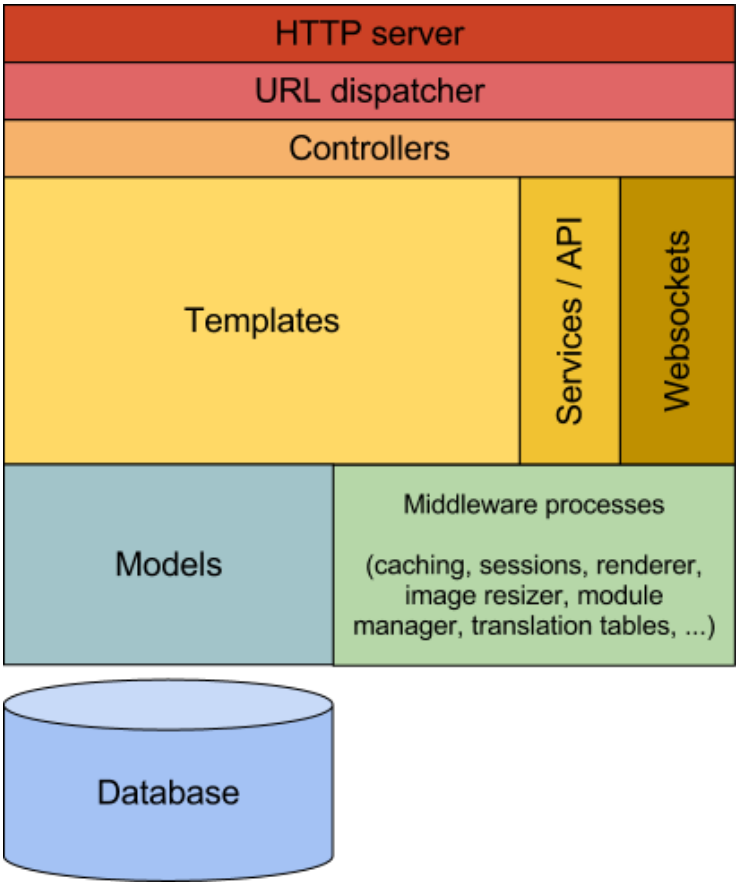
Zotonic

Introduction

Zotonic is an open source framework for doing full-stack web development, all the way from frontend to backend. It consists of a content management system (CMS), a web framework and a web server. Zotonic's main goal is to make it easy to create well-performing websites "out of the box", so that a website scales well from start. You can rapidly develop on top of Zotonic. If you need more flexibility, you can customise all aspects of Zotonic. The included web sever means can start working with Zotonic right away, without the need for installing Apache or something.

Zotonic architecture

Zotonic uses layers architecture like figure below.



http://aosabook.org/en/posa/zotonic.html

Zotonic comes with a built in web server. It does not require an external web server. This keeps deployment dependencies to a minimum.

Next, A URL routing system is used to match requests to controllers. Controllers handle each request in RESTful way, thanks to the Webmachine library.

Next, The general principle in Zotonic is that the templates drive the data requests. The templates decide which data they need, and retrieve it from the model.

Finally, Models expose functions to retrieve data from various data sources, like a database. Models expose an API to the templates, dictating how they can be used. The models are also responsible for caching their results in memory. They decide when and what is cached and for how long. When templates need data, they call a model as if it were a globally available variable.

Quality attribute scenario

1. Performance Scenario

   Source of stimulus : TechEmpower
   Stimulus : HTTP request
   Artifacts : Zotonic's dynamic dispatcher

