What is Baraza Framework: Baraza is a Swahili word that means a deliberation meeting held by a collective group of a people of wisdom. The name ***baraza*** is used because the system came around by many heads with big minds coming together to develop it so as to come up with a working reliable system.

The Baraza Framework is a development Java Development initiative that makes application development easy. The framework takes advantage of XML to define its components. Using the framework, desktop applications, applets can be easily deployed using the same XML document with only slight modification.

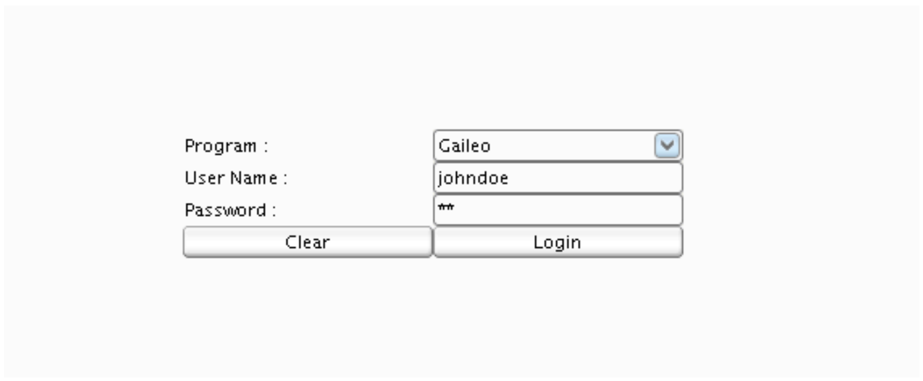
Baraza is divided in three different blocks there is the J*ava code block*, XML *block* and the *Database block*. The J*ava code block* is again divided into four different blocks, that is; *applet*, *servlet*, server and *IDE*.

Baraza is a system that incorporates different independent projects that work independently using the same J*ava code block* that uses the individual project XML *block* to produce a desired result. Feature of the *baraza frame* work:

- High development speed
- Low development cost
- Web enabled applications
- Very stable
- Well defined (well ordered design)
- Ease of operation (user friendly)
- Compatible with pc
- Reports, mailing lists, and forums are available.

Login First window which the user is supposed to enter the user name and password to use the system features that are only visible when you're logged in.

The window looks like this



Program :	Gaileo
User Name :	johndoe
Password :	**

Your ***user name*** is a unique identifier that you will use to login
Your ***password*** is used for authentication when you login.

For your own security, please choose a good password
It should include letters, numbers, and symbols

Passwords are case sensitive. So, if you use lowercase or uppercase letters, you must type them in exactly the same way when logging in.

DATABASE

Database provides standardized information about the system actually it's a large set of structured data, and run operations on the data requested by numerous users who use the system currently used in Baraza is PostgreSQL which probably handles virtually all the standard SQL constructs. It is easy (relatively speaking) to administer, it is fast, it is efficient.

BARAZA XML DEFINATION (**Extensible Markup Language*****)***

The XML file contains all the variable names, method names and properties names along with additional information about the system XML which store and send information.

The illustration below shows an example of an XML file

```
<? xml version="1.0"?>
<APP name="sacco" database="sacco">
 <TREE name="sacco">
  <NODE name="Main Entries">
   <NODE key="1" name="Members"></NODE>
   <NODE key="2" name="Periods"></NODE>
   <NODE key="3" name="Loan Types"></NODE>
  </NODE>
  <NODE name="Members Entries">
   <NODE key="4" name="Shares"></NODE>
   <NODE key="5" name="Loans"></NODE>
  </NODE>
  <NODE name="Reports">
   <NODE key="11" name="Members"></NODE>
   <NODE key="12" name="Contributions"></NODE>
  </NODE>
 </TREE>

 <DESK key="1" name="Members" w="585" h="520" splittype="vert" splitloc="200">
  <GRID name="members" pos="top" table="members" keyfield="memberid" filtered="true">
   <TEXTFIELD title="Member ID" w="240">memberid</TEXTFIELD>
   <TEXTFIELD title="Member Name" w="240">membername</TEXTFIELD>
   <TEXTFIELD title="Staff No" w="240">staffno</TEXTFIELD>
   <TEXTFIELD title="Start Date" w="240">startdate</TEXTFIELD>
   <TEXTFIELD title="Contribution" w="240">contribution</TEXTFIELD>
  </GRID>

  <FORM name="members" pos="bottom" table="members" linkfield="memberid">
   <TEXTFIELD title="Member ID" x="10" y="10" w="150" h="20">memberid</TEXTFIELD>
   <TEXTFIELD title="Staff No" x="290" y="10" w="150" h="20">staffno</TEXTFIELD>
   <TEXTFIELD title="Member Name" x="10" y="30" w="430" h="20">membername</TEXTFIELD>
   <TEXTFIELD title="Address" x="10" y="50" w="150" h="20">address</TEXTFIELD>
   <TEXTFIELD title="Zip code" x="290" y="50" w="150" h="20">zipcode</TEXTFIELD>
   <TEXTFIELD title="Town" x="10" y="70" w="150" h="20">town</TEXTFIELD>
   <TEXTFIELD title="Country" x="290" y="70" w="150" h="20">country</TEXTFIELD>
   <TEXTFIELD title="Contribution" x="10" y="90" w="150" h="20">contribution</TEXTFIELD>
   <TEXTDATE title="Start Date" x="290" y="90" w="150" h="20">startdate</TEXTDATE>
   <TEXTAREA title="Details" x="10" y="110" w="430" h="90">details</TEXTAREA>
  </FORM>

  <FILTER name="Members Filter" pos="bottom">
   <TEXTFIELD title="Member ID" x="10" y="10" w="240" h="20">memberid</TEXTFIELD>
   <TEXTFIELD title="Member Name" x="10" y="30" w="240" h="20">membername</TEXTFIELD>
   <TEXTFIELD title="Staff No" x="10" y="50" w="240" h="20">staffno</TEXTFIELD>
  </FILTER>
 </DESK>
```

