CosmicType:	Basic Admin
FormType:	FormType Admin FormType Basic

Figure 4

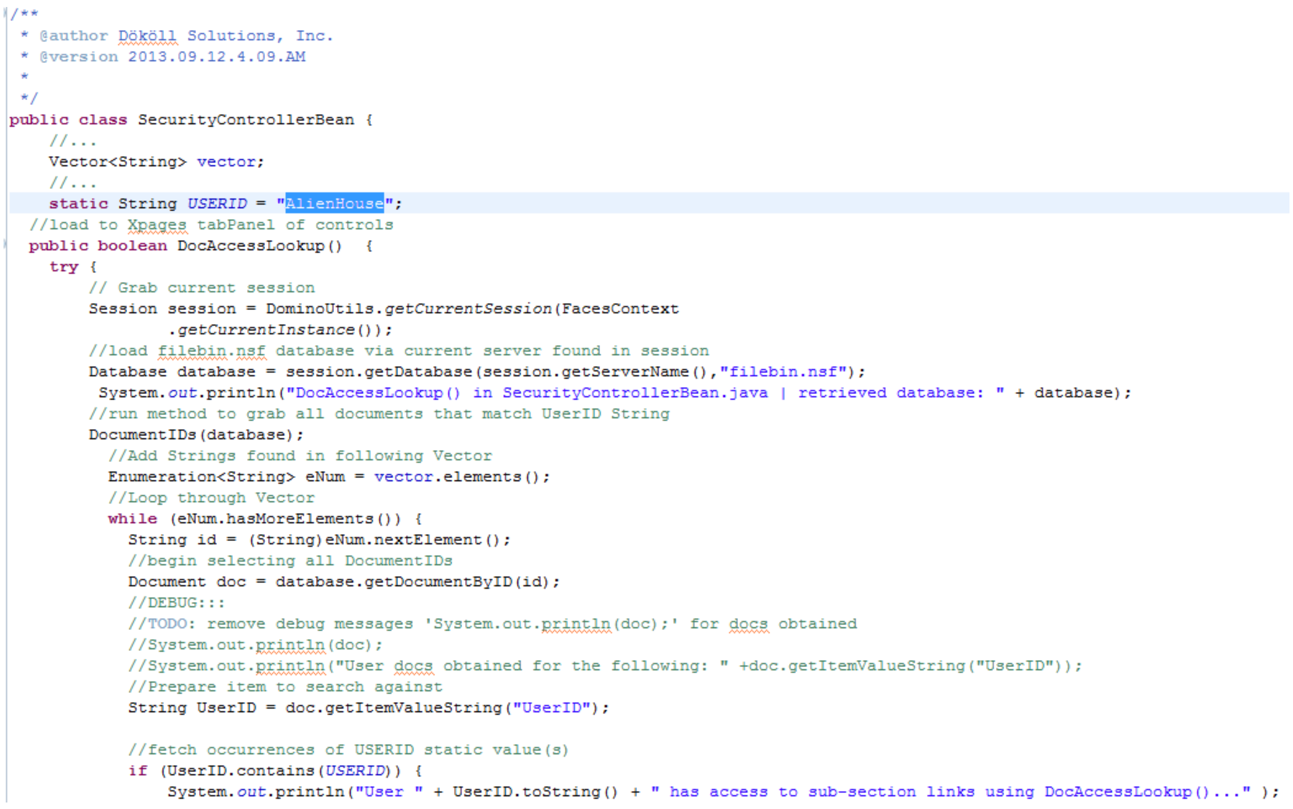
The Ad min section is made available by rendering the link based on the SecurityControllerBean, see **package** `com.dokoll.solutions.inc.developement.Utils;`



```
<xp:section id="section1" initClosed="true" header="Admin Pages">
    <xp:image url="/downloadfolder.jpg" id="image1"
        style="height:24.0px;width:24.0px" rendered="#{javascript:SecurityControllerBean.DocAccessLookup()}" />
    <a href="xppagecosmetics.xsp" target="colours"
        style="color:rgb(0,128,128);font-weight:bold">
        <xp:span style="color:rgb(94,94,255)">
            All Page Cosmetics
```

