Research Article

Developing a Web Framework Based on Inversion of Control

Soner Tuğan[1], Enis Arslan[2*], Ali Murat Tiryaki[3]

[1] Canakkale Onsekiz Mart University Department of Computer Engineering, Orcid ID: https://orcid.org/0009-0002-1890-2339, E-mail: 1sonertugan1@gmail.com
[2] Canakkale Onsekiz Mart University Department of Computer Engineering, Orcid ID: https://orcid.org/0000-0002-2609-3925, E-mail: enisarslan@gmail.com
[3] Canakkale Onsekiz Mart University Department of Computer Engineering, Orcid ID: https://orcid.org/0000-0001-8224-6319, E-mail: tiryaki@comu.edu.tr

[*] Correspondence: enisarslan@gmail.com

Received 22 January 2024
Received in revised form 12 March 2024
In final form 25 May 2024

Reference: Tuğan, S., Arslan., E., Tiryaki, A., M. Developing a Web Framework Based on Inversion of Control. The European Journal of Research and Development, 4(2), 96-109.

Abstract

Increasing the reusability and manageability of software units is one of the most fundamental issues in the field of software engineering. To increase these parameters, which are important factors that determine the quality of the software, it is necessary to develop the units that make up the software in a way that minimizes their dependency on each other. This basic requirement exists in web applications, as in other types of software. Inversion of Control – IoC is a design pattern used to manage the components of a software application and combine them by reducing inter dependencies. This study focuses on the development of a web framework that includes IoC containers and HTTP services, which have an important role in the development of modern web applications. The developed framework has the abilities to automatically analyze dependencies and integrate components flexibly by creating its own IoC container. Thanks to these features, an infrastructure has been created where developers can modularize their codes and easily create sustainable software development projects. In addition, HTTP services developed within this framework provide the basic communication mechanisms that web applications need. These services contain the necessary functions to process and route HTTP requests and present the results. The framework aims to provide a solid foundation for future web projects by offering an innovative and original approach.

Keywords: Web based programming, framework development, HTTP, IoC pattern, software dependencies.

1. **Introduction**

One of the most fundamental issues in the field of software engineering is the search for methods and techniques that will enable the development of software systems in a more manageable and modifiable way [4]. In order to increase manageability in software systems and reduce the cost of change, it is necessary to minimize the interdependence of the units that make up the software. This is one of the most basic factors that determine the quality of a software. All development methods such as waterfall [8], Scrum (9), Rapid Application Development [10], spiral model [11], which have been introduced for different programming paradigms, focus on to develop quality software by keeping the dependencies between software units at a minimum level by using techniques such as aspect oriented programming [5], axiomatic design [6], refactoring [7].

With the rapid advancement of internet technologies, the importance of web-based applications in the software industry is gradually increasing [12][14]. While the majority of newly developed software systems are designed to be web-based, a lot of work is being done on making previously developed desktop applications web-based.

Web based applications often work with many calls to outside the application. Repetitive structures outside the application cause significant problems in the development process by reducing the manageability and testability of the application to be developed. In every web-based application, it is necessary to define what action should be done in which HTTP request. A modular infrastructure is needed to manage the calls of such applications [15], [16].

In the literature, there are some frameworks that can manage external calls of web-based applications developed with different technologies. Spring [1] for Java-based applications, Asp.net [2] for C#-based applications, Laravel [3], [13] for PHP-based applications are among the most well-known of these frameworks.

The aim of this study is to develop an open-source framework based on Inversion of Control – IoC pattern [17], [18] that will enable faster and easier development of PHP-based web applications. The developed framework aims to increase the manageability and reusability of web applications developed with PHP-based technologies by managing external calls such as HTTP requests with minimal dependencies. In the study, the modules were divided into sprints and carried out with requirements analysis, system designs and coding stages within the agile software development cycle.

The following sections of the article are organized as follows: In Section 2, the development method and implementation details of the new framework developed in the study are presented. The subsections of this section contain details about the two main modules of the framework. Section 3 includes a case study that demonstrates the use of the framework in the development of a sample application. Section 4 is the conclusion section where the study results are discussed.

2. Method

Dependency Injection refers to the production of other classes that a class needs, that is, on which it depends, and giving them to that class. With a more concrete explanation, let's assume that there are two classes named AController and BConnector and that AController class calls any method of the class named BConnector. In this case, BConnector must also be produced in order for AController to be produced. If BConnector is a class that can be easily created with coding practices, this class can be created directly in AController, or it can be created where AController is created and accessed with parameters. In large or growing projects, creating classes is often an increasingly complex task. In addition to this complexity, the domain information required to create a class begins to spread among the classes that use it, and in a short time, the necessary codes to create objects from this class are copied and pasted. Creative design patterns are normally used to produce classes without using the Dependency Injection method. In these methods, solutions are developed for a series of complex problems such as where objects will be produced and how they can be produced. Dependency Injection tries to abstract the information about how and where to create a class by taking the information necessary to define it. This will reduce time losses by reducing the creative problems caused by objects in the application to be developed.
With the software called PEL (Powerful Easy to Learn PHP Web Framework) prepared in this study, an infrastructure has been prepared for software developers to develop uncomplicated and rapid web applications. The developed PHP-based web framework basically consists of two main modules, the first of which is the module required for managing HTTP requests, and the second is the IoC Container module for managing class dependencies. These two modules work integrated with the Apache server along with the basic PHP infrastructure.
The structures of these modules are elements that are found in every web application and directly affect the development time of the application. The HTTP module in this framework includes the functions related to which requests will be answered and in what way, and the IoC Container includes the functions related to the production and management of objects from newly prepared and existing classes.

