



Web Application Development Tutorial (Beginners)

Using

DevoSA

Vaakya Rich IDE

Document Last Updated: 21st July 2010

Table of Contents

Introduction.....	3
Intended Audience.....	3
References.....	3
Abbreviations Used.....	3
Develop a web/portal application using Vaakya.....	4
Create Type.....	5
Create Static Html Index Page.....	6
Create Data Object.....	7
Create a Dynamic Html Page.....	8
Create WebProc.....	9
Create Proc.....	10
Extracting HTML Form Data.....	11
Create Executable to Test Project.....	13
DevoSA Manager.....	13
How it works?.....	14
Populating a Html template with data.....	16
Vaakya Block Variables and HTML Template.....	22
Summary.....	24

Introduction

Vaakya is a “Framework based” application development & deployment environment, supported by an IDE (integrated development environment) and a descriptive scripting language.

Intended Audience

This is a practical guide for application programmers, developers and application architects who wish to use Vaakya to create enterprise web applications.

Pre-requisites

To use Vaakya, prior knowledge of one or more programming languages & basic database concepts are recommended.

References

Please refer the following documents along with this document

- Vaakya Keywords & Functions Guide
- Vaakya DevoSA Guide

Abbreviations Used

<i>Abbreviation</i>	<i>Description</i>
IDE	Integrated Development Environment
CFG	Configuration
.prj	Vaakya Project Development File
.vuf	Vaakya Project Deployment File
.vxe	Vaakya Executable File
.cmp	Vaakya IDE Component File
.pck	Vaakya Packed File

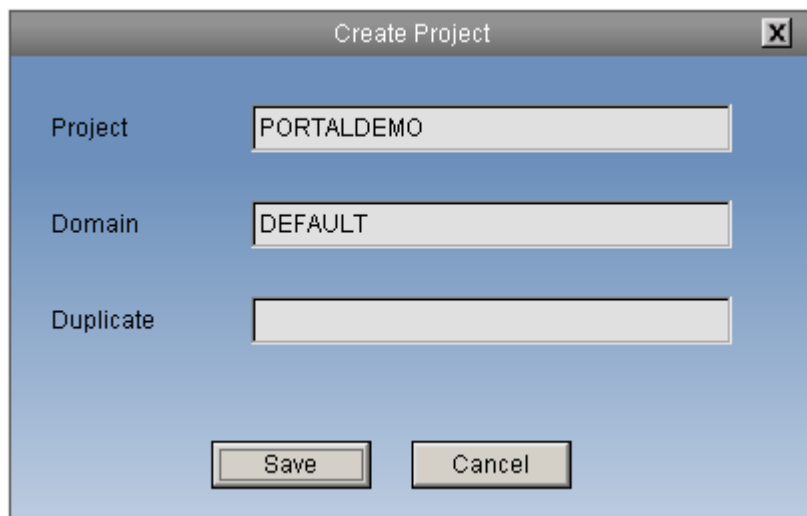
Develop a web/portal application using Vaakya

This document will cover the basics of developing a web or portal application using DevoSA (Vaakya Rich IDE).

You will develop a sample application by just following the simple steps explained in this document.

Steps to Develop:

- To download the DevoSA (Rich IDE), Please click on the below URL <http://www.vaakya.com>
- Install & configure the IDE (DevoSA) – Refer to DevoSA Installation Guide
- Start Vaakya Zone Server using Zone Manager (Zone Server contains a built-in Http Web Server)
- Click on DevoSA to start the IDE
- Go to → Project → Create
- Create a new project named '**PORTALDEMO**'



- Load Project 'PORTALDEMO'. Go to Project → Load → PORTALDEMO

Create Type

Type component is used to store non-persistent data for manipulation in memory.

- Go to Components →Type→New
- Create a new Type 'typCommon' with the fields given in the table below

typCommon

Column	Data type	Size
UserName	Text	40
UserEmail	Text	60
UserMobileNo	Text	16
UserDesignation	Text	30
Message	Text	40

typCommon

Parent Object

Column	DataType	Dims	Size	ParColumn
UserName	Text	0	40	
UserEmail	Text	0	60	
UserMobileNo	Text	0	16	
UserDesignation	Text	0	30	
Message	Text	0	40	
		0	0	
		0	0	
		0	0	
		0	0	

Create Static Html Index Page

Now let us create a Static HTML index page.

- Create an 'index.html' page as shown below

index.html

```
<html>
<head>
<title>Register User</title>
</head>
<body>
<form method="post" action="/cgi-bin
/cgiRouter.cgi/VAAKYA/DEVOSA/CORPUS/PORTALDEMO/DEFAULT/vspRegisterUser">
<label>User Name: </label>
<input name="typCommon.UserName" type="text" maxlength="40" /><br />
<label>Email Id: </label>
<input name="typCommon.UserEmail" type="text" maxlength="60" /><br />
<label>Mobile No: </label>
<input name="typCommon.UserMobileNo" type="text" maxlength="16" /><br />
<label>Designation: </label>
<input name="typCommon.UserDesignation" type="text" maxlength="30" /><br />
<input type="submit" value="Register" name="submit">
</form>
</body>
</html>
<html>
<head>
```

- Save 'index.html' page
- Paste the html file inside folder
\\VAAKYA\ZONESERVER\DEVOSA\CORPUS\STATIC\ PORTALDEMO\DEFAULT

Explanation:

- Each input name in the 'html' is associated with the fields in Type 'typCommon'
 - The fields in the Type component as well as the input names should exactly be the same, because, when the html page is submitted, component 'typCommon' will hold the data for further manipulation
- 📌 **Note:** If the above STATIC folder is not found. Create the folders PORTALDEMO\DEFAULT inside STATIC folder

Create Data Object

Object component is used to store persistent data

Now, create a Vaakya data object component named "UserDetails" to store data.

