|  |  |
| --- | --- |
| Geo Parking | Informe Mensajes GeoParking  2014 |

# 

# Control de la documentación

### Control de la Configuración.

|  |  |
| --- | --- |
| Título: | Informe Mensajes GeoParking |
| Referencia: | GeoP\_Proyecto\_Doc\_GeoParkingMensajes.docx |
| Autores: | Leonel Romero |
| Fecha: | 29/05/2014 |

### Histórico de Versiones.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Versión | Fecha | Estado | Responsable | Cambios |
| 1.0 | 13/08/2014 | Creado | Leonel Romero [autor] |  |
|  |  |  |  |  |

Contenido

[Control de la documentación 3](#_Toc395727367)

[Control de la Configuración. 3](#_Toc395727368)

[Histórico de Versiones. 3](#_Toc395727369)

[Informe de utilización de mensajes personalizados en el proyecto GeoParking 5](#_Toc395727370)

[Introducción 5](#_Toc395727371)

[Arquitectura 5](#_Toc395727372)

[Forma de uso 5](#_Toc395727373)

# Informe de utilización de mensajes personalizados en el proyecto GeoParking

## Introducción

En este documento se mostrara la forma de utilización de los diferentes mensajes del proyecto GeoParking, los cuales son 3, mensaje de Información, mensaje de Error y mensaje de Confirmación. Cada mensaje puede mostrarse indistintamente como un modal o una barra en la parte superior de la página.

La arquitectura de los mensajes es tal que permite una utilización simple e igual para todos los mensajes, pudiendo personalizar todas las características de los mismos.

## Arquitectura

Los tres mensajes son controles de usuario que están ubicados en la master page, los mensajes de error e información son simples mensajes sin funcionalidad, mientras que el mensaje de confirmación capta la acción del usuario (si presiona botón Si o botón No) y realiza una acción.

Para poder setear el evento del botón a través de la master page, se utiliza un tipo delegado (EventHandler) que encapsula un método en una variable permite enviarla por parámetro, es así que se setea el evento click del botón correspondiente a través de las propiedades master.ConfirmarMensaje y master.CancelarMensaje. Dichas propiedades graban el evento en una variable de sesión para que no se pierda al realizarse un postback o postback parcial y se asigna el evento al control en la inicialización de la pagina a través de la variable de sesión mencionada.

Al utilizar los diferentes mensajes, se pueden enviar varios parámetros diferentes para poder personalizar los mismos.

Parámetros comunes en los tres mensajes:

* String mensaje: Es el mensaje que se desea mostrar.
* String título: Es el mensaje que se desea mostrar como titulo del modal (tener en cuenta que las alertas no tienen titulo)

Parámetros de Error e Información:

* TipoMensajeEnum tipo: es el tipo de mensaje que se quiere mostrar (Alerta, Modal o ambas).

Parámetros de Confirmación:

* EventHandler eventoConfirmacion: Es el evento que se desencadenara al confirmar el mensaje.
* EventHandler eventoCancelacion: Es el evento que se desencadenara al cancelar el mensaje.

## Detalles a Tener En Cuenta

1. El mensaje de confirmación **siempre** necesita un evento de confirmación, y solo puede asignarse un evento de cancelación si hay uno de confirmación.

## Código

A continuación se muestra el código de los mensajes, el mismo esta separado en la master page, y en las paginas donde se desea utilizar.

### Site.Master

#region mensajes

#region MensajeInformacion

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">Mensaje a mostrar</param>

public void MostrarMensajeInformacion(string mensaje)

{

MostrarMensajeInformacion(TipoMensajeEnum.MostrarAlertaYModal, mensaje, "Mensaje");

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="tipo">Alerta, modal o ambas</param>

/// <param name="mensaje">mensaje</param>

public void MostrarMensajeInformacion(TipoMensajeEnum tipo, string mensaje)

{

MostrarMensajeInformacion(tipo, mensaje, "Mensaje");

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">Mensaje</param>

/// <param name="titulo">Titulo</param>

public void MostrarMensajeInformacion(string mensaje, string titulo)

{

MostrarMensajeInformacion(TipoMensajeEnum.MostrarAlertaYModal, mensaje, titulo);

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="tipo">Alerta, modal o ambas</param>

/// <param name="mensaje">mensaje</param>

/// <param name="titulo">titulo del modal</param>

public void MostrarMensajeInformacion(TipoMensajeEnum tipo, string mensaje, string titulo)

{

msjInfo.MostrarMensaje(tipo, mensaje, titulo);

