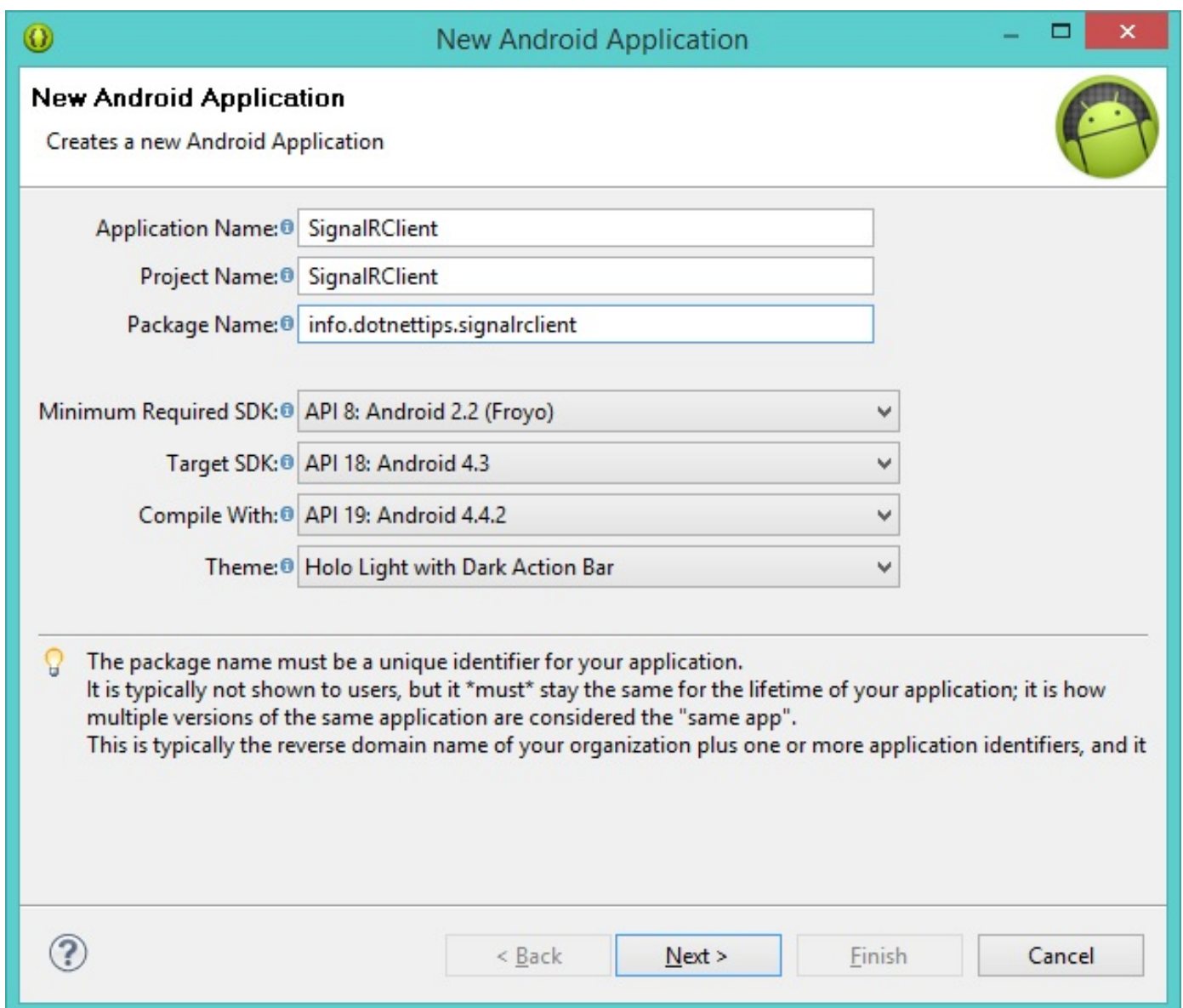


همانطور که مطلع هستید، بخش سورس باز مایکروسافت برای برنامه‌نویس‌های جاوا نیز [SDK](#) ی جهت استفاده از SignalR ارائه کرده است. در [اینجا](#) می‌توانید مخزن کد آن را در گیت‌هاب مشاهده کنید. هنوز مستنداتی برای این SDK به صورت قدم به قدم ارائه نشده است. لازم به ذکر است که مراجعه به قسمت‌های نوشته شده در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



New Android Application
Creates a new Android Application

Application Name:

Project Name:


Package Name:


Minimum Required SDK:

Target SDK:

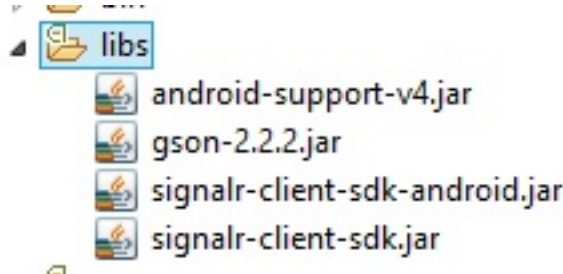
Compile With:

Theme:

 The package name must be a unique identifier for your application. It is typically not shown to users, but it *must* stay the same for the lifetime of your application; it is how multiple versions of the same application are considered the "same app". This is typically the reverse domain name of your organization plus one or more application identifiers, and it



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 دستورالعمل‌هایی 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).getString() + ": " +
                jsonArray.get(1).getString(), 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 صفحه نمایش ظاهر می‌شود:

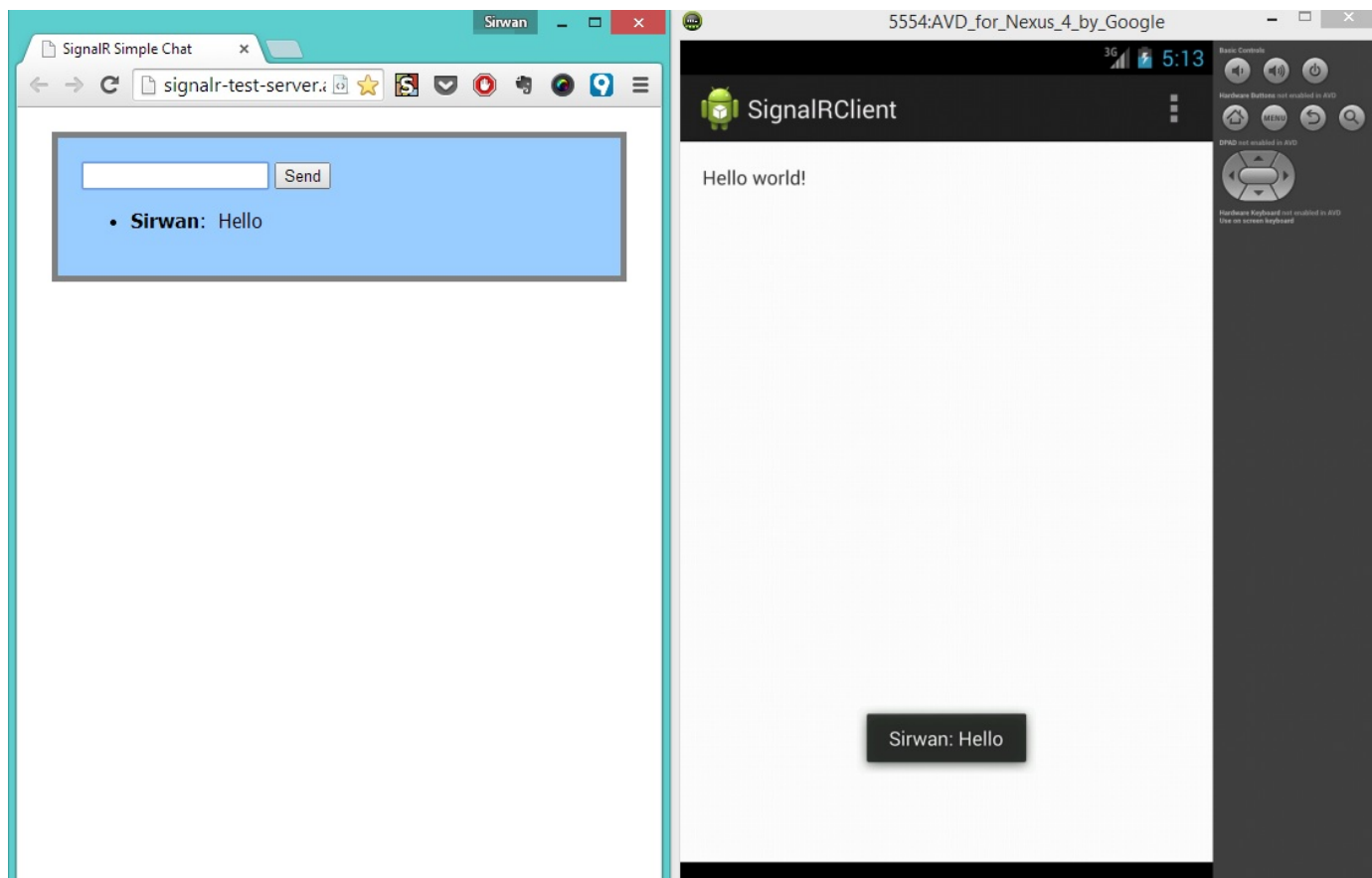
```
connection.start()
    .done(new Action<Void>() {
        @Override
        public void run(Void obj) throws Exception {
            runOnUiThread(new Runnable() {
                public void run() {
                    Toast.makeText(getApplicationContext(), "Done Connecting!", Toast.LENGTH_LONG).show();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

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