'User Details' object holds details like Name, Email, Mobile Number and Designation.

- Go to Components → Object → New
- Add a data object "UserDetails"

UserDetails

Column	Data type	Size
UserName	Text	40
UserEmail	Text	60
UserMobileNo	Text	16
UserDesignation	Text	30

- Add the fields in Storage Constructor as shown below

Column	Type	Dims	Size	Bind
UserName	Text	0	40	
UserEmail	Text	0	60	
UserMobileNo	Text	0	16	
UserDesignation	Text	0	30	
		0	0	

- Click on 'Submit'

Create a Dynamic Html Page

Earlier, we created a static 'index.html' page.

Now, create a dynamic html page called 'home.html', which will contain at least one dynamic variable to display a message and three static links.

- Create a html page 'home.html'
- Paste this html file into
\\VAAKYA\ZONESERVER\DEVOSA\CORPUS\IDES\TEMPLATES\PORTALDEMO

home.html

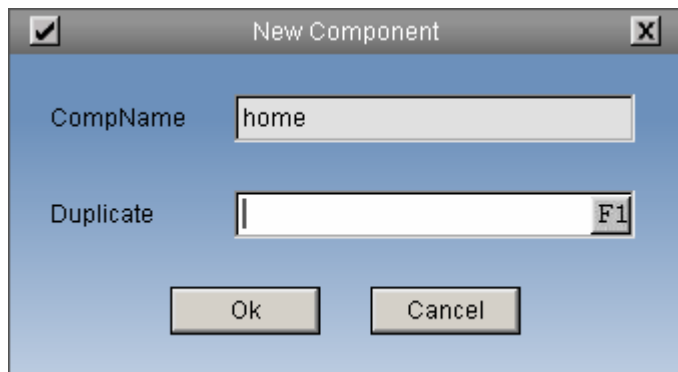
```
<html>
<head>
<title>Register User</title>
</head>
<body>
((-VAAKYA_VAR(typCommon.Message)-))
<br />
<a
href="/VAAKYA/DEVOSA/CORPUS/STATIC/PORTALDEMO/DEFAULT/index.html">Register
a New User</a><br />
<a href="vspModifyUserPage">Modify Last User</a><br />
<a href="vspViewUsers">View List of Users</a>
</body>
</html>
```

Note:

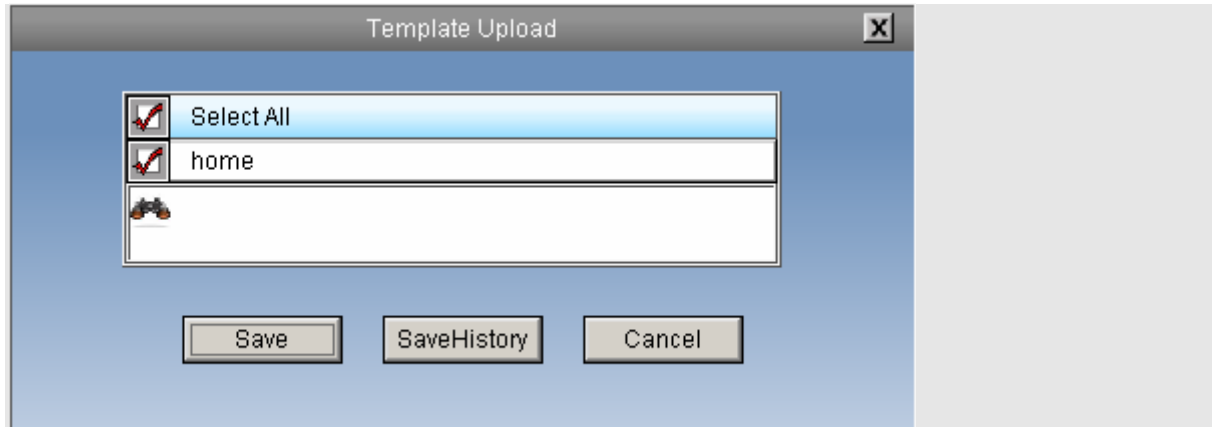
Vaakya variable is represented as ((-VAAKYA_VAR(type/object.fieldname)-)). The details of variable usage are explained in the following sessions.

Now, add this Template in the IDE

- Go to Components→ Template→ New and create the Template 'home'.



- Click on Template → 'home'
- Select the file 'home' and save the file.



Create WebProc

WebProc component is used to compose dynamic html pages and manipulate data using Vaakya dynamic variables

Create the following WebProcs

1. vspRegisterUser

- Go to Components → WebProc → New
- Create a new WebProc '**vspRegisterUser**' and click on OK.

2. vspModifyUserPage

- Go to Components → WebProc → New
- Create a new WebProc '**vspModifyUserPage**' and click on OK.

3. vspModifyUser

- Go to Components → WebProc → New
- Create a new WebProc '**vspModifyUser**' and click on OK.

4. vspViewUsers

- Go to Components → WebProc → New
- Create a new WebProc '**vspViewUsers**' and click on OK.

