WKA Studio for Beginners

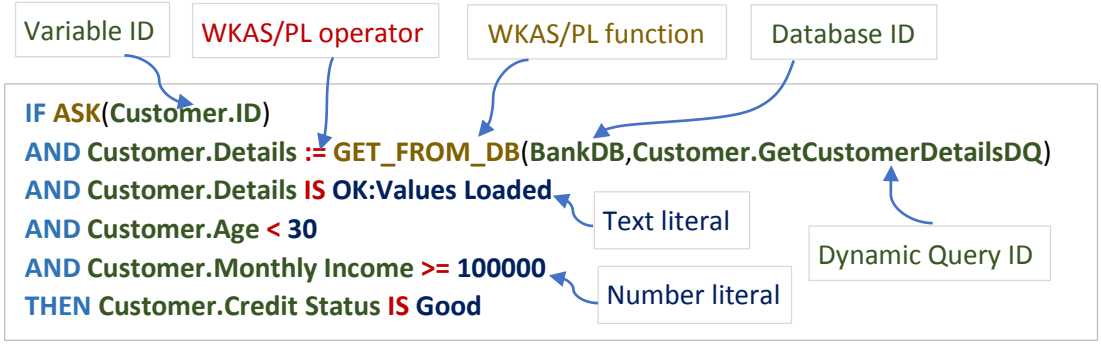
The first and foremost, the WKA Studio app and its development are fundamentally different from conventional apps work and their developments using higher level programming languages such as Java/C++. WKA Studio can be used to develop chat-bots.

1. Apps/chat-bots are developed using knowledge-based approach (WKAS uses expert system technology at its core: expert system is a classical AI technology, refer and read tutorial on this to know more how to develop knowledge-based apps using expert system approach) and 4GL rule-based language (called WKAS/PL). The source code of app is nothing but a set of rules implementing app's business logic. Rules can be changed on-the-fly like scripting languages even for live apps.

   The typical syntax of rule is:

```
IF    Conditional statement 1
AND   Conditional statement 2
AND   ...
THEN  Assignment statement 1
AND   Assignment statement 2
AND   ...
ELSE  Assignment statement 3...
```

Sample rule:



Each rule has unique ID. Complex functionality can be included using user defined functions (implemented like any typical language functions).

2. Knowledge-based apps typically work in Q&A mode asking only one or few inputs at a time using easy-to-use and understand GUI. However, WKAS also allows complex form-based interfaces.

3. Typical conventional programming languages variables are declared inside the code and allow only data type and initial value to be set when they are declared. WKAS variables are not defined inside code but externally through global variable interface.

   More about WKA Studio variables.

   a. Each variable has large number of configurable parameters defined such as data type, possible values, default value, range of values, validation options such as email, number, minimum one special character etc.
   b. In conventional programming, value of variable has to be asked explicitly by creating UI or through statements to read value(s) like cin>> in C++ or readline in Java before it is used in any statement or expression. WKAS variable gets value (or value is asked) at run time automatically from various sources such as HTML web-page (created automatically based on UI parameters), database, report,

user defined function and so on. Value of uninitialized variable is automatically asked, fetched or computed by expert system engine only when it is required (e.g. appearing in the expression/statement). For example, in above sample rule, Customer.Monthly Income will be asked only when Customer.Age is greater than 30. However, variable can be explicitly asked using ASK statement for example first statement in sample rule is asking Customer.ID because it is used to fetch customer data from database using dynamic query: Customer.GetCustomerDetailsDQ. Forms, reports, parameterized queries, text and excel files, etc. can also be connected to variables to get values in groups (using forms), to display reports.

c. Developer can also set lot of UI parameters such as UI control type (radio box/combo box/check box etc.), number of options per line, web-page template, JavaScript for validations, type of main buttons on the page etc.
d. For each variable possible value (e.g. colour), the details can be added such as image, description, inference, group etc.
e. No need to write code/create Web-page to get value of variable from user. Responsive web-page is created with all in-built validations and invoked by expert system engine whenever needed.

f. Variables can be used at all three tiers (client-side, web-server: business logic and to access database) by referring their names: i> in JavaScript code at front-end such as for validation, ii> in the business logic code and iii> as filters in database queries for example variable: Customer.Age can be used in JavaScript to set certain value say risk: if([Customer.Age]>60) SetValue('Customer.Risk',Higher). WKAS automatically creates and adds JavaScript by populating functions e.g. variable is replaced with GetValue('Variable Name') to get variable values.

