
AVOIR -African virtual Initiative Resources http://avoir.uwc.ac.za
CHISIMBA
PRODUCT OF
AVOIR
African Virtual Open Initiatives and Resources

Chisimba Developer

- Assumes you have working knowledge of
- You have at least a basic notion of object oriented programming

What is Chisimba ?

- Chisimba is a Web 2.0 enabled rapid application development framework for creating web applications that are platform independent, browser independent, xHTML compliant, and can use a number of common databases.

What is Chisimba cnt'd

- Chisimba is written in PHP5 using the model-view-controller paradigm, implemented via a modular architecture.
- Over 100 modules of functionality are already available and these can be used out of the box to create a Content Management System, a feature-rich e-learning platform, a group-based collaboration platform, a blogging system that allows posting from mobile phones, and many other applications.

Simple setup and configuration

- Chisimba modules are configured at install time from a simple text-based configuration file that contains common setting, and automates permissions and menu entries.
- Most settings are configurable from a dynamic configuration editor, including menu entries, permissions, and even the type of site.
- The Chisimba module catalogue includes some common configurations, allowing you to install an e-learning platform, a CMS, an organizational portal, or any of several other configurations from the same codebase.
- Modules can also be installed individually or combined in different ways to create entirely new application types without any programming.

Simple permission management

- Chisimba uses a group/role based access control system that is underpinned by and ACL system that allows permissions to be as fine or as corse grained as you like.
- Groups and roles can be edited in a GUI editor, and can be changed on a per module basis if necessary. Developers can use this to assign module, page, or within-page permissions.

internalization

- Chisimba allows full internationalisation and localization using commonly available translation facilities, thus allowing interface elements to be presented in different languages without having to modify any core code.

The name

- Chisimba is the Chichewa (Malawi) word for the framework used to build a traditional African house.

Chisimba as a MVC framework

- Chisimba is the second version of a framework that was known previously as KINKY (Kinky Is Not KEWL Yet). KINKY was mainly designed for e-learning applications, but which quickly evolved as a general framework that could be used of build any kind of web application from simple content management to enterprise functionality.

Model-view-controller (MVC) architectural pattern

- Chisimba is implemented using a model-view-controller (MVC) architectural pattern, with a front controller web implementation, and has a fully modular architecture in which almost everything is a module.
- The framework includes abstractions for common programming tasks such as rendering HTML, sending and receiving SMS, sending and receiving email, creating and consuming webservices, interface localization, interface generation, etc.

The model

- The model is the domain-specific representation of the information on which the application operates. This is typically stored in a database, but can also be represented in other forms, including files in the filesystem, data in XML, etc.
- In Chisimba, the model component typically refers to the data access layer, and contains the data access logic of the framework. This layer makes use of the PEAR MDB2 library, and as such database access is abstracted, giving access to any database supported by MDB2.