Create Proc

Proc component is used to program business logic, store, retrieve and manipulate data, set schedulers and create other stored procedures

1. proSaveUserDetails

- Go to Components → Proc → New
- Create a new Proc '**proSaveUserDetails**' and click on OK.
- Click on Proc component **proSaveUserDetails** and enter the arguments as follows
Arguments: typCommon @prTypCommon

2. proModifyUserPage

- Go to Components → Proc → New
- Create a new Proc '**proModifyUserPage**' and click on OK.
- Click on Proc component **proModifyUserPage** and enter the arguments as follows
Arguments: UserDetails @objUser

3. proModifyUser

- Go to Components → Proc → New
- Create a new Proc '**proModifyUser**' and click on OK.
- Click on Proc component **proModifyUser** and enter the arguments as follows
Arguments: UserDetails @objUser, typCommon @prTypCommon

Extracting HTML Form Data

Now, let us see how to collect data submitted from an HTML form and save the contents to a data object as shown below (index.html).

Go to IDE→Open WebProc component '**vspRegisterUser**' and do these steps

- Select Type (radio button)
- In the FormData input box, press F1 and select the Type '**typCommon**'
- In the alias column, enter '**prTypCommon**' (variable pointer for 'typCommon')

Explanation

- Alias is a variable pointer for a Type or Object
- Type the following code in the WebProc content

vspRegisterUser

?? Variable Declaration

```
Text @cpFooter
```

?? Call proc to save User Details

```
proSaveUserDetails(prTypCommon)
```

?? Initialize and merge valid messages and values to the template

```
ComposerInit("home", "HTTP/1.1 200 OK\nContent-Type: text/html\n\n", TRUE)
```

```
ComposerSubstituteRecord("home", prEnvData)
```

```
ComposerSubstituteRecord("home", prTypCommon)
```

?? Send the Template to the browser

```
ComposerSend("home", cpFooter)
```

- Save the WebProc

Explanation

'**proSaveUserDetails**' is a proc called by the WebProc '**vspRegisterUser**' to save data

Note : Symbol '**??**' are used as comment characters in Proc and WebProc Editors

Now, Go to Proc component,

- Click on Proc component '**proSaveUserDetails**' and enter the following code using the Proc Editor

proSaveUserDetails

??Variable declaration

```
Int          varRet
UserDetails  objUser
Array        @varArray
```

??Allocate memory to the variables.

```
varArray = New(Array, 1)
objUser  = New(UserDetails, 1)
varRet   = 0
??Assign values to object variable
```

```
objUser.UserName      = prTypCommon.UserName
objUser.UserEmail     = prTypCommon.UserEmail
objUser.UserMobileNo  = prTypCommon.UserMobileNo
objUser.UserDesignation = prTypCommon.UserDesignation
```

??Add assigned values to an array to save the record into database

```
ArrayAdd(varArray,objUser)
```

??Save record to the database using VdsSave function.

```
varRet = VdsSave(varArray, VDS_RECORD_TRAN)

If(varRet)
{
    prTypCommon.Message= "User Registered successfully"
}
Else
{
    prTypCommon.Message= "User registration failed"
}
```

- Click on 'Save'

Create Executable to Test Project

This process creates an executable for the above project, which can be viewed through DevoSA Manager explained below

- Click on Test Project → Create Executable
- Check 'Select All' and Click on Ok
- The project executable will be created.

