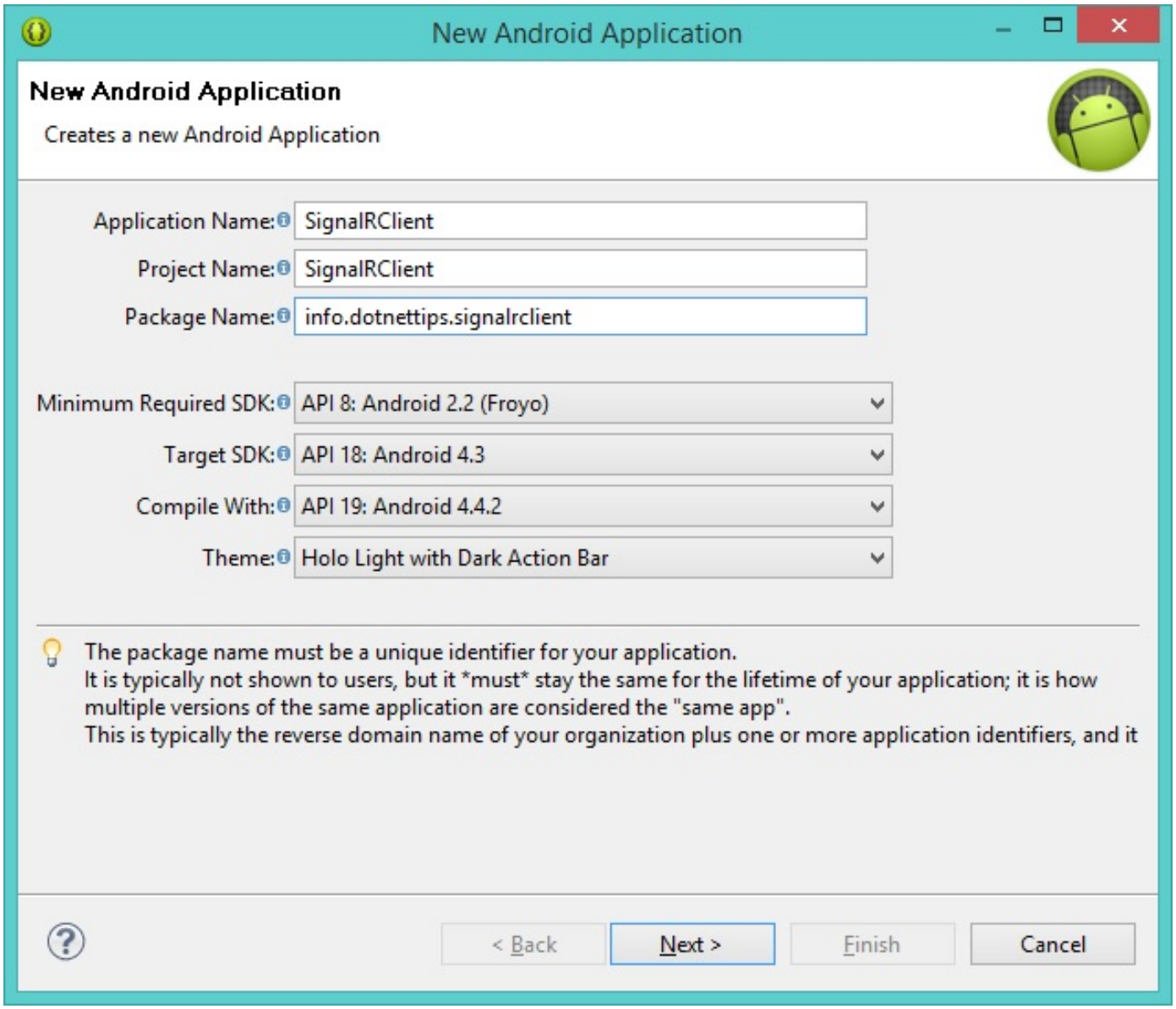
عنوان: **استفاده از SignalR در اندروید**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۷/۲۱ ۱۳:۲۵
آدرس: www.dotnettips.info
گروه‌ها: SignalR, Android, Java

