Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 07, July-2021

Handling Unhandled Exceptions in Python 3 using Dependency Injection (DI) or Inversion of Control (IoC)

Manasa Nageshkumar
Computer Science and Technology
Usha Mittal Institute of Technology
Mumbai, India

Shivani Mandavkar
Computer Science and Technology
Usha Mittal Institute of Technology
Mumbai, India

Shreya Santosh
Computer Science and Technology
Usha Mittal Institute of Technology
Mumbai, India

Sumedh Pundkar
Asst. Professor
Usha Mittal Institute of Technology
Mumbai, India

Abstract— **Programming languages are the basic foundation for the development of any software. There are many programming languages existing and each language has its own set of limitations and exceptions. An exception is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions. When an error occurs the build-in exception handler tries to solve the given exception. There are a set of methods and functions that handle these predefined errors and raise exception when these errors occur. Python supports its user to define user defined exceptions. There are situations where the program may not give us the expected output and also no exceptions are raised. Such conditions can be stated as unhandled exceptions. These conditions are not mentioned in any documentations. Identifying such exceptions and handling them is also an essential requirement. This paper deals with handling unhandled-exception using Dependency Injection or Inversion of Control.**

Keywords— Dependency Injection (DI), Inversion of Control (IoC), Model View Controller (MVC).

I. INTRODUCTION

Most of the high-level languages like Java, .Net, Python 3 has its own exception handling mechanism. Exception handling mechanism helps in resolving the exceptions to ensure proper running of the program. As all the exceptions may not be pre-defined, sometimes the user may come across an unknown exception which does not belong to the built-in exceptions, such exceptions are called as Unhandled Exceptions [1]. Exceptions can occur anywhere in a program that may be in a small block of code or in a more complex system. Exceptions that occur in a small block of code can be handled easily. If there are many exceptions in a complex system then, handling them would be difficult, especially if there are some Unhandled Exceptions. Adding these new unhandled exceptions every time, when an Unhandled Exception occurs would be tedious task [2]. Since, Unhandled Exceptions are difficult to find, finding and handling them beforehand is vital. This paper mainly focuses on dealing with the Unhandled Exceptions in Python 3 using Dependency Injection (DI) or Inversion of Control (IoC).

II. DEPENDENCY INJECTION

Dependency is concretely related to object-oriented concepts. Object Oriented concepts are entirely and purely based on the real-life scenario, it simulates the real-life scenarios into objects and classes. The term dependency simply states that a class is dependent on another class. i.e., the classes are tightly coupled with each other. If a process in one class is disturbed eventually leads to failure of the other. Similarly, if we want to update a piece of code in one class we need to make appropriate changes in the dependent one too. In a smaller scale application these changes do not seem to be a big problem. But, while building large enterprise applications this creates a serious issue. In large enterprise applications, we are dealing with humongous chunks of code, failure of one dependent code will definitely create chaos, since there are many components and services involved in the process. In addition to this, locating the correct service, which service to communicate with and using it also becomes a very tedious task. Therefore, tight coupling makes the system less flexible and difficult to maintain [3].

Dependency injection (DI) helps us overcome this issue. Instead of having the classes depend on each other, an external entity termed as a container or injector will inject the necessary services to the dependent class, thus making the system more cohesive. The class that depends on the services is called the client class and the class which provides the service is called the service class [4]. Dependency Injection can be implemented by injecting the services through constructor (Constructor method), by creating a setter (Setter method) method or by implementing the interface (Interface method) of the client class [5],[6].

III. EXCEPTION HANDLING

Errors and exceptions are the events that occur during the execution of a program that can cause disturbance in the flow of a program [7]. To avoid such disturbance exception handling is used. It is a means to improve the maintainability and reliability of programs that face some errors or exceptions. Exception handling techniques are used to handle any disruption that takes place during the execution of a program. Many programming languages provide built-in exception handling mechanisms. The user can define and implement their exceptions by using try and except commands which are also called as user defined exceptions. Since Python is an interpreter it's exception handling mechanism helps in solving errors line by line which helps in avoiding such errors. The Python interpreter has built-in error and exception handlers that catch and handle exceptions. There are mainly two kinds of errors, namely syntax errors, and exceptions [8]. Syntax errors are the errors that occur when a syntactical mistake is committed. Exceptions are the errors that occur during the execution of a program. In Python, the handling of exceptions can be implemented using try and except clauses. If the code in the try block gets executed successfully, then the except block is skipped, but if an exception occurs during execution of the try block, then the exception is raised and executed.

IV. UNHANDLED EXCEPTIONS

While executing a program there are some exceptions that arise which does not belong to the built-in exception handling mechanism. These exceptions can be called Unhandled Exceptions[9]. The below listed are some of the examples of Unhandled Exceptions in Python 3.

