Introducing ASP.NET SignalR - Push Services with Hubs

Web-based Real-time communication & SignalR

Christian Weyer

christian.weyer@thinktecture.com

http://www.thinktecture.com

@christianweyer

pluralsight
hardcore developer training

Outline

- Problem space
- Push Services pattern
- HTTP & technical approaches for pushing
- ASP.NET SignalR as one solution

Real time: Problem space

- **It is all about the users**
- **Users want data**
  - Now & instant
  - Up-to-date
  - Delivered to any device, over any connection

- **Increasing number of web sites & web applications offer ‚real time' data**
  - Live searches/updates
  - Stock streamers, auctions
  - Live scores, betting, interactive games
  - Collaborative apps
- **In general: Real-time feedback, real-time notifications**

DEMO

Edit data in browser(s)

Real time: Developers' world

- **Developers look for ways to provide real time data**
  - But not only for web applications
  - What about mobile devices & apps?
  - What about traditional desktop applications?
  - What about server-to-server?
- **Web-based push communication beyond the web is a need**

- **We got accustomed to a service-oriented design**
  - Think in service facades
  - Facades provide entry points into our logic & data access

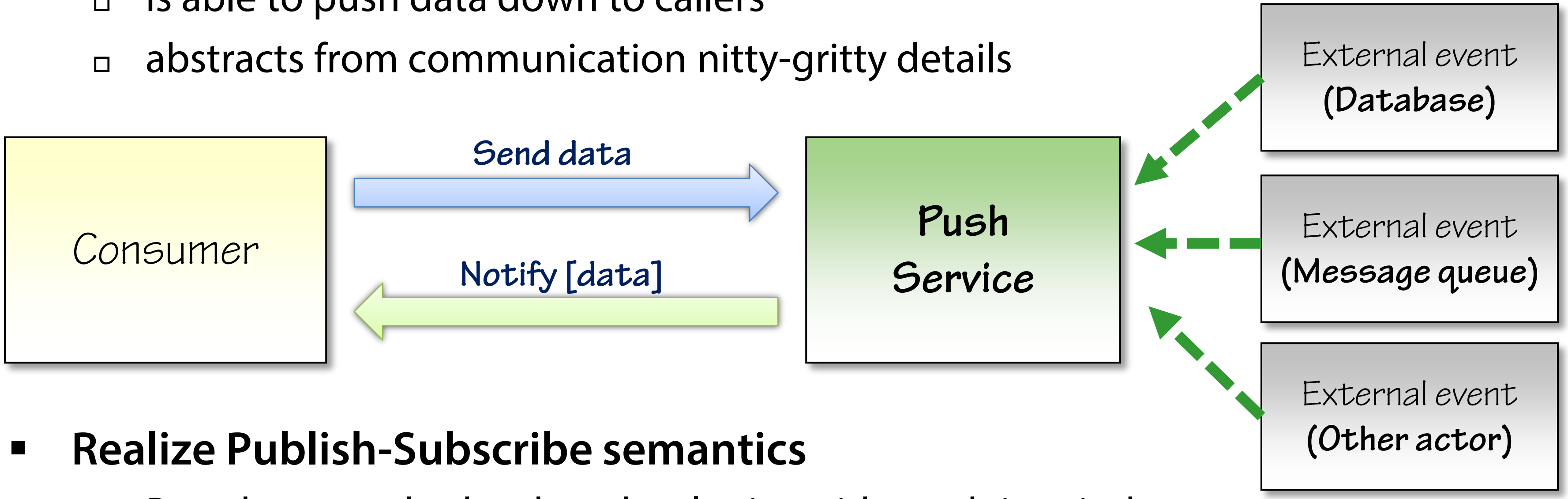
- **Think, design & implement Push Services**

DEMO

Edit data in database & see changes in browser

Push Services pattern

- **Push Services are not an official pattern [1]**
- **Model a service that**
  - accepts incoming connections from callers
  - is able to push data down to callers
  - abstracts from communication nitty-gritty details


Consumer
Send data
Notify [data]
Push Service
External event (Database)
External event (Message queue)
External event (Other actor)

- **Realize Publish-Subscribe semantics**
  - Based on standard web technologies with reach in mind
  - With maximum reach into any device, platform

