بسم الله الرحمن الرحیم

آموزش فریم ورک گتنا نسخه 1.2.GFB

فریم ورک گتنا

گروه تخصصی نت افزار

یافته‌هایت را با باخته‌هایت مقایسه کن اگر "خدا" را یافتی هر چه باختی مهم نیست.
گوش‌های "خدا" پر است از آرزو و دست‌هایش پر از معجزه؛ شاید بزرگترین آرزوی تو کوچکترین معجزه "خدا" باشد.

چه تغییراتی در 1.2.GFB حاصل شده است؟

ویرایش دوم ۱۳۹۱/۰۸/۱۹

http://www.gtna.net

پاییز ۱۳۹۱

GTNA_Session->id()

GTNA_Session->new_id()

GTNA_Session->RBI()

GTNA_Session->SRARO()

GTNA_Session::sess_read()

GTNA_Session->sess_close()

چگونه ازکتابخانه GTNA_Security استفاده کنیم؟

GTNA_Security->sanitize_string()

GTNA_Security->escape()

GTNA_Security->strip_tag()

GTNA_Security->encrypt()

Mysql_sanitize_string()

چگونه با متدهای بانک اطلاعاتی عملیات CRUD انجام دهیم؟

Mysql_open()

Mysql_close()

Mysql_insert()

Mysql_update()

Mysql_select()

Mysql_delete()

چگونه فرم‌های درون یک view را به یک متد خاص از کنترلر ارسال کنیم؟

روش استفاده از کتابخانه‌ی GTNA_HTML؟

کاربرد فریم ورک گتنا

این فریم ورک برای افرادی طراحی شده است که قصد دارند خیلی سریع و آسان پروژه‌های خود را در حد متوسط پیش ببرند. البته گتنا (گروه تخصصی نت افزار) قصد دارد در آینده‌ی نه چندان دور، قابلیت‌های گسترده‌ای را به این فریم ورک اضافه کند. اکنون در حال بررسی و تحقیق بر روی هوش مصنوعی و ترکیب آن با این فریم ورک می‌باشیم. اگر بخواهیم به توانایی‌های آن اشاره کنیم، یکی ازساده‌ترین فریم ورک‌ها از لحاظ یادگیری است. شما تنها در یک روز می‌توانید با کلیه‌ی متدهای آن آشنا شوید و پروژه‌های خود را به وسیله‌ی آن طراحی کنید.

چرا باید از فریم ورک استفاده کنیم؟

دلایل زیادی وجود دارد که خیلی از طراحان سایت‌ها و اپلیکیشن‌های تحت وب را به سمت فریم ورک‌ها سوق داده است. ما به چند مورد اشاره می‌کنیم:

- سایت‌ها و اپلیکیشن های تحت وب در یک چهارچوب مشخص و با نظم خاصی طراحی می‌شود.
- می‌توانید از کلاس‌هایی که از قبل طراحی شده، استفاده کنید.
- طبق استانداردهایی که برای فریم ورک طراحی شده است، به فایل‌ها و پوشه‌ها نظم می‌دهد.
- به دلیل طراحی سایت بر پایه معماری سه لایه، امنیت پروژه‌هایتان را افزایش می‌دهد.
- تولید و توسعه پروژه‌هایتان را تا حد خیلی زیاد آسان می‌کند.
- URLها را تا حد ممکن کوتاه می‌کند.
- به راحتی می‌توانید از طریق URL به کنترلرها و متدهای آن دسترسی داشته باشید و همچنین متدها را با مقدار فراخوانی کنید.

البته این موارد محدودیتی در کدنویسی ایجاد نمی‌کند. شما به راحتی می‌توانید کلاس‌های خود را به عنوان یک کتابخانه به فریم ورک اضافه و از کلاس‌های موجود این فریم ورک استفاده کنید و متدهای آنها را فراخوانی نمایید.

آیا این فریم ورک پشتیبانی مداوم دارد؟

جواب به این سؤال بستگی به کاربران دارد. اگر گتنا به وسیله کاربرانش مورد حمایت قرار بگیرد صد در صد این پروژه تا زمانی که گتنا وجود دارد بروزرسانی می‌شود و در خدمت طراحان ابزارهای تحت اینترنت قرار

خواهد گرفت.گتنا تمام سعی و تلاش خود را برای بهبود عملکرد این فریم ورک می‌کند تا نیازهای طراحان را در حد وسیع مرتفع سازد.

نحوه پیکربندی فایل GTNA_Config.php:

شما به وسیله‌ی آدرس زیر به این فایل دسترسی دارید و می‌توانید تنظیمات اولیه را برای طراحی پروژه‌هایتان را انجام دهید:

application/config/GTNA-Config.php

system_controller="welcome":این متغیر برای مشخص نمودن کنترلر پیش‌فرض می‌باشد. یعنی اگر قصد داشته باشید هنگام بارگذاری سایت در مرحله اول، کنترلر دلخواهتان باز شود کافی است که به جای **welcome** نام کنترلر خود را وارد نمایید. در این فریم ورک به صورت پیش‌فرض،کنترلر **welcome** بارگذاری می شود واگر نام کنترلری وارد نمایید که وجود نداشته باشد پیام زیر برای شما نمایش داده می‌شود.

Controller dose not exist.

مرحله بعدی برای تنظیم فریم ورک تنظیمات مربوط به پورت، آدرس سایت و شاخه‌ای است که شما پروژه خود را در آن قرارداده‌اید و می‌خواهید بر روی آن **URL** کار کنید. به عنوان مثال فرض کنید شما قصد دارید به صورت زیر بر روی پروژه خود کار کنید:

http://127.0.0.1:81/GTNAF

فریم ورک گتنا برای چنین تنظیماتی، چندین متغیر در نظر گرفته است. به متغیرهای زیر توجه کنید.

```
//System root settings
$System_url="http://127.0.0.1;
$System_directory="GTNAF";
$System_port=81;
```

فریم ورگ گتنا از تنظیمات بالا به این نتیجه می‌رسد که سیستم شما به صورت **local** می‌باشد و فایل‌های اصلی شما در شاخه‌ای به نام **GTNAF** قرار دارد و همچنین تمام آدرس‌ها به جز آدرس اصلی سایت یعنی **system_url** بر روی پورت ۸۱ فعال است.تنظیمات پورت به این دلیل است که شاید شما بخواهید بر روی پورتی به غیر از پورت ۸۰ کار کنید. بنابراین درک فریم به صورت یک آدرس یا **URL** به شکل زیر است.

http://127.0.0.1:81/GTNAF

پس از گذر از این مرحله شما باید برای پروژه‌ی خود یک شاخه‌بندی در نظر بگیرید. مثلاً فایل‌های جاوا اسکریپتی و یا فایل‌های مربوط به استایل سایت در شاخه‌های مخصوص خود قرار بگیرند. فریم ورک گتنا برای کنترل و همچنین توسعه راحت‌تر پروژه‌ها این تنظیمات را در نظر گرفته است. همچنین شما با این کار تنها با یک متغیر به فایل‌های خود در **view** دسترسی دارید. (کوتاه سازی آدرس‌های) متغیرهای زیر را در نظر بگیرید:

```
$css_url=SYSTEM_URL.$system_file_directory."styles";
$java_url=SYSTEM_URL.$system_file_directory."java";
$jquery_url=SYSTEM_URL.$system_file_directory."jquery";
$image_url=SYSTEM_URL.$system_file_directory."images";
$other_url=SYSTEM_URL.$system_file_directory."other";
```

styles-java-images-other نام‌شاخه‌هایی است‌که در مسیر **application** قراردارد. یعنی درک سیستم برای فایل‌های استایل و ... به شکل **URL** به صورت زیر است:

برای فایل های استایل:

http://127.0.0.1:81/GTNAF/application/styles

برای فایلهای جاوا:

http://127.0.0.1:81/GTNAF/application/java

برای فایل های تصویری :

http://127.0.0.1:81/GTNAF/application/images

برای فایلهای نامشخص :

http://127.0.0.1:81/GTNAF/application/other

اگر به عنوان مثال، یکی از کدهای بالا را به شکل زیر تغییر دهید

```
$css_url=SYSTEM_URL.$system_file_directory."my-styles";
```

باید یک دایرکتوری در مسیر **application** با نام **my-styles** ایجادکنید.بعد از انجام این تنظیمات از طریق ثابت‌های زیر، در **view**ها به این مسیرها دسترسی دارید. البته نام فایل‌ها را خودتان باید به URL اضافه کنید.

```
CSS_URL
JAVA_URL
JQUERY_URL
OTHER_URL
```

به عنوان مثال از این پس زمانی که بخواهید یک فایل استایل موجود در دایرکتوری style و یا my-styles در یک view بارگذاری کنید کافی است تگ لینک را به شکل زیر بنویسید:

```
<link href="<?php echo CSS_URL/style.css ?>" />
```

برای بارگذاری فایل‌های دیگر هم به همین شکل می توانید عمل کنید.

