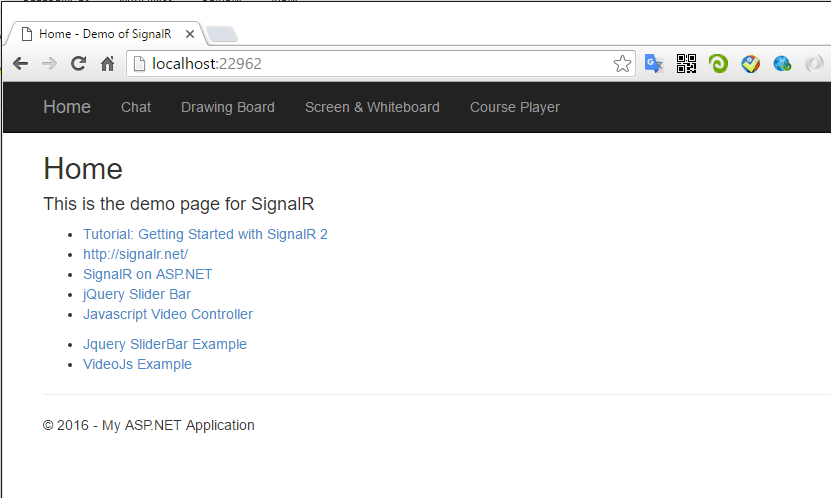
**Online Course Player with SignalR**

This web application is developed by SignalR based on ASP.NET. It contains four samples, named as follows:

* Chart
* Drawing Board
* Screen and Whiteboard
* Course Player

1. **Home Page**

Some resources which help to build this application.

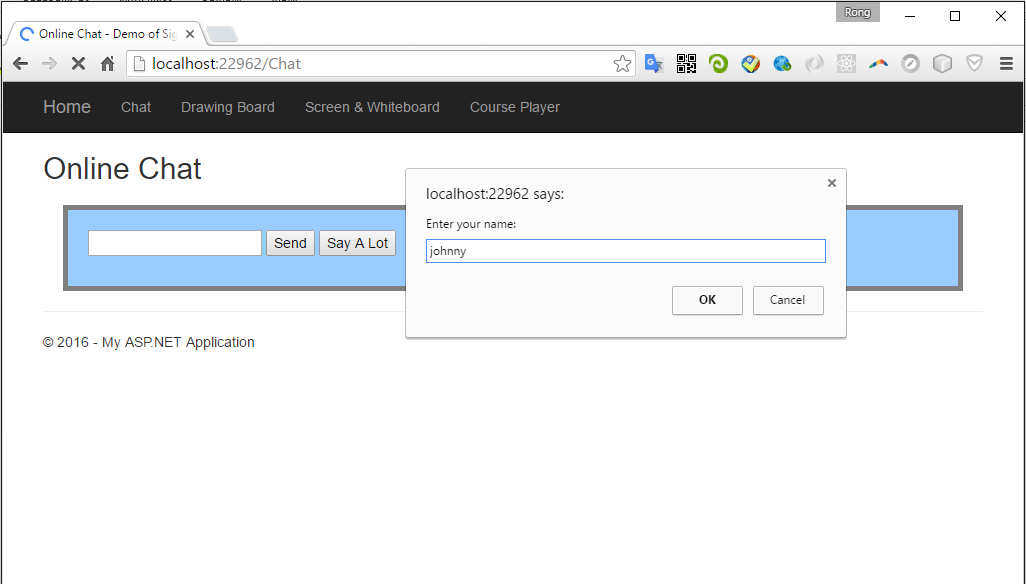


1. **Chat**
   1. Introduction

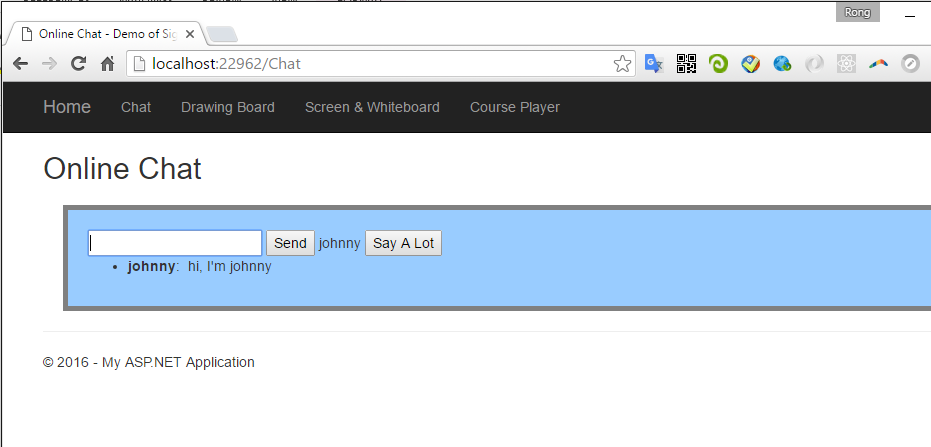
This is an online chatting room application. Any message submitted is broadcasted.

