Smooth web user experience

Se entiende por **experiencia de usuario,** a la **percepción que le genera al usuario** el hecho de interactuar con un entorno, aplicación o dispositivo... y eso dependerá de todo un conjunto de elementos relativos al producto o dispositivo en cuestión.

Las interfaces de usuario generadas con GeneXus proveen una experiencia de usuario intuitiva y **"smooth"**, **término que significa que fluye de forma fácil y amigable**..

..y las aplicaciones web en su totalidad, son cada vez más performantes debido al mecanismo utilizado para resolver el diálogo cliente-servidor.

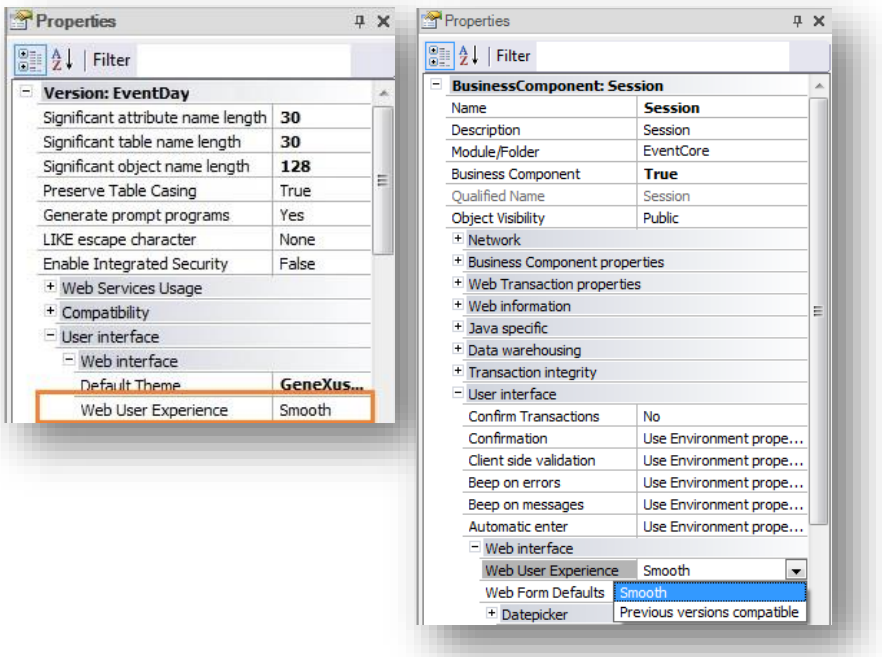
Las características que hacen que la experiencia de usuario sea positiva son:

- **Optimización del esquema de disparo de eventos:** Es decir, la ejecución de eventos es óptima en el sentido que solamente se ejecutan los eventos necesarios. Esto implica que se refresquen únicamente las partes de las páginas que lo requieran… que el usuario reciba feedbacks rápidamente… y que solamente la mínima información necesaria sea enviada al servidor.

- **Óptima navegación entre páginas:** La navegación entre páginas se resuelve utilizando AJAX, y resulta agradable para el usuario un efecto de "esfumado" que hay en la transición. (ese efecto de esfumado por defecto puede personalizarse en una propiedad de 1 theme).

- **Notificaciones al instante:** El usuario puede ser avisado de cualquier evento sin la necesidad de refrescar la página web, ya que es el server el que envía las notificaciones.

Con el fin de que la experiencia del usuario sea amena, como venimos explicando, GeneXus X Ev3 provee la propiedad **Web User Experience** tanto a nivel de la versión como a nivel de los objetos:



y el valor que por defecto que tiene configurado esta propiedad es: "Smooth".

Bien. Uno de los comportamientos que determina la propiedad Web User Experience con valor Smooth, **es cómo se ejecuta el Refresh en una página web cuando se dispara un evento de usuario dentro de la propia página**.

Hace:

- que el evento Refresh no se ejecute en forma implícita cuando se termina de ejecutar un evento de usuario
- y que el evento Start se ejecute una sola vez

Además, cada web component es independiente de la página principal que lo contiene. Esto significa que el Refresh de un web component no afecta al resto de la página.

También habilita **"Single Page Applications"**, lo que significa que navegar hacia una página web que está contenida en la misma Master Page, no provoca el refresh de la página completa en el browser. Solamente actualiza la página contenida en el content place holder.