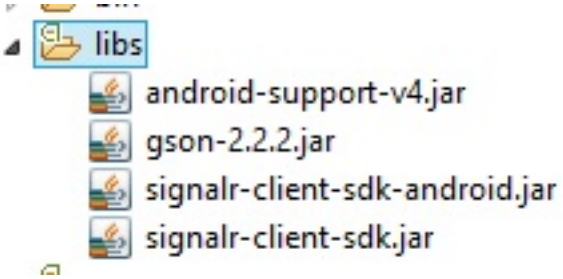
همانطور که مطلع هستید، بخش سورس باز مایکروسافت برای برنامه‌نویس‌های جاوا نیز SDK ی جهت استفاده از SignalR ارائه کرده است. در اینجا می‌توانید مخزن کد آن را در گیت‌هاب مشاهده کنید. هنوز مستنداتی برای این SDK به صورت قدم به قدم ارائه نشده است. لازم به ذکر است که مراجعه به قسمت‌های نوشته شده در اینجا نیز 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:

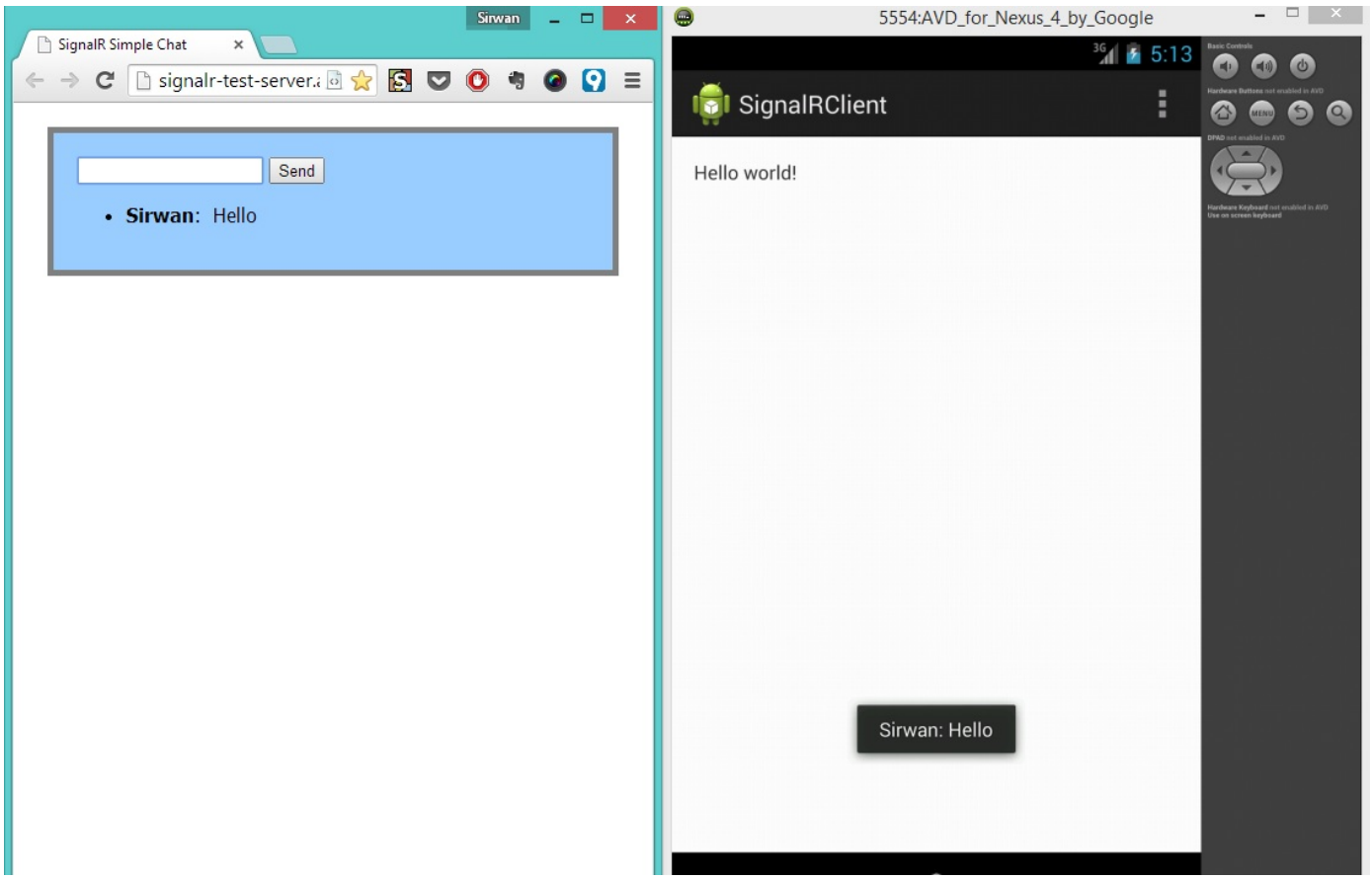
```
connection.start()
.done(new Action<Void>() {

    @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.

نویسنده: علی ساری
تاریخ: ۱۳۹۴/۰۴/۰۳ ۹:۴۵

با تشکر از مطلب خوبتون
در طی این مدت بازخوردهای شما از سیگنال آر در پروژ هایی که ازش استفاده کردید چطور بوده؟
چند برابر بقیه سرویس‌ها مثل gcm یا parse بار روی سرور میاره؟
و اینکه مزیت‌های سیگنال آر در مقایسه با این 2 سرویس چیه (البته parse که میدونم رایگان نیست)
ممنون

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۴/۰۳ ۱۲:۲۰