   Environment : Normal environment

   Response : Processes requests

   Response measure : Handled 8500 request/sec

2. Performance Scenario

   Source of stimulus : Client

   Stimulus : around 1.5 million HTTP requests for voting new Dutch king

   Artifacts : system

   Environment : using bandwidth around 400 mbps

   Response : Processes requests

   Response measure : 99% of request handling times were below 200 milliseconds

3. Performance Scenario

   Source of stimulus : User request

   Stimulus : request data

   Artifacts : Webmachine

   Environment : Normal operation

   Response : Processes requests

   Response measure : 20% less processing time per request

Reference

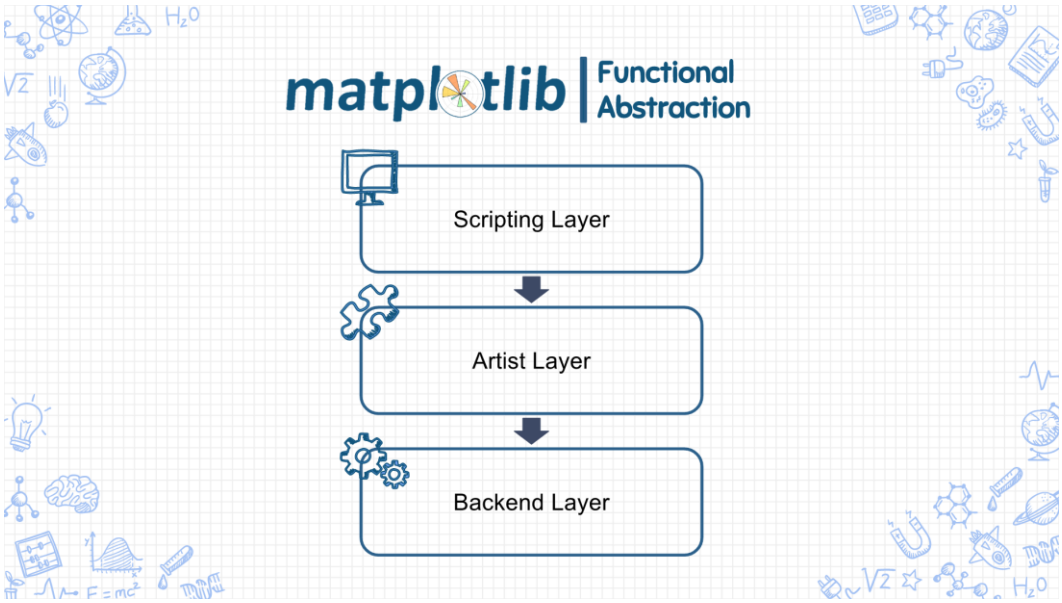
- http://aosabook.org/en/posa/zotonic.html
- https://zotonic.com/docs/1636/introduction

Matplotlib

Introduction

Matplotlib provides a convenient tool for creating 2D plots of data using Python. Matplotlib was initially developed by a group of self-taught programmers with a scientific background. At First, founder of matplotlib just want to create a new tool for handling data better than MATLAB that most popular in that time. But Matplotlib eventually evolved from a small project by a group of researchers into a widely used open-source tool.

Matplotlib Architecture



Architecture of Matplotlib shown at the figure below.

Matplotlib has 3 layers consist of

1. Backend Layer

   Matplotlib encapsulates functionalities that interact directly with the environment it is run on into the backend layer. There are three main components in this layer:

   - FigureCanvas : This component handles the concept of a surface that is drawn into to make the plots, a.k.a. “the canvas”.

- Renderer : This component does the drawing of the plots on the surface, a.k.a. “the paintbrush”.
- Event : This component handles user inputs such as mouse or keyboard events, a.k.a. "the viewer".

2. Artist Layer

   The artist layer is object that knows how to use Renderer to draw on FigureCanvas. Every image component inside a plot made by Matplotlib is an instance of the Artist class and communicates with the backend through the draw function. In fact, most of the heavy-lifting is done in this layer and it comprises most of code inside Matplotlib.

3. Scripting Layer

   The scripting layer however encapsulates the lower level image component renditions with a layer that is designed to be easy to use. his is done to comply with the initial design goal of Matplotlib which is to create a 2D plotting tool that can be used interactively.

Quality attribute scenario

1. Resource utilization

   Source of stimulus : Codecov
   Stimulus : set of tests
   Artifacts : whole system

   Environment : Normal environment

   Response : source code was run

   Response measure : 65.19% code coverage

2. Usability Scenario

   Source of stimulus : End user

   Stimulus : want to plot some line graph

   Artifacts : system

   Environment : configure time

   Response : line graph

   Response measure : End user feel satisfied because they get desired result in few seconds.

3. Portability Scenario

   Source of stimulus : End user

   Stimulus : want to plot line graph on their application by GTK+

   Artifacts : system

   Environment : configure time

   Response : line graph on their application

   Response measure : End user feel satisfied because their goal is accomplished in few seconds.