The View

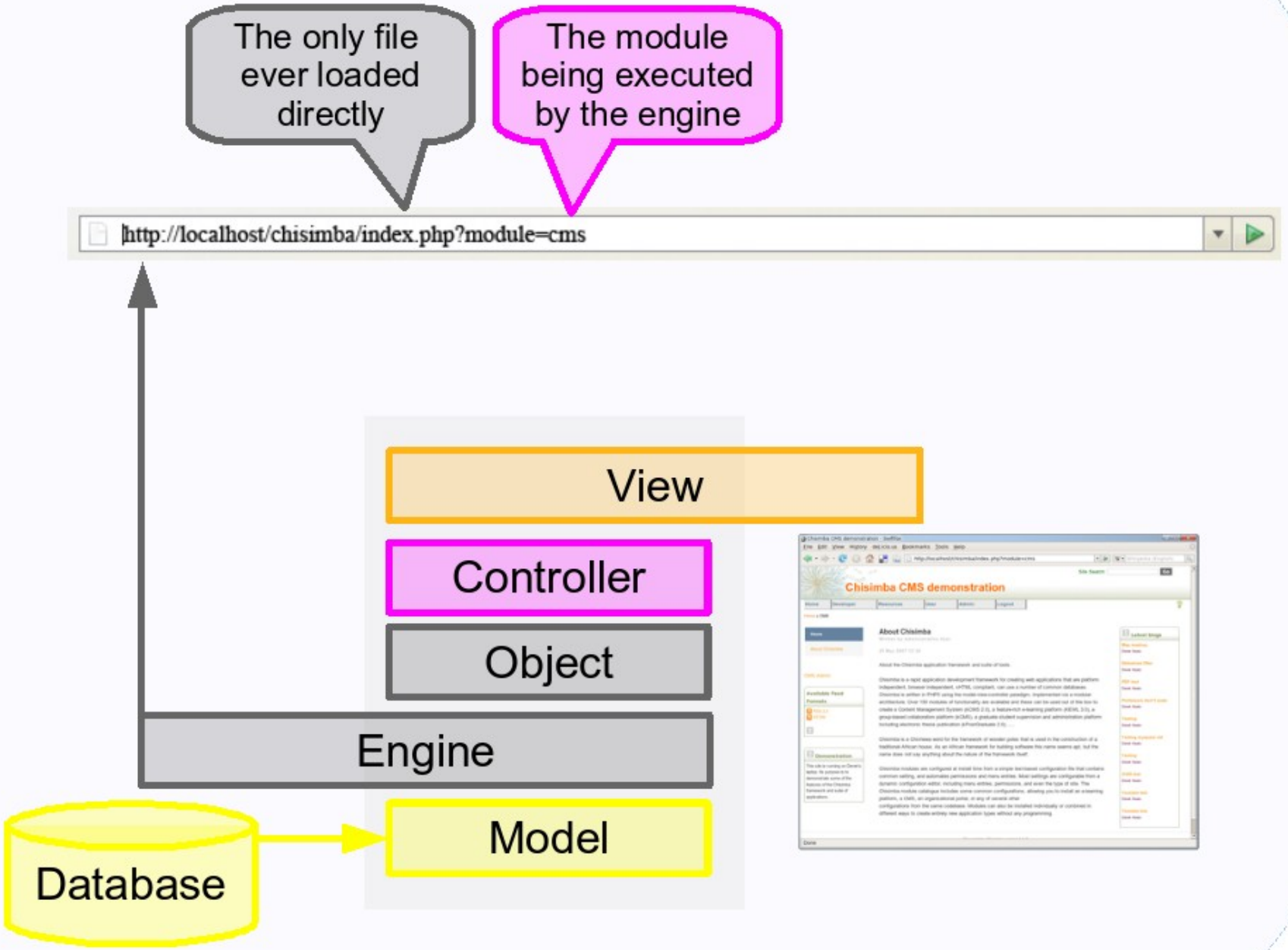
- The view is that component of the architecture that renders the model into a form suitable for interaction, typically a user interface element.
- In Chisimba, this means building a web page to interact with data and files, but it can also mean building an interface that is not meant for humans, such as an XML RPC, webservice, remote API, etc. Because the view is separate from the model, the same model code can be used to provide data for rendering in any of these types of interfaces. Because of the remote API potential of Chisimba, it can also provide data to remote desktop applications.

The controller

- The controller processes and responds to events, typically user actions or remote API calls, and may invoke changes on the model.
- In Chisimba, the framework provides controller functionality in the engine and object classes, and the module controllers extend this functionality to provide for business and application logic.

