

PHP Faces

Component Based MVC Framework

Faces Framework PHP Application

Development Framework

Prepared by Huseyin Bora ABACI

Table of Contents

- Login
- Installation and requirements
- Model View Controller (Structure, View, Controller)
- import function
- Faces View
- The expressive faces
- @import tag directive
- Definitions and Use of Components
- event-oriented programming with PHP input Faces
- a facet
- facescontrol on
- Faces Model
- Faces Core PHP Class
 - she is Out tag
 - she is To teach performance label
 - she is Elsa tag
 - she is Else if tag
 - she is For tag
 - she is For each tag
- Components of
 - she is Button
 - she is Form
 - she is TextBox
 - she is Password
 - she is textarea
 - she is Hidden
 - she is Publisher
 - she is ComboBox
 - she is CheckBoxes
 - she is Radio
 - she is Image tag

she is [Link](#)

she is [Message](#)

she is [File Upload Component](#)

she is [Girdler](#)

- [Style Usage Example of the Access Control Component Attributes](#)

- [Your own components](#)

- [Static Labels](#)

- [Template engine](#)

- [Using View State](#)

- [verifications](#)

she is [Validations Validations](#)

she is [verification with the Validator Class](#)

she is [Multiple Validations](#)

she is [Your own confirmatory](#)

she is [To verify the operating events](#)

she is [Captcha Verification Components Photos](#)

- [Method of Dynamic Load Objects](#)

- [The URI Class](#)

- [Head Class](#)

- [Renderer Directives \(Directives Dşleyic\)](#)

she is [the @definitio](#)

she is [@import](#)

she is [@set](#)

she is [@get](#)

she is [in @inclu](#)

she is [@fa by](#)

she is [I @ite](#)

she is [@pattern](#)

she is [@ u](#)

she is [@htmlpattern](#)

she is [@ html](#)

- [Faces Model](#)

- [Entity \(Asset class\)](#)

- [Relations Between ORM Faces Asset Class](#)

- [Faces ORM Entity Manager \(Asset Manager\)](#)

- [FQL ORM Faces \(Faces Query Language\)](#)

- [Grid Tag](#)

- [SQL Tag](#)

- [FQL Label](#)

- [Page Label](#)

Login

event-oriented programming with PHP input Faces which deal with the Faces section of PHP will include some basic definitions If you're having previously about it from reading these pages **event-oriented programming with PHP input Faces** You can look at the title.

First, let's talk about whether what is happening in PHP Faces. PHP is a PHP based MVC application framework faces. Event-driven or used as unidirectional. Certainly there are also CodeIgniter PHP, CakePHP, Zend should not be like compared to the structure. None of these structures and components is oriented event. The main event oriented PHP development framework developed by Borland Delphi for PHP and firms are Prodon framework.

some shortcomings in developing applications with PHP hissesil which has been the motivating on a framework to develop this style. JTSL popular in the Java platform, and JSF framework of such Struts and was influenced from Coldfusion. Faces of a simple PHP PHP jfs'n adaptation can say. into JPA ORM layer in a similar bulunmakadır.

If you are familiar with the event-driven programs and see that you have adopted the method can be a good alternative Faces PHP Web applications.

PHP Faces

Viewer (to See) FDL (Faces Definition Language) (Face Definition Language)

Pattern (Mold) FQL (Faces) Query Language) (Face Query Language)

The names given to the hosts in two different languages FDL XML template, FQL uses the SQL mold.

Distinctive features

- MVC (Model-View-Controller)
- Component architecture
- UI components
- Event-driven programming
- AJAX
- Template Engine
- Custom Tags
- ORM (Object Relation Mapping)
- Validations (Verification)

The most important feature of PHP Faces studies comparing like with like roof when the roof. I thought it would bring a new concept to the Web programming patterns (patterns) concept. That as a Turkish programmers Microsoft, Sun, IBM, etc. Having developed before companies such as engineers and programming concept adds a new name for being extremely mutluyuyuz. Briefly describe your pattern in your view file and is a method which will save you from writing more code html etc. The pages will deyinil with advanced pattern to use.

In this section I finally giving place to the address where you can see PHP Faces good.

download page

Visit this page for

<http://code.google.com/p/php-faces/> Examples

<http://phpfaces.webmahsulleri.com/>

Installation and Requirements

Requirements:

PHP 5 and Over

Aphach to Rewi Aphach in

Module CGI Module

Activate PHP PDO for ORM

Setup :

Remove the compressed file

Move the folder to your server phpfaces

Work in PHP mode.

Faces can use the mold as PHP MVC PHP applications developed normally do not need to make an adjustment extra for it. Select the base you need to do the URL you want to use PHP PHP files on your faces I phpfaces / system / phpfaces.php file to include. If you prepare a configuration file, this file can be included language that would be more logical to work.

Sample

```
<? Php
treasure( "BASE_URL" , "Http: // localhost / phpfaces /" );
require_once ( "Phpfaces / system / phpfaces.php" ); import ( "Phpf.controllers.facet to" );

class Page extends Facet {
    function Page () {
        parent :: bezel ();
        $ this -> render ( "Manifestation / view.html" );
    }

    function dugmecklicked ( $ e ) {
        $ e -> getcomponent () -> text = "Called me" ; }}

$ Dispatcher :: dispatchclass ( "Page");
?>
```

Work with Directory structure

Open the index.php file in the home directory and edit it with an editor

Set the parameters according to your Dispatcher :: dispatch system.

```
Dispatcher :: dispatch ( "applications" .ds. "Seninuygul Game", "seninkontrolc" and "http: // seninhost. Com /");
```

Dispatcher 's parameters

1- current application directory

2- embodiment of the controller will begin Name

3-base URLs (base address) is the address that the PHP Faces in works

application / config.php file and edit feature to your own system.

```
define ( "DEFAULT_APPLICATION", "myApp"); // primary application directory
```

```
define ( "DB_CONNECTION_STRING", "MySQL: host = localhost; dbname = MyDB"); // connection string for ORM
```

```
define ( "DB_USER", "UserName"); define (
```

```
"DB_PASS", "2w34r5t");
```

```
define ( "BASE_URL", "http: // seninsunuc's /"); // file extension you want to ears as if this definition  
xxx.phpf .htaccess need to make some adjustments to it as necessary.
```

PHP supports the path style URL format faces look good for it. Should contain the following line in your .htaccess file.

RewriteEngine on

```
RewriteCond $ 1! ^ (index \ .php | images | themes | css | js | video_files | robots \ .txt | favicon \ ico) rewritecond  
REQUEST_FILENAME {%}! -f
```

```
rewritecond REQUEST_FILENAME {%}! -d RewriteRule ^ (.
```

```
*) $ | $ index.php index.php / $ 1 [L]
```

To use files with the extension PHPF CGI module it should be turned over to Apathach and your .htaccess file must be found on line similar to the following.

Options + ExecCGI

```
AddType application / x-httpd-phpf .phpf s AddHandler
```

```
application / x-httpd-phpf .phpf
```

```
Action application / x-httpd-phpf www.seninsunucun.com/system/phpf.php
```

When a request comes as the server of our .phpf thanks to this that we have done above PHP files with the extension Faces framework that detects .phpf view (view) files. Faces PHP if such a request. .php with the same name php controller (controller) spends years in motion. waste management controller dir.

Faces MVC (Model View Controller)

Model-view-controller, a software used to as an engineer "architectural design" is. User applications where large amounts of complex data presented is based on data abstraction and representation basis. Thus, data (model) and user interface (view) can be edited without affecting each other. Model-View-Controller, which means that the components of the controller name search, data display and user interaction, solves the problem by removing the data access and business logic.

M (Models), executes the Labor Logic and data processing processes. **C (Controller)** According to the act by sending orders. After the information processing data in C, or directly to other models **V (Viewer)** to send.

Faces Model layer in PHP **PDO** and ORM Uses

V (View) is concerned with the presentation of the data to be displayed to the end user. V, C or this information **M 'Gets den, at the same time demand from end-users C PMU.**

C (Controlle r)

C is the main part of the system. checks incoming requests and other elements of the system (M, V) and receive information properly, allows users to send.

Controller accommodates them in three different PHP Faces

1. ActionController
2. facescontrol on
3. Facet

ActionController is useful in case of a structure is not desired user information input from ordinary non-oriented events.

facescontrol is oriented event. Faces used with components.

Facet facescontrol has expanded from facescontrol and simplify u.

Faces i PHP is distinguished from other MCV roof Controller and View components and code between processors is the presence of the interpreter means simple. A view file rendering method of controlling a browser on the client side created an output. In the Render method parameter file to pass from a simple interpretation process itself.

Faces MVC PHP directory structure is as follows.

AppPath:
/ Phpfaces: Framework folder /
applications: applications

app_b is / controllers /
views / models

app_two / controllers / views
/ models

..htaccess

A small startup with Faces

I remind you that in this example the directory structure if you do not want to use the directory structure is used as a traditional php files as described in this case a setup page. The backbone of the MVC Controller As we mentioned earlier, is part. Expand the controller class you about your controller when you create a controller.

Yes, if everything is ready so we can do our initial PHP setup process results in success if you try Faces

Firstly

applications / ApplicationName he / controller /

applications / ApplicationName he / controller / test.php

Controller create our files in a directory where test.php name as his bride.

Example of our content controller

```
<? Php
import ( "Phpfaces.controllers.facescontrol on" );
class Test extends facescontrol the {
    function Test () {
        parent :: facescontro on ();
        echo "The test controller was initiated" ;}}

?>
```

Now you're thinking working on localhost in the address line of our browser

http: // localhost / phpfacesdiz the / was ugulamaa / test

If you're writing, if everything goes well "test controller was launched" We are getting the message.

Now

applications / ApplicationName / views /

Let's create a simple view point to the directory with a name like test.phpf here.

faces our code <faces> </ faces> We place between FDL telling them we will discuss in depth

```
<Faces> <b> # {} $ This.mesaj </ B> </
faces>
```

Controller as follows in our class edit our render method of the controller creates a client-side processes the output to a file viewer.

```
<? Php
import ( "Phpfacesdiz the / was ugulamaa / test
class Test extends facescontrol the {
    protected $ messages = "The test controller was initiated" ;
    function Test () {
        parent :: facescontro on ();
        $ this -> render ( "Test.phpf" );
    }
?>
```

Again

http: // localhost / phpfacesdiz the / was ugulamaa / test

If you're writing, if everything goes well "test controller was launched" We are getting the message.

Import Function

Import functions shall include in your PHP class files that are required for you to use. PHP's require_once (string filename); function is operating. To include all files in the same directory as the last character asterisk * character.

```
import ( "phpf.controllers.facescontrol on");
```

Phpf above this line / controllers / facescontroller.php file is included.

```
import ( "phpf.ui. *");
```

Phpf above this line / ui / php files in the directory are included in all. The second parameter of the function Import true or false takes will look in the default application directory to import files related to the case of true function of this parameter.

```
import ( "models. *");
```


all php files are imported under the models folder in the default application directory with the line above.
Examples

```
<? Php
import ( "Php.controllers.facet to" ); import ( "model.**", true ); import ( "The
io.ur" );

?>
```

Faces Views

You can view completely separate from other codes similar to Faces Viewer s XML notation. JavaServer Faces
Faces structure PHP, Coldfusion has added some features taken from some unique features.

Faces section shall be subject to the Render I `<faces> </ faces>` Written between the tags. `<Faces> </ faces>` tags
in HTML, CSS, PHP, etc .. You can use the XML languages.

Labels are divided into these three e Faces

- Renderer directive labels
- Static tags
- Component Labels

All of the Renderer directive begins with an @ sign

```
<@Import prefix = "f" taglib = "phpf.u in. *" />
```

This directive tells the server to a renderer included for all classes within the phpf.u and associate with the name space.

```
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
```

This directive tells the renderer to include a car phpf.cor class and associate it with the name space. static reports that
the phrase should be called as static methods of this class.

Sample

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: out value = "Hello Faces" />
</ Faces>
```

Another example

```
<Faces>
```

```
<@Import prefix = "c" taglib = "phpf.cor" type = "static" /> <@import prefix = "f" taglib = "phpf.u in. *" />
```

```
<C: if test = "1 <2" > <C: out value = "It is smaller than one or two" /> </ C:
if> <f: button name = "Button" text = "click" />
```

```
</ Faces>
```

Defining Attributes in single quotes 'or double quotes, use spaces "" Use the character in quotation marks and also' Do not use single quotation mark

Examples

```
< faces>
```

```
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
```

```
<C: out value = trial /> You receive incorrect error
```

```
<C: out value = 'trial' /> You receive incorrect error
```

```
<C: out value = "true" /> works
```

```
<C: out value = "False" /> There work in single quotes
```

```
</ Faces>
```

Faces statements

PHP faces express the view (View) in expression are used. These statements refers to a component output or # {and} Expressions are written to the range.

Examples

```
<B>
```

```
#Variable $ {} </
```

```
b>
```

The above example prints the contents of \$ variable

```
#{1 + 2}
```

```
## $ {$ I + j}
```

```
#{ $ I * $ j}
```

Use of expression in the sample HTML

```
<Faces>
```

```
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
```

```
<input type = "Text" name = "Text" value = "# {} $ Variable" /> <C: fr var = "$ I" begin = "0" to = "10" step = "one" >
```

```
<Input type = "Text" name = "Text # {$ i}" value = "# {$ I}" /> </ C: fr> </ faces>
```

@import directive tag

Directive tags view (view) contained in the file in the identification process are the labels used for each directive faces renderer starts with the @ sign and has specific characteristics of each directive.

Directive shall include php faces ðmport component files specified by TagLib nature and associated with the prefix specified by prefix attribute. Type nitelg relates to the mode of operation of the library to be used if the PHP class library stationary (static) should be given the true value of this attribute contains methods. Core.php class is a static library.

- TagLib: to be or php file and import directory
- prefix: name to be used with front labels
- type: the type of component libraries

The type attribute is used only in static classes and PHP Faces in these single static library dir core.php

Sample

```
<Faces>
<@Import TagLib = "phpf.cor to" prefix = "c" type = "static" />
<C: out value = "Hello Faces" />
</ Faces>
```

Above phpf / Core static class within core.php directory to be imported and "c" is ilişkilendirt with prefix. Core tags associated with the class "c" will be heading.

<C: out value = "Hello Faces" /> Out of line with the core class method is executed and Hello Faces browser font is printed.

Another example

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" /> <@import prefix = "f" taglib = "phpf.u in. *" />

<C: if test = "1 <2" > <C: out value = "It is smaller than one or two" /> </ C:
if> <f: button name = "Button" text = "click" />

</ Faces>
```

Then again above core classes associated with phpf c / u with all classes in the directory "F" prefix it is associated.

