عنوان: **SignalR**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۳/۲۸ ۲۲:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, SignalR

چند وقتی هست که در کنار بدنه اصلی دات‌نت فریم‌ورک چندین کتابخونه به صورت متن‌باز در حال توسعه هستند. این مورد در ASP.NET بیشتر فعاله و مثلا دو کتابخونه **SignalR** و **WebApi** توسط خود مایکروسافت توسعه داده میشه.
SignalR همونطور که در سایت بسیار خلاصه و مفید یک صفحه‌ای! خودش توضیح داده شده (^) یک کتابخونه برای توسعه برنامه‌های وب «زمان واقعی»! (**real-time web**) است:

```
Async library for .NET to help build real-time, multi-user interactive web applications.
```

برنامه‌های زمان واقعی به صورت خلاصه و ساده به‌صورت زیر تعریف میشن (^):

```
The real-time web is a set of technologies and practices that enable users to receive information as soon as it is published by its authors, rather than requiring that they or their software check a source periodically for updates.
```

یعنی کاربر سیستم ما بدون نیاز به ارسال درخواستی صریح! برای دریافت آخرین اطلاعات به روز شده در سرور، در برنامه کلاینتش از این تغییرات آگاه بشه. مثلا برنامه‌هایی که برای نمایش نمودارهای آماری داده‌ها استفاده میشه (بورس، قیمت ارز و طلا و ...) و یا مهمترین مثالش میتونه برنامه «چت» باشه. متاسفانه پروتوکل HTTP مورد استفاده در وب محدودیت‌هایی برای پیاده‌سازی این گونه برنامه‌ها داره. روش‌های گوناگونی برای پیاده‌سازی برنامه‌های زمان واقعی در وب وجود داره که کتابخونه SignalR فعلا از موارد زیر استفاده میکنه:
تکنولوژی جدید **WebSocket** (^) که خوشبختانه پشتیبانی کاملی از اون در دات نت 4.5 (چهار نقطه پنج! نه چهار و نیم!) وجود داره. اما تمام مرورگرها و تمام وب سرورها از این تکنولوژی پشتیبانی نمیکنند و تنها برخی نسخه‌های جدید قابلیت استفاده از آخرین ورژن WebSocket رو دارند که میشه به کروم 16 به بالا و فایرفاکس 11 به بالا و اینترنت اکسپلورر 10 اشاره کرد (برای استفاده از این تکنولوژی در ویندوز نیاز به **IIS 8.0** است که متاسفانه فقط در ویندوز 8.0 موجوده):

Chrome 16, Firefox 11 and Internet Explorer 10 are currently the only browsers supporting the latest specification (RFC 6455).

یه روش دیگه **Server-sent Events** نام داره که داده‌های جدید رو به فرم رویدادهای DOM به سمت کلاینت میفرسته(^).

روش دیگه‌ای که موجوده به **Forever Frame** معروفه که در این روش یک iframe مخفی درون کد html مسئول تبادل داده‌هاست. این iframe مخفی به‌صورت یک بلاک Chunked (^) به سمت کلاینت فرستاده میشه. این iframe که مسئول رندر داده‌های جدید در سمت کلاینت هست ارتباط خودش رو با سرور تا ابد! (برای همین بهش forever میگن) حفظ میکنه. هر وقت رویدادی سمت سرور رخ میده با استفاده از این روش داده‌ها به‌صورت تگ‌های script به این فریم مخفی فرستاده می‌شوند و چون مرورگرها محتوای html رو به صورت افزایشی (incrementally) رندر میکنن بنابراین این اسکریپتها به‌ترتیب زمان دریافت اجرا می‌شوند. (البته ظاهرا عبارت forever frame در صنعت عکاسی! معروف‌تره بنابراین در جستجو در زمینه این روش ممکنه کمی مشکل داشته باشین) (^).
روش آخر که در کتابخونه SignalR ازش استفاده میشه **long-polling** نام داره. در روش polling معمولی پس از ارسال درخواست توسط کلاینت، سرور بلافاصله نتیجه حاصله رو به سمت کلاینت میفرسته و ارتباط قطع میشه . بنابراین برای داده‌های جدید درخواست جدیدی باید به سمت سرور فرستاده بشه که تکرار این روش باعث افزایش شدید بار بر روی سرور و کاهش کارآمدی اون می‌شه. اما در روش long-polling پس از برقراری ارتباط کلاینت با سرور این ارتباط تا مدت زمان معینی (که توسط یه مقدار تایم اوت مشخص میشه و مقدار پیش‌فرضش 2 دقیقه است) برقرار میمونه . بنابراین کلاینت میتونه بدون ایجاد مشکلی در کارایی، داده‌های جدید رو از سرور دریافت کنه. به این روش در برنامه‌نویسی وب اصطلاحا برنامه‌نویسی کامت (**Comet Programming**) میگن (^ ^).

(البته روش‌های دیگری هم برای پیاده‌سازی برنامه‌های زمان اجرا وجود داره مثل کتابخونه **node.js** که جستجوی بیشتر به خوانندگان واگذار میشه)

SignalR برای برقراری ارتباط ابتدا بررسی میکنه که آیا هر دو سمت سرور و کلاینت قابلیت پشتیبانی از WebSocket رو دارند. در غیراینصورت سراغ روش Server-sent Events میره. اگر باز هم موفق نشد سعی به برقراری ارتباط با روش forever frame میکنه و اگر باز هم موفق نشد در آخر سراغ long-polling میره.
با استفاده از SignalR شما میتونین از سرور، متدهایی رو در سمت کلاینت فراخونی کنین. یعنی درواقع با استفاده از کدهای سی شارپ میشه متدهای جاوااسکریپت سمت کلاینت رو صدا زد!
بطور خلاصه در این کتابخونه دو کلاس پایه وجود داره:
کلاس سطح پایین **PersistentConnection**

کلاس سطح بالای **Hub**

علت این نامگذاری به این دلیله که کلاس سطح پایین پیاده‌سازی پیچیده‌تر و تنظیمات بیشتری نیاز داره اما امکانات بیشتری هم در اختیار برنامه‌نویس قرار می‌ده.
خوب پس از این مقدمه نسبتا طولانی برای دیدن یک مثال ساده میتونین با استفاده از نوگت (Nuget) مثال زیر رو نصب و اجرا کنین (اگه تا حالا از نوگت استفاده نکردین قویا پیشنهاد میکنم که کار رو با دریافتش از اینجا آغاز کنین) :

```
PM> Install-Package SignalR.Sample
```

پس از کامل شدن نصب این مثال اون رو اجرا کنین. این یک مثال فرضی ساده از برنامه نمایش ارزش آنلاین سهام برخی شرکتهاست. میتونین این برنامه رو همزمان در چند مرورگر اجرا کنین و نتیجه رو مشاهده کنین.
حالا میریم سراغ یک مثال ساده. میخوایم یک برنامه چت ساده بنویسیم. ابتدا یک برنامه وب اپلیکیشن خالی رو ایجاد کرده و با استفاده از دستور زیر در خط فرمان نوگت، کتابخونه SignalR رو نصب کنین:

```
PM> Install-Package SignalR
```

پس از کامل شدن نصب این کتابخونه، ریفرنس‌های زیر به برنامه اضافه میشن:

```
Microsoft.Web.Infrastructure
Newtonsoft.Json
SignalR
SignalR.Hosting.AspNet
SignalR.Hosting.Common
```

برای کسب اطلاعات مختصر و مفید از تمام اجزای این کتابخونه به اینجا مراجعه کنین.
همچنین اسکریپت‌های زیر به پوشه Scripts اضافه میشن (این نسخه‌ها مربوط به زمان نگارش این مطلب است):

```
jquery-1.6.4.js
jquery.signalR-0.5.1.js
```

بعد یک کلاس با نام SimpleChat به برنامه اضافه و محتوای زیر رو در اون وارد کنین:

```
using SignalR.Hubs;
namespace SimpleChatWithSignalR
{
  public class SimpleChat : Hub
  {
    public void SendMessage(string message)
    {
      Clients.reciveMessage(message);
    }
```

```
  }
}
```

دقت کنین که این کلاس از کلاس Hub مشتق شده و همچنین خاصیت **Clients** از نوع **dynamic** است. (در مورد جزئیات این کتابخونه در قسمت‌های بعدی توضیحات مفصل‌تری داده میشه)
سپس یک فرم به برنامه اضافه کرده و محتوای زیر رو در اون اضافه کنین:

```
<input type="text" id="msg" />
<input type="button" value="Send" id="send" /><br />
<textarea id='messages' readonly="true" style="height: 200px; width: 200px;"></textarea>
<script src="Scripts/jquery-1.6.4.min.js" type="text/javascript"></script>
<script src="Scripts/jquery.signalR-0.5.1.min.js" type="text/javascript"></script>
<script src="signalr/hubs" type="text/javascript"></script>
<script type="text/javascript">
  var chat = $.connection.simpleChat;
  chat.reciveMessage = function (msg) {
   $('#messages').val($('#messages').val() + "-" + msg + "\r\n");
  };
  $.connection.hub.start();
  $('#send').click(function () {
    chat.sendMessage($('#msg').val());
  });
</script>
```

همونطور که میبینین برنامه چت ما آماده شد! حالا برنامه رو اجرا کنین و با استفاده از دو مرورگر مختلف نتیجه رو مشاهده کنین.
نکته کلیدی کار SignalR در خط زیر نهفته است:

```
<script src="signalr/hubs" type="text/javascript"></script>
```

اگر محتوای آدرس فوق رو دریافت کنین می‌بینین که موتور این کتابخانه تمامی متدهای موردنیاز در سمت کلاینت رو با استفاده از کدهای جاوااسکریپت تولید کرده. البته در این کد تولیدی از نامگذاری camel Casing استفاده میشه، بنابراین متد SendMessage در سمت سرور به‌صورت sendMessage در سمت کلاینت در دسترسه.
امیدوارم تا اینجا تونسته باشم علاقه شما به استفاده از این کتابخونه رو جلب کرده باشم. در قسمت‌های بعد موارد پیشرفته‌تر این کتابخونه معرفی میشه.
اگه علاقه‌مند باشین میتونین از این ویکی اطلاعات بیشتری بدست بیارین.

به روز رسانی

در دوره‌ای به نام SignalR در سایت ، به روز شده‌ای این مباحث را می‌توانید مطالعه کنید.

نظرات خوانندگان

نویسنده: روح الله
تاریخ: ۱۳۹۱/۰۳/۲۹ ۰:۸

جالب بود. اتفاقا امروز با دوستان صحبت از همچین نیازمندی بود. ممنون از زحمات شما.

نویسنده: افشار محبی
تاریخ: ۱۳۹۱/۰۳/۲۹ ۹:۱۷

به نظر فریمورک جذاب و مفیدی میاد. همینجا به آقای نصیری و همکارانش به خاطر راه اندازی وبلاگ جدید تبریک می‌گویم.

نویسنده: علیرضا جهانشاهلو
تاریخ: ۱۳۹۱/۰۳/۲۹ ۹:۳۹

خیلی عالی بود! اگه بدونید این کتابخونه پایه اساس خیلی از ایده‌های تجاریه!؟ که من در حال حاضر مشغول به کار بر روی یکی از همین ایده‌ها هستم.