حال ما فرض را بر این می‌گذاریم که شما می‌خواهید فایل‌های خود را در شاخه‌ای به غیر از شاخه‌های موجود قرار دهید. کافی است به شکل زیر عمل کنید:

```
$upload_url =SYSTEM_URL.$system_file_directory."upload";
```

به جای **upload** هر نام دیگری می‌توانید قرار دهید و برای اینکه این مسیر در کل viewها شناخته شود کد زیر را به فایل **GTNA_Config.php** دقیقاً زیر ثابت ها اضافه کنید.

```
define(UPLOAD_URL'," $upload_url" );
```

از این پس شما به وسیله متغیر **UPLOAD_URL** به فایل‌های موجود در این دایرکتوری دسترسی دارید. کافی است مسیرهای زیر را به شکل زیر در یک view به خروجی ببرید تا متوجه منظورمان شوید.

```
echo CSS_URL;
echo JAVA_URL;
echo JQUERY_URL;
echo OTHER_URL;
```

و با تعریف دایرکتوری **upload** که خودمان به فایل **GTNA_Config.php** اضافه کردیم، به صورت زیر به آدرس آن دسترسی داریم:

```
echo UPLOAD_URL
```

$_CS['STATE'] : این متغییر دو مقدار **ON** و **OFF** می پذیرد همچنین نسبت به حروف کوچک و بزرگ حساس است سیستم به صورت پیش فرض این متغییر را برابر با **OFF** قرار داده است با **ON** شدن این متغییر هیچ کس نمی تواند محتوای سایت شما را ذخیره کند بنابراین در حد چشم گیری امنیت را افزایش

می دهد با تنظیم این امکان حتی توسط زبان‌های vb.net,C#.net,php,asp.net و...نمی توان محتوای سایت را ذخیره کرد.

$_GP['STATE']:این متغیر دو مقدار **ON**و**OFF**را می پذیرد و نسبت به حروف بزرگ و کوچک حساس است. زمانی که مقدارش **OFF** باشد تمام فرم های ارسالی خارجی را می پذیرد اگر قصد دارید فرم های خارجی را **block** کنید می توانید مقدارش را برابر با **ON** قرار دهید می دانید که **block** کردن همه فرم های خارجی امکان دارد مشکلاتی را به همراه داشته باشد به عنوان مثال اگر شما بخواهید از درگاه های بانک استفاده کنید بانک به اجبار مقادیری ارسال می کند که شما باید آنها را بپذیرید بنابراین برای رفع این مشکل متغیر دیگری با نام **$_LIST['ACCEPT']** وجود دارد در این متغیر لیست تمام دامین هایی که قرار است فرم های ارسالی از طرف آنها پذیرش شود را به صورت یک آرایه وارد می کنید توجه داشته باشید که حتما آدرس دامین خود را هم به لیست اضافه کنید در غیر این صورت حتی ارسال های داخلی هم **Reject** می شود.

توجه : دامین ها را بدون پرتکل**http** وارد نماید .

```
/* if $_GP['STATE']=ON then
*       $_LIST['ACCEPT']=array('www.gtna.net',
*                              'www.other.com',
*                                 ...)
*/
$_LIST['ACCEPT']=array('127.0.0.1','www.gtna.net','www.other.com');
$_GP['STATE']=OFF; // GET & POST Accept state
$_GP['STATE']=ON;
```

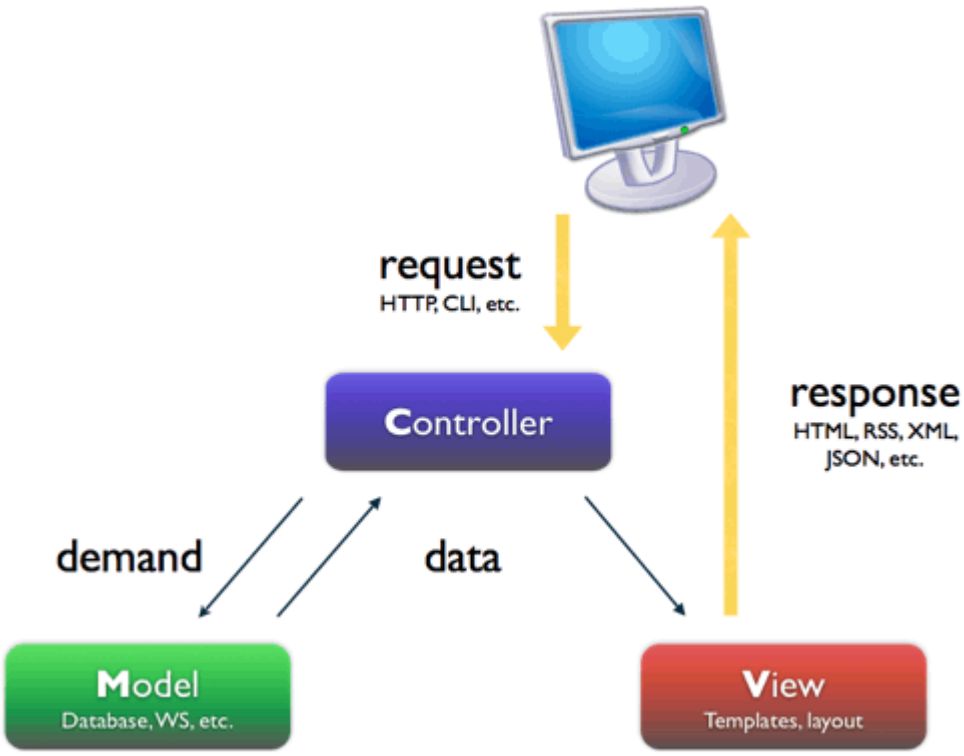
تذکر : اگر مقدار متغیر **$_GP['STATE']** برابر با **ON** باشد ولی متغیر**$_LIST['ACCEPT']**خالی باشد با خطای زیر رو برو می شوید .

```
Please set your host url in $_LIST[ACCEPT]
For EX:
====> $_LIST[ACCEPT]=array('www.gtna.net')
```

گتنا قصد دارد در آینده نزدیک با طراحی یک الگوریتم رمز نگاری تمام داده های رد و بدل شده را رمز کند.

روش ایجاد Controller

اگر به شکل زیر توجه کنید، متوجه می‌شوید که معماری سه لایه یا three layer به شکل زیر عمل می‌کند.



مشخص است که درخواست از سمت کاربر به کنترلر ارسال می‌شود. کنترلر وظیفه دارد امنیت را برقرار کند. ما در این لایه می‌توانیم به لحاظ امنیتی داده‌ها را اعتبار سنجی کنیم، بنابراین کنترلر وظیفه پردازش داده‌ها و همچنین امنیت را بر عهده دارد. در صورتی که در خواست متقاضی توسط کنترلر مورد بررسی قرار گرفت، Viewی مورد تقاضای کاربر توسط کنترلر نمایش داده می‌شود. اگرکاربر بخواهد داده‌هایی را از جداول درون پایگاه داده مورد جستجو قرار دهد باید این بار تقاضای خود را از طریقView به کنترلر ارسال کند. کنترلر بعد از انجام اعتبارسنجی و برقراری امنیت بر روی داده‌ها و یا اسکیپ‌سازی آن‌ها (معتبرسازی)، تقاضای کاربر را به لایه Model ارسال می‌کند. این لایه وظیفه دارد کلیه امور مربوط به بانک اطلاعاتی را انجام دهد. بنابراین لایه مدل کوئری‌های مربوطه را اجرا می‌کند و خروجی را به کنترلر ارسال می‌کند. توجه داشته باشید که این لایه تنها با لایه کنترلر مرتبط است سپس لایه کنترلر داده‌ها رابهView ارسال می‌کند. بستگی به نوع سلیقه خودمان داده‌های مورد تقاضای کاربر را در هر جای صفحه وب که بخواهیم نمایش می‌دهیم. پس لایه کنترلر به عنوان یک واسطه عمل می‌کند و هیچ کاربری نمی‌تواند مستقیماً به یک View و