Reference

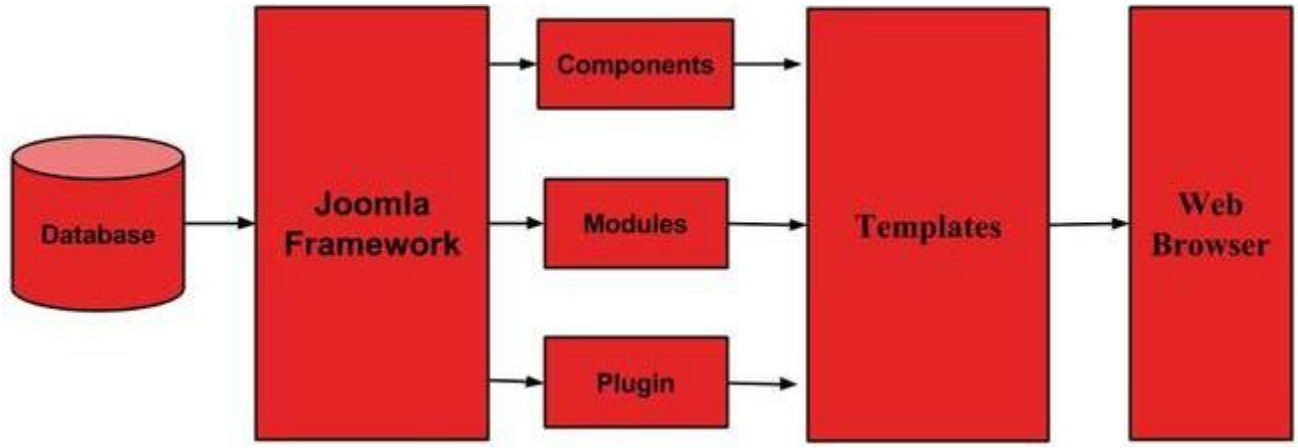
- https://delftswa.gitbooks.io/desosa-2017/content/matplotlib/chapter.html#introduction

Joomla

Introduction

Joomla! is a free and open-source content management system (CMS) for publishing web content. Joomla! is built on a model-view-controller web application framework that can be used independently of the CMS that allows you to built powerful online application. Joomla is designed to be used by people who have basic website creation skills and requires an Apache–MySQL–PHP server like LAMP or WAMP.

Joomla Architecture



The architecture of Joomla includes below layers :

1. Database

   Database itself is a collection of data and can be stored, manipulated and well organized in a specific manner. Using Joomla database layer, it ensures maximum flexibility and compatibility for extension.

2. Joomla Framework

   Framework is a collection of open source software, where the Joomla CMS is built. It breaks the framework into single modular packages which helps each package to develop more easily.

3. Component

   Components are considered as mini applications and it consists of two parts one is Administrator and second is Site. Whenever a page loads, component will be called to render main page body.

The Administrator part manages different aspects of the component and the site part helps in rendering the pages when a request is made by any site visitor.

4. Modules

   Modules is an extension used to render pages in Joomla. It is also used to display new data from the component. In Joomla administrator, modules are managed by the module manager.

   It displays the new content and images when the module is linked to Joomla components.

5. Plug-in

   this is one of the Joomla extension and is very flexible and powerful for extending the framework. It contains a bit of codes that is used to execute the particular event trigger. It is basically used to format the output of a component or module when a page is developed. Plug-in functions that are associated with events are executed in a sequence when a particular event occurs.

6. Templates

   Template drives the look of the Joomla website. There are two types of templates consist of

   - Back-end template is used to control the functions by the administrator.
   - Front-end template is way to present the website to users.

   Templates are easy to built or customize and provides maximum flexibility to style your site.

7. Web Browser

   It is server where the user interacts. It delivers the web pages to the client. The HTTP is used to communicate between the client and the server.

Quality attribute scenario

1. Usability Scenario

   Source of stimulus : Client
   Stimulus : want to create website
   Artifacts : system

   Environment : runtime

   Response : provides user interface for easier to build website.

   Response measure : Client feel satisfied because they can build website easier.

2. Usability Scenario

   Source of stimulus : Client

   Stimulus : want to build website with Joomla but not good in English.

   Artifacts : system

   Environment : runtime

   Response : provides several language in user interface

   Response measure : client can change language to their language in less than 1 seconds.

3. Security Scenario

   Source of stimulus : Client

   Stimulus : want to protect against CSRF attacks

   Artifacts : system

   Environment : runtime

   Response : inserting a random string called a token into each POST form and each GET query string that is able to modify something in the Joomla system.

   Response measure : Client can protecting CSRF attacks just add 1 line for checking to token

Reference

- https://www.wisdomjobs.com/e-university/joomla-tutorial-1207/joomla-architecture-15424.html
- How to add CSRF anti-spoofing to forms - Joomla! Documentation