Figure 5

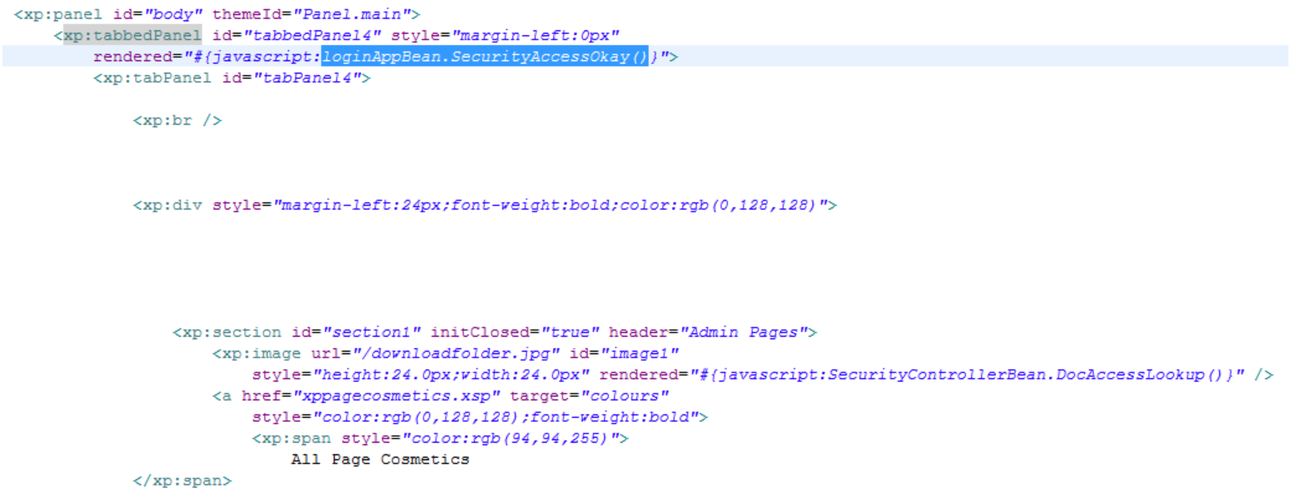
User 'AlienHouse' has access to section and folder icon in question, you must use SecurityControllerBean class to add your userID if you are not using AlienHouse as Login userID. You can modify this code, and add a form control to hold static values to compare against the returning userID from the filebin.nsf view 'SecurityAccessView' housing all userIDs. You can also query this view add plug results into a comparison control variable to match against the returning value added by user in the userID field on xploginform.xsp form



```
/**
 * @author Dököll Solutions, Inc.
 * @version 2013.09.12.4.09.AM
 *
 */
public class SecurityControllerBean {
    //...
    Vector<String> vector;
    //...
    static String USERID = "AlienHouse";
  //load to Xpages tabPanel of controls
  public boolean DocAccessLookup()  {
    try {
        // Grab current session
        Session session = DominoUtils.getCurrentSession(FacesContext
                .getCurrentInstance());
        //load filebin.nsf database via current server found in session
        Database database = session.getDatabase(session.getServerName(),"filebin.nsf");
         System.out.println("DocAccessLookup() in SecurityControllerBean.java | retrieved database: " + database);
        //run method to grab all documents that match UserId String
        DocumentIDs(database);
          //Add Strings found in following Vector
          Enumeration<String> eNum = vector.elements();
          //Loop through Vector
          while (eNum.hasMoreElements()) {
            String id = (String)eNum.nextElement();
            //begin selecting all DocumentIDs
            Document doc = database.getDocumentByID(id);
            //DEBUG:::
            //TODO: remove debug messages 'System.out.println(doc);' for docs obtained
            //System.out.println(doc);
            //System.out.println("User docs obtained for the following: " +doc.getItemValueString("UserID"));
            //Prepare item to search against
            String UserID = doc.getItemValueString("UserID");

            //fetch occurrences of USERID static value(s)
            if (UserID.contains(USERID)) {
                System.out.println("User " + UserID.toString() + " has access to sub-section links using DocAccessLookup()..." );
```