```
connection.receive(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).getString() + ": " +  
        jsonArray.get(1).getString(), 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: ۱۵:۴۶ ۱۳۹۳/۰۷/۲۱

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

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

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

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

با تشکر از مطلب خوبتون
در طی این مدت بازخوردهای شما از سیگنال آر در پروژ‌هایی 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 عقیفی
تاریخ: ۱۲:۲۰ ۱۳۹۴/۰۴/۰۳

یکی از مزایای استفاده از SignalR جهت ارسال push notification سفارشی‌سازیه، یعنی شما با استفاده از قابلیت‌های 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

یکی از وب سرویس‌های سایت [name api](http://nameapi)، 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 `Envelope` را تشکیل داده‌است. اما ... این وب سرور جاوایی دقیقاً 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 **OnWriteStart Envelope** دسترسی یافت و سپس آن را 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.

با آمدن ORM ها به دنیای برنامه نویسی، کار برنامه نویسی نسبت به قبل ساده تر و راحت تر شد. عدم استفاده کوئری های دستی، پشتیبانی از چند دیتابیس و از همه مهمتر و اصلی ترین هدف این ابزار "تنها درگیری با اشیا و مدل شیء گرایی" کار را پیش از پیش آسان تر نمود.

در این بین به راحتی می توان چندین نمونه از این ORM ها را نام برد مثل [Nhibernate](#) , [Hibernate](#) , [IBatis](#) و [EF](#) که از معروفترین آن ها هستند.

من در حال حاضر قصد شروع یک پروژه اندرویدی را دارم و دوست دارم بجای استفاده از Sqlitehelper، از یک ORM مناسب بهره ببرم که چند سوال برای من پیش می آید. آیا ORM ای برای آن تهیه شده است؟ اگر آری چندتا و کدامیک از آن ها بهتر هستند؟ شاید در اولین مورد کتابخانه ی Hibernate جاوا را نام ببرید؛ ولی توجه به این نکته ضروری است که ما در مورد پلتفرم موبایل و محدودیت های آن صحبت می کنیم. 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 وابستگی های متعدد و دارا بودن بسیاری از قابلیت هایی که اصلا در بانک های اطلاعاتی موبایل قابل اجرا نیست، باعث می شود 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 ORM های طراحی شده ی موجود، از سرعت خیلی بدی برخوردار نخواهند بود؛ اگر سر زبان هم افتاده باشند.
ولی باز هم انتخاب سریع بودن یک ORM، مورد علاقه ی بسیاری از ماهر است.
یادگیری آسان و کانفیگ راحت تر.

Ormlight

این فریمورک مختص اندروید طراحی نشده ولی سبک بودن آن موجب شده است که بسیاری از برنامه نویسان از آن در برنامه های 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 getUsername() {
        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 می پردازیم.

سورس این فریمورک را می توان 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 می ماند. همچنین جداول مورد نیاز را به طور خودکار خواهد ساخت. روابط یک به یک و یک به چند را پشتیبانی می کند و عملیات CRUD را با سه متد Save, Delete و Find که البته FindById هم جزء آن است، پیاده سازی می کند.

