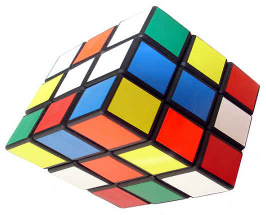
Cut to the Chase Series
*More Walk – Less Talk*

# zenyan webapp framework



# UI Templates
*Implementing and Using*

# zenyan webapp framework – Implementing/Using UI Templates

## Table of Contents

# Implementing and Using UI Templates

UI stands for "User Interface." In plain language and context to zenyan, it is what you view in the browser. The UI can be static or dynamically promulgated at the browser or from the server. But irrespective of how or where the UI gets created it is still the UI!

## Please Readme First

The philosophy behind these templates is really quite simple. You are building an application and there are aspects of the application that are required on each page (screen). There is also context you wish to carry that too is global to your application as a whole.

These particular templates are static. If you have a good sense of the application you are building then they will serve you well. Of course, if you have used a template thirty or forty times in your application and need to make a global change... Well, this means you now have thirty or forty files you need to update. But, this is why we have regular expressions and editors with global search and replace :-).

The templates are designed to be as generic as possible. They need to be tweaked by you to suit you individual needs. My hope is that you will latch on to this as a design principle (one of many zenyan offers up) as use it where it makes sense in your application.

The essence of the UI templates are to offer up a static (**already functioning and tested**) piece of code that reflects common UI patterns used in your application.

## The List (thus far)

You can find theses two templates under the zenyan root directory in the directory named "code_examples." These files will begin with the file name prefix "template_." I would encourage you to continue to use this naming standard, but that is entirely up to you.

| Template | Description |
|---|---|
| template_base.php | General code template containing general application context, reference to frequently used zenyan api's, and general external zenyan configuration parameters. This template also includes your top nested menus. |

| | |
|---|---|
| template_tabs.php | This is a direct derivation of the template_base.php base template. It includes three tabs for the page which can easily be modified to suit your needs. |

These two templates are presented here as the main landing page (mainpg.php) is an inherent example of a base screen with menus and tabs.

Under “Templates” in the main menu you will find a series of other templates. The following is a list.

| Template | Description |
|---|---|
| TEMPLATE_2ColDataDisp.php<br>TEMPLATE_3ColDataDisp.php<br>TEMPLATE_4ColDataDisp.php | Simple templates for read only data display. |
| TEMPLATE_3ColForm.php | Simple template for a form with 3 columns. |
| TEMPLATE_Handler.php | Not a UI template. This is an intermediate data handler meant to be used for complex workflows that involve multiple pages. |
| TEMPLATE-asmt_1col_radio.php<br>TEMPLATE-asmt_2col_radio.php<br>TEMPLATE-asmt_3col_radio.php | Simple template presenting a series of radio buttons. Can be used for creating test questions or assessments. |

## Common Template Code Explained

I must assume you have an understanding of general Php programming, though some of this may certainly be new to you.

### Session Context

The following code is required in order to maintain user/session context within your application.

```
session_start();

import_request_variables('gp', "formval_");
```

This is the standard Php code you need in order to maintain context between pages within your application. In other words, your end user transitions from page foo.php to bar.php and code in bar.php needs to refer to something that was done back via the foo.php page. These two statements allow you to use and share session variables and to maintain session context between your pages. The fact that Php sports inherent session context makes it very cool indeed!

## Main Environment Reference for zenyan

```
require('eConfig/envref.php');
```

This directive provides the path to the zenyan environment. The zenyan path references will not be discussed here but are discussed in the document "zenyan_general_configuration." The point is to abstract physical path references as zenyan can work in Windows and in Posix (Linux, Unix, Aix, etc...). This is where you specify things like drive designators, and paths to your application, as well as paths to the zenyan infrastructure. This is a core file you need to tweak in order to properly implement zenyan. This configuration files assumes that you are using a pc and that you are installing it to you c drive. It also assumes you have also installed all the low level infrastructure required by the zenyan framework (Apache, Php, MySQL, etc...).

The envref.php file is where you register (code) various path references for things you intend to implement server-side within the zenyan framework.

