Server PUSH Notifications

Nemanja Kojic, MScEE
nemanja.kojic@etf.rs

What is Server PUSH?

- Internet-based communication
- Initiated by a cental server (pusher)
  - Contrasted with pull/get – initiated by a client
- Publish/Subscribe interaction model
  - A client subscribes to channels provided by the server
- Examples: instant messaging, notifications, SMTP, auctions, sport results...
- Can be emulated by the periodic polling.

Server PUSH Examples

- HTTP Server Push
  - WebSocket (HTML 5 API)
- Pushlet
  - server never closes the response - HTTP Streaming
  - server sends periodically snippets of JS to a client throught the live channel.
- Long polling
  - When a real push is not feasible due to certain security constraints.

SignalR

- It is a library for ASP.NET
  - for adding real-time functionality to web apps
- Server side code pushes content to connected clients in real-time
- Free software
- Distributed under the Apache License

SignalR – protocols

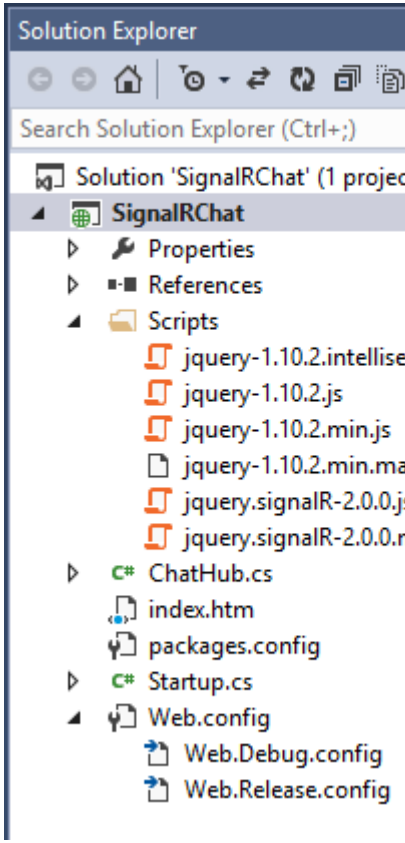
- Relies on several transport protocols
- Auto selection of the best protocol between a client and server.
- Uses WebSocket
  - Bidirectional client-server communication
- Gracefully falls back to other protocols if WebSocket is not available
  - App code remains unchanged!
- API for Server-to-Client RPC
  - Call a JS function browser from server-side .NET code!
  - Connection management hooks.

SignalR Dev Tasks

- Adding the SignalR library to an MVC5 app
  - Install-package Microsoft.AspNet.SignalR
- Creating hub and OWIN startup classes
  - to push content to clients
- Using the SignalR jQuery library in a web page
  - send messages and display updates from the hub

