Documentación SIIGS/TES

Contenido

Paquete default Clases

Clase Index

[line 3]

- **Package** default

Constructor *void* función Index::__construct() *[line 5]*

- **Access** public

void función Index::index() *[line 10]*

- **Access** public

Paquete CodeIgniter Elementos procedurales

Form_validation.php

CodeIgniter

An open source application development framework for PHP 5.1.6 or newer

- **Package** CodeIgniter
- **Author** ExpressionEngine Dev Team
- **Copyright** Copyright (c) 2008 - 2011, EllisLab, Inc.
- **Link** http://codeigniter.com
- **Since** Version 1.0
- **Filesource** Source Code for this file
- **License** http://codeigniter.com/user_guide/license.html

Pagination.php

CodeIgniter

An open source application development framework for PHP 5.1.6 or newer

- **Package** CodeIgniter
- **Author** ExpressionEngine Dev Team
- **Copyright** Copyright (c) 2008 - 2011, EllisLab, Inc.
- **Link** http://codeigniter.com
- **Since** Version 1.0
- **Filesource** Source Code for this file
- **License** http://codeigniter.com/user_guide/license.html

template.php

CodeIgniter

An open source application development framework for PHP 4.3.2 or newer

- **Package** CodeIgniter
- **Author** ExpressionEngine Dev Team
- **Copyright** Copyright (c) 2006, EllisLab, Inc.
- **Link** http://codeigniter.com
- **Since** Version 1.0
- **Filesource** Source Code for this file
- **License** http://codeigniter.com/user_guide/license.html

Paquete CodeIgniter Clases

Clase CI_Form_validation

[line 27]

Form Validation Class

- **Package** CodeIgniter
- **Sub-Package** Libraries
- **Author** ExpressionEngine Dev Team
- **Link** http://codeigniter.com/user_guide/libraries/form_validation.html

CI_Form_validation::$CI

mixed = [line 29]

- **Access** protected

CI_Form_validation::$error_string

mixed = '' [line 36]

- **Access** protected

CI_Form_validation::$_config_rules

mixed = array() *[line 31]*

- **Access** protected

CI_Form_validation::$_error_array

mixed = array() *[line 32]*

- **Access** protected

CI_Form_validation::$_error_messages

mixed = array() *[line 33]*

- **Access** protected

CI_Form_validation::$_error_prefix

mixed = ' ' *[line 34]*

- **Access** protected

CI_Form_validation::$_error_suffix

mixed = ' ' *[line 35]*

- **Access** protected

CI_Form_validation::$_field_data

mixed = array() *[line 30]*

- **Access** protected

CI_Form_validation::$_safe_form_data

mixed = FALSE *[line 37]*

- **Access** protected

Constructor *void* función CI_Form_validation::__construct([$rules = array()]) *[line 42]*
Parámetros de la función:

- **$rules**

Constructor

- **Access** public

bool función CI_Form_validation::alpha($str) *[line 1099]*
Parámetros de la función:

- *string* **$str**

Alpha

- **Access** public

bool función CI_Form_validation::alpha_dash($str) *[line 1127]*
Parámetros de la función:

- *string* **$str**

Alpha-numeric with underscores and dashes

- **Access** public

bool función CI_Form_validation::alpha_numeric($str) *[line 1113]*
Parámetros de la función:

- *string* **$str**

Alpha-numeric

- **Access** public

bool función CI_Form_validation::decimal($str) *[line 1184]*
Parámetros de la función:

- *string* **$str**

Decimal number

- **Access** public

string función CI_Form_validation::encode_php_tags($str) *[line 1373]*
Parámetros de la función:

- *string* **$str**

Convert PHP tags to entities

- **Access** public

void función CI_Form_validation::error([$field = ''], [$prefix = ''], [$suffix = '']) *[line 208]*
Parámetros de la función:

- *string* **$field** the field name
- **$prefix**
- **$suffix**

Get Error Message

Gets the error message associated with a particular field

- **Access** public

str función CI_Form_validation::error_string([$prefix = ''], [$suffix = '']) *[line 240]*
Parámetros de la función:

- *string* **$prefix**

- *string* **$suffix**

Error String

Returns the error messages as a string, wrapped in the error delimiters

- **Access** public

bool función CI_Form_validation::exact_length($str, $val) *[line 1019]*

Parámetros de la función:

- *string* **$str**
- *value* **$val**

Exact Length

- **Access** public

bool función CI_Form_validation::greater_than($str, $min) *[line 1198]*

Parámetros de la función:

- *string* **$str**
- **$min**

Greather than

- **Access** public

bool función CI_Form_validation::integer($str) *[line 1170]*
Parámetros de la función:

- *string* **$str**

Integer

- **Access** public

bool función CI_Form_validation::is_natural($str) *[line 1234]*
Parámetros de la función:

- *string* **$str**

Is a Natural number (0,1,2,3, etc.)

- **Access** public

bool función CI_Form_validation::is_natural_no_zero($str) *[line 1248]*
Parámetros de la función:

- *string* **$str**

Is a Natural number, but not a zero (1,2,3, etc.)

- **Access** public

bool función CI_Form_validation::is_numeric($str) *[line 1156]*
Parámetros de la función:

- *string* **$str**

Is Numeric

- **Access** public

bool función CI_Form_validation::is_unique($str, $field) *[line 951]*
Parámetros de la función:

- *string* **$str**
- *field* **$field**

Match one field to another

- **Access** public

bool función CI_Form_validation::less_than($str, $max) *[line 1216]*
Parámetros de la función:

- *string* **$str**
- **$max**

Less than

- **Access** public

bool función CI_Form_validation::matches($str, $field) *[line 929]*
Parámetros de la función:

- *string* **$str**
- *field* **$field**

Match one field to another

- **Access** public

bool función CI_Form_validation::max_length($str, $val) *[line 994]*
Parámetros de la función:

- *string* **$str**
- *value* **$val**

Max Length

- **Access** public

bool función CI_Form_validation::min_length($str, $val) *[line 969]*
Parámetros de la función:

- *string* **$str**
- *value* **$val**

Minimum Length

- **Access** public

bool función CI_Form_validation::numeric($str) *[line 1141]*
Parámetros de la función:

- *string* **$str**

Numeric

- **Access** public

string función CI_Form_validation::prep_for_form([$data = '']) *[line 1292]*
Parámetros de la función:

- *string* **$data**

Prep data for form

This function allows HTML to be safely shown in a form. Special characters are converted.

- **Access** public

string función CI_Form_validation::prep_url([$str = '']) *[line 1321]*
Parámetros de la función:

- *string* **$str**

Prep URL

- **Access** public

bool función CI_Form_validation::regex_match($str, $regex) *[line 909]*

Parámetros de la función:

- *string* **$str**
- *regex* **$regex**

Performs a Regular Expression match test.

- **Access** public

bool función CI_Form_validation::required($str) *[line 887]*

Parámetros de la función:

- *string* **$str**

Required

- **Access** public

bool función CI_Form_validation::run([$group = '']) *[line 281]*
Parámetros de la función:

- **$group**

Run the Validator
This function does all the work.

- **Access** public

string función CI_Form_validation::set_checkbox([$field = ''], [$value = ''], [$default = FALSE]) *[line 847]*
Parámetros de la función:

- *string* **$field**
- *string* **$value**
- **$default**

Set Checkbox
Enables checkboxes to be set to the value the user selected in the event of an error

- **Access** public

void función CI_Form_validation::set_error_delimiters([$prefix = ' '], [$suffix = ' ']) *[line 189]*
Parámetros de la función:

- *string* **$prefix**
- *string* **$suffix**

Set The Error Delimiter

Permits a prefix/suffix to be added to each error message

- **Access** public

string función CI_Form_validation::set_message($lang, [$val = '']) *[line 165]*

Parámetros de la función:

- *string* **$lang**
- *string* **$val**

Set Error Message

Lets users set their own error messages on the fly. Note: The key name has to match the function name that it corresponds to.

- **Access** public

string función CI_Form_validation::set_radio([$field = ''], [$value = ''], [$default = FALSE]) *[line 803]*

Parámetros de la función:

- *string* **$field**
- *string* **$value**
- **$default**

Set Radio

Enables radio buttons to be set to the value the user selected in the event of an error

- **Access** public

void función CI_Form_validation::set_rules($field, [$label = ''], [$rules = '']) *[line 74]*

Parámetros de la función:

- *mixed* **$field**
- *string* **$label**
- **$rules**

Set Rules

This function takes an array of field names and validation rules as input, validates the info, and stores it

- **Access** public

string función CI_Form_validation::set_select([$field = ''], [$value = ''], [$default = FALSE]) *[line 759]*

Parámetros de la función:

- *string* **$field**
- *string* **$value**
- **$default**

Set Select

Enables pull-down lists to be set to the value the user selected in the event of an error

- **Access** public

void función CI_Form_validation::set_value([$field = ''], [$default = '']) *[line 729]*

Parámetros de la función:

- *string* **$field** the field name

- *string* **$default**

Get the value from a form

Permits you to repopulate a form field with the value it was submitted with, or, if that value doesn't exist, with the default

- **Access** public

string función CI_Form_validation::strip_image_tags($str) *[line 1345]*

Parámetros de la función:

- *string* **$str**

Strip Image Tags

- **Access** public

bool función CI_Form_validation::valid_base64($str) *[line 1275]*

Parámetros de la función:

- *string* **$str**

Valid Base64

Tests a string for characters outside of the Base64 alphabet as defined by RFC 2045 http://www.faqs.org/rfcs/rfc2045

- **Access** public

bool función CI_Form_validation::valid_email($str) *[line 1043]*
Parámetros de la función:

- *string* **$str**

Valid Email

- **Access** public

bool función CI_Form_validation::valid_emails($str) *[line 1057]*
Parámetros de la función:

- *string* **$str**

Valid Emails

- **Access** public

string función CI_Form_validation::valid_ip($ip, [$which = '']) *[line 1085]*
Parámetros de la función:

- *string* **$ip**
- *string* **$which** "ipv4" or "ipv6" to validate a specific ip format

Validate IP Address

- **Access** public

string función CI_Form_validation::xss_clean($str) *[line 1359]*
Parámetros de la función:

- *string* **$str**

XSS Clean

- **Access** public

mixed función CI_Form_validation::_execute($row, $rules, [$postdata = NULL], [$cycles = 0]) *[line 470]*
Parámetros de la función:

- *array* **$row**
- *array* **$rules**
- *mixed* **$postdata**
- *integer* **$cycles**

Executes the Validation routines

- **Access** protected

mixed función CI_Form_validation::_reduce_array($array, $keys, [$i = 0]) *[line 376]*
Parámetros de la función:

- *array* **$array**
- *array* **$keys**
- *integer* **$i**

Traverse a multidimensional $_POST array index until the data is found

- **Access** protected

null función CI_Form_validation::_reset_post_array() *[line 408]*

Re-populate the _POST array with our finalized and processed data

- **Access** protected

string función CI_Form_validation::_translate_fieldname($fieldname) *[line 697]*

Parámetros de la función:

- *string* **$fieldname** the field name

Translate a field name

- **Access** protected

Clase CI_Pagination

[line 27]

Pagination Class

- **Package** CodeIgniter
- **Sub-Package** Libraries
- **Author** ExpressionEngine Dev Team
- **Link** http://codeigniter.com/user_guide/libraries/pagination.html

CI_Pagination::$anchor_class

mixed = '' *[line 61]*

CI_Pagination::$base_url

mixed = '' *[line 29]*

CI_Pagination::$cur_page

mixed = 0 *[line 36]*

CI_Pagination::$cur_tag_close

mixed = '
' *[line 51]*

CI_Pagination::$cur_tag_open

mixed = '
- <a>' *[line 50]*

CI_Pagination::$display_pages

mixed = TRUE *[line 60]*

CI_Pagination::$first_link

mixed = '‹ First' *[line 38]*

CI_Pagination::$first_tag_close

mixed = '
' *[line 46]*

CI_Pagination::$first_tag_open

mixed = '
- ' *[line 45]*

CI_Pagination::$first_url

mixed = '' *[line 49]*

CI_Pagination::$full_tag_close

mixed = '

' *[line 44]*

CI_Pagination::$full_tag_open

mixed = '<ul class="pagination pagination-sm">' *[line 43]*

CI_Pagination::$last_link

mixed = 'Last ›' *[line 41]*

CI_Pagination::$last_tag_close

mixed = '
' *[line 48]*

CI_Pagination::$last_tag_open

mixed = '
- ' *[line 47]*

CI_Pagination::$next_link

mixed = '>' *[line 39]*

CI_Pagination::$next_tag_close

mixed = '
' *[line 53]*

CI_Pagination::$next_tag_open

mixed = '
- ' *[line 52]*

CI_Pagination::$num_links

mixed = 2 *[line 35]*

CI_Pagination::$num_tag_close

mixed = '
' *[line 57]*

CI_Pagination::$num_tag_open

mixed = '
- ' *[line 56]*

CI_Pagination::$page_query_string

mixed = FALSE *[line 58]*

CI_Pagination::$per_page

mixed = 10 *[line 34]*

CI_Pagination::$prefix

mixed = '' *[line 30]*

CI_Pagination::$prev_link

mixed = '<' *[line 40]*

CI_Pagination::$prev_tag_close

mixed = '
' *[line 55]*

CI_Pagination::$prev_tag_open

mixed = '
- ' *[line 54]*

CI_Pagination::$query_string_segment

mixed = 'per_page' *[line 59]*

CI_Pagination::$suffix

mixed = '' *[line 31]*

CI_Pagination::$total_rows

mixed = 0 *[line 33]*

CI_Pagination::$uri_segment

mixed = 3 *[line 42]*

CI_Pagination::$use_page_numbers

mixed = FALSE *[line 37]*

Constructor *void* función CI_Pagination::__construct([$params = array()]) *[line 69]*

Parámetros de la función:

- *array* **$params** initialization parameters

Constructor

- **Access** public

string función CI_Pagination::create_links() *[line 115]*

Generate the pagination links

- **Access** public

void función CI_Pagination::initialize([$params = array()]) *[line 93]*

Parámetros de la función:

- *array* **$params** initialization parameters

Initialize Preferences

- **Access** public

Clase CI_Template

[line 33]

CodeIgniter Template Class

This class is and interface to CI's View class. It aims to improve the interaction between controllers and views. Follow @link for more info

- **Package** CodeIgniter
- **Sub-Package** Libraries
- **Author** Colin Williams
- **Version** 1.4.1
- **Copyright** Copyright (c) 2008, Colin Williams.
- **Link** http://www.williamsconcepts.com/ci/libraries/template/index.html

CI_Template::$CI

mixed = *[line 35]*

CI_Template::$config

mixed = *[line 36]*

CI_Template::$css

mixed = array() *[line 45]*

CI_Template::$js

mixed = array() *[line 44]*

CI_Template::$master

mixed = *[line 38]*

CI_Template::$output

mixed = *[line 43]*

CI_Template::$parser

mixed = 'parser' *[line 46]*

CI_Template::$parser_method

mixed = 'parse' *[line 47]*

CI_Template::$parse_template

mixed = FALSE *[line 48]*

CI_Template::$regions

mixed = array(
'_scripts' => array(),'_styles'=>array(),) *[line 39]*

CI_Template::$template

mixed = *[line 37]*

Constructor *void* función CI_Template::CI_Template() *[line 59]*

Constructor

Loads template configuration, template regions, and validates existence of default template

- **Access** public

TRUE función CI_Template::add_css($style, [$type = 'link'], [$media = FALSE]) *[line 492]*
Parámetros de la función:

- *string* **$style** CSS file to link, import or embed
- *string* **$type** 'link', 'import' or 'embed'
- *string* **$media** media attribute to use with 'link' type only, FALSE for none

Dynamically include CSS in the template

NOTE: This function does NOT check for existence of .css file

- **Access** public

TRUE función CI_Template::add_js($script, [$type = 'import'], [$defer = FALSE]) *[line 433]*
Parámetros de la función:

- *string* **$script** script to import or embed
- *string* **$type** 'import' to load external file or 'embed' to add as-is
- *boolean* **$defer** TRUE to use 'defer' attribute, FALSE to exclude it

Dynamically include javascript in the template

NOTE: This function does NOT check for existence of .js file

- **Access** public

void función CI_Template::add_region($name, [$props = array()]) *[line 233]*
Parámetros de la función:

- *string* **$name** Name to identify the region
- *array* **$props** Optional array with region defaults

Dynamically add region to the currently set template

- **Access** public

void función CI_Template::add_template($group, $template, [$activate = FALSE]) *[line 129]*

Parámetros de la función:

- *string* **$group** array key to access template settings
- *array* **$template** properly formed
- **$activate**

Dynamically add a template and optionally switch to it

- **Access** public

void función CI_Template::empty_region($name) *[line 260]*

Parámetros de la función:

- *string* **$name** Name to identify the region

Empty a region's content

- **Access** public

void función CI_Template::initialize($props) *[line 155]*

Parámetros de la función:

- *array* **$props** configuration array

Initialize class settings using config settings

- **Access** public

void función CI_Template::load([$region = NULL], [$buffer = FALSE]) *[line 602]*

Parámetros de la función:

- **$region**
- **$buffer**

Load the master template or a single region

DEPRECATED!

Use render() to compile and display your template and regions

void función CI_Template::parse_view($region, $view, [$data = NULL], [$overwrite = FALSE]) *[line 391]*

Parámetros de la función:

- *string* **$region** region to write to
- *string* **$view** view file to parse
- *array* **$data** variables to pass into view for parsing
- *boolean* **$overwrite** FALSE to append to region, TRUE to overwrite region

Parse content from a View to a region with the Parser Class

- **Access** public

void función CI_Template::render([$region = NULL], [$buffer = FALSE], [$parse = FALSE]) *[line 548]*

Parámetros de la función:

- *string* **$region** optionally opt to render a specific region
- *boolean* **$buffer** FALSE to output the rendered template, TRUE to return as a string. Always TRUE when $region is supplied
- **$parse**

Render the master template or a single region

- **Access** public

void función CI_Template::set_master_template($filename) *[line 106]*

Parámetros de la función:

- *string* **$filename** filename of new master template file

Set master template

- **Access** public

void función CI_Template::set_parser($parser, [$method = NULL]) *[line 282]*

Parámetros de la función:

- *string* **$parser** name of parser class to load and use for parsing methods
- **$method**

Set parser

- **Access** public

void función CI_Template::set_parser_method($method) *[line 303]*
Parámetros de la función:

- *string* **$method** name of parser class member function to call when parsing

Set parser method

- **Access** public

void función CI_Template::set_regions($regions) *[line 199]*
Parámetros de la función:

- *array* **$regions** properly formed regions array

Set regions for writing to

- **Access** public

void función CI_Template::set_template($group) *[line 83]*
Parámetros de la función:

- *string* **$group** array key to access template settings

Use given template settings

- **Access** public

void función CI_Template::write($region, $content, [$overwrite = FALSE]) *[line 320]*
Parámetros de la función:

- *string* **$region** region to write to
- *string* **$content** what to write
- *boolean* **$overwrite** FALSE to append to region, TRUE to overwrite region

Write contents to a region

- **Access** public

void función CI_Template::write_view($region, $view, [$data = NULL], [$overwrite = FALSE]) *[line 352]*
Parámetros de la función:

- *string* **$region** region to write to
- *string* **$view** view file to use
- *array* **$data** variables to pass into view
- *boolean* **$overwrite** FALSE to append to region, TRUE to overwrite region

Write content from a View to a region. 'Views within views'

- **Access** public

Paquete Libreria Clases

Clase Menubuilder

[line 12]

Menu Builder

- **Package** Libreria
- **Sub-Package** Clase
- **Author** Pascual

Constructor *void* función Menubuilder::__construct() *[line 23]*

- **Access** public

string función Menubuilder::build([$todos = false]) *[line 35]*

Parámetros de la función:

- *boolean* **$todos** Establece si se debe devolver todos los elementos del menu

Construye el menu basandose en los permisos del usuario logeado

- **Static**

- **Access** public

void función Menubuilder::crearMenu(&$strMenu, [$id_padre = 'NULL'], [$todos = false], $strMenu) *[line 68]*
Parámetros de la función:

- *string* **$strMenu** Variable pasada por referencia donde se guardará la cadena de ul y li
- *string* **$id_padre** ID del padre de los elementos del arbol de menu
- **&$strMenu**
- **$todos**

Función recursiva que recorre todo el árbol de elementos del menu

- **Static**
- **Access** public

void función Menubuilder::isGranted($accion, $strMenu, $id_padre) *[line 123]*
Parámetros de la función:

- *string* **$strMenu** Variable pasada por referencia donde se guardará la cadena de ul y li
- *string* **$id_padre** ID del padre de los elementos del arbol de menu
- **$accion**

Función que valida si se visualiza o no la accion especificada (usada en views)

- **Static**
- **Access** public

Clase Graph

[line 11]

Controlador Objetos

- **Package** Libreria
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Graph::__construct() *[line 13]*

- **Access** public

void función Graph::graph_init($title, $titulo, $array, $label, [$grafica = ""], [$nacimiento = ""]) *[line 36]*

Parámetros de la función:

- *string* **$title** Titulo de la pagina en el navegador
- *string* **$titulo** titulo o nombre a mostrar en la vista al crear el grafico
- *array* **$array** datos que se graficaran ver ejemplo
- *array* **$label** datos con las etiquetas que se muestran en cada grafica
- *array* **$grafica** tipo de grafiva puede ser :time, basic, axis, bars, bars-h, stacked, horizontal ó pie
- *string* **$nacimiento** si se necesita mostrar fechas enviar la fecha inicial

Crea una grafica en el lugar que se llame

- **Access** public

void función Graph::map([$lugar = "Chiapas"], [$zoom = 6], [$rewrite = 0], [$datos = ""]) *[line 93]*

Parámetros de la función:

- *string* **$lugar** Especifica el lugar donde se centra el mapa

- *int* **$zoom** Especifica el zoom de acercamiento en el mapa
- *boolean* **$rewrite** Si se desea que sea una pagina o estar enbebida en otra 0=embebido 1=pagina
- *array* **$datos** datos a mostrar en el mapa

crea un objeto mapa con la ayuda de la api de google

- **Access** public

Clase Obtenercurp

[line 11]

Controlador Objeto

- **Package** Libreria
- **Sub-Package** Controlador
- **Author** Eliecer

Obtenercurp::$estados

```
$estados = array(
        array(
          "AGUASCALIENTES"=>"AS",
          "BAJA CALIFORNIA NTE"=>"BC",
          "BAJA CALIFORNIA NORTE"=>"BC",
          "BAJA CALIFORNIA"=>"BC",
          "BAJA CALIFORNIA SUR"=>"BS",
          "CAMPECHE"=>"CC",
          "COAHUILA"=>"CL",
          "COLIMA"=>"CM",
          "CHIAPAS"=>"CS",
          "CHIHUAHUA"=>"CH",
          "DISTRITO FEDERAL"=>"DF",
          "DURANGO"=>"DG",
          "GUANAJUATO"=>"GT",
```

```
    "GUERRERO"=>"GR",
    "HIDALGO"=>"HG",
    "JALISCO"=>"JC",
    "MEXICO"=>"MC",
    "MICHOACAN"=>"MN",
    "MORELOS"=>"MS",
    "NAYARIT"=>"NT",
    "NUEVO LEON"=>"NL",
    "OAXACA"=>"OC",
    "PUEBLA"=>"PL",
    "QUERETARO"=>"QT",
    "QUINTANA ROO"=>"QR",
    "SAN LUIS POTOSI"=>"SP",
    "SINALOA"=>"SL",
    "SONORA"=>"SR",
    "TABASCO"=>"TC",
    "TAMAULIPAS"=>"TS",
    "TLAXCALA"=>"TL",
    "VERACRUZ"=>"VZ",
    "ZACATECAS"=>"ZS",
    "EXTERIOR MEXICANO"=>"SM",
    "NACIDO EN EL EXTRANJERO"=>"NE"
  )) [line 17]
```

Arreglo con los estados y sus abreviaturas, sirven para el cálculo de la CURP

- **Access** public

Constructor *void* función Obtenercurp::__construct() *[line 58]*

- **Access** public

echo función Obtenercurp::calculacurp($paterno, $materno, $nombre, $dia, $mes, $year, $sexo, $estado, [$regresar = ""]) *[line 280]*

Parámetros de la función:

- *string* **$paterno** Apellido paterno de la persona
- *string* **$materno** Apellido materno
- *string* **$nombre** Nombre o nombres
- *int* **$dia** Dia de nacimiento
- *int* **$mes** Mes de nacimiento
- *int* **$year** Año de nacimiento
- *string* **$sexo** Sexo
- *string* **$estado** Lugar de nacimiento

- *string* **$regresar** Tipo de retorno =1 return array !=1 json

Calcula la curp y el rfc con los datos proporcionados

- **Access** public

echo función Obtenercurp::calcular_curp($paterno, $materno, $nombre, $dia, $mes, $year, $sexo, $estado, [$regresar = ""]) *[line 207]*

Parámetros de la función:

- *string* **$paterno** Apellido paterno de la persona
- *string* **$materno** Apellido materno
- *string* **$nombre** Nombre o nombres
- *int* **$dia** Dia de nacimiento
- *int* **$mes** Mes de nacimiento
- *int* **$year** Año de nacimiento
- *string* **$sexo** Sexo
- *string* **$estado** Lugar de nacimiento
- *string* **$regresar** Tipo de retorno =1 return array !=1 json

Calcula la curp y el rfc con los datos proporcionados

- **Access** public

echo función Obtenercurp::curp($paterno, $materno, $nombre, $dia, $mes, $year, $sexo, $estado, [$regresar = ""]) *[line 79]*

Parámetros de la función:

- *string* **$paterno** Apellido paterno de la persona
- *string* **$materno** Apellido materno
- *string* **$nombre** Nombre o nombres
- *int* **$dia** Dia de nacimiento
- *int* **$mes** Mes de nacimiento
- *int* **$year** Año de nacimiento
- *string* **$sexo** Sexo

- *string* **$estado** Lugar de nacimiento
- *string* **$regresar** Tipo de retorno =1 return array !=1 json

Consulta si la curp existe en la base de datos de la condusef

- **Access** public

Clase Tree

[line 11]

Controlador Objeto

- **Package** Libreria
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Tree::__construct() *[line 13]*

- **Access** public

void función Tree::create($title, $titulo, $seleccion, $tipo, $menu, $id, $text, [$idarbol = 1], [$nivel = 1], [$omitidos = array(NULL)], [$seleccionable = ""]) *[line 42]*

Parámetros de la función:

- *string* **$title** Titulo de la pagina en el navegador
- *string* **$titulo** titulo o nombre a mostrar en la vista al crear el arbol
- *int* **$seleccion** tipo de seleccion 1=select. 2=multiselect. 3=multiselect Parcial (Marcan los padres del hijo seleccionado
- *string* **$tipo** tipo de control radio o check

- *boolean* **$menu** si se desea mostrar el menu
- *string* **$id** id del campo oculto donde se guarda el id del elemento seleccionado
- *string* **$text** id del campo donde se muestra la descripcion del elemento seleccionado
- *string* **$idarbol** id del arbol donde comenzara la creacion
- *string* **$nivel** nivel en el que se empezara a mostrar informacion
- *string* **$omitidos** nodos que no se deben mostrar en la vista al crear el arbol
- *string* **$seleccionable** determina si un nodo se puede o no seleccionar

Crea el arbol y lo muestra en la view

- **Access** public

Paquete Session Clases

Clase Session

[line 12]

CodeIgniter Native Session Library

- **Package** Session
- **Sub-Package** Libraries
- **Author** Bo-Yi Wu (appleboy) < appleboy.tw@gmail.com>
- **Author** Marko Martinovi? < marko@techytalk.info>

Session::$ci

mixed = *[line 16]*

- **Access** protected

Session::$flashdata_key

mixed = 'flash' *[line 18]*

- **Access** protected

Session::$sess_expiration

mixed = '' *[line 15]*

- **Access** protected

Session::$sess_namespace

mixed = '' *[line 14]*

- **Access** protected

Session::$store

mixed = array() *[line 17]*

- **Access** protected

Constructor *void* función Session::__construct([$config = array()]) *[line 28]*

Parámetros de la función:

- *array* **$config** config preferences

Constructor

- **Access** public

array función Session::all_userdata() *[line 190]*

Fetch all session data

- **Access** public

string función Session::flashdata($key) *[line 245]*
Parámetros de la función:

- *string* **$key**

Fetch a specific flashdata item from the session array

- **Access** public

void función Session::is_expired() *[line 106]*
Check if session is expired

- **Access** public

void función Session::keep_flashdata($key) *[line 225]*
Parámetros de la función:

- *string* **$key**

Keeps existing flashdata available to next request.

