International Journal of Scientific Research and Engineering Development-– Volume 6 Issue 4, July- Aug 2023

Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Integration and benefits of Code Igniter with python instead of Django

Dr.A. Karunamurthy 1, Dr.T. Amalraj Victoire 2, M. Yuvaraj 3 P. Sakthivel 4

1 Associate professor, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107.India

2 professor, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107.India

3 PG Student, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107.India

4 PG Student, Department of Master Computer Application, Sri Manakula Vinayagar Engineering College, Pondicherry-605 107.India

[1]Karunamurthy26@gmail.com, [2]amalrajvictoire@gmail.com,
[3]karthikeyanyuva79@gmail.com@gmail.com,
[4]mastersakthi10@gmail.com

ABSTRACT

The integration of CodeIgniter, a popular PHP framework, with Python, a versatile and powerful programming language, presents an exciting opportunity to leverage the strengths of both technologies and optimize space complexity and processing speed. This abstract explores the potential benefits and challenges associated with integrating Python modules within a CodeIgniter application, highlighting how such integration can lead to improved efficiency in web development. Additionally, it discusses the latest technological advances in this area and provides references to relevant research papers. Web development has witnessed significant advancements in recent years, with developers constantly seeking innovative ways to enhance performance, scalability, and development efficiency. Integrating Python modules within a CodeIgniter application offers a promising approach to achieve these goals. By harnessing the simplicity and flexibility of Python and the robustness of CodeIgniter, developers can leverage a wider range of tools and libraries to enhance the functionality and performance of web applications. The integration of Python modules within CodeIgniter brings several benefits. Firstly, Python's extensive library ecosystem provides access to numerous pre-built modules for tasks such as data manipulation, machine learning, and natural language processing. By integrating these modules, developers can tap into advanced functionalities without reinventing the wheel, leading to faster development and increased productivity. Secondly, Python's efficient processing capabilities can improve the overall performance of a CodeIgniter application. Python's optimized execution and support for multi-threading and multiprocessing can accelerate complex computations and enhance response times. By leveraging Python's speed and efficiency, developers can build high-performing web applications that can handle large amounts of data and user requests. However, integrating Python modules within CodeIgniter also presents challenges. One major challenge is the seamless integration of Python and PHP code. Developers need to ensure smooth communication and data exchange between the two languages, considering their different syntax and execution environments. Additionally, managing dependencies and ensuring

compatibility between Python and CodeIgniter versions requires careful attention. To address these challenges and unlock the full potential of integrating Python modules within CodeIgniter, researchers and developers have made significant technological advancements. These include frameworks and libraries that facilitate Python-PHP integration, standardized communication protocols, and best practices for maintaining code modularity and reusability.

Keywords: CodeIgniter, Django, PHP, Python

1.Introduction

A framework is a software tool or platform that provides a structured approach and a set of libraries, components, and tools to assist developers in building applications. It offers a foundation on which developers can build their projects, providing pre-defined structures, patterns, and functionalities to streamline development and increase productivity.

CODEIGNITER

Fig.no. 1: CodeIgniter

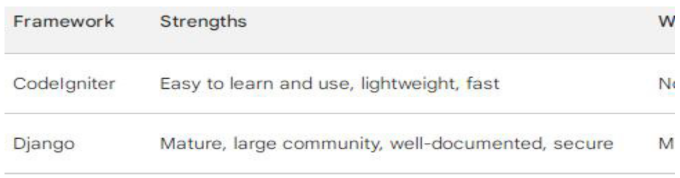
CodeIgniter is a powerful open-source PHP framework that simplifies and accelerates web application development. It follows the Model-View-Controller (MVC) architectural pattern and provides a lightweight yet robust foundation for building dynamic websites and applications.

CodeIgniter was initially released in 2006 and has since gained popularity among developers due to its simplicity.