یکی از مزایای استفاده از SignalR جهت ارسال push notification قابلیت سفارشی‌سازیه، یعنی شما با استفاده از قابلیتی مثل backplan به راحتی می‌تونید messageها رو به سرورهای دیگه فوروارد کنید همچنین می‌تونید از یکسری تکنیک برای داشتن performance بهتر استفاده کنید به طور مثال کاهش سایز اشیاء سریالایز شده و ...
در مجموع انعطاف این روش خیلی بیشتره

عنوان: **کار با وب سرویس جاوایی تشخیص ایمیل‌های موقتی در دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۲۷ ۱۲:۴۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Security, WCF, xml, Validation, Java

یکی از وب سرویس‌های سایت name api ، امکان تشخیص موقتی بودن ایمیل مورد استفاده‌ی جهت ثبت نام در یک سایت را فراهم می‌کند. آدرس WSDL آن نیز در اینجا قرار دارد. اگر مطابق معمول استفاده از سرویس‌های وب در دات نت، بر روی ارجاعات پروژه کلیک راست کرده و گزینه‌ی Add service refrence را انتخاب کنیم و سپس آدرس WSDL یاد شده را به آن معرفی کنیم، بدون مشکل ساختار این وب سرویس دریافت و برای استفاده‌ی از آن به یک چنین کدی خواهیم رسید:

```
var client = new SoapDisposableEmailAddressDetectorClient();

var context = new soapContext
{
    //todo: get your API key here: http://www.nameapi.org/en/register/
    apiKey = "test"
};
var result = client.isDisposable(context, "DaDiDoo@mailinator.com");

if (result.disposable.ToString() == "YES")
{
    Console.WriteLine("YES! It's Disposable!");
}
```

متد isDisposable ارائه شده‌ی توسط این وب سرویس، دو پارامتر context که در آن باید API Key خود را مشخص کرد و همچنین آدرس ایمیل مورد بررسی را دریافت می‌کند. اگر به همین ترتیب این پروژه را اجرا کنید، با خطای Bad request از طرف سرور متوقف خواهید شد:

```
Additional information: The remote server returned an unexpected response: (400) Bad Request.
```

اگر به خروجی این وب سرویس در فیدلر مراجعه کنیم، چنین شکلی را خواهد داشت:

```
<html><head><title>Bad Request</title></head><body><h1>Bad Request</h1><p>No api-key
provided!</p></body></html>
```

عنوان کرده‌است که api-key را، در درخواست وب خود ذکر نکرده‌ایم.
اگر همین وب سرویس را توسط امکانات سایت http://wsdlbrowser.com بررسی کنید، بدون مشکل کار می‌کند. اما تفاوت در کجاست؟
خروجی ارسالی به سرور، توسط سایت http://wsdlbrowser.com به این شکل است:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/">
  <SOAP-ENV:Body>
    <ns1:isDisposable>
      <context>
        <apiKey>test</apiKey>
      </context>
      <emailAddress>sdsdg@site.com</emailAddress>
    </ns1:isDisposable>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

و نمونه‌ی تولید شده‌ی توسط WCF (امکان Add service reference در حقیقت یک WCF Client را ایجاد می‌کند) به صورت زیر می‌باشد:

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
     <isDisposable
```

```
xmlns="http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/">
          <context xmlns=""><apiKey>test</apiKey></context>
          <emailAddress xmlns="">DaDiDoo@mailinator.com</emailAddress>
      </isDisposable>
  </s:Body>
</s:Envelope>
```

از لحاظ اصول XML، خروجی تولیدی توسط WCF هیچ ایرادی ندارد. از این جهت که نام فضای نام مرتبط با http://schemas.xmlsoap.org/soap/envelope/ را به s تنظیم کرده‌است و سپس با استفاده از این نام، Envelope را تشکیل داده‌است. اما ... این وب سرور جاوایی دقیقا با نام SOAP-ENV کار می‌کند و فضای نام ns1 بعدی آن. کاری هم به اصول XML ندارد که باید بر اساس نام xmlns ذکر شده، کار Parse ورودی دریافتی صورت گیرد و نه بر اساس یک رشته‌ی ثابت از پیش تعیین شده. بنابراین باید راهی را پیدا کنیم تا بتوان این s را تبدیل به SOAP-ENV کرد.

برای این منظور به سه کلاس ذیل خواهیم رسید:

```
public class EndpointBehavior : IEndpointBehavior
{
    public void AddBindingParameters(ServiceEndpoint endpoint, BindingParameterCollection
bindingParameters)
    { }

    public void ApplyDispatchBehavior(ServiceEndpoint endpoint, EndpointDispatcher endpointDispatcher)
    { }

    public void Validate(ServiceEndpoint endpoint)
    { }

    public void ApplyClientBehavior(ServiceEndpoint endpoint, ClientRuntime clientRuntime)
    {
        clientRuntime.MessageInspectors.Add(new ClientMessageInspector());
    }
}

public class ClientMessageInspector : IClientMessageInspector
{
    public void AfterReceiveReply(ref Message reply, object correlationState)
    { }

    public object BeforeSendRequest(ref Message request, System.ServiceModel.IClientChannel channel)
    {
        request = new MyCustomMessage(request);
        return request;
    }
}

/// <summary>
/// To customize WCF envelope and namespace prefix
/// </summary>
public class MyCustomMessage : Message
{
    private readonly Message _message;

    public MyCustomMessage(Message message)
    {
        _message = message;
    }

    public override MessageHeaders Headers
    {
        get { return _message.Headers; }
    }

    public override MessageProperties Properties
    {
        get { return _message.Properties; }
    }

    public override MessageVersion Version
    {
        get { return _message.Version; }
    }

    protected override void OnWriteStartBody(XmlDictionaryWriter writer)
    {
```

```
        writer.WriteStartElement("Body", "http://schemas.xmlsoap.org/soap/envelope/");
    }

    protected override void OnWriteBodyContents(XmlDictionaryWriter writer)
    {
        _message.WriteBodyContents(writer);
    }

    protected override void OnWriteStartEnvelope(XmlDictionaryWriter writer)
    {
        writer.WriteStartElement("SOAP-ENV", "Envelope", "http://schemas.xmlsoap.org/soap/envelope/");
        writer.WriteAttributeString("xmlns", "ns1", null, value:
"http://disposableemailaddressdetector.email.services.v4_0.soap.server.nameapi.org/");
    }
}
```

که پس از تعریف client به نحو ذیل معرفی می‌شوند:

```
var client = new SoapDisposableEmailAddressDetectorClient();
client.Endpoint.Behaviors.Add(new EndpointBehavior());
```

توسط EndpointBehavior سفارشی، می‌توان به متد **Envelope** OnWriteStart دسترسی یافت و سپس s آن‌را با SOAP-ENV درخواستی این وب سرویس جایگزین کرد. اکنون اگر برنامه را اجرا کنید، بدون مشکل کار خواهد کرد و دیگر پیام یافت نشدن API-Key را صادر نمی‌کند.