- **Access** public

void función Session::sess_create() *[line 85]*

Create Session

- **Access** public

void función Session::sess_destroy() *[line 119]*

Destroy session

- **Access** public

void función Session::set_flashdata([$newdata = array()], [$newval = '']) *[line 204]*

Parámetros de la función:

- *mixed* **$newdata**
- *string* **$newval**

Add or change flashdata, only available until the next request

- **Access** public

void función Session::set_userdata([$data = array()], [$value = '']) *[line 151]*

Parámetros de la función:

- *array* **$data** list of data to be stored
- *object value* **$value** to be stored if only one element is passed

Set value for specific user data element

- **Access** public

void función Session::unset_userdata([$data = array()]) *[line 169]*
Parámetros de la función:

- *array* **$data** list of data to be removed

remove array value for specific user data element

- **Access** public

object element función Session::userdata($value) *[line 131]*
Parámetros de la función:

- *string* **$value** element key

Get specific user data element

- **Access** public

Paquete SIIGS Clases

Clase Accion

[line 10]

Controlador Accion

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Accion::__construct() *[line 12]*

- **Access** public

void función Accion::delete($id) *[line 246]*

Parámetros de la función:

- *int* **$id** Este parámetro no puede ser nulo

Acción para eliminar una acción, recibe el id de la acción a eliminar

- **Access** public

void función Accion::index([$pag = 0]) *[line 35]*
Parámetros de la función:

- *int* **$pag** Numero de registro para el paginador

Acción por default del controlador, carga la lista
de acciones disponibles y una lista de opciones

- **Access** public

void función Accion::insert() *[line 112]*
Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente

- **Access** public

void función Accion::update($id) *[line 170]*
Parámetros de la función:

- *int* **$id** Este parámetro no puede ser nulo

Acción para preparar la actualización de una acción ya existente,
recibe un ID para obtener los valores de esa acción y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente

- **Access** public

void función Accion::view($id) *[line 83]*
Parámetros de la función:

- *int* **$id** Este parametro no puede ser nulo

Acción para visualizar información de una acción específica, obtiene el objeto acción por medio del id proporcionado.

- **Access** public

Clase Ayuda

[line 11]

Controlador Ayuda

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Ayuda::__construct() *[line 13]*

- **Access** public

void función Ayuda::index([$id_controlador_accion = null]) *[line 25]*

Parámetros de la función:

- *int* **$id_controlador_accion** Id del controlador accion, es opcional

Función que renderiza el contenido de la ayuda dependiendo de la sección donde se encuentre

- **Access** public

Clase Bitacora

[line 11]

Controlador Bitacora

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Bitacora::__construct() *[line 13]*

- **Access** public

void función Bitacora::index([$pag = 0]) *[line 35]*

Parámetros de la función:

- *int* **$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de la bitacora, con su correspondiente paginación

permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico

- **Access** public

void función Bitacora::validateExistUsuario($id_usuario) *[line 367]*

Parámetros de la función:

- *int* **$id_usuario** ID del usuario

callback utilizado por las acciones create y update para validar la existencia de un usuario

- **Access** public

void función Bitacora::view($id) *[line 301]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Catalogo

[line 10]

Controlador Catalogo

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Catalogo::__construct() *[line 12]*

- **Access** public

void función Catalogo::checkpk($campos) *[line 745]*

Parámetros de la función:

- *string* **$campos**

Acción para revisar registros repetidos en las columnas designadas como primary key

- **Access** public

void función Catalogo::checkTypeData($campo, $type) *[line 776]*

Parámetros de la función:

- *string* **$campo** Nombre del campo a revisar
- *string* **$type** Define el tipo de dato del campo

Acción para revisar si los tipos de datos coinciden con los datos contenidos en la tabla temporal que fueron tomados del CSV

- **Access** public

void función Catalogo::delete($nombre) *[line 805]*

Parámetros de la función:

- *string* **$nombre**

Acción para eliminar un catálogo, recibe el nombre del catalogo a eliminar

- **Access** public

void función Catalogo::index() *[line 35]*

Acción por default del controlador, carga la lista de catálogos disponibles y una lista de opciones No recibe parámetros

- **Access** public

void función Catalogo::insert() *[line 548]*

Acción para preparar la inserción de nuevos catálogos , realiza la validación del formulario del lado del servidor y crea la estructura para el catálogo, crea la tabla y obtiene los datos a partir de la tabla tmp_catalogo

- **Access** public

void función Catalogo::load(0) *[line 130]*
Parámetros de la función:

- *$_FILES[]* **0** archivocsv Variable pasada por POST con el archivo csv para cargar datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP Guarda en la tabla tmp_catalogos toda la estructura del CSV e imprime las columnas del archivo. Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Catalogo::loadupdate($nombrecat, [$update = false]) *[line 230]*
Parámetros de la función:

- *$_FILES[]* **$nombrecat** archivocsv Variable pasada por POST con el archivo csv para cargar datos
- **$update**

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar.
Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Catalogo::update($nombre, [$pag = 0]) *[line 681]*
Parámetros de la función:

- *string* **$nombre**
- *int* **$pag** Numero de registro para el paginador

Acción para preparar la actualizacion de un catálogo ya existente,
recibe un string para obtener los valores del catalogo y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y servidor

- **Access** public

void función Catalogo::view($nombre, [$pag = 0]) *[line 68]*
Parámetros de la función:

- *string* **$nombre** Este parámetro no puede ser nulo
- *int* **$pag** Numero de registro para el paginador

Acción para visualizar información de un catálogo específico, obtiene el objeto catálogo por medio del nombre proporcionado

- **Access** public

void función Catalogo::_array_unique_recursive($arr) *[line 531]*
Parámetros de la función:

- *array* **$arr** Arreglo multidimensional

_array_unique_recursive Revisa valores duplicados en arreglos multidimensionales

- **Access** public

Clase CatalogoCsv

[line 10]

Controlador CatalogoCsv

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función CatalogoCsv::__construct() *[line 12]*

- **Access** public

Boolean función CatalogoCsv::ActivaEnCatalogo($id, $catalogo, $activo) *[line 403]*

Parámetros de la función:

- *Int* **$id** Es el id del registro en el catalogo
- *String* **$catalogo** para determinar a que catalogo se va a agregar o quitar el registro
- *Boolean* **$activo** False para quitar del catalogo, true para agregarlo

*** Accion para activar o desactivar elementos en los catalogos indicados en el parametro Solo se permite su acceso por medio de peticiones AJAX**

- **Access** public

void función CatalogoCsv::checkpk($campos) *[line 494]*

Parámetros de la función:

- *string* **$campos**

Acción para revisar registros repetidos en las columnas designadas como primary key

- **Access** public

void función CatalogoCsv::createTableAgeb() *[line 587]*

Acción para ejecutar la creación de la tabla Asu Ageb No recibe parámetros

- **Access** public

void función CatalogoCsv::createTableGeo() *[line 555]*

Acción para ejecutar la creación de la tabla poblacional No recibe parámetros

- **Access** public

void función CatalogoCsv::createTableHemoGlobina() *[line 620]*

Acción para ejecutar la creación de la tabla Asu Ageb No recibe parámetros

- **Access** public

void función CatalogoCsv::createTablePob() *[line 522]*

Acción para ejecutar la creación de la tabla poblacional No recibe parámetros

- **Access** public

void función CatalogoCsv::index() *[line 35]*

Acción por default del controlador, carga la lista de catálogoscsv disponibles y una lista de opciones No recibe parámetros

- **Access** public

void función CatalogoCsv::loadupdate($nombrecat, [$update = false]) *[line 119]*

Parámetros de la función:

- *$_FILES[]* **$nombrecat** archivocsv Variable pasada por POST con el archivo csv para cargar datos
- *boolean* **$update** Define si se actualizará la DB o solo se hará la primera revisión de datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP

compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función CatalogoCsv::update($nombre, [$pag = 0]) *[line 447]*

Parámetros de la función:

- *string* **$nombre**
- *int* **$pag** Numero de registro para el paginador

Acción para preparar la actualizacion de un catálogo ya existente,
recibe un string para obtener los valores del catalogo y mostrarlos en la vista update , realiza la validacion del formulario del lado del cliente y servidor

- **Access** public

void función CatalogoCsv::view($nombre, [$pag = 0]) *[line 68]*
Parámetros de la función:

- *string* **$nombre** Este parametro no puede ser nulo
- *int* **$pag** Numero de registro para el paginador

Acción para visualizar información de un catálogo específico, obtiene el objeto catalogocsv por medio del nombre proporcionado

- **Access** public

void función CatalogoCsv::_array_unique_recursive($arr) *[line 377]*
Parámetros de la función:

- *array* **$arr**

_array_unique_recursive Revisa valores duplicados en arreglos que contienen arreglos

- **Access** public

Clase Catalogo_x_raiz

[line 10]

Controlador Raiz_x_Catalogo

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Catalogo_x_raiz::__construct() *[line 12]*

- **Access** public

boolean función Catalogo_x_raiz::check($id) *[line 174]*

Parámetros de la función:

- *type* **$id** Id del catalogo_x_raiz

Acción que sirve para revisar inconsistencias en el arbol de segmentacion Recibe como parámetro el catálogo x raiz y revisa que todos los registros tengan un correspondiente en el catálogo padre.
Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Catalogo_x_raiz::delete($id) *[line 226]*
Parámetros de la función:

- *int* **$id**

Acción para eliminar un catálogo en el arbol, recibe el id del catálogo en la raiz a eliminar

- **Access** public

void función Catalogo_x_raiz::insert([$id = 0]) *[line 63]*
Parámetros de la función:

- **$id**

Acción para preparar la inserción de nuevas raices para catálogos , realiza la validación del formulario del lado cliente

- **Access** public

void función Catalogo_x_raiz::view($id) *[line 36]*
Parámetros de la función:

- *int* **$id** Este parametro no puede ser nulo

Acción para visualizar información de una raiz_x_catalogo específica, obtiene el

objeto raiz_x_catalogo por medio del id proporcionado.

- **Access** public

Clase Cie10

[line 10]

Controlador Cie10

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Cie10::__construct() *[line 12]*

- **Access** public

Boolean función Cie10::ActivaEnCatalogo($id, $catalogo, $activo) *[line 166]*

Parámetros de la función:

- *Int* **$id** Es el id del registro en el catalogo
- *String* **$catalogo** para determinar a que catalogo se va a agregar o quitar el registro
- *Boolean* **$activo** False para quitar del catalogo, true para agregarlo

*** Accion para activar o desactivar elementos en los catalogos IRA EDA Consultas Solo se permite su acceso por medio de peticiones AJAX**

- **Access** public

Boolean función Cie10::AgregaEnCatalogo($id, $catalogo, $activo) *[line 121]*
Parámetros de la función:

- *Int* **$id** Es el id del registro en el catalogo de CIE10
- *String* **$catalogo** para determinar a que catalogo se va a agregar o quitar el registro
- *Boolean* **$activo** False para quitar del catalogo, true para agregarlo

*** Accion para agregar elementos del catalogo cie10 a los catalogos de EDA, IRA y Consultas dependiendo de los Solo se permite su acceso por medio de peticiones AJAX**

- **Access** public

void función Cie10::index([$pag = 0]) *[line 36]*
Parámetros de la función:

- *int* **$pag** Numero de registro para el paginador

Acción por default del controlador, carga la lista de datos disponibles en el cie10 y una lista de opciones

- **Access** public

void función Cie10::insert([$update = false]) *[line 346]*
Parámetros de la función:

- *$_FILES[]* **$update** archivocsv Variable pasada por POST con el archivo csv para cargar

datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP compara los datos recibidos con los datos que contiene actualmente el catálogo, regresa como resultado las filas nuevas y las filas a modificar

Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Cie10::load(0) *[line 209]*

Parámetros de la función:

- *$_FILES[]* **0** archivocsv Variable pasada por POST con el archivo csv para cargar datos

Acción para cargar datos desde un archivo CSV, recibe el stream desde las variables PHP

Guarda en la tabla tmp_catalogos toda la estructura del CSV e imprime las columnas del archivo Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Cie10::update($id) *[line 266]*

Parámetros de la función:

- *int* **$id**

Acción para preparar la actualización de un registro del CIE10,

recibe un id para obtener los valores del catálogo y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y servidor

- **Access** public

void función Cie10::view($cat) *[line 84]*
Parámetros de la función:

- *string* **$cat** Nombre del catalogo a mostrar

*** Accion para mostrar información de los catalogos IDE , ERA y Consultas**

- **Access** public

void función Cie10::_array_unique_recursive($arr) *[line 578]*
Parámetros de la función:

- *array* **$arr**

_array_unique_recursive Revisa valores duplicados en arreglos que contienen arreglos

- **Access** public

Clase Controlador

[line 10]

Controlador Controlador

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Controlador::__construct() *[line 12]*

- **Access** public

void función Controlador::accion($id) *[line 300]*

Parámetros de la función:

- *int* **$id**

Acción para preparar la actualización de acciones asignadas a un controlador, recibe un ID para obtener las acciones asignadas a ese controlador y mostrarlos en la vista update

- **Access** public

void función Controlador::delete($id) *[line 365]*

Parámetros de la función:

- *int* **$id**

Acción para eliminar un controlador, recibe el id del controlador a eliminar

- **Access** public

Object función Controlador::getGroupPermissions($entorno, $grupo) *[line 400]*
Parámetros de la función:

- *int* **$entorno**
- *int* **$grupo**

Acción para servir un array de objetos con los permisos asignados a un entorno y grupo determinados, esta accion solo es accedida por peticiones AJAX y devuelve un objeto JSON Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Controlador::help($idControladorAccion, $id_controlador_accion) *[line 422]*
Parámetros de la función:

- *int* **$id_controlador_accion**
- **$idControladorAccion**

Establece el texto de ayuda para el controlador accion

- **Access** public

void función Controlador::index([$pag = 0]) *[line 36]*
Parámetros de la función:

- *int* **$pag** Numero de registro para el paginador

Acción por default del controlador, carga la lista de controladores disponibles y una lista de opciones Recibe un parametro en caso de filtrado por entornos

- **Access** public

void función Controlador::insert([$id = FALSE]) *[line 138]*
Parámetros de la función:

- **$id**

Acción para preparar la insercion de nuevos controladores , realiza la validacion del formulario del lado cliente

- **Access** public

void función Controlador::update($id) *[line 205]*
Parámetros de la función:

- *int* **$id**

Acción para preparar la actualización de un controlador ya existente,
recibe un ID para obtener los valores de ese controlador y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente

- **Access** public

void función Controlador::view($id) *[line 109]*
Parámetros de la función:

- *int* **$id** Este parametro no puede ser nulo

Acción para visualizar información de un controlador específico, obtiene el objeto controlador por medio del id proporcionado.

- **Access** public

Clase Entorno

[line 11]

Controlador Entorno

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Entorno::__construct() *[line 13]*

- **Access** public

void función Entorno::delete($id) *[line 290]*
Parámetros de la función:

- *int* **$id**

Acción para eliminar un entorno, recibe el id del entorno a eliminar

- **Access** public

void función Entorno::index() *[line 35]*

Acción por default del controlador, carga la lista de entornos disponibles y una lista de opciones No recibe parámetros

- **Access** public

void función Entorno::insert() *[line 94]*

Acción para preparar la inserción de nuevos entornos , realiza la validación del formulario del lado cliente y del lado servidor para evitar entornos duplicados

- **Access** public

void función Entorno::update($id) *[line 208]*

Parámetros de la función:

- *int* **$id**

Acción para preparar la actualización de un entorno ya existente,

recibe un ID para obtener los valores de ese entorno y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente y del servidor para evitar datos duplicados

- **Access** public

void función Entorno::view($id) *[line 65]*
Parámetros de la función:

- *int* **$id** Este parametro no puede ser nulo

Acción para visualizar de un entorno específico, obtiene el objeto entorno por medio del id proporcionado.

- **Access** public

boolean función Entorno::_ExistEntorno($nombre_entorno) *[line 153]*
Parámetros de la función:

- *string* **$nombre_entorno** Revisa si este valor ya existe como un entorno

Acción para validar que no exista previamente el entorno a insertar (Esta acción no puede ser accedida desde el navegador)

- **Access** public

boolean función Entorno::_ExistEntornoUpdate($nombre_entorno) *[line 178]*
Parámetros de la función:

- *string* **$nombre_entorno** Revisa si este valor ya existe como un entorno

Acción para validar que no exista previamente el entorno a actualizar

esta acción revisa si el nombre a usar ya existe en la base excepto el mismo objeto a actualizar (Esta acción no puede ser accedida desde el navegador)

- **Access** public

Clase Errorlog

[line 11]

Controlador Errorlog

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Errorlog::__construct() *[line 13]*

- **Access** public

void función Errorlog::index([$pag = 0]) *[line 34]*

Parámetros de la función:

- *int* **$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de la tabla error, con su correspondiente paginación

permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, muestra enlaces para ver detalles de un elemento especifico

- **Access** public

void función Errorlog::view($id) *[line 141]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Grupo

[line 10]

Controlador Grupo

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor *void* función Grupo::__construct() *[line 12]*

- **Access** public

void función Grupo::delete($id) *[line 195]*
Parámetros de la función:

- *int* **$id** id del grupo a eliminar

Solicita la eliminación del grupo recibido

- **Access** public

void función Grupo::index([$pag = 0]) *[line 33]*
Parámetros de la función:

- *int* **$pag** número de página a visualizar (paginación)

1) Visualiza los grupos existentes para su interacción CRUD 2) En caso de detectar un texto a buscar se filtran los grupos existentes acorde a la búsqueda

- **Access** public

void función Grupo::insert() *[line 109]*
1) Prepara el formulario para la inserción de un grupo nuevo 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Grupo::update($id) *[line 152]*
Parámetros de la función:

- *int* **$id** id del grupo a modificar

1) Prepara el formulario para la modificación de un grupo existente 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Grupo::view($id) *[line 81]*
Parámetros de la función:

- *int* **$id** id del grupo a visualizar

Visualiza los datos del grupo recibido

- **Access** public

boolean función Grupo::_ifGroupExists($name) *[line 222]*
Parámetros de la función:

- *string* **$name** nombre del grupo a validar

Callback para validar que un nombre de grupo no se duplique

- **Access** public

Clase Menu

[line 11]

Controlador Menu

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Menu::__construct() *[line 21]*

- **Access** public

void función Menu::delete($id) *[line 362]*

Parámetros de la función:

- *int* **$id** ID del elemento a eliminar

Eliminar el registro especificado por el id

- **Access** public

void función Menu::index([$pag = 0]) *[line 47]*

Parámetros de la función:

- *int* **$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de la menu, con su correspondiente paginación

permite hacer filtrados por Usuario, Entorno, Controlador, Acción y Fecha, permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico

- **Access** public

void función Menu::insert([$id = null]) *[line 130]*

Parámetros de la función:

- **$id**

Muestra el formulario para crear un nuevo registro en la menu, las variables se obtienen por el metodo POST

- **Access** public

void función Menu::update($id) *[line 225]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos

- **Access** public

void función Menu::view($id) *[line 326]*
Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Permiso

[line 10]

Controlador Permiso

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor *void* función Permiso::__construct() *[line 12]*

- **Access** public

void función Permiso::index($id) *[line 34]*
Parámetros de la función:

- *int* **$id** id del grupo del cual se obtendrán (y actualizar si aplica) los permisos asignados

1) Visualiza los entornos existentes para su selección
2) Al seleccionar un entorno se obtienen los controladores_x_accion existentes y se indica sobre cuales el grupo tiene permisos asignados 3) Elimina los permisos asignados al grupo anteriormente e inserta los asignados recientemente

- **Access** public

Clase Raiz

[line 11]

Controlador Raiz

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función Raiz::__construct() *[line 13]*

- **Access** public

void función Raiz::createasu($id) *[line 316]*
Parámetros de la función:

- *int* **$id**

Acción para crear el ASU a partir de una raiz Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Raiz::delete($id) *[line 284]*

Parámetros de la función:

- *int* **$id**

Acción para eliminar una raiz, recibe el id de la raiz a eliminar

- **Access** public

Object función Raiz::getChildrenFromLevel($idarbol, $nivel, $claves) *[line 632]*

Parámetros de la función:

- *Int* **$idarbol** parametro pasado por POST y determina el arbol a consultar
- *Int* **$nivel** parametro pasado por POST y determina el nivel superior a desglosar en el arbol
- *Array* **$claves** Este parametro es pasado por POST y es la lista de valores a preseleccionar en el arbol

Accion para regresar el arbol de segmentacion determinado, el objeto regresado contiene estructura de arbol y es consumida solamente por peticiones AJAX

- **Access** public

Object función Raiz::getDataKeyValue($idarbol, $nivel, [$filtro = 0]) *[line 738]*

Parámetros de la función:

- *int* **$idarbol**
- *Int* **$nivel** Nivel de desglose de información requerida
- *Int* **$filtro** (Opcional) filtrar por un valor determinado

Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro

- **Throws** Exception Si ocurre error al recuperar datos de la base de datos
- **Access** public

Object función Raiz::getDataTreeFromId($claves, $desglose) *[line 596]*

Parámetros de la función:

- *Array* **$claves** Este parametro es pasado por POST y es la lista de valores a consultar
- *Int* **$desglose** parametro pasado por POST y determina si se requiere información adicional

Accion para regresar la descripción e informacion adicional de un arreglo de ID's desde el arbol de segmentacion

- **Access** public

Object función Raiz::getTreeBlock($idarbol, $nivel, $seleccionados, $seleccionable, $seleccionables, $omitidos) *[line 678]*

Parámetros de la función:

- *int* **$idarbol** ID del arbol (el arbol usado por la TES es el 1)
- *int* **$nivel** nivel del arbol que se desea obtener

- *array* **$seleccionados** se especifica si dentro del arreglo de retorno, hay valores preseleccionados
- *bool* **$seleccionable** especifica si los elementos del arbol pueden ser seleccionados
- *array* **$seleccionables** especifica que niveles del arbol pueden ser seleccionados
- *array* **$omitidos** especifica niveles omitidos dentro del arbol (Si hay un nivel intermedio omitido, los hijos de este nivel son agregados como hijos de su nivel inmediato superior)

Sirve para obtener bloques del arbol de segmentación única ASU

solo se puede acceder por peticiones AJAX, los parametros son pasados por GET

- **Access** public

void función Raiz::index() *[line 62]*

Acción por default del controlador, carga la lista de Raices disponibles y una lista de opciones No recibe parámetros

- **Access** public

void función Raiz::iniciarasu($id) *[line 38]*

Parámetros de la función:

- *type* **$id** ID del asu

Crea los archivos JSON necesarios para iniciar el ASU en caché y agilizar su carga

- **Access** public

void función Raiz::insert() *[line 134]*

Acción para preparar la inserción de nuevas acciones , realiza la validación del formulario del lado cliente

- **Access** public

void función Raiz::update($id) *[line 188]*

Parámetros de la función:

- *int* **$id**

Acción para preparar la actualización de una raiz ya existente,

recibe un ID para obtener los valores de esa raiz y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente

- **Access** public

void función Raiz::updateasu($id) *[line 439]*

Parámetros de la función:

- *int* **$id**

Acción para actualizar el ASU a partir de una raiz Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

void función Raiz::view($id) *[line 94]*

Parámetros de la función:

- *int* **$id** Este parametro no puede ser nulo

Acción para visualizar información de una raiz específica, obtiene el objeto raiz por medio del id proporcionado.

- **Access** public

Clase ReglaVacuna

[line 10]

Controlador ReglaVacuna

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Geovanni

Constructor *void* función ReglaVacuna::__construct() *[line 12]*

- **Access** public

void función ReglaVacuna::delete($id) *[line 333]*

Parámetros de la función:

- *int* **$id**

Acción para eliminar una regla, recibe el id de la regla a eliminar

- **Access** public

void función ReglaVacuna::index() *[line 35]*

Acción por default del controlador, carga la lista de reglas de vacunas disponibles y una lista de opciones No recibe parámetros

- **Access** public

void función ReglaVacuna::insert() *[line 95]*

Acción para preparar la insercion de nuevas reglas , realiza la validación del formulario del lado cliente

- **Access** public

void función ReglaVacuna::update($id) *[line 203]*

Parámetros de la función:

- *int* **$id**

Acción para preparar la actualización de una regla ya existente,

recibe un ID para obtener los valores de esa regla y mostrarlos en la vista update , realiza la validación del formulario del lado del cliente

- **Access** public

void función ReglaVacuna::view($id) *[line 66]*
Parámetros de la función:

- *int* **$id** Este parametro no puede ser nulo

Acción para visualizar información de una regla específica, obtiene el objeto regla_vacuna por medio del id proporcionado.

- **Access** public

Clase Usuario

[line 10]

Controlador Usuario

- **Package** SIIGS
- **Sub-Package** Controlador
- **Author** Rogelio

Usuario::$mas

mixed = "" *[line 773]*

Acción para hacer autologin en otro sistema

- **Access** public

Constructor *void* función Usuario::__construct() *[line 12]*

- **Access** public

void función Usuario::automatic_access() *[line 774]*

- **Access** public

redirect función Usuario::cerrar_etab() *[line 809]*

Acción para cerrar el login en otro sistema

void función Usuario::delete($id) *[line 325]*

Parámetros de la función:

- *int* **$id** id del usuario a eliminar

Solicita la eliminación del usuario recibido

- **Access** public

void función Usuario::form_init() *[line 384]*

Object función Usuario::getActivesByGroup($grupo) *[line 752]*

Parámetros de la función:

- *int* **$grupo**

Acción para servir un array de objetos con los usuarios activos por grupo AJAX y

devuelve un objeto JSON

- **Access** public

cookies función Usuario::get_galleta() *[line 864]*

Obtiene las cookies que se generan en la session

token función Usuario::get_token($url, $var, [$valor = ""], [$num = ""], [$count = ""], [$par = ""]) *[line 820]*

Parámetros de la función:

- **$url**
- **$var**
- **$valor**
- **$num**
- **$count**
- **$par**

Acción para obtener el token oculto en el login

void función Usuario::index([$pag = 0]) *[line 126]*

Parámetros de la función:

- *int* **$pag** número de página a visualizar (paginación)

1) Visualiza los usuarios existentes para su interacción CRUD 2) En caso de detectar un texto a buscar se filtran los usuarios existentes acorde a la búsqueda

- **Access** public

void función Usuario::insert() *[line 197]*

1) Prepara el formulario para la inserción de un usuario nuevo 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Usuario::load_update() *[line 655]*

- **Access** public

void función Usuario::login() *[line 33]*

Ofrece el inicio de sesión

- **Access** public

void función Usuario::logout() *[line 108]*

Termina la sesión

- **Access** public

void función Usuario::remember() *[line 393]*

- **Access** public

void función Usuario::reset() *[line 588]*

- **Access** public

void función Usuario::send_mail($subject, $body, $from, $rto, $correo, $CC, $CCO, $adj) *[line 556]*
Parámetros de la función:

- **$subject**
- **$body**
- **$from**
- **$rto**
- **$correo**
- **$CC**
- **$CCO**
- **$adj**

- **Access** public

void función Usuario::token($str) *[line 518]*
Parámetros de la función:

- **$str**

- **Access** public

void función Usuario::update($id) *[line 262]*
Parámetros de la función:

- *int* **$id** id del usuario a modificar

1) Prepara el formulario para la modificación de un usuario existente 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Usuario::update_info() *[line 684]*

- **Access** public

void función Usuario::view($id) *[line 170]*
Parámetros de la función:

- *int* **$id** id del usuario a visualizar

Visualiza los datos del usuario recibido

- **Access** public

boolean función Usuario::_ifUserExists($username) *[line 352]*
Parámetros de la función:

- *string* **$username** nombre de usuario a validar

Callback para validar que un nombre de usuario no se duplique

- **Access** public

Clase Accion_model

[line 11]

Modelo Accion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Accion_model::__construct() *[line 129]*

- **Access** public

boolean función Accion_model::delete() *[line 268]*

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Accion_model::getAll() *[line 145]*

Devuelve todos los registros de la tabla acciones

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Accion_model::getById($id) *[line 192]*
Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de una accion por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Accion_model::getDescripcion() *[line 79]*

- **Access** public

void función Accion_model::getId() *[line 64]*
****

- **Access** public

void función Accion_model::getMetodo() *[line 87]*

- **Access** public

string|*boolean* función Accion_model::getMsgError([$value = 'usr'], $value,) *[line 114]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Accion_model::getNombre() *[line 72]*

- **Access** public

int función Accion_model::getNumRows() *[line 171]*

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función Accion_model::insert() *[line 213]*

Inserta en la tabla accion la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Accion_model::setDescripcion($value) *[line 83]*

Parámetros de la función:

- **$value**

- **Access** public

void función Accion_model::setId($value) *[line 68]*
Parámetros de la función:

- **$value**

- **Access** public

void función Accion_model::setMetodo($value) *[line 91]*
Parámetros de la función:

- **$value**

- **Access** public

void función Accion_model::setNombre($value) *[line 75]*
Parámetros de la función:

- **$value**

- **Access** public

void función Accion_model::setOffset($value) *[line 95]*

Parámetros de la función:

- **$value**

  - **Access** public

void función Accion_model::setRows($value) *[line 98]*

Parámetros de la función:

- **$value**

  - **Access** public

boolean función Accion_model::update() *[line 240]*

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Ageb_model

[line 11]

Modelo Ageb

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Ageb_model::__construct() *[line 36]*

- **Access** public

boolean|*string* función Ageb_model::getMsgError([$type = 'usr']) *[line 60]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Ageb_model::process() *[line 82]*

Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion

- **Access** public

Object función Ageb_model::searchageb($idlocalidad, $like) *[line 193]*

Parámetros de la función:

- *int* **$idlocalidad** Id del ASU de la localidad
- **$like**

Devuelve una lista de AGEBS de la localidad pasada como parámetro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Ageb_model::searchUM($idlocalidad, $ageb) *[line 162]*

Parámetros de la función:

- *int* **$idlocalidad** Id del ASU de la localidad
- *string* **$ageb** Ageb

Devuelve la información de una UM de acuerdo a su localidad y ageb

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase ArbolSegmentacion_model

[line 11]

Modelo ArbolSegmentacion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función ArbolSegmentacion_model::__construct() *[line 45]*

- **Access** public

Array función ArbolSegmentacion_model::convertType($arbol, $seleccionable, $seleccionados) *[line 398]*

Parámetros de la función:

- *Array* **$arbol** fila array de valores
- *bool* **$seleccionable** Seleccionable opcion para que este elemento sea seleccionable
- **$seleccionados**

Convierte el tipo de arreglo para enviarlo como se debe recibir en el cliente de Javascript

- **Access** public

object con función ArbolSegmentacion_model::getById($item) *[line 372]*

Parámetros de la función:

- *int* **$item** item seleccionado

Obtener el elemento del ASU por medio de su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getChildrenFromId($id, [$omitidos = array()]) *[line 331]*

Parámetros de la función:

- *int* **$id** Id del elemento en el ASU
- *Array* **$omitidos** int $omitidos Array de niveles omitidos

Accion para devolver los hijos de un elemento en el ASU a partir de su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getChildrenFromLevel($idarbol, $nivel, [$omitidos = array()], [$seleccionados = array()], [$seleccionables = array()]) *[line 706]*

Parámetros de la función:

- *Int* **$idarbol** Id del arbol a crear
- *Int* **$nivel** Nivel de segmentacion desde la cual se desarrolla el arbol
- *Array* **$omitidos** int $omitidos Array de niveles omitidos
- *Array* **$seleccionados** int $seleccionados Array de elementos seleccionados
- *Array* **$seleccionables** int $seleccionables Array de niveles que son seleccionables

Accion para devolver el esquema completo del ASU a partir de un nivel especificado, niveles omitidos y elementos preseleccionados

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ArbolSegmentacion_model::getCluesFromId($id) *[line 299]*

Parámetros de la función:

- *int* **$id** Id del elemento en el asu

  * @return Object un arreglo con la estructura del arbol

*

Obtiene las unidades medicas correspondientes a un ID de un elemento independientemente su nivel en el ASU

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getDataKeyValue($idarbol, $nivel, [$filtro = 0]) *[line 1031]*

Parámetros de la función:

- *int* **$idarbol** Id del arbol a buscar
- *Int* **$nivel** Nivel de desglose de información requerida
- *Int* **$filtro** (Opcional) filtrar por un valor determinado

Accion para obtener los registros de un ASU determinado en cierto nivel y con un ID de filtro

- **Throws** Exception Si ocurre error al recuperar datos de la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getDescripcionById($claves, [$desglose = 0]) *[line 935]*

Parámetros de la función:

- *Int* **$desglose** Nivel de desglose de información requerida
- *Array* **$claves** int $claves arreglo de valores a recuperar

Accion para obtener la descripcion e información adicional del elemento en el ASU

- **Throws** Exception Si ocurre error al recuperar datos de la base de datos
- **Access** public

void función ArbolSegmentacion_model::getListChildrenLevel($resultadotemp, [$clave2 = 0]) *[line 640]*

Parámetros de la función:

- **$resultadotemp**
- **$clave2**

- **Access** public

string|*boolean* función ArbolSegmentacion_model::getMsgError([$value = 'usr'], $value,) *[line 30]*

Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

Object función ArbolSegmentacion_model::getTree($idarbol, $nivel, [$nivelesocultos = array()]) *[line 123]*

Parámetros de la función:

- *int* **$idarbol** ID del arbol a crear
- *int* **$nivel** Nivel de segmentacion desde el cual se iniciará a desarrollar el arbol
- *$nivelesocultos* **$nivelesocultos** Array con los niveles que se deben ocultar

Regresa el objeto del arbol de segmentacion

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getTreeBlock($idarbol, $nivel, [$seleccionados = array()], $seleccionable, $elegido, [$omitidos = array()], [$seleccionables = array()]) *[line 441]*

Parámetros de la función:

- *Int* **$idarbol** Id del arbol a crear
- *Int* **$nivel** Nivel de segmentacion desde la cual se desarrolla el arbol
- *Array* **$seleccionados** int $omitidos Array de niveles omitidos
- *Array* **$seleccionable** int $seleccionados Array de elementos seleccionados
- *Array* **$elegido** int $seleccionables Array de niveles que son seleccionables
- *Array* **$omitidos** int $elegido Clave seleccionada para obtener sus hijos
- **$seleccionables**

Accion para devolver un bloque del ASU a partir de un nivel especificado o una clave seleccionada, niveles omitidos y elementos preseleccionados

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getTreeBlockData($idarbol, $nivel, $elegido) *[line 221]*

Parámetros de la función:

- *int* **$idarbol** ID del arbol a crear
- *int* **$nivel** Nivel de segmentacion desde el cual se iniciará a desarrollar el arbol
- *$nivelesocultos* **$elegido** Array con los niveles que se deben ocultar

Regresa el objeto del arbol de segmentacion por nivel o hijos de un elemento seleccionado

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ArbolSegmentacion_model::getUMParentsById($clave) *[line 63]*

Parámetros de la función:

- *int* **$clave** Clave de la unidad medica u elemento en el ASU

Regresa la información de los padres de una unidad medica en el ASU

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ArbolSegmentacion_model::_addSelectedItems($datos, $nivel, $seleccionados, $seleccionables) *[line 863]*

Parámetros de la función:

- **$datos**
- **$nivel**
- **$seleccionados**
- **$seleccionables**

- **Access** public

void función ArbolSegmentacion_model::_addSelectedItems_($datos, $seleccionados, $nivel, $omitidos, $seleccionables) *[line 894]*

Parámetros de la función:

- **$datos**
- **$seleccionados**
- **$nivel**

- **$omitidos**
- **$seleccionables**

- **Access** public

Clase Bitacora_model

[line 11]

Modelo Bitacora

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Bitacora_model::__construct() *[line 93]*

- **Access** public

void|boolean función Bitacora_model::addFilter($columna, $condicion, $valor) *[line 398]*

Parámetros de la función:

- *string* **$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, parametros, id_controlador_accion)
- *string* **$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access** public

int función Bitacora_model::delete([$id = null]) *[line 278]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

int función Bitacora_model::deleteByFilter() *[line 317]*
Elimina el conjunto de registros que cumplen con el o los criterios de filtrado

- **Access** public

array función Bitacora_model::getAll([$offset = null], [$row_count = null]) *[line 467]*
Parámetros de la función:

- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

object|boolean función Bitacora_model::getById($id) *[line 349]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro de la bitacora que tiene el ID especificado

- **Access** public

void función Bitacora_model::getFecha_hora() *[line 134]*

- **Access** public

void función Bitacora_model::getId() *[line 114]*

- **Access** public

void función Bitacora_model::getId_controlador_accion() *[line 154]*

- **Access** public

void función Bitacora_model::getId_usuario() *[line 124]*

- **Access** public

boolean|*string* función Bitacora_model::getMsgError([$type = 'usr']) *[line 185]*
Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

int función Bitacora_model::getNumRows() *[line 514]*
Obtiene el numero total de registros en la tabla Bitacora en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Bitacora_model::getParametros() *[line 144]*

- **Access** public

boolean función Bitacora_model::insert($path, $parametros) *[line 208]*
Parámetros de la función:

- *string* **$path** Concatenación de directorio del proyeto, clase y metodo, unidos por dos dobles puntos '::'
- *string* **$parametros**

Inserta a la bitacora la información porporcionada

- **Static**
- **Access** public

void función Bitacora_model::resetFilter() *[line 451]*

Elimina todos los filtros registrados

- **Access** public

void función Bitacora_model::setFecha_hora($fecha_hora) *[line 139]*

Parámetros de la función:

- **$fecha_hora**

- **Access** public

void función Bitacora_model::setId($id) *[line 119]*

Parámetros de la función:

- **$id**

- **Access** public

void función Bitacora_model::setId_accion($id_accion) *[line 169]*
Parámetros de la función:

- **$id_accion**

    - **Access** public

void función Bitacora_model::setId_controlador($id_controlador) *[line 164]*
Parámetros de la función:

- **$id_controlador**

    - **Access** public

void función Bitacora_model::setId_controlador_accion($id_controlador_accion) *[line 159]*
Parámetros de la función:

- **$id_controlador_accion**

    - **Access** public

void función Bitacora_model::setId_usuario($id_usuario) *[line 129]*
Parámetros de la función:

- **$id_usuario**

- **Access** public

void función Bitacora_model::setParametros($parametros) *[line 149]*
Parámetros de la función:

- **$parametros**

- **Access** public

boolean función Bitacora_model::update([$id = null]) *[line 241]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase CatalogoCsv_model

[line 11]

Modelo CatalogoCsv

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función CatalogoCsv_model::__construct() *[line 128]*

- **Access** public

boolean función CatalogoCsv_model::activaEnCatalogo($id, $catalogo, $valor) *[line 255]*

Parámetros de la función:

- *int* **$id** el id del registro en el catalogo
- *string* **$catalogo** nombre del catalogo donde se realizara la operacion
- *boolean* **$valor** agregar o eliminar el registro del catalogo

Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas

- **Throws** Exception Si ocurre algun error al consultar y modificar la base de datos
- **Access** public

boolean función CatalogoCsv_model::checkPk($campo) *[line 282]*

Parámetros de la función:

- *string* **$campo** (varios campos delimitados por |)

Revisa en la base de datos por registros duplicados en los campos pasados por parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función CatalogoCsv_model::getAll() *[line 165]*

Devuelve una lista con los catalogos existentes en la DB

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función CatalogoCsv_model::getAllData($nombrecat) *[line 186]*

Parámetros de la función:

- *string* **$nombrecat** Nombre del catalogo

Devuelve los datos de un catalogo pasado como parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función CatalogoCsv_model::getByName($nombre) *[line 213]*

Parámetros de la función:

- *string* **$nombre** (Nombre del catalogo)

Devuelve la informacion de un catalogo por su nombre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función CatalogoCsv_model::getCampos() *[line 79]*

- **Access** public

void función CatalogoCsv_model::getId() *[line 64]*
****

- **Access** public

void función CatalogoCsv_model::getLLave() *[line 86]*

- **Access** public

string|*boolean* función CatalogoCsv_model::getMsgError([$value = 'usr'], $value,) *[line 113]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función CatalogoCsv_model::getNombre() *[line 72]*

- **Access** public

int función CatalogoCsv_model::getNumRows($nombre) *[line 145]*

Parámetros de la función:

- *string* **$nombre** Nombre del catalogo

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función CatalogoCsv_model::setCampos($value) *[line 82]*

Parámetros de la función:

- **$value**

- **Access** public

void función CatalogoCsv_model::setId($value) *[line 68]*

Parámetros de la función:

- **$value**

- **Access** public

void función CatalogoCsv_model::setLlave($value) *[line 89]*
Parámetros de la función:

- **$value**

- **Access** public

void función CatalogoCsv_model::setNombre($value) *[line 75]*
Parámetros de la función:

- **$value**

- **Access** public

void función CatalogoCsv_model::setOffset($value) *[line 93]*
Parámetros de la función:

- **$value**

- **Access** public

void función CatalogoCsv_model::setRows($value) *[line 96]*
Parámetros de la función:

- **$value**

- **Access** public

Clase Catalogo_model

[line 11]

Modelo Catalogo

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Catalogo_model::__construct() *[line 128]*

- **Access** public

boolean función Catalogo_model::checkPk($campo) *[line 254]*

Parámetros de la función:

- *string* **$campo** (varios campos delimitados por |)

Revisa en la base de datos por registros duplicados en los campos pasados por parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_model::checkTypeData($campo, $type) *[line 287]*

Parámetros de la función:

- *string* **$campo** (varios campos delimitados por |)
- **$type**

Revisa en la base de datos por registros que no coincidan con el tipo de dato pasado como parametro en el campo indicado

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_model::delete() *[line 386]*

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_model::getAll() *[line 144]*

Devuelve una lista con los catalogos existentes en la DB

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_model::getAllData($nombrecat) *[line 186]*

Parámetros de la función:

- *string* **$nombrecat** Nombre del catalogo

Devuelve los datos de un catalogo pasado como parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Catalogo_model::getByName($nombre) *[line 213]*

Parámetros de la función:

- *string* **$nombre** (Nombre del catalogo)

Devuelve la informacion de un catalogo por su nombre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_model::getCampos() *[line 79]*

- **Access** public

void función Catalogo_model::getId() *[line 64]*

****

- **Access** public

void función Catalogo_model::getLLave() *[line 86]*

- **Access** public

string|*boolean* función Catalogo_model::getMsgError([$value = 'usr'], $value,) *[line 113]*

Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Catalogo_model::getNombre() *[line 72]*

- **Access** public

int función Catalogo_model::getNumRows($nombre) *[line 165]*

Parámetros de la función:

- *string* **$nombre** Nombre del catalogo

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_model::insert($create, $select) *[line 329]*

Parámetros de la función:

- *string* **$create** (la consulta para crear el catalogo)
- *string* **$select** (la consulta para extraer datos de la tabla tmp_catalogo)

Inserta en la base datos el catálogo y obtiene los datos de la tabla temporal

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_model::setCampos($value) *[line 82]*

Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_model::setId($value) *[line 68]*

Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_model::setLlave($value) *[line 89]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_model::setNombre($value) *[line 75]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_model::setOffset($value) *[line 94]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_model::setRows($value) *[line 97]*
Parámetros de la función:

- **$value**

- **Access** public

boolean función Catalogo_model::updateComentario($nombre, $comentario) *[line 364]*
Parámetros de la función:

- *string* **$nombre**
- *string* **$comentario**

Cambia el comentario de la tabla indicada

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Catalogo_x_raiz_model

[line 11]

Modelo Raiz_x_Catalogo

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Catalogo_x_raiz_model::__construct() *[line 161]*

- **Access** public

Object función Catalogo_x_raiz_model::check($id) *[line 290]*
Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Revisa inconsistencias en los datos de un catalogo x raiz con respecto a su catalogo padre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Catalogo_x_raiz_model::delete() *[line 407]*
Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_x_raiz_model::getByArbol($id) *[line 178]*
Parámetros de la función:

- *int* **$id**

Devuelve todos los registros de la tabla raiz_x_catalogo de una raiz determinada

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Catalogo_x_raiz_model::getById($id) *[line 245]*
Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de un catalogo x accion por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Catalogo_x_raiz_model::getByNivel($idarbol, $nivel) *[line 223]*
Parámetros de la función:

- *int* **$idarbol**
- *int* **$nivel**

Devuelve el catalogo padre de un elemento raiz_x_catalogo

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::getColumnaDescripcion() *[line 113]*

- **Access** public

void función Catalogo_x_raiz_model::getColumnaLLave() *[line 106]*

- **Access** public

void función Catalogo_x_raiz_model::getGrado() *[line 92]*

- **Access** public

void función Catalogo_x_raiz_model::getId() *[line 76]*
****

- **Access** public

void función Catalogo_x_raiz_model::getIdRaiz() *[line 84]*

- **Access** public

string|boolean función Catalogo_x_raiz_model::getMsgError([$value = 'usr'], $value,) *[line 146]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

ArrayObject función Catalogo_x_raiz_model::getNivel($id) *[line 200]*
Parámetros de la función:

- *int* **$id**

Devuelve el nivel siguiente para la tabla raiz_x_catalogo de un arbol determinado

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::getRelacionHijo() *[line 127]*

- **Access** public

void función Catalogo_x_raiz_model::getRelacionPadre() *[line 120]*

- **Access** public

Object función Catalogo_x_raiz_model::getRelations($id) *[line 267]*
Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve las relaciones de una raiz x catalogo

- **Throws** Exception En caso de algun error al consultar la base de datos

- **Access** public

void función Catalogo_x_raiz_model::getTablaCatalogo() *[line 99]*

- **Access** public

int función Catalogo_x_raiz_model::insert() *[line 350]*

Inserta en la base datos la información del objeto actual

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Catalogo_x_raiz_model::setColumnaDescripcion($value) *[line 116]*

Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setColumnaLlave($value) *[line 109]*

Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setGrado($value) *[line 95]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setId($value) *[line 80]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setIdRaiz($value) *[line 88]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setRelacionHijo($value) *[line 130]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setRelacionPadre($value) *[line 123]*
Parámetros de la función:

- **$value**

- **Access** public

void función Catalogo_x_raiz_model::setTablaCatalogo($value) *[line 102]*
Parámetros de la función:

- **$value**

- **Access** public

Clase Cie10_model
[line 11]

Modelo Cie10

- **Package** SIIGS

- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Cie10_model::__construct() *[line 106]*

- **Access** public

boolean función Cie10_model::activaEnCatalogo($id, $catalogo, $valor) *[line 293]*

Parámetros de la función:

- *int* **$id** el id del registro en el catalogo
- *string* **$catalogo** nombre del catalogo donde se realizara la operacion
- *boolean* **$valor** agregar o eliminar el registro del catalogo

Accion para activar o desactivar registros de catalogos como el de EDA, IRA y Consultas

- **Throws** Exception Si ocurre algun error al consultar y modificar la base de datos
- **Access** public

boolean función Cie10_model::agregaEnCatalogo($id, $catalogo, $valor) *[line 243]*

Parámetros de la función:

- *int* **$id** el id del registro en el catalogo cie10
- *string* **$catalogo** nombre del catalogo donde se realizara la operacion
- *boolean* **$valor** agregar o eliminar el registro del catalogo

Accion para agregar registros del CIE10 a otros catalogos como el de EDA, IRA y Consultas

- **Throws** Exception Si ocurre algun error al consultar y modificar la base de datos

- **Access** public

boolean función Cie10_model::checkPk($campo) *[line 343]*
Parámetros de la función:

- *string* **$campo** (varios campos delimitados por |)

Revisa en la base de datos por registros duplicados en los campos pasados por parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getAll() *[line 143]*
Devuelve una lista con los registros existentes en el catalogo cie10

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getAllData($nombrecat) *[line 320]*
Parámetros de la función:

- *string* **$nombrecat** Nombre del catalogo

Devuelve los datos de un catalogo pasado como parametro

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Cie10_model::getById($id) *[line 220]*
Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de un registro del catalogo cie10 por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getCatalogoByName($cat) *[line 194]*
Parámetros de la función:

- **$cat**

Devuelve una lista con los registros existentes en el catalogo requerido

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Cie10_model::getData() *[line 170]*
Devuelve una lista con los registros existentes en el catalogo cie10 omitiendo los ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Cie10_model::getDescripcion() *[line 69]*

- **Access** public

void función Cie10_model::getId() *[line 61]*
****

- **Access** public

string|*boolean* función Cie10_model::getMsgError([$value = 'usr'], $value,) *[line 91]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

int función Cie10_model::getNumRows() *[line 122]*
Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Cie10_model::setDescripcion($value) *[line 73]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Cie10_model::setId($value) *[line 65]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Cie10_model::setOffset($value) *[line 51]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Cie10_model::setRows($value) *[line 54]*
Parámetros de la función:

- **$value**

- **Access** public

boolean función Cie10_model::update() *[line 374]*

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase ControladorAccion_model

[line 11]

Modelo ControladorAccion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función ControladorAccion_model::__construct() *[line 79]*

- **Access** public

Object función ControladorAccion_model::getById($id) *[line 119]*

Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de una accion por controlador de acuerdo a su Id

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función ControladorAccion_model::getId($controlador, $accion) *[line 97]*

Parámetros de la función:

- *int* **$controlador** (Id del controlador)
- *int* **$accion** (Id de la accion)

Devuelve el Id de una accion por controlador de una accion y controlador determinados

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función ControladorAccion_model::getIdByPath($path) *[line 143]*

Parámetros de la función:

- *string* **$path**

Devuelve el id de una accion por controlador de acuerdo al path

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

string|*boolean* función ControladorAccion_model::getMsgError([$value = 'usr'], $value,) *[line 64]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

int función ControladorAccion_model::setHelp($id, $textAyuda) *[line 181]*
Parámetros de la función:

- *int* **$id**
- *string* **$textAyuda**

Establece el mensaje de ayuda

- **Throws** Exception En caso de algun error al guardar el texto de ayuda
- **Access** public

Clase Controlador_model

[line 11]

Modelo Controlador

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Controlador_model::__construct() *[line 159]*

- **Access** public

boolean función Controlador_model::accionesUpdate() *[line 374]*

Actualiza las acciones asignadas a un controlador

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

boolean función Controlador_model::delete() *[line 431]*

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Controlador_model::getAccion() *[line 108]*

- **Access** public

ArrayObject función Controlador_model::getAcciones($id) *[line 271]*

Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve todas las acciones asignadas al controlador por su Id

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Controlador_model::getAll() *[line 175]*

Devuelve todos los registros de la tabla controlador

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Controlador_model::getByEntorno($id) *[line 244]*

Parámetros de la función:

- *int* **$id** ID (Id del entorno)

Devuelve todos los controladores que pertenecen a un entorno por su Id

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Controlador_model::getById($id) *[line 222]*

Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de un controlador por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Controlador_model::getClase() *[line 116]*

- **Access** public

void función Controlador_model::getDescripcion() *[line 92]*

- **Access** public

void función Controlador_model::getId() *[line 77]*
****

- **Access** public

void función Controlador_model::getIdEntorno() *[line 100]*

- **Access** public

string|*boolean* función Controlador_model::getMsgError([$value = 'usr'], $value,) *[line 144]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Controlador_model::getNombre() *[line 85]*

- **Access** public

int función Controlador_model::getNumRows([$entorno = 0]) *[line 201]*

Parámetros de la función:

- *int* **$entorno** , default 0 (Id del entorno)

Devuelve el numero de registros

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Controlador_model::getPermisos($entorno, $grupo) *[line 296]*

Parámetros de la función:

- *int* **$entorno** (Id del entorno)
- *int* **$grupo** (Id del grupo)

Devuelve los permisos asignados a un grupo sobre un entorno determinado (Mapea la información de las acciones asignadas a un controlador y los une con los permisos de un grupo sobre esas acciones)

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función Controlador_model::insert() *[line 317]*

Inserta en la tabla controlador, la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Controlador_model::setAccion($value) *[line 112]*

Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setClase($value) *[line 120]*

Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setDescripcion($value) *[line 96]*
Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setId($value) *[line 81]*
Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setIdEntorno($value) *[line 104]*
Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setNombre($value) *[line 88]*
Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setOffset($value) *[line 124]*
Parámetros de la función:

- **$value**

- **Access** public

void función Controlador_model::setRows($value) *[line 127]*
Parámetros de la función:

- **$value**

- **Access** public

boolean función Controlador_model::update() *[line 345]*
Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Entorno_model

[line 11]

Modelo Entorno

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Entorno_model::__construct() *[line 148]*

- **Access** public

boolean función Entorno_model::delete() *[line 311]*

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Entorno_model::getAll() *[line 164]*

Devuelve todos los registros de la tabla entorno

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Entorno_model::getById($id) *[line 186]*

Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de un entorno por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Entorno_model::getByName($nombre) *[line 208]*

Parámetros de la función:

- *string* **$nombre**

Devuelve la información de un entorno por su nombre

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Entorno_model::getDescripcion() *[line 78]*

- **Access** public

void función Entorno_model::getDirectorio() *[line 99]*

- **Access** public

void función Entorno_model::getHostname() *[line 92]*

- **Access** public

void función Entorno_model::getId() *[line 64]*
****

- **Access** public

string función Entorno_model::getInfo() *[line 142]*
Devuelve la información del objeto en forma de string

- **Access** public

void función Entorno_model::getIp() *[line 85]*

- **Access** public

string|boolean función Entorno_model::getMsgError([$value = 'usr'], $value,) *[line 118]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función Entorno_model::getNombre() *[line 71]*

- **Access** public

Object función Entorno_model::getPermissionsByGroup($grupo) *[line 230]*

Parámetros de la función:

- *string* **$grupo**

Obtiene los permisos asignados al grupo

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

int función Entorno_model::insert() *[line 252]*

Inserta en la tabla entorno la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Entorno_model::setDescripcion($value) *[line 81]*

Parámetros de la función:

- **$value**

- **Access** public

void función Entorno_model::setDirectorio($value) *[line 102]*
Parámetros de la función:

- **$value**

- **Access** public

void función Entorno_model::setHostname($value) *[line 95]*
Parámetros de la función:

- **$value**

- **Access** public

void función Entorno_model::setId($value) *[line 67]*
Parámetros de la función:

- **$value**

- **Access** public

void función Entorno_model::setIp($value) *[line 88]*

Parámetros de la función:

- **$value**

    - **Access** public

void función Entorno_model::setNombre($value) *[line 74]*
Parámetros de la función:

- **$value**

    - **Access** public

boolean función Entorno_model::update() *[line 281]*
Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Errorlog_model
[line 11]

Modelo Errorlog

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Errorlog_model::__construct() *[line 76]*

- **Access** public

void|*boolean* función Errorlog_model::addFilter($columna, $condicion, $valor) *[line 243]*

Parámetros de la función:

- *string* **$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, descripcion, id_controlador_accion)
- *string* **$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access** public

array función Errorlog_model::getAll([$offset = null], [$row_count = null]) *[line 301]*

Parámetros de la función:

- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla Error en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

object|boolean función Errorlog_model::getById($id) *[line 196]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro del Error que tiene el ID especificado

- **Access** public

void función Errorlog_model::getDescripcion() *[line 129]*

- **Access** public

void función Errorlog_model::getFecha_hora() *[line 124]*

- **Access** public

void función Errorlog_model::getId() *[line 94]*

- **Access** public

void función Errorlog_model::getId_controlador_accion() *[line 114]*

- **Access** public

void función Errorlog_model::getId_usuario() *[line 104]*

- **Access** public

int|*boolean* función Errorlog_model::getNumRows() *[line 344]*

Obtiene el numero total de registros en la tabla Error en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Errorlog_model::insert($path, $descripcion, $id_controlador, $id_accion) *[line 158]*

Parámetros de la función:

- *int* **$id_controlador**
- *int* **$id_accion**
- *string* **$descripcion**
- **$path**

Inserta en la base de datos la informacion del error

- **Static**
- **Access** public

void función Errorlog_model::resetFilter() *[line 285]*

Elimina todos los filtros registrados

- **Access** public

string función Errorlog_model::save($model, $method, $modelo) *[line 383]*
Parámetros de la función:

- *string* **$modelo** Nombre del modelo que lanzó la excepción
- *string* **$method** Contiene el nombre de la clase y metodo donde se originó el error o el mensaje de error para mostrar al usuario final
- **$model**

Guardar el mensaje de error descriptivo en la base de datos,
si no puede insertar el registro a la base de datos, el mensaje de error se guarda en el directorio logs, devuelve el mensaje de error para el usuario final

- **Static**
- **Access** public

void función Errorlog_model::setDescripcion($descripcion) *[line 134]*
Parámetros de la función:

- **$descripcion**

- **Access** public

void función Errorlog_model::setId($id) *[line 99]*
Parámetros de la función:

- **$id**

- **Access** public

void función Errorlog_model::setId_accion($id_accion) *[line 144]*
Parámetros de la función:

- **$id_accion**

- **Access** public

void función Errorlog_model::setId_controlador($id_controlador) *[line 139]*
Parámetros de la función:

- **$id_controlador**

- **Access** public

void función Errorlog_model::setId_controlador_accion($id_controlador_accion) *[line 119]*
Parámetros de la función:

- **$id_controlador_accion**

- **Access** public

void función Errorlog_model::setId_usuario($id_usuario) *[line 109]*

Parámetros de la función:

- **$id_usuario**

- **Access** public

Clase Georeferencia_model

[line 11]

Modelo Georeferencia

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Georeferencia_model::__construct() *[line 36]*

- **Access** public

boolean|*string* función Georeferencia_model::getMsgError([$type = 'usr']) *[line 60]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Georeferencia_model::process() *[line 82]*

Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia

- **Access** public

Clase Grupo_model

[line 10]

Modelo Grupo

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Grupo_model::__construct() *[line 42]*

- **Access** public

boolean función Grupo_model::delete() *[line 265]*

Elimina de la base de datos al grupo (id en propiedades)