* 1. First User (chrome)

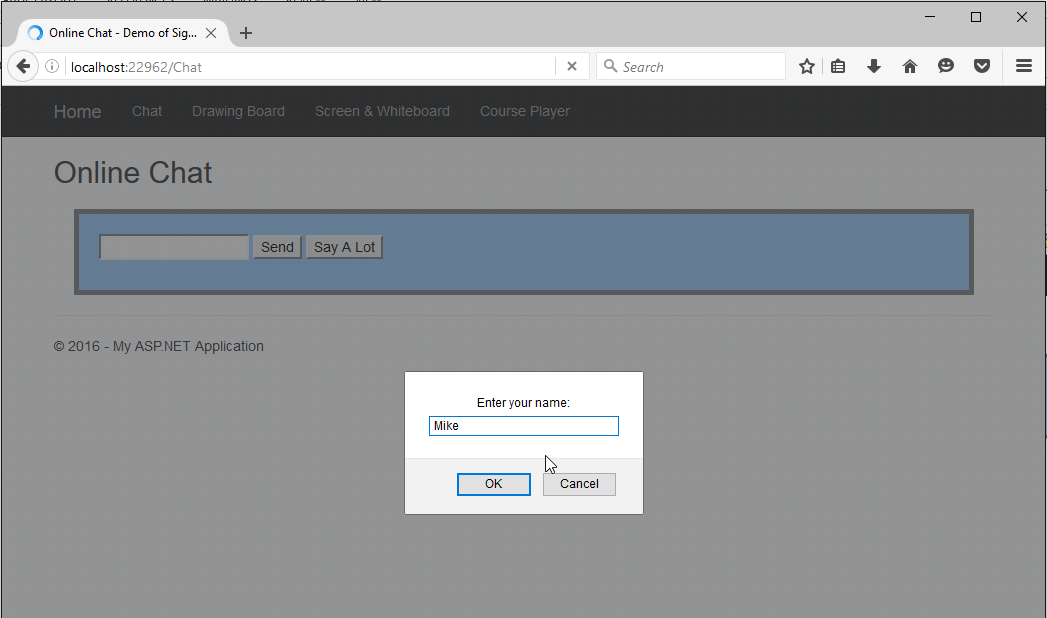
To join the chatting room, you must provide a user name first.



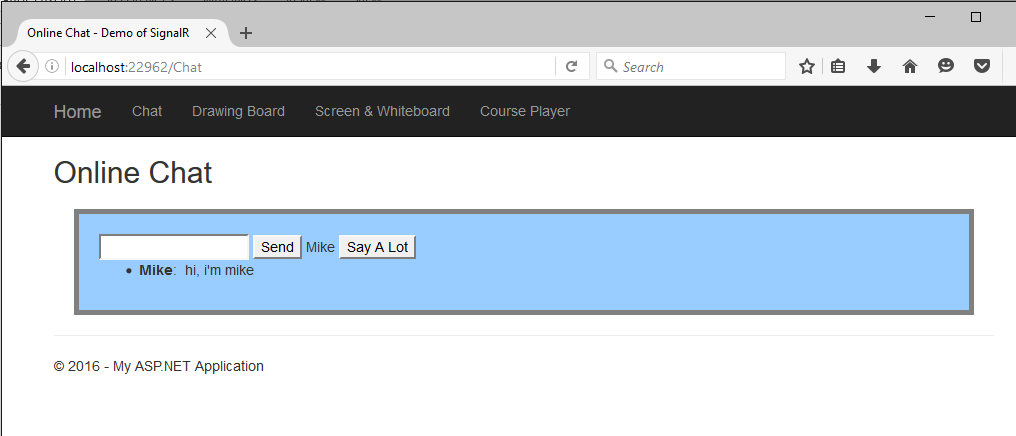
To send message, just input the text and click ‘Send’.