2.1. The HTTP Module

During the software development process, it is defined which function will be run in response to a request coming via the HTTP method for the endpoint that will perform a certain function. In addition to these definitions, it may also be requested to listen at dynamically specified end points. For example, in a sample endpoint such as /user/2332/comment/12323, the user can specify the 'user id' with the information he receives after 'user', and which 'comment' he wants with the information he receives after 'comment'. Here, it is unimportant that the numbers after 'user' and 'comment' change. For example, when an endpoint such as /user/:userId/comment/:commentId is defined as a definition, the point to consider is that it starts with /user, then receives information until /comment, and then receives information again after /comment. After the user defines which function will run when a request comes to the endpoints he has defined, he requests additional structures within this function that will make his job easier. These structures can be returned as a PDF, Image or XML information. For example, when the user makes a specification with Response::file("file-path"), the file type should be determined according to the file path given and this notification should be made automatically in the Response header.

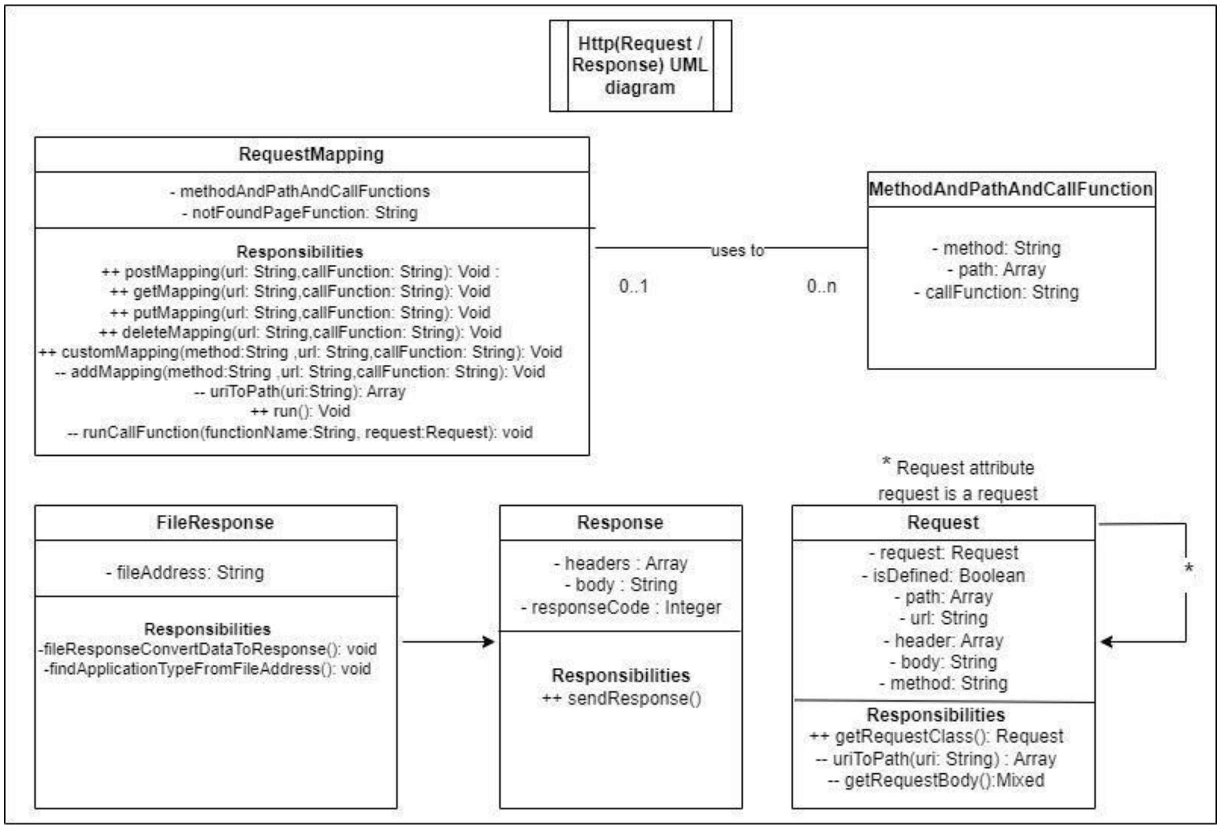

Http(Request / Response) UML diagram
RequestMapping
- methodAndPathAndCallFunctions
- notFoundPageFunction: String
Responsibilities
++ postMapping(url: String,callFunction: String): Void :
++ getMapping(url: String,callFunction: String): Void
++ putMapping(url: String,callFunction: String): Void
++ deleteMapping(url: String,callFunction: String): Void
++ customMapping(method:String ,url: String,callFunction: String): Void
-- addMapping(method:String ,url: String,callFunction: String): Void
-- uriToPath(uri:String): Array
++ run(): Void
-- runCallFunction(functionName:String, request:Request): void
uses to
0..1
0..n
MethodAndPathAndCallFunction
- method: String
- path: Array
- callFunction: String
* Request attribute request is a request
FileResponse
- fileAddress: String
Responsibilities
-fileResponseConvertDataToResponse(): void
-findApplicationTypeFromFileAddress(): void
Response
- headers : Array
- body : String
- responseCode : Integer
Responsibilities
++ sendResponse()
Request
- request: Request
- isDefined: Boolean
- path: Array
- url: String
- header: Array
- body: String
- method: String
Responsibilities
++ getRequestClass(): Request
-- uriToPath(uri: String) : Array
-- getRequestBody():Mixed
*

Figure 1. The class diagram for HTTP Module

The HTTP module has been developed to enable the software developers to define HTTP methods and define new URL-based requests in a dynamic structure, determine which function to run when requests for URLs arrive, and use existing structures for basic response needs when responding to incoming requests.