- **Access** public

void|*array* función Grupo_model::getAll([$keywords = ''], [$offset = null], [$row_count = null]) *[line 102]*
Parámetros de la función:

- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver
- **$keywords**

Obtiene todos los grupos existentes

- **Access** public

void|*object* función Grupo_model::getById($id) *[line 135]*
Parámetros de la función:

- *int* **$id** id del grupo

Obtiene el grupo solicitado

- **Access** public

void|*object* función Grupo_model::getByName($name) *[line 177]*
Parámetros de la función:

- *string* **$name** nombre del grupo

Obtiene el grupo solicitado

- **Access** public

void función Grupo_model::getDescripcion() *[line 68]*

- **Access** public

void|object función Grupo_model::getEntornosById($id) *[line 155]*

Parámetros de la función:

- *int* **$id** id del grupo

Obtiene el grupo solicitado con sus entornos vinculados

- **Access** public

void función Grupo_model::getId() *[line 49]*

- **Access** public

boolean función Grupo_model::getMsgError([$value = 'usr']) *[line 85]*

Parámetros de la función:

- *string* **$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void función Grupo_model::getNombre() *[line 58]*

- **Access** public

int función Grupo_model::getNumRows([$keywords = '']) *[line 197]*

Parámetros de la función:

- *boolean*|*string* **$keywords** false no hay texto a buscar|string con texto a buscar

Obtiene el numero total de grupos

- **Access** public

boolean función Grupo_model::insert() *[line 223]*

Inserta en la base de datos los datos del grupo (datos en propiedades)

- **Access** public

void función Grupo_model::setDescripcion($descripcion) *[line 73]*

Parámetros de la función:

- **$descripcion**

  - **Access** public

void función Grupo_model::setId($value) *[line 54]*
Parámetros de la función:

- **$value**

  - **Access** public

void función Grupo_model::setNombre($nombre) *[line 63]*
Parámetros de la función:

- **$nombre**

  - **Access** public

boolean función Grupo_model::update() *[line 244]*
Actualiza en la base de datos los datos del grupo (datos en propiedades)

  - **Access** public

Clase Hemoglobina_model

[line 11]

Modelo Hemoglobina

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Hemoglobina_model::__construct() *[line 36]*

- **Access** public

boolean|*string* función Hemoglobina_model::getMsgError([$type = 'usr']) *[line 60]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Hemoglobina_model::process() *[line 82]*

Inserta los registros contenidos en la tabla cat_georeferencia a la tabla asu_georeferencia

- **Access** public

Clase Menu_model

[line 11]

Modelo Menu

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Menu_model::__construct() *[line 85]*

- **Access** public

void|boolean función Menu_model::addFilter($columna, $condicion, $valor) *[line 401]*

Parámetros de la función:

- *string* **$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, parametros, id_controlador_accion)
- *string* **$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access** public

int función Menu_model::delete([$id = null]) *[line 287]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

int función Menu_model::deleteByFilter() *[line 326]*

Elimina el conjunto de registros que cumplen con el o los criterios de filtrado

- **Access** public

array función Menu_model::getAll([$offset = null], [$row_count = null]) *[line 469]*

Parámetros de la función:

- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla Menu en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Menu_model::getAtributo() *[line 163]*

- **Access** public

object|boolean función Menu_model::getById($id) *[line 358]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro de la menu que tiene el ID especificado

- **Access** public

boolean función Menu_model::getByPadre($padre) *[line 566]*

Parámetros de la función:

- **$padre**

Obtiene todos los nodos hijos de un padre

- **Access** public

void función Menu_model::getId() *[line 98]*

- **Access** public

void función Menu_model::getId_controlador() *[line 143]*

- **Access** public

void función Menu_model::getId_padre() *[line 108]*

- **Access** public

void función Menu_model::getId_raiz() *[line 118]*

- **Access** public

boolean|*string* función Menu_model::getMsgError([$type = 'usr']) *[line 179]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

void función Menu_model::getNombre() *[line 128]*

- **Access** public

int función Menu_model::getNumRows() *[line 511]*

Obtiene el numero total de registros en la tabla Menu en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Menu_model::getRuta() *[line 153]*

- **Access** public

boolean función Menu_model::hasChild($id) *[line 550]*

Parámetros de la función:

- **$id**

Verifica si el nodo actual tiene hijos

- **Access** public

boolean función Menu_model::insert() *[line 200]*

Inserta en la base de datos, la informacion contenida en el objeto

- **Access** public

void función Menu_model::resetFilter() *[line 453]*

Elimina todos los filtros registrados

- **Access** public

void función Menu_model::setAtributo($atributo) *[line 158]*

Parámetros de la función:

- **$atributo**

- **Access** public

void función Menu_model::setId($id) *[line 103]*

Parámetros de la función:

- **$id**

- **Access** public

void función Menu_model::setId_controlador($id_controlador) *[line 138]*

Parámetros de la función:

- **$id_controlador**

- **Access** public

void función Menu_model::setId_padre($id_padre) *[line 113]*
Parámetros de la función:

- **$id_padre**

- **Access** public

void función Menu_model::setId_raiz($id_raiz) *[line 123]*
Parámetros de la función:

- **$id_raiz**

- **Access** public

void función Menu_model::setNombre($nombre) *[line 133]*
Parámetros de la función:

- **$nombre**

- **Access** public

void función Menu_model::setRuta($ruta) *[line 148]*
Parámetros de la función:

- **$ruta**

- **Access** public

boolean función Menu_model::update([$id = null]) *[line 244]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase Permiso_model

[line 10]

Modelo Permiso

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Permiso_model::__construct() *[line 47]*

- **Access** public

boolean función Permiso_model::deletePermissions($entorno, $grupo) *[line 156]*

Parámetros de la función:

- *int* **$entorno** id de entorno
- *int* **$grupo** id del grupo

Elimina de la base de datos los permisos del entorno y grupo recibidos

- **Access** public

void función Permiso_model::getFecha() *[line 73]*

- **Access** public

void función Permiso_model::getId() *[line 54]*

- **Access** public

void función Permiso_model::getIdControladorAccion() *[line 83]*

- **Access** public

void función Permiso_model::getIdGrupo() *[line 63]*

- **Access** public

boolean función Permiso_model::getMsgError([$value = 'usr']) *[line 100]*
Parámetros de la función:

- *$value* **$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void|object función Permiso_model::getPermission($id) *[line 114]*
Parámetros de la función:

- *int* **$id** id del controlador_x_accion

Obtiene el permiso solicitado

- **Access** public

boolean función Permiso_model::insertBatch($data) *[line 135]*
Parámetros de la función:

- *array* **$data** object $data array con los permisos a insertar

Inserta en la base de datos el arreglo de permisos recibido

- **Access** public

void función Permiso_model::setFecha($fecha) *[line 78]*
Parámetros de la función:

- **$fecha**

- **Access** public

void función Permiso_model::setId($value) *[line 59]*
Parámetros de la función:

- **$value**

- **Access** public

void función Permiso_model::setIdControladorAccion($id_controlador_accion) *[line 88]*
Parámetros de la función:

- **$id_controlador_accion**

- **Access** public

void función Permiso_model::setIdGrupo($id_grupo) *[line 68]*

Parámetros de la función:

- **$id_grupo**

- **Access** public

Clase Poblacion_model

[line 11]

Modelo Poblacion

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Poblacion_model::__construct() *[line 36]*

- **Access** public

boolean|*string* función Poblacion_model::getMsgError([$type = 'usr']) *[line 60]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

boolean función Poblacion_model::process() *[line 82]*

Inserta los registros contenidos en la tabla cat_poblacion a la tabla asu_poblacion

- **Access** public

Clase Raiz_model

[line 11]

Modelo Raiz

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función Raiz_model::__construct() *[line 84]*

- **Access** public

boolean función Raiz_model::delete() *[line 216]*

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Boolean función Raiz_model::ExistInArbol($id) *[line 122]*

Parámetros de la función:

- *int* **$id**

Revisa si la raiz pasada como parametro existe en el ASU

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

ArrayObject función Raiz_model::getAll() *[line 100]*

Devuelve todos los registros de la tabla raiz

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función Raiz_model::getById($id) *[line 144]*

Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de una raiz por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Raiz_model::getDescripcion() *[line 48]*

- **Access** public

void función Raiz_model::getId() *[line 40]*
****

- **Access** public

string|*boolean* función Raiz_model::getMsgError([$value = 'usr'], $value,) *[line 69]*
Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

int función Raiz_model::insert() *[line 165]*
Inserta en la tabla raiz, la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función Raiz_model::setDescripcion($value) *[line 52]*

Parámetros de la función:

- **$value**

- **Access** public

void función Raiz_model::setId($value) *[line 44]*

Parámetros de la función:

- **$value**

- **Access** public

boolean función Raiz_model::update() *[line 190]*

Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase ReglaVacuna_model

[line 11]

Modelo ReglaVacuna

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Geovanni

Constructor *void* función ReglaVacuna_model::__construct() *[line 251]*

- **Access** public

boolean función ReglaVacuna_model::delete() *[line 414]*

Elimina el registro actual de la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ReglaVacuna_model::getAlergias() *[line 209]*

- **Access** public

ArrayObject función ReglaVacuna_model::getAll() *[line 267]*

Devuelve todos los registros de la tabla regla_vacuna

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Object función ReglaVacuna_model::getById($id) *[line 288]*
Parámetros de la función:

- *int* **$id** ID (Llave primaria)

Devuelve la información de una regla de vacuna por su ID

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ReglaVacuna_model::getDiaFinNacido() *[line 146]*

- **Access** public

void función ReglaVacuna_model::getDiaFinPrevia() *[line 160]*

- **Access** public

void función ReglaVacuna_model::getDiaInicioNacido() *[line 139]*

- **Access** public

void función ReglaVacuna_model::getDiaInicioPrevia() *[line 153]*

- **Access** public

void función ReglaVacuna_model::getDosis() *[line 174]*

- **Access** public

void función ReglaVacuna_model::getEsqComp() *[line 195]*

- **Access** public

void función ReglaVacuna_model::getForzarAplicacion() *[line 216]*

- **Access** public

void función ReglaVacuna_model::getId() *[line 118]*
****

- **Access** public

void función ReglaVacuna_model::getIdVacuna() *[line 125]*

- **Access** public

void función ReglaVacuna_model::getIdVacunaPrevia() *[line 132]*

- **Access** public

void función ReglaVacuna_model::getIdViaVacuna() *[line 167]*

- **Access** public

string|boolean función ReglaVacuna_model::getMsgError([$value = 'usr'], $value,) *[line 236]*

Parámetros de la función:

- *string* **$value,** default 'usr' (Tipo mensaje)
- **$value**

Devuelve los mensajes de error en caso de ocurrir alguna excepción 'usr' devuelve el mensaje para la vista de usuario 'log' devuelve el mensaje para el log de errores

- **Access** public

void función ReglaVacuna_model::getObservacionRegion() *[line 188]*

- **Access** public

void función ReglaVacuna_model::getOrdenEsqComp() *[line 202]*

- **Access** public

void función ReglaVacuna_model::getRegion() *[line 181]*

- **Access** public

int función ReglaVacuna_model::insert() *[line 327]*

Inserta en la tabla regla_vacuna la información contenida en el objeto

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

void función ReglaVacuna_model::setAlergias($value) *[line 212]*

Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setDiaFinNacido($value) *[line 149]*

Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setDiaFinPrevia($value) *[line 163]*

Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setDiaInicioNacido($value) *[line 142]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setDiaInicioPrevia($value) *[line 156]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setDosis($value) *[line 177]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setEsqComp($value) *[line 198]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setForzarAplicacion($value) *[line 219]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setId($value) *[line 121]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setIdVacuna($value) *[line 128]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setIdVacunaPrevia($value) *[line 135]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setIdViaVacuna($value) *[line 170]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setObservacionRegion($value) *[line 191]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setOrdenEsqComp($value) *[line 205]*
Parámetros de la función:

- **$value**

- **Access** public

void función ReglaVacuna_model::setRegion($value) *[line 184]*
Parámetros de la función:

- **$value**

- **Access** public

boolean función ReglaVacuna_model::update() *[line 370]*
Actualiza el objeto actual en la base de datos

- **Throws** Exception En caso de algun error al consultar la base de datos
- **Access** public

Clase Usuario_model
[line 10]

Modelo Usuario

- **Package** SIIGS
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Usuario_model::__construct() *[line 81]*

- **Access** public

null|*object* función Usuario_model::authenticate($username, $password) *[line 423]*

Parámetros de la función:

- *string* **$username** nombre de usuario
- *string* **$password** clave

Valida las credenciales recibidas

- **Access** public

void función Usuario_model::checkCredentials($path, $pathURL, $group_id) *[line 444]*

Parámetros de la función:

- *string* **$path** entorno::controlador::accion
- *int* **$group_id** id del grupo a validar permisos
- **$pathURL**

Verifica que el usuario haya iniciado sesión y además tenga permiso en la acción recibida

- **Static**
- **Access** public

void función Usuario_model::check_data($usuario, $correo) *[line 477]*

Parámetros de la función:

- **$usuario**
- **$correo**

- **Access** public

void función Usuario_model::check_token($correo) *[line 490]*

Parámetros de la función:

- **$correo**

- **Access** public

boolean función Usuario_model::delete() *[line 404]*

Elimina de la base de datos al usuario (id en propiedades)

- **Access** public

void|*object* función Usuario_model::getActivesByGroup($group_id, $id, $viewMode) *[line 283]*

Parámetros de la función:

- *int* **$id** id del usuario
- *boolean* **$viewMode** true obtiene el modo visualización, false o null obtiene el registro normal
- **$group_id**

Obtiene los usuarios activos del grupo solicitado

- **Access** public

void función Usuario_model::getActivo() *[line 161]*

- **Access** public

void función Usuario_model::getApellidoMaterno() *[line 141]*

- **Access** public

void función Usuario_model::getApellidoPaterno() *[line 131]*

- **Access** public

void|*object* función Usuario_model::getById($id, [$viewMode = FALSE]) *[line 253]*

Parámetros de la función:

- *int* **$id** id del usuario
- *boolean* **$viewMode** true obtiene el modo visualización, false o null obtiene el registro normal

Obtiene el usuario solicitado, se puede obtener el registro normal o personalizado para visualización (descripciones en tablas vinculadas)

- **Access** public

void|*object* función Usuario_model::getByUsername($username) *[line 303]*
Parámetros de la función:

- *string* **$username** nombre de usuario

Obtiene el usuario solicitado

- **Access** public

void función Usuario_model::getClave() *[line 111]*

- **Access** public

void función Usuario_model::getCorreo() *[line 151]*

- **Access** public

void función Usuario_model::getgrupo($grupo) *[line 532]*
Parámetros de la función:

- **$grupo**

- **Access** public

void función Usuario_model::getId() *[line 92]*

- **Access** public

void función Usuario_model::getIdGrupo() *[line 171]*

- **Access** public

boolean función Usuario_model::getMsgError([$value = 'usr']) *[line 188]*
Parámetros de la función:

- *string* **$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void función Usuario_model::getNombre() *[line 121]*

- **Access** public

void función Usuario_model::getNombreUsuario() *[line 101]*

- **Access** public

int función Usuario_model::getNumRows([$keywords = '']) *[line 323]*
Parámetros de la función:

- *boolean|string* **$keywords** false no hay texto a buscar|string con texto a buscar

Obtiene el numero total de usuarios

- **Access** public

void|*array* función Usuario_model::getOnlyActives([$keywords = ''], [$onlyActives = TRUE], [$offset = null], [$row_count = null]) *[line 207]*

Parámetros de la función:

- *boolean*|*string* **$keywords** false no hay texto a buscar|string con texto a buscar
- *boolean* **$onlyActives** true obtiene solo usuarios activos, false o null obtiene todos los usuarios
- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver

Obtiene todos los usuarios existentes, se puede filtrar por: texto a buscar o solo activos si se desea

- **Access** public

void función Usuario_model::getuser($id) *[line 519]*

Parámetros de la función:

- **$id**

- **Access** public

void función Usuario_model::get_grupo_entorno($nombre) *[line 583]*

Parámetros de la función:

- **$nombre**

- **Access** public

void función Usuario_model::get_permiso_entorno($nombre) *[line 560]*

Parámetros de la función:

- **$nombre**

- **Access** public

void función Usuario_model::get_usuario_entorno($nombre, [$inusuario = ""]) *[line 607]*

Parámetros de la función:

- **$nombre**
- **$inusuario**

- **Access** public

boolean función Usuario_model::insert() *[line 351]*

Inserta en la base de datos los datos del usuario (datos en propiedades)

- **Access** public

void función Usuario_model::setActivo($activo) *[line 166]*
Parámetros de la función:

- **$activo**

- **Access** public

void función Usuario_model::setApellidoMaterno($apellido_materno) *[line 146]*
Parámetros de la función:

- **$apellido_materno**

- **Access** public

void función Usuario_model::setApellidoPaterno($apellido_paterno) *[line 136]*
Parámetros de la función:

- **$apellido_paterno**

- **Access** public

void función Usuario_model::setClave($clave) *[line 116]*
Parámetros de la función:

- **$clave**

    - **Access** public

void función Usuario_model::setCorreo($correo) *[line 156]*
Parámetros de la función:

- **$correo**

    - **Access** public

void función Usuario_model::setId($value) *[line 97]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Usuario_model::setIdGrupo($id_grupo) *[line 176]*
Parámetros de la función:

- **$id_grupo**

- **Access** public

void función Usuario_model::setNombre($nombre) *[line 126]*
Parámetros de la función:

- **$nombre**

- **Access** public

void función Usuario_model::setNombreUsuario($nombre_usuario) *[line 106]*
Parámetros de la función:

- **$nombre_usuario**

- **Access** public

boolean función Usuario_model::update() *[line 378]*
Actualiza en la base de datos los datos del usuario (datos en propiedades)

- **Access** public

void función Usuario_model::update_pass($pass, $id) *[line 503]*
Parámetros de la función:

- **$pass**
- **$id**

- **Access** public

void función Usuario_model::update_user($id, $clave, $correo) *[line 545]*

Parámetros de la función:

- **$id**
- **$clave**
- **$correo**

- **Access** public

Paquete TES Clases

Clase Enrolamiento

[line 10]

Controlador de enrolamiento

- **Package** TES
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Enrolamiento::__construct() *[line 13]*

- **Access** public

echo función Enrolamiento::addForm() *[line 907]*

Pase de parametros para la insercion o actualizacion

- **Access** public

echo función Enrolamiento::autocomplete() *[line 470]*

Crea el autocomplete para facilitar la busqueda de un tutor

- **Access** public

echo función Enrolamiento::brothers_search($tutor) *[line 1088]*
Parámetros de la función:

- *string* **$tutor** id del tutor compartido

Este metodo extrae la informacion de las personas con las que se comparte el mismo tutor si se selecciona una de estas importa los datos para el apartado direccion

- **Access** public

echo función Enrolamiento::brother_found($id) *[line 1071]*
Parámetros de la función:

- *string* **$id** id de la persona

Este metodo verifica si un paciente comparte un mismo tutor

- **Access** public

echo función Enrolamiento::catalog_check($catalog, $tipo, [$col = 1], [$sel = ""], [$orden = ""]) *[line 417]*
Parámetros de la función:

- *string* **$catalog** tabla de donde se extrae la informacion
- *string* **$tipo** tipo de control radio o check

- *int* **$col** numero de columnas en la tabla
- *string* **$sel** identifica si un valor ya esta seleccionado
- *string* **$orden** columna para hacer el ordenamiento

Crea un grupo de radio o check con la informacion de los catalogos

- **Access** public

echo función Enrolamiento::catalog_select($catalog, [$sel = ""], [$orden = ""], [$campo = ""], [$valor = ""]) *[line 338]*

Parámetros de la función:

- *string* **$catalog** tabla de donde se extrae la informacion
- *string* **$sel** identifica si un valor ya esta seleccionado
- *string* **$orden** columna para hacer el ordenamiento
- *string* **$campo** campo de la tabla para hacer el where
- *string* **$valor** valor a comparar en el where

Genera los options de un campo tipo select

- **Access** public

echo función Enrolamiento::categoriacie10_select() *[line 1333]*

Genera los options de un campo tipo select para las categorías de CIE10

- **Access** public

bool función Enrolamiento::checar_session() *[line 1310]*

Revisa si la sesión está activa

- **Access** public

echo función Enrolamiento::cie10_select($categoria) *[line 1360]*
Parámetros de la función:

- *string* **$categoria** Categoría de la CIE10

Genera los options de un campo tipo select para los CIE10 correspondientes a una categoría

- **Access** public

echo función Enrolamiento::comparar_view($id, [$prod1 = ""], [$prod2 = ""], [$prod3 = ""]) *[line 1241]*
Parámetros de la función:

- *string* **$id** identificador de la persona
- **$prod1**
- **$prod2**
- **$prod3**

Crea la pagina para ver la infromacion de la persona y comprararla con la persona capturada

- **Access** public

echo función Enrolamiento::data_tutor($curp) *[line 492]*
Parámetros de la función:

- *string* **$curp** curp del tutor

Obtiene inofrmacion del tutor

- **Access** public

echo función Enrolamiento::file_to_card($id, [$tipo = ""]) *[line 538]*

Parámetros de la función:

- *string* **$id** identificador de la persona
- **$tipo**

crea un archivo descargable el cual se necesita para el envio por nfc a la tarjeta del paciente

- **Access** public

echo función Enrolamiento::ifCurpExists($curp) *[line 985]*

Parámetros de la función:

- *string* **$curp** curp de la persona

Valida que la curp del paciente no exista

- **Access** public

echo función Enrolamiento::ifCurpTExists($curp) *[line 1029]*

Parámetros de la función:

- *string* **$curp** curp de la persona

valida que la curp del tutor no exista

- **Access** public

echo función Enrolamiento::index([$pag = 0], [$id = ""], [$array = ""]) *[line 38]*

Parámetros de la función:

- *string* **$pag** numero de pagina para la posicion
- *string* **$id** id de una persona
- **$array**

Este es el metodo por default, obtiene el listado de las personas

se recibe el parametro $pag de tipo int que representa la paginacion

- **Access** public

echo función Enrolamiento::insert() *[line 766]*

prepara los datos para insertarlos

- **Access** public

json($porcentaje_similitud,$persona) función Enrolamiento::paciente_similar($nombre, $paterno, $materno, $curp, $nacimiento, $lugar, [$calle = ""], [$referencia = ""], [$colonia = ""], [$curpT = ""], $cp, $numero) *[line 1190]*

Parámetros de la función:

- *string* **$nombre** nombre del paciente que se esta capturando
- *string* **$paterno** apellido paterno del paciente
- *string* **$materno** apellido materno del paciente
- *string* **$curp** curp del paciente
- *string* **$nacimiento** fecha de nacimiento del paciente
- *string* **$calle** calle del domicilio del paciente
- *string* **$referencia** referencia del domicilio del paciente
- *string* **$colonia** colonia del paciente
- *string* **$cp** cp de la colonia donde vive el paciente
- *string* **$numero** numero de la vivienda
- **$lugar**
- **$curpT**

Comprueba la similitud de un paciente que se este capturando con los que ya existe en la base de datos, esto con la finalidad de disminuir datos repetidos

- **Access** public

echo función Enrolamiento::print_card($id) *[line 87]*

Parámetros de la función:

- *string* **$id** identificador de la persona

Este metodo estrae la informacion del paciente que sera impreso en la tarjeta

- **Access** public

Object función Enrolamiento::searchageb([$idasulocalidad = ""], [$like = ""]) *[line 1158]*

Parámetros de la función:

- *int* **$idasulocalidad** Id de la localidad en el ASU
- **$like**

Regresa un objeto JSON con la lista de agebs disponibles en la localidad Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

Object función Enrolamiento::searchum($idasulocalidad, $ageb) *[line 1130]*

Parámetros de la función:

- *int* **$idasulocalidad** Id de la localidad en el ASU
- *string* **$ageb** Numero de ageb

Busca dentro del catalogo asu_ageb la unidad médica de acuerdo a la localidad y ageb Solo se permite su acceso por medio de peticiones AJAX

- **Access** public

echo función Enrolamiento::tratamiento_select([$campo = ""], [$valor = ""], [$sel = ""], [$orden = ""]) *[line 371]*

Parámetros de la función:

- *string* **$campo** campo de la tabla para hacer el where
- *string* **$valor** valor a comparar en el where
- *string* **$sel** identifica si un valor ya esta seleccionado
- *string* **$orden** columna para hacer el ordenamiento

Genera los options de un campo tipo select para los tratamientos de consultas

- **Access** public

echo función Enrolamiento::update($id) *[line 194]*
Parámetros de la función:

- *string* **$id** identificador de la persona

Crea el fromulario para editar la informacion de la persona

- **Access** public

echo función Enrolamiento::update_card($persona, $impreso, [$fecha = ""], [$archivo = ""], [$entorno = '4']) *[line 739]*
Parámetros de la función:

- *string* **$persona** id de la persona
- *boolean* **$impreso** identifica si el proceso de impresion fue correcto o no
- *int* **$fecha** fecha del evento
- *string* **$archivo** archivo generado
- *string* **$entorno** tipo de entorno

Este metodo actualiza el estado del archivo descargado si fue escrito correctamente o no en la tarjeta

- **Access** public

echo función Enrolamiento::vacunacion($fecha, [$id = ""]) *[line 1291]*
Parámetros de la función:

- *date* **$fecha**
- *string* **$id**

Obtiene el historial de vacunacion de un paciente cuyo id es pasado por parametro

- **Access** public

echo función Enrolamiento::validarForm([$op = ""]) *[line 833]*

Parámetros de la función:

- *string* **$op** bandera que identifica si una seccion entra en validacion

valida los datos de entrada en el formulario

- **Access** public

echo función Enrolamiento::validarisum($id) *[line 1107]*

Parámetros de la función:

- *string* **$id** id de arbol de segmentacion

valida que el nodo seleccionado en el arbol sea una unidad medica

- **Access** public

echo función Enrolamiento::validate_card($persona, $archivo) *[line 755]*
Parámetros de la función:

- *string* **$persona** id de la persona
- *string* **$archivo** archivo generado

valida que un archivo sea valido para enviar a la tarjeta por nfc

- **Access** public

echo función Enrolamiento::view($id) *[line 134]*
Parámetros de la función:

- *string* **$id** identificador de la persona

Crea la pagina para ver la infromacion de la persona

- **Access** public

Clase Notificacion

[line 10]

Controlador Notificación

- **Package** TES
- **Sub-Package** Controlador

- **Author** Rogelio

Constructor *void* función Notificacion::__construct() *[line 12]*

- **Access** public

void función Notificacion::delete($id) *[line 255]*
Parámetros de la función:

- *int* **$id** id de notificación a eliminar

Solicita la eliminación de la notificación recibida

- **Access** public

void función Notificacion::index([$pag = 0]) *[line 34]*
Parámetros de la función:

- *int* **$pag** número de página a visualizar (paginación)

1) Visualiza las notificaciones existentes para su interacción CRUD 2) En caso de detectar un texto a buscar se filtran las notificaciones existentes acorde a la búsqueda

- **Access** public

void función Notificacion::insert() *[line 139]*
1) Prepara el formulario para la inserción de una notificación nueva 2) Realiza las

validaciones necesarias sobre cada campo del registro

- **Access** public

void función Notificacion::update($id) *[line 197]*
Parámetros de la función:

- *int* **$id** id de la notificación a modificar

1) Prepara el formulario para la modificación de una notificación existente 2) Realiza las validaciones necesarias sobre cada campo del registro

- **Access** public

void función Notificacion::view($id) *[line 103]*
Parámetros de la función:

- *int* **$id** id de notificación a visualizar

Visualiza los datos de la notificación recibida

- **Access** public

Clase Reporteador

[line 10]

Controlador Reporteador

- **Package** TES
- **Sub-Package** Controlador
- **Author** Rogelio

Constructor *void* función Reporteador::__construct() *[line 12]*

- **Access** public

void función Reporteador::index() *[line 32]*

Visualiza los reportes existentes

- **Access** public

void función Reporteador::view($op, $title, $nivel, $id, [$fecha = '']) *[line 73]*

Parámetros de la función:

- *int* **$op**
- *string* **$title**
- *int* **$nivel**
- *int* **$id**
- *string* **$fecha**

Renderiza la vista del reporte

- **Access** public

Clase Reporte_sincronizacion

[line 10]

Controller Usuario

- **Package** TES
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Reporte_sincronizacion::__construct() *[line 13]*

- **Access** public

void función Reporte_sincronizacion::index() *[line 35]*

Este es el metodo por default, crea el listado del reporte de sincronizacion

- **Access** public

void función Reporte_sincronizacion::lote() *[line 175]*

Genera el item del reporte de lote de vacunacion

- **Access** public

void función Reporte_sincronizacion::lote_view($lote, $title, $op, [$lugar = "Chiapas"]) *[line 344]*

Parámetros de la función:

- *string* **$lote** Valor del lote del que se esta generando el reporte
- *string* **$title** Especifica el titulo a mostrar en el reporte
- *string* **$op** Especifica el reporte a mostrar
- *string* **$lugar** Especifica el lugar donde se centrara el mapa

Muestra detallada por cada list del reporte de lote de vacunacion

- **Access** public

void función Reporte_sincronizacion::view($op, $title) *[line 103]*

Parámetros de la función:

- *string* **$op** Especifica el reporte a mostrar
- *string* **$title** Especifica el titulo a mostrar en el reporte

Muestra detallada por cada list del reporte de sincronizacion

- **Access** public

Clase Semana_nacional

[line 11]

Controlador Semana Nacional

- **Package** TES
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Semana_nacional::__construct() *[line 13]*

- **Access** public

void función Semana_nacional::delete($id) *[line 245]*

Parámetros de la función:

- *int* **$id** ID del elemento a eliminar

Eliminar el registro especificado por el id

- **Access** public

json función Semana_nacional::getAll() *[line 278]*

Devuelve un json con todos los registros de semanas nacional

- **Access** public

void función Semana_nacional::index([$pag = 0]) *[line 35]*

Parámetros de la función:

- *int* **$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de semanas nacional, con su correspondiente paginación permite eliminar un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico

- **Access** public

void función Semana_nacional::insert() *[line 92]*

Muestra el formulario para crear un nuevo registro en la semana_nacional, las variables se obtienen por el metodo POST

- **Access** public

void función Semana_nacional::update($id) *[line 151]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos

- **Access** public

void función Semana_nacional::view($id) *[line 205]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

Clase Servicios

[line 10]

Controlador Servicios

- **Package** TES
- **Sub-Package** Controlador
- **Author** Eliecer

Constructor *void* función Servicios::__construct() *[line 11]*

- **Access** public

echo función Servicios::actualiza_estado_tableta($id_sesion, [$id_tes_estado_tableta = ""], [$version = ""], $id_session) *[line 1025]*

Parámetros de la función:

- *int* **$id_session** Representa la session activa para la peticion
- *json* **$id_tes_estado_tableta** Representa el status que tomara la tableta
- *int* **$version** Representa la version de la apk instalada en la tableta
- **$id_sesion**

Actualiza el estatus de la tableta

- **Access** public

echo función Servicios::catalogos_relevantes([$sf = ""]) *[line 1102]*
Parámetros de la función:

- *string* **$sf** bandera que activa el filtro de fechas segun el tipo de sincronizacion

Genera los catalogos relevantes por entorno

- **Access** public

echo función Servicios::esquema_incompleto($id_persona, $fecha, $vacunas) *[line 1053]*
Parámetros de la función:

- *int* **$id_persona** Representa la persona a la que se le calculara su esquema
- *string* **$fecha** Representa la fecha de nacimiento de la persona
- *array* **$vacunas** Representa las vacunas aplicadas a la persona

Genera los esquemas incompletos de las personas que correspondan a la unidad medica de la tableta

- **Access** public

void función Servicios::is_step_0($id_accion, [$id_tab = null], [$id_sesion = null], [$id_version = null], [$datos = null], $id_session, $version_apk) *[line 79]*
Parámetros de la función:

- *int* **$id_accion** Representa el tipo de accion que se ejecutara
- *int* **$id_tab** Representa a la MAC de la tableta

- *int* **$id_session** Representa la session activa para la peticion
- *int* **$version_apk** Representa la version de la apk instalada en la tableta
- *json* **$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor
- **$id_sesion**
- **$id_version**

Paso 0 se procesa las peticiones segun la accion:

Si la acci?n es 1: Valida la disponibilidad del dispositivo especificado y genera una session que se mantiene activa en toda la sincronizacion Si la acci?n es 2: Regresa la informacion de todos los catalogos Si la acci?n es 3: Recibe un mensaje si es ok actualiza el estado de la tableta si es error se crea un archivo log con la descripcion Si la acci?n es 4: Regresa la informacion de la persona que pertenescan a la unidad medica de la tableta Si la acci?n es 5: Recibe la informacion que envia la tableta y la almacena en sus respectivas tablas Si la acci?n es 6: Regresa la informacion de los catalogos y personas que se actuailzaron o agregaron depues de la ultima sincronizacion de la tableta

- **Access** public

session función Servicios::is_step_1($id_tab, $id_version, $version_apk) *[line 128]*

Parámetros de la función:

- *int* **$id_tab** Representa a la MAC de la tableta
- *int* **$version_apk** Representa la version de la apk instalada en la tableta
- **$id_version**

valida que la tableta este asignada a una unidad medica y que tenga un status valido para la sincronizacion

recibe parametros por POST

- **Access** public

echo función Servicios::is_step_2($id_sesion, [$si = ""], [$sf = ""], $id_session) *[line 207]*

Parámetros de la función:

- *int* **$id_session** Representa la session activa para la peticion
- *int* **$si** Bandera que especifica si este paso es llamado por otro paso
- *json* **$sf** Bandera que especifica el comportamiento del armado del json
- **$id_sesion**

Valida que la session este activa, genera los catalogos a enviar en la sincronizacion por primera vez

- **Access** public

void función Servicios::is_step_3($id_sesion, $datos, $id_session) *[line 466]*

Parámetros de la función:

- *int* **$id_session** Representa la session activa para la peticion
- *json* **$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor
- **$id_sesion**

Guarda en la base de datos el estado de la sincronizacion

- **Access** public

echo función Servicios::is_step_4($id_sesion, $id_session) *[line 502]*

Parámetros de la función:

- *int* **$id_session** Representa la session activa para la peticion
- **$id_sesion**

Prepara la informacion de perssonas con su catalogos transaccionales de cada una

- **Access** public

void función Servicios::prueba2($id_accion, [$id_tab = null], [$id_sesion = null], [$version = null]) *[line 1119]*
Parámetros de la función:

- **$id_accion**
- **$id_tab**
- **$id_sesion**
- **$version**

- **Access** public

echo función Servicios::ss_step_5($id_sesion, $datos, $id_session) *[line 689]*
Parámetros de la función:

- *int* **$id_session** Representa la session activa para la peticion
- *json* **$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor
- **$id_sesion**

Recibe los datos que genero la tableta para ser almacenados en la base de datos del servidor

- **Access** public

void función Servicios::ss_step_6($id_sesion, $id_session) *[line 835]*
Parámetros de la función:

- *int* **$id_session** Representa la session activa para la peticion

- **$id_sesion**

prepara los datos para la sincronizacion secuencia, envia unicamente aquellos datos modificados despues de la ultima sincronizacion de la tableta

- **Access** public

void función Servicios::Synchronization($id_accion, $id_tab, $id_session, $version_apk, $datos) *[line 43]*

Parámetros de la función:

- *int* **$id_accion** Representa el tipo de accion que se ejecutara
- *int* **$id_tab** Representa a la MAC de la tableta
- *int* **$id_session** Representa la session activa para la peticion
- *int* **$version_apk** Representa la version de la apk instalada en la tableta
- *json* **$datos** Representa el json que contiene el contenido para la comunicacion tableta - servidor

Metodo principal al que se le hacen las peticiones y es el que se encarga de distribuir la informacion
recibe parametros por POST

- **Access** public

Clase Tableta

[line 11]

Controlador Tableta

- **Package** TES
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Tableta::__construct() *[line 13]*

- **Access** public

void función Tableta::delete($id) *[line 340]*
Parámetros de la función:

- *int* **$id** ID del elemento a eliminar

Eliminar el registro especificado por el id

- **Access** public

void función Tableta::index([$pag = 0]) *[line 35]*
Parámetros de la función:

- *int* **$pag** Establece el desplazamiento del primer registro a devolver

Lista todos los registros de tabletas, con su correspondiente paginación permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico

- **Access** public

void función Tableta::insert() *[line 163]*

Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST

- **Access** public

redirect función Tableta::setUM($id) *[line 461]*

Parámetros de la función:

- *int* **$id**

Asignar unidad medica y tipo de censo

- **Access** public

void función Tableta::update($id) *[line 220]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra el formulario con los datos del registro especificado por el id, para actualizar sus datos

- **Access** public

redirect función Tableta::uploadFile() *[line 393]*

Registra tabletas desde un archivo csv

- **Access** public

void función Tableta::view($id) *[line 285]*

Parámetros de la función:

- *int* **$id** ID del elemento a actualizar

Muestra los datos del registro especificado por el id

- **Access** public

boolean función Tableta::_validateMac($mac) *[line 373]*

Parámetros de la función:

- *type* **$mac**

Valida si existe una MAC en la base de datos

- **Access** public

Clase Usuario_tableta

[line 11]

Controlador Usuario_tableta

- **Package** TES
- **Sub-Package** Controlador
- **Author** Pascual

Constructor *void* función Usuario_tableta::__construct() *[line 13]*

- **Access** public

void función Usuario_tableta::delete($id_usuario, $id_tableta, $$id_usuario) *[line 143]*

Parámetros de la función:

- *int* **$$id_usuario** ID del usuario a eliminar
- *int* **$id_tableta** ID de la tableta que tiene asignada el usuario especificado
- **$id_usuario**

Eliminar un usuario de una tableta especifica

- **Access** public

void función Usuario_tableta::index($idTableta) *[line 36]*

Parámetros de la función:

- *int* **$idTableta** ID de la Tableta

Lista todos los registros de usuarios correspondientes a una tableta en especifico permite eliminar un conjunto de registro o un elemento individual, muestra enlaces para actualizar y ver detalles de un elemento especifico

- **Access** public

void función Usuario_tableta::insert($id_tableta) *[line 102]*
Parámetros de la función:

- **$id_tableta**

Muestra el formulario para crear un nuevo registro en la tableta, las variables se obtienen por el metodo POST

- **Access** public

Clase Enrolamiento_model

[line 10]

Modelo Usuario

- **Package** TES
- **Sub-Package** Modelo
- **Author** Eliecer

Constructor *void* función Enrolamiento_model::__construct() *[line 118]*

- **Access** public

result() función Enrolamiento_model::autocomplete_tutor($keywords) *[line 2297]*
Parámetros de la función:

- *string* **$keywords** palabras claves para hacer el filtro

obtiene informacion del tutor para genberar el autocomplete

- **Access** public

result() función Enrolamiento_model::cns_insert($tabla, $array) *[line 727]*
Parámetros de la función:

- *string* **$tabla** Nombre de la tabla a la que se afectara
- *array* **$array** Datos que se guardaran en la tabla

Hace insert de las tablas cns_control_x que se reciben en la sincronizacion secuencial

- **Access** public

result() función Enrolamiento_model::cns_update($tabla, $array, $id, [$campo = ""], [$valor = ""], [$campo2 = ""], [$valor2 = ""]) *[line 751]*
Parámetros de la función:

- *string* **$tabla** Nombre de la tabla afectada
- *array* **$array** Datos a actualizar
- *string* **$id** identificador para el where
- *string* **$campo** campo si se necesitarar un segundo where
- *string* **$valor** valor del campo a comparar
- **$campo2**

- **$valor2**

Actualiza la tabla especificada

- **Access** public

void función Enrolamiento_model::cns_update_visita($id) *[line 767]*
Parámetros de la función:

- **$id**

- **Access** public

result() función Enrolamiento_model::data_tutor($curp) *[line 2275]*
Parámetros de la función:

- *string* **$curp** Curp del tutor

obtiene informacion del tutor

- **Access** public

result() función Enrolamiento_model::entorno_x_persona($entorno, $persona, $fecha, $archivo, $impreso) *[line 1110]*
Parámetros de la función:

- *string* **$entorno** id del entorno

- *strin* **$persona** id de la persona a la que se le asigna una tarjeta
- *string* **$fecha** fecha en que se genera el evento
- *string* **$archivo** nombre del archivo que se genero
- *boolena* **$impreso** determina si el archivo fue escrita en la tarjeta o no

Este metodo actualiza o inserta los datos que permiten el envio de la informacion a la tarjeta por nfc

- **Access** public

void función Enrolamiento_model::getaccion_nutricional() *[line 548]*

- **Access** public

void función Enrolamiento_model::getafiliacion() *[line 468]*

- **Access** public

result() función Enrolamiento_model::getAfiliaciones([$id = ''], [$order = ""]) *[line 1830]*

Parámetros de la función:

- *string* **$id** identificador de la persona
- *strin* **$order** nombre del campo para hacer el order by

Obtiene las afiliaciones asociadas a una persona

- **Access** public

void función Enrolamiento_model::getageb() *[line 437]*

- **Access** public

result() función Enrolamiento_model::getAlergia([$id = ''], [$order = '']) *[line 1799]*
Parámetros de la función:

- *string* **$id** identificador de la persona
- *strin* **$order** nombre del campo para hacer el order by

Obtiene las alergias asociadas a una persona

- **Access** public

void función Enrolamiento_model::getalergias() *[line 478]*

- **Access** public

void función Enrolamiento_model::getaltura() *[line 578]*

- **Access** public

result() función Enrolamiento_model::getByCurp($curp, $tabla, $id) *[line 2328]*
Parámetros de la función:

- *string* **$curp** Curp a validar
- *strin* **$tabla** Tabla en la que se debe hacer la validacion
- *string* **$id** id de la persona o tutor para excluirse

valida que no se repita la curp en personas y tutor

- **Access** public

result() función Enrolamiento_model::getById($id) *[line 1739]*

Parámetros de la función:

- *string* **$id** identificado de la persona

Obtiene la informacion de la persona

- **Access** public

void función Enrolamiento_model::getcalle() *[line 387]*

- **Access** public

object|boolean función Enrolamiento_model::getCategoriaCIE10() *[line 2717]*

Obtiene el listado de categorias de CIE10

- **Access** public

void función Enrolamiento_model::getcelular() *[line 637]*

- **Access** public

void función Enrolamiento_model::getcelularT() *[line 337]*

- **Access** public

object|boolean función Enrolamiento_model::getCIE10($categoria) *[line 2746]*

Parámetros de la función:

- **$categoria**

Obtiene el listado de CIE10 correspondientes a una CIE10

- **Access** public

void función Enrolamiento_model::getcodigo_barras() *[line 508]*

- **Access** public

void función Enrolamiento_model::getcolonia() *[line 397]*

- **Access** public

void función Enrolamiento_model::getcompania() *[line 627]*

- **Access** public

void función Enrolamiento_model::getcompaniaT() *[line 327]*

- **Access** public

void función Enrolamiento_model::getconsulta() *[line 518]*

- **Access** public

object|boolean función Enrolamiento_model::getControlConsultas($idPersona) *[line 2776]*

Parámetros de la función:

- **$idPersona**

Obtiene todas las consultas asociadas a un paciente

- **Access** public

void función Enrolamiento_model::getcp() *[line 427]*

- **Access** public

void función Enrolamiento_model::getcurp() *[line 179]*

- **Access** public

void función Enrolamiento_model::getcurpT() *[line 298]*

- **Access** public

void función Enrolamiento_model::getestimulacion_capacitado() *[line 687]*

- **Access** public

void función Enrolamiento_model::getestimulacion_fecha() *[line 677]*

- **Access** public

void función Enrolamiento_model::getfaccion_nutricional() *[line 558]*

- **Access** public

void función Enrolamiento_model::getfconsulta() *[line 528]*

- **Access** public

void función Enrolamiento_model::getfechacivil() *[line 347]*

- **Access** public

void función Enrolamiento_model::getfecha_peri_cefa() *[line 667]*

- **Access** public

void función Enrolamiento_model::getfnacimiento() *[line 209]*

- **Access** public

void función Enrolamiento_model::getfnutricion() *[line 607]*

- **Access** public

result() función Enrolamiento_model::getfolio($persona) *[line 1158]*

Parámetros de la función:

- *strin* **$persona** id de la persona a la que se le asigna una tarjeta

Extrae el folio que se anexa en el envio para la tarjeta

- **Access** public

void función Enrolamiento_model::getfvacuna() *[line 498]*

- **Access** public

void función Enrolamiento_model::gethemoglobina() *[line 597]*

- **Access** public

void función Enrolamiento_model::getId() *[line 129]*

- **Access** public

void función Enrolamiento_model::getidtutor() *[line 259]*

- **Access** public

result() función Enrolamiento_model::getListEnrolamiento([$keywords = ''], [$offset = null], [$row_count = null]) *[line 1594]*

Parámetros de la función:

- *string* **$keywords** palabras claves para hacer el filtro
- *strin* **$offset** inicio del registro
- *string* **$row_count** numero de filas a mostrar por pagina

Este metodo retorna el list de las personas enroladas

- **Access** public

void función Enrolamiento_model::getlnacimiento() *[line 169]*

- **Access** public

void función Enrolamiento_model::getlocalidad() *[line 407]*

- **Access** public

void función Enrolamiento_model::getlugarcivil() *[line 357]*

- **Access** public

void función Enrolamiento_model::getmanzana() *[line 457]*

- **Access** public

void función Enrolamiento_model::getmaterno() *[line 159]*

- **Access** public

void función Enrolamiento_model::getmaternoT() *[line 289]*

- **Access** public

void función Enrolamiento_model::getMsgError([$value = 'usr']) *[line 2386]*

Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::getnacionalidad() *[line 647]*

- **Access** public

void función Enrolamiento_model::getnombre() *[line 139]*

- **Access** public

void función Enrolamiento_model::getnombreT() *[line 269]*

- **Access** public

void función Enrolamiento_model::getnumero() *[line 417]*

- **Access** public

result() función Enrolamiento_model::getNumRows([$keywords = '']) *[line 1670]*

Parámetros de la función:

- *string* **$keywords** palabras clave para hacer el filtro

Devuelve el numero de filas en la tabla cns_persona

- **Access** public

void función Enrolamiento_model::getparto() *[line 229]*

- **Access** public

void función Enrolamiento_model::getpaterno() *[line 149]*

- **Access** public

void función Enrolamiento_model::getpaternoT() *[line 279]*

- **Access** public

void función Enrolamiento_model::getperi_cefa() *[line 657]*

- **Access** public

void función Enrolamiento_model::getpeso() *[line 568]*

- **Access** public

void función Enrolamiento_model::getprecurp() *[line 239]*

- **Access** public

void función Enrolamiento_model::getreferencia() *[line 377]*

- **Access** public

result() función Enrolamiento_model::getRegistro_civil($id) *[line 1772]*

Parámetros de la función:

- *string* **$id** identificador de la persona

obtiene informacion del registro civil

- **Access** public

void función Enrolamiento_model::getsales_cantidad() *[line 697]*

- **Access** public

void función Enrolamiento_model::getsales_fecha() *[line 707]*

- **Access** public

void función Enrolamiento_model::getsangre() *[line 199]*

- **Access** public

void función Enrolamiento_model::getsector() *[line 447]*

- **Access** public

void función Enrolamiento_model::getsexo() *[line 189]*

- **Access** public

void función Enrolamiento_model::getsexoT() *[line 308]*

- **Access** public

void función Enrolamiento_model::gettalla() *[line 587]*

- **Access** public

void función Enrolamiento_model::gettamiz() *[line 234]*

- **Access** public

void función Enrolamiento_model::gettbeneficiario() *[line 219]*

- **Access** public

void función Enrolamiento_model::gettconsulta() *[line 538]*

- **Access** public

void función Enrolamiento_model::gettelefono() *[line 617]*

- **Access** public

void función Enrolamiento_model::gettelefonoT() *[line 317]*

- **Access** public

void función Enrolamiento_model::getumt() *[line 367]*

- **Access** public

void función Enrolamiento_model::getvacuna() *[line 488]*

- **Access** public

result() función Enrolamiento_model::get_catalog($catalog, [$campo = ""], [$id = ""], [$orden = ""], $order) *[line 1925]*

Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- *strin* **$campo** nombre del campo para hacer el where
- *string* **$id** valor del campo para el where
- *strin* **$order** nombre del campo para hacer el order by
- **$orden**

Hace select de las tablas cns_x que representa a los catalogos

- **Access** public

result() función Enrolamiento_model::get_catalog2($catalog, [$campo1 = ""], [$id1 = ""], [$campo2 = ""], [$id2 = ""], [$l1 = ""], [$l2 = ""]) *[line 2022]*

Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- *strin* **$campo1** nombre del campo para hacer el where
- *string* **$id1** valor del campo para el where
- *strin* **$campo2** nombre del campo para hacer el where

- *string* **$id2** valor del campo para el where
- *strin* **$l1** nombre del campo para hacer el limit offset
- *strin* **$l2** nombre del campo para hacer el limit count

Obtiene los datos de una tabla

- **Access** public

count función Enrolamiento_model::get_catalog_count($catalog, [$campo = ""], [$valor = ""]) *[line 1995]*

Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- **$campo**
- **$valor**

Obtiene el numero de resultados de una tabla

- **Access** public

result() función Enrolamiento_model::get_catalog_relevante([$fecha = ""]) *[line 2199]*

Parámetros de la función:

- **$fecha**

obtiene los catalogos relevante x entorno para la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_catalog_tratamiento($catalog, $campo, $valor, $orden, $order) *[line 1961]*

Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- *strin* **$campo** nombre del campo para hacer el where
- *string* **$valor** valor del campo para el where
- *strin* **$order** nombre del campo para hacer el order by
- **$orden**

Hace select de los tratamientos de las consultas

- **Access** public

result() función Enrolamiento_model::get_catalog_view($catalog, $id, [$order1 = ""], [$order2 = ""]) *[line 1862]*

Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- *string* **$id** identificador de la persona
- *strin* **$order1** nombre del campo para hacer el order by
- *strin* **$order2** nombre del campo para hacer el order by

Hace select de los catalogos que tengan relacion con una persona para mostrarlos en el view

- **Access** public

result() función Enrolamiento_model::get_cns_cat_persona($catalog, $array, [$l1 = ""], [$l2 = ""]) *[line 2123]*

Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- *strin* **$array** ids de personas
- *string* **$l1** ofsset para el limit
- *strin* **$l2** count para el limit

Este metodo obtiene los controles que le corresponde a cada persona y que seran incluidas en la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_cns_cat_persona_count($catalog, $persona, $personas) *[line 2158]*
Parámetros de la función:

- *string* **$catalog** Nombre de la tabla
- *strin* **$personas** personas que cumplen el requisito
- **$persona**

Hace el count de personas que se envian en la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_cns_persona($array, [$fecha = ""]) *[line 2089]*
Parámetros de la función:

- *string* **$array** arreglo con los ids de las personas que cumplen con los requisitos del envio
- *strin* **$fecha** fecha que determina si se envia o no una persona

Este metodo obtiene las personas que seran enviadas en la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_control_nutricional($id, [$order = ""]) *[line 1893]*
Parámetros de la función:

- *string* **$id** identificador de la persona
- *strin* **$order** nombre del campo para hacer el order by

Obtiene los datos del control nutricional asociados a una persona

- **Access** public

void función Enrolamiento_model::get_datos_grafica($catalogo, $sexo, $edad_meses, $id_persona, [$asu_locali = '']) *[line 2404]*
Parámetros de la función:

- *int* **$catalogo** Determina el catalogo a consultar
- *int* **$sexo** El sexo del paciente (F, M)
- *int* **$edad_meses** Edad del paciente en meses
- *int* **$id_persona** Identificador del paciente
- *int* **$asu_locali** Identificador del asu de la localidad del domicilio

Obtiene los datos especificos de un catálogo para ser visualizados en una gráfica

- **Access** public

result() función Enrolamiento_model::get_estimulacion($id, [$order = ""]) *[line 2661]*
Parámetros de la función:

- *string* **$id** identificador de la persona
- *strin* **$order** nombre del campo para hacer el order by

Obtiene los registros de estimulacion temprana asociados a una persona

- **Access** public

result() función Enrolamiento_model::get_notificacion($id) *[line 2062]*
Parámetros de la función:

- *string* **$id** id del arbol de segmentacion

Este metodo obtiene las notificaciones que se enviaran en la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_pacientes() *[line 2368]*
devuelve todos los pacientes de la base de datos

- **Access** public

result() función Enrolamiento_model::get_peri_cefa($id, [$order = ""]) *[line 2601]*
Parámetros de la función:

- *string* **$id** identificador de la persona
- *strin* **$order** nombre del campo para hacer el order by

Obtiene los registros de perimetro cefalico asociados a una persona

- **Access** public

result() función Enrolamiento_model::get_persona_x_tutor($array) *[line 2176]*
Parámetros de la función:

- *string* **$array** tutores que tienen asignado un paciente

obtiene los tutores de las personas que se envian en la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_sales($id, [$order = ""]) *[line 2866]*
Parámetros de la función:

- *string* **$id** identificador de la persona
- *strin* **$order** nombre del campo para hacer el order by

Obtiene los registros de sales de rehidratación oral asociados a una persona

- **Access** public

result() función Enrolamiento_model::get_transaction_relevante() *[line 2225]*
Obtiene las transacciones relevante para la sincronizacion

- **Access** public

result() función Enrolamiento_model::get_version() *[line 2249]*

obtiene cual es la ultima version de apk de la tableta

- **Access** public

result() función Enrolamiento_model::insert() *[line 787]*

Guarda la persona capturada mediante el formulario web

- **Access** public

void función Enrolamiento_model::setaccion_nutricional($value) *[line 553]*

Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setafiliacion($value) *[line 473]*

Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setageb($value) *[line 442]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setalergias($value) *[line 483]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setaltura($value) *[line 583]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setcalle($value) *[line 392]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setcelular($value) *[line 642]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setcelularT($value) *[line 342]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setcodigo_barras($value) *[line 513]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setcolonia($value) *[line 402]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setcompania($value) *[line 632]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setcompaniaT($value) *[line 332]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setconsulta($value) *[line 523]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setcp($value) *[line 432]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setcurp($value) *[line 184]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setcurpT($value) *[line 303]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setestimulacion_capacitado($value) *[line 692]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setestimulacion_fecha($value) *[line 682]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setfaccion_nutricional($value) *[line 563]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setfconsulta($value) *[line 533]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setfechacivil($value) *[line 352]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setfecha_peri_cefa($value) *[line 672]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setfnacimiento($value) *[line 214]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setfnutricion($value) *[line 612]*

Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setfvacuna($value) *[line 503]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::sethemoglobina($value) *[line 602]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setId($value) *[line 134]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setidtutor($value) *[line 264]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setlnacimiento($value) *[line 174]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setlocalidad($value) *[line 412]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setlugarcivil($value) *[line 362]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setmanzana($value) *[line 462]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setmaterno($value) *[line 164]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setmaternoT($value) *[line 293]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setnacionalidad($value) *[line 652]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setnombre($value) *[line 144]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setnombreT($value) *[line 274]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setnumero($value) *[line 422]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setparto($value) *[line 249]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setpaterno($value) *[line 154]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setpaternoT($value) *[line 284]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setperi_cefa($value) *[line 662]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setpeso($value) *[line 573]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setprecurp($value) *[line 244]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setreferencia($value) *[line 382]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setsales_cantidad($value) *[line 702]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setsales_fecha($value) *[line 712]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setsangre($value) *[line 204]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setsector($value) *[line 452]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setsexo($value) *[line 194]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::setsexoT($value) *[line 313]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::settalla($value) *[line 592]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::settamiz($value) *[line 254]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::settbeneficiario($value) *[line 224]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::settconsulta($value) *[line 543]*
Parámetros de la función:

- **$value**

- **Access** public

void función Enrolamiento_model::settelefono($value) *[line 622]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::settelefonoT($value) *[line 322]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setumt($value) *[line 372]*
Parámetros de la función:

- **$value**

    - **Access** public

void función Enrolamiento_model::setvacuna($value) *[line 493]*
Parámetros de la función:

- **$value**

- **Access** public

result() función Enrolamiento_model::tes_pendientes_tarjeta_delete() *[line 2350]*

Elimina los pendientes de las personas que no tengan asignado una tarjeta

- **Access** public

result() función Enrolamiento_model::update_accion() *[line 1459]*

Actualiza el control accion nutricional del paciente

- **Access** public

result() función Enrolamiento_model::update_alergia() *[line 1360]*

Actualiza los datos de las alergias del paciente

- **Access** public

result() función Enrolamiento_model::update_basico() *[line 1171]*

Actualiza los datos basicos del paciente

- **Access** public

result() función Enrolamiento_model::update_beneficiario() *[line 1528]*

Actualiza el tipo de beneficiario del paciente

- **Access** public

result() función Enrolamiento_model::update_consulta() *[line 1424]*

Actualiza el control consulta del paciente

- **Access** public

result() función Enrolamiento_model::update_direccion() *[line 1227]*

actualiza la direccion del paciente

- **Access** public

result() función Enrolamiento_model::update_estimulacion() *[line 2689]*

Actualiza los registros de estimulacion temprana

- **Access** public

result() función Enrolamiento_model::update_nutricion() *[line 1492]*

Actualiza el control nutricional del paciente

- **Access** public

result() función Enrolamiento_model::update_peri_cefa() *[line 2629]*

Actualiza los registros de perimetro cefalico

- **Access** public

result() función Enrolamiento_model::update_regcivil() *[line 1265]*

actualiza el registro civil del paciente

- **Access** public

result() función Enrolamiento_model::update_sales() *[line 2834]*

Actualiza los registros de sales de rehidratacion oral

- **Access** public

result() función Enrolamiento_model::update_status_tableta($mac, $status, $version, $fecha) *[line 1566]*