یا Model و حتی خود کنترلر دسترسی داشته باشد. یک فایل در سیستم وجود دارد که کتابخانه‌های مورد نیاز کنترلر را بار می‌کند و در صورت وجود، آنها را در کنترلرمان قابل دسترس می‌کند. از آن به بعد، کلیه داده‌ها توسط متدهای کتابخانه‌ها مورد بررسی قرار می‌گیرد و از لایه کنترلر به لایه‌های دیگر هدایت می‌شود. هر تقاضایی که از این دروازه عبور نکند به عنوان یک تقاضای نامعتبر در نظر گرفته می‌شود. بنابراین با کلیه این تفاسیر اگر شما بخواهید پروژه‌های خود را بر پایه معماری سه لایه اصلی یعنی MVC طراحی کنید، کار بسیار دشواری دارید گتنا این سکو را برای شما به ارمغان آورده است و شما به راحتی می‌توانید طبق معماری سه لایه پروژه‌های خود را مدیریت کنید. البته گتنا قصد دارد در آینده لایه هوش مصنوعی را به این لایه‌ها اضافه کند و این مورد باعث تمایز این فریم ورک نسبت به فریم ورک‌های دیگر می‌شود. البته به این مفهوم که این فریم ورک از فریم ورک‌های موجود بهتر است نمی‌باشد، ما تنها سعی کرده‌ایم تا تمام نیازهای اساسی مورد نیاز طراحان را به همراه امنیت به این فریم ورک اضافه کنیم و سادگی فراگیری آن را در نظر گرفته‌ایم. با حمایت شما انشالله درآینده قدرت فریم ورک را به رخ خواهیم کشید. همچنین قصد داریم سرعت پردازش اسکریپت‌ها را تا حد ممکن با ترفندهای خاص خودمان افزایش دهیم.

حالا به سراغ بحث اصلی خودمان می رویم برای اینکه یک کنترلر طراحی کنیم باید در گام اول کنترلرهای خود را در مسیر زیر ایجاد کنیم.

application/controllers/welcome.php

برای بر قراری امنیت کنترلرها، کد زیر را به بالای کنترلر اضافه می‌کنیم.

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
```

حالا با ایجاد یک کلاس دوست، اولین کنترلر را می‌نویسیم:

```
class welcome extends GTNA_controller{
function __construct(){parent::__construct();}
function index(){
                $this->load->view('welcome');
        }// end index method
}// end welcome controller
```

نکته: **زمانی که در مسیر application/controllers یک فایل با نام welcome.php و یا هر نام دیگری که ایجاد می‌کنید باید کلاسی که از GTNA_controller ارث‌بری می‌کند دقیقاً با همان نام باشد.**

همانطور که مشاهده می‌کنید،ما یک کنترلر به نام **welcome** طراحی کرده ایم که از **GTNA_controller** ارث‌بری کرده است و شامل یک متد به نام **index** می‌باشد. توجه کنید که متد index یک متد الزامی است و در تمامی کنترلرها باید وجود داشته باشد.

توجه: به هر تعداد که تمایل دارید می‌توانید کنترلر ایجاد کنید، اما نمی‌توانید آنها را در دایرکتوری‌های متفاوت ایجاد کنید. در آینده این امکان به فریم ورک افزوده می‌شود.

درون متد **index** یک**view** بارگذاری شده است.این به این معنی است که در مسیر **application/views** فایلی به نام **welcome.php** قرار دارد. بنابراین شما باید از قبل این فایل را ایجاد کرده باشید در غیر این صورت با خطای زیر برخورد می‌کنید.

|application/views/welcome.php not found|

روش ایجاد View

کافیست یک فایل دیگر با نام welcome.php را در مسیر application/views ایجاد کنید و کدهای زیر را در آن قرار دهید. به این ترتیب view طراحی می‌شود.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="<?php echo CSS_URL."/style.css"; ?>" type="text/css" rel="stylesheet" />
<title>Untitled Document</title>
</head>
<style>
/* css code ...*/
</style>
<body>
```

```
<?php
		echo SYSTEM_URL."<br>";
		echo CSS_URL."<br>";
		echo JAVA_URL."<br>";
		echo JQUERY_URL."<br>";
		echo OTHER_URL."<br>";
		echo IMAGE_URL."<br>";
		echo "welcome to GTNA framework."
?>
</body>
</html>
```

برای نمایش خروجی کافی است در مرورگر خود آی پی http://127.0.0.1/GTNAF را وارد نمایید، به همین ترتیب شما می‌توانید چند **view** را در یک کنترلر نمایش دهید.

اگر قصد دارید درون یک **view** چندین **view** ی دیگر بار کنید، از کد زیر استفاده نمایید.

```
load::view_is("view name");
```

روش ایجاد Model

قبلاً وظیفه‌ی لایه مدل را گفتیم. در اینجا قصد نداریم یک مدل ایجاد کنیم. جهت اتصال به دیتابیس و استخراج داده‌ها از درون جداول فقط به نحوه ایجاد یک مدل اشاره می‌کنیم و در آینده به تشریح متدهایی که در لایه مدل و همچنین تمامی متدهایی که در لایه کنترلر قرار دارد می‌پردازیم.

شما باید تمامی مدل‌های خود را در دایرکتوری زیر ایجاد کنید:

application/models

اگر ما بخواهیم یک مدل با نام **gtna_model** ایجاد کنیم، باید ابتدا این فایل را به شکل زیر ایجاد کنیم.

application/models/gtna_model.php

```
<?php if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class gtna_model extends GTNA_model{
	function create(param){ your code ...} // method 1
	function read(param){your code ..} // method 2
	function update(param){your code...} // method 3
	function delete(param){your code...} // method 4
}// end gtna_model
```

اکنون شما مدلی به نام **gtna_model** که از **GTNA_model** ارثبری نموده است، ایجاد کرده‌اید که شامل ۴ متد **Create-Read-Update-Delete** یا (**CRUD**) می‌باشد اما کدنویسی‌های مربوط به آنها را در بخش کلاس‌های لایه مدل معرفی می‌کنیم و برای هر یک مثالی می‌آوریم تا با نحوه عملکرد و شیوه استفاده از آنها آشنا شوید.

چگونه از URLها استفاده کنیم ؟

URLها یکی از مهمترین قسمت‌هایی هستند که در فریم ورک‌ها کاربرد دارند. از طریق آنها می‌توان به کنترلرها و متدهای آنها دسترسی داشت و حتی می‌توانید متدهای کنترلرها را با مقدار فراخوانی کنید. برای مثال، فرض کنید که شما یک کنترلر به شکل زیر ایجاد کرده‌اید.

