CRUGE, EXTENSION PARA EL CONTROL DE USUARIOS Y ROLES

Autor:

Christian Salazar. christiansalazarh@gmail.com,

Edición:

julio-oct 2012, Acarigua, Edo. Portuguesa, Venezuela

Contribuyen:

- Ricardo Andrés Obregón (Bogotá Colombia, robregonm@gmail.com)

Enlaces.

Repositorio de Cruge en Bitbucket

Repositorio de un Demo Completo

Mi Blog, con varios temas relacionados.

Yii Framework en Español – Foro Twitter: @yiienespanol

Requisitos de Plataforma:

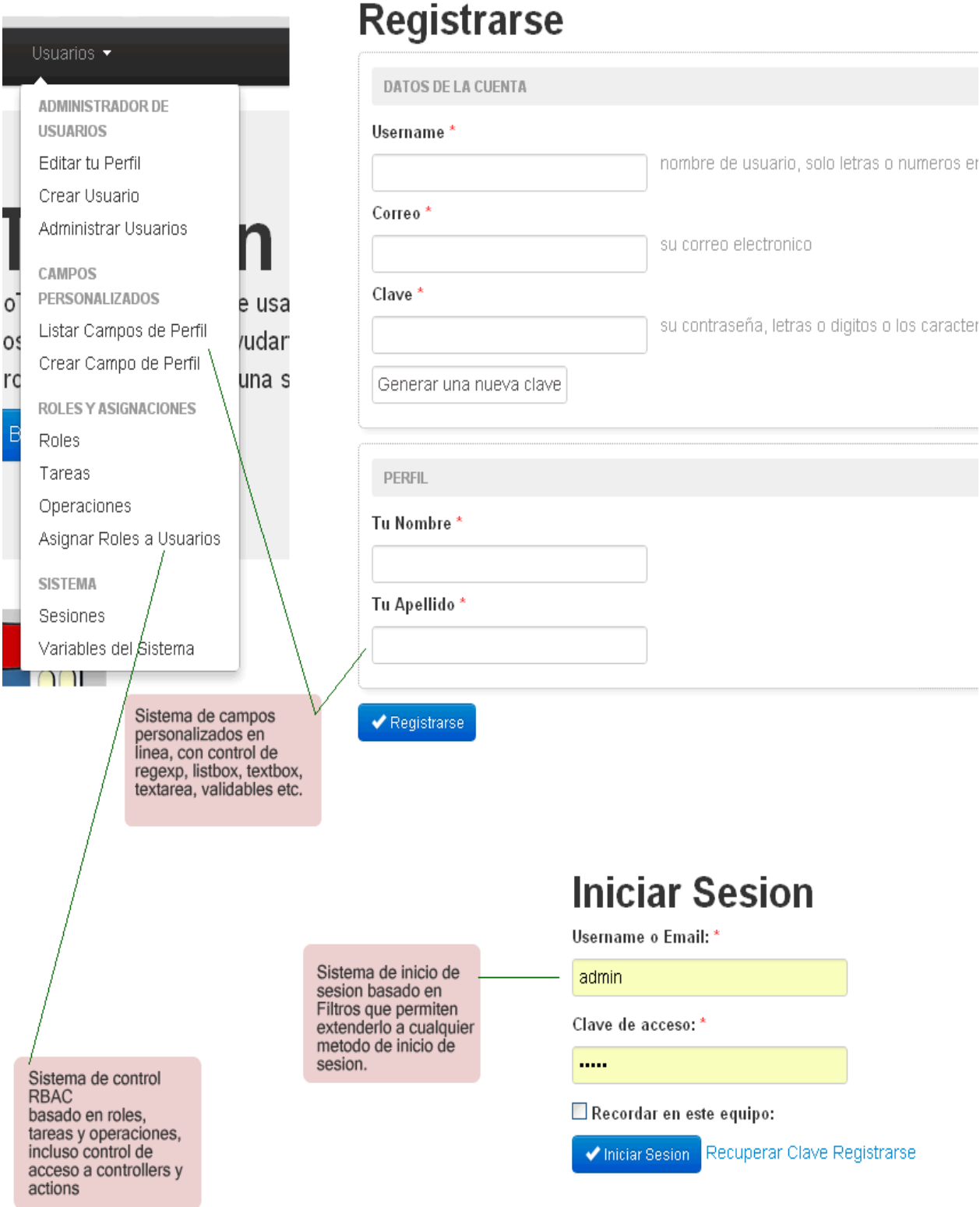
* PHP Version 5.2.6, 5.4.4, 5.4.5

* Yii Framework 1.10, 1.11, 1.12, 1.13

INSTALACIÓN:

Este es un documento para explayar más sobre Cruge, las instrucciones de Instalación se encuentran en el archivo README.md de Cruge, el cual puedes conseguir en línea en:

https://bitbucket.org/christiansalazarh/cruge


Usuarios
ADMINISTRADOR DE USUARIOS
Editar tu Perfil
Crear Usuario
Administrar Usuarios
CAMPOS PERSONALIZADOS
Listar Campos de Perfil
Crear Campo de Perfil
ROLES Y ASIGNACIONES
Roles
Tareas
Operaciones
Asignar Roles a Usuarios
SISTEMA
Sesiones
Variables del Sistema
Registrarse
DATOS DE LA CUENTA
Username *
nombre de usuario, solo letras o numeros en
Correo *
su correo electronico
Clave *
su contraseña, letras o digitos o los caracter
Generar una nueva clave
PERFIL
Tu Nombre *
Tu Apellido *
Registrarse
Sistema de campos personalizados en linea, con control de regexp, listbox, textbox, textarea, validables etc.
Iniciar Sesion
Username o Email: *
admin
Clave de acceso: *
Recordar en este equipo:
Iniciar Sesion
Recuperar Clave
Registrarse
Sistema de inicio de sesion basado en Filtros que permiten extenderlo a cualquier metodo de inicio de sesion.
Sistema de control RBAC basado en roles, tareas y operaciones, incluso control de acceso a controllers y actions

Que es Cruge ?

Cruge te permite administrar y controlar de forma muy eficiente y segura a tus usuarios y los roles que ellos deban tener en tu Aplicación Web.

Cruge tiene una alta Arquitectura OOP, basada en interfaces, lo que ayuda enormemente a usarla sin modificar en lo absoluto su propio core. Si necesitas cambiar de ORDBM, cruge lo permite. Si necesitas extender el funcionamiento de autenticacion para admitir nuevos metodos tambien lo permite mediante la implantacion de filtros de autenticacion, incluso dispones ademas de filtros insertables para controlar el otorgamiento de una sesion a un usuario y finalmente para controlar los registros y actualizaciones de perfil de tus usuarios. Todo eso sin tocar en lo absoluto el core de Cruge.

Cruge es un API, que incluye una interfaz de usuario predeterminada con el objeto que puedas usar el sistema como viene ahorrandote mucho tiempo. Esta interfaz hace uso del API de forma estricta, es decir, no hay "dependencias espaguetti" las cuales son las primeras destructoras de todo software.