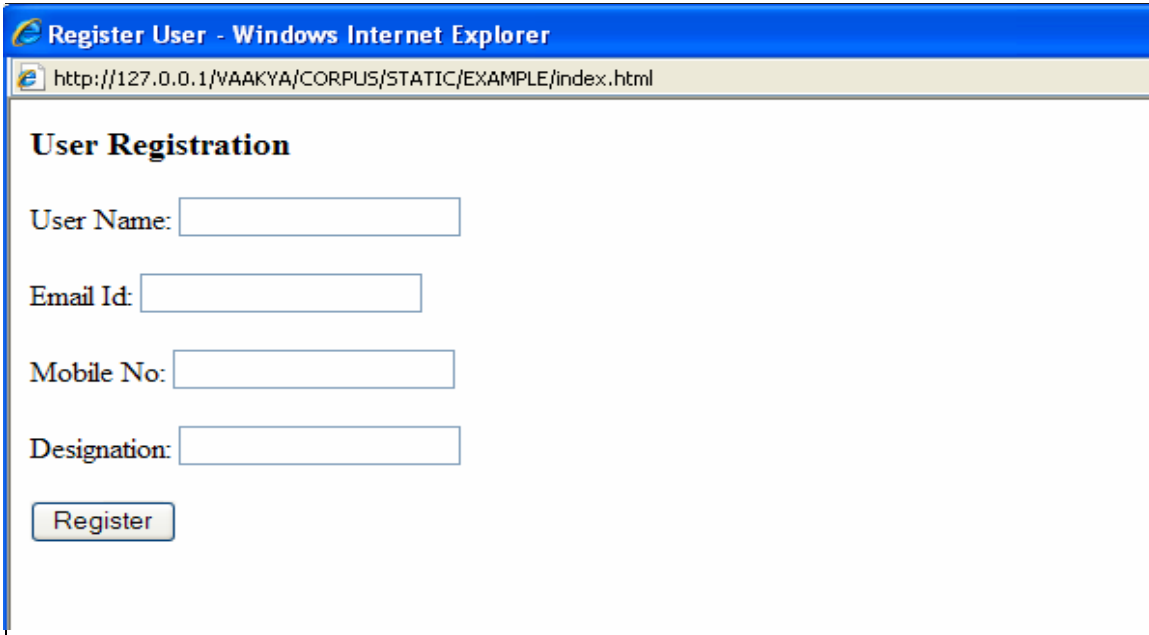
DevoSA Manager

DevoSA Manager is used to test & view projects

- Start DevoSA MANAGER by creating the following link on a browser
<http://127.0.0.1:9000/VAAKYA/DEVOSA/CORPUS/STATIC/IDEMANAGER/DEFAULT/index.html>
(check for port number, as assigned by you in Zone.cfg)
- DevoSA MANAGER will provide a list of applications.
- Click on the 'Start' along the URL of 'PORTALDEMO' application.
- Click on the application URL to view the application (note the application status changes to 'ON' when the project is executed)

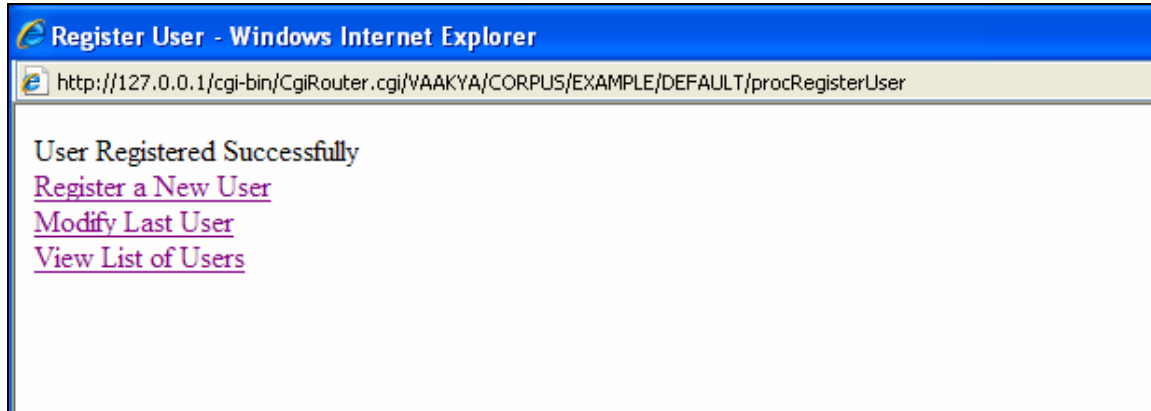
A Screenshot of the index.html page from the application is shown below

- Enter Data and Click on Register.



The screenshot shows a Windows Internet Explorer browser window titled "Register User - Windows Internet Explorer". The address bar displays the URL: <http://127.0.0.1/VAAKYA/CORPUS/STATIC/EXAMPLE/index.html>. The main content area of the browser displays a "User Registration" form with the following fields and a button:

- User Name:
- Email Id:
- Mobile No:
- Designation:
- Register



If you are unable to get the response page shown above, please check the following:

- Check if the IP and Port specified in Vaakya Zone Server configuration are same
- The Html (Form Action or href) service name and WebProc component name are case sensitive. Please verify the same in HTML page and IDE component

How it works?

When the button *"Register"* is clicked, HTTP request sent is

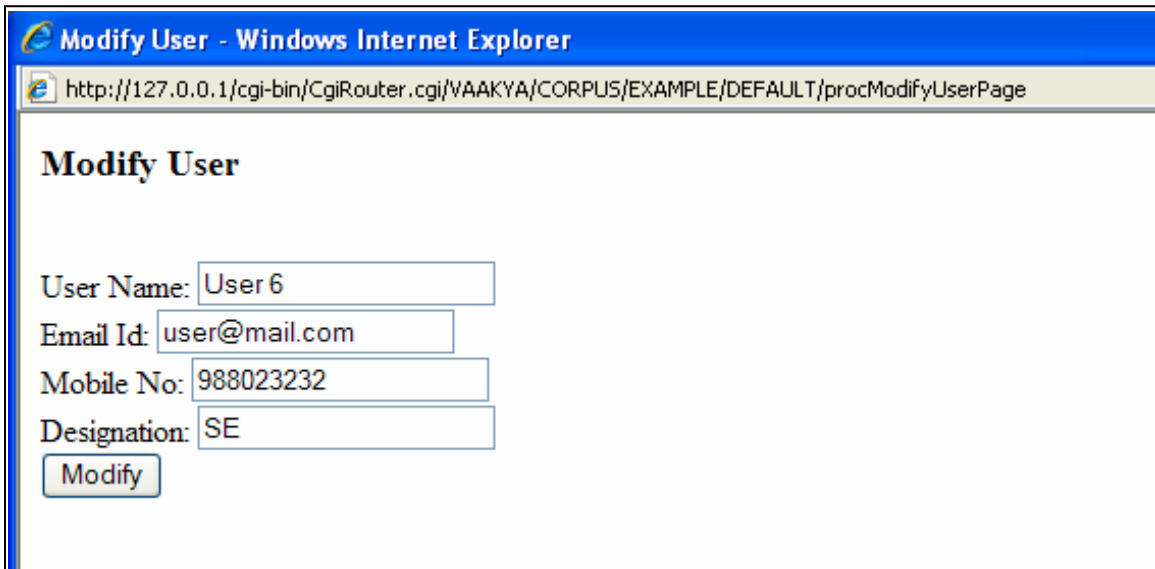
```
http://127.0.0.1 /cgi-bin/CgiRouter.cgi/VAAKYA/DEVOSA/CORPUS/PORTALDEMO/DEFAULT/vspRegisterUser
```