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 می کنیم:

```
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 می کردید، اینجا تمام پراپرتی ها به اسم فیلد شناخته می شوند؛ 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:

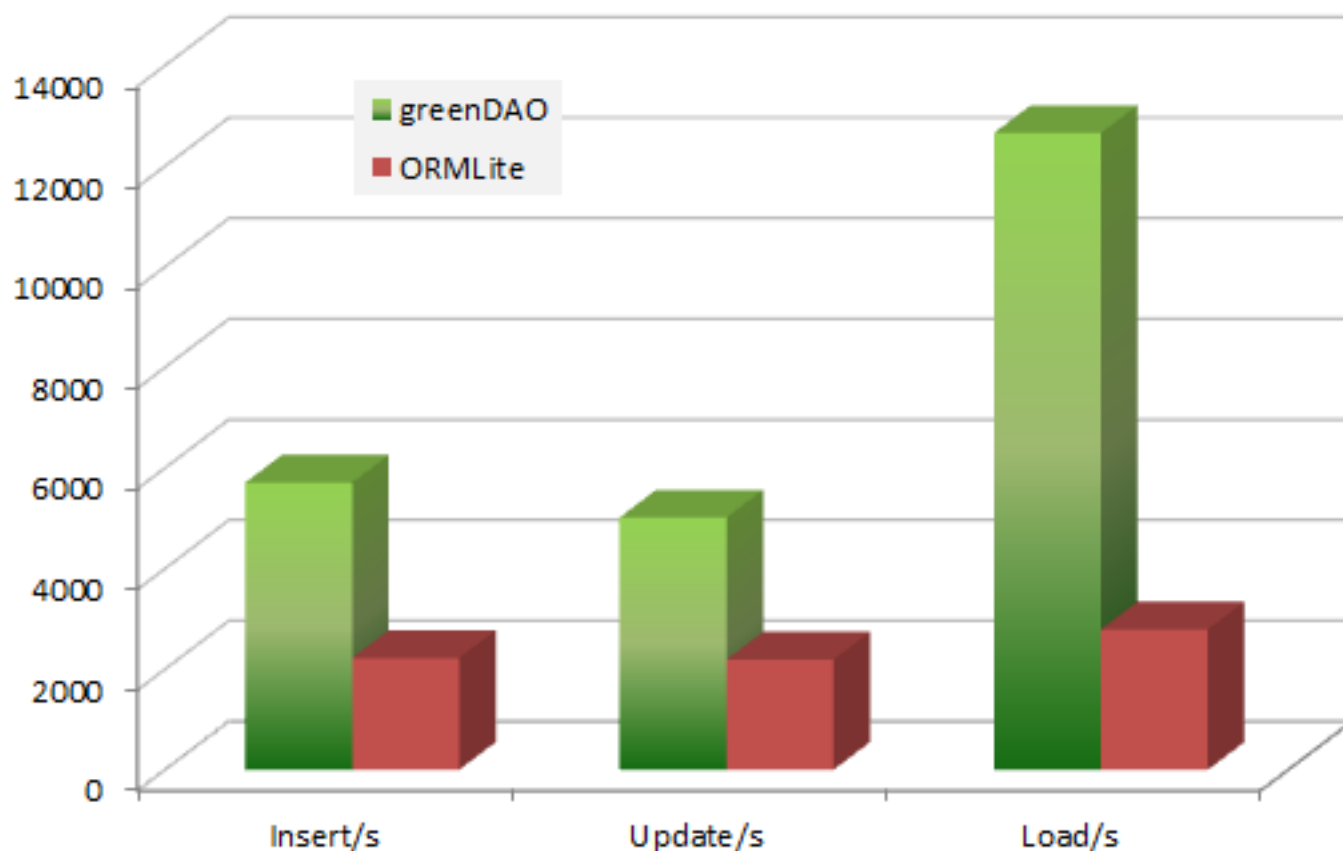
```
User johndoe = new User(getContext(),"john.doe","secret",19);
johndoe.save(); //ذخیره کاربر جدید در دیتابیس

//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 کارایی و سرعت پیش می آید نام GreenDAO هست که می درخشد. [طبق گفتهی سایت رسمی آن](#) این فریمورک میتواند در ثانیه چند هزار موجودیت را اضافه و به روزرسانی و بارگیری نماید. [این لیست](#) حاوی برنامه هایی است که از این فریمورک استفاده می کنند. جدول زیر مقایسه ای است بین این کتابخانه و OrmLight که نشان میدهد 4.5 برابر سریعتر از OrmLight عمل می کند.