La arquitectura que tu usarías en Cruge es así:

[Tu Aplicacion]--->[Yii::app()->user]	acceso a funciones muy basicas
[Tu Aplicacion]--->[Yii::app()->user->ui]	provee enlaces a funciones de la interfaz
[Tu Aplicacion]--->[Yii::app()->user->um]	provee acceso al control de los usuarios
[Tu Aplicacion]--->[Yii::app()->user->rbac]	provee acceso a las funciones de RBAC

internamente esta arquitectura dentro de Cruge es asi:

[Tu Aplicacion]--->[Cruge]--->[API]---->[Factory]---->[modelos]

esto significa, que aun dentro de Cruge las dependencias son estrictamente organizadas, es decir, no verás en ningún lado que el API vaya a instanciar a un modelo cualquiera, si eso fuera asi estariamos hablando de otra "extension" espaguetti, como aquellas que solo le funcionan a su creador, o que en el mejor de los casos, funcionan...pero manipulandoles el core.

Cruge es una extension en todo lo ancho de la palabra, realmente extiende las funciones basicas de manejo de usuario de Yii Framework, incorporando mas funciones en los paquetes originales.

La interfaz de usuario de Cruge es opcional, con esto quiero decirte que puedes usar Cruge en modo API, para lo cual debes conocer el modelo con detalles, aunque es muy intuitivo y de referencias cortas, para hacerlo entendible.

EJEMPLOS DEL USO DEL API DE CRUGE:

Para usar el API de usuarios de Cruge debes acceder por:

```
Yii::app()->user->um
```

BUSCAR UN USUARIO POR SU USERNAME o EMAIL

```
$usuario = Yii::app()->user->um->loadUser('admin@gmail.com',true);
echo $usuario->username;
```

true: es para indicar que cargue los valores de los campos personalizados. por defecto es : false.

BUSCAR UN USUARIO POR SU ID

```
$usuario = Yii::app()->user->um->loadUserById(123,true);
echo $usuario->username;
```

CREAR UN USUARIO NUEVO USANDO EL API

```
public function actionAjaxCrearUsuario(){
	// asi se crea un usuario (una nueva instancia en memoria volatil)
	$usuarioNuevo = Yii::app()->user->um->createBlankUser();
	$usuarioNuevo->username = 'username1';
	$usuarioNuevo->email = 'username1@gmail.com';
	// la establece como "Activada"
	Yii::app()->user->um->activateAccount($usuarioNuevo);
	// verifica para no duplicar
	if(Yii::app()->user->um->loadUser($usuarioNuevo->username) != null)
		{
			echo "El usuario {$usuarioNuevo->username} ya ha sido creado.";
			return;
		}
	// ponerle una clave
	Yii::app()->user->um->changePassword($usuarioNuevo,"123456");
	// guarda usando el API, la cual hace pasar al usuario por el sistema de filtros.
	if(Yii::app()->user->um->save($usuarioNuevo)){
		echo "Usuario creado: id=".$usuarioNuevo->primaryKey;
	}
	else{
		$errores = CHtml::errorSummary($usuarioNuevo);
			echo "no se pudo crear el usuario: ".$errores;
	}
}
```

LEYENDO UN CAMPO PERSONALIZADO DE UN USUARIO

```
Yii::app()->user->um->getFieldValue(Yi::app()->user->id,'nombre');
```

para el usuario activo se puede usar directamente asi:

```
$nombre = Yii::app()->user->getField('nombre');
```

OBTENIENDO EL EMAIL DEL USUARIO ACTIVO

```
$email = Yii::app()->user->email;
```

RECORRIENDO VARIOS USUARIOS Y LEER SU EMAIL

```
foreach(Yii::app()->user->um->listUsers as $user)
    echo "email=".$user->email."<br/>";
```

LEER TODOS LOS CAMPOS PERSONALIZADOS DE UN USUARIO

```
foreach($usuario->fields as $campo)
    echo "<p>campo: ".$campo->longname." es: ".$campo->fieldvalue;"</p>";¡
```

ACCEDER A UN CAMPO PERSONALIZADO DE UN USUARIO:

```
$cedula = Yii::app()->user->um->getFieldValue($usuario,'cedula');
```

ARMANDO UN COMBOBOX CON LA LISTA DE USUARIOS REGISTRADOS MOSTRANDO LOS CAMPOS PERSONALIZADOS

```
$comboList = array();
foreach(Yii::app()->user->um->listUsers() as $user){
    // evitando al invitado
    //
    if($user->primaryKey == CrugeUtil::config()->guestUserId)
        break;
    // en este caso 'firstname' y 'lastname' son
    // campos personalizados
    //
    $firstName = Yii::app()->user->um->getFieldValue($user,'firstname');
    $lastName = Yii::app()->user->um->getFieldValue($user,'lastname');
    $comboList[$user->primaryKey] = $lastName.", ".$firstName;
}
echo "Users: ".CHTML::dropDownList('userlist',",$comboList)."<hr/>";
```

VERIFICAR SI UN USUARIO TIENE ACCESO A UN ROL, TAREA u OPERACION:

```
if(Yii::app()->user->checkAccess('admin')){
        ...hacer algo...
}
```

VARIABLES DEL SISTEMA

Las variables del sistema estan contenidas en el modelo /cruge/models/data/CrugeSystem.php, son usadas para controlar como un usuario inicia sesión, opciones de registro de usuarios, entre otras.

Las variables del sistema se pueden modificar mediante el menu de administracion principal, en el item 'Variables del Sistema', las variables controlan el comportamiento de Cruge, por ejemplo, la pantalla de Registro, detener el sistema, duracion de la sesión, etc.

Tambien puedes acceder a las variables del sistema mediante API:

```
if(Yii::app()->user->um->getDefaultSystem()->getn('registerusingcaptcha')==1){
    ...
}
```

Puedes editar las variables en línea mediante la URL:

```
<?php echo Yii::app()->user->ui->userManagementAdminLink; ?>
```

En este menu verás las siguientes opciones:

1. Detener Sistema
   si esta activada no se permitira iniciar sesion o usar el sistema en general.

2. No Admitir Nuevas Sesiones
   solo bloquea el login, es decir nuevas sesiones de usuario.

3. Minutos de Duracion de la Sesion *
   el tiempo en minutos en que una sesion expira.

4. Registrarse usando captcha
   si esta activada presenta un captcha al momento de registrar a un nuevo usuario.

5. Activacion del usuario registrado
   opciones para aplicar al nuevo usuario registrado, puedes activarlo inmediatamente, o mediante
   activacion manual por parte de un administrador, o mediante un correo de verificacion.

6. Asignar Rol a usuarios registrados
   es el rol que tu quieres que se le asigne por defecto al usuario recien registrado.

7. Ofrecer la opción de Registrarse en la pantalla de Login.

Esta opción habilita el link de "registrarse" en la pantalla de login.
para que esta opcion funcione necesitas actualizar tu base de datos con
el siguiente script sql:

```
"alter table cruge_system add registrationonlogin integer default 1;"
```

8. Registrarse usando terminos

si lo activas, el usuario que se va a registrar debe aceptar los terminos que tu indiques.