نویسنده: احمد
تاریخ: ۱۳۹۱/۰۳/۲۹ ۱۹:۵۶

سلام و تبریک برای سایت جدید
از اینکه مطلب بسیار مفید و خوبی گذاشتید بسیار سپاسگزارم.اما اسکریپت signalr/hubs کجا قرار داره؟

نویسنده: Mona
تاریخ: ۱۳۹۱/۰۳/۲۹ ۲۰:۳۵

با سلام خدمت دوستان عزیز
ممنون از مطالب دقیق و کاربردی شما
لطفاً اگر امکان دارد مقایسه ای بین SignalR و Nod.js انجام دهید، زیرا در هرکجای وب که قدم میگذارید بحث از Nod.js است.
ممنون

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۳/۲۹ ۲۱:۴۶

این مسیر درواقع آدرس پیش‌فرض signalr برای کلاس پایه Hub هست که البته میتونه عوض هم بشه. این آدرس درواقع به یه هندلر مپ شده و این هندلر به محض دریافت یه درخواست از سمت کلاینت شروع به گشتن کل اسمبلی برای یافتن کلاسهای مشتق شده از Hub میگرده و برای تمام اجزای public اونا جهت در دسترس بودن در سمت کلاینت متدهای جاوااسکریپت متناظر تولید کرده و در متن پاسخ ارسالی مینویسه (البته یکسری کد دیگه هم برای کار با کلاس Hub تولید میکنه).
برای دریافت این اسکریپت تولیدی، شما میتونین این مسیر رو در نوار آدرس مرورگرتون وارد کنید و کد جاوااسکریپت تولیدی رو مشاهده کنین. حتی میتونین این کد تولیدشده رو در یک فایل ذخیره کرده و به جای ریفرنس اصلی استفاده کنین. البته به شرطی که در اجزای پابلیک کلاسهاتون بعدا تغییری ایجاد نشه.

نویسنده: ramin
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۰:۵۹

سلام
یعنی مسیر و محتوای signalr/hubs در هنگام اجرا تولید میشه؟
من مثال شما رو دنبال کردم ولی خروجی مورد نظر رو نگرفتم

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۱:۱۰

من خودم تخصصی در زمینه node.js ندارم. البته ازش خوشم هم نمیاد :)
اینجوری که خودشون میگن این یه پلتفرمه که از Chrome's JavaScript runtime استفاده میکنه و شونصدتا ویژگی دیگه هم داره! برای استفاده ازش باید برنامشو رو که بهش node میگن دریافت و نصب کنین (حدود 3 مگابایت نسخه 0.6.19). بعد کد سمت سرور کاملا با استفاده از زبان جاوااسکریپت نوشته میشه (عیب) بنابراین کاملا cross platform هستش (مزیت). یعنی با استفاده از جاوااسکریپت میتونین مثلا یه وب سرور بسیار سبک (مزیت) ایجاد کنین که به یکسری درخواستا پاسخ میده. حالا یکسری اومدن برای این پلتفرم ماژولهایی نوشتن مثل socket.io و nowJS که برقراری ارتباطی راحت و آسان با سرور و تبادل داده‌ها رو به عهده میگیره (مزیت).
در مقابل SignalR یه کتابخونه غیرهمزمان (Async) هستش که برا دات نت نوشته شده (مزیت)، مخصوص وب زمان واقعی چندکاربره و شونصدتا ویژگی دیگه هم نداره (مزیت) :). بنابراین سمت سرور از مزایای بیشماری برخورداره، مثل یک IDE قدرتمند (VS) (مزیت2x) و یک کتابخونه کامل (.NetFx) (مزیت2x)
درضمن یه فریمورک دیگه هم برای ASP.NET وجود داره: PokeIn که نسخه تجاری (پولی) هم داره و از WebSocket استفاده میکنه. جایی خوندم که کاراییش خوبه در حد 10 تا 50 هزار کلاینت همزمان(^).

برای SignalR یه اپلیکیشن توسط خود نویسندگانش نوشته شده (SignalR-Crank) که برای آزمایش بار روی سرور استفاده میشه. طبق گفته خودشون تا 100k (صد هزار!) کلاینت رو تونستن بدون هیچ مشکلی راه بندازن (البته با مصرف 5 گیگ رم). نمیدونم این تست چی بوده ولی این عدد خیلی زیاده.

در مورد بحث کارآیی خودم دارم یه تستایی انجام میدم. نمیدونم بتونم آزمایش مشابهی رو بقیه فریمورکها انجام بدم یا نه. اگه نتیجه‌ای حاصل شد اینجا میزارم.

با تشکر

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۳/۳۰ ۱۱:۱۴

اگه برنامه چت داره درست اجرا میشه، مسیر مورد نظر برای دریافت این کد جاوااسکریپت مثلا رو سیستم من اینه:
http://localhost:16869/signalr/hubs

آره در زمان اجرا تولید میشه و بار کوچیکی روی سرور میزاره. برا همین پیشنهاد میشه تو نسخه ریلیز برنامه‌ها این کد تولیدی تو یه فایل ذخیره بشه و به جای اون مسیر ریفرنس داده بشه.

نویسنده: امیرحسین مرجانی
تاریخ: ۱۳۹۱/۰۷/۲۹ ۱:۲

سلام
ممنونم بابت مطلب خوب کاربردیتون
می تونم بگم یکی از سئوال‌های ذهن من پاسخ داده شد و این امکان راه حل مشکل چند هفته ای من بود
تشکر

نویسنده: hossein
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۲:۲۶

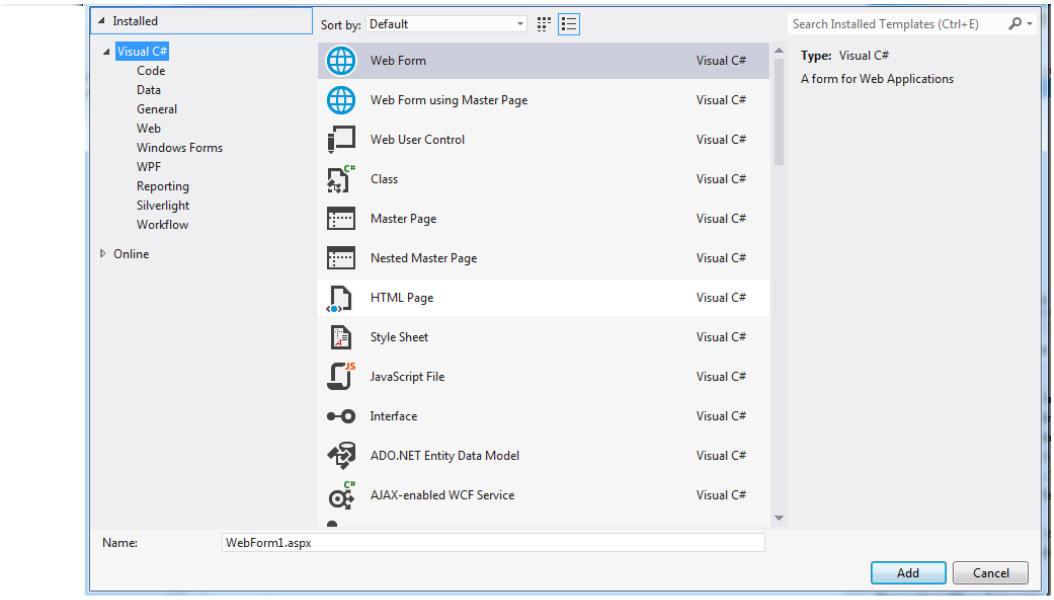
با سلام میخواستم بپرسم ایا این روش (signallr) برای لود کردن مثلا 40-50 تا عکس اسلاید شو کارایی بالاتری داره یا روش lazy loading پلاگینی برای jquery اون وجود داره
با تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۳/۰۸ ۱۲:۳۳

سؤالت ربطی به بحث نداره. کار SiganlR به زبان ساده فرستادن پیغام از طرف سرور به کلاینت‌ها هست. مثلا 2 تا ایمیل جدید داری. رکورد جدیدی به صفحه اضافه شده. یا برای نمونه درصد پیشرفت عملیات سمت سرور مثلا 50 درصد بوده و ارسالش به تمام کلاینت‌های متصل. هدف تبادل اطلاعات همزمان و بلادرنگ و بدون معطلی هست؛ خصوصا از طرف سرور به کلاینت.

نویسنده: ابوالفضل روشن ضمیر
تاریخ: ۱۳۹۲/۰۵/۳۰ ۱۶:۲۵

سلام
من پکیج Jquery و SignalR را در برنامه نصب کردم ... ولی وقتی روی نام پروژه راست کلیک می‌کنم تا کلاس SignalR Hub Class
را اضافه کنم وجود نداره ....
ممنون



نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۳۰ ۱۷:۵۹

- آپدیت سوم VS 2012 را نصب کنید .
+ هیچ نیازی به این پیشنیازها نیست. با VS *2010* کل مطالب SignalR قابل پیاده سازی هستند. فقط باید بتونید با نیوگت کار کنید و بسته‌های لازم رو اضافه کنید. بعد از آن کلاس هاب ، یک کلاس ساده است که امضایی مشخص داره و از کلاسی به نام Hub مشتق می‌شود و همچنین ویژگی HubName هم می‌تواند داشته باشد.

نویسنده: mostafa
تاریخ: ۱۳۹۲/۰۹/۳۰ ۲۳:۱۶

سلام
میشه این رو برای website معمولی visual studio استفاده کرد؟ یعنی غیر از وب اپلیکشن
ممنون

نویسنده: bahman
تاریخ: ۱۳۹۲/۱۰/۰۹ ۱۱:۵۴

سلام.
آیا این امکان وجود داره که SingnalR رو با RestFull سرویس در سمت سرور ترکیب کنم و از سرویس در سمت کلاینت ، داخل یک برنامه WPF استفاده کتم؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۰۹ ۱۳:۴

نگاهی به گزینه‌های مختلف مهیای جهت میزبانی SignalR

نویسنده: عرفان
تاریخ: ۱۳۹۳/۰۳/۲۵ ۱۱:۲۲

(Clients.reciveMessage(message
خطا میده. همچین کلاسی وجود نداره حتی اگه غلط املاییش رو هم درست کنیم!

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۳/۲۵ ۱۱:۳۶

شما که به غلط املاییش دقت کردی، به انتهای بحث که نوشته شده این مباحث به روز شده‌اش در دوره SignalR سایت ارائه شدند، دقت نکردی؟ اون متد dynamic هست؛ یعنی اصلا نیازی نیست وجود خارجی داشته باشه. فقط کمی در نگارش‌های جدید، Refactoring انجام دادن، بعدش باید مشخص کنی به All یا به گروه خاصی این پیام‌ها ارسال بشه. کلیاتش یکی هست. فقط کمی تعاریف اولیه رو Refactor کردن. در بحث معرفی hubs دوره‌ای که نام برده شد این‌ها هست.