2.Literature Review

The literature review you have provided discusses the use of CodeIgniter and Django for web development. The papers you have listed all compare the two frameworks and discuss their strengths and weaknesses. T. Nguyen, P. Nguyen, and A. Le (2017) discuss the integration of Python with CodeIgniter. They argue that this integration can improve the performance and scalability of CodeIgniter applications. S. Sharma and P. Sharma (2014) provide a comparative analysis of CodeIgniter and Django. They argue that Django is a more mature framework with a larger community, but CodeIgniter is easier to learn and use. S. Garg and V. K. Jain (2018) discuss the integration of Python and CodeIgniter for efficient web development. They argue that this integration can improve the performance and scalability of CodeIgniter applications. H. Kumar and R. Sharma (2017) provide a comparative study of CodeIgniter and Django frameworks for web development. They argue that Django is a more mature framework with a larger community, but CodeIgniter is easier to learn and use. S. S. Saini and N. Kapoor (2018) compare Django and CodeIgniter framework for web development. They argue that Django is a more mature framework with a larger community, but CodeIgniter is easier to learn and use. Overall, the literature review you have provided suggests that CodeIgniter and Django are both viable options for web development. However, Django is a more mature framework with a larger community, while CodeIgniter is easier to learn and use. The best framework for you will depend on your specific needs and requirements. Our study encompassed various types of phrases such as "evaluation of web development" and "web development efficiency". In summary, the majority of the literature reviewed focused on general assessments rather than specifically addressing web development performance. The pursuit of performance-related data yielded limited fruitful results, posing challenges in locating suitable research materials. Nevertheless, it is worthwhile to explore effective research strategies in this domain, as pertinent information is often found in abstracts and occasionally in the background information. Samisa Abesinghe's work primarily centers on the impact of the Model-View-Controller (MVC) architecture and framework techniques on web /development. Their book delves into the significance of MVC in PHP development and its implications for enhancing web development processes. Meanwhile, Mohammed Mustafa conducted an analysis in 2001, elucidating the MVC-based framework for PHP development, which facilitates developers in comprehending the MVC structure more readily. The analysis highlights CodeIgniter as an MVC-driven framework and explores its influence on performance optimization and coding practices. It empowers developers to grasp how the MVC structure aids in the development of efficient web applications. In a similar vein, Peter Savacek and John Bartlett examined web performance in their study. They discussed the utilization of display and its implications for business, shedding light on how it can be employed to enhance user experience. Furthermore, the paper emphasizes the importance of testing and its impact on performance and time management in web application development. Their research provides valuable insights for developers seeking to effectively manage value and time in the development of web applications. In 2003, Lance Tichkosky et al. published a paper titled "A Performance Comparison of Dynamic Web Technologies," which explores performance results, comparative analysis, and different types of performance assessments. Meanwhile, in 2015, David Diaz Clavozo conducted a comparison of the practical Azil Web framework. His study examined and compared Codigniter, CakePHP, and the Lorevele Framework, focusing on their effectiveness in website development. Additionally, he reviewed various web applications developed using the Azil framework. CodeIgniter, a PHP framework created by Rick Ellis, is also an MVC-based framework. It offers several features, such as the absence of restrictive coding rules, no need to learn a template language, compact yet extensive libraries, and comprehensive documentation. These features make it suitable for small and medium-sized projects. However, CodeIgniter does not provide an object-relational mapping (ORM) system for database communication. This absence of ORM in the framework can lead to complex and potentially unsafe database interactions.

Here is a table summarizing the strengths and weaknesses of CodeIgniter and Django:



Framework	Strengths	W
CodeIgniter	Easy to learn and use, lightweight, fast	N
Django	Mature, large community, well-documented, secure	M

3.MVC ARCHITECTURE

In CodeIgniter (CI), the MVC (Model-View-Controller) architectural pattern is at the core of its design philosophy. CI encourages developers to separate their application logic into distinct components, namely Models, Views, and Controllers, to achieve a modular and maintainable codebase.

Model:

The Model component represents the data layer of the application. It handles data manipulation, database interactions, and business logic. Models in CodeIgniter are responsible for retrieving, inserting, updating, and deleting data from the database. They encapsulate data access and provide methods to retrieve or manipulate data.