The class diagram for the developed HTTP module, including method prototypes, are given in Figure 1.
As shown in Figure 1, the RequestMapping Class is the main class that manages incoming requests. A Request class is used to understand the details of incoming requests to be used with this class, and Response is used to provide basic functions for the HTTP response to be returned.
The implementation details of the core two modules in the framework is explained in the following sub-sections. The repository containing the source code of the framework can be accessed from the following link: https://github.com/sonerrtgn/pel-framework

2.2. The IoC Container Module

A basic system architecture design was made for the developed IoC container to meet these requirements. This design is shown in Figure 2.

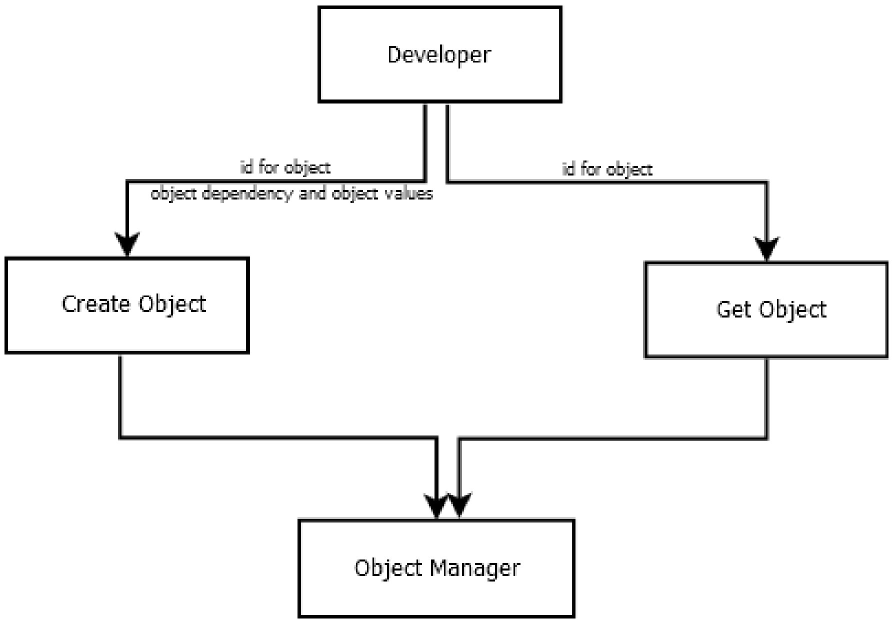


Figure 2. The basic architecture of the IoC Container Module

As shown in Figure 2, the Object Manager, which will manage the dependencies, is called with two structures. The first of these is Create Object, where new dependency information is given, while the other is Get Object, where dependency information is received from the dependency manager. The class designs made before coding to meet the system requirements and system design needs are given in Figure 3.

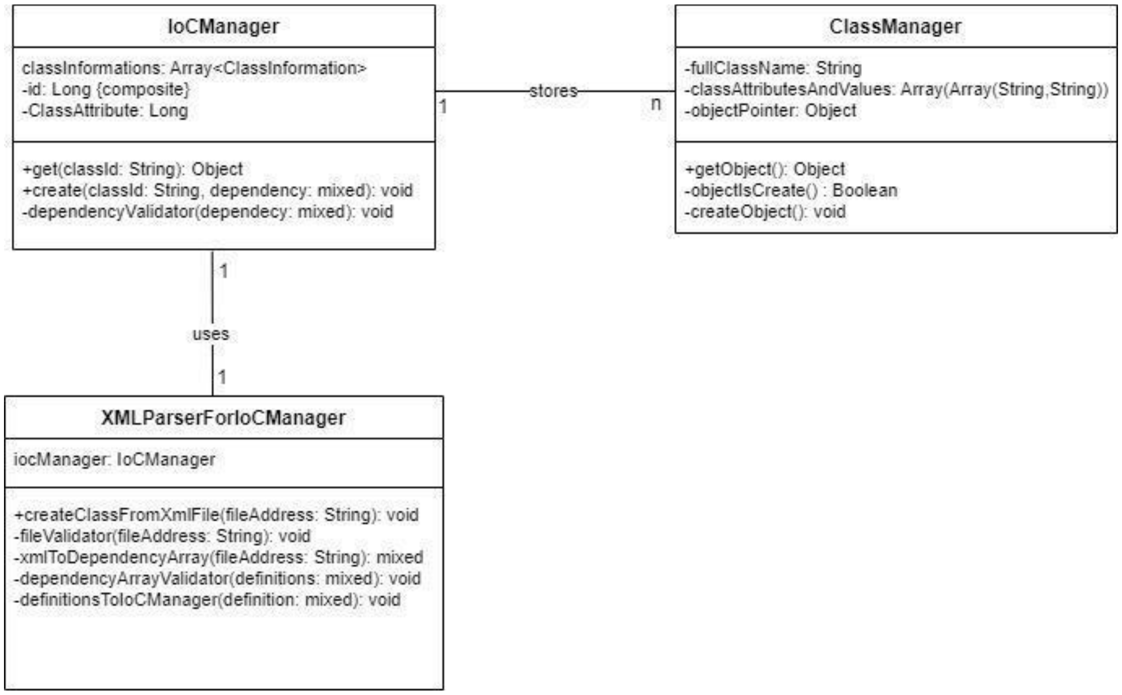
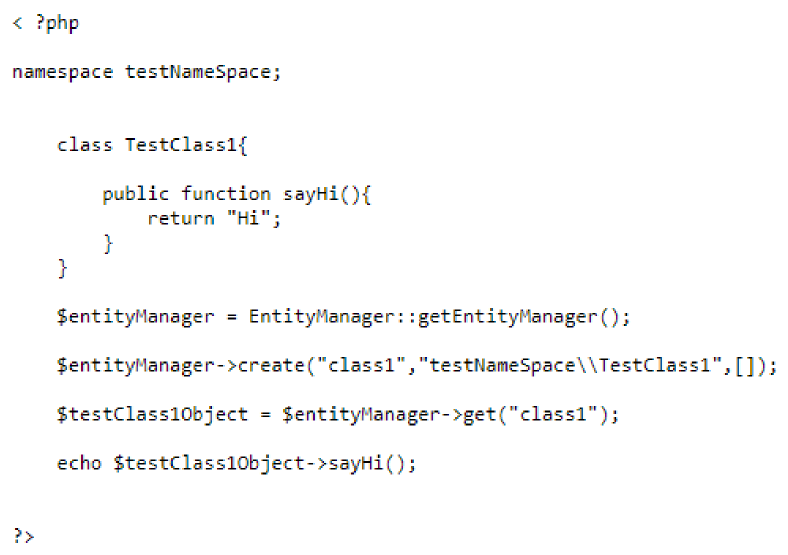