Install SignalR Library

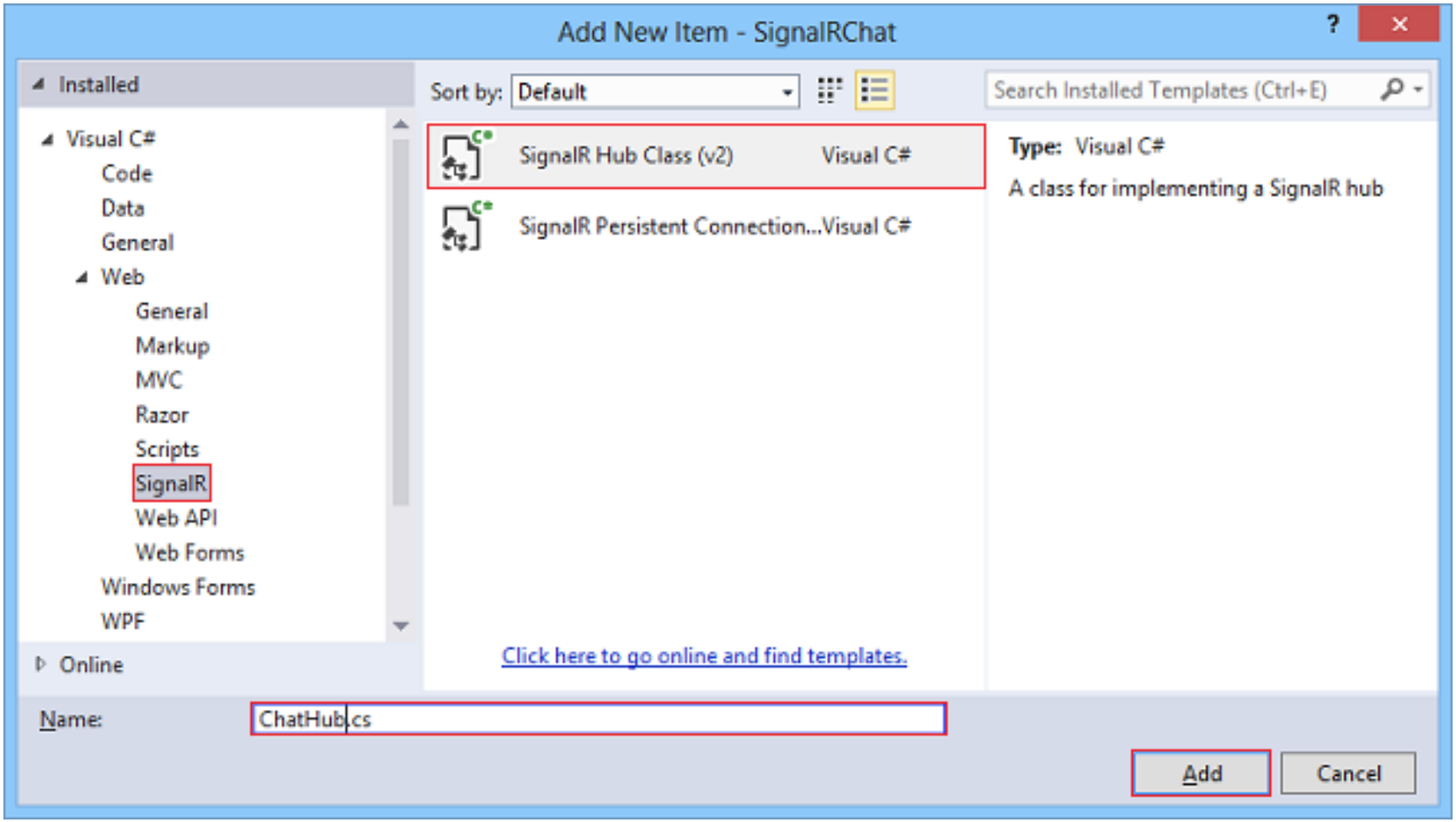
- First create an MVC5 Web app in Visual Studio
- **Tools > Library Package Manager > Package Manager Console**
- Type:
  - install-package Microsoft.AspNet.SignalR
- In **Solution Explorer**,
  expand the Scripts folder.
  Note that script libraries for SignalR have been added to the project.



Add a SignalR Hub Class

- (1) In **Solution Explorer**, right-click the project, select **Add | New Folder**, and add a new folder named **Hubs**.
- (2) Right-click the **Hubs** folder, click **Add | New Item**,
- (3) Select the **Visual C# | Web | SignalR** node in the **Installed** pane,
- (4) Select **SignalR Hub Class (v2)** from the center pane,
- (5) Create a new hub named **ChatHub.cs**.
- This class will be used as a SignalR server hub that sends messages to all clients.

Add a SignalR Hub Class (cont.)