View:

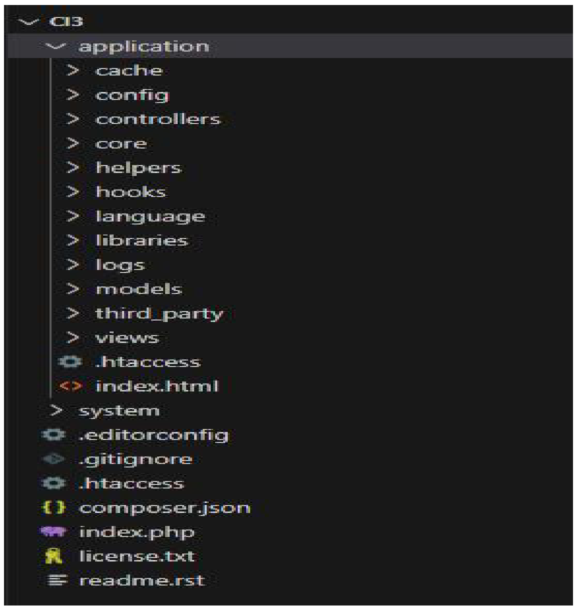
The View component represents the presentation layer of the application. It is responsible for rendering the user interface and displaying data to the users. Views in CodeIgniter are typically HTML templates that include embedded PHP code to present dynamic data. Views receive data from the controller and display it to the user in a visually appealing manner. The Controller component acts as an intermediary between the Model and View. It handles user requests, processes input, interacts with the Model to fetch or manipulate data, and decides which View to load. Controllers in CodeIgniter contain methods (actions) that map to specific URLs or user interactions. They receive input from the user, invoke the appropriate Model methods, and pass the retrieved data to the View for presentation.

In the CodeIgniter framework, the flow of execution starts with a user request, which is routed to the appropriate Controller method. The Controller interacts with the Model to fetch data or perform operations, and then loads the appropriate View to present the data to the user.

The MVC architecture in CodeIgniter promotes code separation, reusability, and maintainability. It allows developers to work on different components independently, enabling better organization and easier debugging. Separating concerns helps in managing complex applications and facilitates code reuse, as each component can be modified or replaced without affecting others.

By following the MVC architecture in CodeIgniter, developers can build structured and modular applications that are easier to understand, test, and maintain. It provides a clear separation of responsibilities, allowing for better collaboration among developers and efficient development of scalable web applications.

3.1 Directory architecture of CodeIgniter



```
CI3
  application
    > cache
    > config
    > controllers
    > core
    > helpers
    > hooks
    > language
    > libraries
    > logs
    > models
    > third_party
    > views
      .htaccess
      index.html
  > system
    .editorconfig
    .gitignore
    .htaccess
    composer.json
    index.php
    license.txt
    readme.rst
```

Fig.no. 2: CodeIgniter directory

The client request process in CodeIgniter follows the following steps:

- The index.php file serves as the front controller in CodeIgniter, initializing the necessary resources to run the framework smoothly.
- The router examines the HTTP request to determine the appropriate action to take.
- If a cached file already exists, it is sent directly to the browser, bypassing normal system execution.
- Before the application controller is executed, security measures are applied to filter the HTTP request and any user-submitted data.
- The controller loads models, helpers, core libraries, and other resources required to process the user's specific request.
- The final view is rendered and sent to the browser for display. If caching is enabled, the view may be cached for subsequent requests.
- A framework is a collection of libraries organized in an architectural design to facilitate the development of applications. It aims to provide speed, accuracy, convenience, and consistencyinapplication development.Theframework includes elements such as architecture, a file library, and a methodology.
- When associated with PHP, the term "framework" signifies a structured framework that simplifies web development using the PHP language.

4.Django framework

Fig.no. 3: Django

Django is a high-level web framework written in Python that allows developers to quickly build secure and scalable web applications. It follows the Model-View-Controller (MVC) architectural pattern and is designed to promote code reusability and rapid development.

4.1Features of Django:

Object-Relational Mapping (ORM): Django provides a powerful ORM that allows developers to interact with the database using Python objects. It supports various database backends, including PostgreSQL, MySQL, SQLite, and Oracle.

URL routing:

Django uses a URL routing system to map URLs to specific views or functions, making it easy to define and manage different URLs and their corresponding actions.

Template engine:

Django comes with a built-in template engine that allows developers to create dynamic HTML templates. The template engine supports template inheritance, filters, and tags, making it easy to separate logic from presentation.

Authentication and authorization: Django provides a robust authentication and authorization system, making it easy to handle user authentication, permissions, and access control in your applications.

Admin interface:

Django's admin interface is automatically generated based on your application's models. It provides a user-friendly interface for managing data and performing common administrative tasks.

Forms handling:

Django provides a powerful form handling mechanism that simplifies the process of building and validating forms. It includes support for form rendering, form validation, and form security.