IoCManager
classInformations: Array<ClassInformation>
-id: Long {composite}
-ClassAttribute: Long
+get(classId: String): Object
+create(classId: String, dependency: mixed): void
-dependencyValidator(dependecy: mixed): void
ClassManager
-fullClassName: String
-classAttributesAndValues: Array(Array(String,String))
-objectPointer: Object
+getObject(): Object
-objectIsCreate() : Boolean
-createObject(): void
stores
1
n
uses
1
1
XMLParserForIoCManager
iocManager: IoCManager
+createClassFromXmlFile(fileAddress: String): void
-fileValidator(fileAddress: String): void
-xmlToDependencyArray(fileAddress: String): mixed
-dependencyArrayValidator(definitions: mixed): void
-definitionsToIoCManager(definition: mixed): void

Figure 3. The class diagram for the IoC Container Module

In Figure 3, the base class is IoCManager and two other classes ClassManager and XMLParserForIoCManager are managed by IoCManager class.

In order to define a class without dependencies in the module, it is first necessary to create a class without dependencies with the namespace information and the name of the class and give a unique id value to this dependency. If the class to be added to the module has a dependency, the name of the dependency and the uniq id value that creates this dependency must be shown for the dependency. Thus, each class can be created at least once and used in every call (Singleton pattern). In addition, this module can be monitored by the software developers in order to enable faster development, and report information can be requested from this module at certain intervals to see which class has been used and how many times.

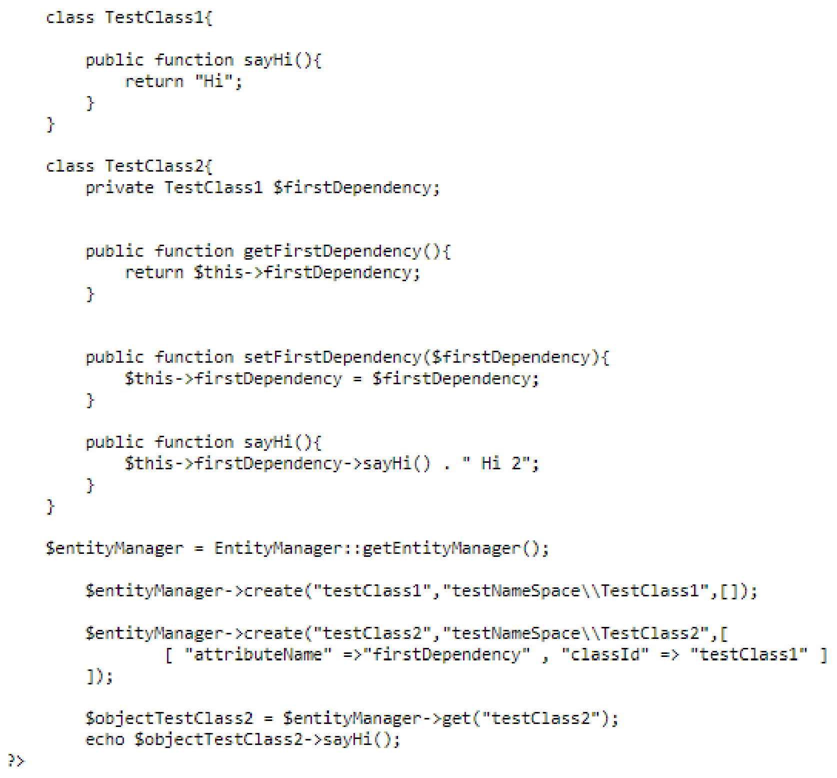
Class additions are made with the create keyword, namespace and class name information. If there are class dependencies, the definition is made by entering the name of the dependency in the class and the id of the dependency previously defined in the module with an array definition. To define the class that does not have a dependency, namespace and class name information is reported to the module. In addition, an empty string is sent, meaning that there is no dependency. If this array is not empty, class dependencies are defined in this array. If the added dependency has not been declared to the module before, the system stops working and IoC Container gives the error 'Trying to add unknown dependency'. Finally, this class added to the IoC Container is given a unique id. If this id has been used before, the IoC Container gives 'Repeat id error'. An example screen for class introduction without dependencies is given in Figure 4.



```
< ?php

namespace testNameSpace;


    class TestClass1{

        public function sayHi(){
            return "Hi";
        }
    }

    $entityManager = EntityManager::getEntityManager();

    $entityManager->create("class1","testNameSpace\\TestClass1",[]);

    $testClass1Object = $entityManager->get("class1");

    echo $testClass1Object->sayHi();


?>
```