Another example of a more

```
<Faces>
<@Import prefix = "core" taglib = "phpf.cor" type = "static" /> <@import prefix = "face" taglib = "phpf.u in. *" />

<@Import prefix = "widgets" taglib = "phpf.ui.widget.djbutto's" />
<Core: out value = "Hello World" /> <Face: button name = "Button" text = "Click me" /> <Widget: the djbutto name
= "The djbutto" text = "Widget button" /> </ Faces>
```

Following the above core classes associated with core prefix php / u with all classes in the directory "face" prefix associated and "widgets". php with prefix / ui / widgets / djbutto in the class that is associated djbutton.php.

Definitions and Use of Components

View component in MVC PHP Faces (View) is defined. Components are a PHP class.

After the prefix specified in the directive `<import name=` it is defined as a colon mark component name and attributes.

Sample

```
<Faces>
<@import prefix = "f" taglib = "phpf.ui.button" />
<f:button name = "button" text = "click" />
</Faces>
```

a new object is created, called the wedding button from grades above the line and click on the text attribute value is given. Text attribute of the paper will be printed on the created button.

name or id attributes of each ingredient label should be left blank. When accessing the object in PHP it will be accessed by those names.

Our php class controller based on the above example has a button object name in the wedding.

The wedding in the PHP object is accessed as follows.

```
$this->button->text = "New post";
```

Faces oriented event with PHP programming input

PHP controller maintains two classes that support event-oriented programming in their faces to facescontrol and Facet are class. Both controllers also write you view as a parameter the name of the file you want to commit to a process the render method when you want to view files

a facet

Facet class A and class expands facescontrol

ActionListener, MouseListener, valuechangedlisten the interface (interface) is a class that implements.

a facet of Vbasic is programmed in a way close to the Delphi until the incident orientation. Facet you expand the class to your class to create a facet to the controller and a founding method you prepare your class with the same name.

Events capture (Event Handling) provided by member methods. The name of the method to capture the event is important. Event method names capture the object name as the name of the event.

Example For example, when a named Facet to our class and we do get a button dugme named in the definition file appearance button is clicked we want to change the text on the button.

Our View File

```
<Faces>
    <@Import TagLib = "phpf.ui.butto the" prefix = "f" />
    <F: button name = "button" text = "Click" onclick = " ActionEvent " />
</ Faces>
```

Our Facet Controller

```
<? Php
import ( "Phpf.controllers.facet to" );
class Sample extends Facet {
    the public function Sample() {
        parent :: bezel ();
        $ this -> render ( "View.html" );
    }
    function dugmeclicked ( $ evt ) {
        $ this -> buttons at> text = "I clicked" ; }}

?>
```

Clicking the button dugme post page is operated by the user and the dugmeclicked method.

If you wish to refresh the page without sending the data with AJAX it is required few changes to your application.

We use to define the appearance of the button object ajaxevent we give value to the onclick event you have said that this object is sent to the server with Ajax. dugmeclicked method finally \$ this -> ajaxrespo possible (); We need to run the method. This method transfers the JSON data to the browser side.

AJAX works with our example is the state.

Our View File

```
<Faces>
    <@Import TagLib = "phpf.ui.butto the" prefix = "f" />
    <F: button name = "button" text = "Click" onclick = " ajaxevent " /> </ Faces>
```

Our Facet Controller

```
<? Php
import ( "Phpf.controllers.facet to" );
class Sample extends Facet {
    the public function Sample() {
        parent :: bezel ();
        $ this -> render ( "View.html" );
    }
    function dugmeclicked ( $ evt ) {
        $ this -> buttons at> text = "I clicked" ;
        $ this -> ajaxrespo possible ();
    }
}

?>
```

facescontrol on

Event-oriented facescontrol Another approach is to use. Faces in the event mechanism contains PHP is good to understand three important concepts. Faces PHP Java event capture patterns and style. If this describes you have previously developed applications with java swing foreigners will come to you.

- Listener: Listeners
- Events: Events
- Component: component for on

Listener interface defines an interface you will be listening to the Controller class you implement these interfaces.

Events are objects related events. These are formed by the framework and the event is passed to the related process parameters.

Component events are usually triggered by a component.

They also found that the listener PHP Faces

- The ActionListener: Used to click event
- MouseListener: used for mouse events
- It is valuechangedlisten: The value used is changed

```
<? Php
interface ActionListener extends faceslisten is {
/ **
 * @my money $ ActionEvent event
 * /
the public function actionPerformed (ActionEvent $ event );
}>
```

The objects found in these events PHP Faces1.0

- ActionEvent: When a click event occurs
- moueseevent: when a mouse event occurs
- valuechangedevent: change event occurs when

controler our class listener calls and records related to the method for implementing class event listener.

PHP faces post method uses two basic methods for post-back events in the first and second method AJAX t.

view to the realization of the event (view) must be reported while the component definitions, rep.

If you want to make components Component post back to the definition of the relevant facts nitelg "ActionEvent"

components related to events in the component definition nitelg If you want to make an AJAX call "ajaxevent" You give one of the values.

Example component and event identification

```
<Faces>
<@Import prefix = "face" taglib = "phpf.ui.butto's" />
<Face: button name = "Button1" text = "Click" onclick = " ajaxevent " />
</ Faces>
```

We create a new instance of a button named button1 class with the above lines. Controller can access through the button1 object

View our example above for the controller

```
<? Php
import ( "Phpf.controllers.facescontrol on" ); import ( "Phpf.events.actionevent" );
import ( "Phpf.listeners.actionlisten is" );

class Sample extends facescontrol on implements ActionListener {
    function Sample () {
        parent :: facescontrol on ();
        $ this -> addactionlisten is ( $ this );
        $ this -> render ( "View.phpf" );
    }
    the public function actionPerformed (ActionEvent $ evt ) {
        $ this -> button1-> setText ( "Click me" );
        $ this -> ajaxrespo possible ();
    }
}

?>
```

To summarize the above lines

class Sample extends facescontrol and implements ActionListener

Applying ActionListener listener.

\$ This-> addactionlisten is (\$ this);

kayıtediliy listener object.

\$ This-> render ("view.phpf");

Our view file is rendered.

Client When the button is clicked on the browser side we generate ActionEvent event is occurring and executing the actionPerformed method.

```
public function actionPerformed (ActionEvent $ evt) {
```

\$ This-> button1-> setText ("click me");

Changing the nature of the text, click the button object to me.

\$ This-> ajaxrespo possible ();

Client-side browser to the JSON data is being sent. Running Ajax update code on the browser side.

facescontrol and the bezel member methods are as follows.

- **rendering** (This meter tells a renderer to interpret faces a file)
- **PreRender** (The rendering process initiated işletilip output file viewer when it is not yet operated oluşturul **controlle** edilerek used in override r)
- **renderend** (rendering process is executed when the tamınlan **Controller** in edilerek used override)
- **append** (A viewer sends a parameter)
- **load** (A new class of objects called the string as a parameter dials)
- **interrupter** (interrupts the rendering process)
- **addListener** (\$ name, faceslisten from \$ listener)
- **addactionlisten is** (A listener interface (interface) kayıted parameters for the event
ActionEvent ActionListener d)
- **addmouselisten is** (A listener interface parameters of the mouse events events would içinkayıt
MouseListener)
- **addvaluechangedlisten is** (Registers a listener interface for valuechangedevent the event.
Valuechangedlisten parameter)
- **getcomponents** (contoller data associated with all components)
- **ajaxrespo Anse** (It takes the rendering process to suspend json and sends the data to the client side)
- **setvalidcallback** (boolean (events in the operation of the verification process used in the method if this
condition gerekiry true value is given))
- **boolean isValid ()** (If there is an error in the results of the verification process returns true or false)

PreRender and renderend override methods are used in the Control Example was

```
<? Php
import ( "Phpfc.controllers.facet to" );
class render extends Facet {
    the public function Render () {
        parent :: bezel ();
        $ this -> render ( "File.html" );
    }
    the public function PreRender () {
        echo "Render process was initiated objects created" ;
    }
    function renderend () {echo "Render process ended page was created" ; }}
?>
```


PHP Faces Model

PHP ORM layer faces models (Object relation mapping) are used to anlatılınca this section as a separate document.

Faces Core PHP Class

Core class library is a static label. Static libraries and component libraries differences between the components of the components to produce a new class object and controller to be a member of. The static classes are classes renderer produces output consisting of static methods. These tags do not receive the name and id values. Core class is a static class that labels of some basic operations. located in the Core class phpface directory. Note that in the examples in this section is applied to create a controller and the sample should be rendered.

```
<@Import prefix = "core" taglib = "phpface" type = "static" />
```

Files can be imported into view as above. It includes the Core class includes the following labels.

- Depart
- iF
- else
- elseif
- for
- foreach

Out Tag

used for the output. This is indicated by the label data will be sent to the output value attribute

Example Out

```
<Faces>
<@Import prefix = "c" taglib = "phpface" type = "static" />
<C: out value = "Hello World" /> <C: out value = "#{That}
birdeğişik" /> </ Faces>
```

iF label

To teach performance label used in the comparison process has a fundamental attribute of the iF label named test and written expression comparison to these qualities. Offered every iF label has to be closed.

```
<C: if test = "#{phrase}" >
All the expression tested case this line is processed. </ C: if>
```

Example teach performance

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: if test = "Gust" # {$ name == " " > <C: out value = "#
{$ Name}" /> </ C: if> </ faces>
```

For example teach performance early grave

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: if test = "# {$ I> 0 AND $ i <10}" > <C: out value = "Number between 10 to 1" />
</ C: if> </ faces>
```

Another example teach performance

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: if test = "# {$ Name! = Huseyin}" > <Strong> <C: out value = "That's not
my name!" /> </ Strong> </ c: if> </ faces>
```

Elsa Label

Elsa tag is used in conjunction with iF label. The expression used to apply if the case is different. PHP is like until use else.

Example else

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: if test = "$ Name == bora" > <C: out value = "# {$
Name}" /> <C: else>

<C: out value = "Another name # {$ name}" /> </ C: else> </ c: if> <faces>
```

In the above example, if the variable \$ name variable is tested out the value of this variable label is printed with borane. If the variable value is a different value "Another printed name and the value of the variable \$ name.

An example that distinguish double and single counts

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: fr var = "$ I" begin = "one" to = "12" step = "one" > <C: if test = "($ I% 2) == 1" >
```

```

<C: out value = "#{ $ I}" /> odd number <br /> <c: else>

<C: out value = "#{ $ I}" /> Even number <br /> </ c: else> </ c: if> </ c: fr>

<faces>

```

Else if Label

elseif tag is used in conjunction with if label. In case of false statements used if compared with the rectilinear nature of the expression in the test elseif tags.

Example elseif

```

<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<Faces>
  <@Import prefix = "c" taglib = "phpf.cor" type = "static" />
  <C: if test = "Gust" # { $ name == } > <C: out value = "#{ $ Name}" /> <C: elseif test
    = "Huseyin" # { $ name == } > <C: out value = "Name huseyin" /> </ C:
    elseif> </ c: if> </ faces>

```

Example if elseif else

```

<Faces>
  <@Import prefix = "c" taglib = "phpf.cor" type = "static" />
  <C: if test = "Gust" # { $ name == } > <C: out value = "Name bora" /> <C: elseif test
    = "Huseyin" # { $ name == } > <C: out value = "Name huseyin" /> </
    C: elseif> <c: else>

    <C: out value = "Undefined name" /> </ C: else> </ c: if> </ faces>

```

For tag

used to create a label for the cycle. There are four attributes.

- There are: variable
- The defense: starting value of an integer
- to: the ending value of an integer
- step: increment

There are variables indicating the increases begin with until it reaches a specified amount to the initial amount specified starting value indicated by the step or reduced.

Examples 1 to 10 show numbers up.