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class show_message extends GTNA_controller{
        function display($message=NULL)
        {
                echo "This is 'GTNA framwork'";
        }
}
```

حال اگر بخواهیم از طریق **URL** به کنترلر و متدهای آن دسترسی داشته باشیم،شکل کلی آن به صورت زیر است.

http://127.0.0.1/GTNAF/index.php/controller/method/value1/value2/.../value5

بنابراین شما می‌توانید تا ۵ مقدار را به پارامترهای یک متد از کنترلر ارسال کنید اما در اینجا فقط یک پارامتر با نام **$message** داریم. انشالله در آینده قصد داریم کاری کنیم که شما بتوانید یک آرایه را از طریق **URL** دریافت کنید و همچنین چندین متد را از طریق**URL** همزمان فراخوانی کنید.حالا اگر بخواهیم به متد **display** ازکنترلر **show_message** دسترسی داشته باشیم، باید به شکل زیر عمل کنیم (یعنی کافیست که آدرس زیر را در مرورگر وارد نمایید).

http://127.0.0.1/GTNAF/show_message/display

اگر متد **display** دارای پارامتر باشد و ما بخواهیم آن را مقدار دهی کنیم به شکل زیر عمل می‌کنیم.

http://127.0.0.1/GTNAF/show_message/display/value1/.../value5

به جای value باید مقادیر مورد نیاز خود را ارسال کنید. در این صورت شما خروجی را بر روی مرورگر مشاهده خواهید کرد .

تذکر: در حال حاظر هیچ متدی نمی‌تواند بیشتر از ۵ پارامتر داشته باشد.

نکته: اگر کنترلری که از طریق URL فراخوانی می‌کنید وجود نداشته باشد، با پیام خطای زیر برخورد خواهید کرد.

Controller dose not exist

نکته : اگر همزمان با کنترلر متد آن را فراخوانی کنید و هیچکدام از آنها موجود نباشد با پیام زیر برخورد خواهید کرد (طبیعی است که وقتی کنترلری وجود ندارد، متدی از آن نیز وجود ندارد).

Controller and method dose not exist

توجه: حتی اگر آدرس صفحه اصلی یا home page خود را به شکل زیر فراخوانی کنید، با پیام خطا روبرو خواهید شد.

http://127.0.0.1/GTNAF/index.php/

در URL بالا یک اسلش اضافی وجود دارد. یعنی شکل درست آن به صورت زیر می‌باشد.

http://127.0.0.1/GTNAF/index.php

نکته بسیار مهم: شما باید در لایه کنترلر تمام مقادیری که از طریق URL به متدهای ارسال می‌شوند را به لحاظ امنیتی بررسی کنید. برای این کار می‌توانید با مراجعه با قسمت کتابخانه امنیتی GTNA_Security این کار را انجام دهید. در غیر این صورت هکرها با وارد کردن مقادیر نامعتبر از طریق URL اسکریپت‌های خود را به سایتتان تزریق می‌کنند. بنابراین تا زمانی که نتوانید با متدهای امنیتی فریم ورک کار کنید قادر به برقراری امنیت نمی‌باشید.

چگونه از Libraryها استفاده کنیم؟

ما برای بارگذاری کتابخانه‌های موجود در این فریم ورک دو روش را در نظر گرفته‌ایم که به صورت زیر برای شما دسته بندی می‌کنیم.

۱- زمانی که قصد دارید از کتابخانه‌های اصلی موجود در هسته فریم ورک استفاده کنید، از کد زیر استفاده نمایید:

برای مثال :

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class Security extends GTNA_Controller
{
	function index()
	{
		$this->library('GTNA_Security');
		echo "<pre>";
		print_r( $this->GTNA_security->sanitize_string("help"));
	}
}
```

اگر به کنترلر **Security** توجه کنید، در متد **index** که به عنوان یک متد پیش فرض در نظر گرفته شده است، به صورت زیر کتابخانه امنیتی را فراخوانی کرده‌ایم.

```
$this->library('GTNA_Security');
```

و زمانی که بخواهیم از متد های **Security** و یا هر کتابخانه‌ی دیگری استفاده کنیم، بعد از دستور بالا به صورت زیر عمل می‌کنیم.

```
$this->GTNA_security->sanitize_string("help");
```

و چون خروجی آن از نوع آرایه می‌باشد، آن را با دستور **print_r** به خروجی برده‌ایم. قصد ما از مثال بالا تنها بارگذاری کتابخانه‌های موجود در هسته و نحوه استفاده از متدهای بود.

۲- زمانی که می‌خواهیم کتابخانه‌ای مورد نظر خودمان را بارگذاری کنیم و از متدهای آن استفاده نماییم.

اگر شما برای پروژه خود یک کتابخانه طراحی کرده‌اید و قصد دارید از متدهای آن در پروژه استفاده کنید، باید به صورت زیر عمل نمایید (فرض کنید کتابخانه‌ای به نام **Encode** ایجاد کرده‌اید که دارای یک متد به نام **string_encode** می باشد).

- در مرحله اول کتابخانه خود را در مسیر زیر قرار دهید.

Application/library/your library.php
Application/library/Encode.php

شکل کلی کدهای مربوط به کتابخانه شما باید به صورت زیر باشد.

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class Encode {
        function string_encode($string=NULL){
        if(!$string)
                echo "please enter a string";
        else
                return md5($string);
        }
}
```

- در مسیر زیر یک کنترلر با نام دلخواهتان ایجاد کنید و سپس به صورت زیر کد نویسی‌ها را انجام دهید.

Application/controllers/code.php

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class code extends GTNA_controller{
        function index()
        {
                $this->load->library(array('Encode'));
                $this->library('Encode');
                echo $this->Encode->string_encode ("GTNAF");
        }
}
```

اگر قصد دارید همزمان از چندین کتابخانه استفاده نمایید به صورت زیر اقدام نمایید

```
$this->load->library(array('Encode','Form',...));
```

شیوه استفاده از کتابخانه GTNA_URL:

اگر قصد دارید با استفاده از کتابخانه **GTNA_URL** یک دایرکتوری جدید برای فایل‌های خود ایجاد کنید ولی خارج از مسیر فریم ورک باشد. ابتدا باید در مسیر زیر یک دایرکتوری با نام دلخواهتان ایجاد کنید.

http://127.0.0.1/GTNAF/Your directory name

به عنوان مثال:

http://127.0.0.1/GTNAF/My_file

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class code extends GTNA_controller{
        function index(){
        $this->library('GTNA_URL');
        define('PATH_FILE', $this->GTNA_URL->system_url("My_file "));
        }
}
```

حالا در viewها می‌توانید از ثابت **PATH_FILE** استفاده نمایید. کافی است برای فهم بیشتر این موضوع بعد از ایجاد کنترلر بالا، کد زیر را در یک view تایپ کنید و خروجی آن را ببینید.

```
<?php echo PATH_FILE ; ?>
```

توجه: اگر دایرکتوری My_file توسط سیستم پیدا نشد با خطای زیر روبرو می‌شوید.

The requested URL /GTNAF/My_file/x.x was not found on this server

x . x نام و پسوند فایلی است که در مسیر **My_file** باید قرار داشته باشد، اما اکنون وجود ندارد.

چگونه باکتابخانه GTNA_Captcha سوال امنیتی ایجاد کنیم؟

قبل از اینکه به نحوه استفاده از کد کپچا بپردازیم کمی در مورد این کد و کاربرد آن توضیح می‌دهیم. در واقع این کد برای تشخیص انسان و ربات می‌باشد. ممکن است فرمهایی زیادی دیده باشید که در پایین آنها کدهایی وجود دارد که شما بعد از تکمیل نمودن آن فرمها باید کد امنیتی را وارد نماییددر واقع با وارد کردن این کد، سیستم تشخیص می دهد که شما یک انسان می‌باشید یا یک ربات. در اینجا قرار نیست به نحوه سوء استفاده از فرمهایی که این کد را ندارد اشاره‌ای داشته باشیم. فقط این نکته را بدانید که کدهای امنیتی یا کد کپچا، امنیت را برای سایت هایتان به ارمغان می‌آورند. گتنا در حال حاضر تنها یک نمونه ساده، اما با امنیت بالادر یک کتابخانه به نام **GTNA_Captcha** برای شما فراهم نموده است.

در فریم ورک گتنا به دو روش می توانید کد کپچا ایجاد کنید

۱- کد کپچا به وسیله **Session** یا نشست
۲- کد کپچا به وسیله جدول **Captcha**
۳- کد کپچا ایجاد شده در نشست‌ها سرعت بسیار بالایی دارد اما امکان دارد هکر به وسیله ربودن نشست موفق به دسترسی به کد کپچا شود. برای جلوگیری از این امر درکنترلر نشست‌ها را رمزنگاری می‌کنیم ولی باز هم به طور کلی نسبت به روش دوم امنیت پایین‌تری دارد.

طبیعتاً روش دوم امنیت بیشتری دارد، چرا که جواب سؤالات درون جدول کپچا ذخیره می‌شود و سپس با جوابی که کاربر ارسال نموده است مقایسه می‌شود. اگر پاسخ کاربر با جواب برابر باشد باید کد ذخیره شده درون جدول کپچا حذف شود این روش نسبت به روش نشست سرعت پایین‌تری دارد اما امنیت را تا حد بسیار بالایی افزایش می‌دهد.

- **پیاده‌سازی کد کپچا با استفاده از Session**

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class Login extends GTNA_controller{
function captcha()
{
        $this->library('GTNA_Session');
        $this ->GTNA_Session->open();
        $this ->GTNA_Session->encode();
        $this ->library('GTNA_Captcha');
        $this ->GTNA_Captcha->simple_captcha();
        $this ->GTNA_Session->register('captcha',array('CR'=>CAPTCHA_RESULT));
        $this ->load->view('login');
}
```

```
function form(){
		$this->library('GTNA_Session');
		$this->library('GTNA_URL');
		$this->GTNA_Session->open();
		if($this->GTNA_Session->RBI('captcha','CR')){
			if ($_POST['captcha_code']==$this->GTNA_Session->RBI('captcha','CR'))
				$this->load->view('admin');
			 else
				$this->GTNA_URL->redirect("captcha");
		}else
				$this->GTNA_URL->redirect("captcha");
}// end form method
}// end of class
```

admin.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>welcome to administrator</title>
</head>
<body>
<center>welcome to admin</center>
</body>
</html>
```

login.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>login </title>
</head>
<body>
```

```
<form action="<?php echo SYSTEM_URL."index.php/Login/form" ?>" method="post">
<p><input type="text" name="user_name"></p>
<p><input type="password" name="user_pass" /></p>
<p><input type="text" name="captcha_code" /></p>
<p><?php echo CAPTCHA_CODE ?></p>
<p><input type="submit" value="login" /></p>
</form>
```