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 اکتیویته‌های 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(eq1("id", id)).execute();
p.telephone = "555-1234";
p.save();
```

کد بالا دقیقاً 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. ORM های 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 ORM های 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: ۲۲:۱۱ ۱۳۹۴/۰۲/۲۷

[Realm](#) هم به نظر گزینه مناسبی هست. یکی از مزیت‌هایش ساده بودنش:

```
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: ۲۳:۱۱ ۱۳۹۴/۰۲/۲۷

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

زمانی که سیستم عامل های GUI مثل ویندوز به بازار آمدند، یکی از قسمت‌های گرافیکی آن‌ها AddressBar نام داشت که مسیر حرکت آن‌ها را در فایل سیستم نشان میداد و در سیستم عامل‌های متنی CLI با دستور cd یا pwd انجام می‌شد. بعدها در وب هم همین حرکت با نام BreadCrumb صورت گرفت که به عنوان مثال مسیر رسیدن به صفحه‌ی یک محصول یا یک مقاله را نشان می‌داد. در یک پروژه‌ی اندرویدی نیاز بود تا یک ساختار درختی را پیاده سازی کنم، ولی در برنامه‌های اندروید ایجاد یک درخت، کار هوشمندانه و مطلوبی نیست و روش کار به این صورت است که یک لیست از گروه‌های والد را نمایش داده و با انتخاب هر آیتم لیست به آیتم‌های فرزند تغییر میکند. حالا مسئله این بود که کاربر باید مسیر حرکت خودش را بشناسد. به همین علت مجبور شدم یک [BreadCrumb](#) را برای آن طراحی کنم که در زیر تصویر آن را مشاهده می‌کنید.



از نکات جالب توجه در مورد این ماژول می‌توان گفت که قابلیت این را دارد تا تصمیمات خود را بر اساس اندازه‌های مختلف صفحه نمایش بگیرد. به عنوان مثال اگر آیتم‌های بالا بیشتر از سه عدد باشد و در صفحه جا نشود از یک مسیر جعلی استفاده می‌کند و همه‌ی آیتم‌ها با اندیس شماره 1 تا index-3 را درون یک آیتم با عنوان (...) قرار می‌دهد که من به آن می‌گویم مسیر جعلی. به عنوان نمونه مسیر تصویر بالا در صفحه جا شده است و نیازی به این کار دیده نشده است. ولی تصویر زیر از آن جا که مسیر، طول width صفحه نمایش رد کرده است، نیاز است تا چنین کاری انجام شود. موقعی که کاربر آیتم ... را کلیک کند، مسیر باز شده و به محل index-3 حرکت می‌کند. یعنی دو مرحله به عقب باز می‌گردد.

حضرت علی اکبر ع و مادرش

مصائب

...

صفحه اصلی



ز رفتن تو رود جانم از بدن بیرون



نگاهی به کارکرد ماژول

قبل از توضیح در مورد سورس، اجازه دهید نحوه‌ی استفاده از آن را ببینیم.

این سورس شامل دو کلاس است که ساده‌ترین کلاس آن `AndBreadCrumbItem` می‌باشد که مشابه کلاس `ListItem` در بخش وب دات نت است و دو مقدار، یکی متن و دیگری `Id` را می‌گیرد:

سورس:

```
public class AndBreadCrumbItem {
    private int Id;
    private String diplayText;

    public AndBreadCrumbItem(int Id, String displayText)
    {
        this.Id=Id;
        this.diplayText=displayText;
    }
    public String getDiplayText() {
        return diplayText;
    }
    public void setDiplayText(String diplayText) {
        this.diplayText = diplayText;
    }
    public int getId() {
        return Id;
    }
    public void setId(int id) {
        Id = id;
    }
}
```

به عنوان مثال می‌خواهیم یک breadcrumb را با مشخصات زیر بسازیم:

```
AndBreadCrumbItem itemhome=new AndBreadCrumbItem(0,"Home");
AndBreadCrumbItem itemproducts=new AndBreadCrumbItem(12,"Products");
AndBreadCrumbItem itemdigital=new AndBreadCrumbItem(15,"Digital");
AndBreadCrumbItem itemhdd=new AndBreadCrumbItem(56,"Hard Disk Drive");
```

حال از کلاس اصلی یعنی `AndBreadCrumb` استفاده می‌کنیم و آیتم‌ها را به آن اضافه می‌کنیم:

```
AndBreadCrumb breadCrumb=new AndBreadCrumb(this);

breadCrumb.AddNewItem(itemhome);
breadCrumb.AddNewItem(itemproducts);
breadCrumb.AddNewItem(itemdigital);
```



```
breadCrumb.AddNewItem(itemhdd);
```

به این نکته دقت داشته باشید که با هر شروع مجدد چرخه‌ی Activity، حتماً شیء Context این کلاس را به روز نمایید تا در رسم المان‌ها به مشکل برنخورد. می‌توانید از طریق متد زیر context را مقداردهی نمایید:

```
breadCumb.setContext(this);
```

هر چند راه حل پیشنهادی این است که این کلاس را نگهداری ننماید و از یک لیست ایستا جهت نگهداری AndBreadCrumbItem ها استفاده کنید تا با هر بار فراخوانی رویدادهای اولیه چون onCreate یا onStart و.. شیء BreadCrumb را پر نمایید.