DEMO

Cross-platform chat

HTTP is *the* protocol

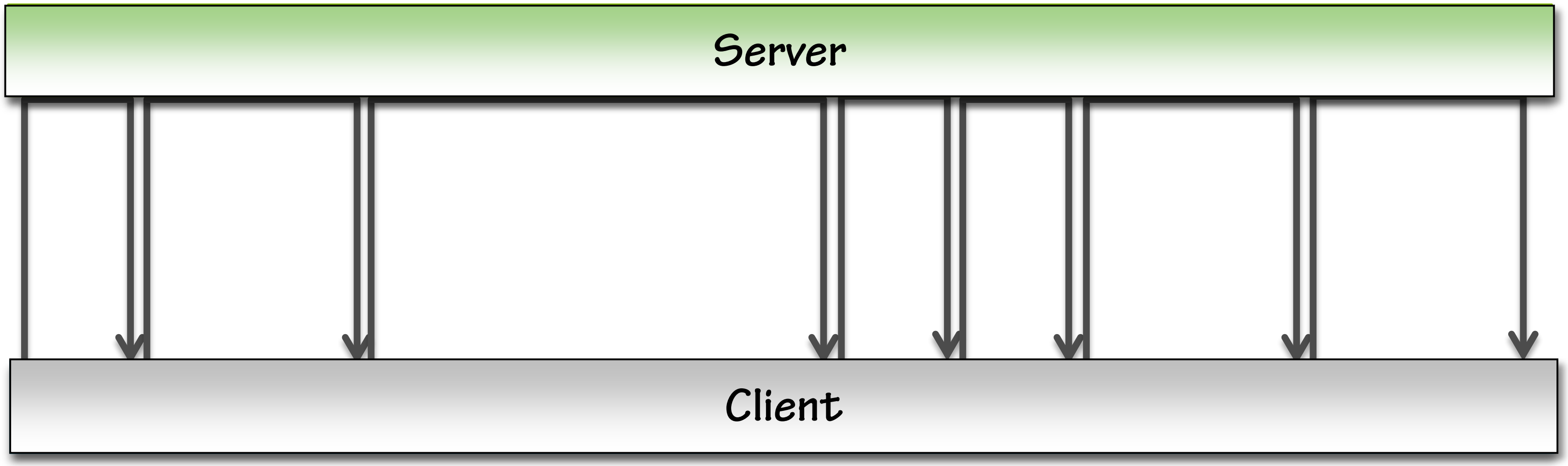
- **When talking about web communication technologies we talk about HTTP**
  - HTTP is warrantor for ubiquity & reach

- **HTTP is inherently request-response, n’est pas?**
- **Still we need to realize Push Services with what HTTP gives us**

Technical approaches for push

- **Periodic Polling**
- **Long Polling**
  - HTTP Streaming / Comet

Long polling



Poll but don't respond untill there's data

- Server holds on to the HTTP request until there is data to return
- (Re-)Poll after data received or after the connection times out
- Consumes server threads & connection resources