Customize ChatHub Class

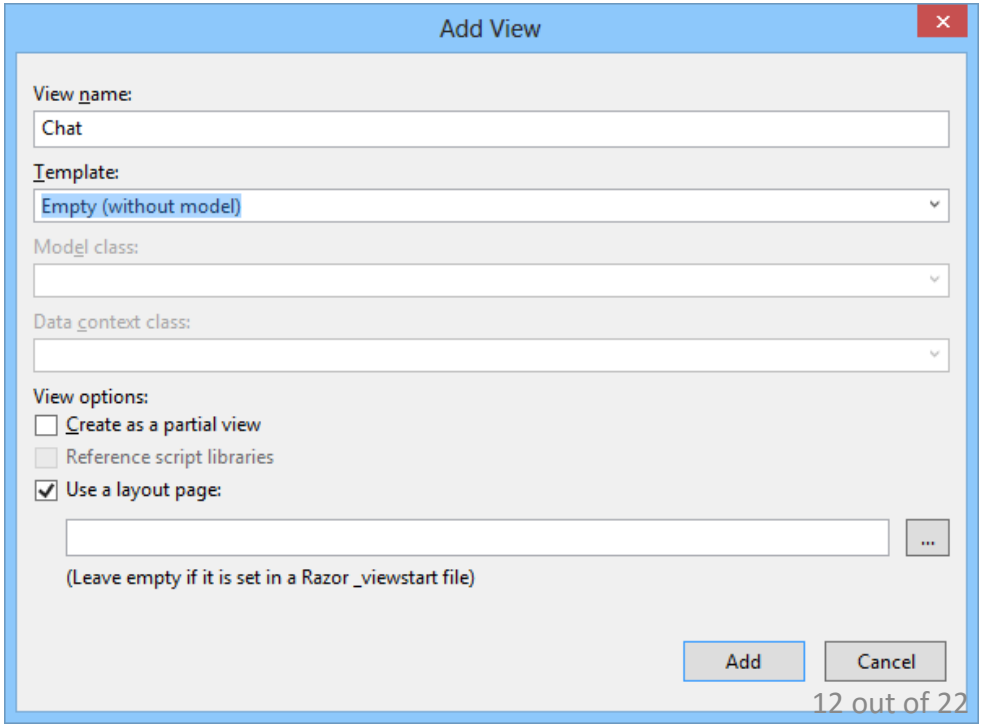
```
using System;
using System.Web;
using Microsoft.AspNet.SignalR;
namespace SignalRChat
{
    public class ChatHub : Hub
    {
        public void Send(string name, string message)
        {
            // Call the addNewMessageToPage method to update clients.
            Clients.All.addNewMessageToPage(name, message);
        }
    }
}
```

Add/Change Startup.cs class

```
using Owin;
using Microsoft.Owin;
[assembly: OwinStartup(typeof(SignalRChat.Startup))]
namespace SignalRChat
{
    public class Startup
    {
        public void Configuration(IAppBuilder app)
        {
            // Any connection or hub wire up and configuration should go here
            app.MapSignalR();
        }
    }
}
```

Add a View

- Right-click the **Views/Home** folder, and select **Add... | View**.
- In the **Add View** dialog, name the new view **Chat**.



View page (Chat.cshtml)

```
<h2>Chat</h2>
<div class="container">
    <input type="text" id="message" />
    <input type="button" id="sendmessage" value="Send" />
    <input type="hidden" id="displayname" />
    <ul id="discussion">
    </ul>
</div>
```