9. Etiqueta

si la opcion anterior esta activa se le hara una pregunta, aqui puedes escribir esa pregunta, por ejemplo: "Por favor lea los terminos y condiciones y aceptelos para proceder con su registro"

10. Terminos y Condiciones de Registro

El texto de las condiciones para registrarse, solo aplica si la opcion de "registrarse usando terminos" esta activada.

MANEJANDO ENLACES

Puedes invocar al API UI de Cruge para acceder a los enlaces mediante:

```
<?php echo Yii::app()->user->ui->getLoginLink('iniciar sesion'); ?>
```

o simplemente:

```
<?php echo Yii::app()->user->ui->loginLink; ?>
```

La lista de enlaces disponibles es:

```
<?php echo Yii::app()->user->ui->loginLink; ?>
<?php echo Yii::app()->user->ui->logoutLink; ?>
<?php echo Yii::app()->user->ui->passwordRecoveryLink; ?>
<?php echo Yii::app()->user->ui->userManagementAdminLink; ?>
<?php echo Yii::app()->user->ui->registrationLink; ?>
```

lo que no debes hacer es**: hacer render de vistas internas de cruge, asi no funciona esto porque le quitas la programacion cuando tu mismo renderizas las vistas

Presentando el menu de administracion de usuarios

Este menu ya viene listo en Cruge, trae todos los items para que te evites el trabajo de ir a CrugeUi a ver cuales son los links. Para acceder al array de menu items lo haces desde:

```
<?php
	$items = Yii::app()->user->ui->adminItems;

	// items sera una array listo para insertar en CMenu, BootNavbar o similares.
?>
```

Si usas bootstrap, puedes hacerlo asi:

```
<?php $this->widget('bootstrap.widgets.BootNavbar', array(
	'fixed'=>false,
	'brand'=>"Tu App",
	'items'=>array(
		array(
			'class'=>'bootstrap.widgets.BootMenu',
			'items'=>array(Yii::app()->user->ui->adminItems),
		),
	),
)); ?>
```

También se dispone de:

```
Yii::app()->user->ui->userManagementAdminUrl
```

La cual brinda toda la lista de actividades a realizarse, de hecho se utiliza como punto de ejemplo tras instalar Cruge.

```
<?php $this->widget('zii.widgets.CMenu',array(
        'items'=>array(
                array('label'=>'Home', 'url'=>array('/site/index')),
                array('label'=>'About', 'url'=>array('/site/page', 'view'=>'about')),
                array('label'=>'Contact', 'url'=>array('/site/contact')),
                array('label'=>'Administrar Usuarios'
                        , 'url'=>Yii::app()->user->ui->userManagementAdminUrl
                        , 'visible'=>!Yii::app()->user->isGuest),
                array('label'=>'Login'
                        , 'url'=>Yii::app()->user->ui->loginUrl
                        , 'visible'=>Yii::app()->user->isGuest),
                array('label'=>'Logout ('.Yii::app()->user->name.')'
                        , 'url'=>Yii::app()->user->ui->logoutUrl
                        , 'visible'=>!Yii::app()->user->isGuest),
        ),
)); ?>
```

USO DE LAYOUTS

Cruge te permite que su interfaz de usuario predeterminada pueda ajustarse a tu sitio web usando lo que en Yii se conoce como Layouts. Por favor conoce mas acerca del uso de layouts en el sitio oficial de Yii Framework, es importante que lo conozcas.

Por ejemplo, quieres que el formulario de registro de nuevo usuario se presente en un esquema de diseño distinto al que yii trae por defecto, entonces tú podrías crear un nuevo layout que se ajuste a tus necesidades y luego indicarle a Cruge mediante la configuración del componente cuál sería ese layout a usar cuando un usuario quiera registrarse, así:

```
<?php
	'components'=> array(
		..otros ajustes aqui...
		'loginLayout'=>'//layouts/bootstrap',
		'registrationLayout'=>'//layouts/bootstrap',
		'activateAccountLayout'=>'//layouts/bootstrap',
		'generalUserManagementLayout'=>'//layouts/bootstrap',
	),
?>
```

Te cuidado especial con "generalUserManagementLayout": este es un layout especial, porque las funciones de administracion de usuarios requieren un Portlet para presentar las opciones administrativas, por defecto Cruge apunta este valor a: "ui", el cual es el nombre de un layout prefabricado que ya trae un Portlet, practicamente idendico al que Yii trae por defecto llamado //layouts/column2.

El Layout para UI de Cruge por defecto es:

tuapp/protected/modules/cruge/views/layouts/ui.php

En este layout (ui.php) hay un Portlet, que será llenado con los items de administración en linea de Cruge, estos items salen del modulo UI de Cruge, el cual es accesible usando:

```
Yii::app()->user->ui->adminItems
```

CRUGE RBAC

El RBAC que Cruge implementa es el mismo de Yii Framework de fábrica, salvo algunos agregados que he notado que han sido incorporados en las nuevas versiones de Yii.

RBAC Es un sistema de control de acceso basado en roles (por sus siglas en ingles). Todo el mecanismo RBAC puede ser manejado mediante la interfaz (UI) de Cruge, o mediante su API. Las dos modalidades para usar en este mecanismo son:

Consulta Manual de Permisos.

Es basicamente el mismo mecanismo que provee Yii, pero en Cruge se ha ampliado un poco mas. Para usar este mecanismo: en cualquier parte de tu codigo fuente puedes poner lo siguiente:

```
<?php
	if(Yii::app()->user->checkAccess('puede_ver_menu_sistema')) {
		...mostar menu sistema...
	}
?>
```

(este mecanismo "directo" fue incorporado en las mas recientes versiones de Yii, para la fecha de salida de Cruge al aire no estaba disponible para Yii estándar, antes había que acceder mediante el componente AuthManager de Yii)

Consulta Automatizada segun controller/action.

Este mecanismo es muy útil, porque permite controlar el acceso a tus controllers/actions de forma totalmente automatizada y controlada mediante la UI de Cruge. Para usar este mecanismo, necesitarás incluir en tu Controller (cualquiera que tú uses y que desees controlar a nivel de permisos) el siguiente código:

```
<?php
	..cuerpo de tu controladora...
	public function filters()
	{
		return array(
			array('CrugeAccessControlFilter'),
		);
	}
	..cuerpo de tu controladora...
?>
```

Al usar CrugeAccessControlFilter estas permitiendo que Cruge controle el acceso tanto al “controller” en general como al “action” especifico.

Ejemplo:

Un usuario cualquiera, incluso un invitado, intenta acceder a la siguiente URL:

index.php?r=empleado/vernomina.

pues bien, Cruge verificara dos cosas:

a) el acceso a la controladora: 'Empleado'.

b) el acceso al action 'Vernomina'.

lo hara de esta forma:

a) verifica si el usuario (aun invitado) tiene asignada la operacion:
'controller_empleado'

b) verifica si el usuario (aun invitado) tiene asignada la operacion:
'action_empleado_vernomina'

si ambas condiciones se cumplen (a y b) entonces tendrá acceso al action.

Si tu quieres denegar el total acceso a un controller simplemente no le asignas al usuario la operacion que tenga el nombre del controller antecedido de la palabra 'controller_'.

