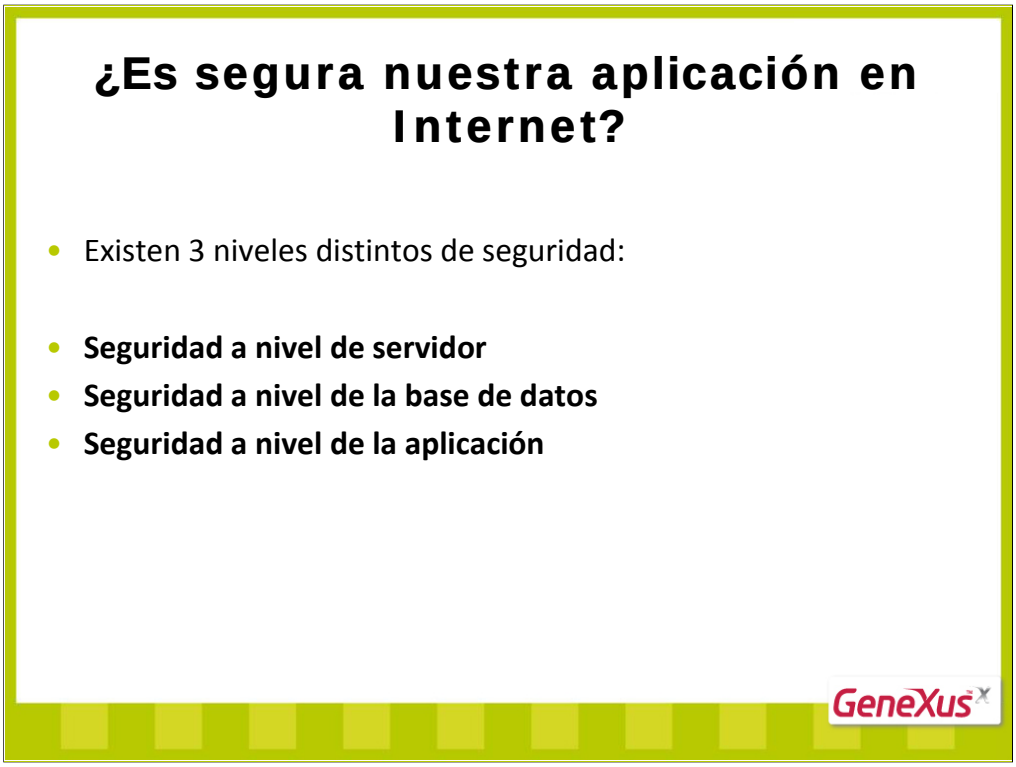
Seguridad



Una pregunta muy importante a la hora de desarrollar una aplicación para Internet, es la siguiente:

¿Es segura nuestra aplicación en Internet?

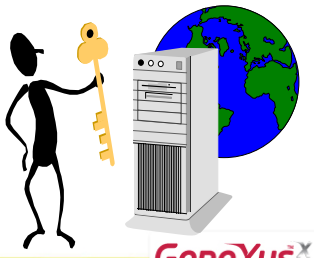
Como la seguridad es un concepto muy amplio, la respuesta a esta pregunta no es sencilla. Sin embargo, dependiendo del tipo de aplicación que se está desarrollando, este punto puede ser crucial.

Se puede decir que al publicar una aplicación en Internet, se tienen 3 niveles distintos de seguridad:

Seguridad a nivel del Servidor Web
Seguridad a nivel de la base de datos
Seguridad a nivel de la aplicación

Seguridad a nivel del Servidor Web

- **Riesgos**:
  - Intercepción de información enviada a través de la red desde el browser al servidor o viceversa
  - Errores o problemas de configuración del servidor Web, que permitan a usuarios no autorizados:
  - Acceder a información confidencial que se encuentra en el servidor
  - Ejecutar comandos que afecten el comportamiento del servidor y les permita ingresar al sistema
- **Soluciones**:
- Protocolos seguros (SSL), firewalls, proxys, proxys reversos, etc. Esto es responsabilidad del adminstrador de la red. No es necesario configurar nada particular en GeneXus.



GeneXus

Tal como lo ha demostrado el crecimiento exponencial de Internet, TCP/IP resuelve muchos problemas de una forma excelente. Sin embargo, TCP/IP no fue diseñado para ofrecer servicios de comunicación seguros. Entonces debemos agregar tecnología adicional para resolver problemas de seguridad.

Desde siempre, existe una única tecnología que provee los principios para resolver estos problemas: CRIPTOGRAFIA.

A nivel de servidor Web, existen servidores seguros, los cuales encriptan cualquier documento enviado. Los Web Panels generados por GeneXus pueden ser ejecutados en un servidor seguro.

La instalación de un certificado digital para un sitio Web y encriptación de datos son simples. Estas son funciones del software que se utilice como servidor Web.

Para activar la encriptación, todo lo que el usuario debe hacer es realizar la solicitud del recurso a través del protocolo https: en lugar de http.

Acceso de usuarios no autorizados

La mayor parte de los riesgos que se presentan al tener un servidor Web público son asumidos por el administrador del sitio. Desde el momento que se instala un servidor Web en un sitio, se abre para la comunidad Internet una “ventana” a la red de área local.

La mayor parte de los usuarios únicamente visitan el sitio, otros, sin embargo intentarán entrar por la “ventana” abierta. Los resultados pueden ser variados, desde el simple descubrimiento que la página principal del sitio Web fue cambiada, hasta el robo de la base de datos que contiene la información de sus clientes.

Es entonces importante que el administrador del sistema defina políticas de seguridad, así como tome las medidas necesarias para evitar el acceso de usuarios no deseados. Estas medidas pueden ser a nivel del sistema operativo (restringir las operaciones que se pueden realizar, limitar los permisos sobre documentos y directorios, deshabilitar servicios no utilizados, etc.), como a nivel del servidor (aislamiento de la red, instalación de firewall).