Figure 4. A sample of class definition without dependencies in PEL framework

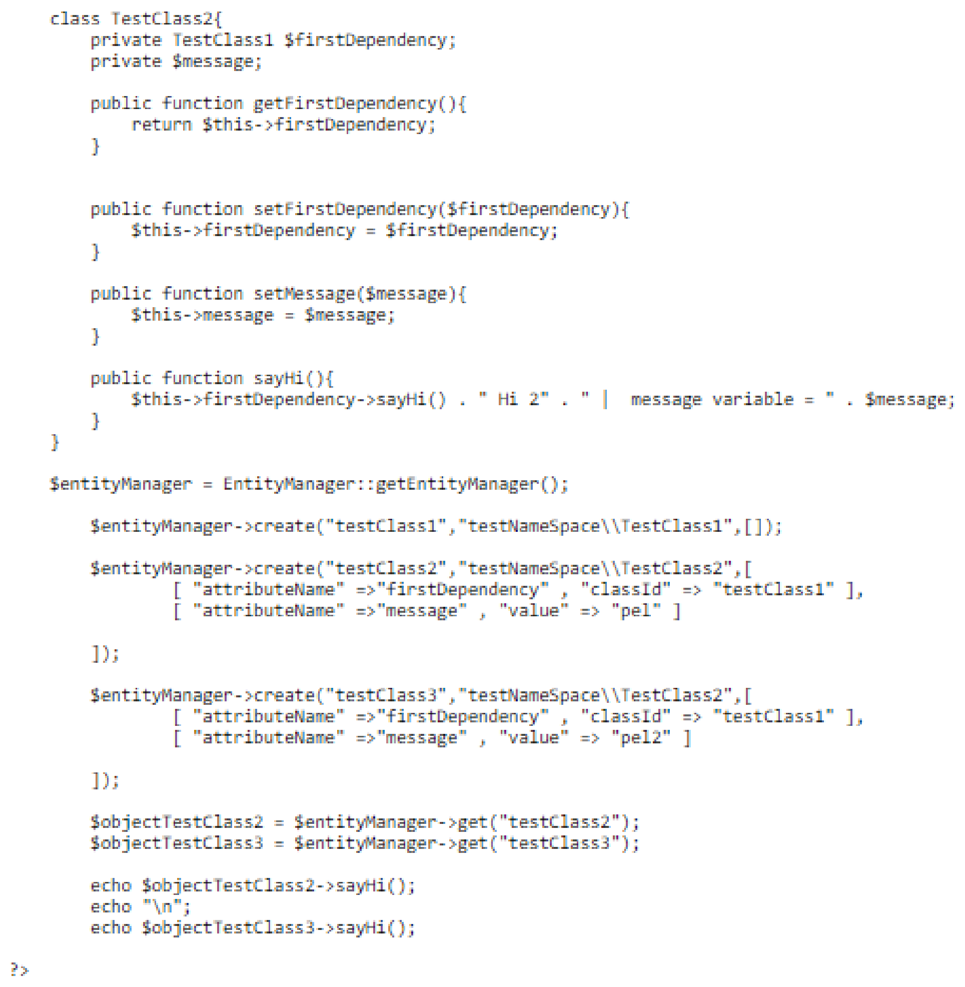
The example given in Figure 4 shows how an object can be created and retrieved with the create() and get() methods, respectively. First, a class named testClass1 was created. This class has a method called sayHi that returns "Hi" as a string. The EntityManager class is the base class that will be used to create classes. To get the object manager of this class, the EntityManager object can be obtained by calling the getEntityManager() method. Afterwards, a create() method is called to the entityManager object. This method shows how to define a new class into the IoC Container. Three parameters are passed to the create() method, the first is the uniq key to access the object to be created, the second is the full address of this class (namespace / class name), and the third is an array showing the dependencies of this class. Since the relevant class does not have a dependency, this section can be left blank. The get() method is used to retrieve an object from the entityManager and the uniq key value of the object to be retrieved is given as the value. As a result of the process so far, in order to use a method from the TestClass1 class, it is no longer necessary to create this class with the new keyword.
To create classes that are dependent on each other, it is actually necessary to create objects from the bottom up, that is, if class A uses B and to create class A, you need to create class B beforehand. An example screen for class introduction without dependencies is given in Figure 5.



```
    class TestClass1{

        public function sayHi(){
            return "Hi";
        }
    }

    class TestClass2{
        private TestClass1 $firstDependency;


        public function getFirstDependency(){
            return $this->firstDependency;
        }


        public function setFirstDependency($firstDependency){
            $this->firstDependency = $firstDependency;
        }

        public function sayHi(){
            $this->firstDependency->sayHi() . " Hi 2";
        }
    }

    $entityManager = EntityManager::getEntityManager();

        $entityManager->create("testClass1","testNameSpace\\TestClass1",[]);

        $entityManager->create("testClass2","testNameSpace\\TestClass2",[
                [ "attributeName" =>"firstDependency" , "classId" => "testClass1" ]
        ]);

        $objectTestClass2 = $entityManager->get("testClass2");
        echo $objectTestClass2->sayHi();
?>
```