```

<Faces>

```

```
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: fr var = "$ I" begin = "one" to = "10" step = "one" > <C: out value = "# {$ I}" /> </ C:
fr>
```

```
</ Faces>
```

The output will be as follows 1,2,3,4,5,6,7,9,10

Example two even numbers increase

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: fr var = "$ I" begin = "0" to = "10" step = "2nd" > <C: out value = "# {$ I}" /> </ C: fr>
</ faces>
```

0,2,4,6,8,10 example of the output is

for backwards

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: fr var = "$ I" begin = "10" to = "0" step = "one" > <C: out value = "# {$ I}" /> ,
</ C: fr> </
faces>
```

Of the output is 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0

Example nested for use

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: fr var = "$ I" begin = "0" to = "10" step = "one" > Reviews <c: out value = "$ I = # {I}" /> <C: fr var
= "$ J" begin = "$ I" to = "10" step = "2nd" > <C: out value = "$ {J} j = #" /> ,
</ C: fr> </ c:
fr> </ faces>
```

The output will be as follows

```
i = 0, j = 0, j = 2, j = 4, J = 6, J = 8, J = 10 i = 1, j =
1, j = 3, j = 5, J = 7, J = 9 i = 2, j = 2, j = 4, j = 6, j =
8, j = 10 i = 3, j = 3, j = 5, j = 7, j = 9, i = 4, j = 4, j =
6, j = 8, j = 10 to i = 5, j = 5, j = 7, j = 9 i = 6, j = 6, j
= 8, j = 10 i = 7, j = 7, j = 9, i = 8, j = 8, j = 10 i = 9, j
= 9
```

Foreach tag

For tag works in a similar way it has an effect on arrays and objects. There are two attributes and the item names

- There are: iteration will be the object or array
- item: available objects during the process of Example

foreach

```
<? Php class Person {
    the public $ id ;
    the public $ name ;
    the public $ PHONES = array (); }
```

```
?>
```

circulating a sequence formed from the above examples person class foreach

Examples of the class is \$ this.list Persona series.

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: foreach var = "$ This.list" item = "$ Item" > <br/>

    name: <C: out value = "{But} $ item.n" />
    id: <C: out value = "{ } $ Item.id" /> </ C: foreach> <faces>
```

Example nested tags foreach

```
<Faces>
<@Import prefix = "c" taglib = "phpf.cor" type = "static" />
<C: foreach var = "$ This.list" item = "$ Item" > <br/>

    name: <C: out value = "{But} $ item.n" />
    id: <C: out value = "{ } $ Item.id" /> <C: foreach var = "$ Item.phones" item = "$ Phone" >

        Phone: <C: out value = "{ $ Phone}" /> </ C: foreach> </ c:
foreach> <faces>
```

components

PHP There are many ingredients in Faces. You can also develop your own components. We begin on the following pages to this subject.

Faces in PHP contains the DOJO Java Script framework. You can find detailed information about Dojo in <http://www.dojotoolkit.org/> address.

Under U and widget name in conjunction with PHP At the moment there are 2 Faces component library. ui.widget under standard HTML form components under the Dojo JavaScript-based components, there are digits.

HTML and Widgets can be found under the heading <http://www.webmahsulleri.com/faces/doc/index.phpf> address for use in instances of components.

The HTML Components

- Button
- TextBox
- Publisher
- combobox
- CheckBoxes
- Image
- Hidden
- Radio
- Form
- Grid
- textarea

Button

Button component generates an HTML button tag is usually ***onclick*** Button is used in conjunction with components of the event means writing on the text attribute enter. HTML input text support for all qualities belonging.

Button definition in Example view

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
  <Head> <title> Example Button </Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import TagLib = "phpf.ui.butto the" prefix = "f" />
  <F: button name = "button" text = "Button" onclick = " ajaxevent " /> </ Faces> </ body> </ html>
```

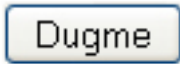
Our example controler side

```
<? Php
import ( "Phpf.controllers.facet to" );
class Sample extends Facet {
  the public function Sample() {
    parent :: bezel ();
    $ this -> render ( "Ornek.html" );
  }

  protected function dugmeclicked ( $ incidents ) {
    $ this -> buttons at> setText ( "Give me Click" );
    $ this -> ajaxrespo possible ();
  }
}

?>
```

previous screen without clicking the node component



after the appearance of the component clicked Dugi



If desired components in the controller knows to be formed. These components *protected* and either *the public* There should be a member. The method of the constituent components of the controller accepts a class parameter.

```
$ this -> dugme = new Button ( $ this );
```

forming the controller component of the sample Dug

```
<? Php
import ( "Php.controllers.facet to" ); import ( "Php.ui. *" );

class Sample extends Facet {
    protected $ Dugme = null ;
    the public function Sample() {
        parent :: bezel ();
        $ this -> dugme = new Button ( $ this );
        $ this -> render ( "Ornek.html" );
    }

    protected function dugmeclicked ( $ incidents ) {
        $ this -> buttons at> setText ( "Give me Click" );
        $ this -> ajaxrespo possible ();
    }
}

?>
```

Form

Forms component defines an HTML form used to hide form Button object. Faces a PHP framework automatically creates an HTML form most similar orientation events and stores all the components in this form. PHP Faces have made this method is user defined. All HTML form that sends all the components within the server. the remaining ingredients except the form they just send their server. Use HTML form is a similar manner. Button or component you want to post a different form *form Game*

You write the name of the component you want to post to the nature of the form.

TextBox

HTML TextBox component is used to input text and tags corresponding to data entry. Component of the TextBox *text* attribute defines the data it hosts the TextBox object.

Use of Representative Form and TextBox Components

View

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head> <title> Example 2 </title> <meta http-equiv="Content-Type" content="Text / html; charset = UTF-8" >

</head>
<body>
<faces>
  <@import TagLib = "phpf.ui." prefix = "f" />
  <f: form name = "registration form" method = "Post" >
    Member Name : <f: TextBox name = "Member Name" />
    Email : <f: TextBox name = "email" /> <f: button name = "Gonder" text = "Gondor" onclick = " ActionEvent "

form = The "registration form" />
  </f: form> </
</faces> </body> </html>
```

Example controller

```
<? Php
import ( "Phpf.controllers.facet to" );
class Example 2 extends Facet {
  the public function Example2 () {
    parent :: bezel ();
    $ this -> render ( "Ornek2.html" );
  }
  protected function gonderclicked ( $ incidents ) {
    echo "Sent u ordinary" . $ this -> uyeadi-> text;
    echo "Member Sent E-Mail:" . $ this -> eposta-> text;
  }
}

?>
```

Gonerilen uye adi :huseyin

Gönderilen E-Posta :bora@huseyin.com

Uye adi : E-Posta :

textarea

textarea component provides data entry for long texts such as HTML is to use until the textarea. As in the TextBox object data to be accessed from the text attribute it contains.

In the example we create a sample form for use in a TextBox input address information of the object TextArea add you.

Gönderilen üye adı :huseyin
 Gönderilen E-Posta :bora@huseyin.com
 Gönderilen Adres :Felaca mahallesi 78/55 Keçiören /Ankara
 Üye adı : E-Posta :
 Adres :

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
  <head> <title>EXAMPLE 3</title> <meta http-equiv="Content-Type" content="Text/html; charset=UTF-8">

</head>
<body>
  <faces>
    <@import TagLib="phpf.ui." prefix="f"/>
    <f:form name="registration form" method="Post">
      Member Name : <f:TextBox name="Member Name"/>
      Email : <f:TextBox name="email"/> Reviews

      address: <f:textarea name="address"> </f:textarea> <f:button name="Gönder" text="Gönder" onclick=" " ActionEvent
    "
  </f:form>
</body>
</html>
```

controller

```
<?php
import ("Php.controllers.facet to");
class EXAMPLE 3 extends Facet {
  the public function Example3 () {
    parent :: bezel ();
    $ this -> render ( "Ornek3.html" );
  }
  protected function gonderclicked ( $ incidents ) {
    echo "The sending u the name" . $ this -> uyeadi-> text;
    echo "Member Sent E-Mail:" . $ this -> eposta-> text;
    echo "Member Sent Address:" . $ this -> Address-> text;
  }
}
```

Password

Parola bileşeni TextBox bileşeniyle tek farkı görünümde olup olmadığıdır. Parola olarak girilen bilgi gizli olarak görünür.

Hidden

Gizli bileşen, kullanıcıya görünmeyen, gizli bilgileri saklayan bir bileşendir. Özellikle AJAX uygulamalarında kullanılır.

Publisher

To add a text label or tag component used to fill the inside of AJAX.

Let's develop the registration form in the previous example and using AJAX Label.

Uye adi : E-Posta :

Adres bilgileri...

Adres :

Gönderilen uye adi :huseyin

Gönderilen E-Posta :bora@huseyin.com

Gönderilen Adres :Adres bilgileri...

View

```
<Html>
```

```
<Head> <title> Example 4 </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >
```

```
</ Head>
```

```
<body>
```

```
<faces>
```

```
<@Import TagLib = "phpf.ui. *" prefix = "f" />
```

```
<F: form name = "registration form" method = "Post" >
```

```
Member Name : <F: TextBox name = "Member Name" />
```

```
Email : <F: TextBox name = "email" /> Reviews
```

```
address: <F: textarea name = "address" > </ F: textarea> <f: button name = "Gonder" text = "Gondor" onclick = " ajaxevent "
```

```
form = The "registration form" />
```

```
</ F: form> <font color = "rejection" > <F: label name = "tag" > </ F: label> </ font> </ faces>
```

```
</ Body> </ html>
```

controler

```
<? Php
```

```
import ( "Phpfc.controllers.facet to" );
```

```
class Example 4 extends Facet {
```

```
the public function Example 4 () {
```

```
parent :: bezel ();
```

```
$ this -> render ( "Ornek4.html" );
```

```
}
```

```
protected function gonderclicked ( $ incidents ) {
```

```
$ data = "Goneril u ordinary" . $ this -> uyeadi-> text;
```

```
$ data . = "Member Sent E-Mail:" . $ this -> eposta-> text;
```

```
$ data . = "Member Sent Address:" . $ this -> Address-> text;
```

```
$ this -> Label-> text = $ data ;
```

```
$ this -> ajaxrespo possible ();
```

```
}}
```

```
?>
```

ComboBox

ComboBox component creates a selection box comes in response to the select tag in HTML view side with these components OnChange Changed the nature and methods used in the controller side. Changed method of changing the value of the object yakalar.combobox

getselected () method contains values selected. The filling for the ComboBox has 3 paths. These three methods are discussed in the following examples.

Our first example is a ComboBox and a Label component to display in our chosen for the selection of the city. If the city changed when the OnChange event occurs and runs Changed method of controlling side. option tag is used to fill the ComboBox component in this example.

Şehirler : Seciminiz : Çankırı

View

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
  <Head> <title> Option </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import TagLib = "phpf.ui." prefix = "f" /> Cities

  <F: ComboBox id = "cities" onchange = " ajaxevent " > <Option> Ankara </ Option> <option> Istanbul
    </ Option> <option> İzmir </ Option> <option> Erzurum </ Option> <option> Van </
    Option> <option> Diyarbakir </ Option> <option> Çankırı </ Option> </ f:
    ComboBox>
```

You Elections:

```
<font color = "Blue" > <F: label id = "selection" > </ F: label>
```

```
</ Font> </
```

```
faces> </ body> </
html>
```

controler

```
<? Php
import ( "Phpf.controllers.facet to" );
class Example 5 extends Facet {
  the public function Example 5 () {
    parent :: bezel ();
    $ this -> render ( "Ornek5.html" );
  }
  protected function şehirlerchanged ( $ incidents ) {
    $ this -> CHOICES> setText ( $ this -> ŞEHİRLER-> getselected ());
    $ this -> ajaxrespo possible ();
  }
}
```

?>

The same example above, this time **binder** We will apply the characteristics of components using **binder** accept an array of attributes or objects. Consider the following example for the use of your Binder.

Example Binder usage.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
  <Head> <title> Option </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import TagLib = "phpf.ui. *" prefix = "f" /> Cities

  <F: ComboBox id = "cities" onchange = " ajaxevent "
    binder = "Array (Ankara, Istanbul, Izmir, Erzurum, Van, Diyarbakir, Çankırı)" >
  </ F: ComboBox>
  You Elections:
  <font color = "Blue" > <F: label id = "selection" > </ F: label>

  </ Font> </
faces> </ body> </
html>
```

In our third example, we create content optionally changing a ComboBox component. In this example, the first tank to the second ComboBox ComboBox selected cities based on the knowledge we will use the method of controlling setmodel side of the ComboBox object of dolduracağız.b with districts in this city.