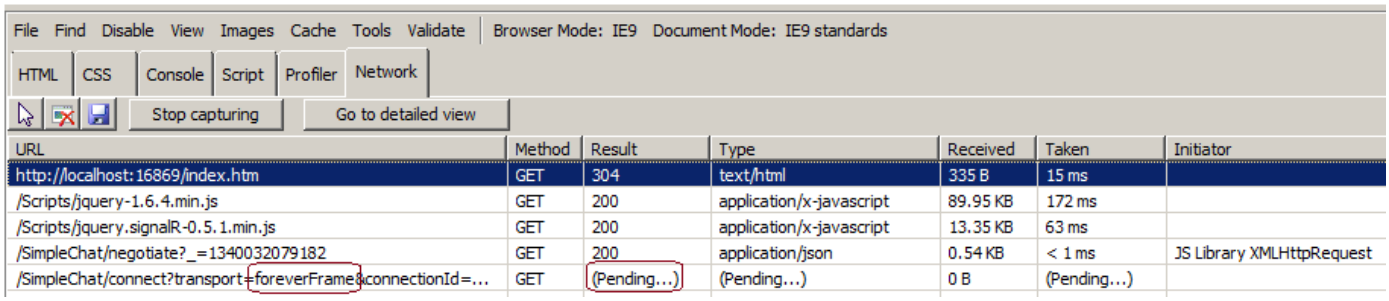
عنوان: **SignalR - قسمت دوم**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۳/۲۹ ۲۱:۳۱
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, SignalR

در قسمت اول بحث‌های مقدماتی درباره وب زمان واقعی (real time web) و معرفی کتابخونه SignalR به همراه یک مثال ساده رو با هم دیدیم. در ادامه به جزئیات ریزی از کتابخونه SignalR که توسط آقای David Fowler توسعه داده میشه میپردازم.

همونطور که قبلا هم اشاره شد قلب این کتابخونه در سمت سرور دو کلاس پایه PersistentConnection و Hub هستن که اولی سطحی پایینتر داره یعنی به تنظیمات و کدنویسی (بسیار) بیشتری برای پیاده‌سازی نیاز داره اما در عوض امکانات سطح پایینتری هم در اختیار برنامه نویس قرار میده که در برخی موارد موردنیاز هستن. در مورد بخشهای مختلف این دو کلاس و نحوه پیاده‌سازی هر دو کلاس فوق، تو آدرسی که قبلا اشاره کردم (آدرس پروژه متن باز این کتابخونه در github) راهنمایی‌های نسبتا مفصلی ارائه شده و نیازی به تکرار این مطالب در اینجا نیست. پیشنهاد میکنم که این مطالب رو با دقت مطالعه کنین.

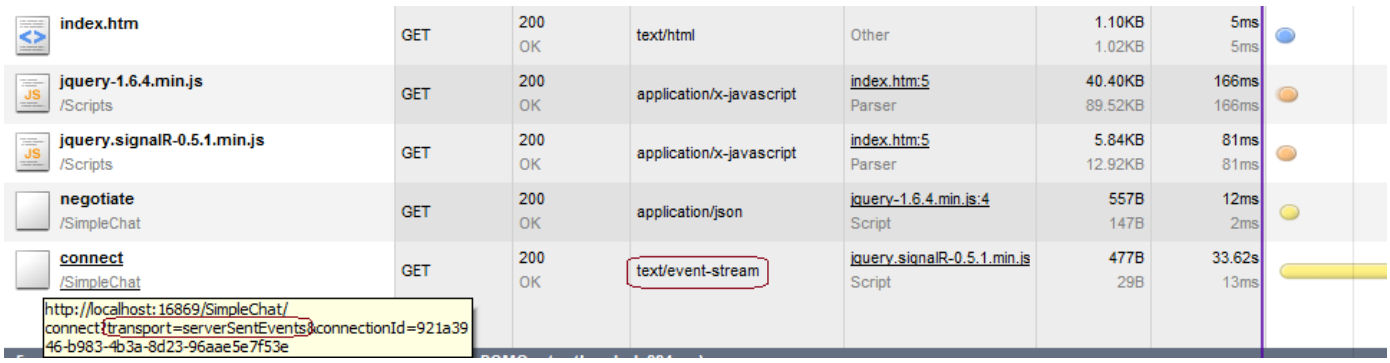
SignalR به صورت توکار از 4 روشی که در قسمت قبل به اون اشاره شد برای برقراری ارتباط استفاده میکنه. WebSocket که به بسترهای جدیدی نیاز داره. Server-sent Events که تنها در مرورگرهایی که پشتیبانی کاملی از html5 دارند قابل استفاده است (بنابراین ie9 نمیتونه از این روش استفاده کنه). forever frame داده‌ها رو به‌صورت chunked (بخشی از استاندارد http 1.1) دریافت میکنه و روش آخر که Long-polling نام داره از هموش روش قدیمی اِیجکس (Ajax) بهره میبره و با استفاده از آبجکت معروف XmlHttpRequest کار ارتباط و تبادل داده‌ها رو انجام میده.

در اینجا برای بررسی این ارتباطات ابتدا برنامه چت ساده قسمت قبل رو در اینترنت اکسپلورر اجرا کنین و با استفاده developer tool (کلید F12) به درخواستهای مختلف ارسال شده به سرور نگاه کنین. پس از اجرای برنامه و قبل ارسال هرگونه داده به سرور در تب Network این ابزار چیزی شبیه به شکل زیر مشاهده خواهید کرد (البته باید قبل از ورود به صفحه برنامه چت روی دکمه Start capturing کلیک کنین):



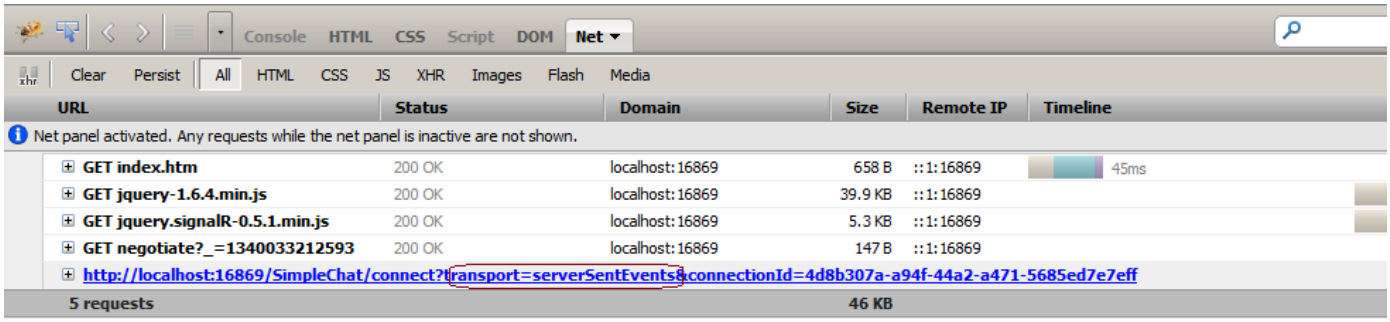
با توجه به تصویر بالا SignalR در شرایط موجود بهترین روش برای برقراری ارتباط با سرور رو forever frame تشخیص داده و مشاهده میشه که این ارتباط دائمیه و فعلا نتیجه‌ای از سمت سرور دریافت نکرده و ارتباط کاملا زنده است. البته اگر در این ابزار درباره درخواستهای ارسالی به سرور بیشتر جستجو بکنین اطلاعات بیشتری نصیبتون میشه که آوردنش اینجا بحث رو طولانی میکنه.

حالا برنامه رو در یه مرورگر دیگه که از html5 پشتیبانی میکنه اجرا کنین. مثلا نتیجه در گوگل کروم و ابزار توسعه اون به شکل زیره:



همونطور که میبینین در اینجا روش استفاده شده Server Sent Events هست.

در فایرفاکس هم با استفاده از ابزار محبوب firebug نتیجه مشابه کروم بدست میاد:



البته اگر علاقه زیادی به کندوکاو در جزئیات این درخواستها دارین (مثل خود من) چیزی بهتر از fiddler2 پیدا نمیشه. میتونین پس از ارسال یک متن دوباره این درخواستها رو مورد بررسی قرار بدین و ببینین که چیجوری کانالهای ارتباط پس از ارسال و دریافت دیتا قطع و برقرار میشه.

این نکته رو هم باید یادآور بشم که هرچند که این کتابخونه بهترین روش رو میتونه انتخاب کنه اما به برنامه نویس امکان تعیین صریح روش ارتباط رو هم میده. اگر به راهنماهای این کتابخونه سر بزنین میبینین که امکانات زیادی بهش اضافه شده و امکانات زیادی هم در آینده به اون اضافه میشه. امکاناتی از قبیل ارسال داده‌ها به یک کلاینت خاص و یا به گروهی خاص از کلاینتها، خصوصی‌سازی آدرس سرور و همچنین پشتیبانی از Cross Domain در آخرین نسخه، امکان استفاده از Reactive Extension (بلاگ)، بحث Self Hosting که امکان خیلی جالبیه و میتونه خیلی جاها یه عنوان یک راه‌حل سبک و سریع به کار بیاد، قابلیت فوق العاده در بایندینگ داده‌ها در سمت سرور و مخصوصا کلاینت، امکان تشخیص برقراری یا قطع ارتباط کلاینتها در سمت سرور، استفاده از امکانات این کتابخونه برای برقراری ارتباط با کلاینتها در خارج از فضای کلاسهای مشتق شده از دو کلاس پایه (Hub و PersistentConnection) و چند مورد دیگه تا نسخه جاری اضافه شدند.

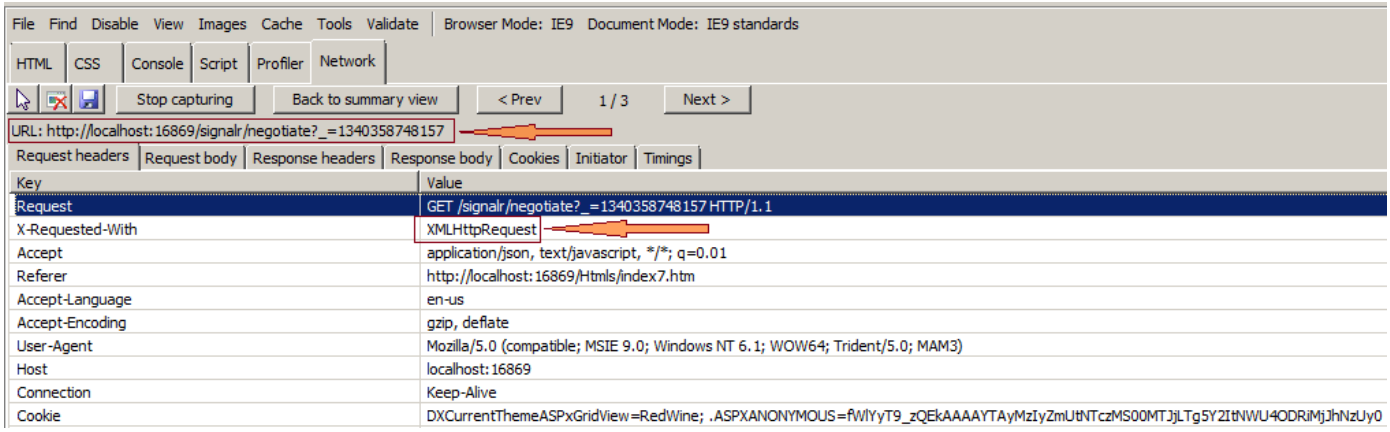
درحال حاضر دارم روی یه برنامه چت با امکانات بیشتر کار میکنم که پس از آماده شدن ارائه میدمش. یکی از پروژه‌های متن بازی که با استفاده از این کتابخونه توسعه داده شده jabbr.net است. یه اتاق گفتگوی کامل با امکانات جالبه که میتونین به اون هم یه سری بزنین.