Figure 5. A sample class definition with dependency to PEL framework

In the example given in Figure 5, a new class named TestClass2 was created in addition to the TestClass1 class. This class depends on and uses a class of type TestClass1. When introducing these classes to the EntityManager, first TestClass1 and then TestClass2 were added and their dependency was defined in the array in the last parameter of the create() method. Class dependencies are expressed as [["attributeName" => "firstDependency" , "classId" => "testClass1"]]. If there is more than one dependency, straight addition can be done within the array.
The created class can have dependencies in different ways. A sample code for such cases is given in Figure 6.
Let's assume that an attribute called 'message' is wanted to be added to TestClass in the code given in Figure 5. To give this value in the EntityManager, a variable named 'message' has been added to the TestClass2 class, as seen in Figure 6.



```
class TestClass2{
    private TestClass1 $firstDependency;
    private $message;

    public function getFirstDependency(){
        return $this->firstDependency;
    }


    public function setFirstDependency($firstDependency){
        $this->firstDependency = $firstDependency;
    }

    public function setMessage($message){
        $this->message = $message;
    }

    public function sayHi(){
        $this->firstDependency->sayHi() . " Hi 2" . " |  message variable = " . $message;
    }
}

$entityManager = EntityManager::getEntityManager();

    $entityManager->create("testClass1","testNameSpace\\TestClass1",[]);

    $entityManager->create("testClass2","testNameSpace\\TestClass2",[
            [ "attributeName" =>"firstDependency" , "classId" => "testClass1" ],
            [ "attributeName" =>"message" , "value" => "pel" ]

    ]);

    $entityManager->create("testClass3","testNameSpace\\TestClass2",[
            [ "attributeName" =>"firstDependency" , "classId" => "testClass1" ],
            [ "attributeName" =>"message" , "value" => "pel2" ]

    ]);

    $objectTestClass2 = $entityManager->get("testClass2");
    $objectTestClass3 = $entityManager->get("testClass3");

    echo $objectTestClass2->sayHi();
    echo "\n";
    echo $objectTestClass3->sayHi();

?>
```

Figure 6. Example of assigning value to class attribute

Afterwards, 2 objects were generated from TestClass2 on the EntityManager and the 'message' variable values of these objects were set. Here, when giving class dependency, information must be passed with the classId key value. If you want to assign data such as a string or int, it must be passed in the value key. In the method based above, when the class created in line with the information given with the create () method is taken with the get() method, the first created object is returned. There are as many objects in memory as created.

3. PEL Framework: An Application

An example web application developed with the PEL framework architecture is given in Figure 7. In this application, a text entered in a text box can be added to other notes with the 'Add' button. This application, which is based on HTTP requests, provides significant practicality by using reusable objects and less code.

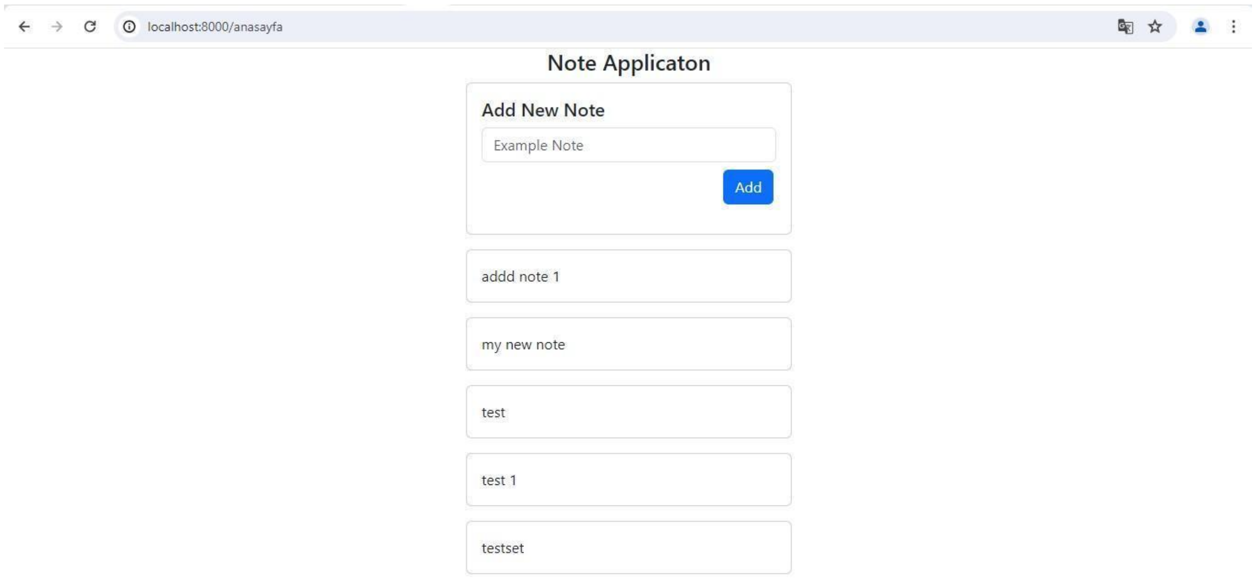
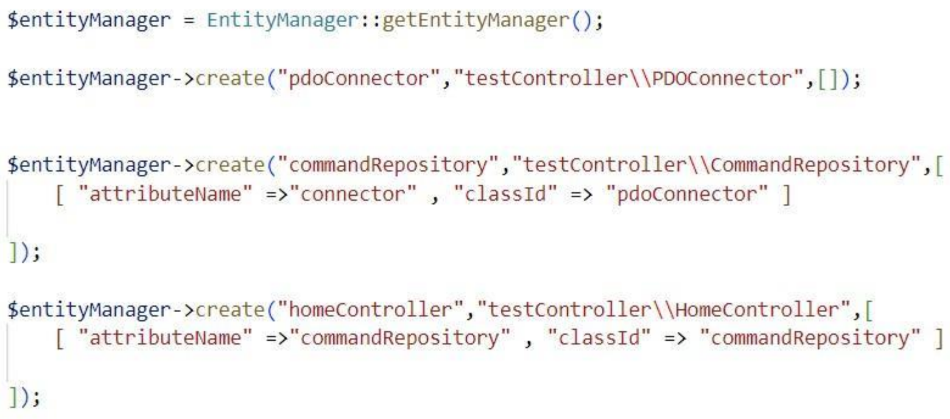