A. Sort Error

In Python, sort is a predefined function that sorts a list. The sort function sorts list and stores the sorted list in the same list. Hence, when a user tries to assign the sorting of list to another variable, the variable used will return 'None' as the output.

B. Function Call Error

While programming if a function is defined but never called in the code, this can result in improper results or sometimes no results.

C. Reverse Error

In Python, reverse is a predefined function that reverses a list. The reverse function reverses the given list and stores it in the same list. Hence, when a user tries to assign the reverse function to a list and stores it in another variable, the variable used will return 'None' as the output.

D. Equality Error ('is' or '==')

In Python, 'is' operator is used only if both variables point to the same object and hence cannot be used for checking if the two object values are identical. To check if the values of two objects are the same we make use of the "==" operator.

E. Arg Error

In Python, an optional argument is an argument having some default value. In some case the optional argument may not be set to the specified default expression every time the function is called without supplying a value for the optional argument.

F. Return Error

While programming if a function or method is defined it is necessary to return the result of the function. When a function does not return anything, the interpreter return a 'None' value.

G. Iterable Error

In Python 3, when a user wants to run a for loop the user needs to specify the range for the loop to continue which can be a list, a string, a tuple or a dictionary. There is another method where the range function is applied for the length of a list, a string, a tuple or even a dictionary. When the length function is applied on an object without the range function the compiler will give result as " 'int' object is not iterable".

H. Unboundlocal Error

In Python, when a variable is assigned some value globally i.e., outside a function or a method, then that variable is called as a global variable. If the user wants to access the global variable inside a function or a method the user needs to specify the variable inside the function using the keyword global. When the user refers the global variable inside a function without using 'global' keyword the interpreter will give Unboundlocal Error.

I. __init__ Error

In Python, when a class is created it should have a __init__() method which is the constructor of that class. The __init__ method will be called every time an object is created for the same class. It is used to initialize the attributes of the class. When a user tries to return some value from the __init__ method, the interpreter will give a TypeError stating '__init__() should return None, not 'int' '.

J. Counter Error

Counter is a predefined python function that counts and returns the occurrence of all the elements of an object that is passed as a parameter to the function. It returns the count in a dictionary format where the user can check the number of times a specific element has occurred. When a user tries to pass the specific element from an object as a parameter to the Counter the interpreter will give a TypeError stating ' 'type' object is not subscriptable'.

K. Xrange Error

In Python 2, the xrange function is used to set a iteration range for the for loop, whereas in Python 3 the range function is used to set a iteration range for the for loop.

L. Except Error

In Python 2, multiple exceptions can be enclosed inside parentheses and they can be defined using a variable by making use of a comma(,) , whereas in Python 3 multiple exceptions can be enclosed inside parentheses and they can be defined using a variable by making use of 'as' keyword for reference.

M. Print Error

In Python 2, a value or a string can be printed using the print command, whereas in Python 3 a value or a string can be printed using the print command and the value or string enclosed in parenthesis '()'.

N. *Input Error*

In a block of code when there are a number of inputs required and a user forgets to provide even a single input which is needed in further code, then there are chances of the code being stuck in infinite loop or it gives the wrong result.

O. *Wrong Source Scrapped Error*

Python 3 supports Web Scrapping where the user can scrap various type of content from an open website. When an image is scrapped from an open website, it may contain 1 or more sources, when a source that does not contain '.jpg' or '.png' extension is scrapped. The interpreter will give no results.

P. *File not found Error*

In Python 3, a user can open and read any type of file using the specific commands. File not found is predefined error which occurs when a file is not found in the working directory. Instead, the interpreter must give the name of the files which are similar to the required file.

V. USE OF DEPENDENCY INJECTION IN EXCEPTION HANDLING

When an exception occurs the built-in exception, handler tries to handle it. When we add user defined exceptions in our system, this built-in exception class is extended to add new exceptions to it. In case of object-oriented programming language these exceptions are added in the form of classes. While adding more than one new exception class to our system Dependency injection can be used to decide which exception as a service to be called. The figure below is a diagrammatic representation of use of Dependency Injection in error handling.

The injector class would inject the necessary error as a service to the client class. The exception handling class is further extended to different error classes.