با اجرای پروژه، در صورتی که کد کپچا را درست وارد نمایید با پیام زیر روبرو می‌شوید.

Welcome to administrator

و در غیر این صورت، به صفحه login منتقل می‌شوید.

توجه: هیچ کاربری نباید بدون وارد نمودن کد کپچا وارد صفحه admin شود.

- **پیاده سازی کد کپچا با استفاده از جدول کپچا**

ابتدا جدول زیر را در دیتابیس خود import کنید.

```
--
-- Table structure for table `captcha`
--
CREATE TABLE `captcha` (
  `id` int(11) NOT NULL auto_increment,
  `answer` varchar(2) NOT NULL,
  `ip_address` varchar(15) NOT NULL,
  `expire` varchar(25) NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=280 ;
--
-- Dumping data for table `captcha`
--
```

در مرحله اول کدکپچارا ایجاد می‌کنیم ونتیجه را درجدول کپچا ذخیره می‌نماییم. برای این کار باید کنترلر **comment** به صورت زیر ایجاد شود.

```
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class comment extends GTNA_controller{
function index(){ $this->qv(); }
function library_load()
        {
          $this->library('GTNA_Session');
          $this->library('GTNA_Captcha');
          $this->library('GTNA_Security');
          $this->load->model('captcha_model');
          $this->library('captcha_model');
        }
function qv()
        {
                $this->library_load();
                $this->captcha_model->delete_by_ip();
                $this->GTNA_Captcha->simple_captcha();
                $this->captcha_model->insert("20"); //تعیین زمان بر حسب ثانیه برای باطل شدن سوال
                $this->load->view('comment');
        }
```

حالا نوبت به طراحی متد form برای کنترلر comment می‌رسد که سنگین‌ترین وظیفه را بر عهده دارد.

در پایان توضیحات مختصری در مورد هر یک از متد ها می‌دهیم.

```
function form()
{
        $this->library_load();
       $captcha_answer=$this->captcha_model->search_by_ip();
        if(!$captcha_answer)
                $this->qv();
         else
       {
          if(time()>CAPTCHA_EXPIRE){
                $this->GTNA_Session->open();
               $this->GTNA_Session->register('message',"زمان پاسخ گویی شما به اتمام رسیده است");
               $this->captcha_model->delete_by_ip();
               $this->qv();
          }else
           {
                if($captcha_answer)
               {
                $captcha_code_user=$this->GTNA_Security->escape($_POST['captcha_code']);
               if($captcha_code_user==$captcha_answer)
                {
                        echo "پیام شما با موفقیت ثبت شد";
                         $this->captcha_model->delete_by_ip();
                 }else
                {
                        $this->captcha_model->delete_by_ip();
                        $this->qv();
                 }
                }else
                        $this->qv();
         }
       }
   }
}
```

```php
<?php
if( ! defined("SYSTEM_ROOT")) exit("No direct script access allowed");
class captcha_model extends GTNA_Model{
 function insert($time=10)
        {
                $data=array(
                'answer'=>CAPTCHA_RESULT,
                'ip_address'=>$_SERVER['REMOTE_ADDR'],
                'expire'=>time()+$time);
                $this->db->Mysql_open();
                $result=$this->db->Mysql_insert('captcha',$data);
                if($result)
                        return $result;
                else
                        return FALSE;
        }
 function search_by_ip(){
        $this->db->Mysql_open();
        $result=$this->db->Mysql_select('captcha',"ip_address='{$_SERVER['REMOTE_ADDR']}'");
        if($result){
                define('CAPTCHA_EXPIRE',$result[0]->expire);
                return $result[0]->answer;
        }else
                return FALSE;
        }
function delete_by_ip(){
    $this->db->Mysql_open();
    $result=$this->db->Mysql_delete("captcha","ip_address='{$_SERVER['REMOTE_ADDR']}' limit 1");
     if($this->db->Mysql_delete("captcha","ip_address='{$_SERVER['REMOTE_ADDR']}' limit 1"))
                return TRUE;
        else
                return FALSE;
        }
}// END MODEL
```

دقت داشته باشید که نام کلاس مدل و نام فایل مدل باید دقیقا مانند هم باشد این نکته را برای ایجاد کنترلرها نیز الزامی است.

بعدازطراحی مدل بایدviewراایجادکنیم برای ایجادفرمcommentابتدادرمسیرزیرفایلی با نامcomment ایجاد کنید

Application/view/comment.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>comment</title>
</head>
<body>
<?php // کد پی اچ پی زیر وظیفه دارد در صورت باطل شدن کپچا پیام ایجاد شده در نشست را به نمایش در آورد
        if(GTNA_Session::sess_read('message')):
                echo GTNA_Session::sess_read('message');
                GTNA_Session::sess_close();
        endif
?>
<form action="<?php echo SYSTEM_URL."index.php/comment/form"; ?>" method="post">
        <p><textarea cols="50" rows="10">this is a comment</textarea></p>
        <p><input type="text" name="captcha_code" /></p>
        <p><?php echo CAPTCHA_CODE?></p>
        <p><input type="submit" value="send" /></p>
</form>
</body>
</html>
```

مزیت‌های این روش :

۱- جواب کپچا در هیچ نشستی ذخیره نمی‌شود. بنابراین هیچ کد کپچایی برای دزدی نشست وجود ندارد.
۲- برای هر شخص، یک زمان پاسخ به سوال در نظر گرفته شده است. با گذشت زمان از حد مجاز حتی اگر پاسخ صحیح را وارد نماید کد کپچا به عنوان کد باطل شده تلقی می‌شود.
۳- آی پی هر کاربر به عنوان یک رمز عبور برای پاسخ سؤالات در نظر گرفته می شود.
۴- طوری طراحی شده است که هیچ افزونگی در جدول به وجود نیاورد. یعنی برای هر شخص فقط در یک رکورد سؤال ایجاد می‌شود. بنابراین هرچند کاربر به طور همزمان می‌توانند کامنت ارسال کنند.

این کد در عین سادگی به لحاظ امنیتی بسیار ایمن می‌باشد. ما فرض را بر این گذاشته‌ایم که یک ربات جواب سؤال را پیدا کند اما باز هم در ارسال کامنت ناموفق خواهد بود.

چگونه از Sessionها (نشست‌ها) استفاده کنیم؟

جدول متدهای کتابخانه Session به صورت زیر است.

۱- **GTNA_Session->open()**: برای کار با نشست‌ها درون کنترلرها ابتدا کتابخانه‌ی نشست را به صورت زیر فعال می‌کنیم. حالا به تمامی متدها دسترسی دارید.

```
$this->library('GTNA_Session');
$this->GTNA_Session->open();
```

۲- **GTNA_Session->register()**: زمانی که بخواید یک نشست با نام دلخواهتان ایجاد کنید به صورت زیر عمل کنید.

```
$this->GTNA_Session_register('your session name');
```

```
$this->GTNA_Session->register('CAPTCHA',$captcha_code);
```

توجه: به جای آرگومان دوم می‌توانیم هر نوع آرایه‌ای قرار دهیم (یک بعدی، دوبعدی و ...) و آن را درون نشست ذخیره کنیم.

۳- **GTNA_Session->read()**: برای دسترسی به نشست‌هایی که ایجاد کرده‌اید به صورت زیر عمل کنید.