* 1. Second User(Firefox)

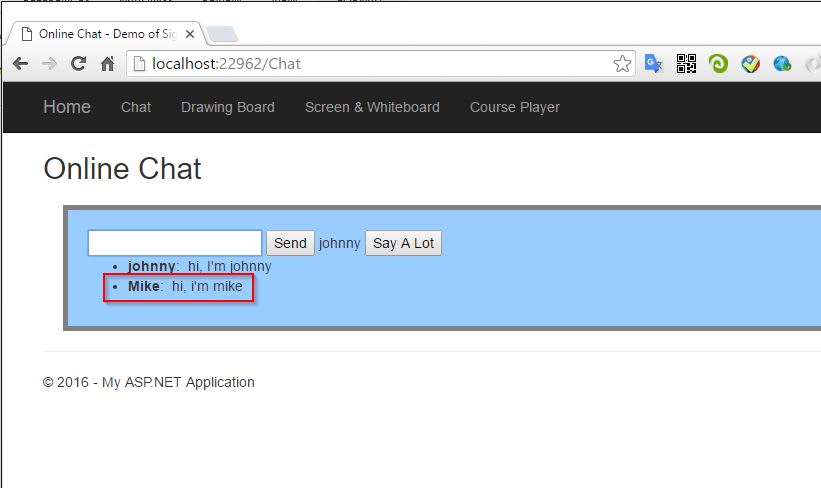


Send a message to others.



* 1. Receive Broadcasting Messages

Go back to the first user. The message from Mike is displayed automatically. Here, we see the real-time message without refreshing the current page.



* 1. Under the Hood

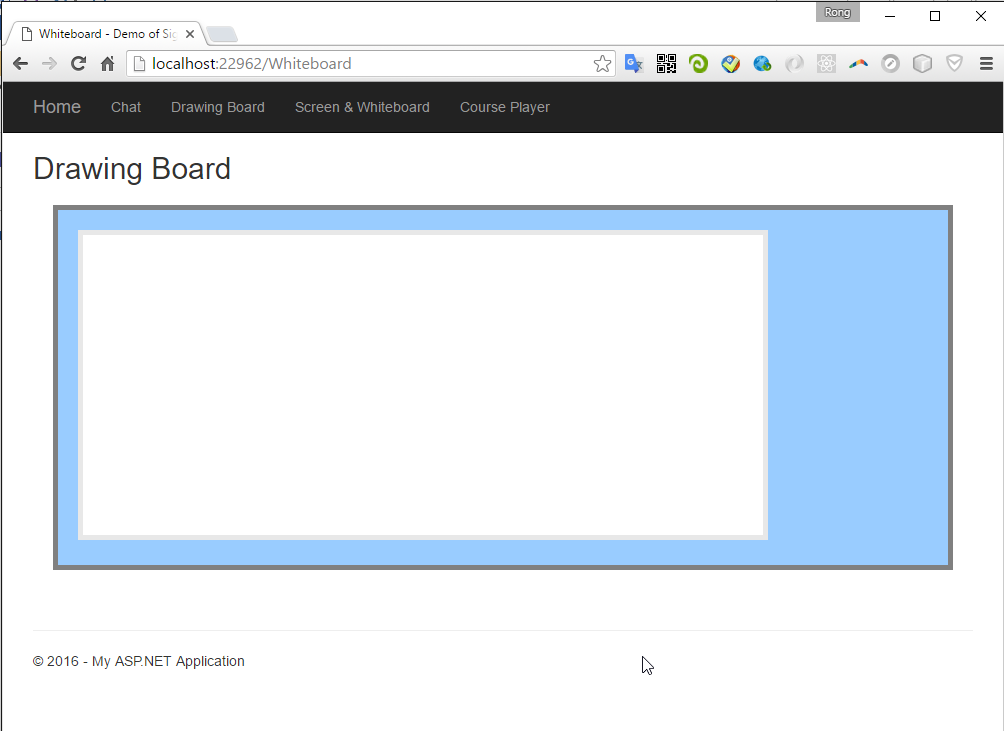
How does it work?