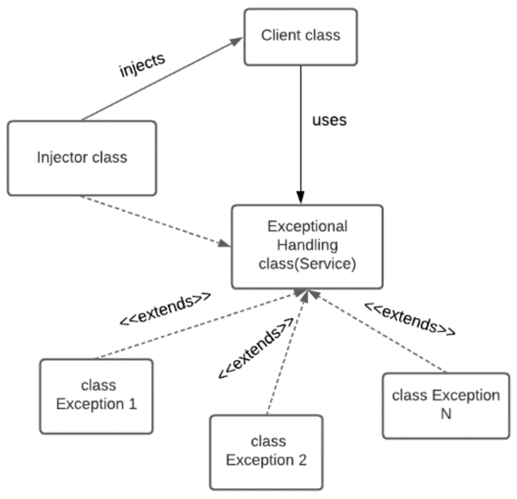

Fig. 1. Exception Handling using Dependency Injection

VI. EVALUATION

Testability is an important factor in any software design process. If there exists dependencies then it proves to be a great discomfort in testing. It is a well-known fact that unit testing in high coupled classes is likely to be inefficient. We have metric which helps us measure testability score for each class.[10]

This can be given by the equation.[10]

$$T(C) = \frac{1+5C_f+4C_j}{1+5C_n \cdot Log_2(C_n+1)\cdot(C_sN)!} \cdot \prod_{D\in C} T(D)$$

C_f	Number of injected interface dependencies
C_j	Number of injected non-interface dependencies
C_s	Number of static-state dependencies for C
C_n	Total number of dependencies
N	Total number of classes in the SUT

Fig. 2. Equation for calculating testability score

Below is the table which gives us the testability score for our main classes without injecting the dependency injection.

Class	T(C)	Cf	Cj	Cs	Cn	T(d)	Dep.
ErrorInfoProvider	0.0094	0.00	0.00	0.00	7.00	1.00	Sort Error Function Call Error Reverse Error Iterable Error Print Error Range Error Except Error
Sort Error	1.00	0.00	0.00	0.00	0.00	1.00	-
Function Call Error	1.00	0.00	0.00	0.00	0.00	1.00	-
Reverse Error	1.00	0.00	0.00	0.00	0.00	1.00	-
Iterable Error	1.00	0.00	0.00	0.00	0.00	1.00	-
Print Error	1.00	0.00	0.00	0.00	0.00	1.00	-
Range Error	1.00	0.00	0.00	0.00	0.00	1.00	-
Except Error	1.00	0.00	0.00	0.00	0.00	1.00	-

Fig. 3. Testability Score without Dependency Injection

Below is the table which gives us the testability score for our main classes with injecting the dependency injection.

Class	T(C)	Cf	Cj	Cs	Cn	T(d)	Dep.
ErrorInfoProvider	0.274	0.00	7.00	0.00	7.00	1.00	Sort Error Function Call Error Reverse Error Iterable Error Print Error Range Error Except Error
Sort Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)
Function Call Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)
Reverse Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)
Iterable Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)
Print Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)
Range Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)
Except Error	1.00	0.00	0.00	0.00	0.00	1.00	Error(Abstract)

Fig. 4. Testability Score with Dependency Injection

By comparing both the scores we can see a major increase in the score which has Dependency Injection. Higher the testability score, the more easily we can test that class [10]. Since DI is used to ensure loose coupling in any system thus it proves to be an improvement in testability.

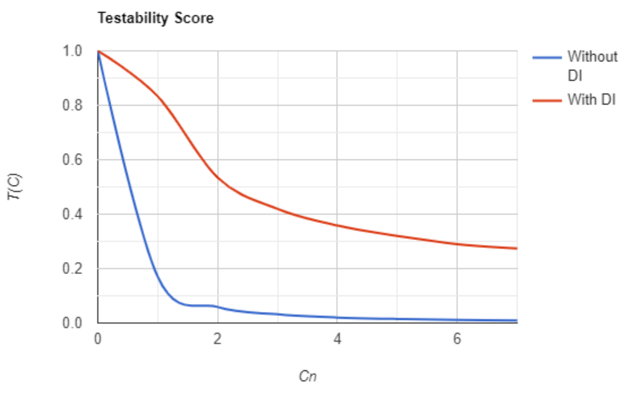

Fig. 5. Testability Score

VII. RESULT

A. *With Dependency Injection*

The above problem of tight coupling can be resolved using Dependency Injection. Each Exception class act as a different Service class to the ErrorInfoProvider class (Client class). These services are injected to the ErrorInfoProvider class using Dependency Injection. Fig. 3, is an implementation of Exception Handling using Constructor method of Dependency Injection. The unhandled exception classes implement the abstract method 'info()'. The ErrorInfoProvider class has a constructor which takes the parameter of the abstract class. Therefore, we can say that any object of the Unhandled Exception classes can be passed to the constructor of the ErrorInfoProvider class.

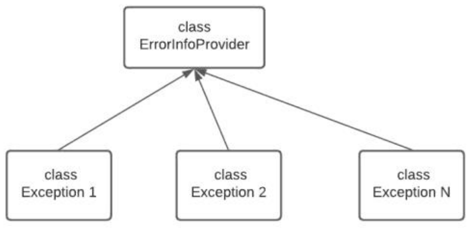

Fig. 6. Implementation without using Dependency Injection

B. *With Dependency Injection*

The above problem of tight coupling can be resolved using Dependency Injection. Each Exception class act as a different Service class to the ErrorInfoProvider class (Client class). These services are injected to the ErrorInfoProvider class using Dependency Injection. Fig. 3, is an implementation of Exception Handling using Constructor method of Dependency Injection. The unhandled exception classes implement the abstract method 'info()'. The ErrorInfoProvider class has a constructor which takes the parameter of the abstract class. Therefore, we can say that any object of the Unhandled Exception classes can be passed to the constructor of the ErrorInfoProvider class.