View

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
  <Head> <title> Option </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import TagLib = "phpf.ui. *" prefix = "f" /> Cities

  <F: ComboBox id = "Provinces" onchange = " ajaxevent "
    binder = "Array (Ankara, Istanbul, Izmir, Erzurum, Van, Diyarbakir, Çankırı)" >
  </ F: ComboBox>
  District Agricultural:
  <F: ComboBox id = "They ILC" onchange = " ajaxevent "
    binder = "Array (Keçiören, Mamak, Cankaya, Yenimahalle, Xinjiang)" >
```

```

</ F: ComboBox> </
faces> </ body> </ html>

```

controler

```

<? Php
import ( "Phpf.controllers.facet to" );
class Example 6 extends Facet {
    the public function Example 6 () {
        parent :: bezel ();
        $ this -> render ( "Ornek6.html" );
    }
    protected function illerchanged ( $ incidents ) {
        $ Tumilc Show = array (
            "Ankara" => array (Keçiören, Mamak, Cankaya, Yenimahalle, Xinjiang),
            "Istanbul" => array (Islands, Fatih, Flat, Bağcılar, Bakirkoy)
            "Izmir" => array (Islands, Fatih, Flat, Bağcılar, Bakirkoy)
            "Çankırı" => array (Eldivan, Ilgaz, leaded, Şabanözü)
            "Erzurum" => array (Ashkali, Khorasan, Karayazı, Pasinler)
        );

        $ province = $ this -> iller-> getselected ();
        iF (Array_key_exists ( $ province , $ Tumilc Show ))
            $ this -> ilceler-> setmodel ( $ Tumilc Show [ $ province ]);
        else
            $ this -> ilceler-> setmodel ( null );
        $ this -> ajaxrespo possible ();
    }
}

?>

```

ComboBox object **setmodel** method **Array** It accepts data type and modifies the contents of the new value.

CheckBoxes

Checkbox component creates a check box and the server-side logic **true** or **false**

It sends values. IsSelected understood by referring to the method by which it is selected CheckBox component on the side of the controller.

View

```

<! DOCTYPE HTML PUBLIC "-// W3C // DTD HTML 4:01 Transitional // EN ">
<Html>
    <Head> <title> Option </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<Faces>
    <@Import TagLib = "phpf.ui. "*" prefix = "f" /> Dşaret box <F: CheckBoxes id = "box" onclick = " ajaxevent " />

    Availability: <F: label id = "status" > </ F: label>

</ Faces>
</ Body> </
html>

```

contröler

```
<? Php
import ( "Phpf.controllers.facet to" );
class Example 7 extends Facet {
    the public function EXAMPLE 7 () {
        parent :: bezel ();
        $ this -> render ( "Ornek7.html" );
    }
    protected function kutuclicked ( $ incidents ) {
        $ this -> situations> text = $ this -> Box-> IsSelected ();
        $ this -> ajaxrespo possible ();
    }
}
?>
```

Radio

It works like Checkbox component.

Đmag to

Đmag component image (picture) is used to display user input, such as HTML files is image image. Đmag component to view the file shown in the src attribute of the desired image file path and name will be written.

Sample

```
<Faces>
<@Import TagLib = "phpf.ui. *" prefix = "f" />
<F: image name = "picture" src = "Images / resim1.png" />
</ Faces>
```

On the side of the controller component image setsour (string filename) file name can be changed using the method of the src attribute.

Example AJAX image gallery

photo1.jpg pictures in the folder in this example, photo2.jpg, 3,4,5 an image component name in to view the picture are 5 photos and in our view the file, there is a label and 5 button component in the change of image.



My visions

```

<Html>
    <Head> <title> Photos </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
    <@Import prefix = "f" taglib = "phpf.u in. *" />
    <F: image name = "picture" src = "Images / photo1.jpg" height = "250" /> Reviews

    Displayed: <F: label name = "tag" > picture 1 </ F: label> Reviews

    <F: button name = "one" text = "one" onclick = " ajaxevent " /> <F: button name = "two" text = "2nd" onclick
= " ajaxevent " /> <F: button name = "three" text = "3" onclick = " ajaxevent " /> <F: button name = "four" text = "4"
onclick = " ajaxevent " /> <F: button name = "five" text = "5" onclick = " ajaxevent " /> </ Faces> </ body> </
html>

```

ActionPerformed method was override controller.

```

<? Php
import ( "Phpf.controllers.facet to" );
class Photos extends Facet {
    the public function Photos () {
        parent :: bezel ();
        $ this -> render ( "Resimler.html" );
    }
the public function actionPerformed (ActionEvent $ event ) {
    $ number = $ event -> getcomponent () -> getText ();
    $ this -> image-> setSource by ( "Images / pictures $ number .jpg " );
    $ this -> Label-> setText ( "Picture $ number " );
    $ this -> ajaxrespo possible ();
}}?>

```

Link

Link component is the equivalent of html tag and is used as a value of this tag is the content of the above gallery view files using the example of making the link.

```

<! DOCTYPE HTML PUBLIC "-// W3C // DTD HTML 4:01 Transitional // EN ">
<Html>
    <Head> <title> Option </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
    <@Import prefix = "f" taglib = "phpf.u in. *" />
    <F: image name = "picture" src = "Images / photo1.jpg" height = "250" /> Reviews

    Displayed: <F: label name = "tag" > picture 1 </ F: label> Reviews <f: link name = "one" text = "one" onclick = " ajaxevent
" /> <F: link name = "two" text = "2nd" onclick = " ajaxevent " /> <F: link name = "three" text = "3" onclick = " ajaxevent " />
<F: link name = "four" text = "4" onclick = " ajaxevent " /> <F: link name = "five" text = "5" onclick = " ajaxevent " /> </
Faces> </ body> </ html>

```

Message

Label component is generally used in a similar way to validate (validation) which will be mentioned in the Verification section for use in the process.

Girdler

Models describing the database will be discussed.

File Upload Component

File component will create a file upload box in the browser and performs the upload process on the server sent with the file upload method. File label nature of the path to the server should be written trafnsf is the directory where the file is recorded.

Sample File Upload

```
<Html>
  <Head>
    <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

  </ Head>
  <body>
    <faces>
      <@Import prefix = "f" taglib = "phpf.u in. *" />
      <F: form id = "form" > <F: file id = "file" path = "Pictures" /> <F: button id = "B" text = "Load" onclick = " ActionEvent " fom = The "form" /> </ F: form> </
    faces> </ body> </ html>
```

```
<? Php
import ( "Phpf.controllers.facet to" );
class UploadFile extends Facet {
  the public function UploadFile () {
    parent :: bezel ();
    $ this -> render ( "Yukle.html" );
  }

  function buttonclicked ( $ evt ) {
    $ this -> resimler-> upload ();
  }
}

?>
```

If you wish to file component with certain file formats with criteria such as the installation process, you must use the filevalidatio these issues will be addressed in the validation title in the following pages.

Access to the components of Quality Control

When accessing an object name qualifications -> Download the form attribute name. For example, when the color of the transform TextBox TextBox clicked a button in the button get our component. faces view definition is as follows.

```
<Faces>
<@Import TagLib = "phpf.ui. *" prefix = "face" />
<Face: TextBox name = "article" text = "Hello there" /> <Face: TextBox name = "Button" text = "Change
Color" /> </ Faces>
```

```
<? Php
class Controller extends Facet {
    function Controller () {
        parent :: facescontrol on ();
        $ this -> render ( "View.html" );
    }
    function buttonclicked ( $ evt ) {
        $ this -> information is printed> setProperty ( "Style" , "Background: blue" );
    }
}

?>
```

In our example, the style attributes of the TextBox component background: We have transferred the blue value, and we have changed the color.

If you want to change an attribute

```
$ This-> kontrol_adi-> attribute = "value";
```

Or

```
$ This-> kontrol_adi-> setProperty ( "quality", "value");
```

Two applies in use.

Now you get a little more color in a ComboBox I should develop our example and let's apply the selected color using ajax textbox object.

I saw the file for our example is as follows.

```
<Faces>
<@Import prefix = "face" taglib = "phpf.u in. *" />
<Face: TextBox name = "TextBox" text = "Hello there" />
Change Color
<Face: ComboBox name = "Colors"
    binder = "Array (Aqua, Blue, aliceblu to, Beige, Black, Brown
, Navy, Pink, Red, Orange, Yellow, White, Green) "
    onchange = " ajaxevent " >
</ Face: ComboBox> </
faces>
```

Controller for our example file is as follows

```
<? Php
import ( "Phpfc.controllers.facet to" );
class Style extends Facet {
    function Style () {
        parent :: bezel ();
        $ this -> render ( "View.phpf" );
    }
    protected function colorschanged ( $ evt ) {
        $ Color = $ this -> Colors-> getselected ();
        $ this -> textbox-> setProperty ( "Style" , "Background: $ Color " );
        $ this -> ajaxrespo possible ();
    }
}

?>
```

Component Architecture (your own components Develop)

One faces component (component) you can expand your class from the Component class to create.

at least two components of a simple method must be found. Founder method and StartTag method.

Founder method has two parameters facescontrol first parameter a second parameter is defined as attributes of the component tag of the file is a sequence view.

The constructor function (facescontrol's & \$ controller, \$ args = null)

StartTag producing method is a method to start the output of the component tag. A tag is executed when opened.

method that is run when the method EndTag component tag is turned off.

A simple example

```
<? Php
import ( "Phpfc.ui.component" );
class Kalinyaz extends Component {
    function Kalinyaz (facescontrol's & $ Controller , $ args = null ) {
        parent :: Component ( $ Controller , $ args );
    }
    function StartTag () {
        return '<Strong> ; }
    }

    function EndTag () {
        return '</ Strong> ; }
    }

?>
```

In our example, we define a component name Kalinyaz <f: Kalinyaz> </ f: Kalinyaz> We want to show the text between the tags in bold type in the browser. We are registered with the directory name into kalinyaz.php we want to use our component.

Kalinyaz components to use in the view.

```
<Faces>
  <@Import prefix = "my" taglib = "dizinadi.kalinyaz" />
  <My: kalinyaz name = "thick" >
    This text should be bold
  </ My: kalinyaz>
  This text normal
</ Faces>
```

Qualifications in access to components.

Component attributes and attribute values includes a series of members of attributes comprising

\$ This-> attributes [were nitelika] accessible form.

Let's make a component that implements the text color and text size to develop a little more our example above. Consider the following example

```
<? Php
import ( "Php.ui.component" );
class Kalinyaz extends Component {
  function Kalinyaz (facescontrol's & $ Controller , $ args = null ) {
    parent :: Component ( $ Controller , $ args );
  }
  function StartTag () {
    $ html = '<Strong> . '<Font color = "" . $ this -> attributes [color].
      ' "Size =" . $ this -> attributes [size]. ' "> ;
    return $ html ; }

  function EndTag () {
    return '</ Font> </ strong> ; }}

?>
```

The above named component to our size and color can be seen that there are two qualifications.

In our view the use of the component.

```
<Faces>
  <@Import prefix = "my" taglib = "dizinadi.kalinyaz" />
  <My: kalinyaz name = "thick" size = "10" color = "rejection" >
    Any text
  </ My: kalinyaz> </ faces>
```

doattributes of the Component class () method places all the attributes with equal name

```
function StartTag () {
    return '<Strong> . '<Font' . $ this -> doattributes (). '>' ; }
```

Consider the following example for use in dizins u place gets Items items items added to your component Components

```
<? Php
import ( "Php.ui.component" );
class Stronger extends Component {
    function Strong (facescontrol's & $ Controller , $ args = null ) {
        parent :: Component ( $ Controller , $ args );
    }
    function StartTag () {
        $ html = '<Strong> . '<Font' . $ this -> doattributes (). '>' ;
        foreach ( $ this -> items ace $ Item )
            $ html . = '<br/> <a href = "" . $ Item [Url]. ' "> . $ Item [Title]. '</a> ;
        return $ html ; }

    function EndTag () {
        return '</ Font> </ strong> ; }

?>
```

In use View

```
<Faces>
<@Import prefix = "my" taglib = "mypath.strong" />
<My: strong name = "test" size = "4" face = "Arial" color = "rejection" >
    <@lte url = "http://www.google.co l" title = "google" /> <@ite url = "http://www.webmahsulleri.co l" title = "webmahsul of" />
    some photogallery

</ My: strong> </
faces>
```

Static class labels (your own label)

Core classes in the early narratives of the document also referred to as static labels does not create a new object to produce a string of rendering information to the controller output. inaccessible by controler for classes are not static components. static class is encoded in the tag it consists of static methods. This label is given to the static name to this reason.

The method they will use for the label names of static class label is important. The name of the static method which will work when the label opened **start tag name Tag** And it must accept the form of an interface array parameter.

```
static function startetiketag ( $ args )
```

The name of the static method which will work when the tag is switched off **end tag name Tag** It shaped.

```
static function endetiketag ()
```

Sample :

The following example phpfaces / phpf directory on trial. Named php will be eligible to be registered.