- 'http' is the protocol
- '127.0.0.1' is the loop back IP of your local machine. You may alternatively use 'localhost' instead of loop back IP address
- 'cgi-bin' indicates that the Web Server must handover the request to a CGI interface
- 'CgiRouter.cgi' is the CGI interface that accepts the request from Web Server (built-in within Vaakya Zone Server)
- Vaakya CGI interface identifies Vaakya Portal Server using the IP and Port specified in Web Server configuration file (CGI environment variable VCGI_SERVERADRS and VCGI_SERVERPORT) and transfers the request to Vaakya Portal Server
- Vaakya Portal Server locates the service name 'vspRegisterUser' (given in a Form action or in a hyper link) and executes the corresponding WebProc 'vspRegisterUser'

- The HTTP header and contents are added to the response stream in WebProc 'vspRegisterUser' using ComposerInit and ComposerSubstitute functions respectively. The contents in the response stream are sent back to browser (through CGI interface and Web Server) using ComposerSend function
- Data from HTML comes to the Portal Server in the form of key-value pairs
Ex: typCommon.UserName=Test& typCommon.UserEmail=test@mail.com
- Since Form data is defined for Type 'typCommon', the key-value pairs get mapped into 'prtypCommon'
- Now, the data in 'prtypCommon' is taken for manipulation
- Data from 'typCommon' is copied into 'UserDetails' Object variable and saved to data store using "VdsSave" function
- Since the Id field in "UserDetails" is not defined, a Record Id will be generated for a unique identification internally
- When all the fields are filled with data, a transaction-completed message 'User Registered Successfully' is sent back. Validations are not provided in this WebProc

Populating a Html template with data

Now, let us understand how to substitute Vaakya template variables with data from an Object and display it in an html page, as shown below.



The screenshot shows a web browser window titled "Modify User - Windows Internet Explorer". The address bar displays the URL: `http://127.0.0.1/cgi-bin/CgiRouter.cgi/VAAKYA/CORPUS/EXAMPLE/DEFAULT/procModifyUserPage`. The main content area displays a form titled "Modify User" with the following fields and values:

- User Name: User 6
- Email Id: user@mail.com
- Mobile No: 988023232
- Designation: SE

A "Modify" button is located below the Designation field.

Steps Involved

Create a WebProc

- Click on **WebProc** component '**vspModifyUserPage**' and follow these steps
- Select **Object** (radio button)
- In the Form Data input box, press F1 and select '**UserDetails**'
- In the alias column, enter '**objUser**'

vspModifyUserPage

??Variable declaration

Text @cpFooter

??Call proc to retrieve data from UserDetails object

proModifyUserPage(objUser)

??Merge and send the last record values into the template and send to the browser

ComposerInit("modifyuser", "HTTP/1.1 200 OK\nContent-Type: text/html\n\n", TRUE)

ComposerSubstituteRecord("modifyuser", prEnvData)

ComposerSubstituteRecord("modifyuser", objUser)

ComposerSend("modifyuser", cpFooter)

Create a Proc

Now, Click on Proc component '**proModifyUserPage**' and enter the following code

proModifyUserPage

??Variable declaration

```
Int  varRecPerPage,varCount, varClose  
Text @varExpression, @uidCursor
```

```
UserDetails  Recset @setUser  
UserDetails  objUser1
```

```
varCount      = 0  
varRecPerPage = 0  
varClose      = 0
```

??Load all records from the UserDetails table

```
setUser  = VdsGetTable("UserDetails", VDS_EXCBLOBS, varExpression, varRecPerPage,  
uidCursor)  
varClose = VdsFetchClose(uidCursor)  
varCount = RecsetCount(setUser)
```

??Get the last record from the set

```
objUser1 = VdsGetRecord("UserDetails", VDS_EXCBLOBS, varCount)  
objUser.ReclId      = objUser1.ReclId  
objUser.UserName    = objUser1.UserName  
objUser.UserEmail   = objUser1.UserEmail  
objUser.UserMobileNo = objUser1.UserMobileNo  
objUser.UserDesignation = objUser1.UserDesignation
```

Create a dynamic html Template

- Create a html template "modifyuser.html"

modifyuser.html

```
<html>
<head>
<title>Modify User</title>
</head>
<body>
<h3>Modify User</h3> <br />
<form method="post" action="vspModifyUser">

<label>User Name: </label>
<input name="UserDetails.UserName" type="text"
value="{{(-VAAKYA_VAR(UserDetails.UserName-))" maxlength="40" /><br />

<label>Email Id: </label>
<input name="UserDetails.UserEmail" type="text"
value="{{(-VAAKYA_VAR(UserDetails.UserEmail-))" maxlength="60" /><br />

<label>Mobile No: </label>
<input name="UserDetails.UserMobileNo" type="text"
value="{{(-VAAKYA_VAR(UserDetails.UserMobileNo-))" maxlength="16" /><br />

<label>Designation: </label>
<input name="UserDetails.UserDesignation" type="text"
value="{{(-VAAKYA_VAR(UserDetails.UserDesignation-))" maxlength="30" />
<br />

<input name="UserDetails.RecId" type="hidden"
value="{{(-VAAKYA_VAR(UserDetails.RecId-))" />
<input type="submit" value="Modify" name="submit">
</form>
</body>
</html>
```