}

#endregion

#region MensajeError

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">mensaje</param>

public void MostrarMensajeError( string mensaje)

{

MostrarMensajeError(TipoMensajeEnum.MostrarAlertaYModal, mensaje, "Mensaje");

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="tipo">Alerta,modal o ambas</param>

/// <param name="mensaje">mensaje</param>

public void MostrarMensajeError(TipoMensajeEnum tipo, string mensaje)

{

MostrarMensajeError(tipo, mensaje, "Mensaje");

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">mensaje</param>

/// <param name="titulo">titulo del modal</param>

public void MostrarMensajeError(string mensaje, string titulo)

{

MostrarMensajeError(TipoMensajeEnum.MostrarAlertaYModal, mensaje, titulo);

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="tipo">Alerta, Modal o Ambos</param>

/// <param name="mensaje">mennsaje</param>

/// <param name="titulo">titulo</param>

public void MostrarMensajeError(TipoMensajeEnum tipo, string mensaje, string titulo)

{

msjError.MostrarMensaje(tipo, mensaje, titulo);

}

#endregion

#region MensajeConfirmacion

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">mensaje</param>

/// <param name="eventoConfirmacion">evento que se llama si se confirma</param>

public void MostrarMensajeConfirmacion(string mensaje, EventHandler eventoConfirmacion)

{

MostrarMensajeConfirmacion(mensaje, "Mensaje", eventoConfirmacion, CancelacionMensaje);

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">mensaje</param>

/// <param name="eventoConfirmacion">evento que se llama si se confirma</param>

/// <param name="eventoCancelacion">evento que se llama si no se confirma</param>

public void MostrarMensajeConfirmacion(string mensaje, EventHandler eventoConfirmacion, EventHandler eventoCancelacion)

{

MostrarMensajeConfirmacion(mensaje, "Mensaje", eventoConfirmacion, eventoCancelacion);

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">mensaje</param>

/// <param name="titulo">titulo</param>

/// <param name="eventoConfirmacion">evento que se llama si se confirma</param>

public void MostrarMensajeConfirmacion(string mensaje, string titulo, EventHandler eventoConfirmacion)

{

MostrarMensajeConfirmacion(mensaje, titulo, eventoConfirmacion, CancelacionMensaje);

}

/// <summary>

/// Setea el mensaje a mostrar

/// </summary>

/// <param name="mensaje">mensaje</param>

/// <param name="titulo">titulo</param>

/// <param name="eventoConfirmacion">evento que se llama si se confirma</param>

/// <param name="eventoCancelacion">evento que se llama si no se confirma</param>

public void MostrarMensajeConfirmacion(string mensaje, string titulo, EventHandler eventoConfirmacion, EventHandler eventoCancelacion)

{

ConfirmarMensaje = eventoConfirmacion;

CancelarMensaje = eventoCancelacion;

msjConfirmacion.MostrarMensaje(mensaje, titulo);

}

#endregion

/// <summary>

/// Guarda el evento que se debe llamar si se confirma

/// </summary>

public EventHandler ConfirmarMensaje

{

get { return (EventHandler)Session["EventoConfirmacion"]; }

set { Session["EventoConfirmacion"] = value; }

}

/// <summary>

/// Guarda el evento que se debe llamar si no se confirma

/// </summary>

public EventHandler CancelarMensaje

{

get { return (EventHandler)Session["EventoCancelacion"]; }

set { Session["EventoCancelacion"] = value; }

}

#endregion

### Utilizacion en clases propias

//Master de la pagina, para poder mostrar mensajes.

SiteMaster master;

protected void Page\_Load(object sender, EventArgs e)

{

master = (SiteMaster)Master;

}

/// <summary>

/// Mensaje de exito

/// </summary>

public string Exito

{

set

{

if (!string.IsNullOrEmpty(value)

{ master.MostrarMensajeInformacion(value,"Exito");

}

}

}

/// Mensaje de confirmación utilizado dentro de un metodo

master.MostrarMensajeConfirmacion("¿Quiere eliminar la playa " + playa.Nombre + "?", "Confirmar Eliminacion", EliminarPlaya);

/// <summary>

/// Elimina una playa de estacionamiento.

/// </summary>

protected void EliminarPlaya(object sender, EventArgs e)

{

var resultado = gestor.EliminarPlaya(IdPlayaSeleccionada);

if (resultado.Ok)

{

Exito = "La playa fue eliminada correctamente.";

ActualizarGrilla();

}

else

{

Error = resultado.MensajesString();

}

}