# HOW TO MANAGE NECESSARY AND BORING EVIL OF PYTHON APPLICATIONS SETTINGS FILES

**Pavel Špulák and Jan Brothánek**

Capt. Pavel Špulák
Ministry of the Interior – General Directorate of Fire and Rescue Service of the Czech Republic
Department of Information and Communication Technologies
Kloknerova 26, PO Box 69, 148 01 Prague 4, Czech Republic
Tel.: +420 950 819 851, Fax: +420 950 819 965, E-mail: pavel.spulak@grh.izscr.cz;

Col. Jan Brothánek
Ministry of the Interior – General Directorate of Fire and Rescue Service of the Czech Republic
Department of Information and Communication Technologies
Kloknerova 26, PO Box 69, 148 01 Prague 4, Czech Republic
Tel.: +420 950 819 655, Fax: +420 950 819 965, E-mail: jan.brothanek@grh.izscr.cz;

***Abstract***
*Python programming language is widely used in GIS, machine learning, and data analysis applications. The richness of powerful libraries, together with the speed of coding and rapid learning curve, makes Python language a valuable tool for various tasks connected with spatial data analysis and data analysis in general.*

*On the other hand, Python language gives coder, in comparison with, e.g., Java language, a high level of freedom, which can lead to poorly arranged and hardly manageable spaghetti application code.*

*Fortunately, such problems can be overcome with the careful organization of the application code and settings (or configurations). Owing to this fact, one of the very first steps in the organization of application code must be a design of the hierarchy and structure of application settings files and their processing mechanism.*

*In this contribution, we would like to show one of the possible approaches to application settings files organization and processing. Our approach is based on the careful analysis of the application setting needs, its separation to the hierarchical structure of files, file processing, and final creation of the tree of the dynamically build classes and its attributes containing easily accessible application task settings.*

***Keywords:*** *Python, settings, configurations, processing*

## INTRODUCTION

Sooner or later, each programmer or coder will face the problems connected with the organization and processing of application settings. Unfortunately, wrong decisions in this area can make further work a nightmare, substantially slow down the progress of work, make application error-prone. They can cost additional demanding efforts to correct it. It also negatively influences the feature development of the application.

It is very tempting to neglect issues concerning the settings because it is “useless work” that provides no immediate results related to the application's main aim. On the other side, if we overcome this temptation, we will soon discover that the time and work invested into the organization and processing of settings will return, and the further development of the application will be much easier.

The approaches described in the further text were used during the development of the application used for the generation of the maps and charts for the Statistical Yearbook of Fire and Rescue Service of the Czech Republic. This Yearbook is published every year in the March issue of the 112 magazine [1, 2] in the Czech language. It contains about 60 maps and charts, which must be carefully prepared from the data collected during the year on various departments of General Directorate and regional directorates of Fire and Rescue Service of the Czech Republic. Besides, some of the maps and charts are selected for the English version of the Yearbook.

The preparation of the Statistical Yearbook of Fire and Rescue Service of the Czech Republic is time and precision demands. It requires careful processing of a large amount of data of various types from various sources. Because of that fact, it was decided to create an application for data processing and map and charts creation. The main aim of the application is to make the whole process as much as possible automatized, standardized, repeatable, and less error-prone.

## ORGANIZATION OF THE SETTINGS FILES

The buildup and maintenance of such an application, as is previously mentioned, is hardly ever possible without external settings files. They enable rapid reaction to the demands concerning the application outputs. Owing to this fact, great attention was paid to its organization, hierarchy, internal structure, and processing. The starting point for the design of structure and procedures for the processing of settings files was a careful analysis of the application processes. During the application design phase, the following application model was proposed:

1. The application will produce in principle two types of graphical outputs, maps, and charts in various formats.
2. Each output (map or chart) will be an application task.