Existe abundante documentación sobre el tema, que es recomendable leer, para saber que pasos debería seguir al publicar documentos, o aplicaciones en su sitio Web.

Bibliografía

Verisign:

http://www.verisign.com/products/site/faq/40-bit.html#6

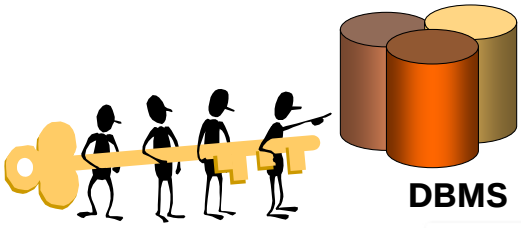
Apache Web Server:

Securing Apache: Step-by-Step

http://www.securityfocus.com/infocus/1694

Seguridad a Nivel de DBMS

- GeneXus protege la información de conexión a la base de datos en los archivos de configuración (la información queda encriptada), y además provee de herramientas para la encriptación de las mismas en caso de querer modificarlas.
- El resto es una tarea del administrador.

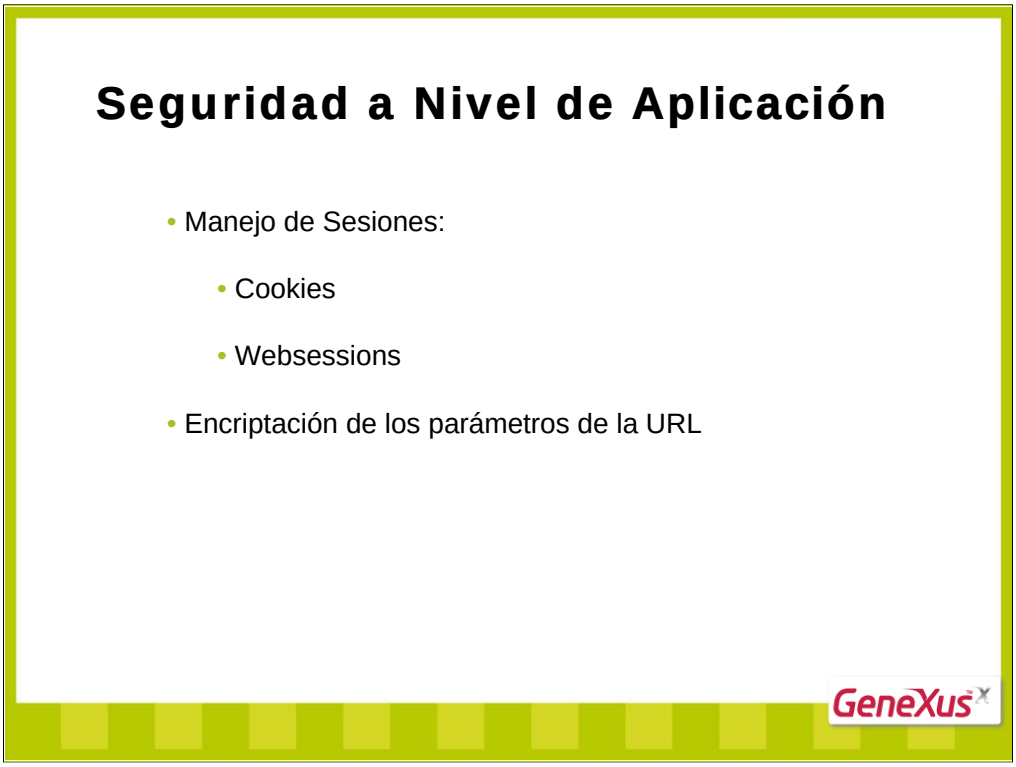


GeneXus

La seguridad a nivel de la base de datos es también un punto sensible a la hora de publicar una aplicación en Internet. Es de fundamental importancia evitar que usuarios no autorizados puedan acceder a los datos corporativos, así como que un usuario autorizado realice operaciones que le deberían estar negadas.

GeneXus protege la información de conexión a la base de datos en los archivos de configuración (la información queda encriptada), y además provee de herramientas para la encriptación de las mismas en caso de querer modificarlas.

El resto es una tarea del administrador.



La seguridad con respecto a la aplicación depende del tipo de aplicación que estemos desarrollando.

La aplicación que desarrollamos puede ser pública, es decir permitir el acceso a todos los usuario, o se puede restringir el mismo. Por ejemplo, si se tiene una aplicación que tiene información del cliente y el mismo puede modificarse sus datos o debe identificarse para realizar una operación.

La dificultad con la que nos enfrentamos es que los parámetros que recibe un Web panel es visible en el browser, por lo que sabiendo el nombre del Web Panel, se podría acceder a los datos de un usuario sin pasar por un Web Panel de validación.

Es por esta razón que se aconseja el uso de sesiones cuando se tienen aplicaciones en Internet que requieren un acceso restringido.