A simple chat form.

```
@section scripts {
    <!--Script references. -->
    <!--The jQuery library is required and is referenced by default in _Layout.cshtml. -->
    <!--Reference the SignalR library. -->
    <script src="~/Scripts/jquery.signalR-2.1.0.min.js"></script>
    <!--Reference the autogenerated SignalR hub script. -->
    <script src="~/signalr/hubs"></script>
    <!--SignalR script to update the chat page and send mes
    <script>
        $(function () {
            // Reference the auto-generated proxy
            var chat = $.connection.chatHub;
            // Create a function that the hub can call back to display messages.
            chat.client.addNewMessageToPage = function (name, message) {
                // Add the message to the page.
                $('#discussion').append('<li><strong>' + htmlEncode(name)
                    + '</strong>: ' + htmlEncode(message) + '</li>');
            };
            // Get the user name and store it to prepend to messages.
            $('#displayname').val(prompt('Enter your name:', ''));
            // Set initial focus to message input box.
            $('#message').focus();
            // Start the connection.
            $.connection.hub.start().done(function () {
                $('#sendmessage').click(function () {
                    // Call the Send method on the hub.
                    chat.server.send($('#displayname').val(), $('#message').val());
                    // Clear text box and reset focus for next comment.
                    $('#message').val('').focus();
                });
            });
        });
        // This optional function html-encodes messages for display in the page.
        function htmlEncode(value) {
            var encodedValue = $('<div />').text(value).html();
            return encodedValue;
        }
    </script>
}
```

Referenced scripts

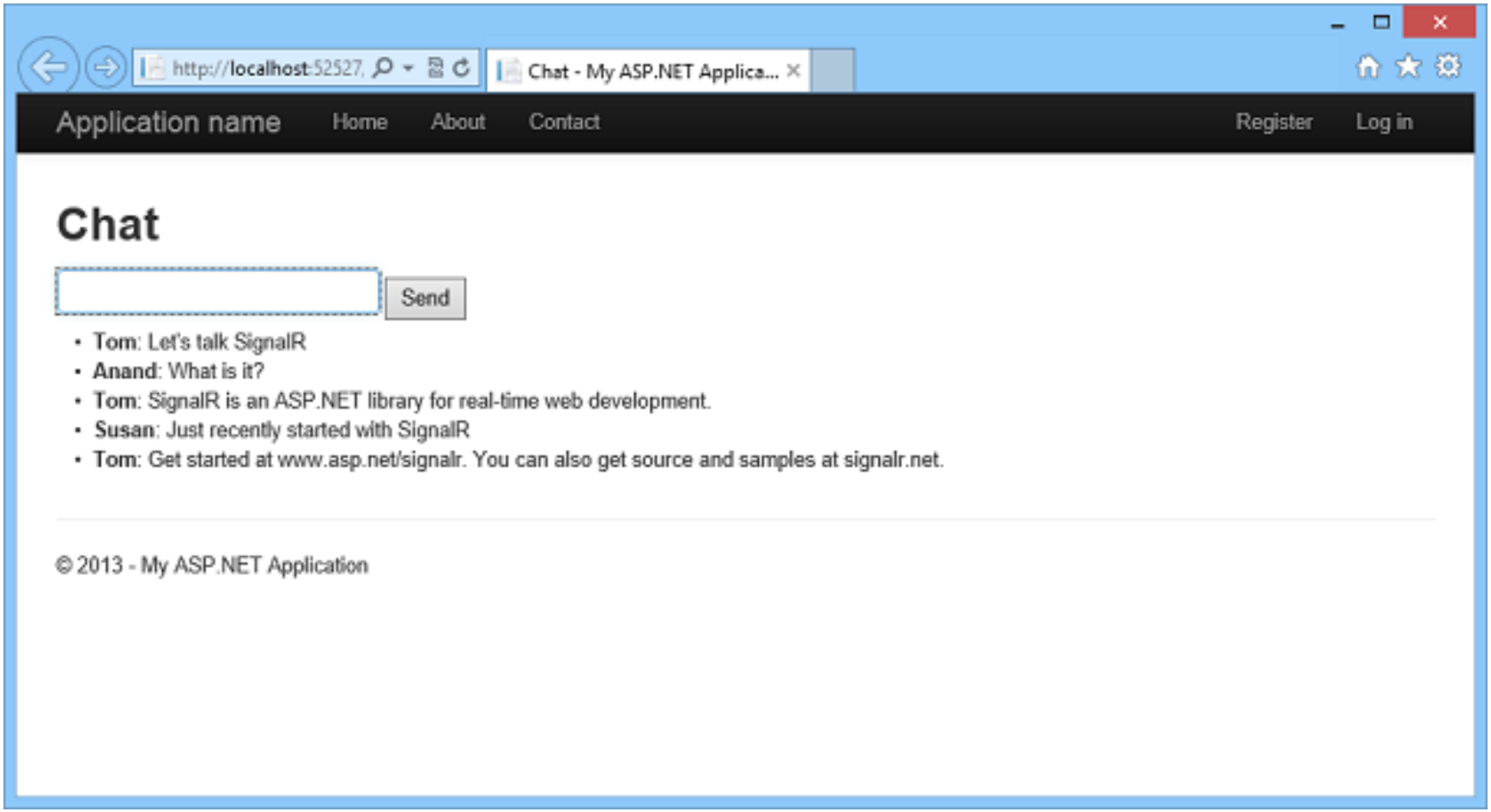
Declares proxy ref.