From the application model, follow the hierarchy of the entities mentioned above:

1. Application.
2. Task type (map or chart).
3. Application task.

Owing to these facts, the five types of settings files where proposed:

1. Main application settings file, which contains application-wide settings. It encompasses both settings of application as is and common and default settings of task types and tasks carried out by the application. The main settings file also contains the instruction for processing the settings in it, which will be taken over by the task type settings and task settings.
2. Settings file of map task. It contains the settings common to all map tasks. During the processing, it takes over the settings from the main settings file and overwrites it, if more specific settings are given. This settings file also contains the instruction for processing the settings in it, which will be taken over by the task settings.
3. Settings file of chart task. It contains the settings common to the all chart task. During the processing, it takes over the settings from the main settings file and overwrites it, if more specific settings are given. This settings file also contains the instruction for processing the settings in it, which will be taken over by the task settings.
4. Settings file of the particular task. It uses main application settings and map or chart task settings and overwrites it if settings that are more specific are given in it.
5. The application uses a logging framework for protocoling of its run [3]. The settings of the logging is independent on the other settings of the application. Because of that fact, it was decided to store logging settings into a separate file. Application mechanism is used for its load, and the standard mechanism of the logging library is used for logging initialization with settings in the form of a dictionary [4]. After its load, the settings file is only slightly modified to set the actual path for the log file because all outputs generated by the application, including the log, are stored in the output folder and subfolder.

## FORMAT OF THE SETTINGS FILES

The application settings must be stored in some hierarchical and structured form. Usually, three types of file formats are used for storing such kinds of information, xml [5], json [6], and yaml [7]. Each of these formats has its strengths and weaknesses. The xml file format is for a long time established and used. There are well-established tools for its processing and validation. Some languages, like, e.g., Java, have a mechanism for its automatic conversion into a tree of full-featured objects.

The Python 3.X language was selected for the creation of the application. The main reasons for its selection were the existence of the well-established libraries for productions of the maps and charts, the ArcPy [8], and Matplotlib [9] libraries. In the light of the selection of the Python language, the possible formats of the settings files were compared, and json format was selected. The main reason for its choice was its simple conversion to the Python natural language structure – dictionary [10]. The settings are stored in the JSON file format, which is in principle kind of text file, both human and machine-readable. It can be quickly loaded from a file as a string and further processed into the Python dictionary. On the other hand, the Python dictionary does not provide dot notation for access to lower levels of information and does not give the possibility to define their own methods. Its conversion to the object with a complex structure and modifiable attributes is not simple. Owing to these facts, the simple mechanism for the transformation of the dictionary to the full-featured object is further discussed.

## SETTINGS FILES PROCESSING

The task settings were processed in the form of a tree of user-defined classes and their attributes. This approach enables a better and more logical organization of the application code. The task settings respects hierarchy of settings files and usage of the values specified on the highest levels of hierarchy enables the basic unification of the tasks, especially the unification of the maps and charts look. The values defined on the highest level are taken over if no more specific value is defined.

The instruction for processing the dictionary of task settings into the tree of the classes and its attributes are created by the merging of the instruction contained in main settings and task type settings. They describe the classes and attributes to which the settings of the task will be converted and its place in the hierarchy tree. It is the dictionary with class names as keys. The value in the key-value pair in the dictionary is a list of the dictionaries with instruction for processing particular class attribute. The instructions consist of the following entries:

1. Name of the attribute.
2. The initial value of the attribute, which is used if the value that is more specific is not given.
3. Description of the attribute, which is used in the string representation of the class.
4. Name of the class, if attribute value will be converted into a class of a particular type.
5. Special parameter signalizing if a list of text string will be joined into one text string. This parameter is used exclusively during the processing of the SQL queries, which are responsible for downloading and treatment the data from the database. The SQL query has a form of a list of strings in the settings file because this enables its better readability in the settings file. This list of strings must be converted to the string before its sending to the database.When task settings as a dictionary are ready together with its instruction for processing, they are send to the special class – task settings factory, which generates the task settings as tree of classes and its attributes. During the generation the more specific values replaces the less specific predefined values. For each class its variables, constructor and string representation are set.

During the creation of the constructor, the class variables are defined, and its values are set. This process is simple except the situation when the class variable is an instance of another class or list of instances of the other classes. In this case, the procedure for class creation is recursively invoked. It can be invoked one time if the creation of the one class is necessary or many times if there is a necessity to create a list of classes.

The class variables use a slot [11] mechanism for its storage. The special attribute __slots__ allows you to explicitly state which instance attributes you expect your object instances to have, with the expected results:

1. faster attribute access,
2. space savings in memory.

The string representation of the class is created by iteration through the class variables. Each variable is represented by the string consisting form the variable description and its value. The string representation also respects the hierarchy of the classes in the task settings class tree and gives appropriate ident to the string representing the particular class.