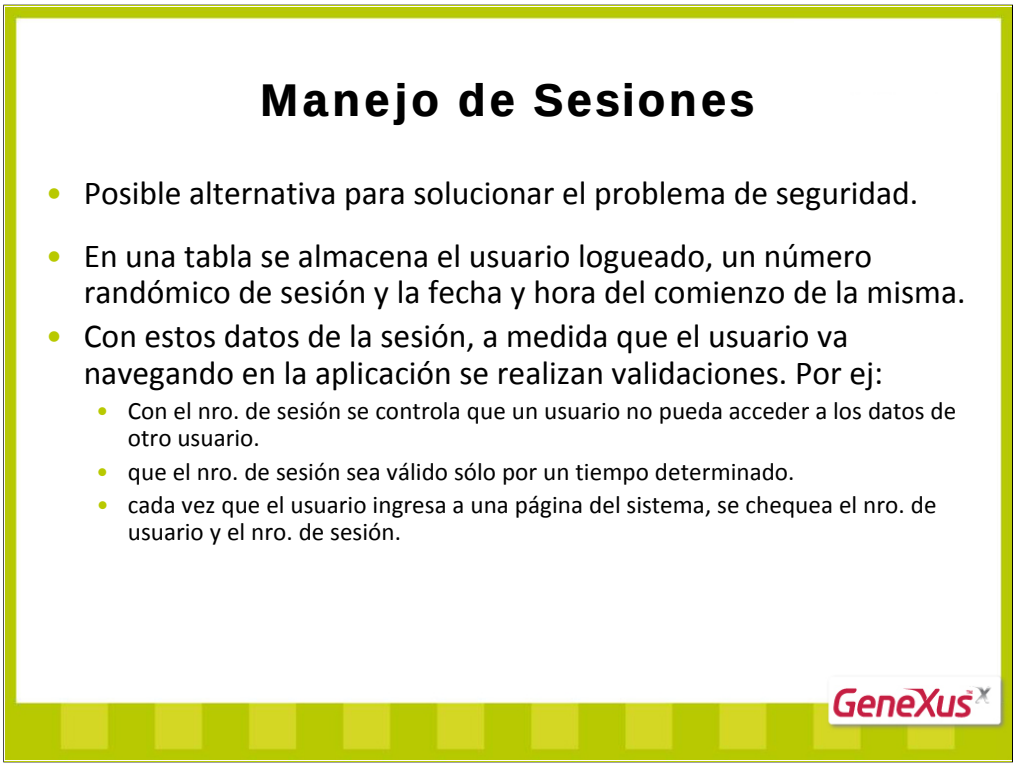
El uso de sesiones se compone de una transacción que almacena el usuario logueado, un número randómico de sesión y la fecha y hora del comienzo de la misma. Mediante este número de sesión se controla que un usuario no pueda acceder a los datos de otro usuario.

Cada vez que el usuario ingresa a una página del sistema, se chequea el número de usuario y el número de sesión. Si estos no se corresponden con los datos almacenados, se muestra una pantalla indicando el error y se vuelve al usuario a la página donde se pide login y password nuevamente.

Este número de sesión es válido por un tiempo determinado que se configura en el procedimiento. Si el usuario está loggeado por más de dicha cantidad de tiempo, se muestra la misma pantalla de error.

Hay dos niveles en cuanto a la seguridad, una se trata de asegurar el acceso a la aplicación, a los datos (esto se debe controlar en el desarrollo de la aplicación Web); y otra es que la aplicación generada, sin necesidad de que el desarrollador GeneXus se preocupe, contempla y cubre los posibles agujeros de seguridad que puede tener una aplicación Web, por ejemplo SQL injection, por ejemplo los riesgos de seguridad que acarrea el uso intensivo de Ajax.

http://wiki.gxtechnical.com/commwiki/servlet/hwikibypageid?6236

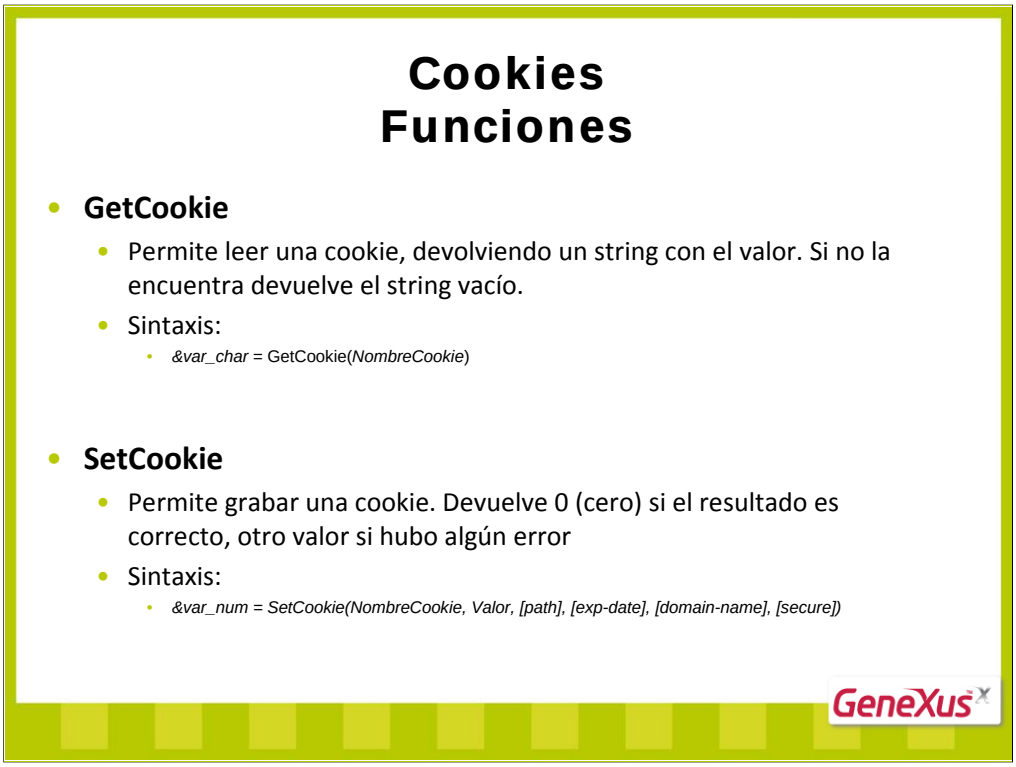


Manejo de sesiones

Una de las posibles alternativas para solucionar el problema de seguridad mencionado anteriormente, es el uso de sesiones.

El uso de sesiones se compone de una Transacción por ejemplo: “SECURITY” que almacena el usuario logueado, un número randómico de sesión y la fecha y hora del comienzo de la misma. Mediante este número de sesión se controla que un usuario no pueda acceder a los datos de otro usuario.