```
Print_r( $this->GTNA_Session->read('CAPTCHA'));
```

توجه: مقدار برگشتی این متد از نوع آرایه می‌باشد.

۴- **GTNA_Session->unregister()**: برای باطل کردن نشست مورد نظرتان استفاده می‌شود. زمانی از این متد استفاده می‌کنیم که به عنوان مثال چند نشست را با نام‌های متفاوت ایجاد کرده‌ایم اما نمی‌خواهیم تمام نشست‌ها باطل شود. به این ترتیب به وسیله نام، نشست مورد نظر آن را از بین می‌بریم.

```
$this->GTNA_Session->register('TEST1','آزمایش شماره ۱');
$this->GTNA_Session->register(TEST2, 'آزمایش شماره ۲');
$this->GTNA_Session->register('TEST3', 'آزمایش شماره ۳');
$this->GTNA_Session->register('TEST4', 'آزمایش شماره ۴');
```

```
echo $this->GTNA_Session->unregister("TEST4");
```

۵- **GTNA_Session->encode()**: زمانی استفاده می‌شود که قصد داریم نشستمان را به صورت کد شده بر روی سرور ذخیره کنیم.

```
$this->GTNA_Session->encode();
```

توجه: این متد هیچ آرگومانی ندارد.

۶- **GTNA_Session->decode()**: اگر بخواهیم نشست ذخیره شده را از حالت کد شده به حالت اولیه بازگردانیم، باید به صورت زیر عمل کنید.

```
echo $this->GTNA_Session->decode('CAPTCHA');
```

به این ترتیب نشست کپچا از حالت کد به حالت اولیه باز می‌گردد.

۷- **GTNA_Session->id()**: برای دسترسی به کدSession مورد استفاده قرار می‌گیرد.

```
echo $this->GTNA_Session->id();
```

۸- **GTNA_Session->new_id()**:برای تغییر کد نشست مورد استفاده قرار می‌گیرد و نحوه استفاده از آن به صورت زیر است.

```
echo $this->GTNA_Session->new_id();
```

۹- **GTNA_Session->RBI()**: (RBI مخفف Read By Index می‌باشد)زمانی استفاده می‌شود که می خواهیم به وسیله اندیس آرایه‌های ذخیره شده در نشست، به مقدار آن دسترسی داشته باشیم.

```
$this->GTNA_Session->register('captcha',array
                                    ('account'=>array('user_name'=>"gtna",'user_pass'=>"12345"),
                                     'profile'=>array('name'=>'hadi','fname'=>"mansoori rad"),
                                     'details'=>"30")
                                    ));
```

```
$data=$this->GTNA_Session->RBI('captcha',"details");
        if($data)
                echo $data;  // خروجی: ۳۰
        else
                echo "not found";
```

```
$data=$this->GTNA_Session->RBI('captcha',"account","user_name");
        if($data)
                echo $data;
        else
                echo "not found";
```

خروجی: gtna

فرض کنید در اندیس details به جای 30، کلمه world را قرار داده‌ایم و می‌خواهیم به حرف d درکلمه world دسترسی پیدا کنیم.

W o r l d
0 1 2 3 4

```
$data=$this->GTNA_Session->RBI('captcha',"details",4);
        if($data)
                echo $data;
        else
                echo "not found";
```

۱۰- GTNA_Session->SRARO(): مخفف SRARO(session Read And Return Object)می

```
$this->GTNA_Session->register('captcha',array
                                ('account'=>array('user_name'=>"gtna",'user_pass'=>"12345"),
                                 'profile'=>array('name'=>'hadi','fname'=>"mansoori rad"),
                                 'details'=>array('expire'=>"30")
                                ));
```

اگر به مثال بالا توجه کنید، می‌بینید که یک آرایه سه بُعدی را درون نشست قرار داده‌ایم. برای اینکه بعدها را به صورت مجزا به نمایش در آورید، فقط کافی است نام آن بُعد را ذکر کنید. مثال زیر برای نمایش خروجی می‌باشد.

```
$data=$this->GTNA_Session->SRARO('captcha');
echo "<pre>";
print_r($data->account) ; // بعد اول
print_r($data->profile) ; //بعد دوم
print_r($data->details) ;//بعد سوم
print_r($data) ;//همه بعدها
```

اگر بخواهیم به یکی از فیلدهای آرایه درون آبجکت ($data) اشاره کنیم، به صورت زیر عمل می‌کنیم.

```
echo "<pre>";
print_r($data->account['user_name']) ; // بعد اول شی دیتا آرایه اکانت فیلد نام کاربری
print_r($data->profile['name']) ; //بعد دوم شی دیتا آرایه پروفایل فیلد نام
print_r($data->details['expire']) ;//بعد سوم شی دیتا آرایه مشخصات فیلد اکسپایر
```

۱۱- GTNA_Session->close(): برای بستن کلیه نشست ها استفاده می شود

۱۲- GTNA_Session::sess_read(): زمانی که در یک کنترلر نشست ایجاد کرده‌اید و قصد دارید از آن نشست درونviewها استفاده کنید از این متد استفاده کنید.

توجه: این متد هم درview و هم درکنترلر قابل استفاده است.

۱۳- GTNA_Session::sess_close(): برای بستن نشست‌ها مورد استفاده قرار می‌گیرد.

جدول متدهای کتابخانه نشست به صورت زیر است:

GTNA_Session->new_id()	GTNA_Session->read()	GTNA_Session->open()
GTNA_Session->SRARO ()	GTNA_Session->decode()	GTNA_Session->close()
GTNA_Session->RBI()	GTNA_Session->encode()	GTNA_Session->register()
GTNA_Session::sess_read()	GTNA_Session->id()	GTNA_Session->unregister()
GTNA_Session::sess_close()		

چگونه از کتابخانه GTNA_Security استفاده کنیم؟

برای استفاده از متدهای کتابخانه ابتدا باید کتابخانه آن را بارگذاری کنید تا بتوانید از متدهای آن استفاده کنید.

۱- متد sanitize_string() دو پارامتر می‌پذیرد شکل کلی آن به صورت زیر است.

```
Sanitize_string(value,sanitize type);
```

اگر این متد را تنها با یک متغیر فراخوانی کنید و مقدار آن را ۱و یا **help** بگذارید، انواع فیلترگذاری بر روی داده‌ها را مشاهده می‌کنید. به قطعه کد زیر و خروجی آن دقت کنید.

```
$this->library('GTNA_Security');
Print_r($this->GTNA_Security->sanitize_string("help"));
```

خروجی:

```
Array
(
    [0] => [The sanitize type]
    [1] => FILTER_SANITIZE_MAGIC_QUOTES
    [2] => FILTER_SANITIZE_SPECIAL_CHARS
    [3] => FILTER_SANITIZE_NUMBER_INT
    [4] => FILTER_SANITIZE_EMAIL
    [5] => FILTER_SANITIZE_ENCODED
    [6] => FILTER_SANITIZE_NUMBER_FLOAT
    [7] => FILTER_SANITIZE_STRING
    [8] => FILTER_SANITIZE_STRIPPED
    [9] => FILTER_SANITIZE_URL
)
```

اگر در مورد نحوه عملکر فیلترهای بالا اطلاعاتی ندارید، باید دانش خود را افزایش دهید تا تشخیص دهید چه زمانی از این فیلترها استفاده نمایید.

یک مثال ساده برای نحوه استفاده از فیلترهای بالا:

```
$data="admin@gtna.net<>\\"<scipt>;";
echo $this->GTNA_Security->sanitize_string("$data "," FILTER_SANITIZE_EMAIL ");
```

متد**strip_tag()**:به وسیله این متد فقط تگ‌هایی اجرا می‌شود که شما به آنها مجوز داده‌اید. به کد زیر توجه کنید.