```
<? Php
class {trial
    static function startkalintag ( $ args ) {
        $ html = '<Strong> . '<Font color = "" . $ args [color].
            ' "Size =" . $ args [size]. ' "> ;
        return $ html ;

    }
    static function endkalintag () {
        return '</ Font> </ strong> ; } }
```

My visions of the sample used in the experiment class

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
    <Head> <title> Stop Label </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>

    <@Import prefix = "c" taglib = "phpf.den up" type = "static" />
    <C: thick color = "Blue" size = "10" > me too </ C: thick> </ faces> </ body> </ html>
```

Static experiment two new classes are added to a plurality of methods may comprise the following method.

```
<? Php
class {trial
    static function startkalintag ( $ args ) {
        $ html = '<Strong> . '<Font color = "" . $ args [color].
            ' "Size =" . $ args [size]. ' "> ;
        return $ html ;

    }
    static function endkalintag () {
        return '</ Font> </ strong> ; } }

    static function startrenkitag ( $ args ) {
        $ Color = array (Blue, aliceblu, Black, Brown, Pink, Red, Orange, Yellow, Green);
        $ writings = Trim ( $ args [article]);
        $ n = Strlen ( $ writings );
        $ html = "" ;
        for ( $ i = 0; $ i <= $ n ; $ i ++ ) {
            $ key = Array_rand ( $ Color );
            $ you = Rand (1,10);
            $ html . = '<Font size = "" . $ you . ' "
color = ' . $ Color [ $ key ] . ' "> .htmlspecialchars ( $ writings [ $ i ] ). "</ Font>" ;
        }
        $ html ;
        return $ html ; }

    static function endrenkitag () {
        return " " ; }?>
```

use within Outlook

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<Html>
    <Head> <title> Option </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
    <@Import prefix = "c" taglib = "phpf.den up" type = "static" />
    <C: thick color = "Blue" size = "10" > thick Summer </ C: thick> Reviews

    <C: color writing = "Colorful Summer" /> </ Faces> </ body> </
html>
```



Template engine

The main purpose of your use other code with HTML Vs Template Tags is to ensure the separation. PHP because it faces the appearance of understanding the labels found running on the server you use too much can not search the template, the usual.

Earlier in the FDL in the statements such as PHP Faces bahsedilgib #} {aralag the mania was written by the invisible Listen to Me is the case or in other way. You can access this key appearance in the public and protected members. You can use the variables are not members of your class or your LAN Controller append method of transferring data view.

The second parameter is a string variable name for the view of the first parameter is the actual data append method.

Examples

```
$ This-> append ( "name", "Huseyin Bora"); $ Name =
"Abaci"
$ This-> append ( "name", $ name); $ This->
append ( "sai", 1001);
```

You can access the above, in view of the belirtiğim as public and protected members. \$ This after the switch. "" You could use Point. Consider the following example

EXAMPLE controller

```
<? Php
import ( "Phpf.controllers.facet to" );
class Example 2 extends Facet {
    the public $ uye1 = "General" ;
    protected $ uye2 = "Storage" ;
    the public function Example2 () {
        parent :: bezel ();
        $ this -> append ( "name" , "Huseyin Bora" );
        $ this -> append ( "surname" , "ABACI" );
        $ this -> append ( "number" , 1001);
        $ this -> append ( "sequence" , array (20,2,123,4100,23,0,1));
```

```

    $ this -> render ( "Sayac.html" );
  }}

?>

```

View sample template

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
  <Head> <title> Template </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import prefix = "x" taglib = "phpf.cor" type = "static" /> Name Title: # {$ name}

  Reviews
  Name: # {$ name}
  Reviews
  Sai: # {} $ sai
  Reviews
  Public ua: # {} $ this.uye1
  Reviews
  Protected u: # {} $ this.uye2
  Reviews
  Sequence
  <X: foreach var = "$ Array" item = "A $ og" > <P> Value: $ {#} og the </
P> </ x: foreach> </ faces> </ body> </ html>

```

View state can use

Viewstate What is it? Among the data you want to lose your postback is a php PHP pages for your storage container faces.

What is a postback? go to the page server is coming again.

Through the use of view state persistence you can make your appearance files. View your page state's main purpose is to ensure no loss of data when using the post.

View state is a data storage structure asp.net users have in addition to this they are familiar with Java JSF is also a similar structure. View state structure known as cumbersome structure, but there is no method has been developed for more convenient now. So I suggest you want your data in PHP or large size when using a statin use asp.net Whether you not keep the JSF Viewer may cause worsening of this page.

PHP Faces ViewState definitions are reported to the controller with the directive @definitio

```

<@Definitio the ViewState = "true" to stroke = "xhtml" />

```

definition ViewState to the first character of the directive = "true" you also have to make ViewState is enabled.

We give you the qualifications you have symptoms of stroke oluşturulacak değeril data storage for these areas **xHTML** ie in the browser or client-side **session** session on the server side may file.

ensure the permanence as we asked components defining **placeholder** nature of **true** You must give its value.

View state in a corner of your page as an example that should be given to the use of usernames think.

You can use this process to the table in the database instead would Viewer every time the user name of state.

Sample

```
<Faces>
<@Import prefix = "phpf" taglib = "phpf.u in. *" /> <@Definitio the ViewState = "true" to stroke
= "session" />
<Phpf: label name = "Member Name" placeholder = "True" > </ Phpf: label> </ faces>
```

uyea was a time we update the courses controler is sufficient.

```
$ This-> uyeadi-> setText ( "XXX Hosgeldin");
```

Validations (Validation)

Verification identification view (appears) is provided in making appointments to the nature of the components. Validation results of the event (the event) depending on the nature of the object is sent to the browser AJAX by post or page.

The nature of the components found in the verification process 5.

These

- * validator
- * rule
- * messagef is
- * message
- * success

Validator: this nature should be given the name of the hosting static method validation class. There are two verification ion directory in the filter and validator class ready isimlerim.

Rule: Validator attribute is the name of the method used for verification of the declared class righting operation.
(Verification method)

Message: Verification of the results of the (false) is a warning message that will be given in case of wrong. For example, "You have entered incorrect e-mail address! "

Success verification of the results of the (true) if there is a message that will be true. . For example, "Your email address has been verified! "If you do not wish to be confirmation messages can be left empty this nature.

Messagef are: message and that message may appear in which components with success.

facescontrol are not operating in the default state is the result of a false event validation methods. If you want the result of the verification process, the operation of a state Clicked method for example, it is wrong facescontrol s setvalidcallback (boolean); Simply not true method.

The results of the validation process that can be understood by looking at the components of true or false value returned by the isValid method.

Example entered by the user to verify the e-mail address.

E-Mail : Hatalı bir email adresi girdiniz !

Viewer view file

```
<Html>
  <Head> <title> the Validation </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import prefix = "f" taglib = "phpf.u in. *" />
  <F: form method = "Post" id = "form" >
    E-mail: <F: TextBox id = "e-mail"
      validator = "Validator"
      rule = "Mail"
      message = "You have entered an incorrect email address!"
      messagef R = "Msg"
    />
    <font color = "rejection" > <F: message id = "Msg" /> </ Font> <f: button id = "Btn" text = "Gonder" onclick = " ajaxevent " form = The "form" />
  </ F: form> </ faces> </ body> </ html>
```

controler

```
<? Php
import ( "Phpf.controllers.facet to" ); import ( "Io.validat is" );

class verification of extends Facet {
  the public function Verification of () {
    parent :: bezel ();
    $ this -> render ( "Dogrula.html" );
  }
  protected function btnclicked ( $ evt ) {
    $ this -> msg-> setText ( "Your email address has been verified" );
    $ this -> ajaxrespo possible ();
  }
}??>
```

In the latter case of the above code in an email address is entered erroneous btnclicked method is not executed mail this method will work.

E-Mail : Email adresiniz Doğrulandı

The verification process results in true success is indicated by the MsgBox component false information specified by the message takes place.

Filter class of methods is as follows.

- required
- mail
- URLs
- rope
- float
- int
- boolean
- regxp

Validator class of methods is as follows.

- required
- eQuaLs
- minLength
- Maxlength
- mail
- alpha
- alpha_numeric
- numeric
- integer
- betweenlength
- between
- boolean
- test

filevalidat Class

- Length
- Types
- exists

Authentication Uses (Validator class)

Be included in your application validation checks sınıfların I gerekmektedir. validat class phpfaces / ion / directory is in the. Remember to include your controller following the row for examples.

```
import ( "Io.validator is" );
```

On request checks whether the content of the post-exchange resin Sample blank

```
name: <F: TextBox name = "TextBox"
                                validator = "Validator" rule = "Required" message = "This
empty space impassable " messagef R = "MsgBox" /> <F: message name = "MsgBox"
/>
```

eQuals checks whether the pump are in line with the test sample match the components of the content attribute of the post

```
name: <F: TextBox name = "TextBox"
                                validator = "Validator"
                                rule = "Equals"
                                test = "200"
                                message = "Error 200 Not entered value"
                                messagef R = "MsgBox" />

<F: message name = "MsgBox" />
```

minLength Is that the nature of the components of the post-character checks whether the content is large Example

```
name: <F: TextBox name = "TextBox"
                                validator = "Validator"
                                rule = "MinLength"
                                min = "10"
                                message = "Up to 10 characters"
                                messagef R = "MsgBox" />

<F: message name = "MsgBox" />
```

Maxlength checks whether the contents of the components of the post is smaller than the maximum character length attribute Example

```
name: <F: TextBox name = "TextBox"
                                validator = "Validator"
                                rule = "MaxLength"
                                min = "5"
                                message = "At least five characters must be entered!"
                                messagef R = "MsgBox" />

<F: message name = "MsgBox" />
```

mail Example checks whether an email address is the content of the post as components

```
name: <F: TextBox name = "TextBox"
      validator = "Validator"
      rule = "Mail"
      message = "This is not an e-mail address!"
      messagef R = "MsgBox" " " />

<F: message name = "MsgBox" />
```

Alpha the post component is a verbal content (string) controls whether. Sample

```
name: <F: TextBox name = "TextBox"
      validator = "Validator"
      rule = "Alpha"
      message = "The value entered is not verbal!"
      messagef R = "MsgBox" " " />

<F: message name = "MsgBox" />
```

alpha_numeric The content of post checks whether the components that might have occurred and the number of characters. Sample

```
name: <F: TextBox name = "TextBox"
      validator = "Validator"
      rule = "Alpha_numeric"
      message = "The entered value should include the number!"
      messagef R = "MsgBox" " " />

<F: message name = "MsgBox" />
```

numeric Of post checks whether the content of the components a numerical value. Sample

```
name: <F: TextBox name = "TextBox"
      validator = "Validator"
      rule = "Numeric"
      message = "The value you entered is not a number Warning!"
      messagef R = "MsgBox" " " />

<F: message name = "MsgBox" />
```

integer The contents of the components post checks whether a whole number. Sample

```
name: <F: TextBox name = "TextBox"
      validator = "Validator"
      rule = "Integer"
      message = "The value you entered is not an integer Warning!"
      messagef R = "MsgBox" " " />

<F: message name = "MsgBox" />
```

betweenlength Min post of the length of the components of the content and checks whether the value of max.

Sample

```
name: <F: TextBox name = "TextBox"
      validator = "Validator"
      rule = "Betweenlength"
      min = "5"
      max = "20"
      message = "Less than twenty-five characters from the character data to enter long!"
      messagef R = "MsgBox" " " />

<F: message name = "MsgBox" />
```

Multiple Validations (Multi Validation)

Multiple verification is useful when you want to make multiple verification a component attribute values in multiple verification ", " it should be separated from one another by commas. For example, planning together and mail verification required.

```
<F: TextBox name = "e-mail"
      validator = "Validator"
      rule = "Mail, required"
      message = "** Incorrect e-mail address * This field is empty impassable"
      success = "The address is verified,"
      messagef R = "Msg1, msg2" />

<F: message id = "Msg1" /> <F: message id
= "Msg2" />
```

Mail verification before then required verification is done in this example. If the verification result is wrong both operated in both posts.

Email verification message of "msg1" the message authentication required "msg2" component printed on.

If the verification of the e-mail "address is verified" message is printed on the msg1 components.

And File Upload component with filevalidat

filevalidat is used together with the new file Bisel example the installation of files on the basis of certain size or may require the loading of specific file format. filevalidat class has three methods of length (acceptable maximum file size) tyeps (file formats accepted) exists (when the same file name, activate the başkab dos status)

Sample size is small and 50000 in GIF or JPG format to upload to the server

```
<Html>
  <Head>
    <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

  </ Head>
  <body>
    <faces>
      <@Import prefix = "f" taglib = "phpf.u in. *" />
      <F: form id = "form" >
```

```

        <F: file id = "net" path = "Images /"
length = "50000"
validator = "Filevalidat is" rule = "Types, length" types = "Image / jpeg, image / gif"
message = "Unsupported File Format, large file 50000 bytes diagnosis"
messagef R = "Error, Error" />
        <F: message id = "error" /> <F: button id = "B" text = "Load" onclick = " ActionEvent " form = The "form" /> </ F: form> </ faces> </ body>
</ html>

```

```

<? Php
import ( "Phpf.controllers.facet to" );
class UploadFile extends Facet {
    the public function UploadFile () {
        parent :: bezel ();
        $ this -> render ( "Yukle.html" );
    }

    function buttonclicked ( $ evt ) {
        $ this -> resimler-> upload ();
    }
}

?>

```

Confirmation Classes (Self Your Verifier)

Authenticator (Validator) classes stationary (static) methods classes are using. To facilitate your work, and if the integrated software components allows a structure to work collectively.

The identification of an authentication method are as follows:

static function required (\$ c Component, Component \$ m, \$ message, \$ success)

The first parameter of "Component \$ c" component of the verification process where the second parameter "Component \$ m" component of the messages will be printed warning message The third parameter is the fourth parameter validation message

For example, that we have a member registration form for new member registration and think that only one member can carry the same name.