Cada vez que el usuario ingresa a una página del sistema, se chequea el número de usuario y el número de sesión. Si estos no se corresponden con los datos almacenados en la tabla SECURITY, se muestra una pantalla indicando el error y se vuelve al usuario a la página donde se pide login y password nuevamente. Este número de sesión es válido por un tiempo determinado que se configura en el procedimiento. Si el usuario está logueado por más de dicha cantidad de tiempo, se muestra la misma pantalla de error.



Funciones de cookies

El objetivo es proveer funciones que permitan leer y grabar cookies desde objetos Web generados por GeneXus.

¿Qué son las cookies?

Las cookies son archivos pequeños que se graban desde un Web site en las máquinas de los clientes. Los programas CGI o cualquier aplicación que corre en un servidor, puede leer o grabar las cookies en el cliente.

El uso más común de las cookies es la identificación de usuarios. Cuando un usuario se registra en un Web site (Portal o E-Store), el sitio graba una cookie en la máquina del cliente con la identificación del cliente. De este modo la próxima vez que el cliente visite este sitio, intenta leer la cookie y si la misma existe usa el valor de la cookie para identificar el usuario y recuperar sus preferencias desde una base de datos.

También existen otros usos de las cookies, como rotación de contenido (especialmente avisos), mantener estado de una aplicación, etc. Incluso se pueden usar como método de almacenar el "carrito de compras" de modo que la información del mismo quede en la máquina cliente y se mantiene entre conexiones.

Como se menciona anteriormente, generalmente se usa una cookie para identificar el usuario (en algunos casos, una para la sesión, en otros para el usuario), aunque se podrían poner todos los valores de las preferencias en cookies. Lo ideal es tener una clave que viaje y con esta clave leer la información del usuario. De este modo la información del usuario no viaja hacia el cliente, ni está en la URL. (Ej. tarjetas de crédito, nombre, dirección) simplemente permanece en el servidor.

Hay que tener en cuenta que existe un límite en cuanto a la cantidad de cookies que el cliente puede aceptar. El máximo son 300 cookies en total por cliente (para todos los servidores juntos por cada browser/cliente) y 20 cookies por servidor o dominio lo cual quiere decir que si una aplicación graba más de 20 cookies las últimas van a borrar los valores de las primeras. Además existe un límite de tamaño de 4K por cookie, si una cookie supera ese límite es truncada.

El usuario puede preferir no grabar la cookie permanentemente (por Ej. si está accediendo desde una máquina pública como podría ser un cybercafe) o incluso puede deshabilitar el uso de cookies, por lo cual esta no debe ser la única manera de identificar al usuario, sino que se debe poder usar un método alternativo en caso de que el browser no soporte o no tenga habilitado el manejo de cookies.

Otra particularidad es que el lugar donde se almacenan las cookies (al menos en Windows) depende del browser, por lo que si un usuario tiene más de un browser, cada uno tendrá un conjunto independiente de cookies.

El ciclo de vida de una cookie es como sigue:

1. El usuario se conecta a un servidor que por alguna razón quiere grabar una cookie.
2. En la respuesta (HTML Headers), se indica el nombre y valor de la cookie a grabar, así como otros valores (el más relevante es la fecha de expiración).
3. El browser recibe la respuesta y, si el valor de la fecha de expiración es en el futuro, la graba; en caso contrario busca una con ese nombre y la borra.
4. Cada vez que el usuario se conecte a una URL de este dominio el browser enviará al servidor las cookies que se hayan grabado desde el dominio y no hayan expirado.
5. Una vez pasada la fecha de expiración, las cookies se borran.

Para obtener más información técnica sobre cookies y su uso :

http://www.cookiecentral.com

Funciones:

Se dispone de las siguientes funciones, las que pueden ser utilizadas en cualquier objeto GeneXus, el resultado tiene sentido únicamente si dicho objeto fue llamado en forma directa o indirecta por un objeto Web o está ejecutando en un ambiente Web.

GETCOOKIE

Permite leer una cookie, devolviendo un string con el valor. Si no encuentra la cookie (o no la puede leer por estar deshabilitada) devuelve el string vacío.

Sintaxis

&var_char = GetCookie(*NombreCookie*)

NombreCookie: carácter

&var_char: carácter

SETCOOKIE

Permite grabar una cookie. Devuelve 0 (cero) si el resultado es correcto, otro valor si hubo algún error.

Sintaxis

&var_num = SetCookie(*NombreCookie*, *Valor*, *[path], [exp-date], [domain-name], [secure]*)

Los parámetros entre paréntesis rectos son opcionales. Si alguno de los parámetros va nulo se asume el default.

&var_num: variable numérica

NombreCookie: Carácter. Indica el nombre de la cookie.

Valor: Carácter. Valor a almacenar.

Path: Carácter. Camino que indica para qué Web Panels la cookie es válida. Si no se especifica, la cookie es válida para los Web Panels que están en el mismo directorio que el que la grabó, o en directorios subordinados. Si se indica "/" la cookie será válida para todo el dominio.

Exp-date: Date/Datetime. Indica la fecha de expiración de la cookie. Si no se especifica, la misma expirará cuando se cierre la sesión en el browser.

Domain-name: Carácter. Dominio donde es válida la cookie. Por defecto es el dominio desde donde se creó.

*Secure:*Numérico. Si está en 1 la cookie se trasmite solamente si la conexión es segura (HTTPS). Si está en 0 se trasmite siempre.

Observaciones

Las funciones mencionadas anteriormente siempre devuelven 0. La única forma de detectar si una cookie fue almacenada es leerla. En consecuencia no se puede obtener la configuración del browser.