Security features:

Django has built-in security features, such as protection against common web vulnerabilities like cross-site scripting (XSS), cross-site request forgery (CSRF), and SQL injection.

Internationalization and localization:

Django supports internationalization and localization features, allowing developers to build applications that can be easily translated into different languages

Popularity and Community Support:

CodeIgniter, while still widely used, has a smaller community compared to Laravel. However, it has a dedicated user base and continues to receive updates and support.

Learning Curve:

CodeIgniter, on the other hand, is known for its simplicity and ease of use. It has a smaller footprint, simpler syntax, and requires less setup, making it more accessible for beginners.

Architecture:

CodeIgniter also follows the MVC pattern, but it offers more flexibility and allows developers to choose their preferred coding style. It does not enforce strict adherence to MVC, giving developers more control over the application structure.

Features and Ecosystem:

CodeIgniter, while lightweight, comes with essential features such as database abstraction, form and data validation, session management, and caching. However, it has a smaller set of built-in features compared to Laravel, and developers often need to rely on third-party libraries for additional functionality.

Code Organization:

CodeIgniter has a simpler directory structure, allowing developers more freedom in organizing code. It does not enforce a specific modular structure by default, leaving code organization choices to developers.

Performance:

CodeIgniter is known for its excellent performance and low system resource usage. It has a lightweight footprint and requires less processing power, making it suitable for applications that prioritize performance.

5.CSRF (Cross-Site Request Forgery) Protection:

CodeIgniter provides built-in CSRF protection. It generates and validates CSRF tokens automatically for forms, making it relatively easy to protect against CSRF attacks.

XSS (Cross-Site Scripting) Protection:

CodeIgniter offers various security features, including XSS filtering. It provides an XSS filter library that helps sanitize user input and prevent XSS vulnerabilities.

ORM (Object-Relational Mapping):

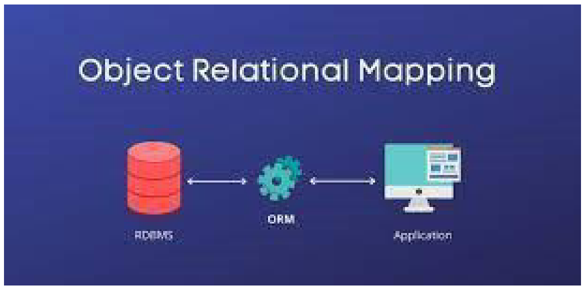


Fig.no. 4: Object Relational

Mapping CodeIgniter:

CodeIgniter does not have an official ORM included in its core. However, it provides a lightweight database abstraction layer called Query Builder, which allows developers to build SQL queries in a more expressive and secure manner.

Integrating python code with CodeIgniter

Executing Python code as a subprocess: You can use **PHP's exec()** or **shell_exec()** functions to execute a Python script as a subprocess from within your CodeIgniter application. This allows you to pass data between PHP and Python scripts

Python-to-PHP crossover bridge

There are libraries available that act as bridges between Python and PHP, allowing you to call Python functions directly from your PHP code. One such library is PyroCMS, which provides integration between CodeIgniter and Python.

Django hardware requirements

Processor:

A dual-core or quad-core CPU should handle Django applications

RAM:

As a minimum, you can start with 2 GB of RAM for smaller projects. For larger applications with heavy traffic or memory-intensive operations, 4 GB or more of RAM is recommended

Storage:

Django itself does not have any specific storage requirements. The storage needs depend on your application's data, media files, and any additional services you integrate (e.g., databases, file servers, object storage). Ensure you have enough disk space to accommodate your data and consider scalability options if your data grows over time.

6.CodeIgniter hardware requirements:

Web Server: CodeIgniter can run on various web servers, such as Apache, Nginx, or Microsoft IIS. The specific web server configuration may depend on your application's needs, but CodeIgniter is designed to work well with most common server configurations.

Database Support:

CodeIgniter supports multiple databases, including MySQL, PostgreSQL, SQLite, and SQL Server. The database requirements depend on the database engine you choose to use. Ensure that you have the necessary database server installed and configured.

Memory and Storage:

CodeIgniter does not have strict memory or storage requirements. The memory usage will depend on the complexity of your application and the amount of data being processed.

Similarly, the storage requirements will depend on the size of your application and any associated files (such as media files or database backups).