کدهای کامل این مثال را از اینجا می‌توانید دریافت کنید.

عنوان: **بررسی ORM های مناسب جهت استفاده در اندروید**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۰:۵۰
آدرس: www.dotnettips.info
گروه‌ها: ORM, Android, Java

با آمدن ORMها به دنیای برنامه نویسی، کار برنامه نویسی نسبت به قبل ساده‌تر و راحت‌تر شد. عدم استفاده کوئری‌های دستی، پشتیبانی از چند دیتابیس و از همه مهمتر و اصلی‌ترین هدف این ابزار "تنها درگیری با اشیا و مدل شیء گرایی" کار را پیش از پیش آسان‌تر نمود.
در این بین به راحتی می‌توان چندین نمونه از این ORMها را نام برد مثل Nhibernate , Hibernate , IBatis و EF که از معروفترین آن‌ها هستند.
من در حال حاضر قصد شروع یک پروژه اندرویدی را دارم و دوست دارم بجای استفاده‌ی از Sqlitehelper، از یک ORM مناسب بهره ببرم که چند سوال برای من پیش می‌آید. آیا ORM ای برای آن تهیه شده است؟ اگر آری چندتا و کدامیک از آن‌ها بهتر هستند؟ شاید در اولین مورد کتابخانه‌ی Hibernate جاوا را نام ببرید؛ ولی توجه به این نکته ضروری است که ما در مورد پلتفرم موبایل و محدودیت‌های آن صحبت می‌کنیم. یک کتابخانه همانند Hibernate مطمئنا برای یک برنامه اندروید چه از نظر حجم نهایی برنامه و چه از نظر حجم بزرگش در اجرا، مشکل زا خواهد بود و وجود وابستگی‌های متعدد و دارا بودن بسیاری از قابلیت‌هایی که اصلا در بانک‌های اطلاعاتی موبایل قابل اجرا نیست، باعث می‌شود این فریمورک انتخاب خوبی برای یک برنامه اندروید نباشد.

معیارهای انتخاب یک فریم ورک مناسب برای موبایل:
سبک بودن: مهمترین مورد سبک بودن آن است؛ چه از لحاظ اجرای برنامه و چه از لحاظ حجم نهایی برنامه
سریع بودن: مطمئنا ORMهای طراحی شده‌ی موجود، از سرعت خیلی بدی برخوردار نخواهند بود؛ اگر سر زبان هم افتاده باشند. ولی باز هم انتخاب سریع بودن یک ORM، مورد علاقه‌ی بسیاری از ماهاست.
یادگیری آسان و کانفیگ راحت تر.
OrmLight
این فریمورک مختص اندروید طراحی نشده ولی سبک بودن آن موجب شده‌است که بسیاری از برنامه نویسان از آن در برنامه‌های اندرویدی استفاده کنند. این فریم ورک جهت اتصالات JDBC و Spring و اندروید طراحی شده است.

نحوه معرفی جداول در این فریمورک به صورت زیر است:

```
@DatabaseTable(tableName = "users")
public class User {
    @DatabaseField(id = true)
    private String username;
    @DatabaseField
    private String password;

    public User() {
        // ORMLite needs a no-arg constructor
    }
    public User(String username, String password) {
        this.username = username;
        this.password = password;
    }

    // Implementing getter and setter methods
    public String getUserame() {
        return this.username;
    }
    public void setName(String username) {
        this.username = username;
    }
    public String getPassword() {
        return this.password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
}
```

با استفاده از کلمات کلیدی @DatabaseTable در بالای کلاس و @DatabaseField در بالای هر پراپرتی به معرفی جدول و فیلدهای جدول می‌پردازیم.

سورس این فریمورک را می‌توان در گیت هاب یافت و مستندات آن در این آدرس قرار دارند.