The following are some typical path references.

```
include($php_envvars);

include($php_dbiox);  //dbms specific to app

include($php_applib);

include($php_loggers);
```

Since the above are Php variables indirected from envref.php further detailed explanation of each is  in order (at a fairly high level here as they are describe in detail elsewhere).

## Environment Variables ($php_envvars)

This reference is to relative path <root>/eLib/envvars.php. This contains important configuration information like the webserver url/domain name, the global state of the application logging subsystem, the zenyan database, etc... .

## DBMS Access API ($php_dbiox)

This is the data access api specific for each dbms. It is relatively easy to introduce a new access api into the mix. While I have created api's for proprietary systems like H-P NonStop Enscribe and H-P NonStop SQL-MP (these are not open source), and for remote MySQL connectivity the api's available in zenyan (at present) are included in the table below:

| DBMS API Ref | Relative Path/File | Description |
|---|---|---|
| $php_dbiom | eLib/o_dbiomyp.php | MySQL local api. Non-ODBC access. Used for direct Php access to the implementation of MySQL within the zenyan webserver framework. |
| $php_dbiox | eLib/o_dbiox.php | General ODBC access. |

Current general function points include:

| Buffer Type | Data Returned As | Description |
|---|---|---|
| Delimited | Array | (default) data only |
| Delimited | Array | Data, first element = column name |
| Delimited | String | Delimited, newline as row separator |
| 2D Array | Array | Two dimensional array |
| HTML Stub | Array | Used in conjunction with UI Grid Control |
| Hash | Hash (Assoc Array) | Key/Value pair used for singleton data (entry forms, etc) |
| Raw | String | Returns data as a text blob. |

It is also relatively easy to add new function points into the api to meets your specific data. For example, soap or xml.

## Application Code Library ($php_applib)

These are generalized application code functions. If it is something that is generic for consumption by an application it is a candidate to go here. Here is, "eLib/applib.php." What is not included here are generic JQuery UI components. Here is what is included in the current implementation of this library.

- ✓ Generate GUID
- ✓ Retrieve Synchronized List
- ✓ Change delimiter

It is important to understand that this is server-side functionality, though portions of the api do dynamically build browser-side code on-demand at runtime.

## Logging Subsystem ($php_logging)

The logging subsystem (loggers.php) consists of two functional sections: 1) application domain logging; and 2) development logging. Both logging mechanisms write to text files (similar to Apache and Php logging activities). It is possible and easy to extend (and I have in commercial implementations) the logging mechanism to write to a structured DBMS table(s). My general theology is to write to text files which are easier to manage, and to parse using regular expressions.

Unlike the Apache logging mechanism which is an all or nothing affair, the zenyan loggers give 100% control to the developers and implementors of the zenyan framework. In other words, the application logger can log every single event that occurs both through the UI and on the back end. Logging can literally be a “big brother is watching every move made” affair. The caveat is that the developer must implement this logging within their code. Logging anything comes down to adding a few lines of code to your program, and a single function call.

Development logging is a means of debugging and prototyping your code during the development cycle. It can also be used to troubleshoot runtime problems, and can come in handy in tracing intermittent problems. This mechanism can log to a file, but it can also be used to log to stdout (which in a webapp is the browser). The following code is intended for this use.

```
<!-- comment out or remove me for production -->

<?php echo $_SESSION['oldebug']; ?>
```

The concept is pretty simple. Concatenate data you wish to echo in your application development work to the session variable above (I liberally use <br /> for better readability). I find this a good way to hunt down any logic errors I may have coded. I also find this enormously useful for prototyping new functionality. It is a ridiculously simple, but powerful means for doing development.

One of the Achilles heals of Php is the lack of an inherent debugger in the language. There are a number of open source and commercial debuggers available. Having never been someone overly reliant on using debuggers I typically use the logging subsystem to debug my code. I have just found it too much of a chore to implement a debugger formally into the framework. Having said this it would be really cool if someone would take this on, do it, and maintain it. I just have not had the time.

The logging subsystem is covered in more detail in another document.