1. When accessing the chatting room, a new connection is setup between your web browser and the server.
2. Then, you provide a user name to identify yourself from others. This name is added to the chatting group.
3. Next, you type some texts and click the ‘Send’ button, the message is sent to the server through WebSocket protocol.
4. This server, actually is a general ASP.NET web application. The magic thing is, SignalR creates an HTTP Handler upon ASP.NET runtime to deal with the WebSocket messages. When it receives any message from client, it will notify other users within the same group. Technically, this process is called: Broadcasting.
5. Another user’s web browser receives your input from server, and display it to the page.
6. If no user input any text, no communication will happen.
7. If user close the web browser, it will be removed from the chatting group.
8. The group survives until no user in the group.
9. **Drawing Board**
   1. Introduction

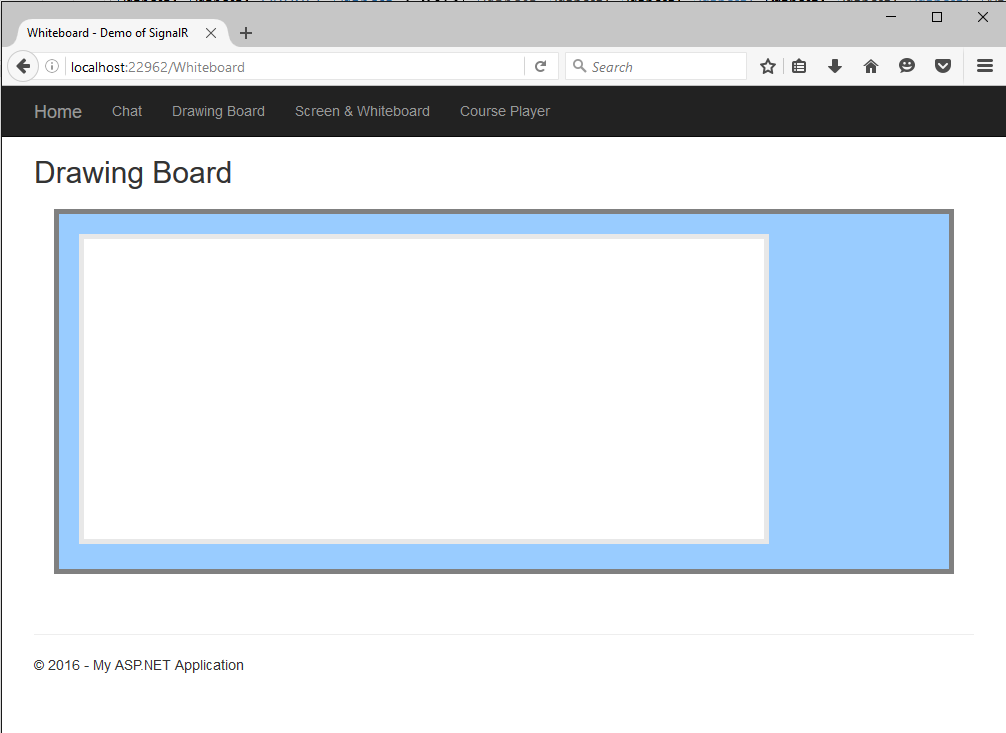
One user draws on the canvas and other users will see the drawing immediately in their own canvas board. This drawing board is implemented by HTML5 canvas and SignalR.

* 1. Open two drawing boards with different browsers.

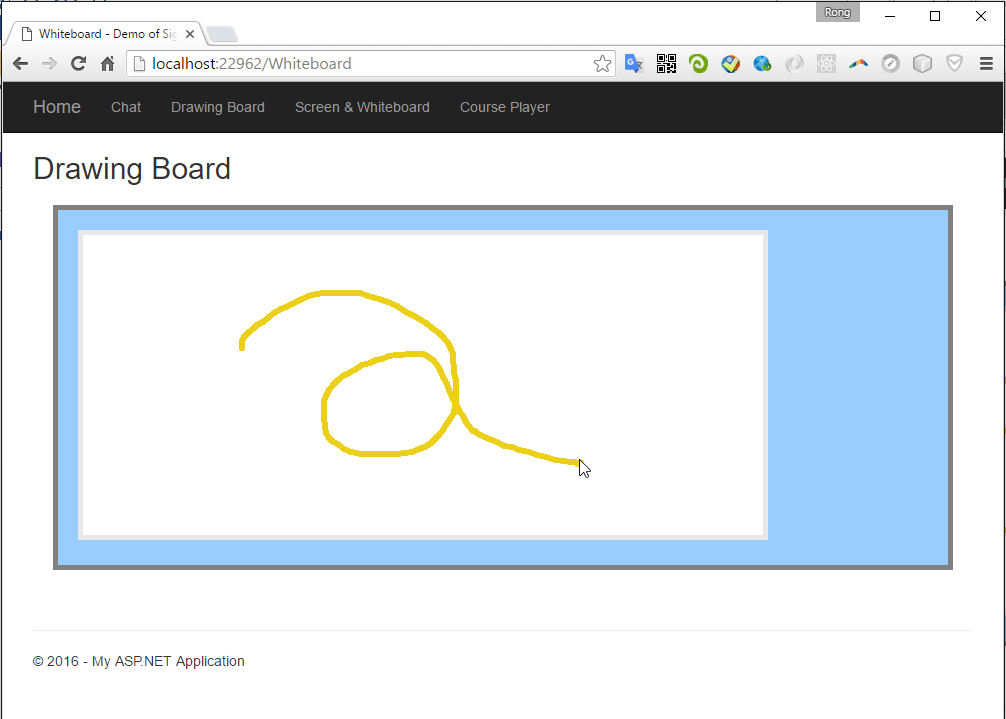
Open the first drawing board in chrome.