I'm adding a verification method which creates a class named Uyekontrol Does the name for it

```

<? Php
class {uyekontrol
    static function Have (Component $ c Component $ m , $ Message , $ success ) {

```

```

    $ query = EntityManager :: getInstance () -> createquery ( "U select from alarm
where he uye.a =: uyead " );
    $ query -> BindParam ( "Member Name" , $ c -> gettext);
    $ query -> execute ();
    $ results = $ query -> getsingl A ();
    IF ( $ results != null ) {
        $ m -> setText ( $ Message );
        return false ; }

    $ m -> setText ( $ success );
    return true ; }}

```

?>

Now the view we form validation class (view) Let's use in

```

<Faces>
<@Import prefix = "f" taglib = "phpf.u in. *" />
<F: form method = "Post" name = "Uyeform" >
    Member Name : <F: TextBox
        name = "Member Name"
        validator = "Uyekontrol"
        rule = "Is there"
        success = "A name is verified"
        message = "Before recording became a member by that name, try a different name!"
        messagef R = "MsgBox"
    /> <F: message id = "MsgBox" /> <F: button name = "Save" text = "Send" onclick = " ActionEvent " form = The "Uyeform" /> </ F: form> </ faces>

```

We are importing our class is in the control Validad

```

<? Php
import ( "Phpf.controllers.facet to" ); import ( "Dbf.persisten by" ); import ( "Dogrulayicilar.uyekontrol" , true ); //
application directory

```

```

class Controller extends Facet {
    function Controller () {
        parent :: bezel ();
        $ this -> render ( "Uyekayit.php" );
    }
    function kaydetclicked ( $ evt ) {
        $ U = new member();
        $ U -> name = $ this -> uyeadi-> text;
        $ U -> save ();
    }
}

```

?>

Operating in Authentication to events

When false returns the result of the verification process, you may want to run the default case study method does not Ancei If you want to run your controller before rendering method in this case **setvalidcallback (boolean)**; You will need to run the method giving the true parameters.

The components related to the status of the verification process **isValid ()** You can find out by looking at the method. If the accuracy of this method one of the provided data is not supplied false to true.

Example view

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
  <Head> <title> verification of </ Title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

</ Head>
<body>
<faces>
  <@Import prefix = "f" taglib = "phpf.u in. *" />
  <F: form id = "form" method = "Post" > <F: TextBox name = "e-mail"

        validator = "Validator"
        rule = "Required" />

  <F: message id = "Msg1" /> <F: button id = "Btn" onclick = "ajaxevent " form = The "form" text = "Gondor" /> </ F: form> </ faces> </ body> </
html>
```

Example controler

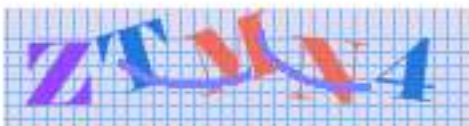
```
<? Php
import ( "Phpf.controllers.facet to" ); import ( "Io.validat is" );

class verification of extends Facet {
  the public function Verification of () {
    parent :: bezel ();
    $ this -> setvalidcallback ( true );
    $ this -> render ( "Dogrula.html" );
  }
  protected function btnclicked ( $ e ) {
    iF (! $ this -> email-> isValid ())
      $ this -> msg1-> text = "This area is empty impassable" ;
    $ this -> ajaxrespo possible ();
  }
}

?>
```

Captcha Verification Components Photos

Captcha is an image that contains a component of karate and prompts the user for verification. This component should be used in conjunction with verification and validation.

İsim	<input type="text"/>
Soyisim	<input type="text"/>
	
Resimdeki kodu giriniz	
<input type="text"/>	<input type="button" value="Gonder"/>

Example Captch Security code

```
<Html>
    <Head>
        <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

    </ Head>
    <body>
    <faces>
        <@Import prefix = "f" taglib = "phpf.u in. *" />
        <F: form method = "Post" id = "form" > <Table>

            <Tr>
                <Td> My Jesus </ Td> <td> <F: TextBox id = "name" /> </ Td>
            </ tr> <tr>

                <Td> Last name </ Td> <td> <F: TextBox id = "last name" /> </ Td>
            </ tr> </ table> Reviews

        <F: captcha id = "Captcha" /> <F: message id = "Msg" text = "Please enter the code in the picture" />
    </ Br>

    <F: TextBox id = "Code" validator = "Validator" rule = "Equals"
        test = "# {} $ In this.captcha.co"
        message = "You entered the wrong code in the picture!"
        messagef R = "Msg"
    />
    <F: button id = "B" text = "Gondor" onclick = " ActionEvent " form = The "form" /> </ F: form> </ faces> </ body> </ html>
```

Load Method of Dynamic User Objects

Facet facescontroller loader attachments and method generates an instance of the class file with the name without the first parameter with the same name and the constituent parameters and Controller as a public member. Dmport user has a similar function.

Sample

```
<? Php
import ( "Phpf.controllers.facet to" ); import ( "lo.validat is" );
```

```
class Example 2 extends Facet {
    the public function Example2 () {
        parent :: bezel ();
        $ this -> loader ( "The server.sessio" );
        $ this -> session-> get ( "Userid" );
    }
}
```

?>

You can specify the name of the new object to be created. The second parameter is the name of the Load method when u will be created.

Sample

```

<? Php
import ( "Phpf.controllers.facet to" ); import ( "Io.validat is" );

class Example 2 extends Facet {
    the public function Example2 () {
        parent :: bezel ();
        $ this -> loader ( "The server.sessio" , "session" );
        $ this -> hearings> get ( "Userid" );
    }
}

?>

```

The third parameter of the load method accepts one of true or false. This parameter is viewed in the file relating to the application directory if given true.

The URI Class

URI class to obtain data from the URL line-shaped path and directory is used to redirect the page.

required to access the controller, considering that we have a controller named Page

URL line with

The following will be in a similar manner.

http: // localhost / phpfaces / page

After the controller name in the URL line "/" character between data and parameters considered "/" number of created a set up. for example **http: // localhost / phpfaces / page / 1** ***Trailing 1 parameter is 0'inc indices.***

Sample

```

<? Php
import ( "Phpf.controllers.facet to" );
class Page extends Facet {
    the public function Page () {
        parent :: bezel ();
        $ this -> loader ( "The io.ur" );
        echo "Parameter 0 =" . $ this -> uri-> get (0);
    }
}

?>

```

for example ***http: // localhost / phpfaces / pages / books / beyaz_ge my / 1125145*** Consider the urlsatır

Books index to 0. In my

Beyazge 1.indis

1125145 in 2.indis

Sample

```

<? Php
import ( "Phpf.controllers.facet to" );
class Page extends Facet {
    the public function Page () {

```

```

parent :: bezel ();
$ this -> loader ( "The io.ur" );
echo "Category =" .           $ this -> uri-> get (0);
echo "Book name =" .         $ this -> uri-> get (1);
echo "ISBN =" .              $ this -> uri-> get (2);

}}

```

?>

The redirect URL as the URI class method parameter tells the browser to redirect.

Sample

```

<? Php
import ( "Phpfc.controllers.facet to" );
class Page extends Facet {
    the public function Page () {
        parent :: bezel ();
        $ this -> loader ( "The io.ur" );
        $ This-> uri -> redirect ( "Www.google.com" );
    }
}

```

?>

Head and object rendering process

facesrender faces class parses the file to the <head> tag automatically when it encounters a Header object created as a result of the rendering process also returns this header object. Head object page <head> </ head> is used to make additions to the range. Header class of methods is as follows

- addmet (string name, string content) visions to add a meta tag
- addscript (string file) adds script tag with the specified file name with visions file
- addle Link (string file) adds visions link tag with the specified file name and file a rel = "stylesheet" type = "text / css" uses nature
- SetTitle (string title) changes the page title
- addtext (string text) text parameter of the <head> </ head> add tags.

Sample

```

<? Php
require_once ( "Config.php" ); import ( "Phpfc.controllers.facet to" );

class Head extends Facet {
    the public function Head () {
        parent :: bezel ();
        $ this -> render ( "View.html" );
    }
    function dugmeclicked ( $ e ) {
        $ header :: = facesrender it is getHeader ();
        $ header -> SetTitle ( "I changed the title on click on the button" );
    }
    the public function PreRender () {
        $ header :: = facesrender it is getHeader ();
        $ header -> addmet to ( "Description" , "Faces of PHP MVC Framework" );
        $ header -> addmet to ( "Keywords" , "PHP, ORM, MVC, AJAX" );
        $ header -> addscript ( "Scrip.js" );
        $ header -> addle Link ( "Style.css" );
    }
}

```

```

}
Dispatcher :: dispatchclass ( "Header" );
?>

View
<Html>
    <Head>
        <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

    </ Head>
    <body>
        <faces>
            <@Import TagLib = "phpf.ui.butto the" prefix = "f" />
            <F: button name = "button" text = "Tıkla1" onclick = " ActionEvent " /> </ Faces> </ body> </ html>

```

Renderer Directives Directives Dışleyic

all the directives in the renderer belirttiğim FDL as in previous writings begin with the @ sign. These

- * the @definitio
- * @import
- * @pattern
- * the @u
- * @htmlpattern
- * @html
- * @set
- * @get
- * I @ite
- * in @inclu
- * @fa by

the @definito

There are three qualities of this directive

- viewstate: true or false receive one forms a storage space for the component data in the view data when the postback sayde page is permanent. asp.net viewstate value of e has a similar structure. Gerekeli is used with caution. very large data can slow down the processing power. The default value is false dr
- stroke A: where the data will be kept until viewstate viewstate used in conjunction with XHTML or receive one session value. xHTML browser on this attribute value is been given to the client-side storage area. session in the case of server-side session files.

- EVENTVALIDATION: a time to the event from the timeout and checks the accuracy of the Tarfa client with a key. The default value is false dr

Sample

```
<Faces>
  <@Definitio the ViewState = "true" to stroke = "xhtml" /> <@definitio the
  EVENTVALIDATION = "true" />
</ Faces>
```

@import

There are 3 qualities.

- prefix: name space to be used with tags
- TagLib: to be or php file and import directory
- type: the type of component libraries

This directive import a method of operating a renderer, says the association with the name associated with the prefix.

The type attribute is used only in static class and PHP Faces in these single static library dir core.php

Sample

```
<Faces>
  <@Import prefix = "c" taglib = "phpf.cor" type = "static" />
  <C: out value = "Hello Faces" /> </ Faces>
```

Another example

```
<Faces>
  <@Import prefix = "c" taglib = "phpf.cor" type = "static" /> <@import prefix = "f" taglib = "phpf.u in. *" />

  <C: if test = "1 <2" > <C: out value = "It is smaller than one or two" /> </ C: if> <f:
  button name = "Button" text = "click" /> </ Faces>
```

@set

@set directive to produce a variable and assign a value to the interpreter faces are the qualities that söyler.i.

The first character of the name of the variable to be produced

Where's second nature that it will be stored in this variable. page or receive one session value.

Usage is as follows.

```
<@Set sai = "2008" scope = "page" />
```

SA created a variable named line with the above, this variable is assigned a value of 2008, and it is stored in the current script is valid.

```
<@Set name = "gust" scope = "session" />
```

the above line with a variable named name is created, this variable is assigned a value bora and stored in log files on the server.

@get

See faces interpreter directive to generate b variable and the value of this variable tells the assignment of another variable. There are three qualities. The first attribute is the name of the variable nature have been reported. The second quality is the quality which you assign variables occur under the SELECT noted. The third attribute is the scope attribute containing the variable location of the session or receives one page value.

```
<@Get var = "istesat" select = "sai" scope = "page" /> <@get var =  
"isteis l" select = "name" scope = "session" />
```

in @inclu

This directive tells the renderer a view to adding a file may be in php file

Usage is as follows.

```
<In @inclu file = "functions.php" /> <@inclu  
file = "helper.php" /> <in @inclu file =  
"view.php" />
```

@fa by

This directive tells the renderer a view to adding the processing faces a file

Usage is as follows.

```
<@Fa by file = "patterns.php" /> <@fa by  
file = "footer.php" /> <@fa by file =  
"banner.html" />
```

/ @ite

@ite directive adds new items to a component label. @ite is used in another label. and it accommodates the unique niteliklr.

Examples of use

```
<SQL: SQL name = "SQL" var = "Results" query = "Select * from table where id =: id" >
  <@Ite My id = "$ {id}" type = "integer" />
</ SQL: SQL>
```

```
<Com: Grid name = "Gridx" height = "one hundred" width = "50" binder = "$ This.b are" border = "one" >
  <@Ite I input = "text" title = "id" key = "id" /> <@ite I input = "button" key = "name" title =
  "name" />
  <@Ite I input = "link" title = "delete" url = "delete / $ id / $ name / $ id" type = "path" separator = "_" />

  <@Ite I input = "link" title = "edit" url = "delete.php? Id = $ id & amp; name = $ name" />
</ Com: grid>
```

patterns

The concept is an innovation in terms of programming patterns in markup languages. and these methods are introduced for the first time with PHP faces.

pattern creates a template for summarizing data in a form of identification, and ensure the application of this definition

More understandable way. Different sees a copy and paste operation. With labels resembling the pattern on birir programmer (tag) is released from the block again.

There are 2 kinds of pattern in PHP faces them:

@pattern

@htmlpattern

@Pattern

Faces components and used with PHP @pattern for applying @u directives tag.

@Htmlpattern the text in a slightly different form in the copy into HTML, XML, etc. can be placed.

@htmlpattern used with @html directive tag. @pattern

@pattern tag directive has three nature of pleasure itself. After these qualities are qualities unique qualities to the future expansion of the component faces.

- name: the nature of the defined pattern (template) is the name.
- prefix: informs nature genişletilecek component which is in a name space
- extends: quality of components is the name of the faces to be expanded.

Sample

```
<Faces>
  <@Import prefix = "f" taglib = "phpf.ui.butto's" /> <@pattern name = "myButton" prefix =
    "f"

    extends = "button"
    value = "any thing" />
</ Faces>
```

Defined a new pattern name myButton button component from the example above. the @u

- It applies a template defined by the directive @pattern restoration stick. There are the two qualities.
- name: The name of the new component to be created.

pattern: is the name that will be used to define the pattern.

Implementation Example patternn

```
<Faces>
  <@U name = "mybutton1" pattern = "myButton" /> <@u name = "mybutton2"
  pattern = "myButton" />
  <@U name = "mybutton3" pattern = "myButton" value = "different text" /> <@u name = "mybutton4" pattern = "myButton" style = "color: navy;
  font-weight: bold;" >
</ Faces>
```

1 and 2 above line in myButton pattern from the two samples produced third satırt also myButton the value attribute produced a new example of the pattern "a different font appointed 4 new sample from the myButton pattern the rows produced style attribute of color: navy; fontWeight: bold; value is assigned.

ultimately html output is as follows.

