Capítulo 8
Servicios en tiempo Real

Objetivo

Al finalizar el capítulo, el alumno:

- Desplegar conexiones persistentes con Signal-R.
- Desplegar Hubs con Signal-R.
- Consumir conexiones de Signal-R haciendo uso de JQuery.

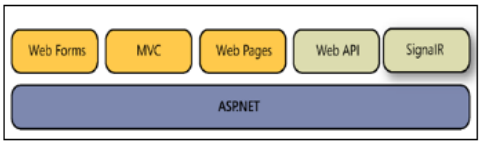
Temas

1. Introducción a SignalR
2. Tiempo real e Introducción a SignalR, transporte, conexiones y Hubs.
3. Hubs API: Server y JavaScript Client.
4. Entender y manejar el tiempo de vida de las conexiones en SignalR.
5. Trabajando con Groups en SignalR.
6. Mapeando usuario de SignalR a conexiones.
7. Seguridad en SignalR.

1. Introducción a SignalR

Introducción a SignalR

- SignalR 2.0 es un framework que facilita la construcción aplicaciones interactivas, multiusuario, y web en tiempo real (aunque no es sólo para aplicaciones web), haciendo uso de técnicas asíncronas para lograr inmediatez y el máximo rendimiento.



- SignalR Core 2.0 aun en desarrollo. Actualmente esta en Preview y planificado para finales del 2017:
  https://github.com/aspnet/Home/wiki/Roadmap

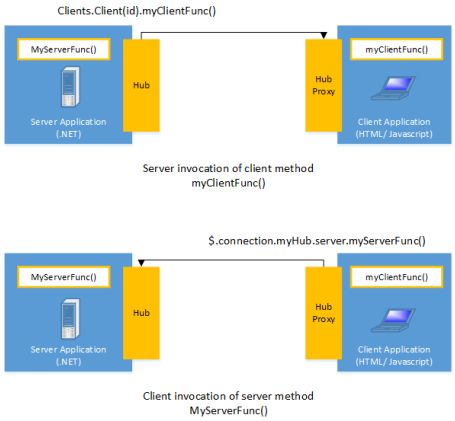
8 - 10

¿Qué es SignalR?

SignalR es una librería creada para desarrolladores de ASP.NET que facilita el proceso de añadir funcionalidades en tiempo real a aplicaciones web, es decir, la capacidad del servidor de enviar información al cliente sin tener que esperar a que el cliente realice una petición.

Ejemplos de aplicaciones web en tiempo real pueden ser chats, monitorización, aplicaciones colaborativas o incluso juegos.

¿Y cómo funciona?



SignalR ofrece una API para crear llamadas remotas (RPC) desde el servidor a funciones Javascript existentes en el cliente. No tenemos que encargarnos de gestionar la conexión, ya que SignalR lo hace automáticamente. Podemos enviar mensajes a un cliente específico o a todos los conectados al servidor.

SignalR se basará en Websocket cuando esté disponible. Cuando no sea el caso, utilizará otra tecnología disponible para funcionalidades en tiempo real, como puede ser Server Sent Requests, Forever Frame o Ajax long polling. En función del navegador en el que se ejecute la aplicación, se escogerá un método de transporte u otro, pero es un proceso totalmente transparente para nosotros, sin que sea necesario tener conocimientos de estos transportes ni de cómo debemos implementar nuestro código en función del navegador que utilicemos.

Para implementar una aplicación SignalR tendremos lo que denominamos un Hub y un Hub proxy.

Un Hub/Hub proxy no es más que un medio que permite al cliente realizar llamadas a métodos servidor y viceversa, como si esos métodos se estuvieran llamando en local.