Figure 7. An example Note Application

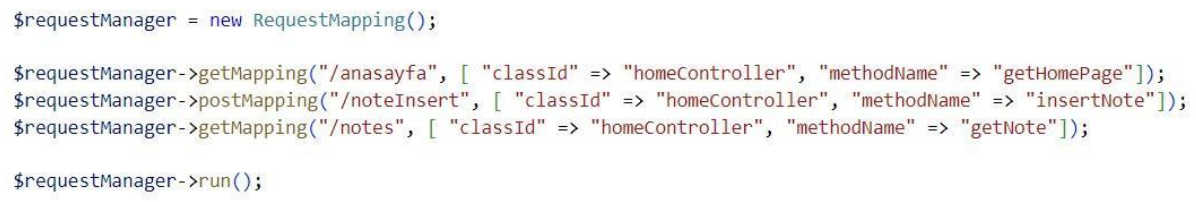
The previously mentioned HTTP and IoC Container modules were used in the background of the application shown in Figure 7. A code example containing the outline of the IoC container structure is given in Figure 8.



```
$entityManager = EntityManager::getEntityManager();

$entityManager->create("pdoConnector","testController\\PDOConnector",[]);


$entityManager->create("commandRepository","testController\\CommandRepository",[
    [ "attributeName" =>"connector" , "classId" => "pdoConnector" ]

]);

$entityManager->create("homeController","testController\\HomeController",[
    [ "attributeName" =>"commandRepository" , "classId" => "commandRepository" ]

]);
```

Figure 8. IoC Container structure

As seen in Figure 8, the dependencies of the PDOConnector, CommandRepository and Homecontroller classes were defined with each other via classId variables using the PEL Framework. If this type of framework was not used, new objects would have to be created and transferred to each other in order to use these three classes. As a result of this type of manual process, the current management would become more complicated and the same processes would repeat in a loop. The main advantage provided here is that the workload required to produce classes can be minimized by using design patterns such as factory patterns in the application. The code screen showing the HTTP requests used in the application shown in Figure 7 is given in Figure 9.



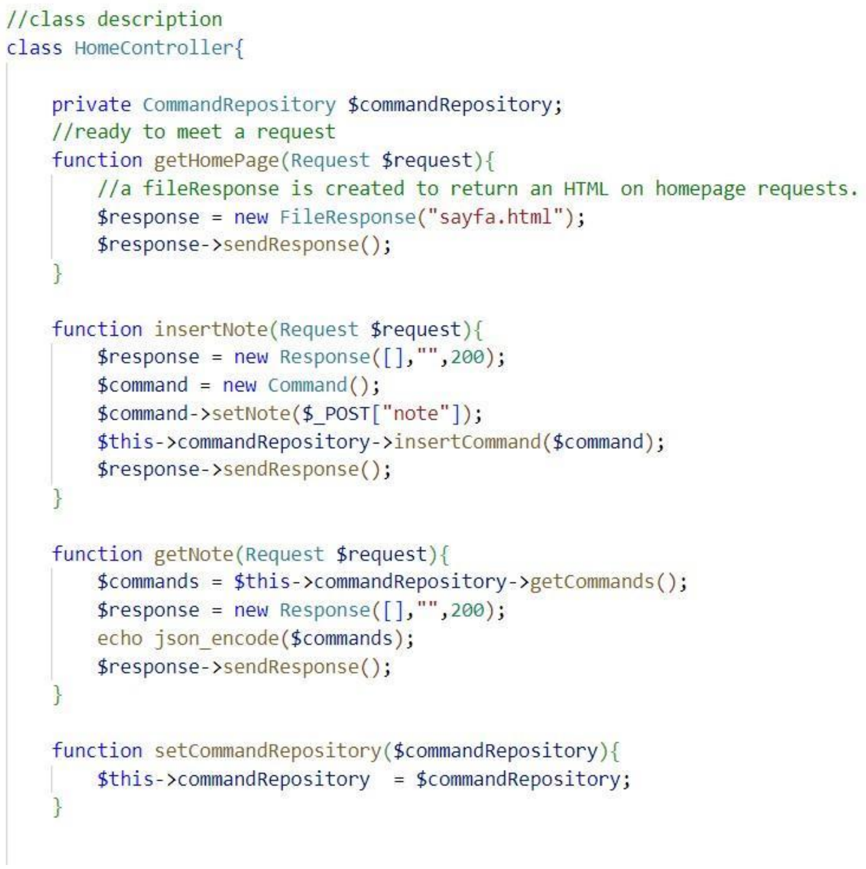
```
$requestManager = new RequestMapping();

$requestManager->getMapping("/anasayfa", [ "classId" => "homeController", "methodName" => "getHomePage"]);
$requestManager->postMapping("/noteInsert", [ "classId" => "homeController", "methodName" => "insertNote"]);
$requestManager->getMapping("/notes", [ "classId" => "homeController", "methodName" => "getNote"]);

$requestManager->run();
```

Figure 9. Example HTTP requests

Figure 9 defines which classes will handle incoming HTTP requests. Here, first, a definition is made with the method type and URL information, then the object that will respond to this request and the relevant method are defined, and finally, when an HTTP request suitable for this definition comes, the relevant method of the relevant object is executed. The home page of the application is called with the getHomePage method, the insertNote method writes the text to the database when the Add button is pressed, and the getNote method pulls data from the database by updating the current text to the list at the bottom of the screen every time the Add button is pressed. Here, the classId part shows the ID of the class that is desired to be used among the classes defined in the IoC container, so that HTTP and IoC container can work integrated with each other. The main goal of this structure is to easily manage requests at the HTTP level on an application basis and to increase their understandability.