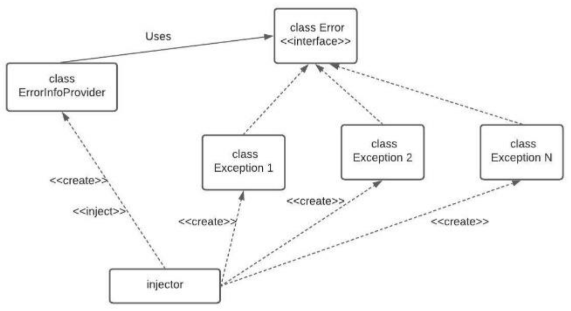

Fig. 7. Implementation with using Dependency Injection

VIII. CONCLUSIONS

Exception handling is a very important part of a system. Having a good exception handling mechanism helps the programmers to focus more on the logical requirement of the model rather than worrying about the exceptions. While most programming languages have a well-defined exception handling mechanism adding a few unhandled exceptions to the system can help programmers to solve their errors more quickly and easily. Above are few examples of Unhandled Exceptions in Python, more such exceptions can be found and added to the system. Dependency injection is used to handle these Unhandled Exceptions. Since dependency injection helps to make the system more robust by providing features like good flexibility, easier maintenance, using dependency injection to handle exceptions in our system would help increase the robustness of the system.

ACKNOWLEDGMENT

We pay our deep sense of gratitude to our guide Prof. Sumedh Pundkar who encouraged and guided us throughout the project. Our sincere thanks to our Principal and The Head of the Computer Science and Technology Department Dr. Sanjay Pawar for setting us a platform to explore. We also thank all the staff, library and laboratory for co-ordinating with us patiently. Last but not the least, sincere thanks to all our family and friends who shared their support morally.

REFERENCES

[1] K. Ježek, L. Holý and P. Brada, "Dependency injection refined by extra-functional properties - IEEE Conference Publication", Ieeexplore.ieee.org, 2012. [Online]. Available: https://ieeexplore.ieee.org/document/6344541. [Accessed: 09- Dec-2020].
[2] S. Hudli and R. Hudli, "A Verification Strategy for Dependency Injection", Pdfs.semanticscholar.org, 2013. [Online]. Available: https://pdfs.semanticscholar.org/e4eb/61b3c9194bb30c01ff8c9440ca94194a3309.pdf. [Accessed: 05- Dec- 2020].
[3] H. Melton, H. Yul Yang; and E. Tempero, "An Empirical Study into Use of Dependency Injection in Java", 2008. Available: https://doi.org/10.1109/ASWEC.2008.4483212 [Accessed 9 December 2020].
[4] "Dependency Injection", Tutorialsteacher.com, 2020. [Online]. Available: https://www.tutorialsteacher.com/ioc/dependency-injection. [Accessed: 09- Dec- 2020].
[5] "Inversion of Control Containers and the Dependency Injection pattern", martinfowler.com, 2004. [Online]. Available: https://martinfowler.com/articles/injection.html. [Accessed: 09- Dec-2020].
[6] S. Roubtsov, A. Serebrenik and M. van den Brand, "Detecting Modularity "Smells" in Dependencies Injected with Java Annotations -

IEEE Conference Publication", Ieeexplore.ieee.org, 2011. [Online]. Available: https://ieeexplore.ieee.org/document/5714443. [Accessed: 09- Dec- 2020].

[7] K. Schneider, C. Roy, M. Asaduzzaman and M. Ahasanuzzaman, "How Developers Use Exception Handling in Java? - IEEE Conference Publication", Ieeexplore.ieee.org, 2017. [Online]. Available: https://ieeexplore.ieee.org/document/7832939. [Accessed: 26- Dec- 2020].

[8] Docs.python.org. 2021. 8. Errors and Exceptions — Python 3.9.4 documentation. [online] Available at: <https://docs.python.org/3/tutorial/errors.html> [Accessed 8 December 2020].

[9] M. Gadge and S. Pundkar, "Undefined Exception Handling in Java using Dependency Injection or Inversion of Control".

[10] A. Lundberg, "Dependency Injection frameworks: an improvement to testability ?", Semanticscholar.org, 2015. [Online]. Available: https://www.semanticscholar.org/paper/Dependency-Injection-frameworks%3A-an-improvement-to-Lundberg/05d68651acb13d18d9e5b295aa9da3e3afca8641. [Accessed: 10- Dec- 2020].