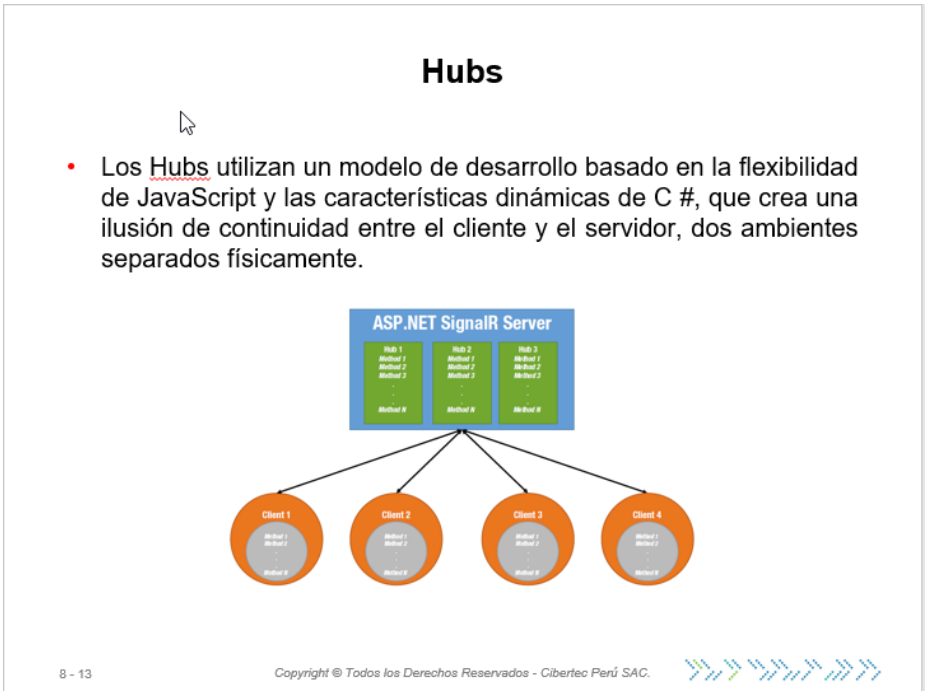
El Hub estará en la parte del servidor, y expondrá los métodos disponibles al cliente. El Hub proxy estará en el lado del cliente, y expondrá los métodos disponibles al servidor.

En el lado del servidor, SignalR se puede ejecutar en cualquier sistema operativo capaz de ejecutar ASP.NET 4.5. Para hacer uso del transporte más eficiente, es necesario tener al menos Windows Server 2012 o Windows 8.

El servidor web ideal para ejecutar aplicaciones SignalR es IIS 8 o superior, tanto en sus ediciones completas y Express, porque esta es la primera versión del servicio que puede aceptar conexiones vía WebSockets. SignalR también funcionará en las versiones anteriores, 7 y 7.5, pero no tiene la capacidad de utilizar WebSockets.

En el desarrollo, se recomienda utilizar Visual Studio 2012 en adelante, y usar el IIS Express para pruebas.

2. HUBS

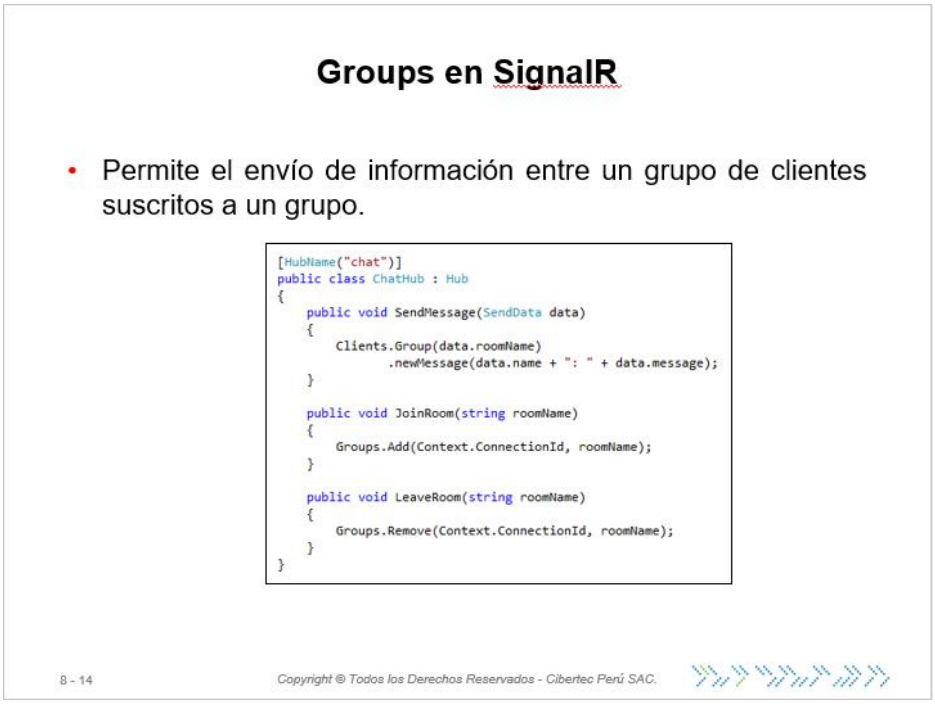


Desde el cliente, se invoca directamente a los métodos disponibles en el servidor, y viceversa.