<? Xml version="1.0"?> this first line in the document - the XML declaration - defines the XML version and the character encoding used in the document. In this case the document conforms to the 1.0 specification of XML

<APP name="sacco" database="sacco"> this line describes the root element of the document actually showing the name of it and the database name.

<TREE name="sacco"> This lines describes how the names are arranged and displayed in hierarchical data
Every tree has a *root* node from which all nodes descend
Example of a tree

<NODE name="Main Entries">this are nodes attached to the tree which actually provides a view of the data.
The node is the row displayed by the tree contains exactly one item of data.

<DESK key="1" name="Members" w="585" h="520" splittype="vert" splitloc="200"> this where the node is created it has key, name which shows the name of the node it also has the height and width (which shows the size of the particular node, splittype shows how the node is being divided while the splitloc shows the size of the of the split.

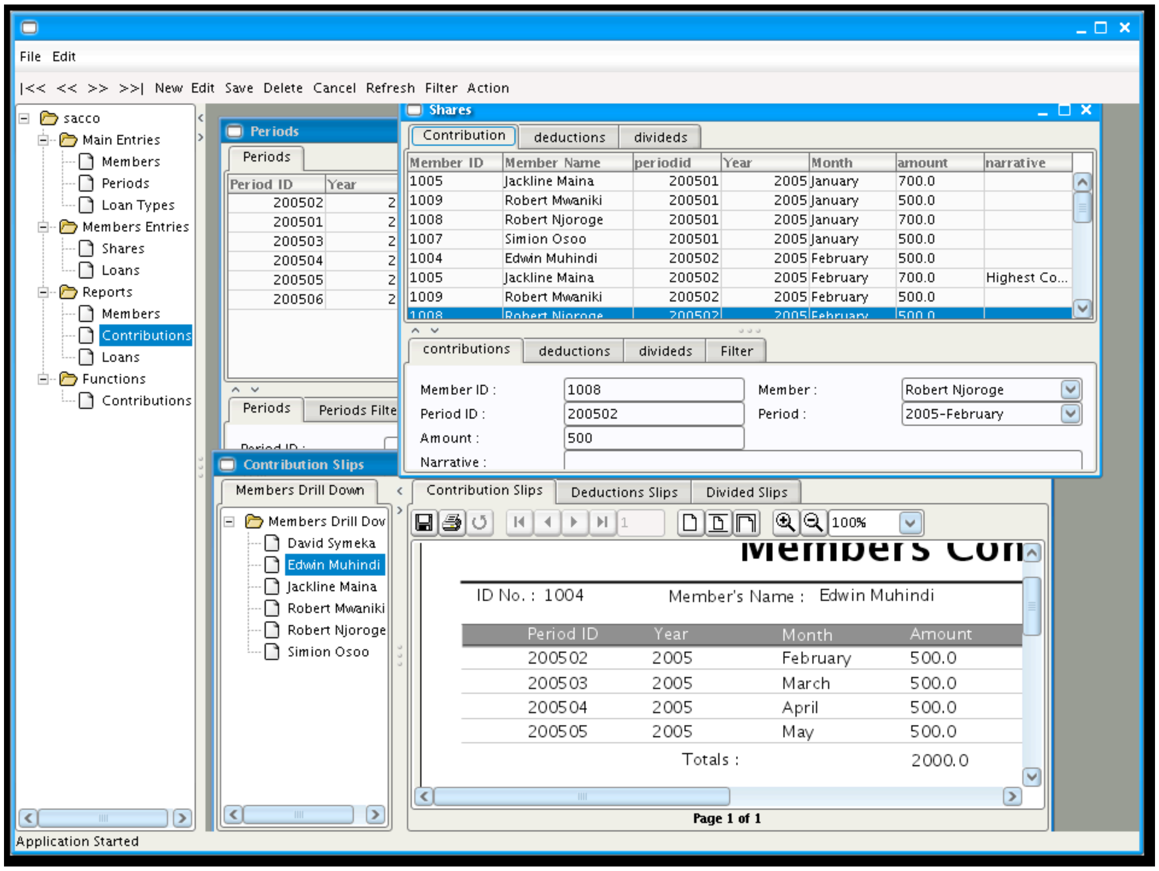
<FORM name="members" pos="bottom" table="members" linkfield="memberid"> this is where the form is being created whereby it has the form name of the table where the database is being referred to.