Ejemplos

Algunos ejemplos sencillos sobre cómo grabar una cookie son:

&Op= SetCookie("ID_USER", Str(UsrId), "/", CTOD("01/01/2002"))

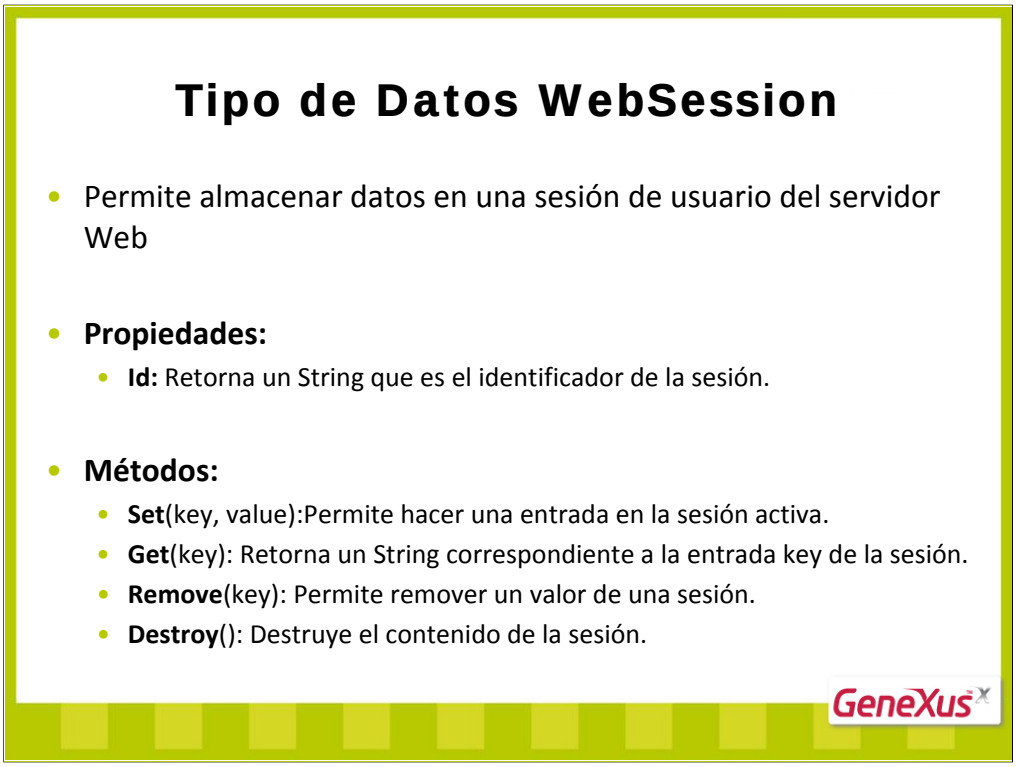
Aquí se está grabando una cookie -válida para todo el dominio- de nombre ID_USER con el valor correspondiente al atributo UsrId y que expirará el 1° de Enero de 2002.

&OK= SetCookie("SESSION_ID_GX", &StrSession, "", Nullvalue(&Fecha))

Aquí se está grabando una cookie -válida para los Web Panels de la misma aplicación -de nombre SESSION_ID_GX- con el valor correspondiente a la variable &Strsession. La cookie expirará al cerrar la sesión del browser.

&Op= SetCookie("USR_PAIS", "UY", "/", ADDYR(&Today, 1), "otrodom.artech.com.uy", 1)

Aquí se está grabando una cookie -válida para todo el dominio 'otrodom'- de nombre USER_PAIS con el valor UY y que expirará exactamente dentro de un año.



WebSession es un tipo de datos de GeneXus que permite almacenar datos en una sesión de usuario del servidor Web. De esta manera se pueden tener variables globales, accesibles mientras la sesión esté activa. Los mismos aplican a los siguientes objetos: Transacciones, Web Panels, Procedimientos.

Los servidores Web permiten manejar el concepto de sesión. Una sesión se identifica por una clave única, que se mantiene mientras el usuario continúe en el sitio Web. El objeto WebSession permite almacenar información que será visible desde cualquier objeto Web dentro de la sesión activa como si fueran variables globales al sitio.

Para utilizar el objeto WebSession, se debe definir una variable de este mismo tipo y aplicarles los métodos y propiedades adecuados.

Propiedades

Id

Esta propiedad retorna un String que es el identificador de la sesión.

Ejemplo:

&Iden = &Session.Id

Métodos

Set(*key, value*)

Permite hacer una entrada en la sesión activa. Key y value deben ser del tipo String.

Ejemplo:

&Session.set(“user”, &User)

Get(*key*)

Retorna un String correspondiente a la entrada key de la sesione. En caso de que la clave no exista retorna el String nulo.

Ejemplo:

&User = &session.get(“user”)

Remove(*key*)

Permite remover un valor de una sesión.

Ejemplo:

&Session.remove(“user”)

Destroy()

Detruye el contenido de la sesión. Es recomendable utilizarlo cuando el usuario hace un “logout”, si es que existe este concepto en la misma.

Ejemplo:

&Session.destroy()