Para que sea posible, en el lado del cliente, SignalR crea automáticamente objetos proxy con las interfaces de las clases Hub del servidor, y en sus métodos se insertan llamadas remotas a sus métodos reales.

A la inversa, cuando el servidor invoca a un método del cliente, se resuelve usando tipos dinámicos y un protocolo que empaqueta estas llamadas al servidor y las envía usando "push" mediante el transporte elegido, para que a continuación lleguen al cliente, donde se interpretan y ejecutan.

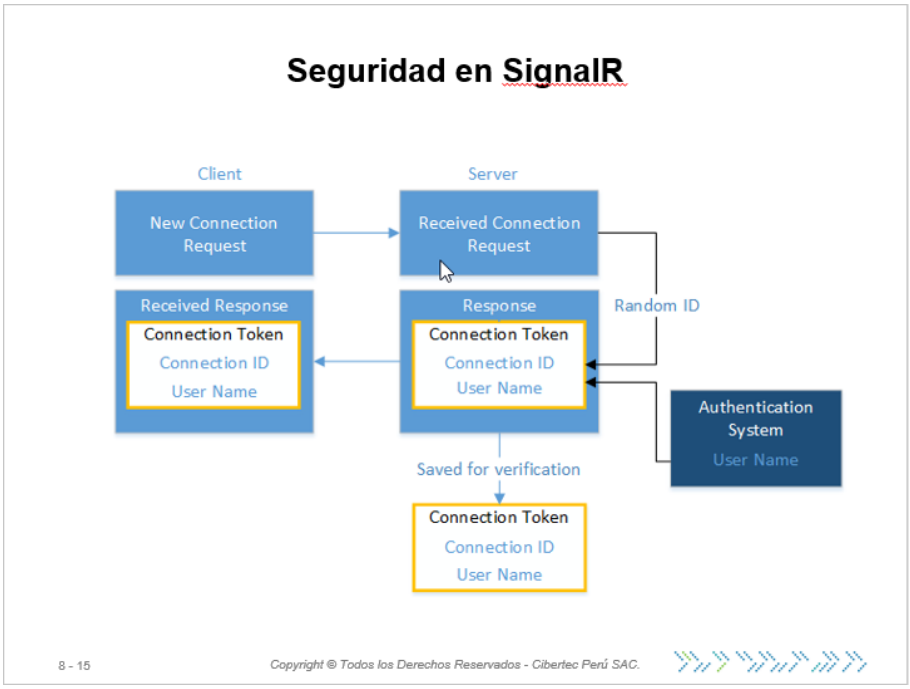
3. Groups en SignalR



Cada cliente que se conecta a un hub pasa un identificador de conexión único. Se puede recuperar este valor con la propiedad Context.ConnectionId del contexto del hub. Si la aplicación necesita asignar un usuario al ID de la conexión y persistir este mapeo, se puede utilizar uno de los siguientes enfoques:

- User ID Provider
- In-memory storage
- SignalR group for each user
- Permanent, external storage (database)

4. Seguridad en Signal-R



SignalR no proporciona funciones para la autenticación de usuarios. En su lugar, las características de SignalR se integran a la estructura de autenticación existente para una aplicación. La autenticación de usuarios se realiza como normalmente se haría en una aplicación, y luego se trabaja con los resultados de la autenticación en el código SignalR. Por ejemplo, es posible autenticar a los usuarios con ASP.NET forms authentication, y luego en el hub, asegurarse que los usuarios o los roles están autorizados a llamar a un método. En el hub, también se puede pasar información de autenticación al cliente, tal como el nombre de usuario o si un usuario pertenece a un rol.

SignalR proporciona el atributo Autorize para especificar qué usuarios tienen acceso a un hub o método.

Se puede aplicar el atributo Autorize solo a hubs, no a conexiones persistentes.