<FILTER name="Members Filter" pos="bottom"> this is used as a search engine in the node.

There are a number of elements that are defined for the form and filter:

<EDITOR> - HTML Text component
<TEXTFIELD> - Single line text
<TEXTAREA> - Multi line text
<CHECKBOX> - True False checkbox
<TEXTDATE> - Date entry
<DATESPIN> - Date selection
<TEXTTIME> - Time selection
<TIMESPIN> - Time selection
<COMBOBOX> - Item selection from a lookup list
<TEXTDECIMAL> - For decimal values

Baraza Example



Period ID	Year
200502	2
200501	2
200503	2
200504	2
200505	2
200506	2

Member ID	Member Name	periodid	Year	Month	amount	narrative
1005	Jackline Maina	200501	2005	January	700.0	
1009	Robert Mwaniki	200501	2005	January	500.0	
1008	Robert Njoroge	200501	2005	January	700.0	
1007	Simion Osoo	200501	2005	January	500.0	
1004	Edwin Muhindi	200502	2005	February	500.0	
1005	Jackline Maina	200502	2005	February	700.0	Highest Co...
1009	Robert Mwaniki	200502	2005	February	500.0	
1008	Robert Njoroge	200502	2005	February	500.0	

Member ID : 1008
Member : Robert Njoroge
Period ID : 200502
Period : 2005-February
Amount : 500
Narrative :

Members Drill Dov
- David Symeka
- Edwin Muhindi
- Jackline Maina
- Robert Mwaniki
- Robert Njoroge
- Simion Osoo

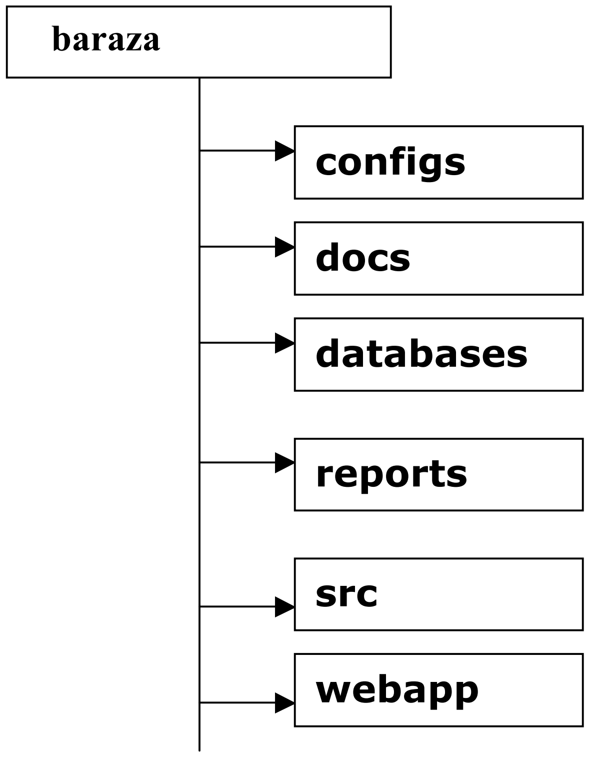
Members Con

ID No. : 1004 Member's Name : Edwin Muhindi

Period ID	Year	Month	Amount
200502	2005	February	500.0
200503	2005	March	500.0
200504	2005	April	500.0
200505	2005	May	500.0
		Totals :	2000.0

Page 1 of 1

DESCRIPTION OF THE BARAZA DIRECTORY



Configs
This is a directory intended to contain all xml files which are used in the system.

Databases
Mandatory directory for the system where the original purpose is to store data used by the system.

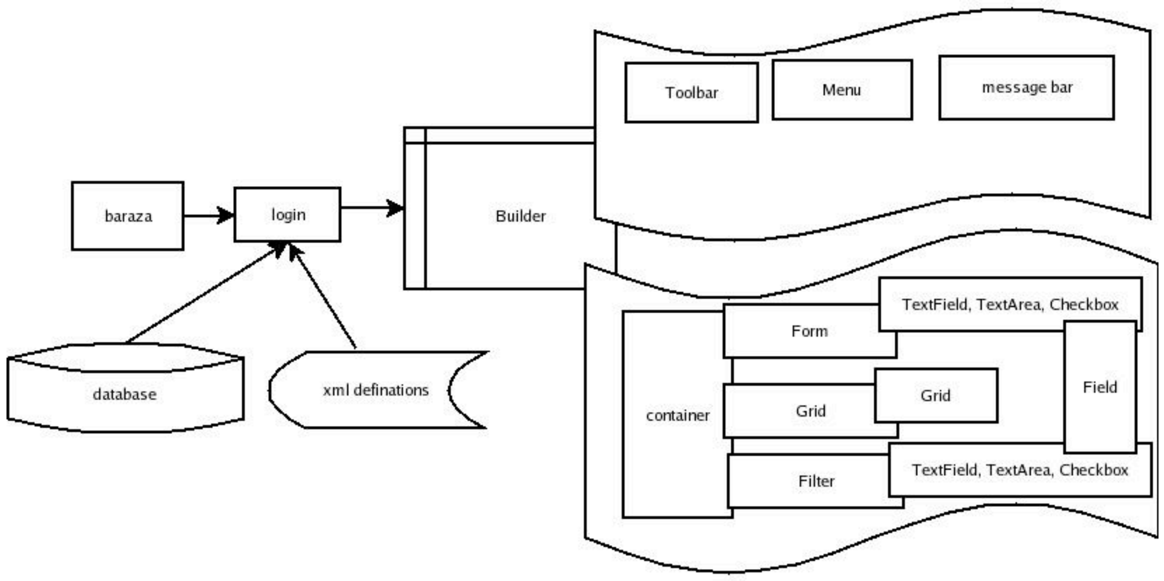
Docs
Has a wide range of use this directory contains subdirectories which include changes, issues, readme, and todo.
Changes reflects the latest development (changes) done to the system, *issues* shows what is partly done and due to completion although it's functioning, *readme* shows what the system does, actually what the system is performing.
Todo this is what is supposed to be done towards improving the system which is actually work pending for the system,