Note:

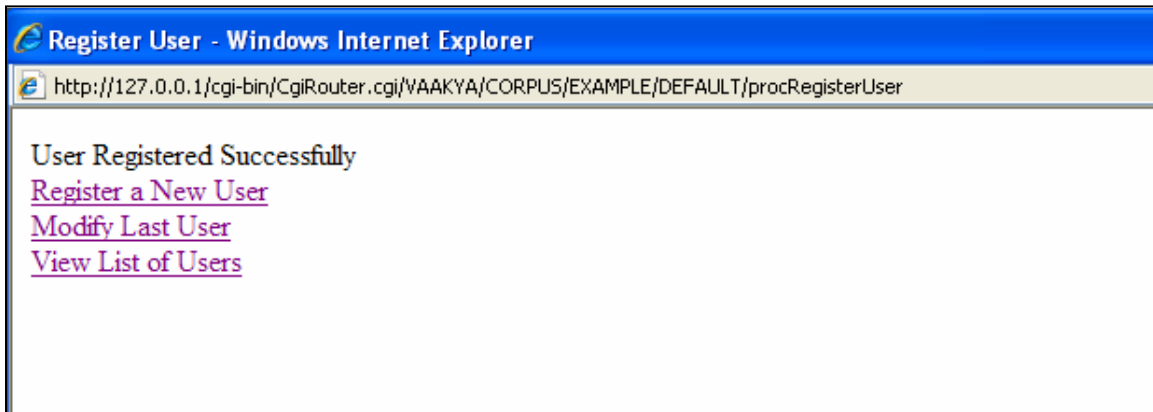
Vaakya template variable name provided here should be same as the field name in "UserDetails". In this example, UserDetails is directly used to display the values. Object or Type can be used according to the requirement.

Vaakya template variables are identified by `{{(-VAAKYA_VAR(VariableName-))}}`. These variables are substituted at runtime.

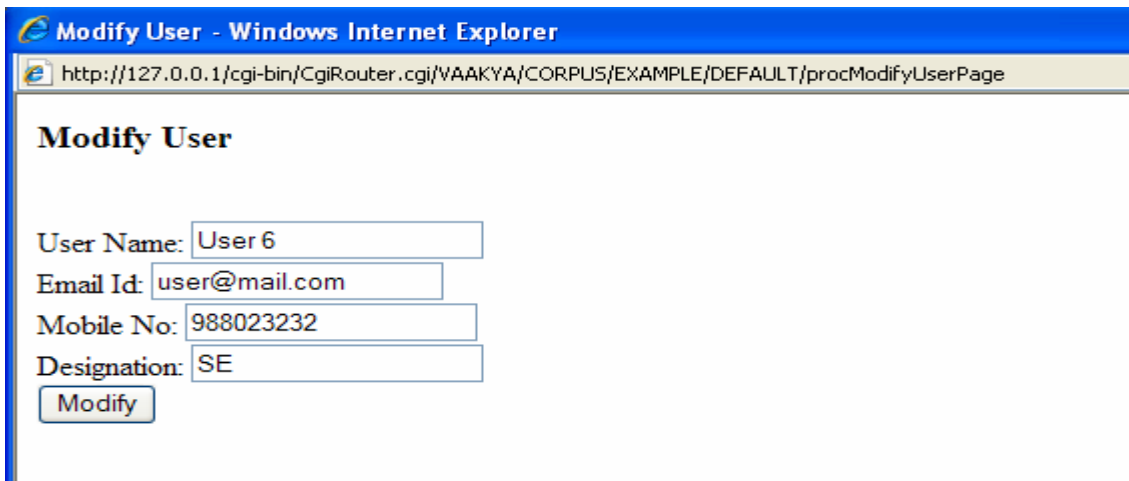
ComposerInit function copies the static contents of html template into response stream. ComposerSubstitute function substitutes the contents of Vaakya variables with the values in "UserDetails". Substitution of value is done only if field name in Object/Type and name of Vaakya template variable are same.

Web Application Development Tutorial

- Paste this file into
\\VAAKYA\ZONESERVER\DEVOSA\CORPUS\IDES\TEMPLATES\PORTALDEMO
- Add this html page to Templates by activating
Components→ Template→ New and add the template name as 'modifyuser'.
- Click on the newly created template 'modifyuser'
- Select the html file and save.
- Repeat the steps above mentioned in **Create Executable** to test
- If the application is in 'ON' status(Green Icon), Stop it and Start again
- Access index.html from browser and Register a new User
- Click on the link '**Modify Last User**'



- You will see the page as shown below



Note:

A hyperlink to access this service is already provided in 'home.html'

In the previous template, the service is already mentioned as 'vspModifyUser'.