در آخر هم یه لینک جالب برای مطالعه معرفی میکنم: Highest voted Signalr Questions - stackoverflow

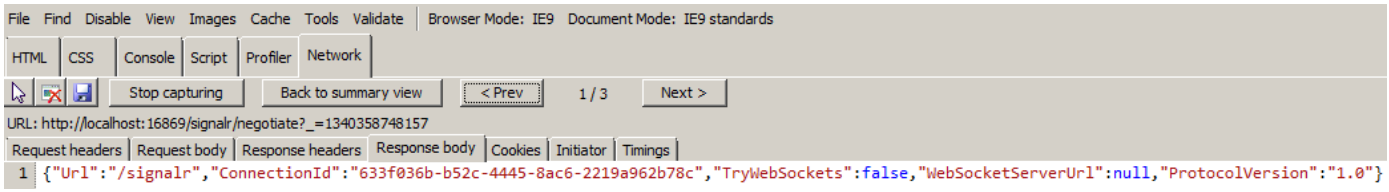
عنوان: **SignalR - قسمت سوم**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۶:۳۱
آدرس: www.dotnettips.info
برچسب‌ها: ASP.Net, SignalR

در قسمت قبل درباره روشهای برقراری ارتباط با سرور در کتابخونه SignalR کمی بحث شد. برای ادامه بهتره که به برنامه چت ساده ای که تو این مدت کمی تکمیلش کردم یه نگاهی بندازین: SimpleChat.rar

لطف کنین این برنامه رو دانلود و اجرا کنین تا کمی با جزئیات این کتابخونه بیشتر آشنا بشین. این برنامه قدم به قدم نوشته شده و حاوی نسخه‌های مختلفی از برنامه چت هست که هر کدوم تو یه فایل html استفاده شده. نسخه آخر شامل عملیات لاگین، چت گروهی، چت خصوصی و امکان تغییر گروه است. درضمن این برنامه کمی با عجله نوشته شده پس اگه باگ یا موردی مشاهده کردین و یا پیشنهادی دارین اشاره کنین تا بقیه هم استفاده کنن.
حالا به یه نکته در مورد آغاز برقراری ارتباط کلاینت با سرور اشاره میکنم. قبل از برقراری این ارتباط (که در قسمت قبل توضیحاتی در این مورد داده شده) برنامه کلاینت یک درخواست به سمت سرور ارسال میکنه. به تصویر زیر دقت کنین:



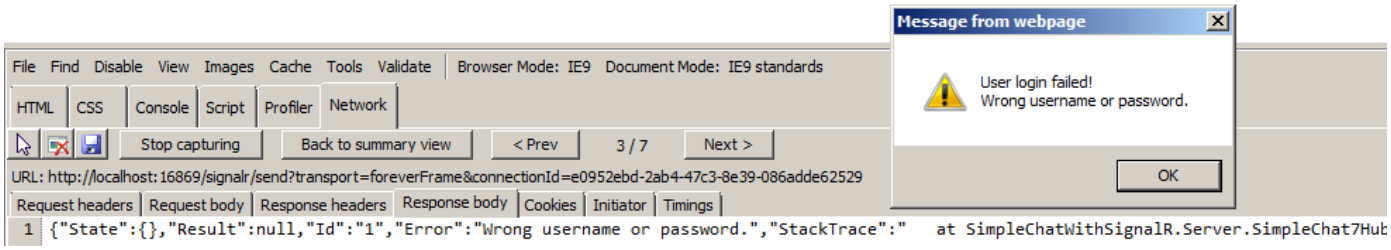
به این درخواست اولیه در کتابخونه SignalR همونطور که مشاهده میشه مذاکره (negotiate) گفته میشه. برقراری ارتباط این درخواست اولیه هم توسط XHR انجام میشه. نتیجه این مذاکره در تب Response Body قابل مشاهده است:



می‌بینین که علاوه بر آی دی ارتباط که یک guid است (تلفظ مرسومش «گوئِد» هستش) امکان برقراری ارتباط از طریق روش WebSocket رو هم از طرف سرور مشخص میکنه که با توجه به استفاده من از ویندوز 7 امکانش وجود نداره. یعنی اگر مثلا شما از ویندوز 8 و IIS 8 استفاده کنین مقدار TryWebSocket برابر true بوده و همچنین پارامتر WebSocketServerUrl نال نخواهد بود. البته این پارامتر تنها مربوط به سروره و برنامه کلاینت مورد استفاده (در اینجا یک مرورگر) هم باید توانایی استفاده از این روش رو داشته باشه. پس از اتمام این مذاکره ارتباط اصلی برقرار میشه.

یکی از قابلیتهای خوب این کتابخونه ارسال خطاهای رخ داده در سمت سرور به کلاینت هست. در تصویر زیر بدنه یک نمونه از

پاسخهای سرور که نمایش دهنده خطای رخ داده در سمت سرور هست رو نشون داده شده:



برای راحتی دوستان در استفاده از راهنماهای این کتابخونه، یه مقدار کار روشون انجام دادم و با خلاصه کردن محتوای اونا (و کاهش حجم 95 درصدی!) برای استفاده آفلاین آماده کردم:

SignarR github docs.rar

اگه فرصتی پیش بیاد و دوستان هم علاقه داشته باشن در قسمت بعدی برنامه چت رو بیشتر با هم بررسی میکنیم.

در ادامه قصد دارم تا روی بازدهی و کارایی این کتابخونه تو بار زیاد یه بررسی هایی انجام بدم (البته اگه وقت کنم چون راه اندازی یه محیط تست برا این جور کتابخونه‌ها چندان آسون نیست) و مطلبی هم در مورد نحوه راه اندازی تست بار ارائه بدم.

نظرات خوانندگان

نویسنده: علی
تاریخ: ۱۳۹۱/۰۴/۰۲ ۱۷:۱۰

مرسی آقای یوسف نژاد, امیدوارم این بحثتون ادامه پیدا کنه

نویسنده: باربد
تاریخ: ۱۳۹۱/۰۴/۰۳ ۸:۱۱

خیلی عالی بود ...
راستش من هم تو فاصله این دو تا مقاله باهاش کار کردم.
چیز خیلی فوق العاده ایه . البته اگر لطف کنید و دو تا ساختار اصلی این مبحث را که Hubs و Persistent Connection را به تفکیک و با جزئیات توضیح بدید ، خیلی خیلی عالی خواهد بود. (فقط یه سوال؟ تو نسخه بعد دات نت ، جزء ساختار اصلی فریم ورک خواهد بود؟)
باز هم متشکرم و موفق باشید

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۴/۰۳ ۹:۱۶

WebSockets Protocol در ASP.NET 4.5 پشتیبانی می‌شود: (^)

نویسنده: باربد
تاریخ: ۱۳۹۱/۰۴/۰۳ ۱۰:۳۵

سپاس :)

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۱/۰۴/۰۳ ۱۵:۵۹

سلام ... عالی بود ... خیلی استفاده بردیم ... برنامه ای که نوشتینم خیلی مفیده ;) ... به نظرم اگه در مورد برنامه ای که نوشتین تو قسمت‌های بعدی سرمایه گذاری کنین خیلی خوب میشه

نویسنده: بنیامین
تاریخ: ۱۳۹۱/۱۰/۱۱ ۱۸:۸

با سلام و خسته نباشید
واقعا ممنون از مطالبی که گذاشتین. خیلی از مشکلات من رو حل کرد. خواهش می‌کنم بحث رو ادامه بدید.
با تشکر

نویسنده: MSN
تاریخ: ۱۳۹۱/۱۰/۲۲ ۱۷:۲۸

خیلی خوب و کاربردی , واقعا خسته نباشید.

سئوال اینجاست , زمانی که پیامی رو ارسال میکنیم دستور زیر رو اجرا میکنیم که این پیام واسه بقیه اعضا هم ارسال بشه

```
public void SendMessage(string message, string id)
{
      Clients.reciveMessage(message, Users[id]);
}
```

زمانی که مثلا توی یک اتاق 10 نفر در یک زمان پیامی رو ارسال میکنند , واسه همه اعضا 10 تا Response جدا میاد ؟ اگه اینطور هست چطور میشه این مورد رو بهینه کرد ؟

نویسنده: میثم ق
تاریخ: ۱۳۹۲/۰۲/۱۵ ۱:۱۱

سلام

خیلی خوب بود.واقعا دستت درد نکنه

امیدوارم این بحث و ادامه بدبن

خیلی ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۵ ۹:۵۸

جهت اطلاع؛ دوره SignalR سایت این مباحث رو بر اساس نگارش جدیدترش تکمیل کرده.

نویسنده: یوسف نژاد
تاریخ: ۱۳۹۲/۰۲/۱۵ ۱۶:۵۷

با توجه به ریلیز شدن نسخه نهایی این کتابخونه با عنوان ASP.NET SignalR و تغییرات بوجود اومده، در حال تهیه مطلبی جدید و نیز بروزرسانی برنامه چت ارائه شده و تهیه چند برنامه جدید مفید دیگه هستم که امیدوارم به زودی آماده بشن.

عنوان: **تخته وایت برد آنلاین توسط SignalR**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۱۱ ۲۱:۵
آدرس: www.dotnettips.info
برچسب‌ها: SignalR

همانطور که در دوره SignalR سایت نیز مطرح شده‌است : " یکی از کاربردهای جالب SignalR می‌تواند به روز رسانی مداوم صفحه نمایش کاربران، توسط اطلاعات ارسالی از طرف سرور باشد. " در ادامه میخواهیم به طراحی یک "تخته وایت برد" آنلاین بپردازیم.
در این پروژه برای ترسم خطوط بر روی صفحه از Canvas در HTML5 استفاده میشود.