Si tu quieres denegar el acceso a <u>un action</u> de un controller simplemente no le asignas al usuario la operacion que tenga el nombre del action: 'action_nombrecontroller_nombreaction'.

Programacion del RBAC: diferencias con otras extensiones.

En el caso de Cruge, la programacion de RBAC no se hace "asumiendo que un usuario va pasar por ahi.." (modo Yii Rights). En cambio en Cruge, debes "tratar de hacer pasar al usuario por donde quieres", este ultimo metodo, a mi juicio, es mas seguro porque te obliga a verificar que realmente el usuario pudo acceder o no a tal o cual parte.

Modo de Programacion del RBAC

Para activarlo, en la configuracion de tu aplicacion debes considerar estos dos argumentos:

'rbacSetupEnabled'=>true,

Permitira que las operaciones se vayan creando (valor true) en la tabla de la base de datos a medida que vas probando el sistema, de lo contrario deberas crear las operaciones a mano. Las operaciones que se crearan automaticamente seran: 'controller_nombredetucontroller' y 'action_tucontroller_tuaction'.

'allowUserAlways'=>true,

Permitirá el paso (valor true) al usuario aunque este no tenga permiso. Cuando estás en produccion ponlo en 'false' lo que causara que el usuario reciba una excepción si no dispone del permiso para usar cualquier rol, tarea u operación. Es bueno ponerlo en TRUE para el momento en que estas configurando los permisos.

Aunque este atributo esté en valor TRUE siempre podrás conocer que permisos fallaron, debido a que puedes agregar a tu layout principal una llamada a displayErrorConsole, asi:

```
<?php
        echo Yii::app()->user->ui->displayErrorConsole();
?>
```

Usando el LOG

Adicionalmente todos los errores de permiso que se generen seran reportados en el log bajo el key 'rbac', para poder visualizar los errores en protected/runtime/application.log deberas configurar tu config/main.php para indicar el key del log:

```
<?php
        'log'=>array(
                'class'=>'CLogRouter',
                'routes'=>array(
                        array(
                                'class'=>'CFileLogRoute',
                                'levels'=>'error, info, rbac', // <--- agregar 'rbac'
                        ),
                        // uncomment the following to show log messages on web pages
                        //array('class'=>'CWebLogRoute'),
                ),
        ),
?>
```

eso causara que en:

protected/runtime/application.log

se emitan mensajes como estos:

2012/07/27 15:24:25 [rbac] [application] PERMISO REQUERIDO:
invitado
iduser=2
tipo:operacion
itemName:action_catalog_imageh

Tips de Programacion del RBAC

Cuando quieras programar el RBAC con cruge, usa dos navegadores: uno abierto con un usuario administrador, para que puedas ir activando las operaciones para un rol especifico a medida que sea necesario, y abre otro navegador con el usuario que tenga el rol que quieres programar. No olvides tener activado el flag: rbacSetupEnabled.

Por ejemplo, quieres que el usuario 'juan' que tiene asignado el rol 'empleado_regular' tenga solo acceso solo a donde quieres, entonces en el segundo navegador inicia sesion con 'juan', y tratas de ir al menu u operacion requerida, cruge ira informando al pie de la pagina los permisos requieridos. Luego con el navegador que tiene abierto el usuario 'admin' entonces vas verificando los permisos y se los asignas al rol.

Considera que cuando rbacSetupEnabled esta habilitado, entonces, asumiendo ademas que no hay ninguna operacion creada iras viendo que cruge creara las operaciones automaticamente de acuerdo a donde el usuario 'juan' vaya pasando, solo las crea si previamente no existen.

Ejemplo: no hay ninguna operacion creada, entonces el usuario juan quiere entrar a 'site/index', por tanto, si la controladora 'siteController' esta manejada con el filtro: 'CrugeAccessControlFilter' (ver tema mas arriba) entonces veras que se creara una operacion llamada 'site_controller' y otra 'action_site_index', deberas entonces asignarle estas dos operaciones al rol al cual 'juan' pertenece.

El superUsuario

Por defecto, cruge considera a un superusuario, para proposito de administracion, depuracion etc.

Para que cruge considere a un usuario como un "superusuario" entonces este debera tener como username el mismo valor configurado en CrugeModule::superuserName. Por defecto este valor viene configurado como '**admin**' (con password admin)

Para cambiar este valor, al igual que otros parametros de Cruge, no debes cambiar nada en CrugeModule, sino mediante

editar:

tuapp/protected/config/main.php asi:

y poner:

```
'cruge'=>array(
        'superuserName'=>'administrador', (suponiendo que no te gusta 'admin')
),
```

Como trata Cruge al superusuario ?

Basica y unicamente en: cruge\components\CrugeWebUser.php, en el metodo: checkAccess, si es un superadmin, entonces siempre le dara permiso haciendo que
Yii::app()->user->checkAccess siempre retorne true.

Como saber si estamos ante un superusuario ?

Consultando a:
Yii::app()->user->getIsSuperAdmin()

o mas corto:

Yii::app()->user->isSuperAdmin

El usuario Invitado

Cruge hace especial tratamiento al usuario invitado. Para esto CrugeModule contiene un atributo llamado **guestUserId** , el cual es usado para indicarle al sistema Cruge cual de sus usuarios existentes en la base de datos de usuarios es el invitado.

Por defecto cuando Cruge es instalado desde el script de base de datos (protected/modules/cruge/data), se crean dos usuarios:

```
insert into `cruge_user`(username, email, password, state) values
 ('admin', 'admin@tucorreo.com','admin',1)
 ,('invitado', 'invitado','nopassword',1)
;
```

Siendo 'admin' el usuario con ID 1, y siendo 'invitado' el usuario con ID 2. Por esto veras que por defecto en CrugeModule.php ya esta predefinido el atributo guestUserId en 2. No lo cambies a menos que cambies el ID del usuario invitado.

Cómo trata Cruge al usuario invitado.

Por defecto en YII cuando tu llamas a Yii::app()->user->id esta devuelve 0 (cero) cuando un usuario es invitado. En Cruge esta misma llamada a Yii::app()->user->id devolverá al valor configurado en CrugeModule::guestUserId, el cual de fábrica es dos (2).

Puedes confiar siempre en Yii::app()->user->isGuest, ya que ésta función considera todo esto para saber si el usuario es un invitado.

TRAS INSTALAR, CONFIGURAR EL USUARIO INVITADO

Esto es importante: Si no asignas al usuario invitado a ningún rol (por defecto es así) entonces no tendrá acceso a ningún “controller” que este siendo manejado por CrugeAccessControlFilter (mas arriba explico acerca de CrugeAccessControlFilter). Por tanto tras instalar tu sistema con Cruge debes hacer lo siguiente:

1. Crea un rol llamado 'invitados'.

2. Asignale a ese rol las operaciones necesarias, por ejemplo 'controller_site', 'action_site_index', 'action_site_contact', 'action_site_login' y otras que vayas viendo que se requieran (usa el LOG).

3. Asigna este rol creado al usuario invitado, el cual viene de fábrica.

Tip: No necesariamente el rol del 'invitado' debe llamarse 'invitado'. Por conveniencia es sano que asi sea pero no es indispensable.