Click on WebProc component '**vspModifyUser**' and follow these steps

- Select **Object** (radio button)
- In the FormData input box, press F1 and select '**UserDetails**'
- In the alias column, enter '**objUser**'

vspModifyUser

```
Text @cpFooter
typCommon prTypCommon
prTypCommon = New(typCommon, 1)
```

??Calling proc to Modify User

```
proModifyUser(objUser, prTypCommon)
```

??Merge and send the last record values in to the template and send to the browser

```
ComposerInit("home", "HTTP/1.1 200 OK\nContent-Type: text/html\n\n", TRUE)
ComposerSubstituteRecord("home", prEnvData)
ComposerSubstituteRecord("home", prTypCommon)
ComposerSend("home", cpFooter)
```

Explanation

Modify Proc 'proModifyUser' to modify the details of the last registered user

Click on Proc component '**proModifyUser**' and enter the following code

proModifyUser

```
Int varRet
Array @varArray
UserDetails objUser1

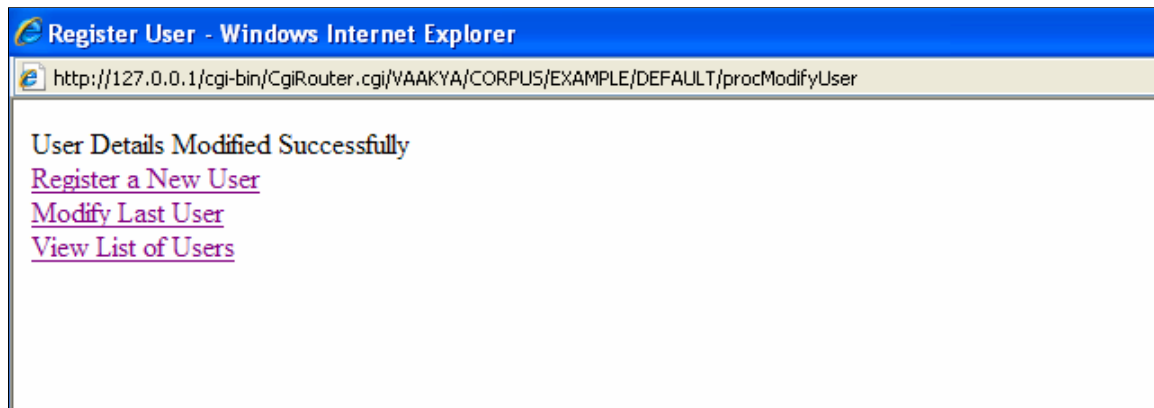
varArray = New(Array, 1)
varRet = 0
??Get the last user record using Record Id
objUser1 = VdsGetRecord("UserDetails", VDS_EXCBLOBS, objUser.ReclId)
ArrayAdd(varArray, objUser)

??Modify the selected record

varRet = VdsModify(varArray, VDS_RECORD_TRAN)
If(varRet)
```

```
{  
  prTypCommon.Message = "User Modified Successfully"  
}  
  
Else  
{  
  prTypCommon.Message = "User Modification Failed"  
}
```

- Save Proc
- Execute the application as explained earlier
- Register a new User
- Click on the link 'Modify Last User'
- Modify the User Details and Click on Button "Modify". The following message will be displayed.



Having understood the concept of html templates and Vaakya template variables, let us see how to view a list of records using Vaakya Block in the next session.

Vaakya Block Variables and HTML Template

In this session, let us extend the previous example to display a list of users using Vaakya Block variables.

Steps Involved

- Create a html page “viewusers”

viewusers.html

```
<html>
<head>
<title>List of Users</title>
</head>
<body>
<h3>List of Users</h3> <br />
<table>
<th>User Name</th>
<th>User Email</th>
<th>User MobileNo</th>
<th>User Designation</th>

((-VAAKYA_BLOCK(LISTOFUSERS-))

<tr>
<td>((-VAAKYA_VAR(typCommon.UserName)-))</td>
<td>((-VAAKYA_VAR(typCommon.UserEmail)-))</td>
<td>((-VAAKYA_VAR(typCommon.UserMobileNo)-))</td>
<td>((-VAAKYA_VAR(typCommon.UserDesignation)-))</td>
</tr>

((-VAAKYA_BLOCKEND-))

</table>
</body>
</html>
```

Contents specified within a Vaakya Block are substituted repeatedly

- Vaakya Block starts with `((-VAAKYA_BLOCK(BlockName)-))`
- Vaakya Block ends with `((-VAAKYA_BLOCKEND-))`

BlockName is just a logical name and must be unique within a single html template file.

Note:

A hyperlink to access this Webproc is already provided in “home.html”

Click on WebProc component '**vspViewUsers**' and follow these steps

- Select **Type** (radio button)
- In the FormData input box, press F1 and select the '**typCommon**'
- In the alias column, enter '**prTypCommon**'
- Enter the following code and click on **SAVE**.

vspViewUsers

??Variable declaration

```
Int      recPerPage, varCount, varClose
Text     @cpFooter, @varExpression, @uidCursor
```

```
UserDetails  Recset @setObjUser
UserDetails  objUser1
varCount     = 0
recPerPage   = 0
varClose     = 0
```

??Initialize template

```
ComposerInit("viewusers", "HTTP/1.1 200 OK \nContent-Type: text/html\n\n", TRUE)
```

??Get all the values in to a record set

```
setObjUser = VdsGetTable("UserDetails", VDS_EXCBLOBS, varExpression,
recPerPage, uidCursor)
varClose   = VdsFetchClose(uidCursor)
varCount   = RecsetCount(setObjUser)
```

??If records available loop record set and merge the values into the template ?? using Vaakya block.