Parámetros de la función:

- *string* **$mac** Mac de la tableta
- *strin* **$status** nuevo status
- *string* **$version** version de la apk de la tableta
- *string* **$fecha** fecha del evento

Hace update de la tableta que este sincronizando dependiendo del resultado

- **Access** public

result() función Enrolamiento_model::update_tutor() *[line 1289]*

Actualiza los datos del tutor del paciente

- **Access** public

result() función Enrolamiento_model::update_umt() *[line 1204]*

Actualiza la unidad medica tratante del paciente

- **Access** public

result() función Enrolamiento_model::update_vacuna() *[line 1391]*

Actualiza las vacunas del paciente

- **Access** public

result() función Enrolamiento_model::valid_card($persona, $archivo) *[line 1144]*

Parámetros de la función:

- *strin* **$persona** id de la persona a la que se le asigna una tarjeta
- *string* **$archivo** nombre del archivo que se genero

Este metodo valida que exista un archivo para enviar a la tarjeta por nfc

- **Access** public

Clase Estado_tableta_model

[line 11]

Modelo Estado_tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Estado_tableta_model::__construct() *[line 49]*

- **Access** public

array función Estado_tableta_model::getAll() *[line 140]*

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Estado_tableta_model::getById($id) *[line 112]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

void función Estado_tableta_model::getDescripcion() *[line 67]*

- **Access** public

void función Estado_tableta_model::getId() *[line 62]*

- **Access** public

boolean|*string* función Estado_tableta_model::getMsgError([$type = 'usr']) *[line 91]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

void función Estado_tableta_model::setDescripcion($descripcion) *[line 76]*

Parámetros de la función:

- **$descripcion**

- **Access** public

void función Estado_tableta_model::setId($id) *[line 71]*
Parámetros de la función:

- **$id**

- **Access** public

Clase Notificacion_model

[line 10]

Modelo Usuario

- **Package** TES
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Notificacion_model::__construct() *[line 74]*

- **Access** public

void|boolean función Notificacion_model::addFilter($columna, $condicion, $valor) *[line 324]*
Parámetros de la función:

- *string* **$columna** Puede ser cualquier campo del objeto (id, id_usuario, fecha_hora, parametros, id_controlador_accion)
- *string* **$condicion** Establece la condicion a evaluar, entre los valores permitidos estan: =, !=, >=, <=, like
- *string* **$valor** Valor contra el cual se realizará la evaluación del campo

Agrega una nueva regla de filtrado al arreglo de filtros

- **Access** public

boolean función Notificacion_model::delete() *[line 304]*

Elimina de la base de datos la notificaci?n (id en propiedades)

- **Access** public

void|*array* función Notificacion_model::getAll([$keywords = ''], [$offset = null], [$row_count = null]) *[line 170]*

Parámetros de la función:

- *boolean*|*string* **$keywords** false no hay texto a buscar|string con texto a buscar
- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver

Obtiene todas las notificaciones existentes, se puede filtrar por: texto a buscar si se desea

- **Access** public

void|*object* función Notificacion_model::getById($id) *[line 209]*

Parámetros de la función:

- *int* **$id** id de notificaci?n

Obtiene la notificaci?n solicitada

- **Access** public

void función Notificacion_model::getContenido() *[line 105]*

- **Access** public

void función Notificacion_model::getFechaFin() *[line 125]*

- **Access** public

void función Notificacion_model::getFechaInicio() *[line 115]*

- **Access** public

void función Notificacion_model::getId() *[line 86]*

- **Access** public

void función Notificacion_model::getIdsTabletas() *[line 135]*

- **Access** public

boolean función Notificacion_model::getMsgError([$value = 'usr']) *[line 152]*
Parámetros de la función:

- *string* **$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

int función Notificacion_model::getNumRows([$keywords = '']) *[line 229]*
Parámetros de la función:

- *boolean*|*string* **$keywords** false no hay texto a buscar|string con texto a buscar

Obtiene el numero total de notificaciones

- **Access** public

void función Notificacion_model::getTitulo() *[line 95]*

- **Access** public

boolean función Notificacion_model::insert() *[line 255]*
Inserta en la base de datos los datos de la notificaci?n (datos en propiedades)

- **Access** public

void función Notificacion_model::setContenido($contenido) *[line 110]*
Parámetros de la función:

- **$contenido**

- **Access** public

void función Notificacion_model::setFechaFin($fecha_fin) *[line 130]*
Parámetros de la función:

- **$fecha_fin**

- **Access** public

void función Notificacion_model::setFechaInicio($fecha_inicio) *[line 120]*
Parámetros de la función:

- **$fecha_inicio**

- **Access** public

void función Notificacion_model::setId($value) *[line 91]*
Parámetros de la función:

- **$value**

- **Access** public

void función Notificacion_model::setIdsTabletas($id_arr_asu) *[line 140]*
Parámetros de la función:

- **$id_arr_asu**

- **Access** public

void función Notificacion_model::setTitulo($titulo) *[line 100]*
Parámetros de la función:

- **$titulo**

- **Access** public

boolean función Notificacion_model::update() *[line 279]*
Actualiza en la base de datos los datos de la notificaci?n (datos en propiedades)

- **Access** public

Clase Reporteador_model

[line 10]

Modelo Reporteador

- **Package** TES
- **Sub-Package** Modelo
- **Author** Rogelio

Constructor *void* función Reporteador_model::__construct() *[line 27]*

- **Access** public

void función Reporteador_model::getCensoNominal($nivel, $id, &$th) *[line 319]*

Parámetros de la función:

- **$nivel**
- **$id**
- **&$th**

- **Access** public

void función Reporteador_model::getCoberturaBiologicoListado($nivel, $id, $fecha, [$fechaFin = null]) *[line 59]*

Parámetros de la función:

- *int* **$nivel** Nivel del elemento del arbol ASU

- *int* **$id** Identificador del elemento ASU
- *date* **$fecha** Fecha de corte de elemento
- **$fechaFin**

Obtiene el reporte de cobertura por tipo de biologico

- **Access** public

void función Reporteador_model::getConcentradoActividades($nivel, $id, $fecha) *[line 289]*

Parámetros de la función:

- **$nivel**
- **$id**
- **$fecha**

- **Access** public

void función Reporteador_model::getEsquemasIncompletos($nivel, $id, &$th) *[line 414]*

Parámetros de la función:

- **$nivel**
- **$id**
- **&$th**

- **Access** public

void función Reporteador_model::getGrupoVacunas() *[line 598]*

boolean función Reporteador_model::getMsgError([$value = 'usr']) *[line 43]*

Parámetros de la función:

- *string* **$value** tipo de error a visualizar: usr o log, default: usr

Asigna el mensaje de error a visualizar: para usuario final (usr) o para bitácora (log)

- **Access** public

void función Reporteador_model::getSeguimientoRV1RV5($nivel, $id, $fecha) *[line 304]*

Parámetros de la función:

- **$nivel**
- **$id**
- **$fecha**

- **Access** public

void función Reporteador_model::getVacunas() *[line 547]*

void función Reporteador_model::getVacunasByGrupo($grupo) *[line 573]*

Parámetros de la función:

- **$grupo**

void función Reporteador_model::object_to_array($data, $campo1) *[line 532]*

Parámetros de la función:

- **$data**
- **$campo1**

Clase Reporte_censo_nominal

[line 11]

Modelo Reporte_censo_nominal

- **Package** TES
- **Sub-Package** Modelo
- **Author** Rogelio

Reporte_censo_nominal::$apellido_materno

varchar = [line 21]

- **Access** public

Reporte_censo_nominal::$apellido_paterno

varchar = [line 16]

- **Access** public

Reporte_censo_nominal::$curp

varchar = [line 36]

- **Access** public

Reporte_censo_nominal::$domicilio

varchar = [line 31]

- **Access** public

Reporte_censo_nominal::$fecha_nacimiento

varchar = [line 41]

- **Access** public

Reporte_censo_nominal::$nombre

varchar = [line 26]

- **Access** public

Reporte_censo_nominal::$parto_multiple

varchar = [line 46]

- **Access** public

Reporte_censo_nominal::$sexo

varchar = [line 51]

- **Access** public

Reporte_censo_nominal::$vacunas

array = [line 56]

- **Access** public

Constructor *void* función Reporte_censo_nominal::__construct() *[line 58]*

- **Access** public

Clase Reporte_sincronizacion_model

[line 10]

Modelo Usuario

- **Package** TES
- **Sub-Package** Modelo
- **Author** Eliecer

num_rows() función Reporte_sincronizacion_model::getCount($tabla, [$sentecia = ""], $sentencia) *[line 55]*

Parámetros de la función:

- *string* **$tabla** nombre de una tabla en la base de datos
- *string* **$sentencia** consulta sql
- **$sentecia**

obtiene el numero de registros de una tabla o consulta

- **Access** public

result() función Reporte_sincronizacion_model::getListado($sql) *[line 31]*
Parámetros de la función:

- *string* **$sql** consulta slq a ejecutar

Obtiene el resultado de una consulta

- **Access** public

void función Reporte_sincronizacion_model::getMsgError([$value = 'usr']) *[line 88]*
Parámetros de la función:

- **$value**

- **Access** public

result() función Reporte_sincronizacion_model::get_version() *[line 71]*
obtiene la ultima version de la apk de las tabletas

- **Access** public

Clase Semana_nacional_model

[line 11]

Modelo Tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Semana_nacional_model::__construct() *[line 61]*

- **Access** public

int función Semana_nacional_model::delete([$id = null]) *[line 203]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

array función Semana_nacional_model::getAll([$offset = null], [$row_count = null]) *[line 281]*

Parámetros de la función:

- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Semana_nacional_model::getById($id) *[line 243]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

void función Semana_nacional_model::getDescripcion() *[line 79]*

- **Access** public

void función Semana_nacional_model::getFecha_fin() *[line 89]*

- **Access** public

void función Semana_nacional_model::getFecha_inicio() *[line 84]*

- **Access** public

void función Semana_nacional_model::getId() *[line 74]*

- **Access** public

boolean|*string* función Semana_nacional_model::getMsgError([$type = 'usr']) *[line 120]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

int función Semana_nacional_model::getNumRows() *[line 316]*

Obtiene el numero total de registros en la tabla

- **Access** public

boolean función Semana_nacional_model::insert() *[line 140]*

Inserta en la base de datos, la informacion contenida en el objeto

- **Access** public

void función Semana_nacional_model::setDescripcion($descripcion) *[line 94]*
Parámetros de la función:

- **$descripcion**

- **Access** public

void función Semana_nacional_model::setFecha_fin($fecha_fin) *[line 104]*
Parámetros de la función:

- **$fecha_fin**

- **Access** public

void función Semana_nacional_model::setFecha_inicio($fecha_inicio) *[line 99]*
Parámetros de la función:

- **$fecha_inicio**

- **Access** public

boolean función Semana_nacional_model::update([$id = null]) *[line 171]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase Tableta_model

[line 11]

Modelo Tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Tableta_model::__construct() *[line 90]*

- **Access** public

int función Tableta_model::delete([$id = null]) *[line 292]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Elimina el registro actual de la base de datos

- **Access** public

array función Tableta_model::getAll([$offset = null], [$row_count = null], $filtro) *[line 419]*

Parámetros de la función:

- *int* **$offset** Establece el desplazamiento del primer registro a devolver, si se define solo el valor de offset el valor especifica el número de registros a retornar desde el comienzo del conjunto de resultados.
- *int* **$row_count** Establece la cantidad de registros a devolver
- **$filtro**

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Tableta_model::getById($id) *[line 332]*

Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

object|boolean función Tableta_model::getByMac($mac) *[line 374]*

Parámetros de la función:

- *int* **$mac** Direccion MAC

Obtiene los datos del registro la MAC especificada

- **Access** public

void función Tableta_model::getId() *[line 103]*

- **Access** public

void función Tableta_model::getIdVersion() *[line 112]*

- **Access** public

void función Tableta_model::getId_asu_um() *[line 132]*

- **Access** public

void función Tableta_model::getId_tes_estado_tableta() *[line 124]*

- **Access** public

void función Tableta_model::getId_tipo_censo() *[line 128]*

- **Access** public

void función Tableta_model::getMac() *[line 108]*

- **Access** public

boolean|*string* función Tableta_model::getMsgError([$type = 'usr']) *[line 188]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

int función Tableta_model::getNumRows() *[line 518]*

Obtiene el numero total de registros en la tabla en caso de existir filtros, estos son aplicados a la consulta

- **Access** public

void función Tableta_model::getPeriodo_esq_inc() *[line 136]*

- **Access** public

void función Tableta_model::getUltima_actualizacion() *[line 116]*

- **Access** public

void función Tableta_model::getUsuarios_asignados() *[line 120]*

- **Access** public

boolean función Tableta_model::insert() *[line 208]*

Inserta en la base de datos, la informacion contenida en el objeto

- **Access** public

void función Tableta_model::setId($id) *[line 140]*

Parámetros de la función:

- **$id**

- **Access** public

void función Tableta_model::setIdVersion($id_version) *[line 149]*

Parámetros de la función:

- **$id_version**

- **Access** public

void función Tableta_model::setId_asu_um($id_asu_um) *[line 169]*

Parámetros de la función:

- **$id_asu_um**

    - **Access** public

void función Tableta_model::setId_tes_estado_tableta($id_tes_estado_tableta) *[line 161]*
Parámetros de la función:

- **$id_tes_estado_tableta**

    - **Access** public

void función Tableta_model::setId_tipo_censo($id_tipo_censo) *[line 165]*
Parámetros de la función:

- **$id_tipo_censo**

    - **Access** public

void función Tableta_model::setMac($mac) *[line 145]*
Parámetros de la función:

- **$mac**

- **Access** public

void función Tableta_model::setPeriodo_esq_inc($periodo_esq_inc) *[line 173]*
Parámetros de la función:

- **$periodo_esq_inc**

- **Access** public

void función Tableta_model::setUltima_actualizacion($ultima_actualizacion) *[line 153]*
Parámetros de la función:

- **$ultima_actualizacion**

- **Access** public

void función Tableta_model::setUsuarios_asignados($usuarios_asignados) *[line 157]*
Parámetros de la función:

- **$usuarios_asignados**

- **Access** public

boolean función Tableta_model::update([$id = null]) *[line 239]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Actualiza los datos del objeto actual

- **Access** public

Clase Tipo_censo_model

[line 11]

Modelo Tipo_censo

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Tipo_censo_model::__construct() *[line 49]*

- **Access** public

array función Tipo_censo_model::getAll() *[line 141]*

Obtiene todos los registros de la tabla

- **Access** public

object|boolean función Tipo_censo_model::getById($id) *[line 113]*
Parámetros de la función:

- *int* **$id** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene los datos del registro que tiene el ID especificado

- **Access** public

void función Tipo_censo_model::getDescripcion() *[line 67]*

- **Access** public

void función Tipo_censo_model::getId() *[line 62]*

- **Access** public

boolean|string función Tipo_censo_model::getMsgError([$type = 'usr']) *[line 92]*
Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

void función Tipo_censo_model::setDescripcion($descripcion) *[line 77]*
Parámetros de la función:

- **$descripcion**

- **Access** public

void función Tipo_censo_model::setId($id) *[line 72]*
Parámetros de la función:

- **$id**

- **Access** public

Clase Usuario_tableta_model

[line 11]

Modelo Estado_tableta

- **Package** TES
- **Sub-Package** Modelo
- **Author** Pascual

Constructor *void* función Usuario_tableta_model::__construct() *[line 49]*

- **Access** public

boolean función Usuario_tableta_model::delete($id_usuario, [$id_tableta = null]) *[line 161]*

Parámetros de la función:

- *int* **$id_usuario**
- *int* **$id_tableta** Si no se proporciona el parametro, se toma el id del objeto actual

Elimina la relacion entre usuario y tableta

- **Access** public

boolean|*string* función Usuario_tableta_model::getMsgError([$type = 'usr']) *[line 91]*

Parámetros de la función:

- *string* **$type** usr si se quiere devolver el mensaje de error a mostrar en la vista, log obtiene el mensaje de error con mas detalles para depuración, valor por defecto usr

Devuelve el mensaje de error, en caso de existir un error despues de ejecutar un metodo, de lo contrario false

- **Access** public

void función Usuario_tableta_model::getTableta() *[line 62]*

- **Access** public

object|boolean función Usuario_tableta_model::getTabletasByUsuario([$id_usuario = null]) *[line 136]*
Parámetros de la función:

- *int* **$id_usuario** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene el id de todas las tabletas relacionadas con el usuario especificado

- **Access** public

void función Usuario_tableta_model::getUsuario() *[line 67]*

- **Access** public

object|boolean función Usuario_tableta_model::getUsuariosByTableta([$id_tes_tableta = null]) *[line 112]*
Parámetros de la función:

- *int* **$id_tes_tableta** Si no se establece el valor de ID, se toma el valor del objeto actual

Obtiene el id de todos los usuarios asignados a la tableta especificada

- **Access** public

boolean función Usuario_tableta_model::insert([$id_usuario = null], [$id_tableta = null]) *[line 204]*
Parámetros de la función:

- **$id_usuario**
- **$id_tableta**

Inserta en la base de datos, la informacion contenida en el objeto

- **Access** public

void función Usuario_tableta_model::setTableta($id_tes_tableta) *[line 71]*
Parámetros de la función:

- **$id_tes_tableta**

- **Access** public

void función Usuario_tableta_model::setUsuario($id_usuario) *[line 76]*
Parámetros de la función:

- **$id_usuario**

- **Access** public

Apéndices

Apéndice A - Arbol de clases

Paquete SIIGS

Accion

- CI_Controller
  - Accion

Accion_model

- CI_Model
  - Accion_model

Ageb_model

- CI_Model
  - Ageb_model

ArbolSegmentacion_model

- CI_Model
  - ArbolSegmentacion_model

Ayuda

- CI_Controller
  - Ayuda

Bitacora

- CI_Controller
  - Bitacora

Bitacora_model

- CI_Model
  - Bitacora_model

Catalogo

- CI_Controller
  - Catalogo

CatalogoCsv

- CI_Controller
  - CatalogoCsv

CatalogoCsv_model

- CI_Model
  - CatalogoCsv_model

Catalogo_model

- CI_Model
  - Catalogo_model

Catalogo_x_raiz

- CI_Controller
  - Catalogo_x_raiz

Catalogo_x_raiz_model

- CI_Model
  - Catalogo_x_raiz_model

Cie10

- CI_Controller
  - Cie10

Cie10_model

- CI_Model
  - Cie10_model

Controlador

- CI_Controller
  - Controlador

ControladorAccion_model

- CI_Model
  - ControladorAccion_model

Controlador_model

- CI_Model
  - Controlador_model

Entorno

- CI_Controller
  - Entorno

Entorno_model

- CI_Model
  - Entorno_model

Errorlog

- CI_Controller
  - Errorlog

Errorlog_model

- CI_Model
  - Errorlog_model

Georeferencia_model

- CI_Model
  - Georeferencia_model

Grupo

- CI_Controller
  - Grupo

Grupo_model

- CI_Model
  - Grupo_model

Hemoglobina_model

- CI_Model
  - Hemoglobina_model

Menu

- CI_Controller
  - Menu

Menu_model

- CI_Model
  - Menu_model

Permiso

- CI_Controller
  - Permiso

Permiso_model

- CI_Model
  - Permiso_model

Poblacion_model

- CI_Model
  - Poblacion_model

Raiz

- CI_Controller
  - Raiz

Raiz_model

- CI_Model
  - Raiz_model

ReglaVacuna

- CI_Controller
  - ReglaVacuna

ReglaVacuna_model

- CI_Model
  - ReglaVacuna_model

Usuario

- CI_Controller
  - Usuario

Usuario_model

- CI_Model
  - Usuario_model

Paquete default

Index

- CI_Controller
  - Index

Index

- CI_Controller
  - Index

Paquete TES

Enrolamiento

- CI_Controller
  - Enrolamiento

Enrolamiento_model

- CI_Model
  - Enrolamiento_model

Estado_tableta_model

- CI_Model
  - Estado_tableta_model

Notificacion

- CI_Controller
  - Notificacion

Notificacion_model

- CI_Model
  - Notificacion_model

Reporteador

- CI_Controller
  - Reporteador

Reporteador_model

- CI_Model
  - Reporteador_model

Reporte_censo_nominal

- CI_Model
  - Reporte_censo_nominal

Reporte_sincronizacion

- CI_Controller
  - Reporte_sincronizacion

Reporte_sincronizacion_model

- CI_Model
  - Reporte_sincronizacion_model

Semana_nacional

- CI_Controller
  - Semana_nacional

Semana_nacional_model

- CI_Model
  - Semana_nacional_model

Servicios

- CI_Controller
  - Servicios

Tableta

- CI_Controller
  - Tableta

Tableta_model

- CI_Model
  - Tableta_model

Tipo_censo_model

- CI_Model
  - Tipo_censo_model

Usuario_tableta

- CI_Controller
  - Usuario_tableta

Usuario_tableta_model

- CI_Model
  - Usuario_tableta_model

Paquete Libreria

Graph

- CI_Controller
  - Graph

Menubuilder

- Menubuilder

Obtenercurp

- CI_Controller
  - Obtenercurp

Tree

- CI_Controller
  - Tree

Paquete CodeIgniter

CI_Form_validation

- CI_Form_validation

CI_Pagination

- CI_Pagination

CI_Template

- CI_Template

Paquete Session

Session

- Session

Apéndice C - Código fuente

Paquete TES

Archivo fuente para Form_validation.php

La documentación para este archivo está disponible en Form_validation.php

```
1     <?php     if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2     /**
3      * CodeIgniter
4      *
5      * An open source application development framework for PHP 5.1.6 or newer
6      *
7      * @package        CodeIgniter
8      * @author        ExpressionEngine Dev Team
9      * @copyright    Copyright (c) 2008 - 2011, EllisLab, Inc.
10     * @license        http://codeigniter.com/user_guide/license.html
11     * @link        http://codeigniter.com
12     * @since        Version 1.0
13     * @filesource
14     */
15
16    // ------------------------------------------------------------------------
17
18    /**
19     * Form Validation Class
20     *
21     * @package        CodeIgniter
22     * @subpackage    Libraries
23     * @category    Validation
24     * @author        ExpressionEngine Dev Team
25     * @link        http://codeigniter.com/user_guide/libraries/form_validation.html
26     */
27    class CI_Form_validation {
28
29        protected $CI;
30        protected $_field_data           = array();
31        protected $_config_rules        = array();
32        protected $_error_array          = array();
33        protected $_error_messages       = array();
34        protected $_error_prefix        = '<div class="error">'               ;
35        protected $_error_suffix        = '</div>'      ;
36        protected $error_string          = '';
37        protected $_safe_form_data       = FALSE;
38
39        /**
40         * Constructor
41         */
42        public function __construct($rules = array())
43        {
44            $this->   CI =&     get_instance();
45
46            // Validation rules can be stored in a config file.
47            $this->   _config_rules = $rules;
48
49            // Automatically load the form helper
50            $this->   CI->   load->   helper('form');
51
52            // Set the character encoding in MB.
53            if (function_exists('mb_internal_encoding'))
54            {
55                mb_internal_encoding($this->   CI->   config->   item('charset'));
56            }
57
58            log_message('debug', "Form Validation Class Initialized"          );
59        }
60
61        // ------------------------------------------------------------------------
62
63        /**
64         * Set Rules
65         *
66         * This function takes an array of field names and validation
67         * rules as input, validates the info, and stores it
```

```
68          *
69          * @access    public
70          * @param    mixed
71          * @param    string
72          * @return    void
73          */
74         public function set_rules($field, $label = '', $rules = '')
75         {
76             // No reason to set rules if we have no POST data
77             if (count($_POST) == 0)
78             {
79                 return $this;
80             }
81
82             // If an array was passed via the first parameter instead of indidual string
83             // values we cycle through it and recursively call this function.
84             if (is_array($field))
85             {
86                 foreach ($field as $row)
87                 {
88                     // Houston, we have a problem...
89                     if ( ! isset($row['field']) OR ! isset($row['rules']))
90                     {
91                         continue;
92                     }
93
94                     // If the field label wasn't passed we use the field name
95                     $label = ( ! isset($row['label'])) ? $row['field'] : $row['label'];
96
97                     // Here we go!
98                     $this->   set_rules($row['field'], $label, $row['rules']);
99                 }
100                return $this;
101            }
102
103            // No fields? Nothing to do...
104            if ( ! is_string($field) OR  ! is_string($rules) OR $field == '')
105            {
106                return $this;
107            }
108
109            // If the field label wasn't passed we use the field name
110            $label = ($label == '') ? $field : $label;
111
112            // Is the field name an array?  We test for the existence of a bracket "[" in
113            // the field name to determine this.  If it is an array, we break it apart
114            // into its components so that we can fetch the corresponding POST data later
115            if (strpos($field, '[') !== FALSE AND preg_match_all('/\[(.*?)\]/', $field, $matches))
116            {
117                // Note: Due to a bug in current() that affects some versions
118                // of PHP we can not pass function call directly into it
119                $x = explode('[', $field);
120                $indexes[] = current($x);
121
122                for ($i = 0; $i <    count($matches['0']); $i++)
123                {
124                    if ($matches['1'][$i] != '')
125                    {
126                        $indexes[] = $matches['1'][$i];
127                    }
128                }
129
130                $is_array = TRUE;
131            }
132            else
133            {
134                $indexes    = array();
135                $is_array    = FALSE;
136            }
137
138            // Build our master array
139            $this->   _field_data[$field] = array(
140                'field'                =>    $field,
141                'label'                =>    $label,
142                'rules'                =>    $rules,
143                'is_array'            =>    $is_array,
144                'keys'                =>    $indexes,
145                'postdata'            =>    NULL,
146                'error'                =>    ''
147            );
```

```
148
149             return $this;
150         }
151
152         // --------------------------------------------------------------------
153
154         /**
155          * Set Error Message
156          *
157          * Lets users set their own error messages on the fly.  Note:  The key
158          * name has to match the  function name that it corresponds to.
159          *
160          * @access    public
161          * @param    string
162          * @param    string
163          * @return    string
164          */
165         public function set_message($lang, $val = '')
166         {
167             if ( ! is_array($lang))
168             {
169                 $lang = array($lang =>    $val);
170             }
171
172             $this->   _error_messages = array_merge($this->   _error_messages, $lang);
173
174             return $this;
175         }
176
177         // --------------------------------------------------------------------
178
179         /**
180          * Set The Error Delimiter
181          *
182          * Permits a prefix/suffix to be added to each error message
183          *
184          * @access    public
185          * @param    string
186          * @param    string
187          * @return    void
188          */
189         public function set_error_delimiters($prefix = '<p>'      , $suffix = '</p>'      )
190         {
191             $this->   _error_prefix = $prefix;
192             $this->   _error_suffix = $suffix;
193
194             return $this;
195         }
196
197         // --------------------------------------------------------------------
198
199         /**
200          * Get Error Message
201          *
202          * Gets the error message associated with a particular field
203          *
204          * @access    public
205          * @param    string    the field name
206          * @return    void
207          */
208         public function error($field = '', $prefix = '', $suffix = '')
209         {
210             if ( ! isset($this->   _field_data[$field]['error']) OR $this-
>   _field_data[$field]['error'] == '')
211             {
212                 return '';
213             }
214
215             if ($prefix == '')
216             {
217                 $prefix = $this->   _error_prefix;
218             }
219
220             if ($suffix == '')
221             {
222                 $suffix = $this->   _error_suffix;
223             }
224
225             return $prefix.$this->   _field_data[$field]['error'].$suffix;
226         }
```

```
227
228         // --------------------------------------------------------------------
229
230         /**
231          * Error String
232          *
233          * Returns the error messages as a string, wrapped in the error delimiters
234          *
235          * @access    public
236          * @param    string
237          * @param    string
238          * @return    str
239          */
240         public function error_string($prefix = '', $suffix = '')
241         {
242             // No errrors, validation passes!
243             if (count($this->   _error_array) === 0)
244             {
245                 return '';
246             }
247
248             if ($prefix == '')
249             {
250                 $prefix = $this->   _error_prefix;
251             }
252
253             if ($suffix == '')
254             {
255                 $suffix = $this->   _error_suffix;
256             }
257
258             // Generate the error string
259             $str = '';
260             foreach ($this->   _error_array as $val)
261             {
262                 if ($val != '')
263                 {
264                     $str .= $prefix.$val.$suffix."\n"          ;
265                 }
266             }
267
268             return $str;
269         }
270
271         // --------------------------------------------------------------------
272
273         /**
274          * Run the Validator
275          *
276          * This function does all the work.
277          *
278          * @access    public
279          * @return    bool
280          */
281         public function run($group = '')
282         {
283             // Do we even have any data to process?  Mm?
284             if (count($_POST) == 0)
285             {
286                 return FALSE;
287             }
288
289             // Does the _field_data array containing the validation rules exist?
290             // If not, we look to see if they were assigned via a config file
291             if (count($this->   _field_data) == 0)
292             {
293                 // No validation rules?  We're done...
294                 if (count($this->   _config_rules) == 0)
295                 {
296                     return FALSE;
297                 }
298
299                 // Is there a validation rule for the particular URI being accessed?
300                 $uri = ($group == '') ? trim($this->   CI->   uri->   ruri_string(), '/') : $group;
301
302                 if ($uri != '' AND isset($this->   _config_rules[$uri]))
303                 {
304                     $this->   set_rules($this->   _config_rules[$uri]);
305                 }
306                 else
```

```
307             {
308                 $this->   set_rules($this->   _config_rules);
309             }
310 
311             // We're we able to set the rules correctly?
312             if (count($this->   _field_data) == 0)
313             {
314                 log_message('debug', "Unable to find validation rules"          );
315                 return FALSE;
316             }
317         }
318 
319         // Load the language file containing error messages
320         $this->   CI->   lang->   load('form_validation');
321 
322         // Cycle through the rules for each field, match the
323         // corresponding $_POST item and test for errors
324         foreach ($this->   _field_data as $field =>    $row)
325         {
326             // Fetch the data from the corresponding $_POST array and cache it in the
_field_data array.
327             // Depending on whether the field name is an array or a string will determine where
we get it from.
328 
329             if ($row['is_array'] == TRUE)
330             {
331                 $this->   _field_data[$field]['postdata'] = $this->   _reduce_array($_POST,
$row['keys']);
332             }
333             else
334             {
335                 if (isset($_POST[$field]) AND $_POST[$field] != ""          )
336                 {
337                     $this->   _field_data[$field]['postdata'] = $_POST[$field];
338                 }
339             }
340 
341             $this->   _execute($row, explode('|', $row['rules']), $this-
>   _field_data[$field]['postdata']);
342         }
343 
344         // Did we end up with any errors?
345         $total_errors = count($this->   _error_array);
346 
347         if ($total_errors >    0)
348         {
349             $this->   _safe_form_data = TRUE;
350         }
351 
352         // Now we need to re-set the POST data with the new, processed data
353         $this->   _reset_post_array();
354 
355         // No errors, validation passes!
356         if ($total_errors == 0)
357         {
358             return TRUE;
359         }
360 
361         // Validation fails
362         return FALSE;
363     }
364 
365     // --------------------------------------------------------------------
366 
367     /**
368      * Traverse a multidimensional $_POST array index until the data is found
369      *
370      * @access    private
371      * @param    array
372      * @param    array
373      * @param    integer
374      * @return    mixed
375      */
376     protected function _reduce_array($array, $keys, $i = 0)
377     {
378         if (is_array($array))
379         {
380             if (isset($keys[$i]))
381             {
382                 if (isset($array[$keys[$i]]))
```

```
383                 {
384                     $array = $this->   _reduce_array($array[$keys[$i]], $keys, ($i+1));
385                 }
386                 else
387                 {
388                     return NULL;
389                 }
390             }
391             else
392             {
393                 return $array;
394             }
395         }
396 
397         return $array;
398     }
399 
400     // --------------------------------------------------------------------
401 
402     /**
403      * Re-populate the _POST array with our finalized and processed data
404      *
405      * @access    private
406      * @return    null
407      */
408     protected function _reset_post_array()
409     {
410         foreach ($this->   _field_data as $field =>    $row)
411         {
412             if ( ! is_null($row['postdata']))
413             {
414                 if ($row['is_array'] == FALSE)
415                 {
416                     if (isset($_POST[$row['field']]))
417                     {
418                         $_POST[$row['field']] = $this->   prep_for_form($row['postdata']);
419                     }
420                 }
421                 else
422                 {
423                     // start with a reference
424                     $post_ref =&     $_POST;
425 
426                     // before we assign values, make a reference to the right POST key
427                     if (count($row['keys']) == 1)
428                     {
429                         $post_ref =&     $post_ref[current($row['keys'])];
430                     }
431                     else
432                     {
433                         foreach ($row['keys'] as $val)
434                         {
435                             $post_ref =&     $post_ref[$val];
436                         }
437                     }
438 
439                     if (is_array($row['postdata']))
440                     {
441                         $array = array();
442                         foreach ($row['postdata'] as $k =>    $v)
443                         {
444                             $array[$k] = $this->   prep_for_form($v);
445                         }
446 
447                         $post_ref = $array;
448                     }
449                     else
450                     {
451                         $post_ref = $this->   prep_for_form($row['postdata']);
452                     }
453                 }
454             }
455         }
456     }
457 
458     // --------------------------------------------------------------------
459 
460     /**
461      * Executes the Validation routines
462      *
```

```
463         * @access    private
464         * @param    array
465         * @param    array
466         * @param    mixed
467         * @param    integer
468         * @return    mixed
469         */
470        protected function _execute($row, $rules, $postdata = NULL, $cycles = 0)
471        {
472            // If the $_POST data is an array we will run a recursive call
473            if (is_array($postdata))
474            {
475                foreach ($postdata as $key =>    $val)
476                {
477                    $this->   _execute($row, $rules, $val, $cycles);
478                    $cycles++;
479                }
480
481                return;
482            }
483
484            // ----------------------------------------------------------------
485
486            // If the field is blank, but NOT required, no further tests are necessary
487            $callback = FALSE;
488            if ( ! in_array('required', $rules) AND is_null($postdata))
489            {
490                // Before we bail out, does the rule contain a callback?
491                if (preg_match("/(callback_\w+(\[.*?\])?)/"          , implode(' ', $rules),
$match))
492                {
493                    $callback = TRUE;
494                    $rules = (array('1' =>    $match[1]));
495                }
496                else
497                {
498                    return;
499                }
500            }
501
502            // ----------------------------------------------------------------
503
504            // Isset Test. Typically this rule will only apply to checkboxes.
505            if (is_null($postdata) AND $callback == FALSE)
506            {
507                if (in_array('isset', $rules, TRUE) OR in_array('required', $rules))
508                {
509                    // Set the message type
510                    $type = (in_array('required', $rules)) ? 'required' : 'isset';
511
512                    if ( ! isset($this->   _error_messages[$type]))
513                    {
514                        if (FALSE === ($line = $this->   CI->   lang->   line($type)))
515                        {
516                            $line = 'The field was not set';
517                        }
518                    }
519                    else
520                    {
521                        $line = $this->   _error_messages[$type];
522                    }
523
524                    // Build the error message
525                    $message = sprintf($line, $this->   _translate_fieldname($row['label']));
526
527                    // Save the error message
528                    $this->   _field_data[$row['field']]['error'] = $message;
529
530                    if ( ! isset($this->   _error_array[$row['field']]))
531                    {
532                        $this->   _error_array[$row['field']] = $message;
533                    }
534                }
535
536                return;
537            }
538
539            // ----------------------------------------------------------------
540
541            // Cycle through each rule and run it
```

```
542             foreach ($rules As $rule)
543             {
544                 $_in_array = FALSE;
545
546                 // We set the $postdata variable with the current data in our master array so that
547                 // each cycle of the loop is dealing with the processed data from the last cycle
548                 if ($row['is_array'] == TRUE AND is_array($this-
>   _field_data[$row['field']]['postdata']))
549                 {
550                     // We shouldn't need this safety, but just in case there isn't an array index
551                     // associated with this cycle we'll bail out
552                     if ( ! isset($this->   _field_data[$row['field']]['postdata'][$cycles]))
553                     {
554                         continue;
555                     }
556
557                     $postdata = $this->   _field_data[$row['field']]['postdata'][$cycles];
558                     $_in_array = TRUE;
559                 }
560                 else
561                 {
562                     $postdata = $this->   _field_data[$row['field']]['postdata'];
563                 }
564
565                 // ----------------------------------------------------------------------
566
567                 // Is the rule a callback?
568                 $callback = FALSE;
569                 if (substr($rule, 0, 9) == 'callback_')
570                 {
571                     $rule = substr($rule, 9);
572                     $callback = TRUE;
573                 }
574
575                 // Strip the parameter (if exists) from the rule
576                 // Rules can contain a parameter: max_length[5]
577                 $param = FALSE;
578                 if (preg_match("/(.*?)\[(.*)\]/"          , $rule, $match))
579                 {
580                     $rule    = $match[1];
581                     $param    = $match[2];
582                 }
583
584                 // Call the function that corresponds to the rule
585                 if ($callback === TRUE)
586                 {
587                     if ( ! method_exists($this->   CI, $rule))
588                     {
589                         continue;
590                     }
591
592                     // Run the function and grab the result
593                     $result = $this->   CI->   $rule($postdata, $param);
594
595                     // Re-assign the result to the master data array
596                     if ($_in_array == TRUE)
597                     {
598                         $this->   _field_data[$row['field']]['postdata'][$cycles] =
(is_bool($result)) ? $postdata : $result;
599                     }
600                     else
601                     {
602                         $this->   _field_data[$row['field']]['postdata'] = (is_bool($result)) ?
$postdata : $result;
603                     }
604
605                     // If the field isn't required and we just processed a callback we'll move on...
606                     if ( ! in_array('required', $rules, TRUE) AND $result !== FALSE)
607                     {
608                         continue;
609                     }
610                 }
611                 else
612                 {
613                     if ( ! method_exists($this, $rule))
614                     {
615                         // If our own wrapper function doesn't exist we see if a native PHP
function does.
616                         // Users can use any native PHP function call that has one param.
617                         if (function_exists($rule))
```

```
618                         {
619                             $result = $rule($postdata);
620 
621                             if ($_in_array == TRUE)
622                             {
623                                 $this->   _field_data[$row['field']]['postdata'][$cycles] =
(is_bool($result)) ? $postdata : $result;
624                             }
625                             else
626                             {
627                                 $this->   _field_data[$row['field']]['postdata'] =
(is_bool($result)) ? $postdata : $result;
628                             }
629                         }
630                         else
631                         {
632                             log_message('debug', "Unable to find validation rule:
"     .$rule);
633                         }
634 
635                         continue;
636                     }
637 
638                     $result = $this->   $rule($postdata, $param);
639 
640                     if ($_in_array == TRUE)
641                     {
642                         $this->   _field_data[$row['field']]['postdata'][$cycles] =
(is_bool($result)) ? $postdata : $result;
643                     }
644                     else
645                     {
646                         $this->   _field_data[$row['field']]['postdata'] = (is_bool($result)) ?
$postdata : $result;
647                     }
648                 }
649 
650                 // Did the rule test negatively?  If so, grab the error.
651                 if ($result === FALSE)
652                 {
653                     if ( ! isset($this->   _error_messages[$rule]))
654                     {
655                         if (FALSE === ($line = $this->   CI->   lang->   line($rule)))
656                         {
657                             $line = 'Unable to access an error message corresponding to your field
name.';
658                         }
659                     }
660                     else
661                     {
662                         $line = $this->   _error_messages[$rule];
663                     }
664 
665                     // Is the parameter we are inserting into the error message the name
666                     // of another field?  If so we need to grab its "field label"
667                     if (isset($this->   _field_data[$param]) AND isset($this-
>   _field_data[$param]['label']))
668                     {
669                         $param = $this->   _translate_fieldname($this-
>   _field_data[$param]['label']);
670                     }
671 
672                     // Build the error message
673                     $message = sprintf($line, $this->   _translate_fieldname($row['label']),
$param);
674 
675                     // Save the error message
676                     $this->   _field_data[$row['field']]['error'] = $message;
677 
678                     if ( ! isset($this->   _error_array[$row['field']]))
679                     {
680                         $this->   _error_array[$row['field']] = $message;
681                     }
682 
683                     return;
684                 }
685             }
686         }
687 
688         // --------------------------------------------------------------------
```

```
689
690         /**
691          * Translate a field name
692          *
693          * @access    private
694          * @param    string    the field name
695          * @return    string
696          */
697         protected function _translate_fieldname($fieldname)
698         {
699             // Do we need to translate the field name?
700             // We look for the prefix lang: to determine this
701             if (substr($fieldname, 0, 5) == 'lang:')
702             {
703                 // Grab the variable
704                 $line = substr($fieldname, 5);
705
706                 // Were we able to translate the field name?  If not we use $line
707                 if (FALSE === ($fieldname = $this->   CI->   lang->   line($line)))
708                 {
709                     return $line;
710                 }
711             }
712
713             return $fieldname;
714         }
715
716         // --------------------------------------------------------------------
717
718         /**
719          * Get the value from a form
720          *
721          * Permits you to repopulate a form field with the value it was submitted
722          * with, or, if that value doesn't exist, with the default
723          *
724          * @access    public
725          * @param    string    the field name
726          * @param    string
727          * @return    void
728          */
729         public function set_value($field = '', $default = '')
730         {
731             if ( ! isset($this->   _field_data[$field]))
732             {
733                 return $default;
734             }
735
736             // If the data is an array output them one at a time.
737             //     E.g: form_input('name[]', set_value('name[]');
738             if (is_array($this->   _field_data[$field]['postdata']))
739             {
740                 return array_shift($this->   _field_data[$field]['postdata']);
741             }
742
743             return $this->   _field_data[$field]['postdata'];
744         }
745
746         // --------------------------------------------------------------------
747
748         /**
749          * Set Select
750          *
751          * Enables pull-down lists to be set to the value the user
752          * selected in the event of an error
753          *
754          * @access    public
755          * @param    string
756          * @param    string
757          * @return    string
758          */
759         public function set_select($field = '', $value = '', $default = FALSE)
760         {
761             if ( ! isset($this->   _field_data[$field]) OR ! isset($this-
>   _field_data[$field]['postdata']))
762             {
763                 if ($default === TRUE AND count($this->   _field_data) === 0)
764                 {
765                     return ' selected="selected"'          ;
766                 }
767                 return '';
```

```
768           }
769
770           $field = $this->   _field_data[$field]['postdata'];
771
772           if (is_array($field))
773           {
774               if ( ! in_array($value, $field))
775               {
776                   return '';
777               }
778           }
779           else
780           {
781               if (($field == '' OR $value == '') OR ($field != $value))
782               {
783                   return '';
784               }
785           }
786
787           return ' selected="selected"'          ;
788       }
789
790       // ------------------------------------------------------------------
791
792       /**
793        * Set Radio
794        *
795        * Enables radio buttons to be set to the value the user
796        * selected in the event of an error
797        *
798        * @access    public
799        * @param    string
800        * @param    string
801        * @return    string
802        */
803       public function set_radio($field = '', $value = '', $default = FALSE)
804       {
805           if ( ! isset($this->   _field_data[$field]) OR ! isset($this-
>   _field_data[$field]['postdata']))
806           {
807               if ($default === TRUE AND count($this->   _field_data) === 0)
808               {
809                   return ' checked="checked"'          ;
810               }
811               return '';
812           }
813
814           $field = $this->   _field_data[$field]['postdata'];
815
816           if (is_array($field))
817           {
818               if ( ! in_array($value, $field))
819               {
820                   return '';
821               }
822           }
823           else
824           {
825               if (($field == '' OR $value == '') OR ($field != $value))
826               {
827                   return '';
828               }
829           }
830
831           return ' checked="checked"'          ;
832       }
833
834       // ------------------------------------------------------------------
835
836       /**
837        * Set Checkbox
838        *
839        * Enables checkboxes to be set to the value the user
840        * selected in the event of an error
841        *
842        * @access    public
843        * @param    string
844        * @param    string
845        * @return    string
846        */
```

```
847       public function set_checkbox($field = '', $value = '', $default = FALSE)
848       {
849           if ( ! isset($this->   _field_data[$field]) OR ! isset($this-
>   _field_data[$field]['postdata']))
850           {
851               if ($default === TRUE AND count($this->   _field_data) === 0)
852               {
853                   return ' checked="checked"'          ;
854               }
855               return '';
856           }
857
858           $field = $this->   _field_data[$field]['postdata'];
859
860           if (is_array($field))
861           {
862               if ( ! in_array($value, $field))
863               {
864                   return '';
865               }
866           }
867           else
868           {
869               if (($field == '' OR $value == '') OR ($field != $value))
870               {
871                   return '';
872               }
873           }
874
875           return ' checked="checked"'          ;
876       }
877
878       // ------------------------------------------------------------------
879
880       /**
881        * Required
882        *
883        * @access    public
884        * @param    string
885        * @return    bool
886        */
887       public function required($str)
888       {
889           if ( ! is_array($str))
890           {
891               return (trim($str) == '') ? FALSE : TRUE;
892           }
893           else
894           {
895               return ( ! empty($str));
896           }
897       }
898
899       // ------------------------------------------------------------------
900
901       /**
902        * Performs a Regular Expression match test.
903        *
904        * @access    public
905        * @param    string
906        * @param    regex
907        * @return    bool
908        */
909       public function regex_match($str, $regex)
910       {
911           if ( ! preg_match($regex, $str))
912           {
913               return FALSE;
914           }
915
916           return  TRUE;
917       }
918
919       // ------------------------------------------------------------------
920
921       /**
922        * Match one field to another
923        *
924        * @access    public
925        * @param    string
```

```
926         * @param    field
927         * @return    bool
928         */
929        public function matches($str, $field)
930        {
931            if ( ! isset($_POST[$field]))
932            {
933                return FALSE;
934            }
935
936            $field = $_POST[$field];
937
938            return ($str !== $field) ? FALSE : TRUE;
939        }
940
941        // --------------------------------------------------------------------
942
943        /**
944         * Match one field to another
945         *
946         * @access    public
947         * @param    string
948         * @param    field
949         * @return    bool
950         */
951        public function is_unique($str, $field)
952        {
953            list($table, $field)=explode('.', $field);
954            $query = $this->   CI->   db->   limit(1)->   get_where($table, array($field =>
$str));
955
956            return $query->   num_rows() === 0;
957        }
958
959        // --------------------------------------------------------------------
960
961        /**
962         * Minimum Length
963         *
964         * @access    public
965         * @param    string
966         * @param    value
967         * @return    bool
968         */
969        public function min_length($str, $val)
970        {
971            if (preg_match("/[^0-9]/"          , $val))
972            {
973                return FALSE;
974            }
975
976            if (function_exists('mb_strlen'))
977            {
978                return (mb_strlen($str) <    $val) ? FALSE : TRUE;
979            }
980
981            return (strlen($str) <    $val) ? FALSE : TRUE;
982        }
983
984        // --------------------------------------------------------------------
985
986        /**
987         * Max Length
988         *
989         * @access    public
990         * @param    string
991         * @param    value
992         * @return    bool
993         */
994        public function max_length($str, $val)
995        {
996            if (preg_match("/[^0-9]/"          , $val))
997            {
998                return FALSE;
999            }
1000
1001            if (function_exists('mb_strlen'))
1002            {
1003                return (mb_strlen($str) >    $val) ? FALSE : TRUE;
1004            }
```

```
1005
1006          return (strlen($str) >    $val) ? FALSE : TRUE;
1007      }
1008
1009      // --------------------------------------------------------------------
1010
1011      /**
1012       * Exact Length
1013       *
1014       * @access    public
1015       * @param    string
1016       * @param    value
1017       * @return    bool
1018       */
1019      public function exact_length($str, $val)
1020      {
1021          if (preg_match("/[^0-9]/"          , $val))
1022          {
1023              return FALSE;
1024          }
1025
1026          if (function_exists('mb_strlen'))
1027          {
1028              return (mb_strlen($str) != $val) ? FALSE : TRUE;
1029          }
1030
1031          return (strlen($str) != $val) ? FALSE : TRUE;
1032      }
1033
1034      // --------------------------------------------------------------------
1035
1036      /**
1037       * Valid Email
1038       *
1039       * @access    public
1040       * @param    string
1041       * @return    bool
1042       */
1043      public function valid_email($str)
1044      {
1045          return ( ! preg_match("/^([a-z0-9\+_\-]+)(\.[a-z0-9\+_\-]+)*@([a-z0-9\-]+\.)+[a-
z]{2,6}$/ix"      , $str)) ? FALSE : TRUE;
1046      }
1047
1048      // --------------------------------------------------------------------
1049
1050      /**
1051       * Valid Emails
1052       *
1053       * @access    public
1054       * @param    string
1055       * @return    bool
1056       */
1057      public function valid_emails($str)
1058      {
1059          if (strpos($str, ',') === FALSE)
1060          {
1061              return $this->   valid_email(trim($str));
1062          }
1063
1064          foreach (explode(',', $str) as $email)
1065          {
1066              if (trim($email) != '' &&         $this->   valid_email(trim($email)) === FALSE)
1067              {
1068                  return FALSE;
1069              }
1070          }
1071
1072          return TRUE;
1073      }
1074
1075      // --------------------------------------------------------------------
1076
1077      /**
1078       * Validate IP Address
1079       *
1080       * @access    public
1081       * @param    string
1082       * @param    string "ipv4" or "ipv6" to validate a specific ip format
1083       * @return    string
```

```
1084        */
1085       public function valid_ip($ip, $which = '')
1086       {
1087           return $this->   CI->   input->   valid_ip($ip, $which);
1088       }
1089
1090       // --------------------------------------------------------------------
1091
1092       /**
1093        * Alpha
1094        *
1095        * @access    public
1096        * @param    string
1097        * @return    bool
1098        */
1099       public function alpha($str)
1100       {
1101           return ( ! preg_match("/^([a-z])+$/i"          , $str)) ? FALSE : TRUE;
1102       }
1103
1104       // --------------------------------------------------------------------
1105
1106       /**
1107        * Alpha-numeric
1108        *
1109        * @access    public
1110        * @param    string
1111        * @return    bool
1112        */
1113       public function alpha_numeric($str)
1114       {
1115           return ( ! preg_match("/^([a-z0-9])+$/i"          , $str)) ? FALSE : TRUE;
1116       }
1117
1118       // --------------------------------------------------------------------
1119
1120       /**
1121        * Alpha-numeric with underscores and dashes
1122        *
1123        * @access    public
1124        * @param    string
1125        * @return    bool
1126        */
1127       public function alpha_dash($str)
1128       {
1129           return ( ! preg_match("/^([-a-z0-9_-])+$/i"          , $str)) ? FALSE : TRUE;
1130       }
1131
1132       // --------------------------------------------------------------------
1133
1134       /**
1135        * Numeric
1136        *
1137        * @access    public
1138        * @param    string
1139        * @return    bool
1140        */
1141       public function numeric($str)
1142       {
1143           return (bool)preg_match( '/^[\-+]?[0-9]*\.?[0-9]+$/', $str);
1144
1145       }
1146
1147       // --------------------------------------------------------------------
1148
1149       /**
1150        * Is Numeric
1151        *
1152        * @access    public
1153        * @param    string
1154        * @return    bool
1155        */
1156       public function is_numeric($str)
1157       {
1158           return ( ! is_numeric($str)) ? FALSE : TRUE;
1159       }
1160
1161       // --------------------------------------------------------------------
1162
1163       /**
```

```
1164        * Integer
1165        *
1166        * @access    public
1167        * @param    string
1168        * @return    bool
1169        */
1170       public function integer($str)
1171       {
1172           return (bool) preg_match('/^[\-+]?[0-9]+$/', $str);
1173       }
1174
1175       // --------------------------------------------------------------------
1176
1177       /**
1178        * Decimal number
1179        *
1180        * @access    public
1181        * @param    string
1182        * @return    bool
1183        */
1184       public function decimal($str)
1185       {
1186           return (bool) preg_match('/^[\-+]?[0-9]+\.[0-9]+$/', $str);
1187       }
1188
1189       // --------------------------------------------------------------------
1190
1191       /**
1192        * Greather than
1193        *
1194        * @access    public
1195        * @param    string
1196        * @return    bool
1197        */
1198       public function greater_than($str, $min)
1199       {
1200           if ( ! is_numeric($str))
1201           {
1202               return FALSE;
1203           }
1204           return $str >    $min;
1205       }
1206
1207       // --------------------------------------------------------------------
1208
1209       /**
1210        * Less than
1211        *
1212        * @access    public
1213        * @param    string
1214        * @return    bool
1215        */
1216       public function less_than($str, $max)
1217       {
1218           if ( ! is_numeric($str))
1219           {
1220               return FALSE;
1221           }
1222           return $str <    $max;
1223       }
1224
1225       // --------------------------------------------------------------------
1226
1227       /**
1228        * Is a Natural number  (0,1,2,3, etc.)
1229        *
1230        * @access    public
1231        * @param    string
1232        * @return    bool
1233        */
1234       public function is_natural($str)
1235       {
1236           return (bool) preg_match( '/^[0-9]+$/', $str);
1237       }
1238
1239       // --------------------------------------------------------------------
1240
1241       /**
1242        * Is a Natural number, but not a zero  (1,2,3, etc.)
1243        *
```

```
1244        * @access    public
1245        * @param    string
1246        * @return    bool
1247        */
1248       public function is_natural_no_zero($str)
1249       {
1250           if ( ! preg_match( '/^[0-9]+$/', $str))
1251           {
1252               return FALSE;
1253           }
1254
1255           if ($str == 0)
1256           {
1257               return FALSE;
1258           }
1259
1260           return TRUE;
1261       }
1262
1263       // --------------------------------------------------------------------
1264
1265       /**
1266        * Valid Base64
1267        *
1268        * Tests a string for characters outside of the Base64 alphabet
1269        * as defined by RFC 2045 http://www.faqs.org/rfcs/rfc2045
1270        *
1271        * @access    public
1272        * @param    string
1273        * @return    bool
1274        */
1275       public function valid_base64($str)
1276       {
1277           return (bool) ! preg_match('/[^a-zA-Z0-9\/\+=]/', $str);
1278       }
1279
1280       // --------------------------------------------------------------------
1281
1282       /**
1283        * Prep data for form
1284        *
1285        * This function allows HTML to be safely shown in a form.
1286        * Special characters are converted.
1287        *
1288        * @access    public
1289        * @param    string
1290        * @return    string
1291        */
1292       public function prep_for_form($data = '')
1293       {
1294           if (is_array($data))
1295           {
1296               foreach ($data as $key =>    $val)
1297               {
1298                   $data[$key] = $this->   prep_for_form($val);
1299               }
1300
1301               return $data;
1302           }
1303
1304           if ($this->   _safe_form_data == FALSE OR $data === '')
1305           {
1306               return $data;
1307           }
1308
1309           return str_replace(array("'"          , '"'     , '<'    , '>'    ),
1310   array("&#39;"              , "&quot;"             , '&lt;'    , '&gt;'    ), stripslashes($data));
1311       }
1312
1313       // --------------------------------------------------------------------
1314
1315       /**
1316        * Prep URL
1317        *
1318        * @access    public
1319        * @param    string
1320        * @return    string
1321        */
1322       public function prep_url($str = '')
1323       {
```

```
1323          if ($str == 'http://' OR $str == '')
1324          {
1325              return '';
1326          }
1327
1328          if (substr($str, 0, 7) != 'http://' &&         substr($str, 0, 8) != 'https://')
1329          {
1330              $str = 'http://'.$str;
1331          }
1332
1333          return $str;
1334      }
1335
1336      // --------------------------------------------------------------------
1337
1338      /**
1339       * Strip Image Tags
1340       *
1341       * @access    public
1342       * @param    string
1343       * @return    string
1344       */
1345      public function strip_image_tags($str)
1346      {
1347          return $this->   CI->   input->   strip_image_tags($str);
1348      }
1349
1350      // --------------------------------------------------------------------
1351
1352      /**
1353       * XSS Clean
1354       *
1355       * @access    public
1356       * @param    string
1357       * @return    string
1358       */
1359      public function xss_clean($str)
1360      {
1361          return $this->   CI->   security->   xss_clean($str);
1362      }
1363
1364      // --------------------------------------------------------------------
1365
1366      /**
1367       * Convert PHP tags to entities
1368       *
1369       * @access    public
1370       * @param    string
1371       * @return    string
1372       */
1373      public function encode_php_tags($str)
1374      {
1375          return str_replace(array('<?php'    , '<?PHP'    , '<?'    , '?>'    ),
1376  array('&lt;?php'    , '&lt;?PHP'    , '&lt;?'    , '?&gt;'    ), $str);
1377      }
1378
1379  }
1380  // END Form Validation Class
1381
1382  /* End of file Form_validation.php */
1383  /* Location: ./system/libraries/Form_validation.php */
```