```
$tag="<p>this is a test</p><strong>GTNAF</strong>";
echo $this->GTNA_Security->strip_tag($tag,"<strong>");
```

به این شکل، فقط تگ strong بر روی کلمه GTNAF اجرا می‌شود. برای اینکه بهتر متوجه شوید، پس از اجرای کد بالا توسط مرورگر سورس آن را مشاهده کنید.

اگر می‌خواهید هر دو تگ اجرا شود، یعنی تگ p وتگ strong به شکل زیر عمل کنید:

```
$tag="<p>this is a test</p><strong>GTNAF</strong>";
echo $this->GTNA_Security->strip_tag($tag,"<strong><p>");
```

متد escape(): همانطور که از نامش پیداست وظیفه معتبرسازی داده‌ها را دارد. بیشترین کاربرد آن در ارسال فرم‌ها و کار با متدPOST,GET است، چون امکان دارد کاربران داده‌های نامعتبر به فرم ارسال کنند. باید آنها را به لحاظ داده‌ای مجاز کنیم و سپس داده‌ها را برای ثبت به لایه دیتا ارسال کنیم و در مرحله آخر عمل درج را انجام دهیم.

```
$data="<script>alert('this is a test');</script>";

echo $this->GTNA_Security->escape($data);
```

خروجی:

```
&#60;script&#62;alert(&#39;this is a test&#39;);&#60;/script&#62;
```

با مشاهده خروجی، می‌بینیدکه به دلیل اینکه داده غیرمجاز است، تمامی تگ‌ها به کدهای نامعتبر تبدیل شده‌اند. به این ترتیب حتی اگر این داده درون جدول ذخیره شود، به عنوان یک رشته به خروجی باز می‌گردد و هیچ خطری برای سایت ندارد. البته باید آنقدر دانش داشته باشید که بتوانید با جستجو درون رشته اسکریپت‌های خطرناک را پیدا کنید و در صورت وجود آنها، داده‌های فرم را به عنوان داده‌های نامعتبر تلقی کنید و پیام مربوطه را به کاربر نمایش دهید. حتی در صورتی که داده مجاز باشد به وسیله متد زیر داده‌ها را در لایه دیتا بررسی کنید.

db->Mysql_sanitize_string();

توجه: متد**db->Mysql_sanitize_string()** فقط درلایه یا داده‌ها و زمانی که به بانک اطلاعاتی متصل هستید عمل می‌کند، در غیر این صورت با پیامی مبنی بر اینکه شما به بانک اطلاعاتی ارتباط ندارید برخورد می‌کنید. برای آشنایی با طرز استفاده از این متد، به قطعه کد زیر توجه کنید.

```
$this->db->open();
$data="<script>alert('this is a test');</script>";
echo $this->db->Mysql_sanitize_string($data);
```

تذکر: تمام فیلد هایی که قرار است درون یک جدول ذخیر شوند، باید از این فیلتر عبور کنند.

متد**GTNA_Security->encrypt()**: وظیفه این متد رمز نگاری داده‌ها می‌باشد. بیشترین زمانی که به این متد نیاز پیدا می‌کنید، برای ذخیره‌سازی پسورد کاربران درون بانک اطلاعاتی می‌باشد. قطعه کد زیر نحوه‌ی استفاده از این متد را بیان می‌کند.

```
$pass="123"
echo $this->GTNA_Security->encrypt($pass);
```

در این قسمت مثال‌هایی در مورد عملیات CRUD برای شما آورده‌ایم تا راحت‌تر بتوانید عملیات‌هایی مانند

درج، حذف، ویرایش و همچنین نحوه خواندن داده‌ها از درون جداول می‌پردازیم.

جدول زیر تمامی متدهایی می‌باشد که در لایه دیتا و یا همان مدل قابل استفاده می‌باشد:

db->Mysql_delete(table,query_string)	db->Mysql_open()
db->Mysql_update(table,array())	db->Mysql_close()
db->Mysql_select(table,query_string)	db->Mysql_insert(table,array)
db->Mysql_sanitiz_string(value)	

توجه: دستورات نسبت به حروف کوچک و بزرگ حساس است یعنی بین Mysqlو Mysql تفاوت قائل می شود.

ابتدا در مسیر زیر تنظیمات مربوط به بانک اطلاعاتی را انجام دهید

application/config/database.php

۱- **db->Mysql_open()**: برای اتصال به بانک اطلاعاتی استفاده می‌شود.

هرزمان که قصد دارید در یک جدول عملیات های درج، حذف، و یا ویرایش انجام دهید قبل از آن باید کد زیر را بنویسید.

```
$this->db->Mysql_open();
```

۲- **db->Mysql_close()**: بعد از انجام هر یک از عملیات های CRUD به وسیله این متد حتماً اتصال با بانک اطلاعاتی را ببندید.

```
$this->db->Mysql_close();
```

۳- **db->Mysql_insert()**: برای درج درون جدول استفاده می‌شود. قطعه کد زیر مثالی در این رابطه می‌باشد.

فرض کنید جدولی با نام user به شکل زیر درون پایگاه داده ایجادکرده‌ایم.

```
$user_name="gtna";
$user_pass=$this->db->mysql_sanitize_encrypt("123");
$email="admin@gtna.net";
$data=array(
        'user_name'=>$user_name,
        'user_pass'=>$user_pass,
        'email'=>$email);
$this->db->Mysql_open();
$this->db->Mysql_insert("user",$data);
$this->db->Mysql_close();
```

اگر به اندیس های آرایه $data وفیلد های جدول نگاه کنید، می‌بینید که دقیقاً مانند یکدیگرند. بنابراین این نکته را در هنگام درج داده‌ها فراموش نکنید، در غیر این صورت با خطا برخورد می‌کنید.

۴- متد (Mysql_update(table,array(: طبیعتاً از این متد برای به‌روزرسانی داده‌ها استفاده می‌شود. برای درک بهتر نحوه عملکرد این متد به مثال زیر توجه نمایید.

فرض کنید id شماره ۱۲ مربوط به رکوردی است که اکنون قصد بروزرسانی آن را داریم.

```
function update($data=NULL){
$user_name="gtna_user"; // تغییر داده
$user_pass=$this->db->mysql_sanitize_encrypt("456");//تغییر داده
$email="gtna@att.net";//تغییر داده
$data=array(
        'user_name'=>$user_name,
        'user_pass'=>$user_pass,
        'email'=>$email);
$this->db->Mysql_open();
$this->db->Mysql_update("user",$data,"12"); // به روز رسانی توسط آی دی کاربر
$this->db->Mysql_close();
```

۵- (Mysql_select(table,query_string: به وسیله این متد عملیات مربوط به واکشی داده‌ها انجام می‌شود به قطعه کد زیر توجه فرمایید.

```
function select(){
        $this->db->open();
        $data=$this->db->Mysql_select('user',"id=12");//خروجی از نوع آبجکت می باشد
        return $data;
}
```

Application/models/select_model.php

حالا باید به وسیله کنترلر داده‌ها را دریافت کنیم:

```
$this ->load->model('select_model');
$this ->library('select_model');
$result['data']= $this ->select_model->select();
$this ->load->view('dislay',$result);
```

Application/views/display.php

نوبت به نوشتن کدهای مربوط به view برای نمایش داده ها می‌رسد.

Display.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>display view</title>
</head>
<body>
```

```
<?php
        foreach ($data as $data)
            for($i=0 ;$i<$data['numrow'];$i++):
                echo $data[$i]->id."<br>";
                echo $data[$i]->user_name."<br>";
                echo $data[$i]->email."<br>";
                echo "_____<br>";
            endfor
?>
</body>
</html>
```

numrow شماره کل رکوردهای موجود در جدول user می باشد که توسط سیستم در اختیار شما قرار می‌گیرد.

۶- متد **Mysql_delete(table,query_string)**: به وسیله این متد می‌توانید عملیات حذف را انجام دهید. برای آشنایی با نحوه استفاده از آن، مثالی ساده آورده‌ایم.

```
function delete(){
    $this->db->Mysql_open();
    If($this ->db->Mysql_delete("user","id=12"))
        return TRUE;
    else
        return FALSE;
    }
```

چگونه فرم‌های درون یک view را به یک متد از کنترلر خاص ارسال کنیم؟

یک **view** با نام **form** در مسیر زیر ایجاد کنید.