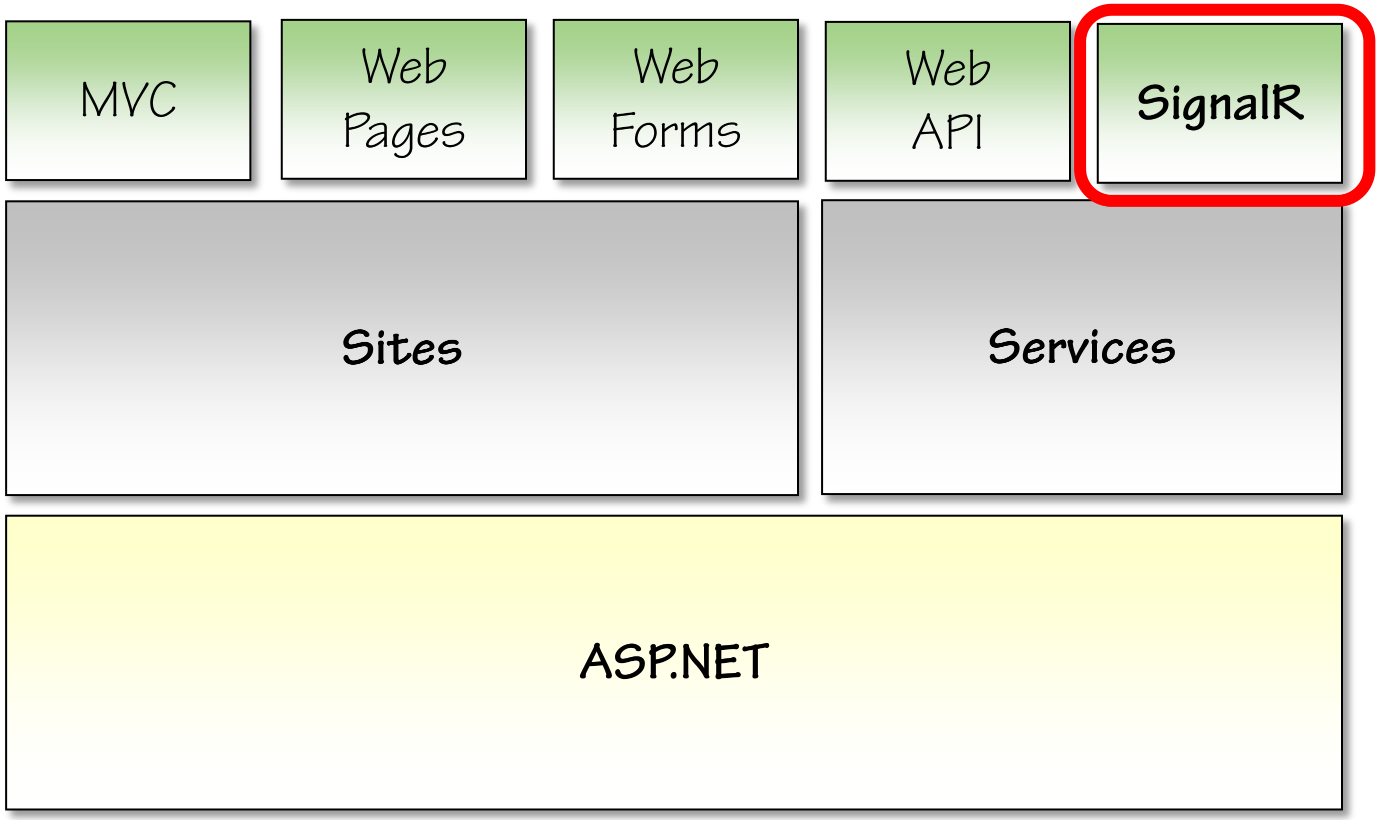
Technical approaches for push

- **Periodic Polling**
- **Long Polling**
  - HTTP Streaming / Comet
- **Forever Frame**
- **Server-Sent Events (SSE)**
- **Web Sockets**

- **Easy: Web Sockets are *the* way to go!**
  - Only with Windows 8/Server 2012
  - Network considerations
  - Maybe some time, but not today

- **Alright, let’s just write code for any technique!**
  - Erm… really? For server and client?

ASP.NET platform



ASP.NET SignalR as a solution

- **SignalR is**
  - a server-side framework to write push services
  - a set of client libraries to make push service communication easy to use on any platform
  - optimized for asynchronous processing

- **Abstracts from the different techniques to implement pushing data**
  - Mental model is a persistent connection
  - Volatile, no-durable
- **‘Signal’, anyone?**
  - Sending data to a signal. E.g. represented by a connection ID

- **Part of the ASP.NET brand, but not tied into ASP.NET runtime and APIs**

ASP.NET SignalR development

- **Extensible framework & pipeline**
  - Based on interfaces & DI
- **Two programming models**
  - Persistent connections
  - Hubs
- **Hubs offer a pre-defined application-level protocol in an RPC-ish style**
  - Easy-to-get-going means for 80/20 situations

ASP.NET SignalR project

- **SignalR is completely open source**
  - Public GitHub repository

- **SignalR packages available via NuGet**
  - Microsoft.AspNet.SignalR: package that brings in everything you need to run it on IIS and ASP.NET
  - Microsoft.AspNet.SignalR.Core: server side components needed to build SignalR endpoints
  - Microsoft.AspNet.SignalR.SystemWeb: pulls in the required packages to host SignalR in ASP.NET (via OWIN ASP.NET host)
  - Microsoft.AspNet.SignalR.Owin: OWIN host for SignalR
  - Microsoft.AspNet.SignalR.Js: jQuery client for SignalR
  - Microsoft.AspNet.SignalR.Client: .NET client for SignalR (includes WinRT, Windows Phone 8 and Silverlight5 clients)
  - Microsoft.AspNet.SignalR.Utils: command line utilities including performance counter installation and Hub JavaScript proxy generation

DEMO

Quick SignalR Hubs demo

Summary

- **Increasing need for near-real-time data**
  - Based on web technologies, like HTTP
  - Beyond pure web & browser scenarios
- **Think, design & implement Push Services**

- **ASP.NET SignalR offers hubs to easily realize push**
  - Server-side framework for ASP.NET or any other .NET host
  - Client-side frameworks for various platforms & devices

References

- **Wikipedia Push Technology [1]**
  - http://en.wikipedia.org/wiki/Push_technology
- **What came before WebSockets?**
  - http://blog.pusher.com/what-came-before-websockets/

- **Server-Sent Events**
  - http://www.whatwg.org/specs/web-apps/current-work/multipage/comms.html
- **Web Sockets**
  - http://www.whatwg.org/specs/web-socket-protocol/
  - http://dev.w3.org/html5/websockets/

- **Real time, Asynchronous Web Pages using jTable, SignalR and ASP.NET MVC**
  - http://www.codeproject.com/Articles/315938/Real-time-Asynchronous-Web-Pages-using-jTable-Sign