پس از افزودن آیتم‌ها، تنظیمات زیر را اعمال نمایید:

```
LinearLayout layout=(LinearLayout)getActivity().findViewById(R.id.breadcumblayout);
layout.setPadding(8, 8, 8, 8);
breadCrumb.setLayout(layout);
breadCrumb.SetTinyNextNodeImage(R.drawable.arrow);
breadCrumb.setTextSize(25);
breadCrumb.SetViewStyleId(R.drawable.list_item_style);
```

در سه خط اول، یک layout از نوع Linear جهت رسم اشیاء به شیء breadcrumb معرفی می‌شود. سپس در صورت تمایل می‌توانید از یک شیء تصویر گرافیکی کوچک هم استفاده کنید که در تصاویر بالا می‌بینید از تصویر یک فلش جهت دار استفاده شده است تا بین هر المان ایجاد شده از آیتم‌ها قرار بگیرد. سپس در صورت تمایل اندازه‌ی قلم متون را مشخص می‌کنید و در آخر هم متد SetViewStyleId هم برای نسبت دادن یک استایل یا selector و ... استفاده می‌شود. حال برای رسم آن متد UpdatePath را صدا می‌زنیم:

```
breadCrumb.UpdatePath();
```

الان اگر برنامه اجرا شود باید breadcrumb از چپ به راست رسم گردد. برای استفاده‌های فارسی، راست به چپ می‌توانید از متد زیر استفاده کنید:

```
breadCrumb.setRTL(true);
```

در صورت هر گونه تغییری در تنظیمات، مجدداً متد UpdatePath را فراخوانی کنید تا عملیات رسم، با تنظیمات جدید آغاز گردد.

در صورتیکه قصد دارید تنظیمات بیشتری چون رنگ متن، فونت متن و ... را روی هر المان اعمال کنید، از رویداد زیر استفاده کنید:

```
breadCrumb.setOnTextViewUpdate(new ITextViewUpdate() {
    @Override
    public TextView UpdateTextView(Context context, TextView tv) {
        tv.setTextColor(...);
        tv.setTypeface(...);
        return tv;
    }
});
```

با هر بار ایجاد المان که از نوع TextView است، این رویداد فراخوانی شده و تنظیمات شما را روی آن اجرا می‌کند. همچنین در صورتیکه می‌خواهید بدانید کاربر بر روی چه عنصری کلیک کرده است، از رویداد زیر استفاده کنید:

```
breadCumb.setOnClickListener(new IClickListener() {
    @Override
    public void onClick(int position, int Id) {
        //...
    }
});
```

```
});
```

کد بالا دو آرگومان را ارسال میکند که اولی position یا اندیس مکانی عنصر کلیک شده را بر می‌گرداند و دومی id هست که با استفاده از کلاس AndBreadCrumbItem به آن پاس کرده‌اید. هنگام کلیک کاربر روی عنصر مورد نظر، برگشت به عقب به طور خودکار صورت گرفته و عناصر بعد از آن موقعیت، به طور خودکار حذف خواهند شد.

آخرین متد موجود که کمترین استفاده را دارد، متد SetNoResize است. در صورتیکه این متد با True مقداردهی گردد، عملیات تنظیم بر اساس صفحه‌ی نمایش لغو می‌شود. این متد برای زمانی مناسب است که به عنوان مثال شما از یک HorizontalScrollView استفاده کرده باشید. در این حالت layout شما هیچ گاه به پایان نمی‌رسد و بهتر هست عملیات اضافه را لغو کنید.

نگاهی به سورس

کلاس زیر شامل بخش‌های زیر است:

فیلدهای خصوصی

```
//----- Private Properties -----
private List<AndBreadCrumbItem> items=null;
private List<TextView> textViews;
private int tinyNextNodeImage;
private int viewStyleId;
private Context context;
private boolean RTL;
private float textSize=20;
private boolean noResize=false;

LinearLayout layout;
IClickListener clickListener;
ITextViewUpdate textViewUpdate;
LinearLayout.LayoutParams params ;
```

با نگاهی به نام آن‌ها می‌توان حدس زد که برای چه کاری استفاده می‌شوند. به عنوان نمونه از اصلی‌ترین‌ها، متغیر items جهت نگهداری آیتم‌های پاس شده استفاده می‌شود و textviews هم برای نگهداری هر breadcrumb یا همان المان TextView که روی صفحه رسم می‌شود.