Figure 10 shows the structure of the HomeController class, the methods discussed here correspond to the methods that will respond to HTTP requests shown in Figure 9.

The methods in the HomeController class receive a Request type parameter when HTTP requests are made. This parameter contains the details of the incoming HTTP request. Response type objects contain what will be returned in this HTTP request. The main advantage at this point is that the Request class is used for the details of the incoming request, and the details to be used in the Response to return a response.



```
//class description
class HomeController{

    private CommandRepository $commandRepository;
    //ready to meet a request
    function getHomePage(Request $request){
        //a fileResponse is created to return an HTML on homepage requests.
        $response = new FileResponse("sayfa.html");
        $response->sendResponse();
    }

    function insertNote(Request $request){
        $response = new Response([],"",200);
        $command = new Command();
        $command->setNote($_POST["note"]);
        $this->commandRepository->insertCommand($command);
        $response->sendResponse();
    }

    function getNote(Request $request){
        $commands = $this->commandRepository->getCommands();
        $response = new Response([],"",200);
        echo json_encode($commands);
        $response->sendResponse();
    }

    function setCommandRepository($commandRepository){
        $this->commandRepository  = $commandRepository;
    }

```

Figure 10. HomeController class

For example, the getHomePage method returns the content of the test.html file as a response to the HTTP request, while the insertNote method returns a response with empty content with an HTTP status code of 200. Thus, an understandable structure is provided for both making sense of HTTP requests and managing the content returned. With this example web application, the application practices of the solutions provided in parsing and managing HTTP requests with the IoC structure are shown, in the face of the difficulty of producing and managing objects in an HTTP-based application.

4. Discussion and Conclusion

With the PEL Framework presented in this study, it is aimed to reduce the effort required to create and manage the relevant classes by using the IoC container, and on the other hand, to gain an advantage in RAM and CPU needs by producing only one of these classes throughout the life of a web request. The classes created in this way are also used to respond to later requests. This allows incoming requests to be easily managed and answered. Thanks to the PEL Framework, infrastructures such as sending files or using them as a service are provided to respond to incoming requests. Here, the definitions to be made to manage these requests are both low-effort and manageable. Since class creation and management are operations that will be frequently needed in a web application or a service-based application, PEL framework greatly reduces the development process and management complexity.

References

[1] Johnson, R., Hoeller, J., Arendsen, A., & Thomas, R. (2009). Professional Java development with the Spring framework. John Wiley & Sons.

[2] Galloway, J., Haack, P., Wilson, B., & Allen, K. S. (2012). Professional ASP. NET MVC 4. John Wiley & Sons.

[3] He, R. Y. (2015, January). Design and implementation of web-based on Laravel framework. In 2014 International Conference on Computer Science and Electronic Technology (ICCSET 2014) (pp. 301-304). Atlantis Press.

[4] Sommerville, I. (2010). Software Engineering. Harlow, England: Addison-Wesley. ISBN: 978-0-13-703515-1.

[5] Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C., Loingtier, J. M., & Irwin, J. (1997). Aspect-oriented programming. In ECOOP'97—Object-Oriented Programming: 11th European Conference Jyväskylä, Finland, June 9–13, 1997 Proceedings 11 (pp. 220-242). Springer Berlin Heidelberg.

[6] Suh, N. P., & Suh, N. P. (2001). Axiomatic design: advances and applications (Vol. 4). New York: Oxford university press.

[7] Fowler, M. (2018). Refactoring: improving the design of existing code. Addison-Wesley Professional.

[8] Petersen, K., Wohlin, C., & Baca, D. (2009). The waterfall model in large-scale development. In Product-Focused Software Process Improvement: 10th International Conference, PROFES 2009, Oulu, Finland, June 15-17, 2009. Proceedings 10 (pp. 386-400). Springer Berlin Heidelberg.

[9] Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017, May). SCRUM model for agile methodology. In 2017 International Conference on Computing, Communication and Automation (ICCCA) (pp. 864-869). IEEE.

[10] Martin, J. (1991). Rapid application development. Macmillan Publishing Co., Inc..

[11] Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61-72.

[12] Ruby, S., Copeland, D. B., & Thomas, D. (2020). Agile web development with rails 6. Pragmatic Bookshelf.

[13] Azran, N. Z. A. Z., & Wahid, N. (2022). Design and Development of a Web-Based System using Laravel Framework: A Competition Management System. Applied Information Technology And Computer Science, 3(2), 514-532.

[14] Jazayeri, M. (2007, May). Some trends in web application development. In Future of Software Engineering (FOSE'07) (pp. 199-213). IEEE.

[15] Gharibeh, S., Melhem, S., & Najadat, H. (2020, April). Classification on Web Application Requests. In 2020 11th International Conference on Information and Communication Systems (ICICS) (pp. 1-5). IEEE.

[16] Althubiti, S., Yuan, X., & Esterline, A. (2017). Analyzing HTTP requests for web intrusion detection.

[17] Fowler, M. (2008). Inversion of control containers and the dependency injection pattern. http://www. martinfowler. com/articles/injection. html.

[18] Machacek, J., Vukotic, A., Chakraborty, A., & Ditt, J. (2008). Introducing Inversion of Control. Pro Spring 2.5, 31-72.