

The Pylons Web Framework

Pylons is an open source web framework written in Python. It uses the model view controller architecture, follows the approach of convention over configuration and puts emphasis on loose coupling and clean separation. This helps developers to quickly create sophisticated web applications without hiding what is really going on.¹ In the following paper I will describe the way web applications have been developed in the past and what changes modern web frameworks have led to, especially Pylons.

Developing Web Applications the Old Way

In the past web applications were written as a series of simple CGI scripts. Although these scripts are easy to understand and give developers a lot of power, these scripts have many disadvantages. Some of them are redundant code, no separation of logic and design, a difficult to understand application structure, suboptimal data structures and database access, unreadable URLs and possibly a bad performance since interpreters and modules used by the scripts need to be loaded into memory on each request.²

Web Application Development with Pylons

Modern web frameworks have improved web application development and maintainability by addressing each of the issues described above. Taking a closer look into Pylons internal architecture helps to understand how things have changed.

Compared to other web frameworks such as Django or Ruby on Rails Pylons' core is remarkably slim and clear, reflecting its focus on loose coupling and clean separation. Instead of using lots of "glue code", Pylons uses low-level APIs that allow quickly and easy assembling of the desired components. Since Pylons also follows the approach of convention over configuration, a Pylons web application can be easily setup by using some of its carefully chosen default components. These include an http server, session management, template engines, URL rewriting, interactive debugging and so forth.

After a Pylons web application has been built, it is deployed on a server. This can be a Paste HTTP server that comes with Pylons, but other HTTP servers such as Apache work as well. Instead of being initialized on each request again, a Pylons web application is set up only once and held in memory by the server (as a Python object), which makes Pylons very fast.³

Internally Pylons uses the model view controller architecture, leading to a clear separation of the storage and retrieval of data (model), the business logic (controller) and the information representation (view). This separation not only improves the application's maintainability and expandability, it also helps to increase security and facilitate the development process as a whole (e. g. programmers and designers can work on the application independently).

¹ Gardner, 2009, p. 3-5.

² Some information about advantages and disadvantages can be found in V. Jayaram, 2007.

³ Gardner, 2009, p. 404-405.

Every HTTP request is processed by the Pylons application object in the HTTP server's memory (Fig. 1). Before a controller handles a request, it has to pass Pylons completely customizable middleware. At first the *Cascade* uses the *StaticURLParser* to see whether the URL requested matches a (static) file in the project's public directory. If no file is found "404 Not Found" is returned to the *Cascade* and from there forwarded to the *RegistryManager*. Since there is only one central application object, a function is needed that separates the different simultaneous threads, each with duration from the incoming request to the outgoing response, from each other. For this purpose the *RegistryManager* is set up. It keeps every thread in its scope and provides variables that seem to be global but are in fact limited to their thread. After that the request goes to the *StatusCodeRedirect* and the *ErrorHandler*, neither do anything relevant just yet, because they are only needed when it comes to the response after a controller action is called. As the names of the following *CacheMiddleware* and *SessionMiddleware* already suggest, they add further information to the thread-variables initialized by the *RegistryManager*, which automatically offer cache and session functionality in Pylons and its controllers. The last step in Pylons original middleware chain is the *RoutesMiddleware*, which is an improved version of Ruby on Rails routing system.⁴ The *RoutesMiddleware* compares the requested URL to a route map and extracts the routing results to the variables that have been initialized by the *RegistryManager*. After a request passed all these middleware components, Pylons uses the now available information to delegate the request to the controller action identified by the *RoutesMiddleware*.⁵