The objects

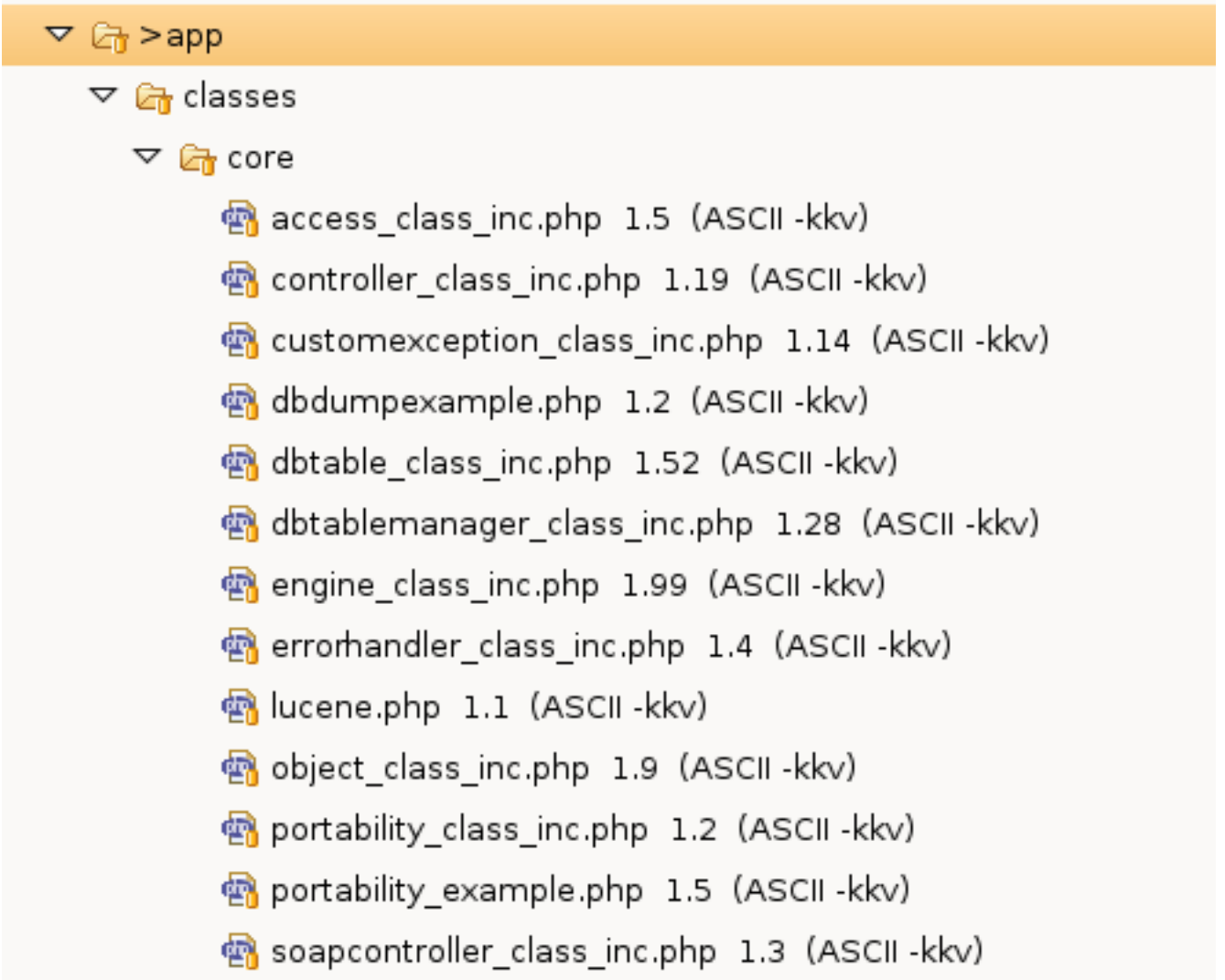
- Central to the framework are objects that reside in helper modules, and provide abstraction of various items such as HTML.
- Each module then consists of its own data access object, a set of objects that contain logic, and the templates used for display. One of the logic elements is the module controller which controls execution and is responsible for loading the templates used for rendering. We will be creating a simple controller, creating a very simple view template, and using the engine to run the module


The only file ever loaded directly
The module being executed by the engine
http://localhost/chisimba/index.php?module=cms
View
Controller
Object
Engine
Model
Database
Chisimba CMS demonstration

Controller execution

- As noted, Chisimba implements a front controller web implementation of the MVC pattern.
- There is only one file that is ever executed by the Chisimba engine class. This file loads and runs the controller of a particular module, which extends the object base class, thus giving the controller access to the core framework properties and methods. It does this by passing a module parameter in the querystring having the value of the module whose controller should be run.