اینترفیس‌ها هم با حرف I شروع و برای تعریف رویدادها ایجاد شده‌اند. در ادامه از تعدادی متد get و Set برای مقدار دهی بعضی از فیلدهای خصوصی بالا استفاده شده است:

```
//----- Constructor -----

public AndBreadCrumb(Context context)
{
    this.context=context;
    params = new LinearLayout.LayoutParams
        (LinearLayout.LayoutParams.WRAP_CONTENT, LinearLayout.LayoutParams.WRAP_CONTENT);
}

//----- Public Properties -----

//each category would be added to create path
public void AddNewItem(AndBreadCrumbItem item)
{
    if(items==null)
        items=new ArrayList<>();
    items.add(item);
}

// if you want a pointer or next node between categories or textviews
public void SetTinyNextNodeImage(int resId) {this.tinyNextNodeImage=resId;}

public void SetViewStyleId(int resId) {this.viewStyleId=resId;}
```

```

public void setTextColor(float textSize) {this.textColor = textSize;}

public boolean isRTL() {
    return RTL;
}

public void setRTL(boolean RTL) {
    this.RTL = RTL;
}

public void setLayout(LinearLayout layout) {
    this.layout = layout;
}

public void setContext(Context context) {
    this.context = context;
}

public boolean isNoResize() {
    return noResize;
}

public void setNoResize(boolean noResize) {
    this.noResize = noResize;
}

```

بعد از آن به متدهای خصوصی می‌رسیم که متد زیر، متد اصلی ما برای ساخت breadcrumb است:

```

//primary method for render objects on layout
private void DrawPath() {

    //stop here if essential elements aren't present
    if (items == null) return ;
    if (layout == null) return;
    if (items.size() == 0) return;

    //we need to get size of layout,so we use the post method to run this thread when ui is ready
    layout.post(new Runnable() {
        @Override
        public void run() {

            //textviews created here one by one
            int position = 0;
            textViews = new ArrayList<>();
            for (AndBreadCrumbItem item : items) {
                TextView tv = MakeTextView(position, item.getId());
                tv.setText(item.getDisplayText());
                textViews.add(tv);
                position++;
            }

            //add textviews on layout
            AddTextViewsOnLayout();

            //we dont manage resizing anymore
            if(isNoResize()) return;

            //run this code after textviews Added to get widths of them
            TextView last_tv=textViews.get(textViews.size()-1);
            last_tv.post(new Runnable() {
                @Override
                public void run() {
                    //define width of each textview depend on screen width
                    BatchSizeOperation();
                }
            });
        }
    });
}

```

متد DrawPath برای ترسیم breadcrumb است و می‌توان گفت اصلی‌ترین متد این کلاس است. در سه خط اول، عناصر الزامی را که باید مقداردهی شده باشند، بررسی می‌کند. این موارد وجود آیتم‌ها و layout است. اگر هیچ یک از اینها مقداردهی نشده باشند، عملیات رسم خاتمه می‌یابد. بعد از آن یک پروسه‌ی UI جدید را در متد post شیء Layout معرفی می‌کنیم. این متد زمانی این پروسه را صدا می‌زند که layout در UI برنامه جا گرفته باشد. دلیل اینکار این است که تا زمانی که ویوها در UI تنظیم نشوند، نمی‌توانند اطلاعاتی چون پهنا و ارتفاع را برگردانند و همیشه مقدار 0 را باز می‌گردانند. پس ما بامتد post اعلام می‌کنیم زمانی این پروسه را اجرا کن که وضعیت UI خود را مشخص کرده‌ای.

به عنوان نمونه کد زیر را ببینید:

```
TextView tv=new TextView(this);
tv.getWidth(); //return 0
layout.add(tv);
tv.getWidth(); //return 0
```

در این حالت کنترل در هر صورتی عدد ۰ را به شما باز می‌گرداند و نمی‌توانید اندازه‌ی آن را بگیرید مگر اینکه درخواست یک callback بعد از رسم را داشته باشید که این کار از طریق متد post انجام می‌گیرد:

```
TextView tv=new TextView(this);
tv.post(new Runnable() {
    @Override
    public void run() {
        tv.getWidth(); //return x
    }
});
```

در اینجا مقدار واقعی x بازگردانده می‌شود.