Defines a callback function

Starts connection with the hub

Invokes the hub's server-side method

Run…


http://localhost:52527/
Chat - My ASP.NET Applica...
Application name
Home
About
Contact
Register
Log in
Chat
Send
Tom: Let's talk SignalR
Anand: What is it?
Tom: SignalR is an ASP.NET library for real-time web development.
Susan: Just recently started with SignalR
Tom: Get started at www.asp.net/signalr. You can also get source and samples at signalr.net.
© 2013 - My ASP.NET Application

SignalR Code Examination

- Two basic SignalR development tasks:
  - creating a hub as the main coordination object on the server
  - using the SignalR jQuery library to send and receive messages

SignalR Hub Class (e.g. ChatHub)

- Derives from Microsoft.AspNet.SignalR.Hub
- Define your own public methods (e.g. Send)
- The methods are called from JQuery scripts in a web page
- Clients call ***ChatHub.Send*** to send a message
- The hub sends the message to all subscribed clients by calling ***Clients.All.addNewMessageToPage***

ChatHub.Send

- Demonstrates several hub concepts:
  - Declare public methods on a hub so that clients can call them
  - Use **Microsoft.AspNet.SignalR.Hub.Clients** property to access all clients connected to the hub
  - Call a function (e.g. addNewMessageToPage)on the client to update clients
  - Observer DP

```
public class ChatHub : Hub
{
    public void Send(string name, string message)
    {
        Clients.All.addNewMessageToPage(name, message);
    }
}
```

SignalR and JQuery

- Essential tasks on the view page:
  - creating a reference to the auto-generated proxy for the hub,
  - declaring a function that the server can call to push content to clients, and
  - starting a connection to send messages to the hub.

Declare ref to the hub proxy

```
var chat = $.connection.chatHub;
```

Declares a ref to a hub proxy.

- In JavaScript the reference to the server class and its members is in camel case (ChatHub -> chatHub).
- Can be redefined by annotating the hub class with [HubName("ChatHub")].

Create a callback function (script)

- The hub class on the server calls this function to push content updates to each client.

```
chat.client.addNewMessageToPage = function (name, message) {
    // Add the message to the page.
    $('#discussion').append('<li><strong>' + htmlEncode(name)
        + '</strong>: ' + htmlEncode(message) + '</li>');
};
```

The htmlEncode function shows a way to HTML encode the message content before displaying it in the page, as a way to prevent script injection.

Open a connection with the hub

```
$.connection.hub.start().done(function () {
    $('#sendmessage').click(function () {
        // Call the Send method on the hub.
        chat.server.send($('#displayname').val(), $('#message').val());
        // Clear text box and reset focus for next comment.
        $('#message').val('').focus();
    });
});
```

- The code starts the connection and then passes it a function to handle the click event on the **Send** button in the Chat page.
- This approach ensures that the connection is established before the event handler executes.

References

- The official SignalR project page. http://signalr.net/

- A SignalR Tutorial http://www.asp.net/signalr/overview/getting-started/tutorial-getting-started-with-signalr-and-mvc