ENCRIPTADO DE CLAVES

Por defecto Cruge trae dos usuarios, admin e invitado. En el caso de admin la clave es 'admin', por defecto además Cruge trae la encriptacion de claves desactivada, esto es para facilitar el trabajo mientras se instala.

Para activar o desactivar la encriptación se usa el atributo 'useEncryptedPassword' => false (en tuapp/protected/config/main.php)

Si quieres encriptar las claves y hacer que en la tabla de usuarios estas no sean visibles, entonces deberas tomar un paso extra:

a) encriptar tu clave de admin y guardarla via base de datos en la tabla de usuarios (cruge_user) de forma manual, mediante un script sql directo.

b) luego activas la encriptación poniendo a true el atributo useEncryptedPassword y con eso podrás acceder y verás que los nuevos usuarios que se vayan creando tendrán su nueva clave encriptada.

MÉTODOS DE ENCRIPTACIÓN

Por defecto Cruge encripta la clave en MD5. Sin embargo, este comportamiento es posible modificarlo desde el archivo "tuapp/protected/config/main.php", en la sección "modules"

```
'cruge'=>array(
    ...
    'hash' => 'sha1',
    ...
)
```

Los valores aceptados por el parámetro hash son cualquiera aceptado por la función “hash” (PHP 5.1+), para mayor información debes visitar: [hash_algos()](http://www.php.net/manual/en/function.hash-algos.php). (Gracias a Ricardo Andrés Obregón).

EVENTOS DE INICIO, CIERRE Y EXPIRACIÓN DE SESIÓN

Puedes hacer que Cruge vaya a una URL cuando se inicie sesión, se cierre sesión, o cuando esta expire. Esto se configura en tuproyecto/protected/config/main.php, allí puedes colocar las URL en las siguientes variables del módulo Cruge:

```
<?php
        'afterLoginUrl'=>array('/site/welcome'),
        'afterLogoutUrl'=>array('/site/page','view'=>'about'),
        'afterSessionExpiredUrl'=>null,
?>
```

IMPORTANTE: no olvidar el slash inicial "/" sino la url no funcionará.

LOGIN, LOGOUT, SESIONES. CÓMO FUNCIONA EN CRUGE.

Debes comprender que hay dos estados en un sistema RBAC estándar como lo es Cruge:

a) La autenticación.

> Es el estado del sistema RBAC en el cual se verifíca que un usuario es realmente quien dice ser, se verifica mediante un usuario y una clave, un código, un mecanismo de OPEN-ID (pronto será incorporado en Cruge, al igual que Facebook)

b) La sesión.

> Tras autenticarse, a un usuario se le asigna un espacio y un tiempo para operar en el sistema. El tiempo de la sesión se configura en las “Variables del Sistema” (ver tema al inicio). Tras cumplirse el tiempo, Cruge abortará al usuario emitiendo el evento onSesionExpired (de la clase: cruge.models.filters.DefaultSessionFilter), el cual derivará en una llamada a la URL afterSessionExpiredUrl (ver arriba).
>
> Cruge maneja la sesión bajo una arquitectura delicada que será explicada a futuro con mayor detenimiento y con soporte de diagramas UML, por ahora no viene al caso detallar tanto.

Debes saber por tanto que Cruge tiene un 'filtro para otorgar sesiones' y un 'filtro de autenticacion' estos funcionan asi:

Primero se pasa por el filtro de autenticacion, el cual le da sentido a 'Yii::app()->user->getUser()', luego el filtro de sesión verifica si el sistema esta apto para recibir una nueva sesión, quizás está en mantenimiento o quizás no, por tanto es el filtro de sesión quien ahora entra en juego.

Una vez que el filtro de autenticación determina que se puede dar una sesion a 'juan perez', entonces se le crea una sesion y se llama a un evento llamado 'onLogin' de la clase 'cruge.models.filters.DefaultSessionFilter'.

Este evento de onLogin es quien establece el valor a

```
<?php
    Yii::app()->user->returnUrl
?>
```

el cual es procesado por UiController para redirigir el browser a una pagina que tu indicas.

Ahora, importante, cuando tu haces logOff manualmente, o si por alguna razón tú usando el api estándar de Yii haces una llamada a

```
<?php
    Yii::app()->user->logout()
?>
```

verás que también serás redirigido a la URL que hayas especificado en 'afterLogoutUrl', la razón de esto es simple:

Cruge es una extensión real del paquete de autenticación estándar de Yii Framework, por tanto para ti es transparente si haces logout o login a mano o de forma automática.

Conviviendo con accessControl filter.

Supongamos que quieres que tras iniciar sesion exitosamente con Cruge el usuario sea redirigido al actionBienvenido de siteController (index.php?r=site/bienvenido), pero utilizando el filtro accessControl que Yii trae por defecto y no mediante los eventos de sesión que te he mostrado en la página anterior.

Pues bien el metodo que aqui describo es algo estandar para Cruge o para Yii en general, no es nada nuevo.

1. en siteController (en el controller de tu gusto) creas un action el cual desplegara la pagina que solo vera aquel usuario que haya iniciado sesion exitosamente.

```
<?php
	public function actionBienvenido(){
		$this->render('bienvenido');
	}
?>
```

2. en siteController usas el filtro accessControl y los rules (que vienen de caja en Yii), asi:

```
<?php
	public function filters()
	{
		return array(
			'accessControl',
		);
	}
	public function accessRules()
	{
		return array(
			array('allow',
				'actions'=>array('index','contact','captcha'),
				'users'=>array('*'),
			),
			array('allow',
				'actions'=>array('bienvenido'),
				'users'=>array('@'),
			),
			array('deny',  // deny all users
				'users'=>array('*'),
			),
		);
	}
?>
```

con esto le estas diciendo a tu aplicacion que para el action "site/bienvenido" se requiere que el usuario deba haber iniciado sesion exitosamente (con cruge o con yii tradicional, ambos funcionan por la misma via de autenticacion, por eso y mas Cruge es una extension real y no solo un monton de codigo raro).

De este modo, si un usuario invitado presiona el enlace "login" entonces tu lo envias a site/bienvenido, si no ha iniciado sesion se le pediran credenciales y luego se le enviara a la vista site/bienvenido. por tanto el paso 3 es requerido, a continuacion:

3. finalmente sustituye tu enlace a login por un enlace a site/bienvenido.

Que sucederá ?

(podriamos dibujar esto como un diagrama de secuencia en UML)

a) Tu invitado visita tu website (no ha iniciado sesion aun por eso es un invitado) y sigue el enlace 'login' o 'iniciar sesion' que tu has provisto (y que apunta a site/bienvenido como dice el paso 3).

b) Tu invitado sera redirigido automaticamente a "index.php?r=cruge/ui/login" (o a la url de login del sistema que este registrado para autenticar, en este caso Cruge), esto debido al "accessControl" que implementaste en el paso 2.

c) Luego tras iniciar sesion exitosamente sera redirigido automaticamente a "index.php?r=site/bienvenido",

(funciona asi debido a que en el paso 2 al detectarse que no se ha iniciado sesion entonces se establecio el valor de returnUrl a "site/bienvenido", por tanto cuando Cruge o Yii estandar hacen un login correcto redirigen a tu usuario a la direccion que tenga almacenada en returnUrl, en este caso site/bienvenido)

Como has visto Cruge es un sistema orquestado para trabajar en conjunto con el actual sistema estandar de autenticacion de Yii, por eso como dije antes es una extension: porque extiende la funcionalidad basica de Yii a un nivel mas alto.