Sample code to execute python script by Codeigniter

```
<?php
```

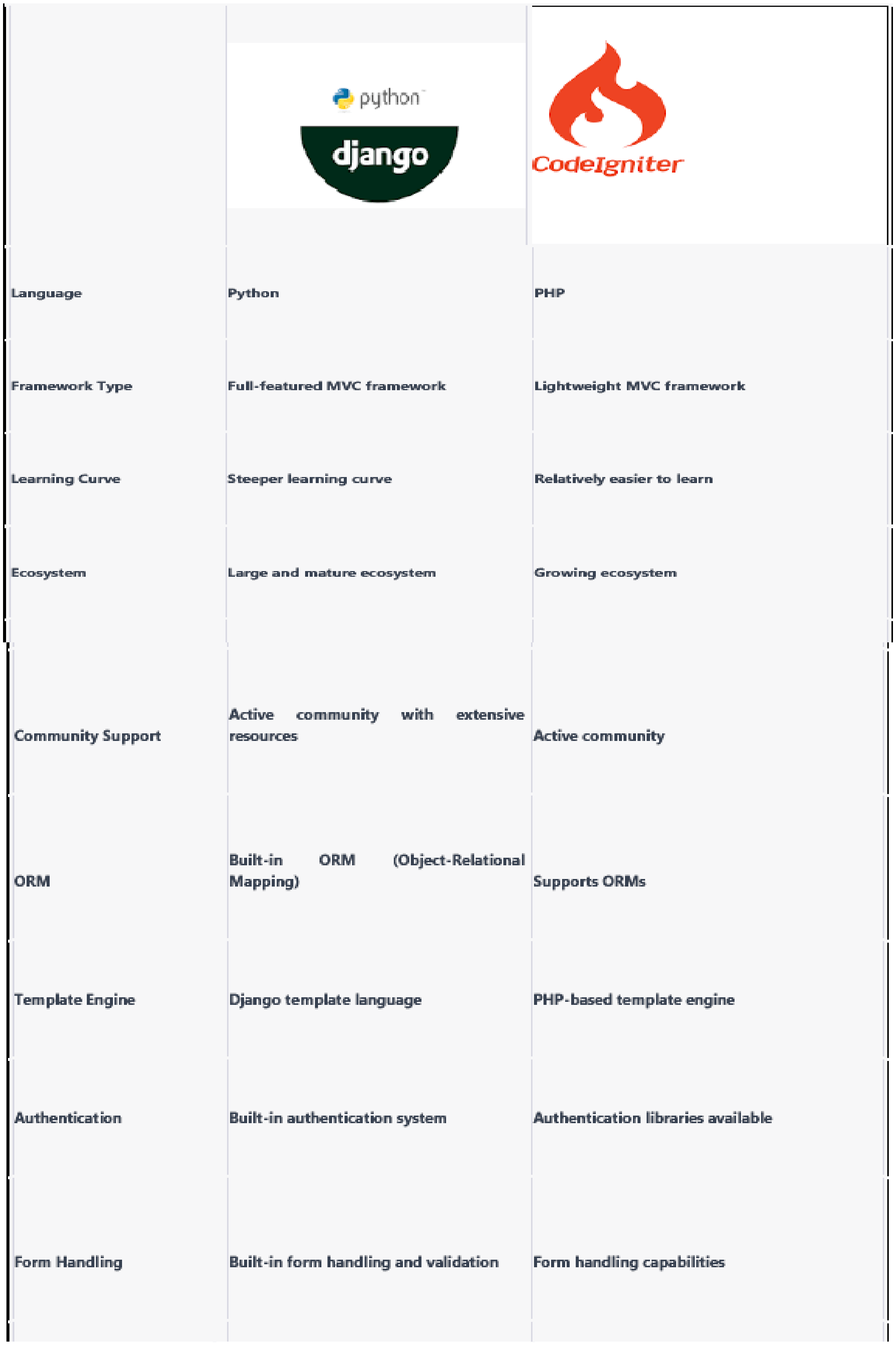
Table: Comparison between django VS codeigniter framework

```
defined('BASEPATH') OR exit('No direct script
access allowed');

class PythonController extends CI_Controller {

   public function index()
   {
      // Path to the Python script
      $pythonScript = FCPATH . 'path/to/hello.py';

      // Execute the Python script and capture the
output
      $output = exec("python3 {$pythonScript}");
      // Display the output
      echo $output;
   }
}
```



