Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas

Diseño Orientado a Objetos

Comunicación entre cliente/Servidor

Profesor: Miguel Salazar
Alumno: Héctor Iván Arrieta Jaime
Matrícula: 1604738

La arquitectura cliente-servidor

Es un modelo de diseño de software en el que tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

Algunos ejemplos de aplicaciones computacionales que usen el modelo cliente-servidor son el Correo electrónico, un Servidor de impresión y la World Wide Web

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

Cookies

En Internet se han topado alguna vez en su camino con el término "cookies", y desde hace no mucho tiempo es casi imposible no haberse encontrado con un sitio en donde se muestre un aviso informando al internauta de que se almacenarán cookies en su navegador. Muchos probablemente cerraron el mensaje y no le dieron importancia a las "galletitas" que aceptaron.

Dado que nunca se está a salvo de los problemas de seguridad y privacidad que conlleva el uso de Internet, es importante saber qué es lo que estamos aceptando cuando decimos que no nos importa que un sitio web guarde cookies en nuestro navegador.

Una *cookie* es un archivo creado por un sitio web que contiene pequeñas cantidades de datos y que se envían entre un emisor y un receptor. En el caso de Internet el emisor sería el servidor donde está alojada la página web y el receptor es el navegador que usas para visitar cualquier página web.

Su propósito principal es identificar al usuario almacenando su historial de actividad en un sitio web específico, de manera que se le pueda ofrecer el contenido más apropiado según sus hábitos. Esto quiere decir que cada

vez que se visita una página web por primera vez, se guarda una *cookie* en el navegador con un poco de información. Luego, cuando se visita nuevamente la misma página, el servidor pide la misma *cookie* para arreglar la configuración del sitio y hacer la visita del usuario tan personalizada como sea posible.

Estas *cookies* pueden tener una finalidad simple, como saber cuándo fue la última vez que el usuario entró a cierta página web; o algo más importante como es guardar todos los artículos puestos en el carrito de compras de una tienda, una acción que se va guardando en tiempo real.

Seguridad en la Aplicación

Control de acceso

Un aspecto muy importante de una aplicación web es el control de acceso de los usuarios a zonas restringidas de la aplicación.

Aquí intervienen dos conceptos:

- Autentificación
- Autorización
- Autentificación

Es el proceso de determinar si un usuario es quien dice ser Esto se puede hacer de varias maneras. Algunas de ellas son:

- Autentificación HTTP básica
- Autentificación basada en la aplicación
- Autentificación HTTP básica

Ventajas:

Es muy simple de implementar

Se pueden fijar restricciones de acceso por usuario y contraseña o por otros conceptos como por ejemplo el dominio o dirección IP de la máquina

Inconvenientes:

- Los datos viajan por la red sin encriptar
- No se puede hacer logout, la única forma es cerrar el navegador
- No hay control sobre el diseño de la ventana de diálogo.
- Autentificación basada en la aplicación
- La propia aplicación puede implementar un mecanismo de autentificación que implica la presentación de un formulario para que el usuario introduzca sus credenciales y el uso de una base de datos para verificar la corrección de éstas
- Es más costosa pero más flexible ya que permite establecer diferentes permisos y niveles de acceso en función del usuario que solicita la autentificación

Passwords

Siempre que se utilizan passwords para autentificar usuarios hay que seguir unas recomendaciones:

• Restringir los valores para los nombres de usuarios. Los que representan nombres reales suponen dar pistas a los atacantes.

• Almacenar los passwords de forma segura, protegiendo el acceso a la base de datos.

• Seguir reglas de seguridad para su elección.

• Bloquear una cuenta cuando se detecta un número determinado de intentos de acceso incorrectos.

• Actualizar los passwords periódicamente y mantener un histórico para evitar repeticiones.

Sesiones

• Una vez que el usuario se ha autentificado introduciendo su nombre de usuario y su clave, es preciso mantener esta autentificación en cada conexión subsiguiente

• Para evitar tener que mostrar nuevamente la ventana de autentificación se recurre habitualmente al uso de sesiones, un mecanismo que permite mantener el estado entre diferentes peticiones HTTP

• Una vez autentificado, al usuario se le asigna un identificador de sesión

• Este identificador acompañará invisiblemente a cada petición del usuario, con lo cual se garantizará que la petición proviene de un usuario previamente autentificado

• El identificador de sesión se suele almacenar en la propia máquina del cliente, mediante una cookie

• Sólo se debe almacenar el identificador de la sesión; cualquier otro dato del usuario se almacenará en el servidor

• La gestión de las sesiones es responsabilidad del programador.

• Normalmente los lenguajes de servidor disponen de funciones diseñadas específicamente para ello.

Un buen sistema de gestión de sesiones debe:

• Establecer un tiempo límite de vida para la sesión.

• Regenerar el identificador de sesión cada cierto tiempo.

• Detectar intentos de ataque de fuerza bruta con identificadores de sesión.

• Requerir una nueva autentificación del usuario cuando vaya a realizar una operación importante.

• Proteger los identificadores de sesión durante su transmisión.

• Destruir la cookie al finalizar la sesión para evitar el acceso de otro usuario en un entorno público.

• Autorización

• Es el acto de comprobar si un usuario tiene el permiso adecuado para acceder a un cierto fichero o realizar una determinada acción, una vez que ha sido autentificado

• Modelos para el control de acceso:

• Control de Acceso Discrecional: se basa en la identidad de los usuarios o su pertenencia a ciertos grupos. El propietario de una información puede cambiar sus permisos a su discreción

• Control de Acceso Obligatorio: cada pieza de información tiene un nivel de seguridad y cada usuario un nivel de acceso, lo cual permite determinar los permisos de acceso de cada usuario a cada pieza de información

• Control de Acceso Basado en Roles: cada usuario tiene un rol dentro de la organización y en función de él unos permisos de acceso

Validación de datos de entrada

El problema más frecuente que presentan las aplicaciones web es no validar correctamente los datos de entrada. Esto da lugar a algunas de las vulnerabilidades más importantes de las aplicaciones, como la Inyección SQL, el Cross-Site Scripting y el Buffer.

Overflow

 Fuentes de entrada – cadenas URL

 La aplicación debe chequear el valor recibido aunque proceda de una lista desplegable con unos valores predefinidos, ya que el usuario ha podido modificar manualmente la URL, este problema se da también en los hipervínculos que incluyen parámetros

 Siempre que se envíen datos sensibles hay que acompañarlos de un identificador de sesión y comprobar que el usuario asociado a la sesión tiene acceso a la información requerida.

Fuentes de entrada - cookies

Es un método habitual de mantener el estado o almacenar preferencias del usuario. Pueden ser modificadas por el cliente para engañar al servidor. El peligro dependerá de lo que se almacene en la cookie.

Por ejemplo, la cookie Cookie: lang=en-us; ADMIN=no; y=1; time=10:30GMT; puede ser modificada fácilmente por: Cookie: lang=en-us; ADMIN=yes; y=1; time=12:30GMT;

Lo mejor es almacenar en la cookie únicamente el identificador de sesión, manteniendo la información relevante en el servidor.

Fuentes de entrada - formularios

Los formularios pueden ser modificados para enviar lo que el usuario desee. Basta con guardar la página, modificar el código y recargarlo en el navegador. Las limitaciones impuestas en el propio formulario se pueden saltar perfectamente.

Ejemplo:

<input type="text" name="titulo" maxlength="100">

• Este elemento podría modificarse así:

<input type="text" name="titulo" maxlength="100000">

• con el consiguiente riesgo para la aplicación si el valor no se chequea

adecuadamente