## User Validation / URL Validation

Most applications are going to employ some level of login security. The zenyan framework comes with a basic login module that is tied to the application context management framework. This is covered in more detail under another document. But for now, the following code will provider you with an inkling of how it works.

```
$fullrefpg = $_SERVER['HTTP_REFERER'];


//get only page name.

if (preg_match("/[A-Za-z0-9_-]+[.]php.*$/", $fullrefpg, $matches)) {$refpg =
$matches[0];}

$refpg = preg_replace("/^([A-Za-z0-9_-]+[.]php)(.*)$/", "$1", $refpg);


//// commented out during development ////

check_referring_pg($refpg);

//// commented out during development ////


function check_referring_pg($refpg){
 //if referring page context is required you code conditional here as wrapper to
below conditional
 if ( $_SESSION['initentry'] == $Scontxttoll)
  {
   //in-context... presuming me
  } else { header("Location: login.php"); }
}
```

At its most rudimentary core the logger checks to see if the user has logged in. If they have not they are taken to the login page. In this simple depiction of the baseline template once the user successfully clears the login process the session variable $_SESSION['logtoken'] is set to the value set in environmental variable $Scontxttoll in eLib/envvars.php. This number is always checked against each page being accessed during session context.

As you can also see in the code (and it is not implemented in the baseline template) it is also possible for the page being accessed to check the referring page as a further level of granularity.

I want to take a short but relevant sidebar right now. Php is not a compiled language. It is also the most popular language today in internet web development. The internet is a wide open space with many evil people lurking about. If you want to be safe and secure in developing internet applications your need to essentially do two things:

1. Implement and maintain a secure web server environment and secure file system
2. Make sure hackers do not have access to your Php code on the server or elsewhere.

Hence, we really have a dilemma here. If I or you open source our code then everyone, even the hackers can see it. Being able to see the code allows someone to much easier figure out how to exploit it.

The freedom of zenyan allows you to add or subtract from any of these general code items for you own specific implementation of the framework. Or, you can just take everything here "as-is."

## A Quick Commercial Regarding JQuery

It was really easy to get my head around JQuery My hat is off to John Resig for inventing this incredible library. His work and efforts really embody the open source market.

## Regarding the JQuery implementation in zenyan

The JQuery implementation lives under the root directory in a directory named "jquery." The main jquery library is currently jquery.min132.js where 132 represents version 1.3.2. I took the liberty of renaming the files as

1. I am a purist and believe the "." should only be reserved to delineate the file name from the extension.

2. In renaming the file, I make it unique to zenyan to minimize the possibility of

another bogus file with that name being introduced into the framework in a way that could break something.

3. I wanted to maintain overt version control (beyond source management) within a commonly shared infrastructure.

You will also find other plugins here as well.

## Criteria for inclusion into the zenyan framework

First, you need to understand that zenyan is a full web development  application framework. All of the available UI components I am using require JQuery. But, many are using different versions of JQuery. It is impractical to implement a web application that requires maintaining references to different versions of the same api!!! I cannot be more emphatic about this. Also, there is a huge difference between an on-going developmental environment, a development environment directly supporting production customers, and a production environment. This is not a bag on anyone or anything this is merely a cold-hard reality that makes this business so compelling.

Components I evaluate for inclusion into zenyan must all work with the same version of jquery. And, they must work with the minified version (which should not ever be an issue, but I never take anything for granted.)

I may be sounding here like I am switching from open to restrictive mode. I am not, but I do not think I have adequately made my case so I am going to elaborate. This common jquery domain is designed and intended for use with common UI components that one might use within an application page or workflow. For example, on your page you may wish to place a date lookup control, a set of tabs, a nested menu, and a treeview control. These are some of the typical UI controls one uses in developing applications. Well, my criteria is that all of these types of controls that share common code, actually need to share common code!

As someone that has lived through “DLL hell”, “Jar hell” and other hells where the solution is to maintain different versions of common code bases in order to support the functionality I realized that there was really a need for tighter control in this area. Luckily I have yet to have a problem implementing JQuery plugins that use the same commonly shared code base.

And, if you desire to reference more than one version of common api's in your implementation you are free to do so. This is just something I wish to avoid at all cost.

## Criteria for upgrading JQuery version in the Common Library

I have never been one to upgrade just for the sake of upgrading. I require a valid business case for upgrading that may include...