پیشنیازها :

پیشنیازهای این مطلب با مطلب « مثال - نمایش درصد پیشرفت عملیات توسط SignalR » یکی است. برای مثال، نحوه دریافت وابستگی‌ها، تنظیمات فایل global.asax و افزودن اسکریپت‌ها، تفاوتی با مقاله‌ی ذکر شده ندارد و تنها تعدادی اسکریپت و CSS جهت زیبایی کار افزوده شده است :

```
<link href="Styles/bootstrap.min.css" rel="stylesheet" />

//برای نمایش و استفاده از جعبه‌ی رنگ در بوت استراپ
<link href="Styles/bootstrap-colorpalette.css" rel="stylesheet" />
<script src="Scripts/jquery-1.8.2.js"></script>
<script type="text/javascript" src='Scripts/jquery.signalR-1.1.3.js'></script>
<script src="Scripts/bootstrap.min.js"></script>
<script src="Scripts/bootstrap-colorpalette.js"></script>
<script type="text/javascript" src='<%: ResolveClientUrl("~/signalr/hubs") %>'></script>

//حاوی متدهایی برای رسم خط با استفاده از HTML5
<script src="Scripts/draw.js" type="text/javascript"></script>

//تعریف کلاینت‌های متصل به هاب
<script src="Scripts/Whiteboard.js"></script>
```

تعریف کلاس Hub برنامه :

```
using Microsoft.AspNet.SignalR;
using Microsoft.AspNet.SignalR.Hubs;
using OnlineSignalRWhiteboard.Model;

namespace OnlineSignalRWhiteboard.Hubs
{
    [HubName("onWhiteboard")]
    public class Whiteboard : Hub
    {
        public void OnDrawPen(Point prev, Point current, string color, int width)
        {
            Clients.All.drawPen(prev, current, color, width);
        }

        public void ClearBoard()
        {
            Clients.All.clear();
        }

    }
}
```

در ابتدا توسط ویژگی HubName یک نام مشخص برای هاب خود انتخاب کرده ایم. سپس متدی به نام OnDrawPen تعریف شده است که پس از فراخوانی از سمت کلاینت، مقادیر دریافتی خود را با صدا زدن متدی در سمت کلاینت به نام drawPen ، به تمام کلاینت‌ها ارسال و نقاط مربوطه در سمت کلاینت، بر روی صفحه رسم میشوند. متد دیگری به نام ClearBoard نیز تعریف شده است که روال رخدادگردان clear در سمت کلاینت را فراخوانی میکند و باعث پاک شدن صفحه نمایش میشود.
کدهای کلاینت‌های متصل به هاب در فایل Whiteboard.js :

```
$(function () {
    $.connection.hub.logging = true;
    var whiteboard = $.connection.onWhiteboard;

    $("#clear").click(function () {
        پاک کردن صفحه نمایش//
        whiteboard.server.clearBoard();
    });

    var color = function (colors) {
        تغییر رنگ قلم//
        draw.colour = colors;
    };

    $(".size").click(function () {
        تغییر سایز قلم//
        draw.lineWidth = $(this).height();
        $('#size').css('height', $(this).height());
    });

    $.connection.hub.start().done(function () {
    })
    .fail(function () {
        alert("Could not Connect!");
     });

    draw.onDraw = function (prev, current, color, width) {
        ارسال پارامترها به سمت سرور تا سرور بتواند داده‌ها را به سمت سایر کلاینت‌ها نیز ارسال کند//
        whiteboard.server.onDrawPen(prev, current, color, width);
    };

    whiteboard.client.drawPen = function (prev, current, color, width) {
        draw.drawPen(prev, current, color, width);
    };

    whiteboard.client.clear = function () {
        draw.clear();
    };

    $('#colorpalette3').colorPalette()
      .on('selectColor', function (e) {
          $('#color').css('background-color', e.color);
          color(e.color);
      });

    var canvas = document.getElementById('draw');
    تغییر سایز کانواس متناسب با پنجره‌ی مرورگر//
    window.addEventListener('resize', resizeCanvas, false);

    function resizeCanvas() {
        canvas.width = window.innerWidth - 20;
        canvas.height = window.innerHeight - 70;
    }
    resizeCanvas();

});
```

در این قطعه کد ابتدا ارجاعی به هاب مشخص شده است و سپس روال‌های رخداد گردانی مانند whiteboard.client.drawPen و whiteboard.client.clear تعریف شده اند که به ترتیب متصل هستند به فراخوانی های Clients.All.drawPen و Clients.All.clear در سمت سرور.

همچنین یک متد به نام draw.onDraw تعریف شده است که وظیفه‌ی آن ارسال اطاعاتی نظیر موقعیت موس در صفحه، رنگ، اندازه و ... به سمت متدی به نام onDrawPen در هاب است.

کدهای کامل سمت کلاینت :

```
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <link href="Styles/bootstrap.min.css" rel="stylesheet" />
    <link href="Styles/bootstrap-colorpalette.css" rel="stylesheet" />
    <script src="Scripts/jquery-1.8.2.js"></script>
    <script type="text/javascript" src='Scripts/jquery.signalR-1.1.3.js'></script>
    <script src="Scripts/bootstrap.min.js"></script>
```

```
    <script src="Scripts/bootstrap-colorpalette.js"></script>
    <script type="text/javascript" src='<%: ResolveClientUrl("~/signalr/hubs") %>'></script>
    <script src="Scripts/draw.js" type="text/javascript"></script>
    <script src="Scripts/Whiteboard.js"></script>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <div style="position: static;">
              <div>
                <div>
                  <a data-toggle="collapse" data-target=".navbar-inverse-collapse">
                    <span></span>
                    <span></span>
                    <span></span>
                  </a>
                  <a href="#">تخته وایت برد آنلاین توسط SignalR و HTML5</a>
                  <div>
                    <ul>
                      <li>
                          <div>
                              <a id="selected-color2" data-toggle="dropdown">
                                  <div style="width: 20px; height: 20px; background: black" id="color">
                                  </div>
                              </a>
                              <ul style="width:293px;">
                                <li style="display:inline-block;">
                                  <div>رنگ پس زمینه</div>
                                  <div id="colorpalette3"></div>
                                </li>
                              </ul>
                          </div>
                      </li>
                      <li></li>
                      <li>
                          <div>
                              <a data-toggle="dropdown" style="width: 120px; height: 20px" href="#">
                                <hr style="width: 120px; height: 1px; background-color: black;margin:
0px; display: table-cell" id="size">
                              </a>
                              <ul style="width: 120px">
                                <li><hr style="width: 140px; height: 1px; background-color:
black"></li>
                                  <li></li>
                                <li><hr style="width: 140px; height: 3px; background-color:
black"></li>
                                  <li></li>
                                <li><hr style="width: 140px; height: 7px; background-color:
black"></li>
                                  <li></li>
                                <li><hr style="width: 140px; height: 10px; background-color:
black"></li>
                                  <li></li>
                                <li><hr style="width: 140px; height: 20px; background-color:
black"></li>
                              </ul>
                            </div>
                      </li>
                      <li></li>
                      <li>
                          <div>
                            <a href="#" id="clear"><i></i>جدید</a>
                          </div>
                      </li>
                    </ul>
                  </div><!-- /.nav-collapse -->
                </div>
              </div><!-- /navbar-inner -->
            </div>
        </div>

        <div>
            <div>
                <canvas id="draw" style="cursor: crosshair;border: 1px solid black;"></canvas>
            </div>
        </div>
    </form>
</body>
</html>
```

در ابتدا یک دکمه برای تغییر رنگ قلم و یک لیست بازشو برای تعیین اندازه‌ی قلم و همچنین یک دکمه برای پاک کردن صفحه نمایش قرار داده شده است. سپس یک Canvas به نام draw ایجاد کرده ایم که بتوانیم خطوط خود را بر روی آن رسم کنیم.
مشاهده نسخه نمایشی

کدهای کامل این مثال را میتوانید از اینجا دانلود کنید : OnlineSignalRWhiteboard.zip و همچنین میتوانید نمونه‌ی بهینه شده‌ی آن را نیز از این قسمت دانلود کنید : OnlineCustomSignalRWhiteboard.zip

نظرات خوانندگان

نویسنده: فاطمه محمدی
تاریخ: ۱۳۹۲/۰۹/۱۱ ۲۱:۵۴

ممنون اما چرا پروژه کامپایل نمیشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۱ ۲۱:۵۹

برای اینکه حجم دریافت شما کمتر شود، پوشه‌ی bin حذف شده. بازسازی آن‌ها یا دریافت آن‌ها هم ساده است. در پیشنیاز بحث، به آن اشاره شده و روش دریافت آن ذکر شده است.

عنوان: **نمایش بلادرنگ اعلامی به تمام کاربران در هنگام درج یک رکورد جدید**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۷/۱۵ ۱۲:۰
آدرس: www.dotnettips.info
گروه‌ها: MVC, SignalR

در ادامه می‌خواهیم اعلام عمومی نمایش افزوده شدن یک پیام جدید را بعد از ثبت رکوردی جدید، به تمامی کاربران متصل به سیستم ارسال کنیم. پیش نیاز مطلب جاری موارد زیر می‌باشند:

دوره "معرفی SignalR و ارتباطات بلادرنگ"

نگاهی به اجزای تعاملی Twitter Bootstrap

ابتدا مدل زیر را در نظر داشته باشید:

```
namespace ShowAlertSignalR.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public string Description { get; set; }
        public float Price { get; set; }
        public Category Category { get; set; }

    }

    public enum Category
    {
        [Display(Name = "دسته بندی اول")]
        Cat1,
        [Display(Name = "دسته بندی دوم")]
        Cat2,
        [Display(Name = "دسته بندی سوم")]
        Cat3
    }
}
```

در اینجا مدل ما شامل عنوان، توضیح، قیمت و یک enum برای دسته‌بندی یک محصول ساده می‌باشد.
کلاس context نیز به صورت زیر می‌باشد:

```
namespace ShowAlertSignalR.Models
{
    public class ProductDbContext : DbContext
    {
        public ProductDbContext() : base("productSample")
        {
            Database.Log = sql => Debug.Write(sql);
        }
        public DbSet<Product> Products { get; set; }
    }
}
```

همانطور که در ابتدا عنوان شد، می‌خواهیم بعد از ثبت یک رکورد جدید، پیامی عمومی به تمامی کاربران متصل به سایت نمایش داده شود. در کد زیر اکشن متد Create را مشاهده می‌کنید:

```
[HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Product product)
        {
            if (ModelState.IsValid)
            {
                db.Products.Add(product);
                db.SaveChanges();
                return RedirectToAction("Index");
            }
```

```
            return View(product);
        }
```

می‌توانیم از ViewBag برای اینکار استفاده کنیم؛ به طوریکه یک پارامتر از نوع bool برای متد Index تعریف کرده و سپس مقدار آن را درون این شیء ViewBag انتقال دهیم، این متغییر بیانگر حالتی است که آیا اطلاعات جدیدی برای نمایش وجود دارد یا خیر؟ بنابراین اکشن متد Index را به اینصورت تعریف می‌کنیم:

```
public ActionResult Index(bool notifyUsers = false)
        {
            ViewBag.NotifyUsers = notifyUsers;
            return View(db.Products.ToList());
        }
```

در اینجا مقدار پیش‌فرض این متغیر، false می‌باشد. یعنی اطلاعات جدیدی برای نمایش موجود نمی‌باشد. در نتیجه اکشن متد Create را به صورتی تغییر می‌دهیم که بعد از درج رکورد موردنظر و هدایت کاربر به صفحه‌ی Index، مقدار این متغییر به true تنظیم شود:

```
[HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create(Product product)
        {
            if (ModelState.IsValid)
            {
                db.Products.Add(product);
                db.SaveChanges();
                return RedirectToAction("Index", new { notifyUsers = true });
            }

            return View(product);
        }
```

قدم بعدی ایجاد یک هاب SignalR می‌باشد:

```
namespace ShowAlertSignalR.Hubs
{
    public class NotificationHub : Hub
    {
        public void SendNotification()
        {
            Clients.Others.ShowNotification();
        }
    }
}
```

در ادامه کدهای سمت کلاینت را برای هاب فوق، داخل ویوی Index اضافه می‌کنیم:

```
@section scripts
{

    <script src="~/Scripts/jquery.signalR-2.0.2.min.js"></script>
    <script src="~/signalr/hubs"></script>
    <script>

        var notify = $.connection.notificationHub;
        notify.client.showNotification = function() {
            $('#result').append("<div class='alert alert-info alert-dismissable'>" +
                "<button type='button' class='close' data-dismiss='alert' aria-
hidden='true'>&times;</button>" +
            "رکورد جدیدی هم اکنون ثبت گردید، برای مشاهده آن صفحه را بروزرسانی کنید" + "</div>");
        };
        $.connection.hub.start().done(function() {
            @{
                if (ViewBag.NotifyUsers)
                {
                    <text>notify.server.sendNotification();</text>
                }
```

```
            }
        });
    </script>
}
```

همانطور که در کدهای فوق مشاهده می‌کنید، بعد از اینکه اتصال با موفقیت برقرار شد (درون متد done) شرط چک کردن متغییر NotifyUsers را بررسی کرده‌ایم. یعنی در این حالت اگر مقدار آن true بود، متد درون هاب را فراخوانی کرده‌ایم. در نهایت پیام به یک div با آی‌دی result اضافه شده است.