Archivo fuente para Pagination.php

La documentación para este archivo está disponible en Pagination.php

```
1     <?php     if ( ! defined('BASEPATH')) exit('No direct script access allowed');
2     /**
3      * CodeIgniter
4      *
5      * An open source application development framework for PHP 5.1.6 or newer
6      *
7      * @package        CodeIgniter
8      * @author        ExpressionEngine Dev Team
9      * @copyright    Copyright (c) 2008 - 2011, EllisLab, Inc.
10     * @license        http://codeigniter.com/user_guide/license.html
11     * @link        http://codeigniter.com
12     * @since        Version 1.0
13     * @filesource
14     */
15
16    // ------------------------------------------------------------------------
17
18    /**
19     * Pagination Class
20     *
21     * @package        CodeIgniter
22     * @subpackage    Libraries
23     * @category    Pagination
24     * @author        ExpressionEngine Dev Team
25     * @link        http://codeigniter.com/user_guide/libraries/pagination.html
26     */
27    class CI_Pagination {
28
29        var $base_url            = ''; // The page we are linking to
30        var $prefix                = ''; // A custom prefix added to the path.
31        var $suffix                = ''; // A custom suffix added to the path.
32
33        var $total_rows           =  0; // Total number of items (database results)
34        var $per_page            = 10; // Max number of items you want shown per page
35        var $num_links            =  2; // Number of "digit" links to show before/after
the currently viewed page
36        var $cur_page            =  0; // The current page being viewed
37        var $use_page_numbers    = FALSE; // Use page number for segment instead of offset
38        var $first_link           = '&lsaquo; First'    ;
39        var $next_link            = '&gt;'    ;
40        var $prev_link            = '&lt;'    ;
41        var $last_link            = 'Last &rsaquo;'    ;
42        var $uri_segment        = 3;
43        var $full_tag_open        = '<ul class="pagination pagination-sm">'                ;
44        var $full_tag_close        = '<ul>'      ;
45        var $first_tag_open        = '<li>'      ;
46        var $first_tag_close    = '</li>'      ;
47        var $last_tag_open        = '<li>'      ;
48        var $last_tag_close        = '</li>'      ;
49        var $first_url            = ''; // Alternative URL for the First Page.
50        var $cur_tag_open        = '<li><a><strong>'                ;
51        var $cur_tag_close        = '</strong></a></li>'                ;
52        var $next_tag_open        = '<li>'      ;
53        var $next_tag_close        = '</li>'      ;
54        var $prev_tag_open        = '<li>'      ;
55        var $prev_tag_close        = '</li>'      ;
56        var $num_tag_open        = '<li>'      ;
57        var $num_tag_close        = '</li>'      ;
58        var $page_query_string    = FALSE;
59        var $query_string_segment = 'per_page';
60        var $display_pages        = TRUE;
61        var $anchor_class        = '';
62
63        /**
64         * Constructor
65         *
66         * @access    public
```

```
67          * @param    array    initialization parameters
68          */
69         public function __construct($params = array())
70         {
71             if (count($params) >    0)
72             {
73                 $this->   initialize($params);
74             }
75
76             if ($this->   anchor_class != '')
77             {
78                 $this->   anchor_class = 'class="'     .$this->   anchor_class.'" '     ;
79             }
80
81             log_message('debug', "Pagination Class Initialized"          );
82         }
83
84         // ------------------------------------------------------------------
85
86         /**
87          * Initialize Preferences
88          *
89          * @access    public
90          * @param    array    initialization parameters
91          * @return    void
92          */
93         function initialize($params = array())
94         {
95             if (count($params) >    0)
96             {
97                 foreach ($params as $key =>    $val)
98                 {
99                     if (isset($this->   $key))
100                    {
101                        $this->   $key = $val;
102                    }
103                }
104            }
105        }
106
107        // ------------------------------------------------------------------
108
109        /**
110         * Generate the pagination links
111         *
112         * @access    public
113         * @return    string
114         */
115        function create_links()
116        {
117            // If our item count or per-page total is zero there is no need to continue.
118            if ($this->   total_rows == 0 OR $this->   per_page == 0)
119            {
120                return '';
121            }
122
123            // Calculate the total number of pages
124            $num_pages = ceil($this->   total_rows / $this->   per_page);
125
126            // Is there only one page? Hm... nothing more to do here then.
127            if ($num_pages == 1)
128            {
129                return '';
130            }
131
132            // Set the base page index for starting page number
133            if ($this->   use_page_numbers)
134            {
135                $base_page = 1;
136            }
137            else
138            {
139                $base_page = 0;
140            }
141
142            // Determine the current page number.
143            $CI =&     get_instance();
144
145            if ($CI->   config->   item('enable_query_strings') === TRUE OR $this-
>   page_query_string === TRUE)
```

```
146             {
147                 if ($CI->   input->   get($this->   query_string_segment) != $base_page)
148                 {
149                     $this->   cur_page = $CI->   input->   get($this->   query_string_segment);
150 
151                     // Prep the current page - no funny business!
152                     $this->   cur_page = (int) $this->   cur_page;
153                 }
154             }
155             else
156             {
157                 if ($CI->   uri->   segment($this->   uri_segment) != $base_page)
158                 {
159                     $this->   cur_page = $CI->   uri->   segment($this->   uri_segment);
160 
161                     // Prep the current page - no funny business!
162                     $this->   cur_page = (int) $this->   cur_page;
163                 }
164             }
165 
166             // Set current page to 1 if using page numbers instead of offset
167             if ($this->   use_page_numbers AND $this->   cur_page == 0)
168             {
169                 $this->   cur_page = $base_page;
170             }
171 
172             $this->   num_links = (int)$this->   num_links;
173 
174             if ($this->   num_links <    1)
175             {
176                 show_error('Your number of links must be a positive number.');
177             }
178 
179             if ( ! is_numeric($this->   cur_page))
180             {
181                 $this->   cur_page = $base_page;
182             }
183 
184             // Is the page number beyond the result range?
185             // If so we show the last page
186             if ($this->   use_page_numbers)
187             {
188                 if ($this->   cur_page >    $num_pages)
189                 {
190                     $this->   cur_page = $num_pages;
191                 }
192             }
193             else
194             {
195                 if ($this->   cur_page >    $this->   total_rows)
196                 {
197                     $this->   cur_page = ($num_pages - 1) * $this->   per_page;
198                 }
199             }
200 
201             $uri_page_number = $this->   cur_page;
202 
203             if ( ! $this->   use_page_numbers)
204             {
205                 $this->   cur_page = floor(($this->   cur_page/$this->   per_page) + 1);
206             }
207 
208             // Calculate the start and end numbers. These determine
209             // which number to start and end the digit links with
210             $start = (($this->   cur_page - $this->   num_links) >    0) ? $this->   cur_page -
($this->   num_links - 1) : 1;
211             $end   = (($this->   cur_page + $this->   num_links) <    $num_pages) ? $this-
>   cur_page + $this->   num_links : $num_pages;
212 
213             // Is pagination being used over GET or POST?  If get, add a per_page query
214             // string. If post, add a trailing slash to the base URL if needed
215             if ($CI->   config->   item('enable_query_strings') === TRUE OR $this-
>   page_query_string === TRUE)
216             {
217                 $this->   base_url = rtrim($this->   base_url).'&amp;'    .$this-
>   query_string_segment.'=';
218             }
219             else
220             {
221                 $this->   base_url = rtrim($this->   base_url, '/') .'/';
```

```
222             }
223
224             // And here we go...
225             $output = '';
226
227             // Render the "First" link
228             if  ($this->   first_link !== FALSE AND $this->   cur_page > (   $this->   num_links +
1))
229             {
230                 $first_url = ($this->   first_url == '') ? $this->   base_url : $this->   first_url;
231                 $output .= $this->   first_tag_open.'<a '    .$this-
>   anchor_class.'href="'     .$first_url.'">'        .$this->   first_link.'</a>'      .$this-
>   first_tag_close;
232             }
233
234             // Render the "previous" link
235             if  ($this->   prev_link !== FALSE AND $this->   cur_page != 1)
236             {
237                 if ($this->   use_page_numbers)
238                 {
239                     $i = $uri_page_number - 1;
240                 }
241                 else
242                 {
243                     $i = $uri_page_number - $this->   per_page;
244                 }
245
246                 if ($i == 0 &&         $this->   first_url != '')
247                 {
248                     $output .= $this->   prev_tag_open.'<a '    .$this-
>   anchor_class.'href="'     .$this->   first_url.'">'        .$this-
>   prev_link.'</a>'      .$this->   prev_tag_close;
249                 }
250                 else
251                 {
252                     $i = ($i == 0) ? '' : $this->   prefix.$i.$this->   suffix;
253                     $output .= $this->   prev_tag_open.'<a '    .$this-
>   anchor_class.'href="'     .$this->   base_url.$i.'">'        .$this-
>   prev_link.'</a>'      .$this->   prev_tag_close;
254                 }
255
256             }
257
258             // Render the pages
259             if ($this->   display_pages !== FALSE)
260             {
261                 // Write the digit links
262                 for ($loop = $start -1; $loop <=    $end; $loop++)
263                 {
264                     if ($this->   use_page_numbers)
265                     {
266                         $i = $loop;
267                     }
268                     else
269                     {
270                         $i = ($loop * $this->   per_page) - $this->   per_page;
271                     }
272
273                     if ($i >=    $base_page)
274                     {
275                         if ($this->   cur_page == $loop)
276                         {
277                             $output .= $this->   cur_tag_open.$loop.$this->   cur_tag_close; //
Current page
278                         }
279                         else
280                         {
281                             $n = ($i == $base_page) ? '' : $i;
282
283                             if ($n == '' &&         $this->   first_url != '')
284                             {
285                                 $output .= $this->   num_tag_open.'<a '    .$this-
>   anchor_class.'href="'     .$this->   first_url.'">'        .$loop.'</a>'      .$this-
>   num_tag_close;
286                             }
287                             else
288                             {
289                                 $n = ($n == '') ? '' : $this->   prefix.$n.$this->   suffix;
290
291                                 $output .= $this->   num_tag_open.'<a '    .$this-
```

```
>   anchor_class.'href="'      .$this->   base_url.$n.'">'        .$loop.'</a>'       .$this-
>   num_tag_close;
292                         }
293                     }
294                 }
295             }
296         }
297
298         // Render the "next" link
299         if ($this->   next_link !== FALSE AND $this->   cur_page <    $num_pages)
300         {
301             if ($this->   use_page_numbers)
302             {
303                 $i = $this->   cur_page + 1;
304             }
305             else
306             {
307                 $i = ($this->   cur_page * $this->   per_page);
308             }
309
310             $output .= $this->   next_tag_open.'<a '    .$this-
>   anchor_class.'href="'      .$this->   base_url.$this->   prefix.$i.$this-
>   suffix.'">'        .$this->   next_link.'</a>'       .$this->   next_tag_close;
311         }
312
313         // Render the "Last" link
314         if ($this->   last_link !== FALSE AND ($this->   cur_page + $this->   num_links) <
$num_pages)
315         {
316             if ($this->   use_page_numbers)
317             {
318                 $i = $num_pages;
319             }
320             else
321             {
322                 $i = (($num_pages * $this->   per_page) - $this->   per_page);
323             }
324             $output .= $this->   last_tag_open.'<a '    .$this-
>   anchor_class.'href="'      .$this->   base_url.$this->   prefix.$i.$this-
>   suffix.'">'        .$this->   last_link.'</a>'       .$this->   last_tag_close;
325         }
326
327         // Kill double slashes.  Note: Sometimes we can end up with a double slash
328         // in the penultimate link so we'll kill all double slashes.
329         $output = preg_replace("#([^:])//+#"          , "\\1/"          , $output);
330
331         // Add the wrapper HTML if exists
332         $output = $this->   full_tag_open.$output.$this->   num_tag_open.'<span> Total de
registros: '.number_format($this->   total_rows).'</span>'      .$this->   num_tag_close.$this-
>   full_tag_close;
333
334         return $output;
335     }
336   }
337   // END Pagination Class
338
339   /* End of file Pagination.php */
340   /* Location: ./system/libraries/Pagination.php */
```

Archivo fuente para template.php

La documentación para este archivo está disponible en template.php

```
1     <?php     if (!defined('BASEPATH')) exit('No direct script access allowed');
2     /**
3      * CodeIgniter
4      *
5      * An open source application development framework for PHP 4.3.2 or newer
6      *
7      * @package CodeIgniter
8      * @author  ExpressionEngine Dev Team
9      * @copyright  Copyright (c) 2006, EllisLab, Inc.
10     * @license http://codeigniter.com/user_guide/license.html
11     * @link http://codeigniter.com
12     * @since   Version 1.0
13     * @filesource
14     */
15
16    // ------------------------------------------------------------------------
17
18    /**
19     * CodeIgniter Template Class
20     *
21     * This class is and interface to CI's View class. It aims to improve the
22     * interaction between controllers and views. Follow @link for more info
23     *
24     * @package        CodeIgniter
25     * @author        Colin Williams
26     * @subpackage    Libraries
27     * @category    Libraries
28     * @link        http://www.williamsconcepts.com/ci/libraries/template/index.html
29     * @copyright  Copyright (c) 2008, Colin Williams.
30     * @version 1.4.1
31     *
32     */
33    class CI_Template {
34
35       var $CI;
36       var $config;
37       var $template;
38       var $master;
39       var $regions = array(
40          '_scripts' => array(),
41          '_styles' => array(),
42       );
43       var $output;
44       var $js = array();
45       var $css = array();
46       var $parser = 'parser';
47       var $parser_method = 'parse';
48       var $parse_template = FALSE;
49
50       /**
51         * Constructor
52         *
53         * Loads template configuration, template regions, and validates existence of
54         * default template
55         *
56         * @access    public
57         */
58
59       function CI_Template()
60       {
61          // Copy an instance of CI so we can use the entire framework.
62          $this->   CI =&     get_instance();
63
64          // Load the template config file and setup our master template and regions
65          include(APPPATH.'config/template'.EXT);
66          if (isset($template))
67          {
```

```
68              $this->   config = $template;
69              $this->   set_template($template['active_template']);
70           }
71        }
72
73        // --------------------------------------------------------------------
74
75        /**
76         * Use given template settings
77         *
78         * @access  public
79         * @param   string   array key to access template settings
80         * @return  void
81         */
82
83        function set_template($group)
84        {
85           if (isset($this->   config[$group]))
86           {
87              $this->   template = $this->   config[$group];
88           }
89           else
90           {
91              show_error('The "'      . $group .'" template group does not exist. Provide a
valid group name or add the group first.');
92           }
93           $this->   initialize($this->   template);
94        }
95
96           // --------------------------------------------------------------------
97
98        /**
99         * Set master template
100        *
101        * @access  public
102        * @param   string   filename of new master template file
103        * @return  void
104        */
105
106       function set_master_template($filename)
107       {
108          if (file_exists(APPPATH .'views/'. $filename) or file_exists(APPPATH .'views/'. $filename
. EXT))
109          {
110             $this->   master = $filename;
111          }
112          else
113          {
114             show_error('The filename provided does not exist in <strong>'      . APPPATH
.'views</strong>. Remember to include the extension if other than ".php"'               );
115          }
116       }
117
118       // --------------------------------------------------------------------
119
120       /**
121        * Dynamically add a template and optionally switch to it
122        *
123        * @access  public
124        * @param   string   array key to access template settings
125        * @param   array properly formed
126        * @return  void
127        */
128
129       function add_template($group, $template, $activate = FALSE)
130       {
131          if ( ! isset($this->   config[$group]))
132          {
133             $this->   config[$group] = $template;
134             if ($activate === TRUE)
135             {
136                $this->   initialize($template);
137             }
138          }
139          else
140          {
141             show_error('The "'      . $group .'" template group already exists. Use a
different group name.');
142          }
143       }
```

```
144
145       // --------------------------------------------------------------------
146
147       /**
148        * Initialize class settings using config settings
149        *
150        * @access  public
151        * @param   array   configuration array
152        * @return  void
153        */
154
155       function initialize($props)
156       {
157          // Set master template
158          if (isset($props['template'])
159             && (        file_exists(APPPATH .'views/'. $props['template']) or file_exists(APPPATH
.'views/'. $props['template'] . EXT)))
160          {
161             $this->   master = $props['template'];
162          }
163          else
164          {
165             // Master template must exist. Throw error.
166             show_error('Either you have not provided a master template or the one provided does
not exist in <strong>'      . APPPATH .'views</strong>. Remember to include the extension if
other than ".php"'          );
167          }
168
169          // Load our regions
170          if (isset($props['regions']))
171          {
172             $this->   set_regions($props['regions']);
173          }
174
175          // Set parser and parser method
176          if (isset($props['parser']))
177          {
178             $this->   set_parser($props['parser']);
179          }
180          if (isset($props['parser_method']))
181          {
182             $this->   set_parser_method($props['parser_method']);
183          }
184
185          // Set master template parser instructions
186          $this->   parse_template = isset($props['parse_template']) ? $props['parse_template'] :
FALSE;
187       }
188
189       // --------------------------------------------------------------------
190
191       /**
192        * Set regions for writing to
193        *
194        * @access  public
195        * @param   array   properly formed regions array
196        * @return  void
197        */
198
199       function set_regions($regions)
200       {
201          if (count($regions))
202          {
203             $this->   regions = array(
204                '_scripts' => array(),
205                '_styles' => array(),
206             );
207             foreach ($regions as $key =>    $region)
208             {
209                // Regions must be arrays, but we take the burden off the template
210                // developer and insure it here
211                if ( ! is_array($region))
212                {
213                   $this->   add_region($region);
214                }
215                else {
216                   $this->   add_region($key, $region);
217                }
218             }
219          }
```

```
220     }
221
222     // --------------------------------------------------------------------
223
224     /**
225      * Dynamically add region to the currently set template
226      *
227      * @access  public
228      * @param   string   Name to identify the region
229      * @param   array Optional array with region defaults
230      * @return  void
231      */
232
233     function add_region($name, $props = array())
234     {
235        if ( ! is_array($props))
236        {
237           $props = array();
238        }
239
240        if ( ! isset($this->   regions[$name]))
241        {
242           $this->   regions[$name] = $props;
243        }
244        else
245        {
246           show_error('The "'      . $name .'" region has already been defined.'      );
247        }
248     }
249
250     // --------------------------------------------------------------------
251
252     /**
253      * Empty a region's content
254      *
255      * @access  public
256      * @param   string   Name to identify the region
257      * @return  void
258      */
259
260     function empty_region($name)
261     {
262        if (isset($this->   regions[$name]['content']))
263        {
264           $this->   regions[$name]['content'] = array();
265        }
266        else
267        {
268           show_error('The "'      . $name .'" region is undefined.'      );
269        }
270     }
271
272     // --------------------------------------------------------------------
273
274     /**
275      * Set parser
276      *
277      * @access  public
278      * @param   string   name of parser class to load and use for parsing methods
279      * @return  void
280      */
281
282     function set_parser($parser, $method = NULL)
283     {
284        $this->   parser = $parser;
285        $this->   CI->   load->   library($parser);
286
287        if ($method)
288        {
289           $this->   set_parser_method($method);
290        }
291     }
292
293     // --------------------------------------------------------------------
294
295     /**
296      * Set parser method
297      *
298      * @access  public
299      * @param   string   name of parser class member function to call when parsing
```

```
300         * @return  void
301         */
302
303       function set_parser_method($method)
304       {
305          $this->   parser_method = $method;
306       }
307
308       // --------------------------------------------------------------------
309
310       /**
311         * Write contents to a region
312         *
313         * @access    public
314         * @param    string    region to write to
315         * @param    string    what to write
316         * @param    boolean    FALSE to append to region, TRUE to overwrite region
317         * @return    void
318         */
319
320       function write($region, $content, $overwrite = FALSE)
321       {
322          if (isset($this->   regions[$region]))
323          {
324             if ($overwrite === TRUE) // Should we append the content or overwrite it
325             {
326                $this->   regions[$region]['content'] = array($content);
327             } else {
328                $this->   regions[$region]['content'][] = $content;
329             }
330          }
331
332          // Regions MUST be defined
333          else
334          {
335             show_error("     Cannot write to the '{$region}' region. The region is
undefined."     );
336          }
337       }
338
339       // --------------------------------------------------------------------
340
341       /**
342         * Write content from a View to a region. 'Views within views'
343         *
344         * @access    public
345         * @param    string    region to write to
346         * @param    string    view file to use
347         * @param    array    variables to pass into view
348         * @param    boolean    FALSE to append to region, TRUE to overwrite region
349         * @return    void
350         */
351
352       function write_view($region, $view, $data = NULL, $overwrite = FALSE)
353       {
354          $args = func_get_args();
355
356          // Get rid of non-views
357          unset($args[0], $args[2], $args[3]);
358
359          // Do we have more view suggestions?
360          if (count($args) >    1)
361          {
362             foreach ($args as $suggestion)
363             {
364                if (file_exists(APPPATH .'views/'. $suggestion . EXT) or file_exists(APPPATH
.'views/'. $suggestion))
365                {
366                   // Just change the $view arg so the rest of our method works as normal
367                   $view = $suggestion;
368                   break;
369                }
370             }
371          }
372
373          $content = $this->   CI->   load->   view($view, $data, TRUE);
374          $this->   write($region, $content, $overwrite);
375
376       }
377
```

```
378      // --------------------------------------------------------------------
379
380      /**
381       * Parse content from a View to a region with the Parser Class
382       *
383       * @access  public
384       * @param   string   region to write to
385       * @param   string   view file to parse
386       * @param   array variables to pass into view for parsing
387       * @param   boolean  FALSE to append to region, TRUE to overwrite region
388       * @return  void
389       */
390
391      function parse_view($region, $view, $data = NULL, $overwrite = FALSE)
392      {
393         $this->   CI->   load->   library('parser');
394
395         $args = func_get_args();
396
397         // Get rid of non-views
398         unset($args[0], $args[2], $args[3]);
399
400         // Do we have more view suggestions?
401         if (count($args) >    1)
402         {
403            foreach ($args as $suggestion)
404            {
405               if (file_exists(APPPATH .'views/'. $suggestion . EXT) or file_exists(APPPATH
.'views/'. $suggestion))
406               {
407                  // Just change the $view arg so the rest of our method works as normal
408                  $view = $suggestion;
409                  break;
410               }
411            }
412         }
413
414         $content = $this->   CI->{   $this->   parser}->{   $this->   parser_method}($view,
$data, TRUE);
415         $this->   write($region, $content, $overwrite);
416
417      }
418
419      // --------------------------------------------------------------------
420
421      /**
422       * Dynamically include javascript in the template
423       *
424       * NOTE: This function does NOT check for existence of .js file
425       *
426       * @access  public
427       * @param   string   script to import or embed
428       * @param   string  'import' to load external file or 'embed' to add as-is
429       * @param   boolean  TRUE to use 'defer' attribute, FALSE to exclude it
430       * @return  TRUE on success, FALSE otherwise
431       */
432
433      function add_js($script, $type = 'import', $defer = FALSE)
434      {
435         $success = TRUE;
436         $js = NULL;
437
438         $this->   CI->   load->   helper('url');
439
440         switch ($type)
441         {
442            case 'import':
443               $filepath = base_url() . $script;
444               $js = '<script type="text/javascript" src="'                  . $filepath .'"'     ;
445               if ($defer)
446               {
447                  $js .= ' defer="defer"'          ;
448               }
449               $js .= "></script>"                  ;
450               break;
451
452            case 'embed':
453               $js = '<script type="text/javascript"'             ;
454               if ($defer)
455               {
```

```
456                $js .= ' defer="defer"'          ;
457             }
458             $js .= ">"            ;
459             $js .= $script;
460             $js .= '</script>'      ;
461             break;
462
463          default:
464             $success = FALSE;
465             break;
466       }
467
468       // Add to js array if it doesn't already exist
469       if ($js != NULL && !        in_array($js, $this->   js))
470       {
471          $this->   js[] = $js;
472          $this->   write('_scripts', $js);
473       }
474
475       return $success;
476    }
477
478    // ------------------------------------------------------------------------
479
480    /**
481     * Dynamically include CSS in the template
482     *
483     * NOTE: This function does NOT check for existence of .css file
484     *
485     * @access  public
486     * @param   string   CSS file to link, import or embed
487     * @param   string  'link', 'import' or 'embed'
488     * @param   string  media attribute to use with 'link' type only, FALSE for none
489     * @return  TRUE on success, FALSE otherwise
490     */
491
492    function add_css($style, $type = 'link', $media = FALSE)
493    {
494       $success = TRUE;
495       $css = NULL;
496
497       $this->   CI->   load->   helper('url');
498       $filepath = base_url() . $style;
499
500       switch ($type)
501       {
502          case 'link':
503
504             $css = '<link type="text/css" rel="stylesheet" href="'                          .
$filepath .'"'     ;
505             if ($media)
506             {
507                $css .= ' media="'     . $media .'"'     ;
508             }
509             $css .= ' />'   ;
510             break;
511
512          case 'import':
513             $css = '<style type="text/css">@import url('               . $filepath
.');</style>'      ;
514             break;
515
516          case 'embed':
517             $css = '<style type="text/css">'               ;
518             $css .= $style;
519             $css .= '</style>'      ;
520             break;
521
522          default:
523             $success = FALSE;
524             break;
525       }
526
527       // Add to js array if it doesn't already exist
528       if ($css != NULL && !        in_array($css, $this->   css))
529       {
530          $this->   css[] = $css;
531          $this->   write('_styles', $css);
532       }
533
```

```
534         return $success;
535      }
536
537      // --------------------------------------------------------------------
538
539      /**
540        * Render the master template or a single region
541        *
542        * @access    public
543        * @param    string    optionally opt to render a specific region
544        * @param    boolean    FALSE to output the rendered template, TRUE to return as a string.
Always TRUE when $region is supplied
545        * @return    void or string (result of template build)
546        */
547
548      function render($region = NULL, $buffer = FALSE, $parse = FALSE)
549      {
550         // Just render $region if supplied
551         if ($region) // Display a specific regions contents
552         {
553            if (isset($this->   regions[$region]))
554            {
555               $output = $this->   _build_content($this->   regions[$region]);
556            }
557            else
558            {
559               show_error("     Cannot render the '{$region}' region. The region is
undefined."     );
560            }
561         }
562
563         // Build the output array
564         else
565         {
566            foreach ($this->   regions as $name =>    $region)
567            {
568               $this->   output[$name] = $this->   _build_content($region);
569            }
570
571            if ($this->   parse_template === TRUE or $parse === TRUE)
572            {
573               // Use provided parser class and method to render the template
574               $output = $this->   CI->{   $this->   parser}->{   $this->   parser_method}($this-
>   master, $this->   output, TRUE);
575
576               // Parsers never handle output, but we need to mimick it in this case
577               if ($buffer === FALSE)
578               {
579                  $this->   CI->   output->   set_output($output);
580               }
581            }
582            else
583            {
584               // Use CI's loader class to render the template with our output array
585               $output = $this->   CI->   load->   view($this->   master, $this->   output,
$buffer);
586            }
587         }
588
589         return $output;
590      }
591
592      // --------------------------------------------------------------------
593
594      /**
595       * Load the master template or a single region
596       *
597       * DEPRECATED!
598       *
599       * Use render() to compile and display your template and regions
600       */
601
602       function load($region = NULL, $buffer = FALSE)
603       {
604          $region = NULL;
605          $this->   render($region, $buffer);
606       }
607
608      // --------------------------------------------------------------------
609
```

```
610      /**
611        * Build a region from it's contents. Apply wrapper if provided
612        *
613        * @access    private
614        * @param    string    region to build
615        * @param    string    HTML element to wrap regions in; like '<div>'
616        * @param    array    Multidimensional array of HTML elements to apply to $wrapper
617        * @return    string    Output of region contents
618        */
619
620      function _build_content($region, $wrapper = NULL, $attributes = NULL)
621      {
622         $output = NULL;
623
624         // Can't build an empty region. Exit stage left
625         if ( ! isset($region['content']) or ! count($region['content']))
626         {
627            return FALSE;
628         }
629
630         // Possibly overwrite wrapper and attributes
631         if ($wrapper)
632         {
633            $region['wrapper'] = $wrapper;
634         }
635         if ($attributes)
636         {
637            $region['attributes'] = $attributes;
638         }
639
640         // Open the wrapper and add attributes
641         if (isset($region['wrapper']))
642         {
643            // This just trims off the closing angle bracket. Like '<p>' to '<p'
644            $output .= substr($region['wrapper'], 0, strlen($region['wrapper']) - 1);
645
646            // Add HTML attributes
647            if (isset($region['attributes']) &&         is_array($region['attributes']))
648            {
649               foreach ($region['attributes'] as $name =>    $value)
650               {
651                  // We don't validate HTML attributes. Imagine someone using a custom XML
template..
652                  $output .= "      $name=\"     $value\"     "     ;
653               }
654            }
655
656            $output .= ">"             ;
657         }
658
659         // Output the content items.
660         foreach ($region['content'] as $content)
661         {
662            $output .= $content;
663         }
664
665         // Close the wrapper tag
666         if (isset($region['wrapper']))
667         {
668            // This just turns the wrapper into a closing tag. Like '<p>' to '</p>'
669            $output .= str_replace('<'   , '</'   , $region['wrapper']) . "\n"          ;
670         }
671
672         return $output;
673      }
674
675   }
676   // END Template Class
677
678   /* End of file Template.php */
679   /* Location: ./system/application/libraries/Template.php */
```

Indice

A

B

C

E

F

G

H

I

M

N

O

P

R

S

T

U