- ◆ Support for the version we are on is coming to an end.
- ◆ There is a cool new widget we really want that is not compatible with existing the infrastructure.

Whatever the reason, upgrading is never a trivial event as it requires re-testing the existing infrastructure against the implementation.

### Modules within zenyan

Now having said that, in zenyan there is also the notion of higher order application components. For example, I use a specific implementation of the tree viewer control in order to implement a “documentation management subsystem” widget. In a case like this it becomes my design goal and architectural criteria for this code to stand on its own two feet and not be required to share code from a common base. My rationale is that such a component is useful on its own and I do want to require a full implementation of zenyan in order to implement and use it. In other words, components like this can run under any LAMP/WAMP implementation. The other important criteria I have for making this type of decision is that the widget must be able to stand entirely on its own.

Within zenyan I refer to these components as modules. The document management subsystem is a standalone zenyan module. In fact, as I said before, it does not even require the full zenyan framework for you to implement. In this case, since the module is essentially its own encapsulated module it includes its own version of JQuery and of the file tree plugin that are separate from the common JQuery library.

### General Information Regarding the CSS for a Control

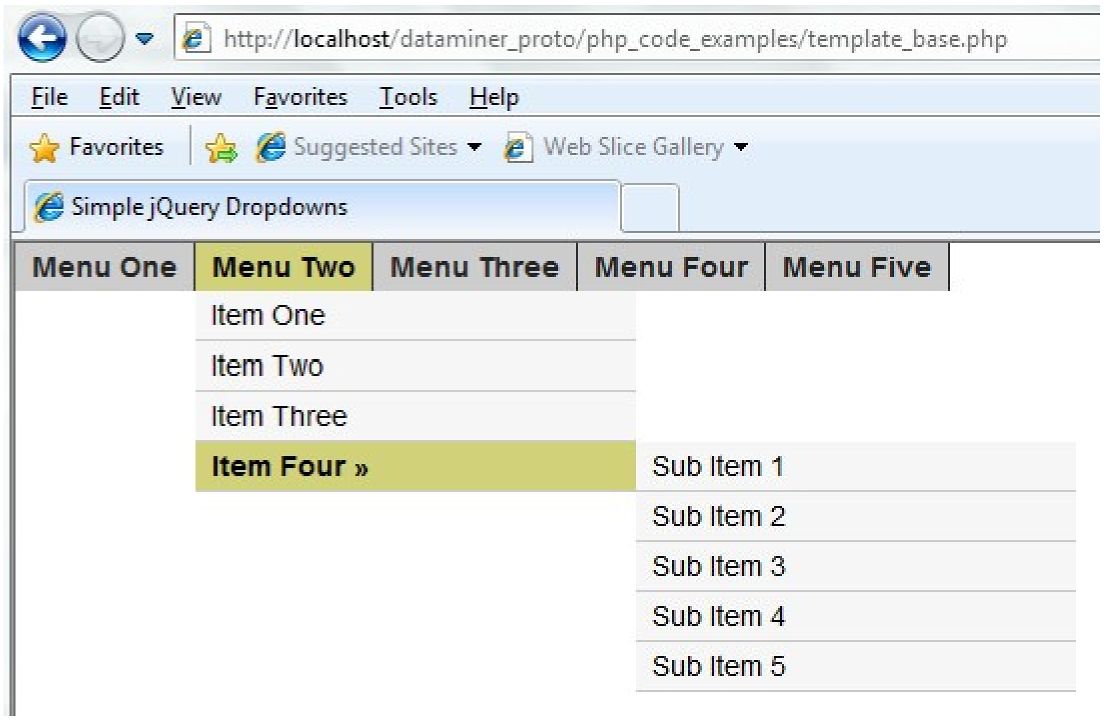
**IMPORTANT NOTE:** DO NOT add your own common HTML tags to these stylesheets! Content styling is purposely separated from UI control styling within zenyan. DO NOT replace the css files with the originals or modify path to the originals as these CSS files have been modified and tweaked to work within the zenyan framework. The postfix “_a” in the css filename indicates that the css was modified specifically to conform to zenyan rules for css implementation.