لازم به ذکر است برای حالت‌های حذف و به‌روزرسانی نیز روال کار به همین صورت می‌باشد.

سورس مثال جاری : ShowAlertSignalR.zip

نظرات خوانندگان

نویسنده: ایمان قربانی
تاریخ: ۱۳۹۳/۰۷/۱۵ ۱۹:۴۹

سلام - میخواستم ببینم به نظر شما میشه با همین روش طوری عمل کنیم که نیاز به پیغام بروزرسانی صفحه نباشه و رکورد اطلاعاتی زیر همه رکوردها اضافه بشه ؟
احتمالا" ایجکس و اینا نیاز داریم....
کامپوننتی نیست که این مورد رو خودش داشته باشه و نیاز نباشه کد بنویسیم(برای گرید)

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۷/۱۵ ۲۱:۸

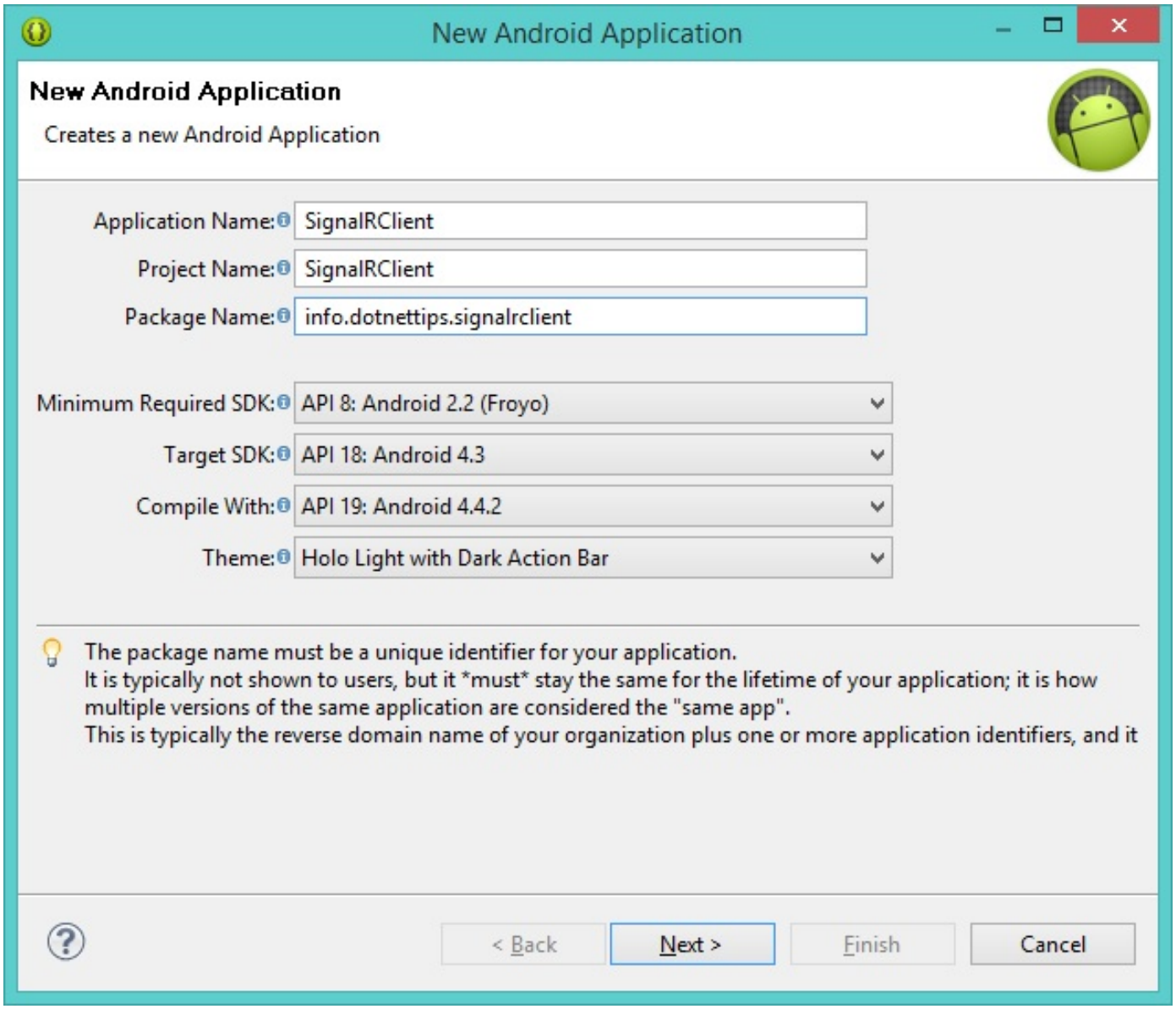
میشه قسمت سمت کلاینت فراخوانی مستقیم هاب رو (ViewBag.NotifyUsers) حذف کرد بجاش از GlobalHost.ConnectionManager.GetHubContext استفاده کرد (قسمت ارسال اطلاعات از یک Hub به Hub دیگر در برنامه).

عنوان: **استفاده از SignalR در اندروید**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۷/۲۱ ۱۳:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SignalR, Android, Java

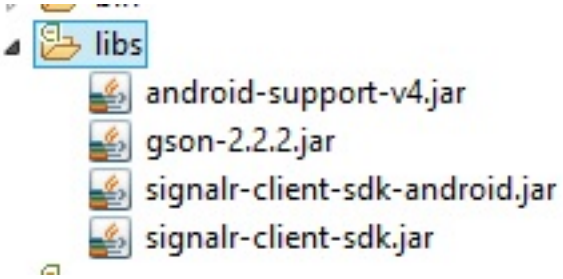
همانطور که مطلع هستید، بخش سورس باز مایکروسافت برای برنامه‌نویس‌های جاوا نیز SDK ی جهت استفاده از SignalR ارائه کرده است. در اینجا می‌توانید مخزن کد آن را در گیت‌هاب مشاهده کنید. هنوز مستنداتی برای این SDK به صورت قدم به قدم ارائه نشده است. لازم به ذکر است که مراجعه به قسمت‌های نوشته شده در اینجا نیز می‌تواند منبع خوبی برای شروع باشد. در ادامه نحوه استفاده از این SDK را با هم بررسی خواهیم کرد.
ابتدا در سمت سرور یک Hub ساده را به صورت زیر تعریف می‌کنیم:

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

برای سمت کلاینت نیز یک پروژه Android Application داخل Eclipse به صورت زیر ایجاد می‌کنیم:



خوب، برای استفاده از SignalR در پروژه‌ی ایجاد شده باید کتابخانه‌های زیر را به درون پوشه libs اضافه کنیم، همچنین باید ارجاعی به کتابخانه Gson نیز داشته باشیم.



قدم بعدی افزودن کدهای سمت کلاینت برای SignalR می‌باشد. دقت داشته باشید که کدهایی که در ادامه مشاهده خواهید کرد دقیقاً مطابق دستورالمعل‌هایی است که قبلاً مشاهده کرده‌اید. برای اینکار داخل کلاس MainActivity.java کدهای زیر را اضافه کنید:

```
Platform.loadPlatformComponent( new AndroidPlatformComponent() );
HubConnection connection = new HubConnection(DEFAULT_SERVER_URL);
HubProxy hub = connection.createHubProxy("ChatHub");
connection.error(new ErrorCallback() {

    @Override
    public void onError(final Throwable error) {
        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_LONG).show();
            }
        });

    }
});
hub.subscribe(new Object() {
    @SuppressWarnings("unused")
    public void messageReceived(final String name, final String message) {

        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(), name + ": " + message,
Toast.LENGTH_LONG).show();
            }
        });
    }
});
connection.start()
.done(new Action<Void>() {

    @Override
    public void run(Void obj) throws Exception {
        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(), "Done Connecting!", Toast.LENGTH_LONG).show();
            }
        });
    }
});
connection.received(new MessageReceivedHandler() {
    @Override
    public void onMessageReceived(final JsonElement json) {
        runOnUiThread(new Runnable() {
            public void run() {
                JsonObject jsonObject = json.getAsJsonObject();
                JsonArray jsonArray =jsonObject.getAsJsonArray("A");
                Toast.makeText(getApplicationContext(), jsonArray.get(0).getAsString() + ": " +
jsonArray.get(1).getAsString(), Toast.LENGTH_LONG).show();
            }
        });
```

```
    }
});
```

همانطور که مشاهده می‌کنید توسط قطعه کد زیر SKD مربوطه در نسخه‌های قدیمی اندروید نیز بدون مشکل کار خواهد کرد:

```
Platform.loadPlatformComponent( new AndroidPlatformComponent() );
```

در ادامه توسط متد createHubProxy ارجاعی به هابی که در سمت سرور ایجاد کردیم، داده‌ایم:

```
HubProxy hub = connection.createHubProxy("ChatHub");
```

در ادامه نیز توسط یک روال رویدادگردان وضعیت اتصال را چک کرده‌ایم. یعنی در زمان بروز خطا در نحوه ارتباط یک پیام بر روی صفحه نمایش داده می‌شود:

```
connection.error(new ErrorCallback() {

    @Override
    public void onError(final Throwable error) {
        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(), error.getMessage(), Toast.LENGTH_LONG).show();
            }
        });

    }
});
```

در ادامه نیز توسط کد زیر متد پویایی که در سمت سرور ایجاد کرده بودیم را جهت برقراری ارتباط با سرور اضافه کرده‌ایم:

```
hub.subscribe(new Object() {
    @SuppressWarnings("unused")
    public void messageReceived(final String name, final String message) {

        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(), name + ": " + message,
Toast.LENGTH_LONG).show();
            }
        });
    }
});
```

برای برقراری ارتباط نیز کدهای زیر را اضافه کرده‌ایم. یعنی به محض اینکه با موفقیت اتصال با سرور برقرار شد پیامی بر روی صفحه‌نمایش ظاهر می‌شود:

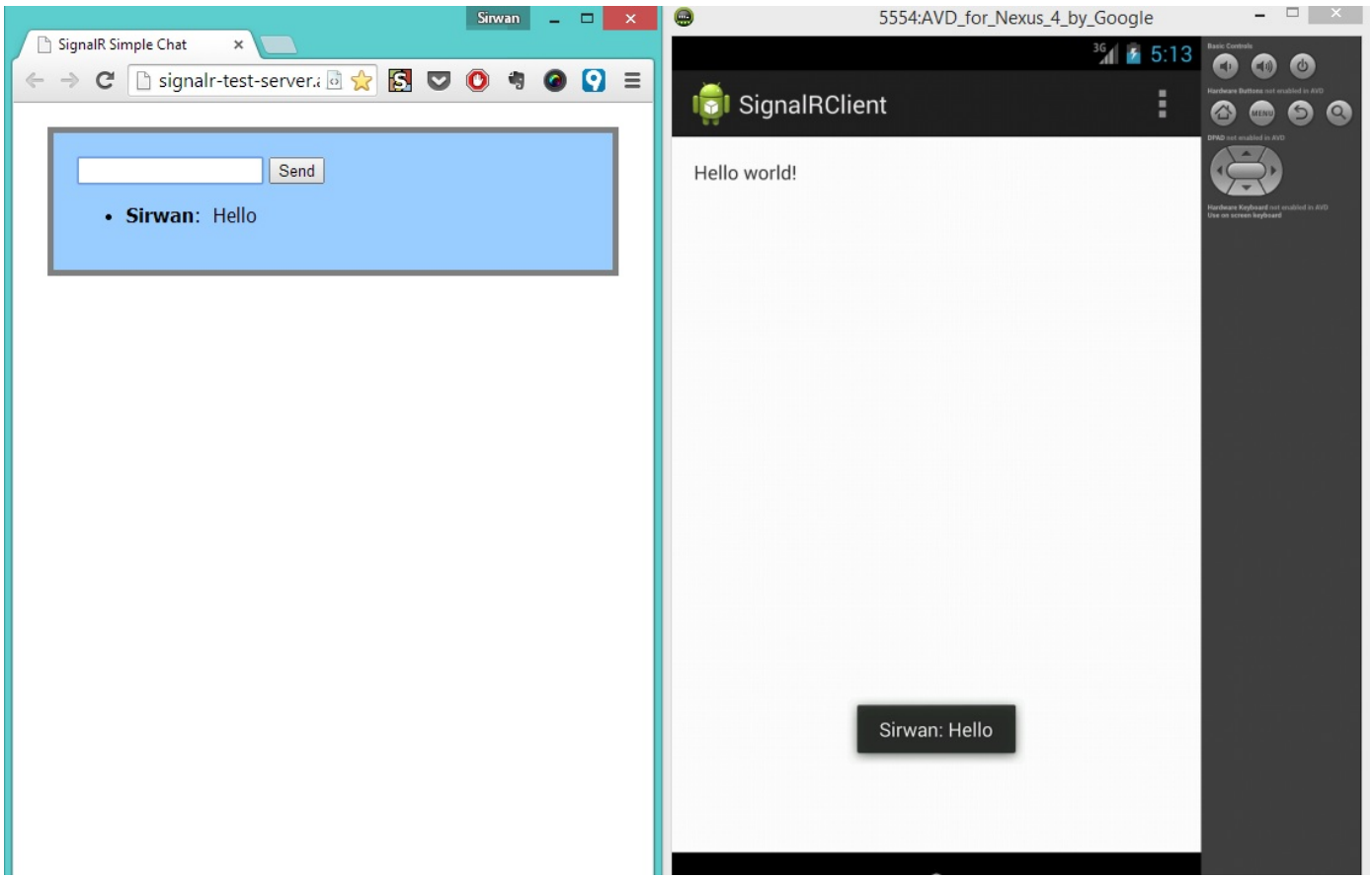
```
connection.start()
.done(new Action<Void>() {

    @Override
    public void run(Void obj) throws Exception {
        runOnUiThread(new Runnable() {
            public void run() {
                Toast.makeText(getApplicationContext(), "Done Connecting!", Toast.LENGTH_LONG).show();
            }
        });
    }
});
```

در نهایت نیز برای نمایش اطلاعات دریافت شده کد زیر را نوشته‌ایم:

```
connection.received(new MessageReceivedHandler() {
    @Override
    public void onMessageReceived(final JsonElement json) {
```

```
        runOnUiThread(new Runnable() {
            public void run() {
                JsonObject jsonObject = json.getAsJsonObject();
                JsonArray jsonArray =jsonObject.getAsJsonArray("A");
                Toast.makeText(getApplicationContext(), jsonArray.get(0).getAsString() + ": " +
jsonArray.get(1).getAsString(), Toast.LENGTH_LONG).show();
            }
        });
    }
});
```

همانطور که عنوان شد کدهای فوق دقیقاً براساس قواعد و دستورالعمل استفاده از SignalR در سمت کلاینت می‌باشد.



نظرات خوانندگان

نویسنده: رشیدیان
تاریخ: ۱۳۹۳/۰۷/۲۱ ۱۵:۴۶

ممنون - بسیار عالی
یک سئوال: آیا از این طریق میشه به همون قابلیت‌های Push Notification در GCM دست یافت؟
و اینکه چقدر این روش قابل اتکا هست؟

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۷/۲۱ ۱۶:۵۱

دقیقاً یکی از استفاده‌هایی که برای خودم داره بحث Push Notification و ارسال پیام به کاربران متصل هست.

عنوان: **افزودن خودکار کلاس های WebAPI و SignalR Hub به برنامه در حالت SelfHost**
نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۳/۰۸/۰۵ ۱۳:۵۵
آدرس: www.dotnettips.info
گروه‌ها: SignalR, ASP.NET Web API, Self-Hosting, ASP.NET Web API 2

یکی از گزینه‌های میزبانی WebAPI و SignalR حالت SelfHost می‌باشد که روش آن قبلا در مطلب « نگاهی به گزینه‌های مختلف مهیای جهت میزبانی SignalR » توضیح داده شده است.

ابتدا نگاه کوچکی به یک مثال داشته باشیم:
هاب زیر را در نظر بگیرید.

```
public class MessageHub : Hub
    {
        public void NotifyAllClients()
        {
            Clients.All.Notify();
        }
}
```

برای selfHsot کردن از یک برنامه‌ی کنسول استفاده می‌کنیم:

```
static void Main(string[] args)
        {
            const string baseAddress = "http://localhost:9000/"; // "http://*:9000/";

            using (var webapp = WebApp.Start<Startup>(baseAddress))
            {
                Console.WriteLine("Start app...");

                var hubConnection = new HubConnection(baseAddress);
                IHubProxy messageHubProxy = hubConnection.CreateHubProxy("messageHub");

                messageHubProxy.On("notify", () =>
                {
                    Console.WriteLine();
                    Console.WriteLine("Notified!");
                });

                hubConnection.Start().Wait();

                Console.WriteLine("Start signalr...");

                bool dontExit = true;
                while (dontExit)
                {
                    var key = Console.ReadKey();
                    if (key.Key == ConsoleKey.Escape) dontExit = false;

                    messageHubProxy.Invoke("NotifyAllClients");
                }

            }
        }
```

با کلاس start-up ذیل:

```
public partial class Startup
    {
        public void Configuration(IAppBuilder appBuilder)
        {
            var hubConfiguration = new HubConfiguration()
            {
                EnableDetailedErrors = true
            };

            appBuilder.MapSignalR(hubConfiguration);

            appBuilder.UseCors(CorsOptions.AllowAll);
```

```
        }
    }
```

اکنون اگر برنامه را اجرا کنیم، با زدن هر کلید در کنسول، یک پیغام چاپ می‌شود که نشان دهنده صحت کارکرد هاب پیام می‌باشد.
خوب؛ تا الان همه چیز درست کار میکند.

صورت مساله:

معمولا برای منظم کردن و مدیریت بهتر کدهای نرم افزار، آن‌ها را در پروژه‌های مجزا یا در واقع همان class libraryهای مجزا نگاه داری میکنیم.
اکنون در برنامه‌ی فوق ، اگر کلاس messageHub را به یک class library دیگر منتقل کنیم و آن را به برنامه‌ی کنسول ارجاع دهیم و برنامه را مجدد اجرا کنیم، با خطای زیر مواجه می‌شویم:

```
{"StatusCode: 500, ReasonPhrase: 'Internal Server Error', Version: 1.1, Content:
System.Net.Http.StreamContent, Headers:\r\n{\r\n  Date: Mon, 27 Oct 2014 09:36:48 GMT\r\n  Server:
Microsoft-HTTPAPI/2.0\r\n  Content-Length: 0\r\n}"}
```

مشکل چیست؟
همانطور که در مطلب « نگاهی به گزینه‌های مختلف مهیای جهت میزبانی SignalR » عنوان شده‌است، «در حالت SelfHost بر خلاف روش asp.net hosting ، اسمبلی‌های ارجاعی برنامه اسکن نمی‌شوند» و طبیعتا مشکل رخ داده شده در بالا از اینجا ناشی می‌شود.

راه حل:

- این کار باید به صورت دستی انجام پذیرد. با افزودن کد زیر به ابتدای برنامه (قبل از شروع هر کدی) اسمبلی‌های مورد نظر افزوده می‌شوند:

```
AppDomain.CurrentDomain.Load(typeof(MessageHub).Assembly.FullName);
```

طبیعتا افزودن دستی هر اسمبلی مشکل و در خیلی مواقع ممکن است با خطای انسانی فراموش کردن مواجه شود!
کد خودکار زیر، میتواند تکمیل کننده‌ی راه حل بالا باشد:

```
class LoadAssemblyHelper
    {
        public static void Load(string searchPattern)
        {
            var path = Assembly.GetExecutingAssembly().Location;
            var entityAssemblies = Directory.GetFiles(Path.GetDirectoryName(path), searchPattern:
searchPattern);
            var assemblyNames = entityAssemblies.Select(e => AssemblyName.GetAssemblyName(e)).ToList();
            assemblyNames.ToList().ForEach(e => AppDomain.CurrentDomain.Load(e));
        }
    }
```

و برای فراخوانی آن در ابتدای برنامه می‌نویسیم:

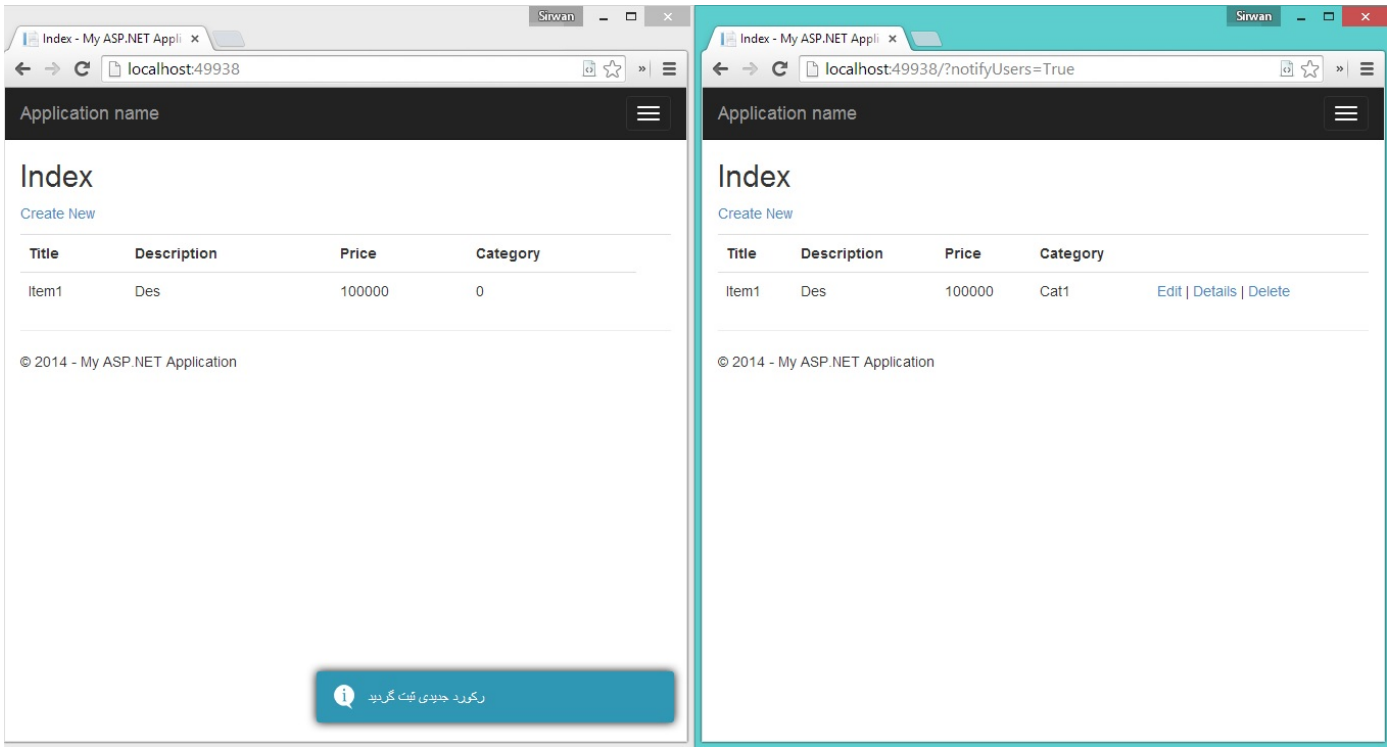
```
static void Main(string[] args)
        {
            //AppDomain.CurrentDomain.Load(typeof(MessageHub).Assembly.FullName);
            //AppDomain.CurrentDomain.Load(typeof(MessageController).Assembly.FullName);

            LoadAssemblyHelper.Load("myFramework.*.dll");

            const string baseAddress = "http://*:9000/";
            using (var webapp = WebApp.Start<Startup>(baseAddress))
            {
                ...
            }
        }
```

نکته‌ی مهم

این خطا و راه حل آن، در مورد hubهای signalr و هم controllerهای webapi صادق می‌باشد.

عنوان: **نمایش بلادرنگ اعلامی به تمام کاربران در هنگام درج یک رکورد جدید به صورت notification**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۱۰/۱۲ ۱۳:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, SignalR

در ادامه می‌خواهیم مثالی را که در این مطلب مورد بررسی قرار گرفت، به صورتی تغییر دهیم که با ثبت یک آیتم جدید درون دیتابیس، یک notification، به تمامی کاربران متصل به هاب ارسال شود. همچنین با کلیک بر روی Notification سطر جدید نیز بلافاصله نمایش داده شود:



در این مثال برای نمایش پیام به صورت notification، از کتابخانه toastr استفاده می‌کنیم که از طریق nuget می‌توانید آن را به پروژه اضافه کنید:

```
PM> Install-Package toastr
```

کار با این کتابخانه خیلی ساده است؛ کافی است فایل‌های js و css آن را به فایل layout اضافه کرده و به این صورت از آن استفاده کنیم:

```
toastr.info("نمایش یک پیام - info");
toastr.success("نمایش یک پیام - success");
toastr.error("نمایش یک پیام - error");
toastr.warning("نمایش یک پیام - warning");
```

دستورات فوق خروجی‌های زیر را نمایش می‌دهد:

برای پیام‌های فوق نیز می‌توانید عنوانی را انتخاب کنید:

```
toastr.success("نمایش یک پیام - success", "عنوان");
```

اگر به فایل js این کتابخانه مراجعه کنید، می‌توانید مقادیر پیش‌فرض آن را برای نمایش یک پیام مشاهده کنید. برای سفارشی‌سازی آن نیز می‌توانید به این صورت عمل کنید:

```
toastr.options = {
            tapToDismiss: true,
            toastClass: 'toast',
            containerId: 'toast-container',
            debug: false,

            showMethod: 'fadeIn', //fadeIn, slideDown, and show are built into jQuery
            showDuration: 300,
            showEasing: 'swing', //swing and linear are built into jQuery
            onShown: undefined,
            hideMethod: 'fadeOut',
            hideDuration: 1000,
            hideEasing: 'swing',
            onHidden: undefined,

            extendedTimeOut: 1000,
            iconClasses: {
                error: 'toast-error',
                info: 'toast-info',
                success: 'toast-success',
                warning: 'toast-warning'
            },
            iconClass: 'toast-info',
            positionClass: 'toast-top-right',
            timeOut: 5000, // Set timeOut and extendedTimeOut to 0 to make it sticky
            titleClass: 'toast-title',
            messageClass: 'toast-message',
            target: 'body',
            closeHtml: '<button>&times;</button>',
            newestOnTop: true,
            preventDuplicates: false,
            progressBar: false
};
```

اکنون برای نمایش این نوع پیام‌ها در زمان اتصال به هاب (در واقع در زمان ثبت یک رکورد جدید) نیاز به ارسال پارامتر خاصی به

سرور (از سمت کلاینت) نمی‌باشد. تنها باید کدهای سمت سرور یعنی هاب را به گونه‌ایی تغییر دهیم تا به محض فراخوانی SendNotification، آخرین رکورد ثبت شده در دیتابیس را به تمامی کلاینت‌های متصل به هاب ارسال کند:

```
public class NotificationHub : Hub
{
        private readonly IProductService _productService;

        public NotificationHub(IProductService productService)
        {
            _productService = productService;
        }

        public void SendNotification()
        {
            Clients.Others.ShowNotification(_productService.GetLastProduct());
        }
}
```

در سمت کلاینت نیز کدها همانند مثال قبل هستند؛ با این تفاوت که در متد سمت کلاینت باید اطلاعات ارسال شده از سمت سرور را با نمایش یک notification به کاربران اطلاع دهیم:

```
var notify = $.connection.notificationHub;
notify.client.showNotification = function (data) {
toastr.info("رکورد جدیدی ثبت گردید جهت نمایش اینجا کلیک کنید");
};
$.connection.hub.start().done(function () {
            @{
                if (ViewBag.NotifyUsers)
                {
                    <text>notify.server.sendNotification();</text>
                }
            }
});
```

تا اینجا همانند مثال قبلی عمل کردیم. یعنی به جای نمایش یک alert بوت‌استرپ، از کتابخانه toastr استفاده گردید. در مثال قبلی کاربر برای دیدن تغییرات می‌بایستی یکبار صفحه را ریفرش کند، اکنون می‌خواهیم کاربر بعد از کلیک بر روی پیام، بلافاصله سطر جدید را نیز مشاهده کند:

```
var positionClasses = {
            topRight: 'toast-top-right',
            bottomRight: 'toast-bottom-right',
            bottomLeft: 'toast-bottom-left',
            topLeft: 'toast-top-left',
            topCenter: 'toast-top-center',
            bottomCenter: 'toast-bottom-center'
        };
        var notify = $.connection.notificationHub;
        notify.client.showNotification = function (data) {
            toastr.options = {
                showDuration: 300,
                positionClass: positionClasses.bottomRight,
                onclick: function () {
                    $('#table tr:last').after("<tr>" +
                    "<td>" + data.Title + "</td>" +
                    "<td>" + data.Description + "</td>" +
                    "<td>" + data.Price + "</td>" +
                    "<td>" + data.Category + "</td>" +
                    "<td>  </td>" +
                    "</tr>");

                }
            };
            toastr.info("رکورد جدیدی ثبت گردید جهت نمایش اینجا کلیک کنید");


        };
        $.connection.hub.start().done(function () {
            @{
                if (ViewBag.NotifyUsers)
                {
                    <text>notify.server.sendNotification();</text>
                }
```

```
            }
});
```

همانطور که مشاهده می‌کنید از onClick برای toastr استفاده کرده‌ایم. با این callback گفته‌ایم که اگر بر روی پیام کلیک شد، اطلاعات را به صورت یک سطر جدید به جدول اضافه کن:

```
onclick: function () {
                    $('#table tr:last').after("<tr>" +
                    "<td>" + data.Title + "</td>" +
                    "<td>" + data.Description + "</td>" +
                    "<td>" + data.Price + "</td>" +
                    "<td>" + data.Category + "</td>" +
                    "<td>  </td>" +
                    "</tr>");

}
```

مقادیر به صورت یک شیء جاوااسکریپتی برگردانده خواهند شد:

```
data {Id: 12, Title: "Item1", Description: "Des", Price: 100000, Category: 0}
```

که توسط data می‌توانیم به هر کدام از فیلدها، جهت نمایش در خروجی، دسترسی داشته باشیم. **دریافت سورس مثال جاری** : ShowAlertSignalR

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۱۰/۱۲ ۱۷:۱۱

نظرتون درباره استفاده از SqlDependency برای اینکار چیه؟ فکر می‌کنید توی یک پروژه بزرگ استفاده از کدامیک بهتر است؟

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۱۰/۱۲ ۲۱:۴۵

SqlDependency محدود هست به SQL Server ضمن اینکه تنظیم آن هم فقط در سمت برنامه نیست. یعنی شخص استفاده کننده باید دسترسی مدیریتی به سرور SQL داشته باشه تا بتونه اون رو تنظیم کنه. ولی زمانیکه با یک ORM کار می‌کنید، تا زمانیکه از API همون ORM استفاده می‌کنید، با عوض کردن پروایدر اون، می‌تونید یک روز از SQL Server و یک روز از اوراکل یا سایر بانک‌های اطلاعاتی استفاده کنید. اینجا است که عدم نیاز به دسترسی مدیریتی و همچنین عمومی‌تر شدن راه حل یک مزیت مهم خواهد شد.

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۳/۱۰/۱۲ ۲۱:۵۳

اگر بخواهیم در یک ویندوز سرویس متوجه تغییرات بشیم چطور؟ یعنی هیچ واسط دیگه ای وجود نداشته باشه

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۱۰/۱۲ ۲۲:۳۸

SignalR محدود به وب نیست: نگاهی به گزینه‌های مختلف مهیای جهت میزبانی SignalR (در مورد سرور) و کلاینت دات نتی هم می‌تونه داشته باشه: نگاهی به SignalR Clients . حتی کلاینت جاوایی هم می‌تونه داشته باشه: استفاده از SignalR در اندروید