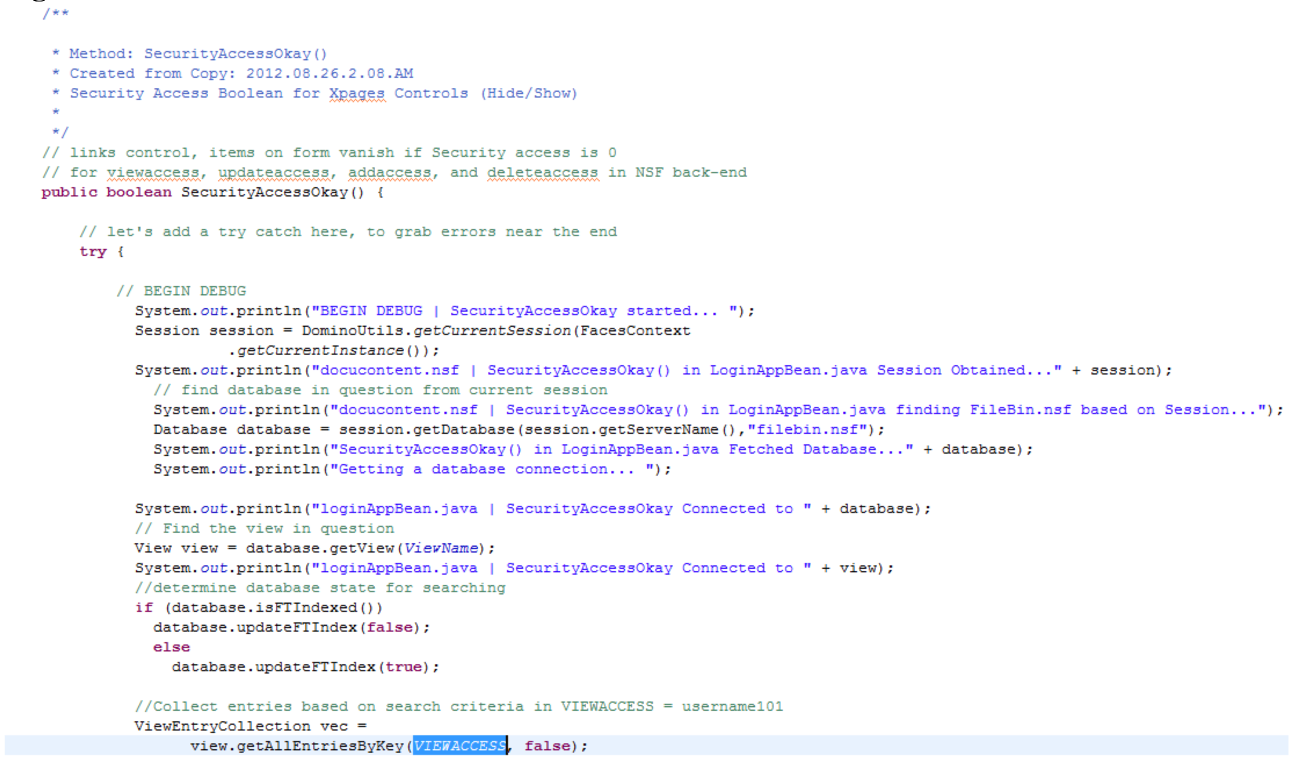
Figure 6

The section itself is controlled by the LoginAppBean class and the code resides in the DöcuContent.nsf App, having a method that reads a view in the filebin.nsf App containing user Ids that have access to that section. Again, code can be modified to allow seamless interaction against static variables within the code.