**IMPORTANT NOTE:** I added z-index setting to the css so that this control would play well with the tabs control.

Stylesheets specific to a JQuery UI control reside under the “jquery” directory in a directory named “stylesheet.” Further, any images/graphics associated with a control live under the “stylesheet” directory in a directory named “images.”

Finally, there are situations where certain controls have CSS that can be tweaked outside the common library. In these circumstances where a separate CSS is needed it is kept in the same directory location as the Php code that consumes it.

## Base Template (template_base.php)



The presumption is that you want nested menus. The other presumption is that like a typical application these menus need to appear on every page. If these are true read on and I shall show you what you need to do to get this template in a working state.

## Tweaking the Content

Under the "jquery" directory you will find a file named topmenu.php. This file contains the example menus.

This common code file includes the common menus for your application. The whole point of doing business this way is that when you add new menu functionality to your application, you only need to make the change once!

You need to modify the menu code for your specific implementation of the menus.

## Tweaking the CSS

The menu css can be found under jquery/stylesheet/. The file is named "nestedmenu.css."

You can set certain menu configuration parameters in jquery.dropdown.js. Shown are the default settings

```
sensitivity: 3, // number = sensitivity threshold (must be 1 or higher)
interval: 200,  // number = milliseconds for onMouseOver polling interval
timeout: 200,   // number = milliseconds delay before onMouseOut
```

### The Code

This information is really needed in order to utilize the control. So, unless you are interested in the specific references in the template required to implement the control you can stop reading now.

Code under <head> specific to the menu

```
   <link rel="stylesheet" href="../jquery/stylesheet/nestedmenu_a.css"
type="text/css" media="screen, projection"/>

   <!--[if lte IE 7]>

        <link rel="stylesheet" type="text/css"
href="../jquery/stylesheet/nestedmenuie.css" media="screen" />

    <![endif]-->
```

```
   <script type="text/javascript" language="javascript"
src="../jquery/jquery.dropdownPlain.js"></script>
```

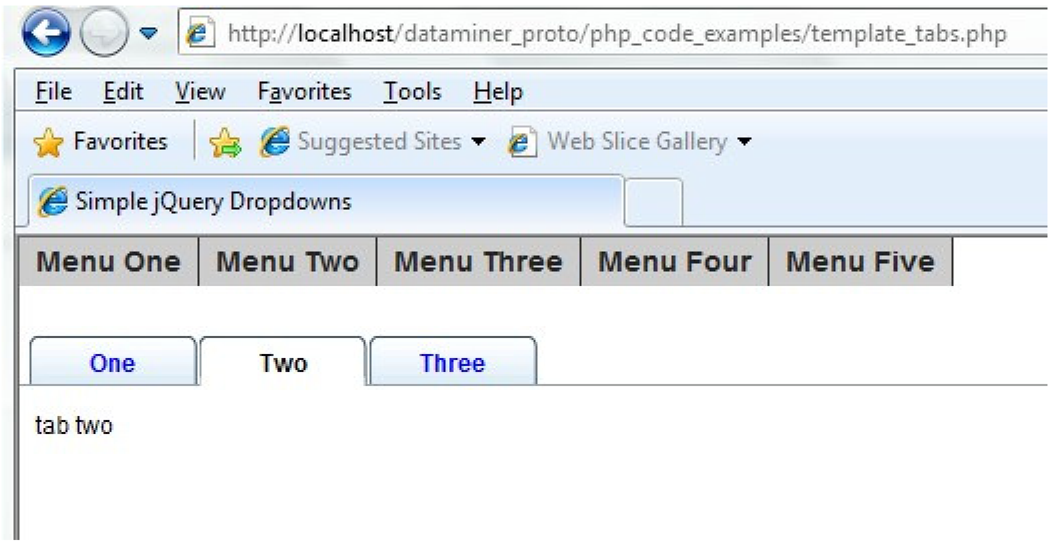
Code under <body> specific to the menu

```
<?php

  include('../jquery/topmenu.php');

?>
```

## Tabs Template (template_tabs.php)



Tabs within a page can be a great means of segregating content that belongs together but needs to be differentiated for ease of navigation. For instance, you might want to implement webpages that demonstrate code examples that others can see and use. You could have three tabs: 1) Code

Demo; 2) The Code; 3) Code Description. Another example might be to return demographic information for a patient. One tab could represent their contact information; another tab could represent their health insurance; another their primary providers and their contact information.