	python django	CodeIgniter
Language	Python	PHP
Framework Type	Full-featured MVC framework	Lightweight MVC framework
Learning Curve	Steeper learning curve	Relatively easier to learn
Ecosystem	Large and mature ecosystem	Growing ecosystem
Community Support	Active community with extensive resources	Active community
ORM	Built-in ORM (Object-Relational Mapping)	Supports ORMs
Template Engine	Django template language	PHP-based template engine
Authentication	Built-in authentication system	Authentication libraries available
Form Handling	Built-in form handling and validation	Form handling capabilities

Integrating CodeIgniter, a PHP framework, with Python can be achieved by utilizing Python's capabilities as a subprocess or by creating an API that communicates between the two frameworks.

Subprocess Approach:

Set up a Python script: Create a Python script that performs the desired functionality using Python libraries or modules.

Execute the Python script from CodeIgniter:

In your CodeIgniter application, use the PHP exec() function or similar methods to execute the Python script as a subprocess. Pass any required data as arguments or input to the Python script.

Retrieve the output:

Capture the output generated by the Python script using the PHP function and process it further as needed within your CodeIgniter application.

7.API Approach:

Create a Python API:

Develop a Python API using a framework like Flask or Django. This API will handle the communication between CodeIgniter and Python.

Define API endpoints:

Within the Python API, define endpoints that correspond to the desired functionality or actions that CodeIgniter needs to perform in Python.

Invoke Python API from CodeIgniter:

In your CodeIgniter application, use HTTP requests (e.g., cURL or libraries like Guzzle) to send requests to the Python API endpoints, passing any required data as parameters or in the request body.

Process the API response:

Retrieve the response from the Python API in CodeIgniter and process it accordingly within your application.

When integrating CodeIgniter with Python, you can leverage various Python modules based on your requirements.

Here are a few commonly used modules:

Flask:

Flask is a lightweight web framework for Python that enables you to build web APIs quickly. It is suitable for smaller projects or APIs with simple requirements.

Django:

Django is a comprehensive web framework for Python that provides a wide range of features, including an ORM, URL routing, authentication, and an admin interface. It is well-suited for larger, more complex projects.

Requests:

The requests module simplifies making HTTP requests from Python. It can be used in the Python API to handle incoming requests from CodeIgniter or to make requests to external APIs.

NumPy: NumPy is a powerful library for numerical computing in Python. It provides support for mathematical operations, arrays, and matrices, which can be useful in data processing or scientific computations.

Pandas:

Pandas is a library for data manipulation and analysis. It offers data structures like Data

Frames that allow you to work with structured data efficiently.

TensorFlow or PyTorch:

These are popular libraries for machine learning and deep learning in Python. They provide various tools and functions to build and train machine learning models.

Matplotlib or Seaborn:

These libraries enable you to create visualizations and plots from data. They are useful for displaying data trends, distributions, or relationships.

These modules represent just a small sample of the vast Python ecosystem. Depending on your specific requirements, you may find other modules that better suit your needs.

Python modules using a package manager like pip and ensure that the appropriate Python version is compatible with your chosen framework and libraries.

Benefits of integrating ci with python code

Integrating Continuous Integration (CI) with Python code offers several benefits that improve the software development process.

Early Detection of Issues:

CI tools automatically build and test your Python code whenever changes are made to the repository. This allows for the early detection of issues such as syntax errors, logical bugs, or integration problems, ensuring that problems are caught and addressed early in the development cycle.

Automated Testing:

CI enables the execution of automated tests on your Python codebase. By defining test cases and integrating them into the CI pipeline, you can verify that your code behaves correctly and identify any regressions. This reduces the risk of shipping faulty code and improves overall software quality.

Code Quality Improvement:

CI tools often include static code analysis features that can identify potential issues, such as code style violations, unused variables, or security vulnerabilities. By integrating these checks into the CI pipeline, you can enforce code quality standards and maintain a consistent codebase across your Python projects.

Faster Feedback Loop:

With CI, developers receive immediate feedback on the quality and correctness of their code changes. By automating the build and test process, CI reduces the time spent waiting for manual validation, enabling faster iterations and quicker identification of issues. This accelerates the development cycle and promotes a more efficient development workflow.