Veremos a continuación en la práctica, cómo es el nuevo esquema de disparo de eventos en un web panel, al tener la propiedad Web user experience, configurada con el valor=Smooth.

Vamos a GeneXus.

Hemos creado el web panel de nombre SpeakersAndSessions, donde es posible seleccionar una sesión de un combo dinámico y se muestran todos los oradores que participan en dicha sesión.

También se visualiza la cantidad de oradores debajo del grid .

En este ejemplo que corresponde a 1 consulta interactiva con filtro definido, el comportamiento no ha cambiado en absoluto.

Recordemos entonces que al hacer Get, 1ero se dispara el evento Start, ejecutándose el código definido en dicho evento. Veamos que dentro del mismo, tenemos definido que se cargue en la variable &Status, la imagen del símbolo Confirmado/Cancelado.

En segundo lugar se dispara el evento Refresh, que se ejecutará 1 vez, provocando el comienzo de la ejecución de la consulta..

e inmediatamente después vendrá la ejecución del evento Load, una vez por cada registro navegado, porque el web panel del ejemplo tiene tabla base.

Como consecuencia, los datos de cada registro que cumplan con las conditions definidas, se agregarán en el html que se enviará al cliente y se dibujará el grid en la página en el cliente.

Veamos que en el evento Refresh, hemos inicializado la variable &Quantity con 0 y en el evento Load estamos incrementando la misma variable .

¿Y qué sucede cuando el usuario ingresa (o selecciona en este caso) un valor en la variable que se usa para filtrar?

Por defecto, las variables que intervienen en las Conditions, provocan un Refresh automático.

Es decir, la consulta vuelve a ejecutarse y la página vuelve a cargarse. O sea que el programa asociado al web panel debe volver a ejecutarse en el servidor… y en este orden se ejecuta:

- 1ero el evento Start.
- En 2do lugar se leen los datos de pantalla (y allí se obtiene el valor que el usuario dio a la variable &SessionId).
- En 3er lugar el Refresh, e inmediatamente los Loads que correspondan. Así se cargan los oradores que cumplen la condición al archivo.. y terminado el proceso, se devuelve el archivo al navegador del cliente, que dibuja la página con esos datos.

Este comportamiento, como decíamos, es el que ya conocíamos.

Ahora pasemos a ver el caso de la ejecución de un evento de usuario.

La propiedad Web user experience configurada con valor Smooth, provoca **que los eventos de usuario no disparen automáticamente el evento Refresh** (de todos modos, el desarrollador puede decidir ejecutar Refresh explícitamente, utilizando el comando Refresh, cuando lo considere necesario).

Entonces, cuando se ejecuta un evento de usuario, sucede lo siguiente:

- **Se leen la variables presentes en el form**
- **Se ejecuta el evento de usuario que provocó el Post.**

Veamos entonces el siguiente ejemplo:

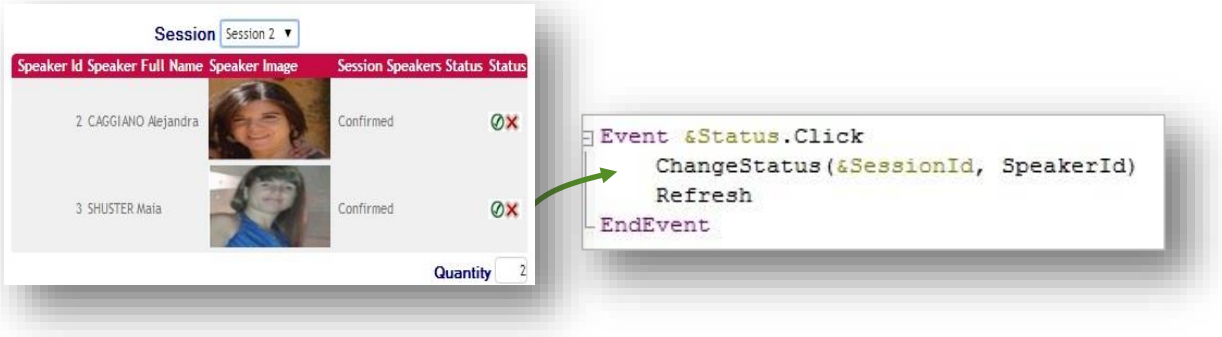
Tenemos otro web panel, su nombre es: SessionStatus, y permite seleccionar una sesión (Session) en la variable &SessionId definida como un combo dinámico para luego mostrar los oradores (Speakers) registrados para esa sesión.