Reports
It's a mandatory directory where all reports for the system are stored

Src
This where the main baraza class is stored actually the whole system code is stored in here.

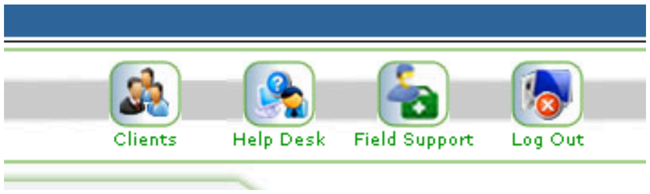
Webapp
Consists of (configs, jar, images, index.html, Lib, search temp, web-Inf)

Component Flow



1.1. The Top Level Menu

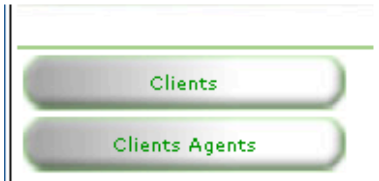
The Top level menu gives the user options on the existing application modules. By clicking on any one of these options, the servlet application returns a page displaying the sub menu options available for the selected module. It may display a default data page which may be a form or data grid table.



Example of a top level Menu

1.2. Sub Menus

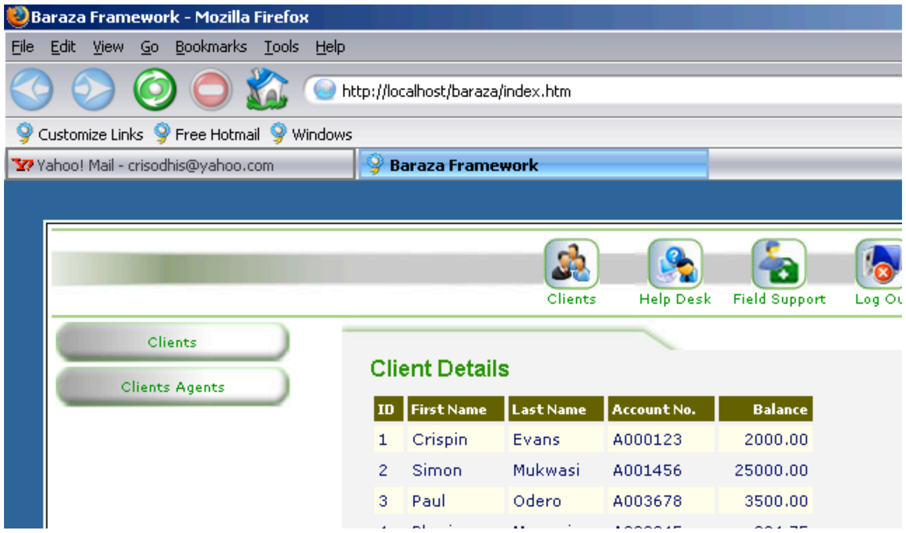
Each page within a module has sub menu items to enable the user carry out various tasks within the current module.



Example of a sub menu items.

1.3. *The Information Display Area*

This area of a page displays data for example a web form or table grid.


Baraza Framework - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost/baraza/index.htm
Customize Links
Free Hotmail
Windows
Yahoo! Mail - crisodhis@yahoo.com
Baraza Framework
Clients
Help Desk
Field Support
Clients
Clients Agents

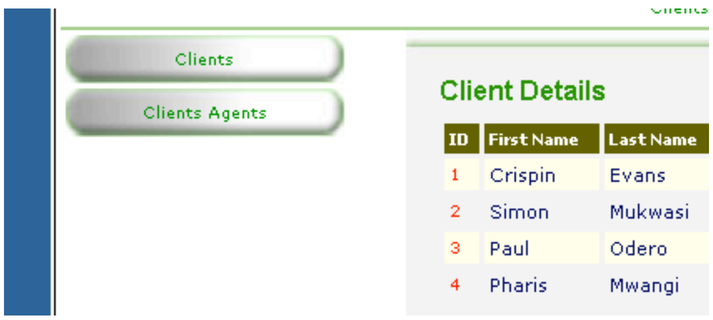
Client Details

ID	First Name	Last Name	Account No.	Balance
1	Crispin	Evans	A000123	2000.00
2	Simon	Mukwasi	A001456	25000.00
3	Paul	Odero	A003678	3500.00

Example Showing the client details grid in the information display area.