Application/views/form.php

کدهای مربوط به فرم را به صورت زیر به یک متد خاص از کنترلر ارسال می‌کنیم. لطفاً به مثال زیر توجه کنید.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>display view</title>
</head>
<body>
```

```
<form action="<?php echo SYSTEM_URL."index.php/welcome/form"; ?>" method="post">
<p><input type="text" name="user_name">
<p><input type="password" name="user_pass " /></p>
<p><input type="submit" value="login" /></p>
</form>
```

اگر به مقدار خاصیت action فرم توجه کنید، متوجه می‌شوید که طبق قانون URLها که قبلاً درمورد آنها توضیحات کامل را داده‌ایم. این فرم به وسیله متد POST به کنترلر welcome و متد form ارسال شده است. بنابراین اگر کد زیر را در متد form بنویسید و سپس فرم را با اطلاعات مربوطه ارسال کنید، یک آرایه از داده‌های ارسال شده از طریق فرم دریافت و نمایش داده می‌شود.

```
print_r($_POST);
```

به این ترتیب می‌توانید داده‌ها را درون کنترلر اعتبارسنجی کنید و سپس در صورت لزوم ازآنها درمدل‌ها استفاده نمایید.

روش استفاده از کتابخانه GTNA_HTML

اگربه وسیله زبان پی اچ پی یک interface ساده طراحی کرده باشید می دانید که کد های HTMLوPHP درهم می شوند و زمانی که بخواهید کدهای قبلی را ویرایش کنید کارتان بسیار دشوار می شود. بنابراین گتنا یک کتابخانه در فریم ورک تعبیه کرده است که تگهای HTML توسط متدهای آن پردازش می شود و پس از اجرا اشیاء به وجود می آیند . به جدول متد های این کتابخانه توجه نمایید .

GTNA_HTML::select(array(property));	GTNA_HTML::input(array(peroperty));
GTNA_HTML::option_property(array(property),'name');	GTNA_HTML::img(array(property));
GTNA_HTML::option('value');	GTNA_HTML::textarea(array(property),value);
GTNA_HTML::end_select(void);	GTNA_HTML::form_open(array(property));
GTNA_HTML::a(array(property),linkname);	GTNA_HTML::form_close(void);

حتما با خود می گویید با این چند متد چطور می توان یک فرم کامل را به وجود آورد؟

۱-**GTNA_HTML::input(array(peroperty))** : تمام اشیایی که با تگ input ایجاد می شوند توسط این متد پردازش و ایجاد می شود همانطور که می بینید یک آرایه از خصوصیات را دریافت می کند و به کد HTML تبدیل می کند .

توجه داشته باشید که تمام کدها را در لایه view بنویسید.

```
<?php//
	$text=array(
		'name'=>'user_name',
		'type'=>'text',
		'class'=>'txt');	//.تمام خصوصیاتی که لازم است را می توانید به آرایه بیافزایید
?>
<?php echo GTNA_HTML::input($text); ?>
```

اگر درقطعه کد بالا درآرایه به فیلد type دقت کنید نوع شیء را مشخص می کند،این فیلد می تواند مقادیر زیر را بپذیرد .

type
- text
- password
- hidden
- file
- checkbox
- radio
- submit
- reset
- image

بنابراین کافی است بجای مقدار فیلد type از مقادیر بالا استفاده کنید ولی ممکن است خاصیت های بعضی از اشیاء فرق داشته باشد کافی است فیلد جدید اضاف کنید و نام آن خاصیت را بنویسد .

۲- GTNA_HTML::select(array(property)) : از این متد برای ایجاد **ComboBox** استفاده می شود اما زمانی که از این متد استفاده می کنید حتما باید از متد GTNA_HTML::option('value'); و در پایان از متد GTNA_HTML::end_select(void); استفاده کنیداگر قصد داریدoptionهارا به همراه خاصیت های خاص ایجاد کنید بایدازمتد;GTNA_HTML::option_ property(array(property)) استفاده کنید .

```
<?php//
	$select=array(
		'name'=>'user_name',
		'type'=>'select',
		'id'=>'select-style');
?>
<?php echo GTNA_HTML::select($select); ?>
	<?php echo GTNA_HTML::option('مرد'); ?>
	<?php echo GTNA_HTML::option('زن'); ?>
<?php echo GTNA_HTML::end_select(); ?>
```

به این شکل یک ComboBox ساختیم اما اگر قصد دارید که option های خود را با استایل و خاصیت های مورد نظرتان ایجاد کنید به شکل زیر عمل کنید .

```
<?php//
        $select=array(
                'name'=>'user_name',
                'type'=>'select',
                'id'=>'select-style');
        $option_property=array(
                        'id'=>'option-style',
                        '...','...')//.می توانید خاصیت های دیگر را به عنوان فیلد به آرایه بیافزاید
?>
<?php echo GTNA_HTML::select($select); ?>
        <?php echo GTNA_HTML::option_property($option_property,'مرد'); ?>
        <?php echo GTNA_HTML::option_property($option_property,'زن'); ?>
<?php echo GTNA_HTML::end_select(); ?>
```

۳- GTNA_HTML::img(array(property)) : از این متد برای ایجاد تصویر استفاده می شود

```
<?php//
        $image=array(
                'height'=>'400px',
                'width'=>'400px',
                'src'=>IMAGE_URL."/your-pic.jpg");
?>
<?php echo GTNA_HTML::img($image); ?>
```

می توانید از طریق کنترلر بعد از آپلود تصاویر مشخصات آن را به صورت یک آرایه به view ارسال کرد و سپس به وسیله متد GTNA_HTML::img(array(property)) تصاویر را ایجاد کنید در چنین موقعیت هایی استفاده از این متد معنا دار می شود ولی به صورت معمول کاربرد خاصی ندارد و می توان از همان تگ img استفاده کنید.

۴- GTNA_HTML::a(array(property),linkname):مشخص است که از این متد برای ایجاد لینک استفاده می شود .

```
<?php
	$link=array(
		'title'=>'my link',
		'href'=>'http://www.google.com'
		);
?>
<?php echo GTNA_HTML::a($link,'ورود به گوگل'); ?>
```

۵- GTNA_HTML::textarea(array(property),value): برای ایجاد textarea استفاده می شود به مثال زیر توجه کنید.

```
<?php
	$textarea=array(
		'cols'=>'40',
		'rows'=>'30',
		'...','...' //  در صورت لزوم خاصیت های دیگر را اضاف کنید
		);
?>
<?php echo GTNA_HTML::textarea($textarea,'این یک مثال است'); ?>
```

۶- GTNA_HTML::form_open(array(property)): برای ایجاد فرم استفاده می شود و در پایان این متد باید متد GTNA_HTML::form_close(void); قرار گیرد.

```
<?php
	$form=array(
		'action'=>SYSTEM_URL."index.php/controller/method",
		'method'=>'post',
		'...','...' //  در صورت لزوم خاصیت های دیگر را اضاف کنید
		);
?>
<?php echo GTNA_HTML::form_open($textarea,'این یک مثال است'); ?>
<?php echo GTNA_HTML::input(array('type'=>'text','name'=>'user_name')); ?>
 <?php echo GTNA_HTML::input(array('type'=>'text','password'=>'password')); ?>
<?php echo GTNA_HTML::input(array('type'=>'submit','value'=>'ورود به سایت')); ?>
<?php echo GTNA_HTML::form_close() ?>
```

شاید این تفکر به ذهنتان خطور کند که چرا دستورات ساده تر نشده اند مثلا برای ایجا یک شیء مانند text از input استفاده نکنیم و با دستور جداگانه بدون مشخص نمودن type آن را ایجاد کنیم در جواب به این سوال می گوییم بله این امکان وجود داشت ولی به لحاظ SEO مشکل به وجود می آمد و پویایی یا بهتر است بگوییم آزادی از طراح گرفته می شد.

اگر به تمام اشیاء ایجاد شده دقت کنید می بینید که شیء چند انتخابی وجود ندارد برای ایجاد این شی به صورت زیر عمل کنید .

```
<?php
	$select=array(
		'multiple'=> 'multiple',
		); // هر صفتی دیگری می توانید بیافزایید
?>
<?php echo GTNA_HTML::select($select); ?>
	<?php echo GTNA_HTML::option('مرد'); ?>
	<?php echo GTNA_HTML::option('زن'); ?>
<?php echo GTNA_HTML::end_select(); ?>
```

چه تغییراتی در 1.2.GFB حاصل شده است؟

۱- افزودن کتابخانه GTNA_HTML جهت کار با اشیاء بر روی فرم

۲- روش جلوگیری از ذخیره سازی صفحات

۳- روش جلوگیری از ارسال فرم های خارجی

۴- رفع مشکلات لایه View

۵- رفع مشکلات لایه Model