FILTROS

Cruge permite que se pueda extender más allá usando filtros. Existen varios tipos de filtros, todos se instalan en config/main y disponen de una interfaz (interface) que debes respetar, a continuación la lista de filtros, si necesitas crear un filtro nuevo fíjate en como está hecho el filtro por defecto:

Filtros de autenticación:
permite que amplies como se busca un usuario para ser autenticado. Se ubican en:

protected\modules\cruge\models\auth\CrugeAuthDefault.php

Filtros de sesión:
permite que puedas controlar como se entrega una sesion a un usuario, inclusive puedes denegarla. Se ubican en:

protected\modules\cruge\models\filters\DefaultSessionFilter.php

Filtro de actualización:
permite saber si un usuario actualizo su perfil. Se ubican en:

protected\modules\cruge\models\filters\DefaultUserFilter.php

Debido a lo extenso de Cruge no he tenido tiempo de documentar bien estos filtros, pero es bastante intuitivo, aunque pronto iré documentandolo en detalle.

La idea principal a cubrir con los filtros es que mediante config/main tu puedas decirle a Cruge cuales filtros va a usar, para que estos hagan el trabajo que tu necesitas pero sin tocar ni modificar el core de Cruge, en cambio usando el concepto de Override de la programación orientada a objetos (POO).

EL ENVIO DE CORREOS CON CRUGEMAILER

CrugeMailer es un componente que envía correos "en base a vistas de Yii", de forma centralizada, dejandole a una sola función la responsabilidad de decidir por dónde o cómo se envía el correo (si es por phpmailer o por mail directo). Además, te ayuda a modelar las funciones de negocio para que sea fácil usarlas. Por ejemplo, podrías invocar a

```
Yii::app()->crugemailer->enviarEmailNuevaClave($usuario);
```
(es un ejemplo, esta función no existe en Cruge, pero si verás otras similares)

En primer lugar olvida por completo el hecho de que Cruge llame a la función mail() por todos lados para enviar un email. Eso sería un error de diseño, por tanto, el uso de emails esta fuertemente modelado usando un componente llamado “crugemailer”, el cual es accedido desde dentro de cruge así:

```
Yii::app()->crugemailer
```

CrugeMailer no solo está hecho para Cruge, sino para toda tu aplicación. Puedes ampliar el conocimiento acerca de CrugeMailer leyendo:

https://github.com/yiiframeworkenespanol/crugemailer

CrugeMailer se configura asi (ya lo hiciste al configurar a Cruge al inicio) :

```
'crugemailer'=>array(
        'class' => 'application.modules.cruge.components.CrugeMailer',
        'mailfrom' => 'email-desde-donde-quieres-enviar-los-mensajes@xxxx.com',
        'subjectprefix' => 'Tu Encabezado del asunto - ',
        'debug' => true,
),
```

Como te he comentado podrás usar a Yii::app()->crugemailer desde toda tu aplicación, no solo Cruge, para eso fue diseñado. Lo primero que debes comprender acerca de CrugeMailer es que es un componente que envía correos "en base a vistas de Yii", es decir, se crea una vista y se le indica a CrugeMailer que mande ese correo a un destinatario de la misma manera en como tu renderizarías una vista en Yii.

Caso de ejemplo:

1. Quieres crear un nuevo correo especifico basado en una vista, por tanto, creas una clase que extienda de CrugeMailer, la llamarias por ejemplo: "MiClaseCrugeMailer" y supon que la guardas en tu directorio de 'protected/components/MiClaseCrugeMailer.php' (!!no dentro de cruge module!!)

2. Esta clase contendría un nuevo método llamado "nuevoCorreoAlCliente", he aquí la bondad de CrugeMailer, te permite "modelar la solucion" dándole a ese método un nombre coherente con aquello que quieres hacer, eso hace que tu código sea ordenado y siempre sabrás qué hace, además de poder reutilizarlo en cualquier mommento.

3. En "config/main" le indicarías al componente crugemailer que use esa nueva clase en vez de la que viene por defecto. Se cambiaría por: 'class' => 'application.components.MiClaseCrugeMailer',

Para invocar esa función tu solo harías lo siguiente:

```
Yii::app()->crugemailer->nuevoCorreoAlCliente($usuarioSeleccionado,"asunto");
```

otro ejemplo:

```
Yii::app()->crugemailer->elStockHaCambiado($usuarioSeleccionado,"hola");
```

Que contendria ese nuevo metodo ? (nuevoCorreoAlCliente), algo tan simple como:

```
public function nuevoCorreoAlCliente($usuario,$asunto){
                $this->sendemail($usuario->email,$asunto,
                $this->render('enviarclave',array('data'=>$usuario)
        )
  );
}
```

Es muy importante que te fijes en dos cosas:

1) **Se usa a: $this->sendemail**. No andas poniendo aquí y por todas partes tu método específico de correo, en cambio "centralizas el envío de correos hacia una función sendemail" que podrías dejar por defecto como viene o que podrías extender tú en esta nueva clase que has creado: MiClaseCrugeMailer.

2) **Se usa a: $this->render**, este método recibe el nombre de una vista la cual tiene la forma html que se enviará por correo, esa forma, al igual que cualquier vista de Yii recibe por argumentos un array con los parámetros que serán accesibles en la vista. Tal cual como lo hace Yii en su mecanismo de vistas.

Lo que no debes hacer es: sustituir $this->sendemail por una montón de código que sólo sirva para una vez en la vida... porque estarías matando el concepto de encapsulamiento del modelado de objetos, acabando con el trabajo de todos aquellos que hacen de la programación orientada a objetos un arte.

¿ Que se logra al usar a CrugeMailer ?:

Se logra que tú hagas envíos de correos orientados a tu modelo de negocio, sin hacer espaguetti con el código!!, centralizando y encapsulando el código... todos tus correos invocarán a sendEmail...y no tendrás que crear y duplicar o triplicar el esfuerzo de escribir las funciones de correo.

Se logra que puedas clarificar el codigo...envias un correo con un proposito muy definido:

```
Yii::app()->crugemailer->nuevoCorreoAlCliente($usuario, "hola")
```

Se logra que en cualquier momento puedas decidir por cual via se envia el correo, si por mail() o por phpmailer() o por cualquier otra que tu decidas..solo cambiando un metodo..sin hacer espaguetti (codigo interdependiente necio y ofuscado que solo complica la vida y termina acabando con la vida del proyecto entero).

Quieres cambiar el metodo mail() por otro de tu preferencia. ¿ Cómo hacerlo ?