1.4. *Data Links*

Within forms or table grids, special hyperlinks exist that enable navigation to specific information groups.



Client Details

ID	First Name	Last Name
1	Crispin	Evans
2	Simon	Mukwasi
3	Paul	Odero
4	Pharis	Mwangi

Example showing data links under the ID Field.

2. Page Layout

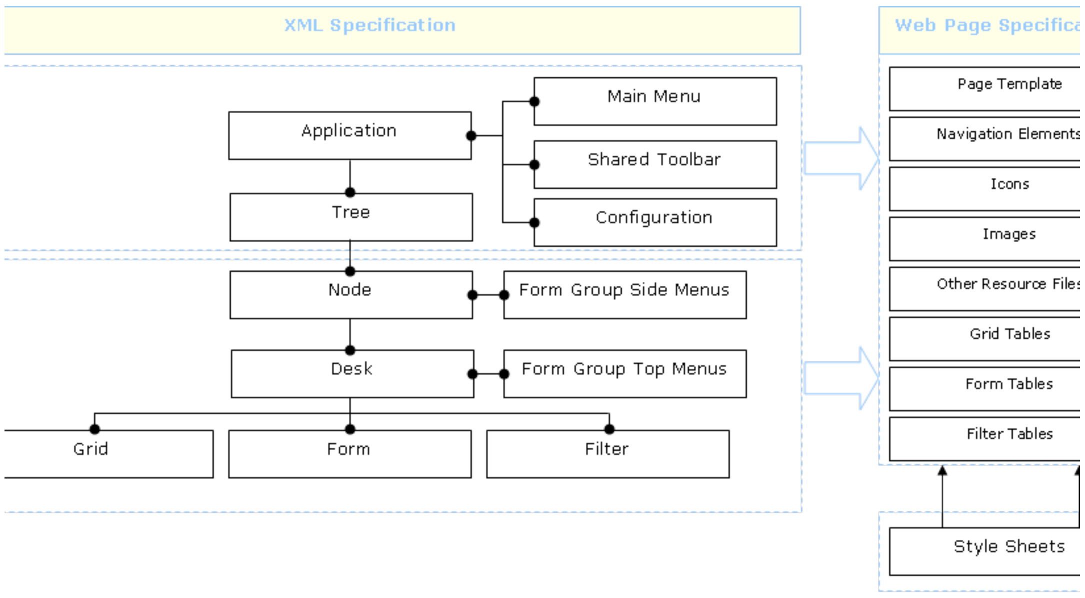
To enable uniform layout as well as ease of customization, the baraza web framework uses Cascaded Style Sheets (CSS) heavily. The Style sheets are used to determine:

1. **Link Appearance and Behavior**
2. **Table Structure and Appearance**
3. **Cross Browser Presentation**
4. **General page appearance**

Currently all styles are stored in a file called **baraza.css.** Each page links to this style sheet if it needs to take advantage of the variety of predefined styles.

The Style Definitions are important because they make page development to depend on pure HTML as much as possible to limit the use of client side scripting like Java Script. For Example the side buttons are simply created as blank graphics with no label, these are then placed as background images to a table cell. The button label can be written within the cell directly and easily as opposed to using java Script for image slicing. For this background image to be presented properly, CSS is used to determine the precise cell width and height.

3. Application Development Model



The above diagram illustrates the application development model. The application is developed by specifying it in an XML document. This is then processed by the java runtime environment to produce the resultant web components as highlighted in the web specification. The Core Development Model should take into account the following key issues.

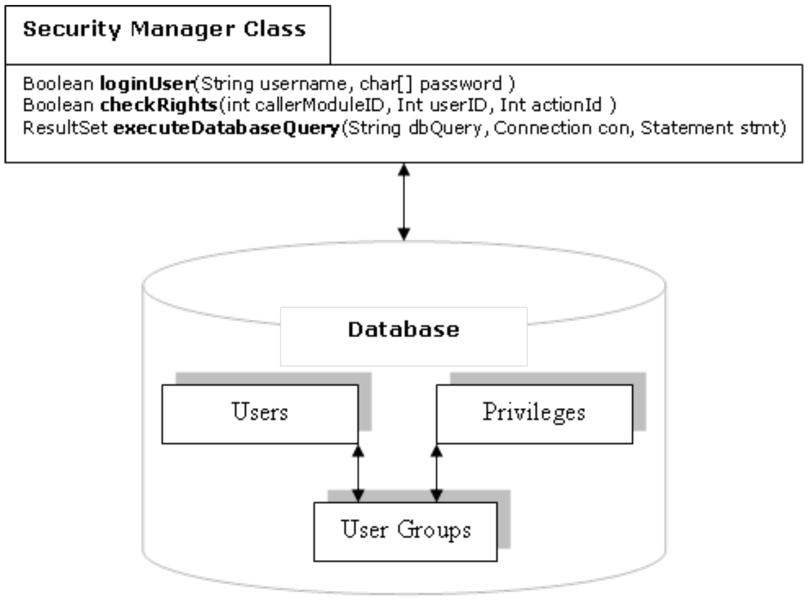
1. Security Management
2. Report Generation
3. Errors
4. Command Processing
5. Data Capture
6. Audit trail Analysis
7. Web Page creation

The following discussion will specify the generic or basic requirements of the above issues. These requirements should be met to ensure that the applications developed using this framework are stable and easily extensible.