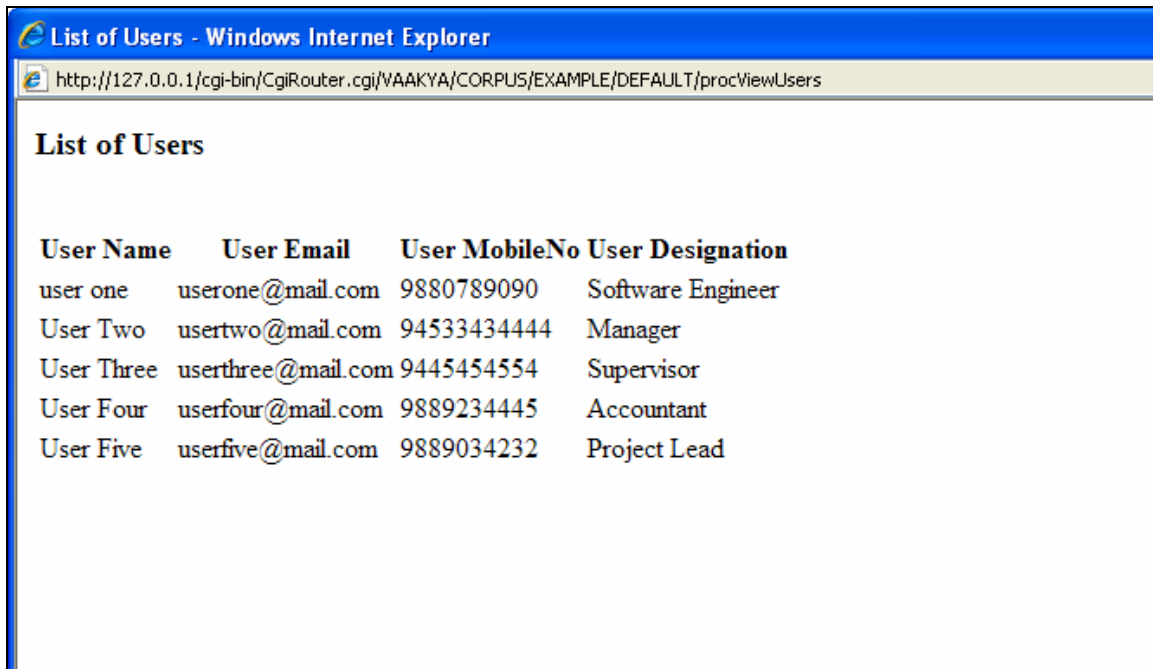
```
If(varCount)
{
  Loop(setObjUser,objUser1)
  {
    prTypCommon.UserName       = objUser1.UserName
    prTypCommon.UserEmail     = objUser1.UserEmail
    prTypCommon.UserMobileNo  = objUser1.UserMobileNo
    prTypCommon.UserDesignation = objUser1.UserDesignation
    ComposerSubstituteBlock("viewusers", "LISTOFUSERS", prTypCommon)
  }
  ComposerSubstituteRecord("viewusers", prEnvData)
  ComposerSubstituteRecord("viewusers", prTypCommon)
  ComposerSend("viewusers", cpFooter)
}
Else
{
```

??If records are not available display proper message

```
ComposerInit("home", "HTTP/1.1 200 OK \nContent-Type: text/html\n\n", TRUE)
prTypCommon.Message = "No Record Found. Please Register"
ComposerSubstituteRecord("home", prEnvData)
ComposerSubstituteRecord("home", prTypCommon)
ComposerSend("home", cpFooter)
}
```

-

- Data from Object is copied into Type variable
- Contents of each record are merged into template using *ComposerSubstituteBlock*
- Save WebProc
- Execute the application as explained earlier
- Click on 'ViewUser' to view the page below



User Name	User Email	User MobileNo	User Designation
user one	userone@mail.com	9880789090	Software Engineer
User Two	usertwo@mail.com	94533434444	Manager
User Three	userthree@mail.com	9445454554	Supervisor
User Four	userfour@mail.com	9889234445	Accountant
User Five	userfive@mail.com	9889034232	Project Lead

Note:

- *ComposerSubstituteBlock* function cannot be called before calling a *ComposerInit*
- Nesting of blocks are not allowed
- You may have any number of blocks within a template. Ensure that block names are unique within a template.

Summary

We have completed and covered the basics of Vaakya portal application development:

- Collecting data from html page.
- Storing and retrieving data from Object including basic validations
- Using html template with Vaakya template variables
- Using html templates with Vaakya block

Contact Us

Corporate Office:

601, First Floor,
12th Main, HAL 2nd Stage,
Indiranagar, Bangalore
PIN - 560008
India

support@vaakya.com

Copyright

"The copyright of any and all material contained herein is owned and reserved by VAAKYA TECHNOLOGIES PRIVATE LIMITED ("**OWNER**") except where otherwise stated. Any reproduction, copying and/or distribution in any form of the material, in whole or in part, are not permitted without prior written consent from the OWNER. Trademarks, logos, images, text or content of third parties ("**THIRD PARTY PROPERTY**") used herein are the property of their respective owners, and have been used without permission except where otherwise indicated. The Owner makes no claim to such THIRD PARTY PROPERTY used herein. All use of THIRD PARTY PROPERTY contained herein is for non-profit purposes only, for the proper guidance of the users, considering the convenience and familiarity of the user in associating the THIRD PARTY PROPERTY with the owners of THIRD PARTY PROPERTY and not for advertisement or establishing any connection with the owners of THIRD PARTY PROPERTY. Other than THIRD PARTY PROPERTY, all trademarks, trade names, logos, designs and all related product and service names are the sole property of the OWNER, and may not be used in any manner without the prior written consent of the OWNER. The materials contained herein may include inaccuracies or typographical errors and the owner shall not be held responsible for the same. The Owner reserves the sole right to modify, amend, delete, omit, edit or include any material or make periodic changes to the material herein.