Core-File Structure



Modular Structure

- Other than these classes, everything else in Chisimba is a module according to the modular architecture that Chisimba implements. This modular design allows Chisimba modules to be assembled in a Lego-block fashion, to create applications with different functionality from the same code base.
- There are modules that provide end-user functionality, and there are modules that simply provide a helper function. Helper modules contain one or more classes that is accessible to other modules to extend the framework's functionality. For example, *htmlelements* provides for abstraction of common HTML properties, and allows for consistent appearance of HTML and xHTML. The *blocks* module provides for the consistent rendering of widgets and other elements that should appear in the left or right margins. These are both helpers for building view elements. There are helpers to support other layers in the MVC pattern as well.

Module

View

Controller

Model

Module

View

Controller

Model

Module

View

Controller

Model

View

Controller

Object

Engine

Model

Helper modules

e.g. htmlelements

utilities

API

blocks

strings

sysconfig

security

modulecatalogue

lucene

Core Modules and Non Core

- As of version 1.0.3 of Chisimba, the code base has been split into chisimba_framework and chisimba_modules
- The chisimba_framework contains the core, as well as any modules that are considered essential for a working Chisimba implementation, including a basic user interface with a means to add content. Most of the helper modules are also contained in chisimba_framework.

Non core modules

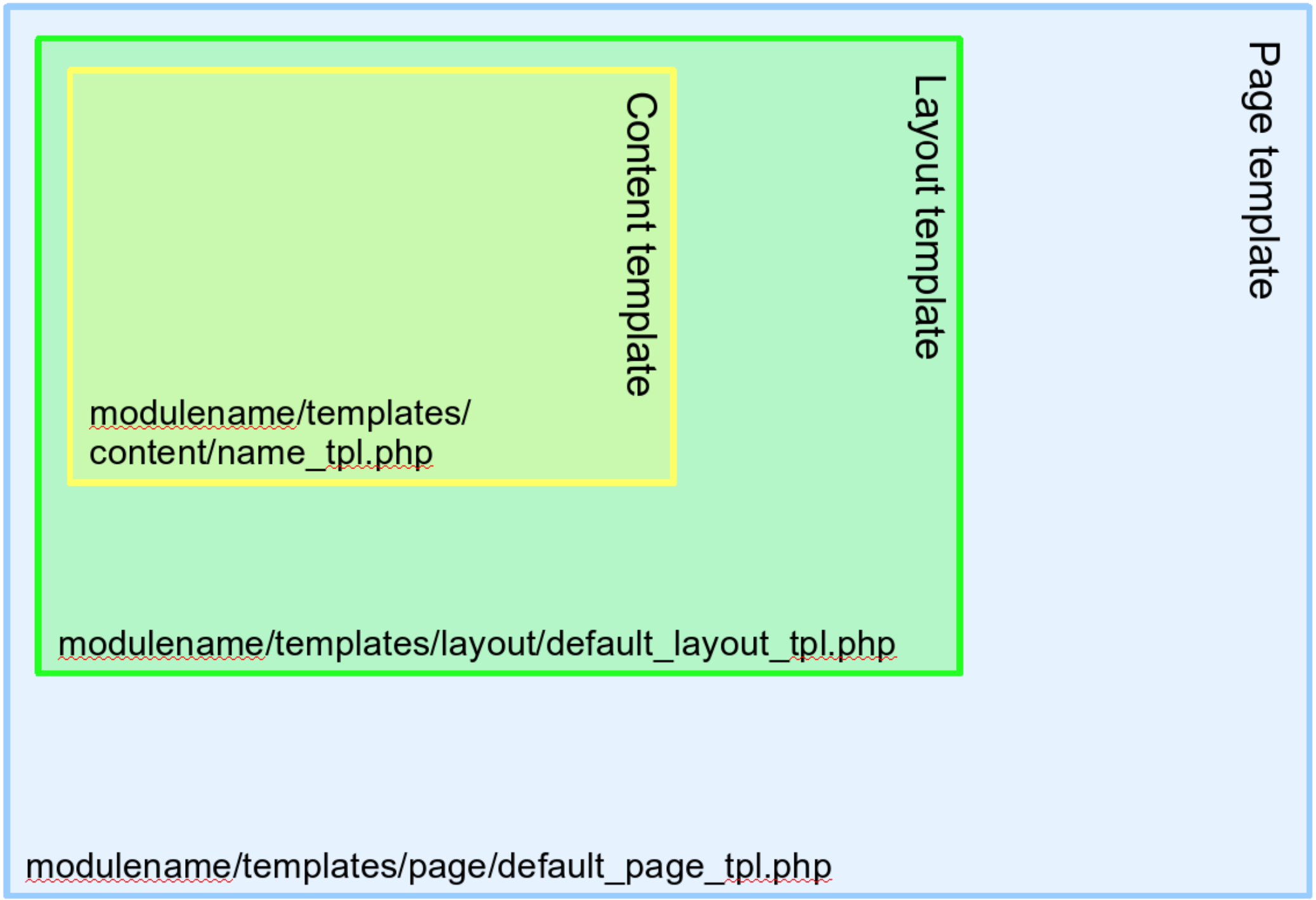
- The chisimba_modules repository contains a large number of modules providing for end user functionality, but none of these is essential for a working version of Chisimba, although you may wish to use some of those modules in order to reuse their classes, including view classes. On a working system, you do not need to check these out of CVS, they can be installed over the Web using the Chisimba Module catalogue.