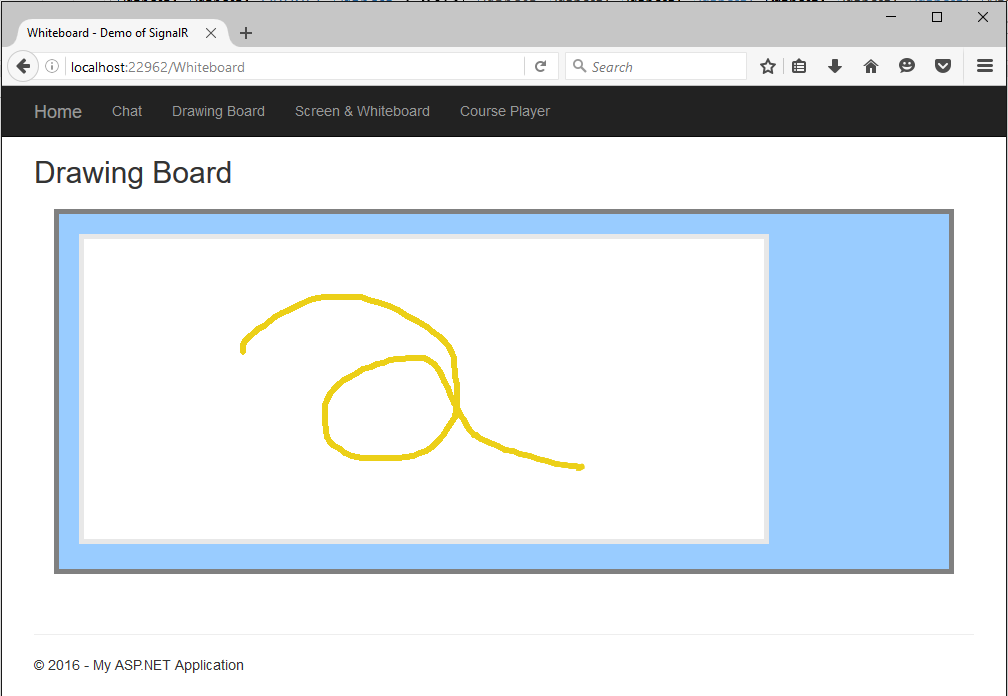
Open the second drawing board in firebox.



* 1. Begin drawing in chrome.



* 1. The drawing is synchronized to others(eg. here is Firefox) simultaneously.



* 1. Under the Hood

The process is same with the chatting room. The only difference is, the format of data used for the communication between server and client. For drawing, we send out coordinates of the points in the canvas instead of a single string message.