The description of the procedures for the dynamic creation of the class during the application run can be found in the literature [12]. The algorithms described there was taken as a background for task settings processing and was substantially modified and extended.

## CONCLUSIONS

The proposed approach to the settings file processing was successfully tested and deployed for processing of the settings files of the application for maps and charts preparation for the Statistical Yearbook of the Fire and Rescue Service of the Czech Republic.

It provides an easy and straightforward way from the settings files in the json format to the dynamically created classes and its attributes containing application settings. The process of class creation is fully under coder's control and can be easily modified according to the current needs. The settings, which control the creation of the tree of classes, are simple parts of the json files.

The application uses direct access to the class variables containing settings. Still, the addition of the code for generation of the setters and getters methods for accessing the class attributes is simple, if it will be needed in the future.

## REFERENCES

1. Časopis 112. Hasičský záchranný sbor ČR [online]. Praha: Hasičský záchranný sbor ČR, 2020 [cit. 2020-04-15]. Available from: https://www.hzscr.cz/casopis-112.aspx

2. Časopis „112“ :: Magazine “112” :: Die Zeitschrift „112“. Hasičský záchranný sbor ČR [online]. Praha: Hasičský záchranný sbor ČR, 2020 [cit. 2020-04-15]. Available from: https://www.hzscr.cz/clanek/casopis-112-magazine-112-die-zeitschrift-112-976128.aspx

3. Logging — Logging facility for Python. Welcome to Python.org [online]. Python Software Foundation, 2020 [cit. 2020-04-28]. Available from: https://docs.python.org/3/library/logging.html

4. Logging Cookbook. Welcome to Python.org [online]. Python Software Foundation, 2020 [cit. 2020-04-28]. Available from: https://docs.python.org/3/howto/logging-cookbook.html

5. Extensible Markup Language (XML). World Wide Web Consortium (W3C) [online]. World Wide Web Consortium (W3C), 2020 [cit. 2020-04-28]. Available from: https://www.w3.org/XML/

6. JSON [online]. 2020 [cit. 2020-04-28]. Available from: https://www.json.org/json-en.html

7. The Official YAML Web Site [online]. Oren Ben-Kiki, Clark Evans, Ingy döt Net, 2020 [cit. 2020-04-28]. Available from: https://yaml.org/

8. What is ArcPy?—ArcPy Get Started | Documentation [online]. Redlands: ESRI, 2020 [cit. 2020-04-23]. Available from: https://pro.arcgis.com/en/pro-app/arcpy/get-started/what-is-arcpy-.htm

9. Matplotlib [online]. The Matplotlib development team, 2020 [cit. 2020-04-23]. Available from: https://matplotlib.org/

10. Json — JSON encoder and decoder. Welcome to Python.org [online]. Python Software Foundation, 2020 [cit. 2020-04-28]. Available from: https://docs.python.org/3/library/json.html

11. Usage of __slots__. Stack Overflow [online]. New York City: Stack Overflow, 2019 [cit. 2020-05-27]. Available from: https://stackoverflow.com/questions/472000/usage-of-slots

12. RAMALHO, Luciano. Fluent Python. Beijing: O'Reilly, 2015. ISBN 978-1-491-9-46008.

## BIOGRAPHY

***Capt. Pavel Špulák***

I was born in Prague in the year 1973. In 1991 I had finished The Institute of Chemical Technology, Prague, and obtained a degree in chemical engineer. After that, I worked in various research institutions and universities. Since 2005, I became a member of the Fire and Rescue Service of the Czech Republic, and I am working at the General Directorate. At the present time, I am a member of the Department of Information and Communication Technologies. My work is focused on various ICT, GIS, and coding issues. My favorite technologies are Python, Java, JavaScript, CSS3, HTML, Oracle, PostgreSQL, SQLite.

***Col. Jan Brothánek***

I was born in Ostrava in the year 1983. I have finished the College of Civil Engineering in 2002 and 2008, I have finished the Technical University of Ostrava and obtained a degree of the engineer. My studies there were specialized on GIS. After completing University, I worked as an implementation specialist in the software development center of the PPF group. Since 2010, I am a member of the Fire and Rescue Service of the Czech Republic, and I am working at the General Directorate as a team leader responsible for ICT development and project management.