Typical Module Structure

Directory or file	Purpose
/classes	The directory containing all the module's classes except the controller class, including the data access classes (M in MVC)
/templates	The templates (V in MVC), which are simple PHP or HTML files without any body or header information. The templates directory must contain a /content directory for the content templates. There may also be optional /page and /layout directories, but these are almost never needed, as there are methods to override page and layout in the main page and layout templates.
/sql	An XML file describing the data structure needed by the model. There may be more than one SQL file as long as they are all specified in register.conf. The XML is used to generate the database tables at install time.
register.conf	The registration file for the module. This file contains the information needed to register the module.
controller.php	The controller (C in MVC) for the module. The controller contains the logic and knows how to proceed depending on the value of various querystring parameters, the most important one being a parameter named *action*.

Component View

- A critical element to understanding Chisimba is the view component of the framework. The view component is implemented via templates. Every module needs templates for content rendering, but can optionally have page and layout templates. In general, modules rarely need page or layout templates, with the default templates serving needs adequately.


Page template
Layout template
Content template
modulename/templates/
content/name_tpl.php
modulename/templates/layout/default_layout_tpl.php
modulename/templates/page/default_page_tpl.php

Templates

- Templates are very powerful, but also can be complex if one wishes to manipulate page or layout templates. This is so because there is a cascade of templates that start in the common skins folder (skins provide for overall look and feel changes in Chisimba), may be over-ridden by page and layout templates in individual skins, which may in turn be over-ridden by page and layout templates within modules.

Templates

- Effectively, the page template controls everything up to the body tag in HTML terms, but can also be manipulated programmatically to render XML or any other form of content required. The layout template is responsible for elements that should be on all pages, as well as fetching the content rendered by the content template and outputting it to the interface. It is best not to alter the page or layout templates at the module level unless there is a real need for it. If you think you may need to do page or layout templates in a module, put a query on the mailing list before you proceed. Unless you really know what you are doing, the chances are there is a better, programmatic way to achieve your goals.