1. **Screen & Whiteboard**
   1. Introduction

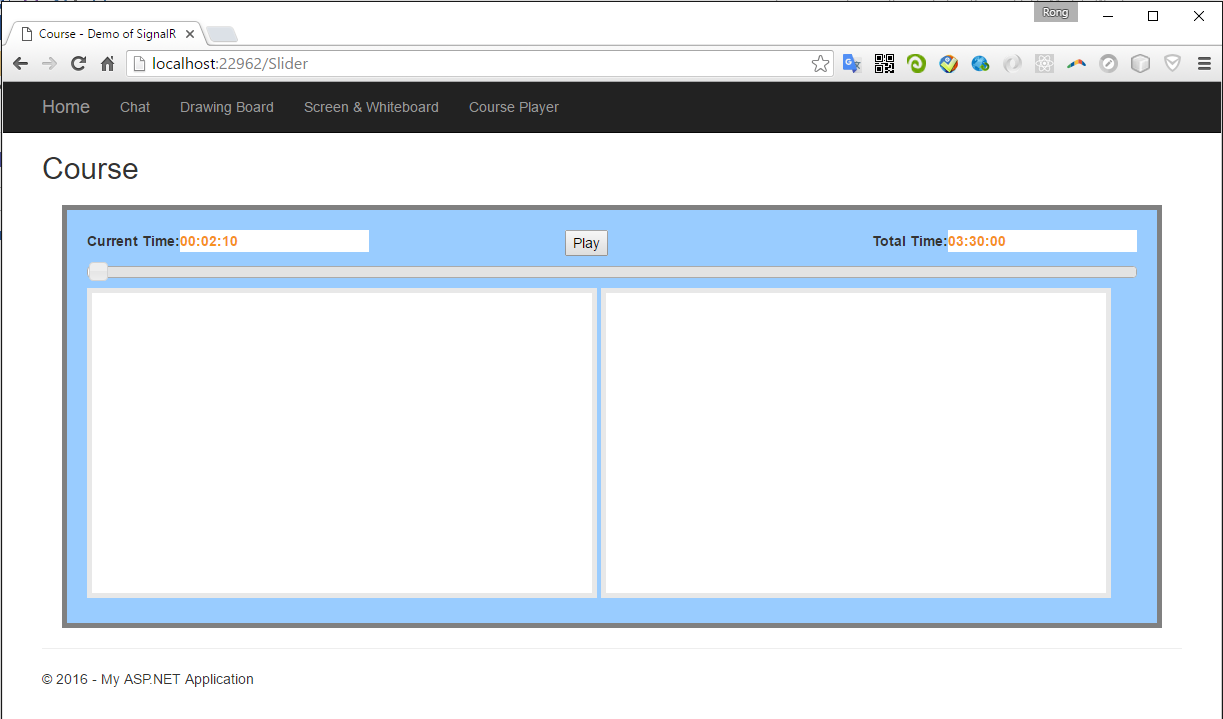
A course player consists of three components, video, screenshot and whiteboard.

* Video is captured by a camera during the lecturing time, and saved as mp4.
* Screenshot is captured from computer monitor through which teachers share their handouts/materials to the students. Screenshot are images which are saved in a single file.
* Whiteboard is captured from special pens and boards. Any operation on the board, such as writing, drawing or brushing is logged and stored to a single file.

For this course player, video is played independently. The content of the screen and whiteboard is synchronized with the playing process of the video. In this sample, I use a slider bar to simulate the video player.

