MONASH University

FIT5032 - Internet Applications Development

Email, SignalR and etc

Prepared by - Jian Liew
Updated by ABM Russel

What is email automation?

- A clear understanding of email automation sets the stage for a successful strategy and provides a high-level view of the possibilities with email automation.
- Automated email, at its core, is any **marketing or transactional email** that is sent based on predefined rules or triggers the sender defines and that doesn't require anyone to actually hit a "send" button.
- Automated email is not necessarily an additional form of email; it's best viewed as a strategy that sits on top of both your marketing and transactional emails.
- The most common form of email automation would be when a user has successfully registered to your website, an email should be send informating the user regarding this matter.

Example Use Cases of Email Automation

- You can consider these to be **email marketing**.
- Automate hotel room upgrades. You book a hotel room somewhere, and then as you approach your sign-in date, you would get an email **asking if you'd like to upgrade the type of room** you've booked a week or so before you check-in.
- Let's say a customer books a trip to Tasmania, Australia. Based on the location of their booking, they'd then get **three follow up emails** about the location and some things to do their, or places to stay there.
- Consider sending special email with a discount or free gift that is triggered on the year after a user signs up. Just make sure that the discount or gift can be redeemed with a week or two of wiggle room in case the user is not checking his or her email all the time. (How do you think can this be accomplished?)

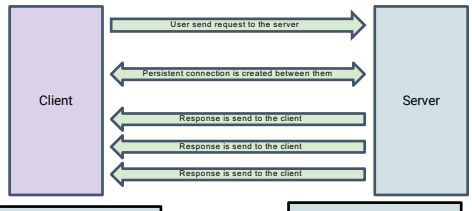
SendGrid

- SendGrid developed an industry-disrupting, cloud-based email service to solve the challenges of reliably delivering emails on behalf of growing companies.
- Today, SendGrid is responsible for sending billions of emails for some of the best and brightest companies in the world.

SignalR

- ASP.NET SignalR is a library for ASP.NET developers that simplifies the process of adding **real-time web functionality to applications**.
- Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.
- There are multiple use cases for SignalR. Examples include dashboards and monitoring applications, collaborative applications (such as simultaneous editing of documents), job progress updates, and real-time forms. **One of the more obvious use case is the ability to create a "chat" room.**
- Simplifies the process of building real-time applications. It includes an **ASP.NET server library and a JavaScript client library** to make it easier to manage client-server connections and push content updates to clients.

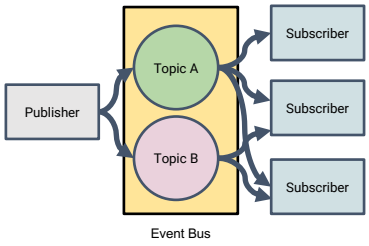
Real Time Web Functionality



This happens **without the need to refresh** the browser. (hence real time)

The client receives update as soon as there is an update on the server.

Publish & Subscribe



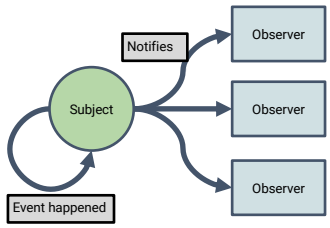
In the Publish and Subscribe architecture, each subscriber would be subscribed to a topic.

Whenever the publisher pushes a entry into the topic, these topic will update each subscriber.

Each subscriber and subscribe to different topics and as many topics as they want.

SignalR uses a Pub/Sub architecture.

Observer



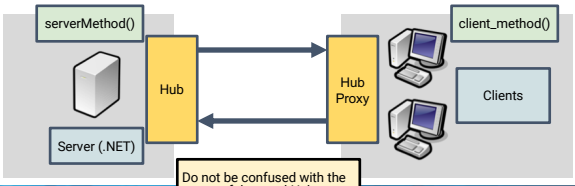
In the observer architecture, when an event happens in the subject, it will notify the observers.

In comparison to the Pub/Sub architecture, the observer does not go through an **event channel**. **So, there is a difference between the Observer and the Pub/Sub architecture.**

The "subject" itself will notify the observer of a change in event.

SignalR continued...

SignalR provides a simple API for creating server-to-client **remote procedure calls (RPC)** that call JavaScript functions in client browsers (and other client platforms) from server-side .NET code. **In a way, the server would expose methods to the clients so it can be used.**



Do not be confused with the usage of the word Hub. (From networking subjects)

SignalR

- SignalR handles connection **management automatically**, and lets you broadcast messages to all connected clients simultaneously, like a chat room.
- You can also send messages to specific clients.
- The connection between the client and server is persistent, unlike a classic HTTP connection, which is re-established for each communication.
- SignalR supports "server push" functionality, in which server code can call out to client code in the browser using Remote Procedure Calls (RPC), rather than the request-response model common on the web today.

SignalR and WebSocket

- SignalR uses the new WebSocket transport where available, and falls back to older transports where necessary.
- While you could certainly write your application using WebSocket directly, using SignalR means that a lot of the extra functionality you would need to implement will already have been done for you.
- Most importantly, this means that you can code your application to take advantage of WebSocket without having to worry about creating a separate code path for older clients.
- SignalR also shields you from having to worry about updates to WebSocket, since SignalR will continue to be updated to support changes in the underlying transport, providing your application a consistent interface across versions of WebSocket.
- **Do not be concerned about WebSocket as SignalR has simplified this process for us.**

Connections and Hubs

- The SignalR API contains two models for communicating between clients and servers: Persistent Connections and Hubs.
- A Connection represents a simple endpoint for sending single-recipient, grouped, or broadcast messages. The Persistent Connection API (represented in .NET code by the PersistentConnection class) gives the developer direct access to the low-level communication protocol that SignalR exposes.
- A Hub is a more high-level pipeline built upon the Connection API that allows your client and server to call methods on each other directly. SignalR handles the dispatching across machine boundaries as if by magic, allowing clients to call methods on the server as easily as local methods, and vice versa.