ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 1 Issue 2, November, 2021 / Page No: 257-259
Paper Id: JETA-V1I2P126 / Doi: 10.56472/25832646/JETA-V1I2P126

Original Article

Configuration Data as Code- Evolution in Java/J2EE Platform

Rakesh Rikhi[1], Deepika Rikhi[2]
[1,2]*Austin, Texas, USA*

Received Date: 26 September 2021 Revised Date: 29 October 2021 Accepted Date: 27 November 2021

Abstract**:** *This paper explores the evolution of configuration data usage in code. The example taken is of Java/J2EE Platform, where in the early 2000s, the configuration data was used to be stored in files, and with the upgrades of Java, new features, like annotations, were introduced to store configuration data in the code. This paper also discusses the merits and pitfalls of approaches, configuration data in file and configuration data in code.*

Keywords: *Config As Code, Config Files, MVC Pattern, Struts Framework, Spring Framework, Java, J2EE, Spring Web Framework, Annotations.*

I. INTRODUCTION

Configuration files have been used in computer programs since the start of programming. These files are used to configure various parameters and settings of a computer program. These are used in a wide range of situations, from operating systems to deployment systems like containers.

The main use of the configuration files is to separate program logic in the form of code from the configuration data, which is used to state the static properties of the program. These parameters are used by the program at the start of the program and are generally not altered while the program is running. The config files are very useful in large, complex software systems where changing the code can be risky, involves regression risk, and is time-consuming.

Simplistically, the config file can be a text file that contains a key-value pair of the parameter or settings and its value. The parameters can be like the color scheme of an app or a default start directory of the app.

There are different parts of the configuration files-
a) Config data- This is the actual data that gets used in the program logic.
b) Comments- This is the text that is used for annotations, comments, headers, and footers to describe the configuration file or data.

i) Comments- These add notes of the config file
ii) Header- A specialized comment at the start of the file, commonly used to give the context of the file or section.
iii) Footer- A specialized comment at the end of the file

The configuration files can be written in different formats. It can be a simple text-based file having key-value pairs, XML format, or JSON format.

In larger and more complex config files, the data is organized into sections. Headers are donated at the start with a special character. The data is also organized into sections or logical groups. For example, in the pictures below, there are sections for network parameters and sections for display settings. These settings make it easier to read and maintain the config file.

One can also use hierarchical overrides and settings in the config file. This is very useful when there is a tree or hierarchical dependencies among the parameters. The leaf level parameters or keys can be reused at various other tree nodes.

The main benefit of using the config file is the separation of code and configuration data. This means that one can change certain behaviors in the code without changing the source code or just by changing the parameters in the config file.With the advent of server-side programming, config files also make it easier to deploy code and scale applications. The continuous build and deploy (CBD) pipeline makes use of config files to build and deploy the code in various environments like dev, QA, and production by using the same config files but with different parameter values.

Web application frameworks like Struts and Spring, which use the MVC (Model View Controller) pattern, created config templates that enabled the template-based approach of making and deploying web applications inside the web/application containers using Java/J2EE architecture. The config files also play a vital role in the portability or platform interoperability of the software solution. The same config file formats can be used with different values to make cross-platform tools.

```json
{
    "glossary": {
        "title": "example glossary",
        "GlossDiv": {
            "title": "S",
            "GlossList": {
                "GlossEntry": {
                    "ID": "SGML",
                    "SortAs": "SGML",
                    "GlossTerm": "Standard Generalized Markup Language",
                    "Acronym": "SGML",
                    "Abbrev": "ISO 8879:1986",
                    "GlossDef": {
                        "para": "A meta-markup language, used to create markup languages such as DocBook.",
                        "GlossSeeAlso": ["GML", "XML"]
                    },
                    "GlossSee": "markup"
                }
            }
        }
    }
}
```

Figure 1: Config – JSON file having key-value pairs.

```xml
<!DOCTYPE glossary PUBLIC "-//OASIS//DTD DocBook V3.1//EN">
 <glossary><title>example glossary</title>
  <GlossDiv><title>S</title>
   <GlossList>
    <GlossEntry ID="SGML" SortAs="SGML">
     <GlossTerm>Standard Generalized Markup Language</GlossTerm>
     <Acronym>SGML</Acronym>
     <Abbrev>ISO 8879:1986</Abbrev>
     <GlossDef>
      <para>A meta-markup language, used to create markup
languages such as DocBook.</para>
      <GlossSeeAlso OtherTerm="GML">
      <GlossSeeAlso OtherTerm="XML">
     </GlossDef>
     <GlossSee OtherTerm="markup">
    </GlossEntry>
   </GlossList>
  </GlossDiv>
 </glossary>
```

Figure 2: Config – XML file having key-value pairs.

II. CONFIG DATA AS CODE

The pivotal moment of using metadata or config data as code came in Java Platform with Java5 release in 2011. The main driver behind this integration was that the programmer should be able to read and maintain all the source code in one place. However, the challenge of keeping the code for config parameters always remained.

In the early 2000s, there were two tools that brought annotations to the mainstream-

- EJBGen- add Javadoc annotations to the source code and then generate complex EJB XML descriptors
- XDoclet- provided a general framework for using Javadoc tags as annotations.

There has always been debate on whether specific metadata should be stored in annotations or in config files. The general rule is that if the metadata is associated with Java elements (packages, classes, fields, variables, methods), then annotation is used. Otherwise, use the configuration file.

Please note that the power of annotations is that these are completely ignored by the compilers, so they are not part of the generated machine code. It also makes it clear that annotations have to be interpreted by associated tools so that they can make use of the configuration data.

The above methodology became popular for web applications where most of the program flow adheres to a template, and tuning and configuring the parameters are needed. Spring is a popular web framework that uses Java Configuration. The core annotations are @Bean and @Configuration.

a) Because of so much variety in the usage of configuration data, there are certain best practices for managing this data-

- Securing sensitive data- When the data is not in the generated machine code, the config files are human-readable, so encryption should be used to store the data. Proper encryption and decryption algorithms have to be used to configure this data.
- Using the Version Control Systems- The config data has to be treated like code and stored in the VCS systems so that the audit trail can be traced in the future for changes.
- Naming Convention- As config data is written like plain English, proper naming conventions have to be used; otherwise, the names and hierarchy of the parameters can quickly go out of hand and become non-readable, defeating the essence of configuration data.

b) Challenges

As the source code has always taken center stage for the program logic, there are many best practices and guidelines available for writing the source code, which is extensible and maintainable. A similar maturity has yet to be experienced for the configuration data. Because the configuration data can now be written in code and in separate config files, understanding the program logic can become cumbersome for large, complex systems.

Industry-level and organization-level guidelines and protocols have to be floated so that configuration data is modeled in a uniform and standard way across the industry.

A. Futuristic View

Configuration as a Code concept is becoming popular day by day in the software engineering space. Many tools like Jenkins are floating plugins. The config as code is an essential part of modern CI/CD pipelines, policy modules, and build and test pipelines.

III. REFERENCES

[1] Configuration as Code- Feb 2020- Configuration as Code
[2] Configuration Testing: Testing Configuration Values Together with Code Logic – May 2019- Configuration Testing: Testing Configuration Values Together with Code Logic
[3] A Systematic Mapping Study of infrastructure as Code research- Apr 2019 - A systematic mapping study of Infrastructure as code research - ScienceDirect