Consideraciones Generales

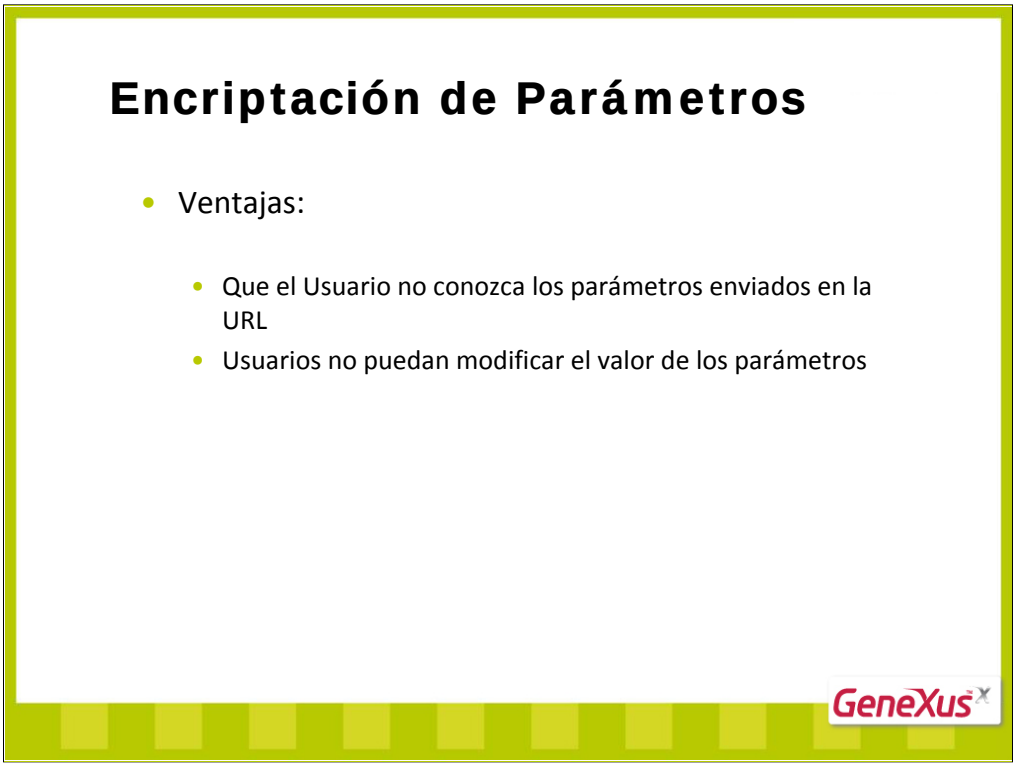
El ID de la sesión se guarda en una cookie en el cliente, aunque esto es transparente para el programador.

La validez de la Websession es similar a la validez de las cookies que solo valen por la sesión. Esto quiere decir que si se abre una instancia nueva del browser, se pierde la sesión, pero si se abre en una ventana nueva se mantiene.

Los datos y el ID de una sesión son diferentes para cada generador. Esto implica que no puedo hacer un link de un Web Panel .NET a un Web Panel Java y mantener los valores de la sesión.

Si no se ejecuta el “destroy()”, el servidor Web destruye la sesión cuando ha pasado un determinado tiempo desde la última vez que se utilizó. Esto depende del servidor Web/plataforma, y se configura de forma nativa en cada una.

La forma en que se almacena la información de la sesión depende de la plataforma. Básicamente, si la sesión se almacena de forma local al servidor Web, solo es visible en ese servidor Web.



Los objetos Web (Web Panels, Transacciones Web) generados con GeneXus, permiten visualizar los parámetros que se pasan entre los objetos en la barra de dirección del navegador.

Esto hace que, si se pasa información reservada como parámetro entre objetos Web (Número de cliente, por ejemplo), las aplicaciones no sean muy confiables en cuanto a la seguridad, porque un usuario podría simplemente cambiar el valor de dicho parámetro en la URL y disponer de información sobre la que no debería tener acceso.

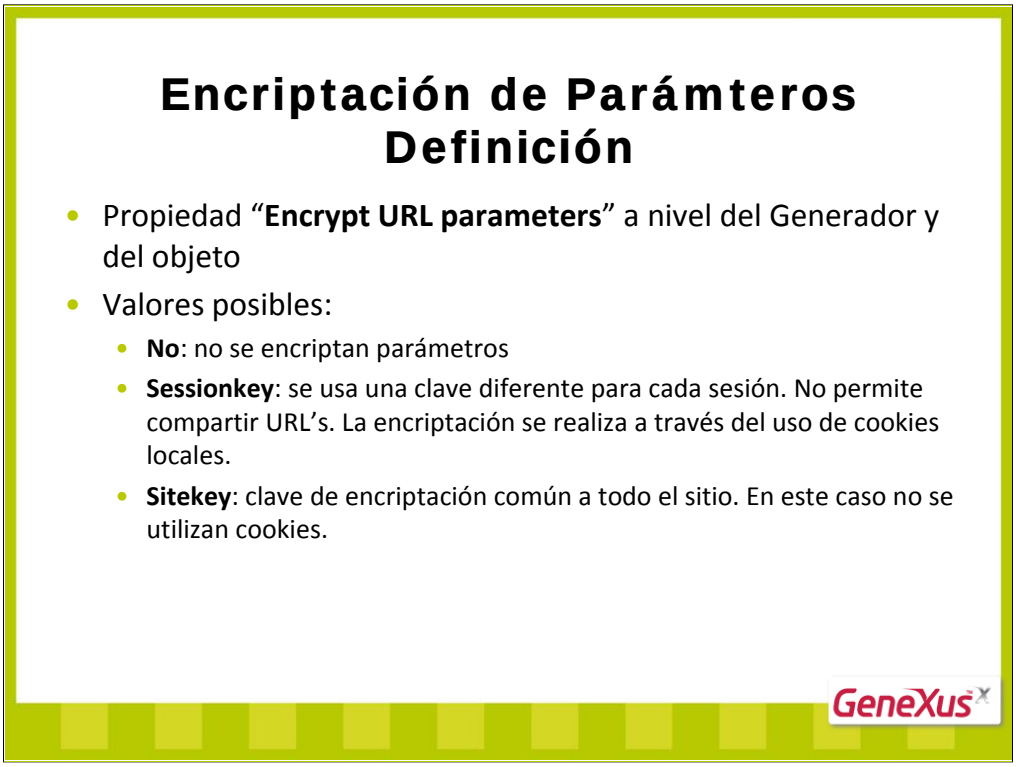
Es por eso que se hace necesario pasar los parámetros sin que el usuario de la aplicación los conozca o sea **encriptar** los parámetros.

Para poder realizar la encriptación de parámetros en objetos Web se implementaron funciones estándar que contienen las funciones básicas de encriptación y algunas funciones adicionales (las que requieren manejo de parámetros y cookies).