SugarORM

این فریمورک مختص اندروید طراحی شده است. یادگیری آن بسیار آسان است و به راحتی به یاد می‌ماند. همچنین جداول مورد نیاز را به طور خودکار خواهد ساخت. روابط یک به یک و یک به چند را پشتیبانی می‌کند و عملیات CURD را با سه متد Save,Delete و Find که البته FindById هم جزء آن است، پیاده سازی می‌کند.

برای استفاده از این فریمورک نیاز است ابتدا متادیتاهای زیر را به فایل manifest اضافه کنید:

```
<meta-data android:name="DATABASE" android:value="my_database.db" />
<meta-data android:name="VERSION" android:value="1" />
<meta-data android:name="QUERY_LOG" android:value="true" />
<meta-data android:name="DOMAIN_PACKAGE_NAME" android:value="com.my-domain" />
```

برای تبدیل یک کلاس به جدول هم از کلاس این فریم ورک ارث بری می‌کنیم:

```
public class User extends SugarRecord<User> {
    String username;
    String password;
    int age;
    @Ignore
    String bio; //this will be ignored by SugarORM

    public User() { }

    public User(String username, String password,int age){
        this.username = username;
        this.password = password;
        this.age = age;
    }
}
```

بر خلاف OrmLight که باید فیلد جدول را معرفی می‌کردید، اینجا تمام پراپرتی‌ها به اسم فیلد شناخته می‌شوند؛ مگر اینکه در بالای آن از عبارت @Ignore استفاده کنید.

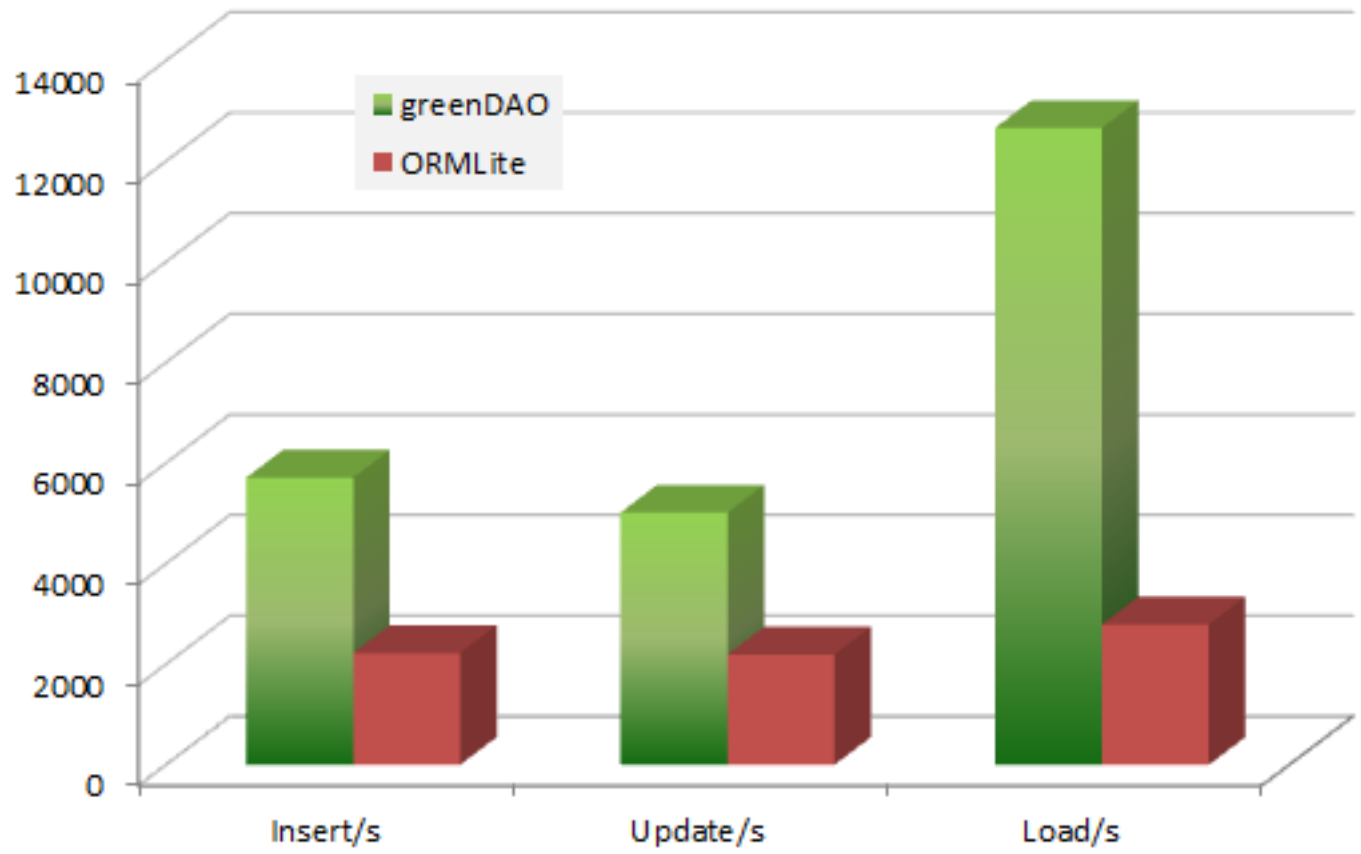
باقی عملیات آن از قبیل اضافه کردن یک رکورد جدید یا حذف رکورد(ها) به صورت زیر است:

```
User johndoe = new User(getContext(),"john.doe","secret",19);
johndoe.save(); //ذخیره کاربر جدید در دیتابیس


//حذف تمامی کاربرانی که سنشان 19 سال است
List<User> nineteens = User.find(User.class,"age = ?",new int[]{19});
foreach(user in nineteens) {
    user.delete();
}
```