(Por favor lee acerca de CrugeMailer para que conozcas a qué me refiero aquí)

Crearias una nueva clase que extiende de CrugeMailer, harías un Override o Sobrecarga de la función "sendEmail" que ya crugemail trae pre-configurada, solo que ahora cada vez que tus funciones que ya has creado llamen a sendEmail llamarán entonces a esta nueva y no a la que viene por defecto.

ERRORES FRECUENTES

****"Excepcion: no se pudo hallar el sistema de configuracion, quiza la tabla cruge_system esta vacia o ha indicado un identificador de sistema inexistente."****

Este error es generado cuando se ha instalado cruge pero la tabla cruge_system no tienen ningun dato. por defecto cuando el script sql se instala por primera vez este trae datos para cruge_system. Cruge_system es una fila que informa acerca de la variables de un sistema, pueden crearse varias configuraciones a las cuales se les hace referencia por su nombre, si en config/main hay una referencia a esta configuracion y esta no existe en cruge_system entonces este error aparecera.

****"Excepción: por favor cambie las referencias a 'useridentity' por 'crugeuser'"****

Cuando ocurre ?

cuando ejecutas: http://localhost/tuapp/index.php?r=site/login, llenas el form de login y le das clic a Aceptar. Eso causar# la excepcion.

Por qué ocurre ?

Esto sucede principalmente porque estas usando el formulario de login por defecto de Yii, en cambio debes usar el de Cruge, al cual se accede mediante:

```
<?php echo Yii::app()->user->ui->loginLink; ?>
```

Sucede porque por defecto el formulario de login de yii llama a Yii::app()->user->login pasandole como argumento a una instancia de autenticacion que implementa el modelo basico de Yii y no el de Cruge, lo escribo así para explicarlo en palabras cortas. Explicandolo mas allá, tiene que ver con que por defecto Yii crea un formulario cuyo modelo implementa a "new UserIdentity" entonces este UserIdentity va a invocar a Cruge y este ultimo lo desconoce porque no es una instancia con la que pueda trabajar.

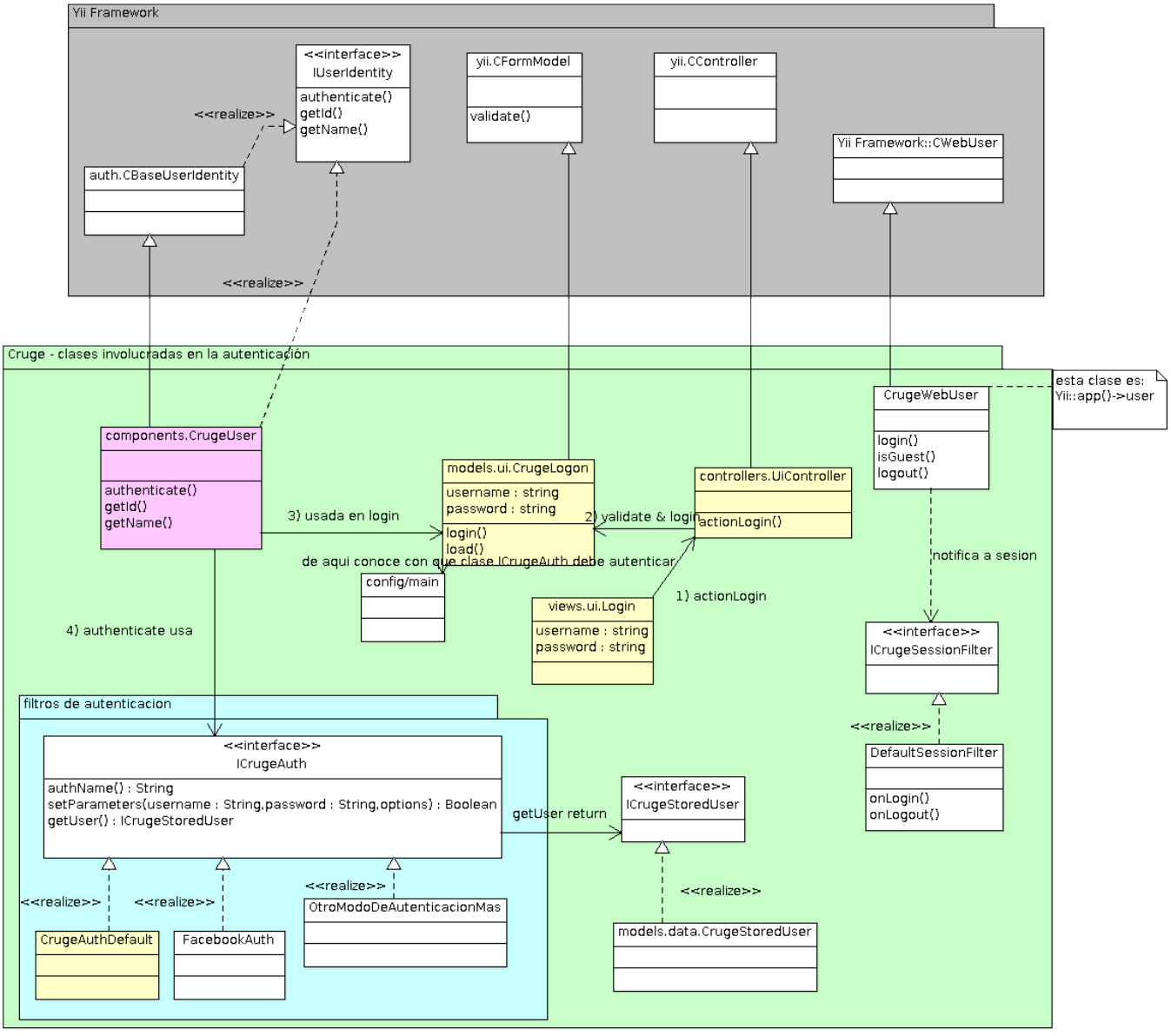
I18N

Si ves, Cruge escribe los mensajes en español, mientras que su código esta en inglés. Todos los mensajes se dirigen a la clase CrugeTranslator::t("mensaje en español"), por tanto ese es el punto para traducir a otro idioma. En un futuro nuevo commit se harán traducciones. Pronto.