Con respecto al diseño de los objetos la encriptación de parámetros no implica ningún cambio, se programan de la misma forma que hasta el momento.

Las ventajas del uso de la encriptación de parámetros son:

- Que los usuarios finales no sepan el o los datos que van en los parámetros
- Que los usuarios finales no puedan modificar el o los datos que van en los parámetros



Se agrega la preferencia **Encrypt URL Parameters** a nivel de modelo, en el grupo "Web Information" y también a nivel de objeto.

Para la preferencia a nivel de modelo los valores posibles son:

No: Indica que No se van a encriptar los parámetros que van en la URL de los objetos Web, siendo éste el valor por defecto.

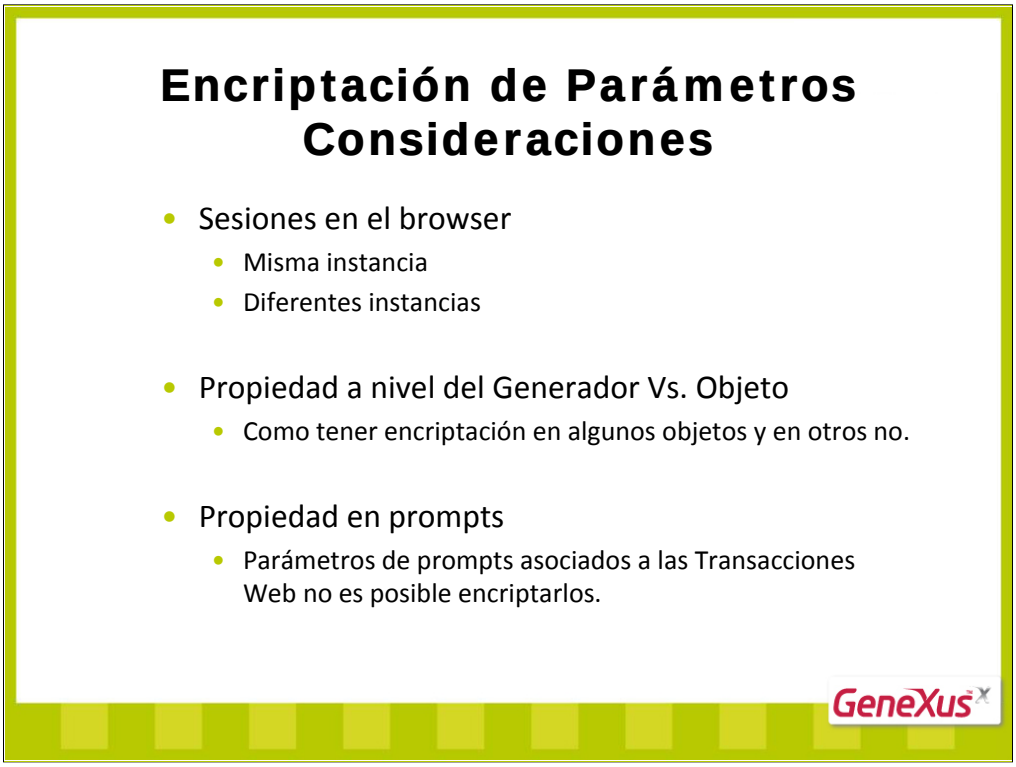
Session Key: Indica que se van a encriptar los parámetros que van en la URL, utilizando una clave diferente para cada sesión. La encriptación se realiza a través del uso de cookies locales.

Este valor ofrece un nivel de seguridad mayor, pero no permite compartir URLs. Esto significa que no es posible para un usuario X enviar una URL que tenga parámetros a otro usuario Y, ya que en este caso la URL no va a funcionar porque se necesita la cookie correspondiente para la desencriptación.

Site Key: Se encriptan los parámetros que van en la URL de los objetos Web, pero la clave de encriptación va a ser la misma para todo el sitio.

En este caso no se utilizan cookies. Esto da un nivel de seguridad menor pero facilita el traspaso de links.

La propiedad a nivel de objeto, además de los valores mencionados tiene el valor "Use model's preference value". Este valor indica que se va a tomar el valor de la preferencia del modelo para realizar la encriptación de ese objeto. Este es el valor por defecto.



Sesiones en el browser:

Una sesión del navegador queda determinada por una instancia del mismo. Por ejemplo, si en un máquina se ejecuta el navegador de Internet y a partir de ese navegador se abre otra sesión (a partir de la opción de menú File/New/Window o a partir de un link), ambas sesiones pertenecen a la misma instancia del navegador.

En cambio, si se abre una sesión del navegador, y luego se ejecuta nuevamente el exe del navegador para abrir una nueva ventana, las dos ventanas no pertenecen a la misma instancia.

Con esto, si se ejecutan objetos Web, y se configuró la preferencia del modelo (o la propiedad a nivel de objeto) con el valor “Session Key”, la cookie que se defina para guardar este valor va a funcionar en las sesiones del navegador que compartan la misma instancia.

Preferencia a nivel del Generador Vs. Propiedad a nivel de objeto:

Los valores “Session Key” y “Site Key” a nivel del Generador, determinan que todos los llamados entre objetos Web se harán con parámetros encriptados. Para tener unicamente las llamadas entre algunos objetos con parámetros encriptados se debe indicar el valor “No” en la preferencia a nivel de modelo y el valor “Session Key” o “Site Key” en el objeto Web que lo requiera.

Si se tienen valores configurados para el Generador y la propiedad a nivel de objeto para encriptar parámetros, ésta última tiene prioridad sobre la primera.

Propiedad “Encrypt URL Parameters” en Prompts:

Los parámetros de los prompts asociados a las Transacciones Web no es posible encriptarlos. Esto es porque el llamado a los prompts se realiza desde el cliente y para realizar la encriptación se debe ir al servidor.