برای اطلاعات بیشتر به مستندات آن رجوع کنید.

GreenDAO

موقعیکه بحث کارآیی و سرعت پیش می‌آید نام GreenDAO هست که می‌درخشد. طبق گفته‌ی سایت رسمی آن این فریمورک میتواند در ثانیه چند هزار موجودیت را اضافه و به روزرسانی و بارگیری نماید. این لیست حاوی برنامه‌هایی است که از این فریمورک استفاده می‌کنند. جدول زیر مقایسه‌ای است بین این کتابخانه و OrmLight که نشان میدهد 4.5 برابر سریعتر از OrmLight عمل می‌کند.



غیر از این‌ها در زمینه‌ی حجم هم حرف‌هایی برای گفتن دارد. حجم این کتابخانه کمتر از 100 کیلوبایت است که در اندازه‌ی APK اثر چندانی نخواهد داشت.

آموزش راه اندازی آن در اندروید استادیو ، سورس آن در گیت هاب و مستندات رسمی آن.

Active Android

این کتابخانه از دو طریق فایل JAR و به شیوه maven قابل استفاده است که می‌توانید روش استفاده‌ی از آن را در این لینک ببینید و سورس اصلی آن هم در این آدرس قرار دارد. بعد از اینکه کتابخانه را به پروژه اضافه کردید، دو متادیتای زیر را که به ترتیب نام دیتابیس و ورژن آن هستند، به manifest اضافه کنید:

```
<meta-data android:name="AA_DB_NAME" android:value="my_database.db" />
<meta-data android:name="AA_DB_VERSION" android:value="1" />
```

بعد از آن عبارت ;()ActiveAndroid.Initialize را در اکتیویتی‌های مدنظر اعمال کنید:

```
public class MyActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ActiveAndroid.initialize(this);

        //ادامه برنامه
    }
}
```

برای معرفی کلاس‌ها به جدول هم از دو اعلان Table و Column مانند کد زیر به ترتیب برای معرفی جدول و فیلد استفاده می‌کنیم.

```
@Table(name = "User")
public class User extends Model {
    @Column(name = "username")
    public String username;

    @Column(name = "password")
    public String password;

    public User() {
        super();
    }

    public User(String username,String password) {
        super();
        this.username = username;
        this.password = password;
    }
}
```

جهت اطلاعات بیشتر در مورد این کتابخانه به مستندات آن رجوع کنید.