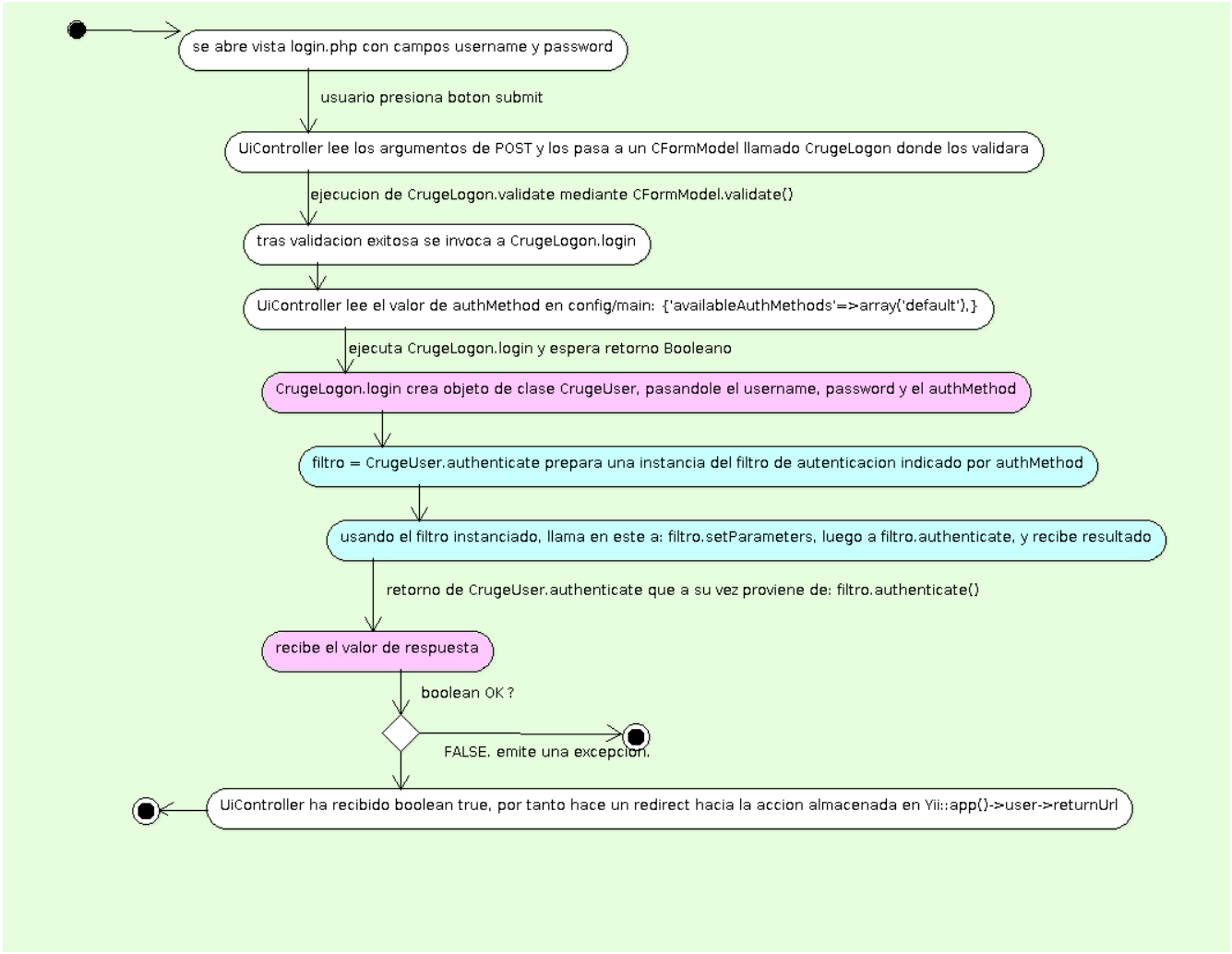
DIAGRAMAS UML DE CRUGE

Es importante conocer cómo esta diseñado Cruge, para esto proveeré tres diagramas importantes, no son todos, hay más, pero estos son los indispensables para conocer qué sucede cuando se hace click en "Login":

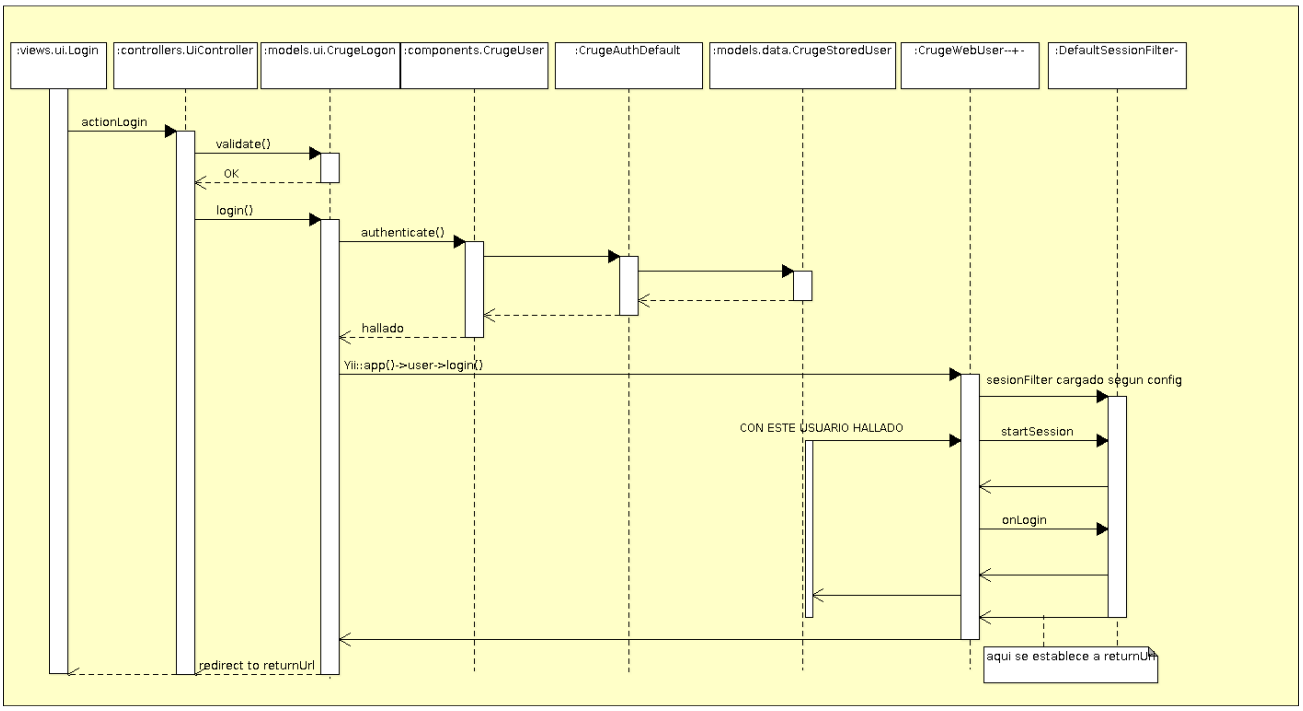
En este primero diagrama se muestran las clases que están involucradas en el proceso de autenticación (junto a su relacion con otras clases). **Estos diagramas están en línea en el repositorio de Cruge**.



En el segundo diagrama puedes observar que sucede en lineas generales cuando se presiona el boton login.



Finalmente aqui hay una secuencia en el tiempo y una vista de las clases involucradas, este diagrama se lee de izquierda a derecha, arriba en los cuadros grandes al tope se listan las clases involucradas, de cada clase sale una linea vertical larga de la cual a su vez salen flechas hacia otras clases, esas flechas son acciones a realizarse, llamadas, etc.



DIAGRAMAS EN LINEA

https://bitbucket.org/christiansalazarh/cruge/downloads/screenshots.gif

https://bitbucket.org/christiansalazarh/cruge/downloads/Diagrama-de-clases-proceso-de-autenticacion2.png

https://bitbucket.org/christiansalazarh/cruge/downloads/diagrama-de-actividad--autenticaion.png

https://bitbucket.org/christiansalazarh/cruge/downloads/diag.secuencia-de-login2.png