Jam.py web application framework

By definition, a web framework is a code library that makes the development of web applications faster and easier, providing common approaches for building accessible, scalable, and easy to maintain applications. For more than 15 years, professional web development has always assumed the use of a web framework, except in extremely unusual situations.

Jam.py framework
The true refreshment comes from the northeast of the European continent. New framework, new ideas. Jam.py is a client-server, event-driven, monolithic framework designed to create web business applications. It's a beautiful combination of HTML, Java Script, jQuery, CSS and Python code. And that is all. Application creation starts and ends in an application builder, which is nothing other than web applications written in Jam.py framework, and runs on the same server as the desired application.

The developer here creates a new database schema, or uses the existing one, creates forms, reports, menus and other application details. Jam.py supports all known open source databases: SQLite, MySQL, PostgreSQL, Firebird and proprietary MS SQL Server and Oracle. Jam.py has built-in authentication and authorization as well as session data management. Jam.py supports multilingualism and has the ability to expand to new languages.

Jam.py has a code organization according to the event-driven principle. There is a strong literature that explains the benefits of the MVC way of organizing the code. The author of this text does not share such an opinion, if you know anything about how the event-driven system works, you will easily manage and immediately embark on an adventure with Jam.py.

An absolute novelty is to organize the code in a hierarchical form, where the ability to define functionality appears at different, predefined levels. In this way, the author of the framework successfully defines the predefined functionality at the basic level - the task level, and it is for the user to change the proposed functionality at the level of the group of objects or at the object level.

Group of objects gather objects similar to some of the essential features, so there are journals, catalogs, reports and details. The very existence of a group provides the developer with the ability to define common code at group level.

Journals gather objects that contain transactional data, catalogs are objects that contain rarely changed data and define the basic system factors, object reports are pre-prepared spreadsheet templates in which Jam.py maps the data and converts it to the format as desired by the user. In order to do everything properly, it is necessary to install a spreadsheet software on the server with the corresponding possibilities of conversion. The details group look a little mysterious - these are objects that come with predetermined properties necessary to join with master objects from the journal and catalog groups. Another unprecedented feature is the ability to join one detail with several master objects and vice versa.

The objects are nothing but the user interface and data representation, and most closely correspond to action queries from some other known systems. The main carrier of an object, depending on its properties, can be a table in the database. The most important parts of the object are, of course, fields with all the necessary data types and a standard programming tool - lookup fields. There are also delicious as lookup over SQL Joins, lookup of lookups, etc.

Jam.py builds a complete application that runs and manages data without the need for writing and one single source code line.

And what does the programmer do here? The code is needed for more complex validations, for different data management while entering or writing data to the database and in many other places. The code is written in the editor that is part of the builder. Editor places the code in separate modules for the client and server part.

Jam.py is not a code generator. The developer builds the application, and the framework enters the programmer's selections and settings into the project's sqlite database. When a user accesses the Jam.py application, Jam.py loads the settings from the project's sqlite database and the application enters its work cycle.

Good sides of Jam.py framework

- Standard *pip install jam.py* and *python setup.py install* installation modes
- The learning curve is very mild due to the presence of online application builder and excellent, comprehensive documentation with a multitude of graphics and case studies.
- The programmer is dedicated only to working with business rules. There are no details related to HTTP protocol, routing, and similar things that should not be expected from a web business developer.
- From an HTML template, this is one of the most common html files, the structure of which you can get in 15 minutes.
- Absence of the need for system configuration.
- The presence of all classic web application elements, such as encoding, sessions, authentication, authorization, static content management, wsgi dev server with built-in debugger.
- The ability to run on Python2.7 and Python 3.5, Python 3.6 and Python 3.7 wsgi to web servers.
- Import and export applications to other servers and other databases.
- Online maintenance with built-in builder and editors.

Bad sides of Jam.py frejmworka

- Although Jam.py has a python in the name, it's primarily a Java Script web framework that uses Python for the background. Users can be surprised by the fact that a lot of work ends up on the client's Java Script side.
- There is no additional code in the form, add-on or plug-in, because of the simple fact that the frame is still very young and unknown and with a small number of followers. In this way, complete development is completely dependent on the author.
- A complete predefined part of the code and objects required during development and in production is defined in the form of a SQLite database. Therefore, the standard procedure for versioning the code is almost unfeasible, since the known SVC systems are poorly managed with binary formats.

Conclusion

The author of this text has many years of experience in trying to find a good web framework. Consequently, the idea with a graphic online bilder and editor was not a side, as well the elementary

theory of relational database manager systems. Therefore, the start with the framework was very light, the framework enabled the construction of a complete application with all supporting elements in a really short time.

But as soon as there was a need for writing code, things changed. First of all, the need for java script is unavoidable. Furthermore, the hierarchical way of processing events in the application tree is something really unusual and needs to be accepted. It is similar for the server code, although it is considerably less than the client's and its writing is directed to the hierarchical structure of the frameworks. However, the amount of code necessary to obtain the results is much smaller compared to the MVC framewroks.

The saying says that good things are easy to get used to. That's the case with Jam.py. There is not a single product on the market that allows the construction of a web application to be as fast and easy. Due to its capacity and focus, the framework aims to develop enterprise applications. Of course, it takes time and commitment of the author and the community created around this project to promote and spread the ideas Jam.py brought us.