3.1. Security Management

The security management class should offer the following core functions.

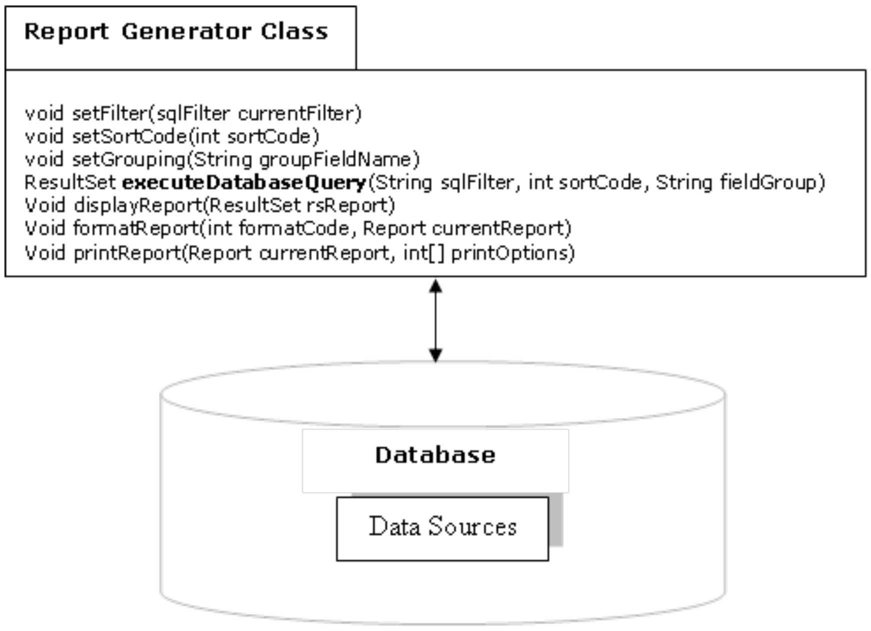
1. User Login
2. User Rights Verification
3. Ability to read a data source e.g. database or data file on disk.


Security Manager Class
Boolean loginUser(String username, char[] password)
Boolean checkRights(int callerModuleID, Int userID, Int actionId)
ResultSet executeDatabaseQuery(String dbQuery, Connection con, Statement stmt)
Database
Users
Privileges
User Groups

3.2. Report Generation

The report generation class should offer the following functionality.

1. Allow data filtering by a certain data field or combination of several fields and attached conditions.
2. Allow data sorting numerically or alphabetically.
3. Allow for the grouping of a data set into subsets.
4. Ability to read a data source
5. Ability to display a report for printing or on screen.
6. Ability to format a report.
7. Ability to print a report