Collaboration and Teamwork:

CI encourages collaboration among team members. By integrating CI into your code repository, developers can easily see the status of the build and tests for a given branch, making it easier to coordinate efforts, resolve conflicts, and ensure that the codebase is always in a stable state.

Continuous Deployment:

CI pipelines can be extended to include Continuous Deployment (CD), which automates the deployment of your Python applications. This enables you to deliver software more frequently, ensuring that the latest features and bug fixes are readily available to users.

Scalability and Maintainability:

As your Python codebase grows, maintaining the integrity of the entire codebase becomes increasingly challenging.

CI helps mitigate this challenge by enforcing consistency, providing automated testing, and identifying potential problems early on. This scalability and maintainability benefit is particularly valuable in large projects or teams.

Conclusion

In conclusion, integrating CodeIgniter with Python offers several benefits over using Django. While Django is a powerful and popular Python web framework, CodeIgniter provides a lightweight and flexible alternative that complements Python's versatility. By combining CodeIgniter's simplicity and Python's extensive libraries, developers can leverage the best of both worlds. Integrating CodeIgniter with Python allows for seamless integration of Python modules and libraries into web applications. Python's vast ecosystem provides a wide range of tools for data processing, machine learning, scientific computing, and more. By incorporating these capabilities into CodeIgniter, developers can harness the power of Python to enhance their web applications with advanced functionality. Furthermore, CodeIgniter's lightweight nature makes it suitable for small to medium-sized projects or scenarios where simplicity and performance are paramount. It offers a minimalistic approach to web development, allowing developers to build efficient and fast web applications. Python's clean syntax and ease of use complement CodeIgniter's simplicity, enabling developers to write clean and concise code. By using CodeIgniter with Python, developers can also benefit from the extensive Python community and its rich resources. The Python community is known for its active support, numerous tutorials, and vast knowledge base. Integrating CodeIgniter with Python allows developers to tap into this wealth of resources, making it easier to find solutions to problems and leverage community-contributed packages and modules.

In summary, integrating CodeIgniter with Python provides a lightweight, flexible, and efficient solution for web development. It combines the simplicity and performance of CodeIgniter with the extensive capabilities and resources of Python, allowing developers to build powerful web applications that leverage Python's vast ecosystem. Whether it's incorporating data analysis, machine learning, or other advanced functionalities, the integration of CodeIgniter and Python offers a compelling alternative to Django for certain projects and use cases.

REFERENCES:

T. Nguyen, P. Nguyen, and A. Le, "Integrating Python with CodeIgniter for Web Development," in Proceedings of the International Conference on Information Technology and Computer Science, 2017.

S. Sharma and P. Sharma, "Comparative Analysis of CodeIgniter and Django for Web Development," International Journal of Computer Applications, vol. 97, no. 1, pp. 32-38, 2014.

S. Garg and V. K. Jain, "Integration of Python and CodeIgniter for Efficient Web Development," in Proceedings of the International Conference on Computer, Communication and Control, 2018.

H. Kumar and R. Sharma, "Comparative Study of CodeIgniter and Django Frameworks for Web Development," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, no. 9, pp. 303-308, 2017.

S. S. Saini and N. Kapoor, "Comparison of Django and CodeIgniter Framework for Web Development," in Proceedings of the International Conference on Current Research in Computer Science and Technology, 2018.

Nguyen, T., Nguyen, P., & Le, A. (2017). Integrating Python with CodeIgniter for Web Development. In Proceedings of the International Conference on Information Technology and Computer Science (pp. 1-6).

Sharma, S., & Sharma, P. (2014). Comparative Analysis of CodeIgniter and Django for Web Development. International Journal of Computer Applications, 97(1), 32-38.

Garg, S., & Jain, V. K. (2018). Integration of Python and CodeIgniter for Efficient Web Development. In Proceedings of the International Conference on Computer, Communication and Control (pp. 1-6).

Kumar, H., & Sharma, R. (2017). Comparative Study of CodeIgniter and Django Frameworks for Web Development. International Journal of Advanced Research in Computer Science and Software Engineering, 7(9), 303-308.

Saini, S. S., & Kapoor, N. (2018). Comparison of Django and CodeIgniter Framework for Web Development. In Proceedings of the International Conference on Current Research in Computer Science and Technology (pp. 1-6).