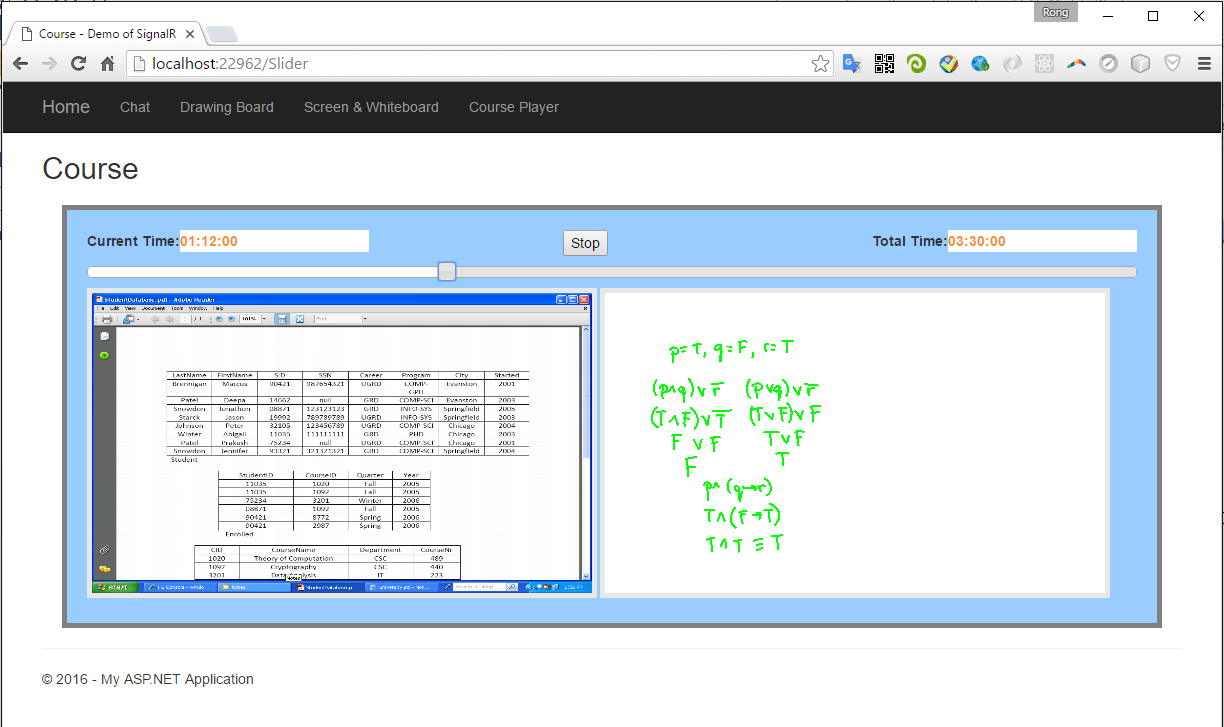
* 1. UI

On the top of the player, there is the process bar and a Play button. There are two canvases below the process bar. The left one is for screenshot and the right one is for whiteboard.



* 1. Play

When you click the play button, the process begins to move, the current time will be refreshed as well, one second for interval. The screenshot and whiteboard canvas show the content simultaneously. You can drag the process bar forward or afterward.