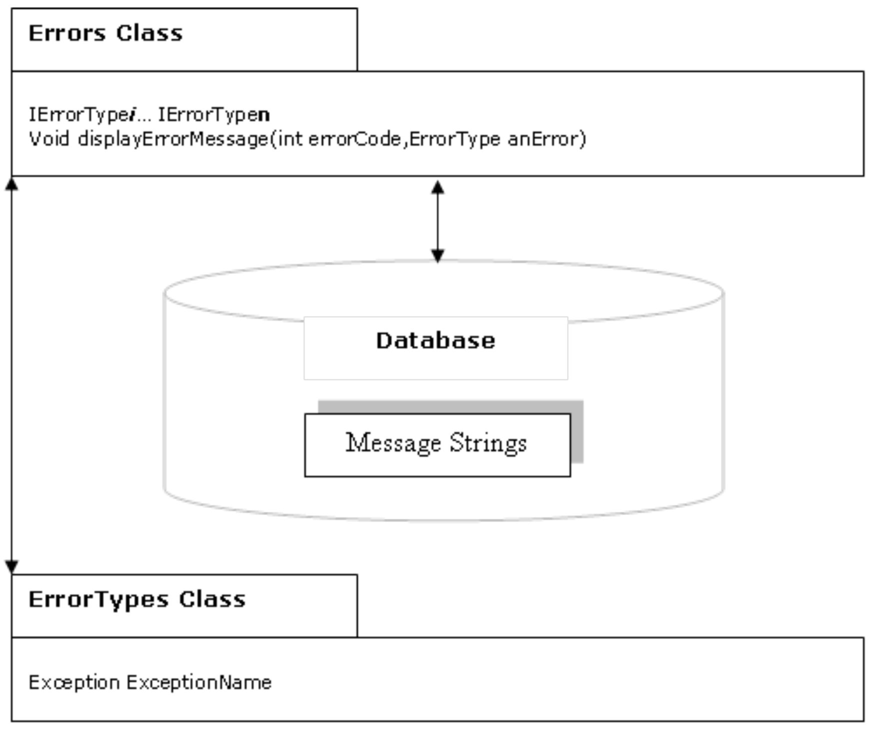

Report Generator Class
void setFilter(sqlFilter currentFilter)
void setSortCode(int sortCode)
void setGrouping(String groupFieldName)
ResultSet **executeDatabaseQuery**(String sqlFilter, int sortCode, String fieldGroup)
Void displayReport(ResultSet rsReport)
Void formatReport(int formatCode, Report currentReport)
Void printReport(Report currentReport, int[] printOptions)
Database
Data Sources

3.3. Errors

To make the application framework as user-friendly as possible. It should provide a means for user defined error messages. Special Interfaces can be created tthat define these errors as types.

Ideally the definition should follow the following model:

1. Error Type
2. Corresponding Messages


Errors Class
IErrorType*i*... IErrorType**n**
Void displayErrorMessage(int errorCode,ErrorType anError)
Database
Message Strings
ErrorTypes Class
Exception ExceptionName

3.4. Command Processor Class

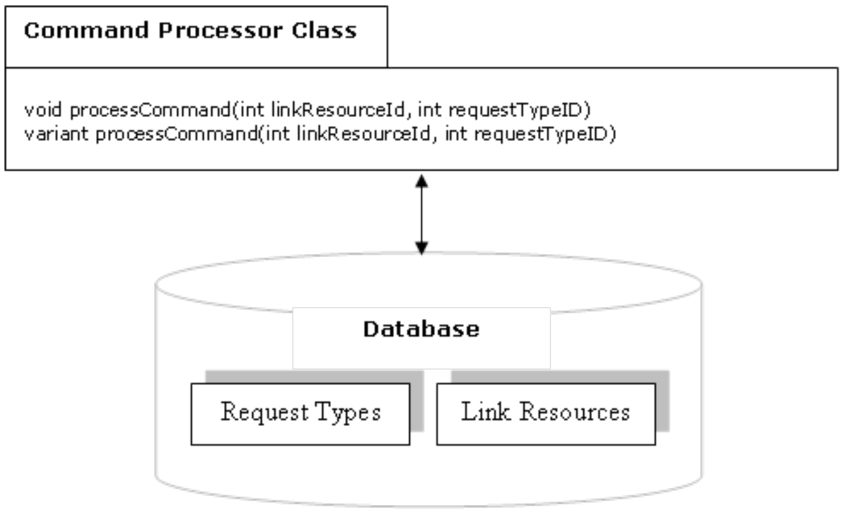
This corresponds to a generic library class that processes function calls. The model for this class is provision of 2 functions types. The first type does not return a value and the second type does. In whichever case, the function is told what resource is required and the code of the operation.
Eg..

If an image resource such as a button has an id of 34 and a resize operation has code 0020 (where 00= resize and 20 = twice) and we want the page being loaded to display buttons of this type at twice the size the resultant function call would be.

Image process Command(34,0020)

The specification of this class requires the definition of

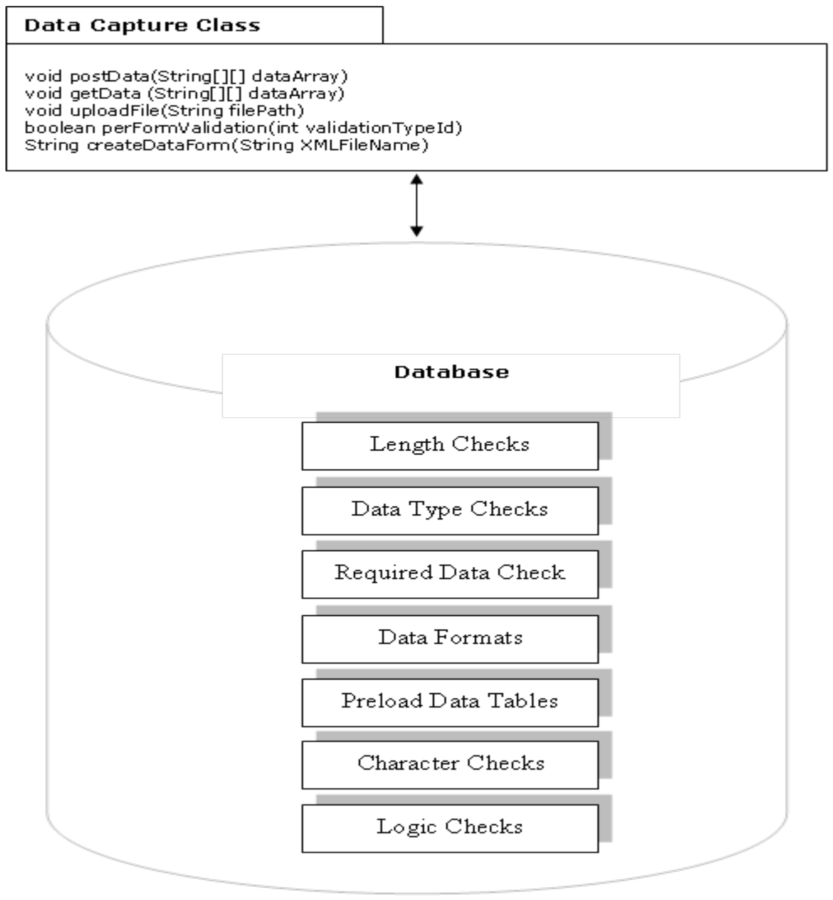
1. Request types
2. Resource Link Definitions