ORMDroid

از آن دست کتابخانه‌هایی است که سادگی و کم حجم بودن شعار آنان است و سعی دارند تا حد ممکن همه چیز را خودکار کرده و کمترین کانفیگ را نیاز داشته باشد. حجم فعلی آن حدود 20 کیلوبایت بوده و نمی‌خواهند از 30 کیلوبایت تجاوز کند.

برای استفاده‌ی از آن ابتدا دو خط زیر را جهت معرفی تنظیمات به manifest اضافه کنید:

```
<meta-data
  android:name="ormdroid.database.name"
  android:value="your_database_name" />

<meta-data
  android:name="ormdroid.database.visibility"
  android:value="PRIVATE||WORLD_READABLE||WORLD_WRITEABLE" />
```

برای آغاز کار این کتابخانه، عبارت زیر را در هرجایی که مایل هستید مانند کلاس ارث بری شده از Application یا در ابتدای هر اکتیویتی که مایل هستید بنویسید. طبق مستندات آن صدا زدن چندباره این متد هیچ اشکالی ندارد.

```
ORMDroidApplication.initialize(someContext);
```

معرفی مدل جدول بانک اطلاعاتی هم از طریق ارث بری از کلاس Entity می‌باشد .

```
public class Person extends Entity {
  public int id;
  public String name;
  public String telephone;
}

//====================

Person p = Entity.query(Person.class).where("id=1").execute();
p.telephone = "555-1234";
p.save();

// یا

Person person = Entity.query(Person.class).where(eql("id", id)).execute();
p.telephone = "555-1234";
p.save();
```

کد بالا دقیقا یادآوری به EF هست ولی حیف که از Linq پشتیبانی نمی‌شود.

سورس آن در گیت هاب

در اینجا سعی کردیم تعدادی از کتابخانه‌های محبوب را معرفی کنیم ولی تعداد آن به همین جا ختم نمی‌شود. ORMهای دیگری نظیر AndRom و سایر ORM هایی که در این لیست معرفی شده اند وجود دارند.

نکته نهایی اینکه خوب می‌شود دوستانی که از این ORMهای مختص اندروید استفاده کرده اند؛ نظراتشان را در مورد آن‌ها بیان کنند و مزایا و معایب آن‌ها را بیان کنند.

نظرات خوانندگان

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۲:۱۱

Realm هم به نظر گزینه مناسبی هست. یکی از مزیت‌هاش ساده بودنشه:

```
Realm realm = Realm.getInstance(this);

// All writes are wrapped in a transaction
// to facilitate safe multi threading
realm.beginTransaction();

// Add a person
Person person = realm.createObject(Person.class);
person.setName("Young Person");
person.setAge(14);

realm.commitTransaction();

RealmResults<User> result = realm.where(User.class)
                                 .greaterThan("age", 10)  // implicit AND
                                 .beginGroup()
                                      .equalTo("name", "Peter")
                                      .or()
                                      .contains("name", "Jo")
                                 .endGroup()
                                 .findAll();
```

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۴/۰۲/۲۷ ۲۳:۱۱

بله این رو هم دیدم ولی موردی که هست این یک ORM برای sqlite نیست و در واقع این یه لایه برای برقراری ارتباط با دیتابیس درونی خودش هست.

در سایت رسمی خودش هم در صفحه اول نوشته:

```
Realm is not an ORM on top SQLite.
 Instead it uses its own persistence engine,
 built for simplicity (& speed). Users tell us
they get started with Realm in minutes,
port their apps in hours & save weeks on each app.
```

در ابتدا برای IOS نوشتن و بعد هم برای اندروید ولی نکته ای که توی مقالات هست اینه که این دیتابیس به خاطر اینکه کمپایل شده هست و نه مفسری، برای همین سرعت بالاتری داره ولی در مورد اندروید فکر نکنم صحت داشته باشه چون به این صورت وابسته به معماری سی پی یو خواهد شد و ممکن هست روی همه گوشی‌ها جواب نده.

ولی به نظر باید سر یک فرصت مناسب چکش کرد.به هر حال چیز جدید و نابیه و ارزش امتحان کردن رو داره