Si bien la sesión tiene inicialmente confirmados a todos sus oradores, es posible que en algún momento surja la necesidad de cancelar a alguno.

Por lo tanto, ofrecemos que haciendo click en la imagen con el Status de un orador, el mismo sea marcado como Cancelado.

Para ello, hemos definido el evento &Status.Click, es decir el evento que se ejecutará cuando el usuario haga click sobre el control &Status, de tipo imagen.

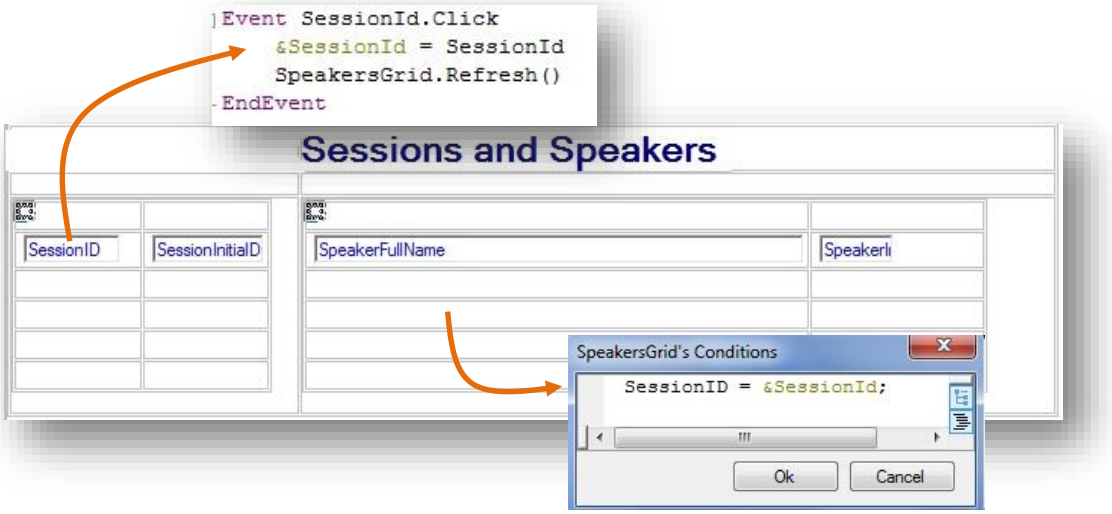
Dentro de este evento, invocamos al procedimiento ChangeStatus, pasándole los parámetros necesarios para que se encargue de realizar la actualización física del estado para el orador de la sesión.



Como sabemos que no se ejecutará un Refresh de forma implícita cuando termine el evento de usuario, hemos incluído al comando Refresh expresamente, a continuación de la invocación al procedimiento.

También debemos saber, que cuando un evento de usuario está asociado a 1 línea del grid, el Refresh solamente **aplicará a esa línea, no cargándose todo el grid nuevamente.**

Veamos ahora cómo es el comportamiento cuando tenemos un web panel con 2 grids



Al ejecutar este web panel desde el cliente, es decir al hacer GET, se ejecutará en este orden:

1. El evento Start

2. El evento Refresh general (que si hay código incluído en el mismo, se ejecutará) y el refresh y load de cada grid consecutivamente.

Como la variable &SessionId está vacía,pues no se ejecutó aún el evento Click que la carga , para el segundo grid no se cargará ninguna línea.

Veamos ahora qué sucede cuando el usuario hace click sobre el identificador de una sesión (SessionId), o sea, cuando se realiza un POST.