Ejemplos

Si se tiene un Web Panel que recibe parámetros y no se utiliza la encriptación de parámetros, la URL correspondiente al Web Panel será del estilo:

http://localhost/HINGRESO_WebObj/hdospar.asp?2,3

Si, en cambio se utiliza encriptación de parámetros (propiedad “Encrypt URL Parameters” en “Session Key” ó “Site Key”), la misma URL se generará de la siguiente forma:

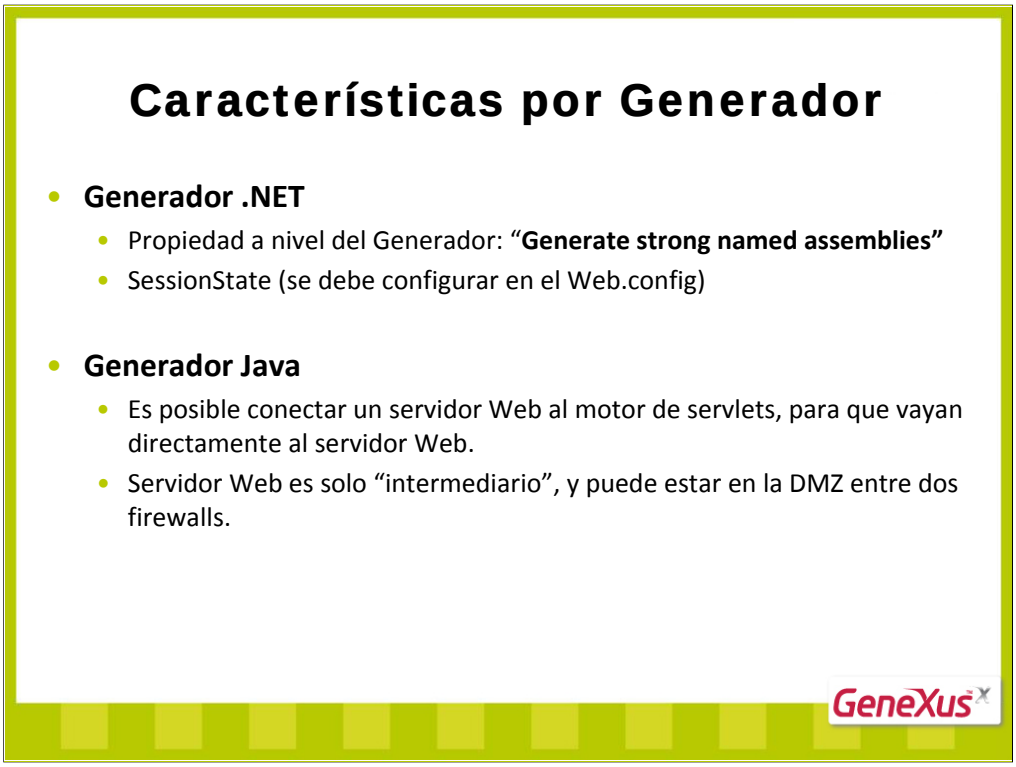
http://localhost/HINGRESO_WebObj/hdospar.asp?lQ/tK1lefxCZMVoXrnmrTQ==

Comparación alternativas

- **Cookies**
  - Ventajas:
    - No es necesario el pasaje de parámetros entre los objetos Web, ya que los valores se graban en la cookie.
    - Si las cookies no son temporales, se puede saber si el usuario ya accedió anteriormente a la aplicación.
  - Desventajas:
    - Tienen limitaciones en cuanto al tamaño máximo, cantidad de cookies que se pueden grabar. Se tiene dependencia de la configuración del browser.
- **Tipo de datos Web Session**
  - Ventajas:
    - Presenta las mismas ventajas que el uso de cookies.
    - La información se almacena en el servidor Web, por lo cual, se están usando los recursos del servidor, en contrapartida de usar los recursos del PC cliente.
  - Desventajas
    - Depende de la configuración del browser (al igual que las cookies). No permite compartir datos si los objetos son generados en diferentes lenguajes. (plataformas) y tampoco si son la misma plataforma pero físicamente diferentes equipos.
    - La destrucción de los datos depende del browser utilizado.

Comparación alternativas

- **Encriptación de los parámetros de la URL**
  - Ventajas:
    - No se requiere programación alguna, simplemente se modifica el valor de la propiedad.
  - Desventajas:
    - El tamaño de la URL aumenta al ser encriptada, por lo tanto si el pasaje de parámetros es importante, se puede llegar al máximo aceptado por el browser. Requieren del uso de cookies.



Características por Generador

Además de los temas vistos anteriormente, existen algunas características particulares para cada generador que permiten brindar mayor seguridad a las aplicaciones.

Generador .NET

Para el Generador .NET existen las siguientes propiedades:

- Generate strong named assemblies (propiedad del Generador)
- SessionState (se debe configurar en el Web.config)

La primera, determina que los objetos main y/o dlls (assemblies) generados tengan un nombre único o no. Esto permite acceder a un conjunto de ventajas importantes que provee el Framework .NET, como ser la configuración de seguridad para el assembly.

En cuanto a la propiedad SessionState, se refiere a la implementación del manejo de sesiones (Tipo de dato Websession) el generador utiliza el HttpSessionState provisto por el Framework. Por lo que existen diversos modos de almacenar la session state. Con lo cual es posible configurar por ejemplo, el tiempo que viven las variables de sesión.

Generador Java

Con el Generador Java existe la posibilidad de conectar un servidor Web al motor de servlets, de forma de que los requerimientos vayan directamente al servidor Web (al puerto 80). Luego éste los redirecciona al motor de servlets (puerto 8080 por defecto), quien es que los resuelve, y manda la respuesta al servidor Web. El motor de servlets ejecuta los servlets, y es quien establece la conexión a la base de datos. El servidor Web solo sirve de "intermediario" como una capa más de seguridad, pero allí no se ejecuta nada. Este servidor puede estar en el DMZ entre dos firewalls, uno que lo une a Internet, y otro a la red local.