## Tweaking the Content

Since tabs are unique to a specific page they are coded in-line for that page.

### Code related to tab labels

```
<div id=etabs>
<ul>
  <li><a
  href="#tab-1"><span>One</span></a>
  <li><a
  href="#tab-2"><span>Two</span></a>
  <li><a
  href="#tab-3"><span>Three</span></a>
</ul>
```

### Code pertaining to content within the tab

```
<div id="tab-1">
<p>tab one</p>
</div>

<div id="tab-2">
<p>tab two</p>
</div>

<div id="tab-3">
<p>tab three</p>
</div>
```

## Tweaking the CSS

ui.tabs.css

## The Code

This information is really needed in order to utilize the control. So, unless you are interested in the specific references in the template required to implement the control you can stop reading now.

Code under <head> specific to the menu

```
<link media="print, projection, screen" href="../jquery/stylesheet/ui.tabs.css"
type=text/css rel=stylesheet>

<script src="../jquery/ui.core16rc5.js" type=text/javascript></script>

<script src="../jquery/ui.tabs16rc5.js" type=text/javascript></script>


<script type=text/javascript>

            $(function() {

                $('#etabs> ul').tabs({ fx: { opacity: 'toggle' } });

            });

</script>
```

Code under <body> specific to the menu

Already described in the previous section.

# Creating Your Own UI Templates the zenyan Way

## What is the zenyan way?

### What is a UI Template?

A UI template is a piece of working code that represents a design pattern for a specific web page that has the potential of being reused. The general theology is to create design patterns that promote reuse and offer consistency in visual design through your user interface.

### Guidelines

- ✓ The template needs to be working code that someone can execute and view. Why would anyone want to use your stuff if it does not work to begin with?
- ✓ A template should be for a UI pattern that has a high probability or reuse.
- ✓ A template should be well documented within the code so that the person using it does not need to figure out how to reuse it.
- ✓ A template should be added to the "View_UI_Templates.php" page in an applicable category so that the user can see what is available. How do you expect someone to reuse stuff and follow your design patterns if you make it difficult to determine what the heck they are?
- ✓ If you offer up your template to a general development community at large you always do so with the understanding that the template may very well be tweaked to fit the consumers needs. Whatever strict design controls you wish to implement in your own domain and workplace is your business. In the open domain DO NOT think you can impose your will or rules on others. This even extends to these guidelines.
- ✓ Consumers must always respect the initial template designers code and leave them as the originator of the template. Credit where credit is due. This is of course in accordance with the license agreements zenyan uses.
- ✓ Put your templates under the zenyan directory "php_code_examples" so as to provide a one-stop shopping location for these zenyan artifacts.

### How do I know that what I am creating should really be defined as a template?

Is your effort something that represents a design pattern for a single web page?

Does (pretty much) all the consumer have to do is make a copy, rename the page and add their content?

Is the design pattern something that can be reused without having to re-tweak the design?

Is their a likelihood that this design pattern has some value for reuse?

If you can answer yes to these four questions then you have a good candidate for a UI pattern others may wish to implement and reuse.

## Answers To Commonly Asked Questions?

*If I make a UI template for an application where every page uses menus but I do not include the menus would this be considered a UI Template?*

No. It would be considered toycode. Classify it as toycode. In order to classify as a UI template your work must contain everything that is common and required to every page within the application or within an area within an application. This would include things like menus, toolbars, standard headers and footers.

*Must I offer total freedom in design to anyone?*

**NO!** Freedom also means the freedom to be restrictive within your own domain. For example, you are the design authority responsible for creating an application within your place of work. It is your right and responsibility to impose whatever design control you deem necessary to get the job done in a way that meets your standards and requirements.

What I have already stated is that if you wish to share your work with the proverbial world you do so within the confines of the licensing agreement which pretty much states that beyond identifying you are the originator of the work they are pretty much free to to do anything to your work, including totally screwing up (by your definition) your efforts. Since they are not allowed to modify the original work effort there will always be a trail back to the original to create a trail of accountability.