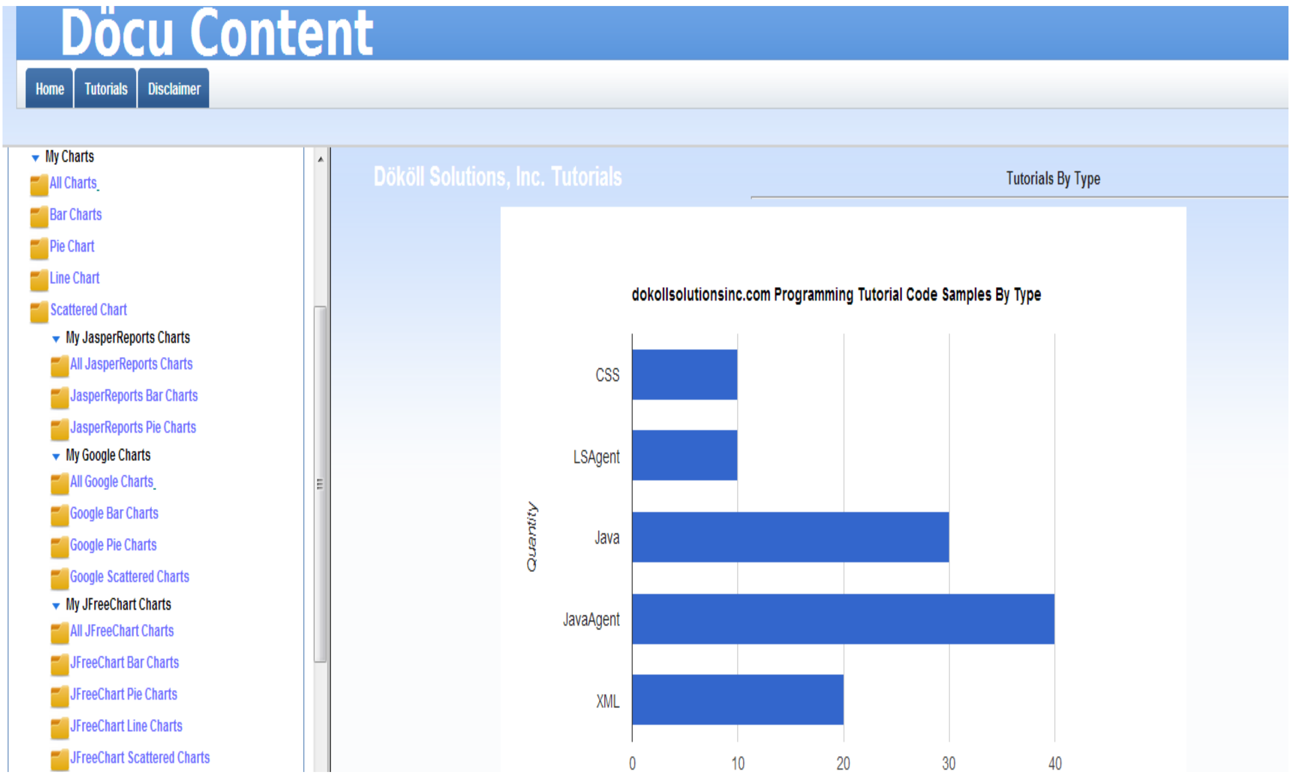
```
<xp:panel id="body" themeId="Panel.main">
    <xp:tabbedPanel id="tabbedPanel4" style="margin-left:0px"
        rendered="#{javascript:loginAppBean.SecurityAccessOkay()}">
        <xp:tabPanel id="tabPanel4">

            <xp:br />


            <xp:div style="margin-left:24px;font-weight:bold;color:rgb(0,128,128)">




                <xp:section id="section1" initClosed="true" header="Admin Pages">
                    <xp:image url="/downloadfolder.jpg" id="image1"
                        style="height:24.0px;width:24.0px" rendered="#{javascript:SecurityControllerBean.DocAccessLookup()}" />
                    <a href="xppagecosmetics.xsp" target="colours"
                        style="color:rgb(0,128,128);font-weight:bold">
                        <xp:span style="color:rgb(94,94,255)">
                            All Page Cosmetics
            </xp:span>
```

Figure 7



```
    /**

     * Method: SecurityAccessOkay()
     * Created from Copy: 2012.08.26.2.08.AM
     * Security Access Boolean for Xpages Controls (Hide/Show)
     *
     */
    // links control, items on form vanish if Security access is 0
    // for viewaccess, updateaccess, addaccess, and deleteaccess in NSF back-end
    public boolean SecurityAccessOkay() {

        // let's add a try catch here, to grab errors near the end
        try {

            // BEGIN DEBUG
              System.out.println("BEGIN DEBUG | SecurityAccessOkay started... ");
              Session session = DominoUtils.getCurrentSession(FacesContext
                        .getCurrentInstance());
              System.out.println("docucontent.nsf | SecurityAccessOkay() in LoginAppBean.java Session Obtained..." + session);
                // find database in question from current session
                System.out.println("docucontent.nsf | SecurityAccessOkay() in LoginAppBean.java finding FileBin.nsf based on Session...");
                Database database = session.getDatabase(session.getServerName(),"filebin.nsf");
                System.out.println("SecurityAccessOkay() in LoginAppBean.java Fetched Database..." + database);
                System.out.println("Getting a database connection... ");

              System.out.println("loginAppBean.java | SecurityAccessOkay Connected to " + database);
              // Find the view in question
              View view = database.getView(ViewName);
              System.out.println("loginAppBean.java | SecurityAccessOkay Connected to " + view);
              //determine database state for searching
              if (database.isFTIndexed())
                database.updateFTIndex(false);
                else
                  database.updateFTIndex(true);

              //Collect entries based on search criteria in VIEWACCESS = username101
              ViewEntryCollection vec =
                    view.getAllEntriesByKey(VIEWACCESS, false);
```

TIP: See the SecurityAccessView screenshot in the filebin.nsf documentation added earlier... Lastly, below chart and others alike are based on security settings set up in views throughout the App; simply put, modify class files or use **AlienHouse** userID to access page content.



Version:2013.11.05.12.54.AM