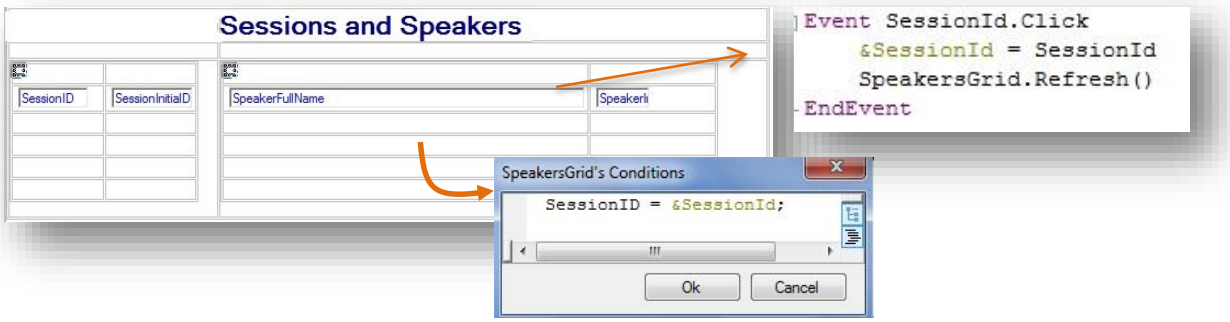
Observemos el evento de usuario SessionId.Click.

Al hacer click sobre 1 SessionId, se realiza la lectura de variables en pantalla (en este ejemplo no hay ninguna), y luego se ejecuta el evento definido.

En este evento se guarda en la variable &SessionId el identificador de la sesión seleccionada, y se solicita explícitamente el Refresh del grid que muestra los oradores (SpeakersGrid).

Es necesario provocar este Refresh en forma explícita ya que automáticamente no se va a realizar.

Como vemos, el SpeakersGrid tiene definida una condición que indica que se filtre por la sesión almacenada en la variable &SessionId.



Una vez ejecutado entonces explícitamente el Refresh del SpeakersGrid, se disparará a continuación el evento Load del mismo, y se visualizarán los oradores registrados en la sesión seleccionada en el SessionsGrid.

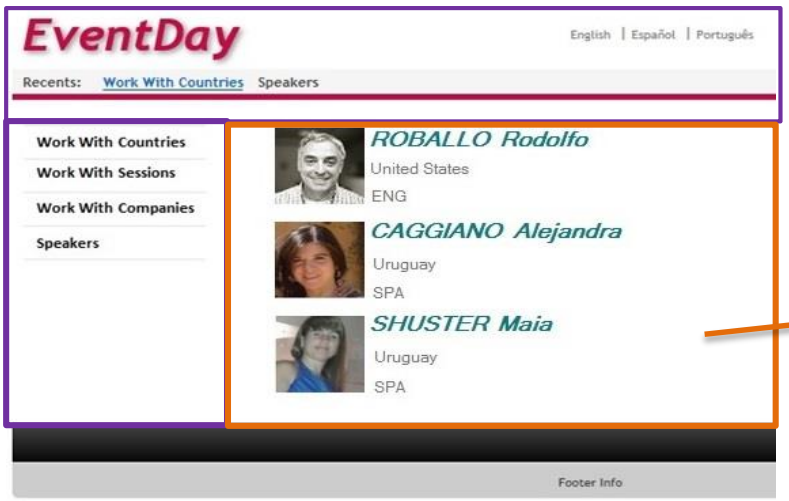
El SpeakersGrid se actualiza, **mientras que el SessionsGrid no se vuelve a cargar**.

En cuanto al evento Start, solamente se ejecutó una vez, al efectuarse el GET del web panel.

Aplicaciones de una página sola

Para finalizar, veamos ahora el impacto que genera el valor Smooth en las aplicaciones de una página sola (o single page applications).

Se le llama aplicaciones de una sola página, a las aplicaciones donde lo que cambia es solamente el contenido del Content place holder.



Navegar hacia una página que es sustancialmente similar a la página actual no debería forzar a refrescar la página entera en el browser, sino solamente actualizar lo necesario.

En GeneXus, los comandos Call y Link se utilizan para navegar entre páginas. Ambos utilizan Ajax y esto hace posible llamar a la nueva página sin la necesidad de refrescar la página en su totalidad. El resultado es que los elementos que ya se encuentran presentes en la página no serán recargados.

Generalmente las aplicaciones GeneXus utilizan una Master Page donde el desarrollador establece los componentes que son comunes a todas las páginas de la aplicación (cabezales, menúes, etc).

De modo que cuando el usuario navegue entre las páginas contenidas en la misma Master Page, no se recargarán los elementos en común.

Además:

La Master Page actualizará sólo los elementos necesarios (como los Recent Links del pattern Work With).