```
<input type = "Button" name = "Mybutton1" id = "Mybutton1" value = "anything" /> <input type = "Button" name = "Mybutton2" id = "Mybutton2" value = "anything" />
<input type = "Button" name = "Mybutton3" id = "Mybutton3" value = "A different font" /> <input type = "Button" name = "Mybutton4" id = "Mybutton4" value = "anything"

style = " color: navy; font-weight: bold;" />
```


@htmlpattern

Copy the text from the template and the template is applied into htmlpattern as above belirttiğim with @html directive.

name attribute is the name of the pattern formed. @html mentioned here direktif tag name is written on the nature of the pattern.

Example identification @htmlpattern

```
<Faces>
  <@Htmlpattern name = "html">
    <div style = " color: red; font-weight: bold; " > Write your </ Div> </ @htmlpattern>

</ Faces>
```

Implementation of htmlpattern Example @html

```
<Faces>
  <@Html pattern = "html" />
</ Faces>
```

The above example will output would be as follows.

```
<div style = " color: red; font-weight: bold; " > Write your </ Div>
```

You could use any number of paternn

Implementation of htmlpattern Example @html

```
<Faces>
  <@Html pattern = "html" /> <@html pattern =
  "html" /> <@html pattern = "html" />

</ Faces>
```

The above example will output would be as follows.

```
<div style = " color: red; font-weight: bold; " > Write your </ Div> <div style = " color: red; font-weight: bold; " > Write your </
Div> <div style = " color: red; font-weight: bold; " > Write your </ Div>
```

html pattern can be used together with PHP PHP ifadeleri faces faces expressions #
{and} is written into the interval.

Another example of an identification htmlpattern

```
<Faces>
  <@Htmlpattern name = "HTML2">
    <div style = " color: red; font-weight: bold; " > tests htmlpattern # {$ i} </ Div> </ @htmlpattern>

</ Faces>
```

the implementation of the above pattern

```
<Faces>
  <C: fr var = "$ I" begin = "one" to = "5" step = "one" >
    <@Html pattern = "HTML2" />
  </ C: fr> </
faces>
```

For example, output

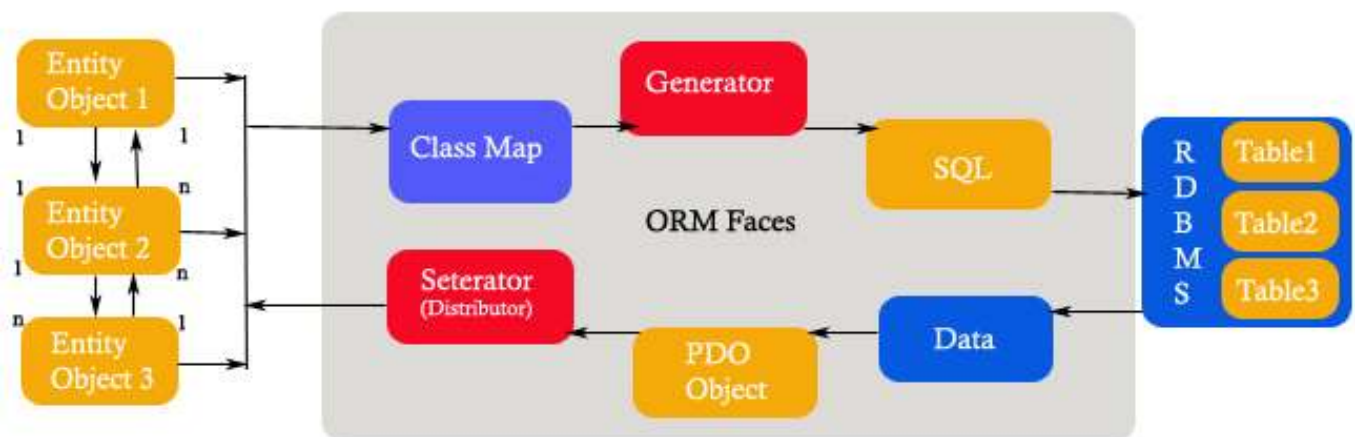
```
<div style = " color: red; font-weight: bold; " > test htmlpattern 1 </ Div> <div style = " color: red; font-weight: bold; " > test htmlpattern
2 </ Div> <div style = " color: red; font-weight: bold; " > test htmlpattern 3 </ Div> <div style = " color: red; font-weight: bold; " > test
htmlpattern 4 </ Div> <div style = " color: red; font-weight: bold; " > tests htmlpattern 5 </ Div>
```

ORM Faces

PHP ORM Faces

What is ORM? Object Relational Mapping (Object Relation Mapping) corresponding to the table in the database with a simple narrative of classes and properties of this class is based on a software architecture which can generate SQL statements can say.

In general, this class corresponds to our table **Presence(Entity)** We call classes. In object-oriented programming it should be considered as an object We have to consider everything as objects of our operations in the state database. nesneleştirme our tables and in our ORM us here in Doing Business database **CRUD** (create, read, update, delete) in our operations allow us a kind of simplification.



Below are some small examples

```
$ User = new User ( "Username" , "Password" );
```

```
$ User -> save ();
$ User = EntityManager :: getInstance () -> find ( "User" ,12);
$ User -> delete ();
```

configuration database

applications / Open config.php file with a text editor to edit the PHP Faces here ORM database access configuration in the edit line according to their sisteminz bulunur.aşa.

```
treasure( "DB_CONNECTION_STRING" , "Mysql: host = servename; dbname = database" ); treasure( "DB_USER" , "user name" ); treasure( "DB_PASS"
, "password" );
```

Faces Entity (Assets)

Entity Entity Entity class are extended from the class. Usually you do in the import control procedures. controller in the presence of the following line is good.

```
import ( "Dbf.persisten by" ); import ( "Models" , true );
```

Giving the import of second parameter function to the true situation in this function will be included in the application file application / ApplicationName was / is looked at in.

Let's go back to that again konumz the asset class. Your table in the database, you use the asset classes of associating an Annotation When mapping.

Annotation on

@Tabl (name = "blog") The name of the class defined table is overwritten words Name: parameter name is written on the table

@Column (name = "name" type = "string") field identification is performed

Name: nature of the domain name is written. yazılmal on private members are associated. members must be private because it is important to recognize that these ORM Faces private members.

@ID Defines the primary key

Annotation and relational

@onetoo to

@onetomany

There are three parametres

mappedby = "Class name"

pk = "primary key"

fk = "foreign key"

See the example for you to understand better the event.

```
CREATE TABLE `blog` (  
  `id` int (11) NOT NULL auto_increment, `name`  
  varchar (255) default NULL, `content` text,  
  
  PRIMARY KEY ( `id` )  
)
```

We have a table as above.

```
<? Php  
/ **  
 *   @Tabl (name = "blog")  
 */  
class Blogs extends Entity {  
/ ** @id  
  
  @Column (name = "id" type = "integer")  
  */  
  
  private $ id ;  
/ **  
  @Column (name = "name" type = "string")  
  */  
  
  private $ name ;  
/ **  
  @Column (name = "content" type = "text")  
  */  
  
  private $ content ;  
  
  the public function set( $ name , $ Value ) {  
    $ this -> $ name = $ Value ; }  
  
  the public function get ( $ name ) {  
    return $ this -> $ name ; }  
  
?>
```

In the Entity class and __get __set methods to detect which fields are updated it is overloaded. Use the set and get methods so

If you wish, you know to get and set the metotlarını apply for each member adopts Faces ORM both in approach.

Asset classes your application / ApplicationName / models / directory to place the application /
ApplicationName / models / as blog.php

Let's talk about how we reach our presence in the object's Controller.

```
<? Php
import ( "Phpf.controllers.facescontrol on" ); import ( "Dbf.persisten by" ); import ( "Models. **", true ); //
application / models directory

class blogcontrol on extends facescontrol the {
    the public function blogcontrol on () {
        parent :: facescontrol on ();
        $ blog = new Blog ();
        $ blog -> name = "Hello World" ;
        $ blog -> content = "Some text" ; EntityManager :: getInstance () -> save ( $ blog );

        // EntityManager :: getInstance () -> delete ( $ blogs );
    }
}

?>
```

Blogs are assigned a new object is created in the preceding line value and lastly insert the block object database is not a new object edilir.eg block objects that we create will be on his own, this meant update

EntityManager :: save (Entity) or veritabindanm of the object method that detects the presence of a new object, and accordingly create or insert or update the database server sends words.

You do not have to write by hand each time the asset class. There is a small java application I developed for this purpose.

Download the file from generator.zip <http://code.google.com/p/php-faces/downloads/list> address. To use the application on your system's java vm is enough to be installed. generator program will constitute the asset class for you.

Relations Between ORM Faces Asset Class

Annotation and relational

@onetoo to

@onetomany

@manytoo to

@manytomany

There are three parametres

mappedby = "Class name"

pk = "primary key"

fk = "foreign key"

Consider the following example

```
CREATE TABLE `blog` (  
  
  `id` int (11) NOT NULL auto_increment,  
  
  `Name` varchar (255) default NULL,  
  
  `Content` text,  
  
  PRIMARY KEY ( `id` )  
)  
CREATE TABLE comment` (  
  
  `Comentid` INT AUTO_INCREMENT NOT NULL PRIMARY KEY,  
  
  `Content` TEXT NOT NULL,  
  
  `Blogid` INT NOT NULL  
  
)
```

many e a comment block in the form of the above two tables to get our structure

There is a relationship between E block comment.

Accordingly, we must define our asset classes as follows.

Blogs class

```
<? Php  
/ **  
*   @Tabl (name = "blog")  
* /  
class Blogs extends Entity {  
/ ** @id  
  
  @Column (name = "id" type = "integer")  
  * /  
  
  private $ id ;  
/ **  
  @Column (name = "name" type = "string")  
  * /  
  
  private $ name ;  
/ **  
  @Column (name = "content" type = "text")  
  * /
```

```

private $ content ;
/ **
    @onetomany (mappedby = "Comment" pKa = "id", fk = "blogid")
    * /

private $ comments ; // ArrayObject

function Blogs () {
    parent :: Entity ();
    $ this -> comments = new ArrayObject (); }

the public function set( $ name , $ Value ) {
    $ this -> $ name = $ Value ; }

the public function get ( $ name ) {

    return $ this -> $ name ; }?>

```

Comment Class <?

Php

```

/ **
    *   @Tabl (name = "comment")
    * /
class comment extends Entity {
/ ** @id

@Column (name = "commentid" type = "integer")
* /

private $ id ;
/ **
    @Column (name = "content" type = "text")
    * /

private $ content ;
/ **
    *   @Column (name = "blogid" type = "INT")
    *   @onetoo to (mappedby = "block" pk = "id", fk = "blogid")
    * /

private $ blogid ;

the public function set( $ name , $ Value ) {
    $ this -> $ name = $ Value ; }

the public function get ( $ name ) {

    return $ this -> $ name ; }}

```

?>

Our class Controller

```

<? Php
import ( "Php.controllers.facescontrol on" ); import ( "Dbf.persisten by" ); import ( "Models.
** , true );

```

```

class Blogtest extends facescontrol the {

    the public function Blogtest () {
        parent :: facescontrol on ();
        $ blog = new Blog ();
        $ blog -> name = "Hello World" ;
        $ blog -> content = "Some text" ;
        $ comment = new Comment ();
        $ comment -> content = "New Comment" ;
        $ blog -> comments-> append ( $ comment ); EntityManager :: getInstance () ->
        save ( $ blog );
    }
}

?>

```

If you want to build your relationship with your asset classes Faces Generator will be in place in advance to notify your SQL query. Faces Generator recognizes relationships and creates classes accordingly.

Faces ORM Entity Manager (Asset Manager)

Entity Manager (Asset Manager) enables you to perform basic operations on the name of your asset classes via suggests. If you want to use in your class asset manager asks a controller in your classes within the entity. methods

insert or update tags are generated by storing a presence status in the database

```
EntityManager :: save (Entity $ e)
```

Sample

```

$ Blog = new Blog ();
$ Blog-> name = "Hello world"; $ Blog->
content = "xxxx";
EntityManager :: getInstance () -> save ($ blogs);

```

EntityManager find the automatic approach is used to search for data in the base of an asset. The first asset class to call the second parameter of this method is the primary key parameters corresponding to the presence of a primary key is.