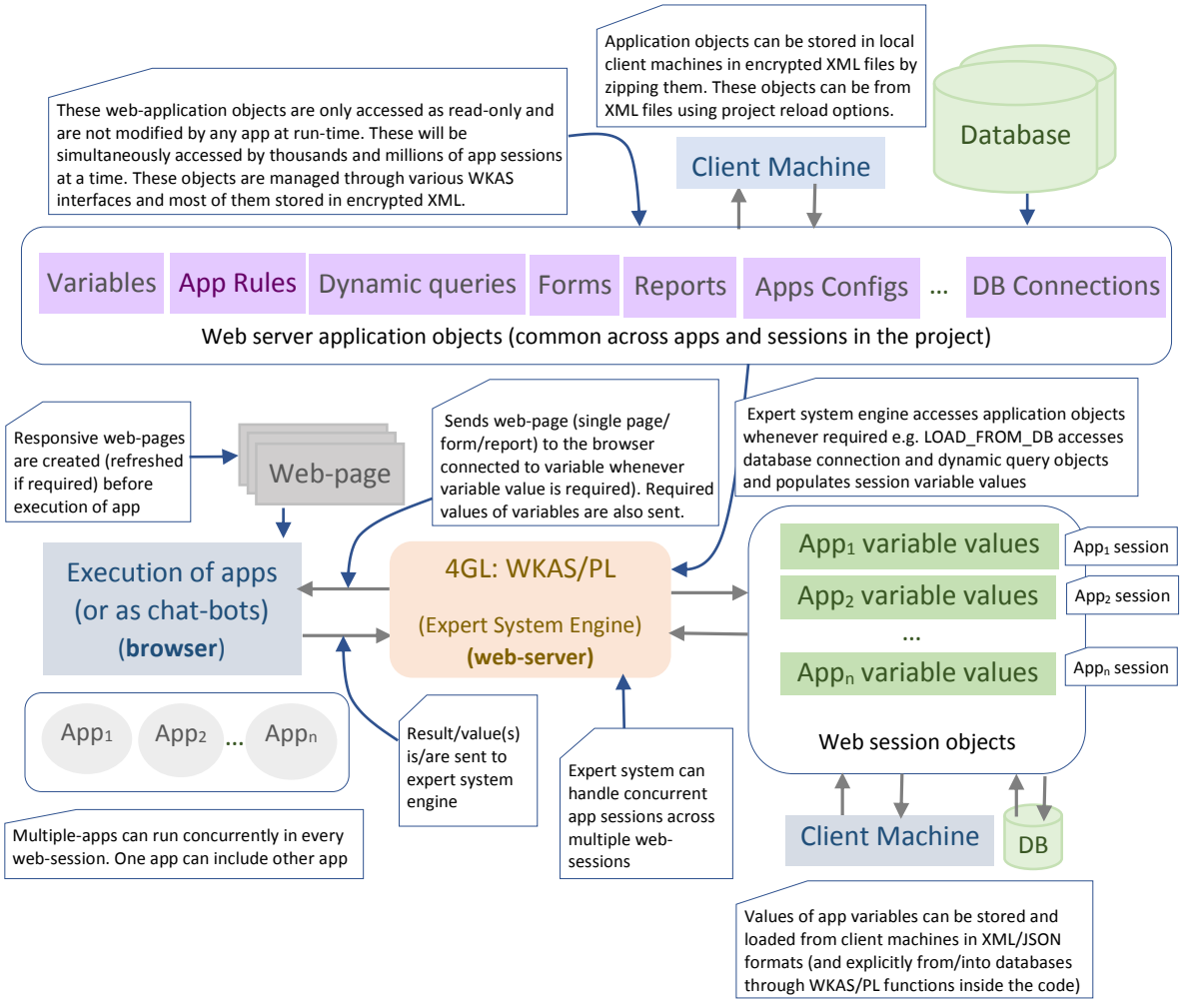
g. Related variable names are grouped under variable group name that corresponds to logical entity which holds related information e.g. Customer, Item etc. All Customer variables such as Name, Age, ID, Monthly Income will be grouped under Customer variable group. Variables are always referred using complete name prefixed with group name they belong to, for example, Customer.Age. Note Age cannot be called variable but Customer.Age is.