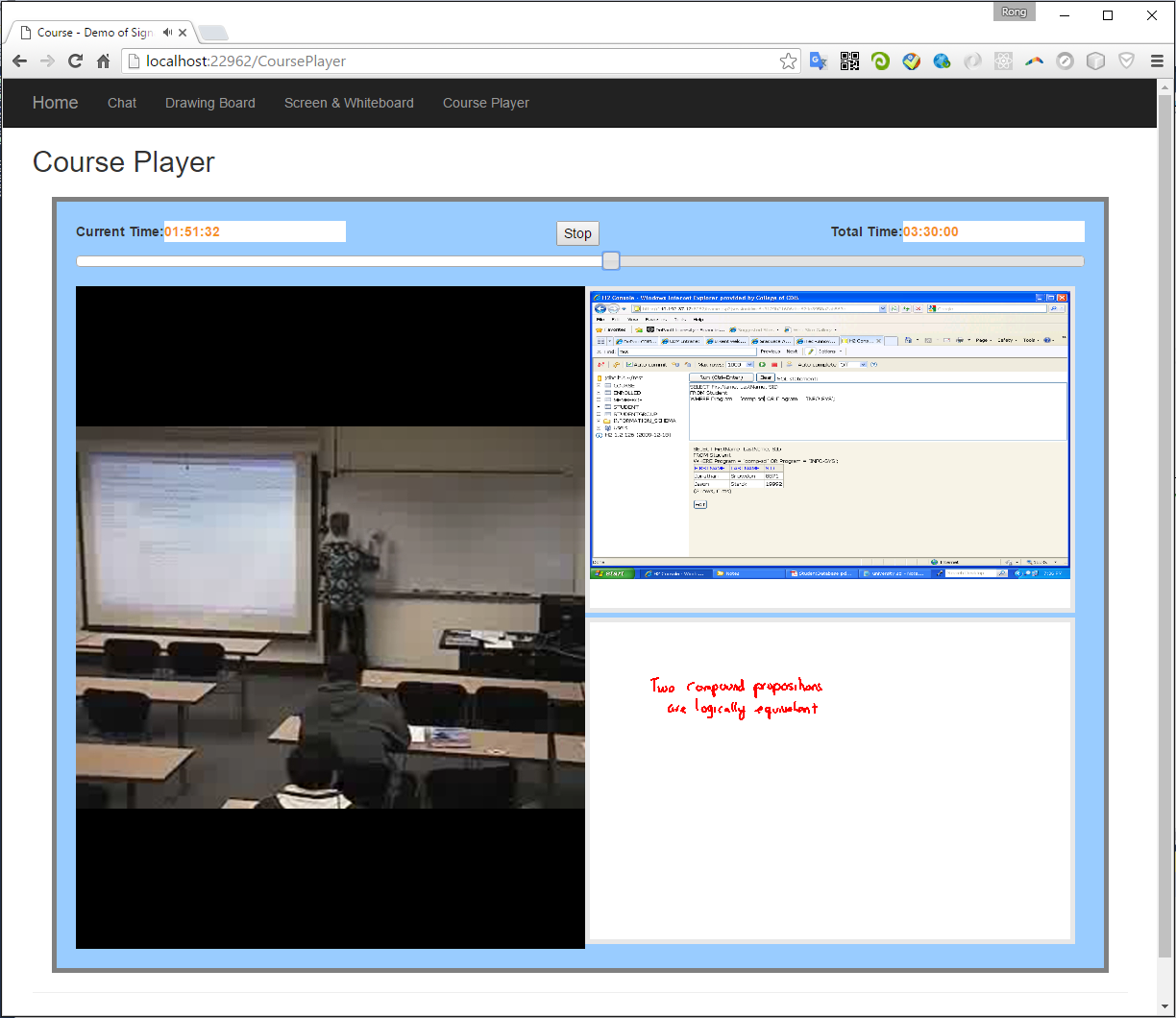


* 1. Under the Hood

How does this dummy player work?

1. When the page is opened, the connection is setup between server and client(web browser).
2. Next, the course starts to play after the Play button is clicked. At the same time, a notification is sent to the server, and the server knows that the player has begun to work.
3. Then, server starts a timer, run the task(step 4) for every second.
4. Server reads data for screenshot and whiteboard based on the current time.
5. If there is any update(new image or new drawing), it will send data(JSON format) to client. Otherwise, no communication occurs from server to client.
6. If web browser gets data, it will draw images or lines accordingly.
7. The communication from client to server occurs only when the play button is click or the process bar is dragged.
8. The communication from server to client occurs only when new data is found.
9. **Course Player**
   1. Introduction

Based on the previous sample, add a HTML5 Video control to make it a real course player. Here is the js video control for html5, http://videojs.com/.



1. **Conclusion**
   1. Easy to Implement

If you are familiar with C# and ASP.NET, it is really easy to develop such real time online application. Of course, you need write some javascript code to use SignalR at the client side.

* 1. Low Bandwidth Consumption

Communication occurs only when necessary. Unlike traditional web application, WebSocket makes the web application react at real time. This improve the user experience at client side and system performance at server side.

* 1. Cross-platform(For customers/students)

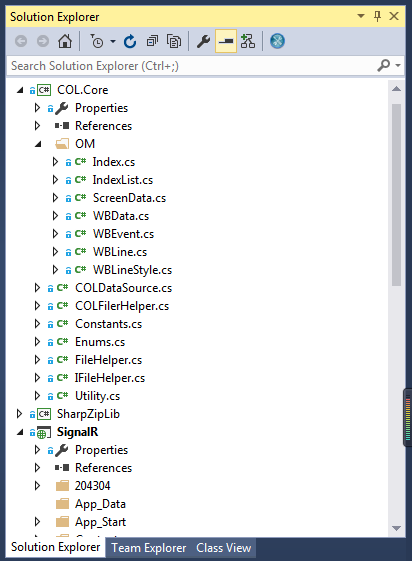
This player is web based, no installation on client’s machine is required. Besides, this course player is based on HTML5, so it can be accessed in different web browsers and on different platforms. No need to install extra plugin in web browser, such as flash player or Silverlight.

* 1. Cross-platform(For developer)

For developer, since this WebSocket based player is a cross-platform application, it is a better solution than other platform specific solutions. Compared with our existing Flash and Silverlight player, this course player is simple and easy to maintain, since there is only one copy of the code.

* 1. Reuse

The core module(COL.Core) of this course player is shared with Xamarin solution, which is for cross-platform mobile development.



This means, we have the cross-platform solution for developing applications with only using C#.

First, use Xamarin to develop mobile apps for iOS and Android Platform.

Second, use ASP.NET and SignalR to develop web application for different web browsers and platforms.

Technically, the core module can be shared and reused by mobile and web application, even, it can be shared with winform applications.

Two parts cannot be reused, one is the UI, web(html) and mobile(native UI) are obviously different. And another is file operation, reading/writing file on windows/ios/linux platform varies apparently.

However, the business logics are same, which can be reused.