```
EntityManager :: find ($ Entity E, int id)
```

Sample

```

EntityManager $ em = :: getInstance (); $ Blogs = $
em-> find ( "blog", 2); echo $ blog-> name; echo $
blog-> content;

```

EntityManager to delete is used in a method to delete data from the base of an asset.

EntityManager::delete (Entity \$ e)

Sample

```
EntityManager $ em = :: getInstance (); $ Blogs = $  
em-> find ( "blog", 2); $ Em-> delete ($ blogs);
```

EntityManager createquery automatic approach is used to operate the FQL queries. The ice cream this proce takes a string parameter, and a Query object.

Query objects are extended from the PDOStatement object.

Değiniceg forward in my writings la FQL.

Createquery EntityManager :: Query (String FQL)

Sample

```
EntityManager $ em = :: getInstance ();  
$ Query = $ em-> createquery ( "select blog from b = b where b.id: id"); $ Id = 11;  
  
$ Query-> BindParam ( 'id', $ id); $  
Query-> execute ();  
print_r ($ query-> getresultlist ())
```

SQL queries nativequery EntityManager the automatic approach is used to operate. The ice cream this proce takes a string parameter, and a Query object.

Query objects are extended from the PDOStatement object.

Nativequery EntityManager :: Query (String SQL)

Sample

```
EntityManager $ em = :: getInstance ();  
$ Query = $ em-> createquery ( "select * from block"); $ Query->  
execute ();
```

Query objects to PDOStatement I encourage you to review the genişletig.

PDO supports Transacito lara

```
beginTransaction ();  
commit (); rollback ();
```

Sample

EntityManager use of an asset class

```
<? Php
/ **
 *   @Tabl (name = "comment")
 */
class comment extends Entity {
/ ** @id

@Column (name = "id" type = "integer")
*/

private $ id ;
/ **
    @Column (name = "content" type = "text")
    */

private $ content ;

the public function set( $ name , $ Value ) {
    $ this -> $ name = $ Value ; }

the public function get ( $ name ) {

    return $ this -> $ name ; }

function save () {
    $ confident = Manager :: getInstance () ;
    $ confident -> BeginTransaction () ;

    try {
        $ confident -> save ( $ this ) ;
        $ confident -> commit () ;
    } catcher (Exception $ e ) {
        $ confident -> rollback () ;
    }
}

function delete () {
    $ confident = Manager :: getInstance () ;
    $ confident -> BeginTransaction () ;

    try {
        $ confident -> delete ( $ this ) ;
        $ confident -> commit () ;
    } catcher (Exception $ e ) {
        $ confident -> rollback () ;
    }
}

?>
```

Using the example above in the controller entity classes

```
<? Php
import ( "Php.controllers.facescontrol on" ); import ( "Dbf.persisten by" ); import ( "Models.
** , true );

class Commenttest extends facescontrol the {

the public function Commenttest () {
    parent :: facescontrol on ();
    $ blog = new Comment ();
    $ blog -> content = "Some articles" ;
    $ blog -> save ();
```

```
}}
echo "The record is okay" ;
?>
```

FQL ORM Faces (Faces Query Language)

that allows you to query over objects being FQL is a query language similar to SQL. FQL query word are converted to native SQL PHP Faces Framework, rep.

EXPRESSES the SELECT

It was takmais sififa takmais SELECT FROM WHERE having etc ..

Examples

SELECT b from b blog

SELECT c from c Categories

Blogs from SELECT b b = 1 WHERE bi.id

SELECT WHERE bi.id b from b = 1 and Block B. N = The 'www'

SELECT limits b 0.5 b from blog

C from the SELECT GROUP BY c.id c Categories HAVING AVG (c.id)> 1

FQL use with entity manager

```
$ Query = $ this-> em-> createquery ( "SELECT b from b WHERE blog b.comment.id  
> one");
```

```
$ Query-> execute ();
```

```
$ Blogs = $ query-> getResultlist ();
```

The use SQL functions together with FQL

SELECT COUNT (b.id) FROM Block B

SELECT MAX (b.id) FROM Block B

use with entity manager

```
$ Query = $ this-> em-> createquery ( "SELECT count (b.id) from a b"); $ Query-> execute  
();  
echo "count =". $ query-> getsingl to (). "Member";
```

Internal nested subqueries

```
SELECT b FROM WHERE b.coment.id blog b = (SELECT MAX (c.id) FROM Comment  
c)  
She SELECT FROM WHERE o.id Object o = (SELECT AVG (i.id) FROM Object i) ORDER BY o.n Game
```

Grid Component

Grid component class with the specified sequence itself from the nature of the binder creates an HTML table. Which directive will be notified by @ite I found the column of the created table. u The name of the asset class key nature of the directive @ite I Shot the title of the title attribute is written. painting class with my directive @ite input attribute is used to add external vilg.

```
<@lte I input = "link" title = "Edit" url = "testet.php? $ Edit = no" />
```

For example, it created @ite organized a connection with the above directive and is indicated by the nature of the connection URL address.

Example The following as an Asset to get our Sinf

```
<? Php  
/ **  
* @Tabl (name = "product")  
* /  
class Product extends Entity {  
  
/ **  
* @ID  
* @Column (name = "no")  
* /  
private $ noun ;  
  
/ **  
* @Column (name = "name")  
* /  
private $ name ;  
  
/ **  
* @Column (name = "price")  
* /  
private $ prices ;  
  
function get ( $ name ) {  
return $ this -> $ name ; }  
  
function set( $ name , $ Value ) {
```

```
$ this -> $ name = $ Value ;}}
```

?>

To create a list of our products get a controller as follows: \$ list variable, we transfer the information we pulled into the database.

```
<? Php
import ( "Php.controllers.facet to" ); import ( "Dbf.persisten by" );
import ( "The entities.ur" , true );

class urungrid extends Facet {
    protected $ list ;
    the public function urungrid () {
        parent :: bezel ();
        $ confident = EntityManager :: getInstance ();
        $ query = $ confident -> createquery ( "Select u from ur u" );
        $ query -> execute ();
        $ this -> list = $ q -> getResultList ();
        $ this -> render ( "Grid.html" );
    }
}
```

?>

View our files to bind the nature of the grid label binder = "\$ This.list" Our major additions

```
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4:01 Transitional//EN">
<Html>
    <Head>
        <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

    </ Head>
        <Body>
            <faces>

                <@Import prefix = "f" taglib = "phpf.u in. *" />
                <F: Grid name = "Grid" height = "30%" width = "75%" binder = "$ This.list" border = "one" >
                    <l @ite key = "none" title = "no" />
                    <l @ite key = "name" title = "Product name" />
                    <l @ite key = "price" title = "price" />
                    <@lte l input = "link" title = "Edit" url = "testet.php? $ Edit = no" />
                </ F: grid> </
            <faces> </ body> </
        <html>
```

no	Urun adı	fiyat	Edit
2	deneme14433	5555111	Edit
3	denememe11	1111	Edit
4	222	3434	Edit

SQL Tag

SQL query the nature of the label given to all parameters that operates the SQL query and returns a result set with the name given to the id attribute.

Sample

```
<Html>
  <Head>
    <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

  </ Head>
  <body>
    <faces>
      <@Import prefix = "c" taglib = "phpf.cor" type = "static" /> <@import prefix = "SQL" taglib = "phpf.db.sql"
      />
      <SQL: SQL id = "Results" query = "Select * from product" /> <Table border = "one" align = "Center" >
        <Thead> <th> Product No. </ Th> <th> Name of the product </ Th> <th> Price </ Th> <th> Picture
        </ Th> </ thead> <c: foreach var = "# {} $ This.list" item = "$ Product" > <Tr>

          <Td> # {} $ Urun.no </ Td> <td> # {} $ Urun.ad </ Td> <td> # {} $ Urun.fiyat </
          Td> <td> <img src = "# {But} $ urun.acikl" /> </ Td> </ tr> </ c: foreach> </ table> </ faces>
        </ body> </ html>
```

FQL Label

FQL FQL query tag given to all parameters that are operating in the nature of the query and returns a result set with the name given to the id attribute.

Sample

```
<Html>
  <Head>
    <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

  </ Head>
  <body>
    <faces>
      <@Import prefix = "f" taglib = "phpf.u in. *" /> <@Import prefix = "FQL" taglib =
      "phpf.db.fql" />
      <FQL: FQL id = "Results" query = Select "u from ur u" /> <F: Grid name = "Grid" height = "30%" width = "75%" binder = "$ Results" border = "one"
      >
        <l @ite          key = "none" title = "no" />
        <l @ite          key = "name" title = "Product name" />
        <l @ite          key = "price" title = "price" />
      </ F: grid> </
    <faces>
```

```
</ Body> </
html>
```

SQL and FQL @ite label to label the nature of the transferred parameter value. Sample price range

```
<Html>
  <Head>
    <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

  </ Head>
  <body>
    <faces>
      <@Import prefix = "f" taglib = "phpf.u in. *" /> <@Import prefix = "FQL" taglib =
        "phpf.db.fql" />
      <FQL: FQL id = "Results" query = Select "u from ur u WHERE u.fiyat=? AND u.fiyat <?" >
      <@Ite value = "100" /> <@ite value =
        "500" />
      </ Sql: FQL>




      <F: Grid name = "Grid" height = "30%" width = "75%" binder = "$ Results" border = "one" >
        <l @ite      key = "none" title = "no" />
        <l @ite      key = "name" title = "Product name" />
        <l @ite      key = "price" title = "price" />
      </ F: grid> <f: classpag id = "Pager" request = "Page" limit = "3" count = "10" /> </ Faces> </ body> </ html>
```

Page Label

Page is used to make labels for paging.



Two different uses the entity name of the entity class attributes in the first use. Limit the number of records to be displayed as if nature is written on a label FQL set of results to be obtained isteniyor resource to be recovered to an extent that a set of attribute name should be written.

Urun no	Urun adı	fiyat	Resim	Detay
2	CPU	100		Detay
3	Ana kart	80		Detay
4	RAM	50		Detay



If you are using PHP framework to faces in the mode of the pager tags for URI uri way paging create the page object, you must provide the path to the type attribute value.

Sample Page and Grid Usage

```
<Html>
  <Head>
    <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >
  </ Head>
  <body>
    <faces>
      <@Import prefix = "f" taglib = "phpf.u in. *" /> <@Import prefix = "db" taglib =
      "phpf.db. *" />
      <F: Grid id = "Grid" width = "one hundred%" binder = "$ List" border = "one" >
        <l @ite      key = "none" title = "Product no" />
        <l @ite      key = "name" title = "Product name" />
        <l @ite      key = "price" title = "price" />
        <@lte l input = "img" key = "image" title = "Image" /> <@ite l input = "link" title = "Detail" url = "testet.php? $ Edit = no" />

      </ F: grid> <db: pager id = "Pager" limit = "3" entity = "product" resource = "list"

previous = "Previous" next = "Next" />
    </ Faces> </
  body> </ html>
```

tag query attribute of the Page FQL FQL accept the next sentence from the electoral process Sample Price range

```
<Db: pager id = "Pager" type = "Path" limit = "3" entity = "product" resource = "list"
      query = "WHERE urun.fiyat> = 50 AND urun.fiyat <= 100" />
```

Page Kontrollc the components within the database query of whether to use pager bileleş field while the other counter and limit the number of records collected NATURE niteliğine should be given the number of records to be displayed on the page.

View

Sample

```
<Html>
  <Head>
    <Title> </ title> <meta http-equiv = "Content-Type" content = "Text / html; charset = UTF-8" >

  </ Head>
  <body>
    <faces>
      <@Import prefix = "c" taglib = "phpf.cor" type = "static" /> <@import prefix = "db" taglib = "phpf.db. *" />

      <Table border = "one" align = "Center" > <Thead> <th> Product
        No. </ Th> <th> Name of the product </ Th> <th> Price </
        Th> <th> Picture </ Th> </ thead>
```



```

<C: foreach var = "{0} $ This.list" item = "$ Product" > <Tr>

    <Td> {0} $ Urun.no </ Td> <td> {0} $ Urun.ad </ Td> <td> {0} $ Urun.fiyat </
Td> <td> <img src = "{0} {But} $ urun.acikl" /> </ Td> </ tr> </ c: foreach> </ table> <db:
pager id = "Pager" limit = "3" count = "10" /> </ Faces> </ body> </ html>

```

Controller

```

import ( "Phpfc.controllers.facet to" ); import ( "Dbf.persisten by" );
import ( "Entityes. ** , true" ); import ( "Phpfc.db.pag is" );

```

```

class Products extends Facet {
    protected $ list ;
    the public function Products() {
        parent :: bezel ();
        $ this -> render ( "Manifestation / test.html" );
    }
    the public function PreRender () {
        $ confident = EntityManager :: getInstance ();
        $ query = $ confident -> createquery ( "Select count (u.no) from ur u" );
        $ this -> pager-> setcount ( $ query -> execute () -> getsingl A ());
        $ query = $ confident -> createquery ( "Select u u from ur GROUP BY u.no" . $ this -> pager-
> getlimitquery ());
        $ this -> list = $ query -> execute () -> getResultlist ();
    }
}

?>

```

Page queryString component of the Order if requested knows that p is replaced by the request to be processed while NATURE Enthousiast in the view definition for that variable file name is written.

```

<Db: pager id = "Pager" limit = "3" count = "10" request = "page" />

```