h. Variables are automatically created when apps are created using app create interface. For example, when Database app is created, variables are created corresponding to field names of selected database table under given variable group (may be similar to table name) as parameter during app creation.

i. At run-time, app holds variable values called as session variables (or app session or session).
j. Once app starts its execution, it asks, gets or computes values whenever required and holds them as long as app session is going on. Once app session gets over and app is restarted again, all variables app was using are initialised to null. However, WKA Studio expert system engine maintains the values from previous app session called as last values. These values are automatically selected as default value when input is asked through web-pages. There is an option in variable configuration to retain values across the app sessions (e.g. user ids, passwords need to kept for multiple app sessions) even across the apps.
k. Variable values can be saved in files on client machines in XML/JSON format or in databases. These values can be loaded back into current session. App will not ask values which are loaded from XML file.
l. App itself is started using variable typically has name same as app name. This variable is referred as goal variable.

4. Lot of objects such as database connections, predefined parameterized database queries to (execute DB functionality, bring data-sets), menu lists which store possible values a variable can take, forms, reports, user defined functions etc. are

configured using various interfaces of WKA Studio (refer to document: Understanding WKA Studio Interfaces_Quick Intro) These objects are classified into two broad categories called WKAS variable group objects and WKAS application group objects. Their IDs are prefixed with either variable group or application group. For example, menus are prefixed with variable groups e.g. Customer.GenderOptions where Customer is variable group. and forms are prefixed with application groups e.g. Payroll.MonthlyDataEntryForm where Payroll represents application group.



Each object is accessed using ID and invoked using just their IDs in code using WKAS/PL e.g. database can be accessed using logical database ID, a query can be executed using its ID and values will be populated at run-time in query before execution. All these objects may be referred in each other (for example, parameterised query can be connected to form to fetch and populate data from database in the form). This does not require lot of code inside business logic thus reducing code. Sample rule shown on page 1 includes Database ID, Dynamic query ID to fetch data from database and populate into variables using LOAD_FROM_DB function.

Major objects which are frequently used in app rules are:

a. Global Variable and Menus (predefined list of values)
b. Rules, User defined functions (UDFs)
c. Database connections, dynamic queries (DQs), etc.
d. Forms and Reports

5. Apps can upload various files into database, import data from excel files and text files at run-time. There are predefined configurations which check type of text file and excel files being imported at run-time.

6. Best way to understand WKAS apps:

    a. Go through the document which explains Fruit Recognition System: How does app based on expert system work? Quickly glance through the document which illustrates how rules can be put in excel in specific format and how it can be used to develop Fruit Recognition System using App Create interface.
    b. Create apps using 'Create and Manage App'. One excel based app like math app and database app by taking simple CSV file like student, employee etc. with basic options on. This will help to see how various objects created variables, menus, dynamic queries, form, report, etc. and understand them. Be careful when selecting right options. If not done properly, need to sort out things manually latter on.

App Create and Manage interface helps to create various apps with basic functionality (e.g. database apps to view, add, update, remove records and reports) quickly. These apps can be modified to add complex functionality. Developers can develop app in few minutes, hours or days based on type of applications and complete customizable or made-to-order enterprise apps (like ERPs) in few months. These apps can be modified on-the-fly

whenever required as nothing is stored in compiled form like conventional programming languages.

a. Rule-based apps can be quickly created from excel sheets where each row represents a rule. Refer to tutorial.
b. Database apps can be created using CSV/text/Excel files in few steps. Refer to tutorial.
c. Quiz apps can be created from quizzes details through excel files. Refer to tutorial.

7. What is next?
    a. Make sure you know SQL well and especially MySQL DBMS.
    b. Read tutorials and manuals: WKAS/PL, functions and operators, WKAS interfaces.
    c. Go through prototype apps developed.
    d. Practice app development.
    e. Start developing serious apps and chat-bots.