Command Processor Class
void processCommand(int linkResourceId, int requestTypeID)
variant processCommand(int linkResourceId, int requestTypeID)
Database
Request Types
Link Resources

3.5. Data Processing

The core of this model is the handling of data. Key issues must be addressed to ensure that the data is properly handled. The Key data processing activities are:

1. Data Posting or Getting from Web Forms.
2. HTTP Header processing such as requests for a page or specification of redirects.
3. File Uploading
4. Validation



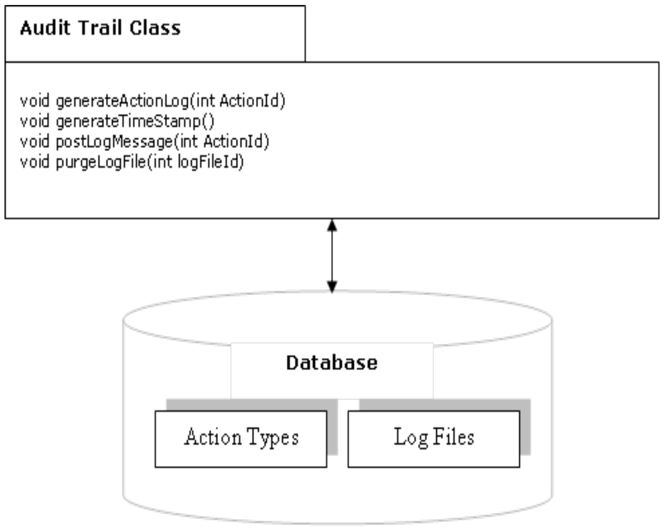
```
void postData(String[][] dataArray)
void getData (String[][] dataArray)
void uploadFile(String filePath)
boolean perFormValidation(int validationTypeId)
String createDataForm(String XMLFileName)
```

3.6. Audit Trail Analysis

The Audit trail Class is key to making the application development frame conform to standards in data security and accountability. In Practice it implements an integral logging function as depicted by the diagram below.

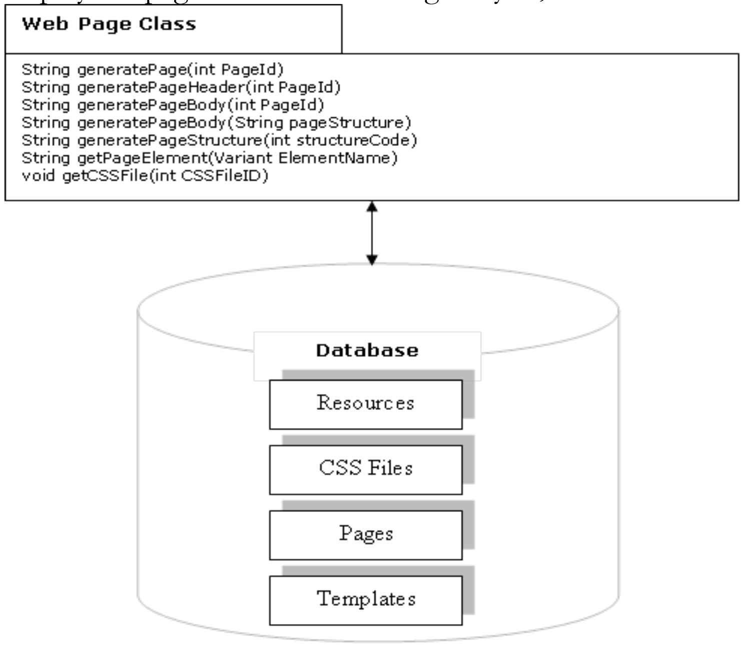
The Specification for this class requires:

1. Definition of Action Types
2. Action group to Log File Mapping.


Audit Trail Class
void generateActionLog(int ActionId)
void generateTimeStamp()
void postLogMessage(int ActionId)
void purgeLogFile(int logFileId)
Database
Action Types
Log Files

3.7. Web Page Definition

The page definition process has several key functions that are used to create, format and display the pages. The Class manages styles, resources and applies templates.


Web Page Class
String generatePage(int PageId)
String generatePageHeader(int PageId)
String generatePageBody(int PageId)
String generatePageBody(String pageStructure)
String generatePageStructure(int structureCode)
String getPageElement(Variant ElementName)
void getCSSFile(int CSSFileID)
Database
Resources
CSS Files
Pages
Templates