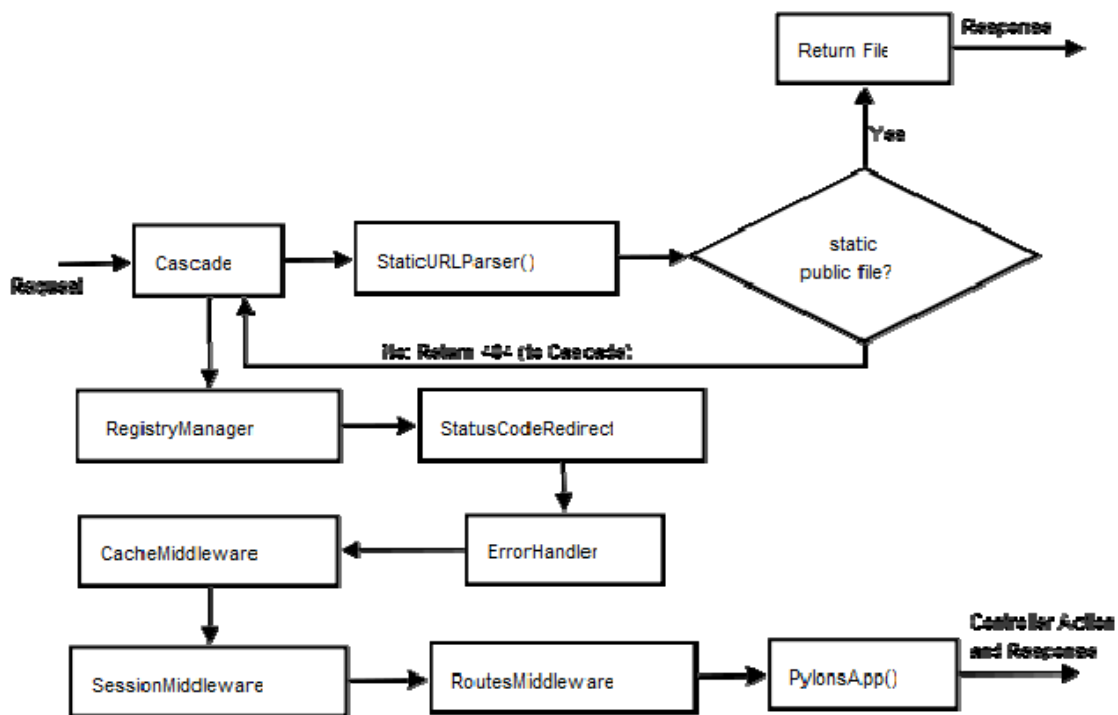


Fig. 1: The Pylons Middleware Architecture and HTTP Communication Handling

As shown so far, the default setup of Pylons already solves many of the issues of the early web's CGI script applications. Pylons has a straightforward architecture that clearly separates logic and design, comes with caching and session functionality, uses easily readable URLs and offers the possibility to customize everything as needed.

But there's a lot more Pylons can do to facilitate web application development. On the model layer Pylons offers many different ways to store data, e. g. in files in the file system, via web services, in object

⁴ Gardner, 2009, p. 195.

⁵ See Gardner, 2009, p. 408-411 for a detailed explanation of the Pylons middleware.

and XML databases and in relational database management systems (RDBMS).⁶ When choosing to use a RDBMS such as PostgreSQL or MySQL, object-relational mappers such as the famous SQLAlchemy can be used to map the database's data structures to objects in the Pylons application. As soon as objects in the application are changed, object-relational mappers automatically manipulate the underlying data.⁷

For authentication and authorization several third-party packages are available. AuthKit as well as repoze.who and repoze.what are some popular examples.⁸ These packages are implemented as middleware components and authentication and authorization checks can be easily added to controller actions by using decorators and predicates respectively, i. e. every method knows what (right/role) is required to perform it.

One of the most common responses of web applications is to return a HTML document. Templating systems such as Mako, Genshi or Jinja can be added to a Pylons application to support creating dynamic HTML documents. As always, controller actions pass the data that is necessary for the view to a chosen template. The template then uses this data and let it become part of a good looking design.⁹

There is still more Pylons has to offer: Besides a rapidly growing community and an increasing number of references there are internationalization tools, testing tools and logging facilities inside Pylons. Unicode support and systems for easily deployment in different environments are also included. Because Pylons is open source and has exceptionally open low-level APIs, missing functionality can be developed and added easily without changing Pylons core.

Conclusion

The architecture of modern web applications has dramatically changed since simple separate CGI scripts haven been used. Although modern web frameworks – and Pylons as one of them – may appear to be nontransparent and hard to understand at the beginning, they can, as soon as their general functionality and behavior is understood, facilitate the web application development process enormously. Caching, session management, authentication and authorization, URL mapping, database layer, object mapping and user interface generation are only some issues web frameworks can take care of.

References

- AGENDALESS CONSULTING. 2009a. *repoze.who – WSGI Authentication Middleware*. [WWW] <http://static.repoze.org/whodocs/>. (April 2009).
- AGENDALESS CONSULTING. 2009b. *repoze.what – Authorization for WSGI applications*. [WWW] <http://static.repoze.org/whatdocs/>. (April 2009).
- BEL-EPA. 2009. *AuthKit*. [WWW] <http://bel-epa.com/docs/authkit/>. (April 2009).
- GARDNER, J. 2009. *The Definitive Guide to Pylons*. Berkeley, CA: Apress.
- V. JAYARAM. 2007. *cgi Scripting Language*. [WWW] <http://www.hinduwebsite.com/webresources/cgiscripts.asp>. (April 2009).

⁶ Gardner, 2009, p. 127.

⁷ Of course a lot of configuration is possible to keep the performance on a good level.

⁸ See Bel-EPA, 2009 for AuthKit and Agendaless Consulting 2009a and 2009b for repoze.who and repoze.what.

⁹ See <http://www.makotemplates.org> for Mako, <http://genshi.edgewall.org> for Genshi and <http://jinja.pocoo.org> for Jinja.