باز می‌گردیم به متد DrawPath و داخل متد post

در اولین خط این پروسه به ازای هر آیتم، یک TextView توسط متد MakeTextView ساخته می‌شود که شامل کد زیر است:

```
private TextView MakeTextView(final int position, final int Id)
{
    //settings for crumbs
    TextView tv=new TextView(this.context);
    tv.setEllipsize(TextUtils.TruncateAt.END);
    tv.setSingleLine(true);
    tv.setTextSize(TypedValue.COMPLEX_UNIT_PX, textSize);
    tv.setBackgroundResource(viewStyleId);

    /*call custom event - this event will be fired when user click on one of
    textviews and returns position of textview and value that user sat as id
    */
    tv.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            SetPosition(position);
            clickListener.onClick(position, Id);
        }
    });

    //if user wants to update each textviews
    if(textViewUpdate!=null)
        tv=textViewUpdate.UpdateTextView(context,tv);

    if(isRTL())
        tv.setRotationY(180);

    return tv;
}
```

در خطوط اولیه، یک TextView ساخته و متد Ellipsize را با Truncate.END مقداردهی می‌نماید. این مقداردهی باعث می‌شود

اگر متن، در TextView جا نشد، ادامه‌ی آن با ... مشخص شود. در خط بعدی TextView را تک خطه معرفی می‌کنیم. در خط بعدی اندازه‌ی قلم را بر اساس آنچه کاربر مشخص کرده است، تغییر می‌دهیم و بعد هم استایل را برای آن مقداردهی می‌کنیم. بعد از آن رویداد کلیک را برای آن مشخص می‌کنیم تا اگر کاربر بر روی آن کلیک کرد، رویداد اختصاصی خودمان را فراخوانی کنیم. در خط بعدی اگر rtl با true مقدار دهی شده باشد، textview را حول محور Y چرخش می‌دهد تا برای زبان‌های راست به چپ چون فارسی آماده گردد و در نهایت TextView ساخته شده و به سمت متد DrawPath باز می‌گرداند.

بعد از ساخته شدن TextView ها، وقت آن است که به Layout اضافه شوند که وظیفه‌ی اینکار بر عهده‌ی متد AddTextViewOnLayout است:

```
//this method calling by everywhere to needs add textviews on the layout like master method :drawpath
private void AddTextViewsOnLayout()
{
    //prepare layout
    //remove everything on layout for recreate it
    layout.removeAllViews();
    layout.setOrientation(LinearLayout.HORIZONTAL);
    layout.setVerticalGravity(Gravity.CENTER_VERTICAL);
    if(isRTL())
        layout.setRotationY(180);

    //add textviews one by one

    int position=0;
    for (TextView tv:textViews)
    {
        layout.addView(tv,params);

        //add next node image between textviews if user defined a next node image
        if(tinyNextNodeImage>0)
            if(position<(textViews.size()-1)) {
                layout.addView(GetNodeImage(), params);
                position++;
            }
    }
}
```

در چند خط اول، Layout آماده سازی می‌شود. این آماده سازی شامل پاکسازی اولیه Layout یا خالی کردن ویوهای درون آن است که می‌تواند از رندر قبلی باشد. افقی بودن جهت چینش Layout، در مرکز نگاه داشتن ویوها و نهایتا چرخش حول محور Y در صورت true بودن خاصیت RTL است. در خطوط بعدی یک حلقه وجود دارد که TextView های ایجاد شده را یک به یک در Layout می‌چیند و اگر کاربر تصویر گرافیکی را هم به (همان فلش‌های اشاره‌گر) متغیر tinyNextNodeImage نسبت داده باشد، آن‌ها را هم بین TextView ها می‌چیند و بعد از پایان یافتن کار، مجدداً به متد DrawPath باز می‌گردد.

تا به اینجا کار چیدمان به ترتیب انجام شده است ولی از آنجا که اندازه‌ی Layout در هر گوشی و در دو حالت حالت افقی یا عمودی نکه داشتن گوشی متفاوت است، نمی‌توان به این چینش اعتماد کرد که به چه نحوی عناصر نمایش داده خواهند شد و این مشکل توسط متد BatchSizeOperation (تغییر اندازه دسته جمعی) حل می‌گردد. در اینجا هم باز متد post به آخرین textview اضافه شده است. به این علت که موقعی که همه‌ی textview ها در ui جا خوش کردند، بتوانیم به خاصیت‌های ui آن‌ها دسترسی داشته باشیم. حالا بعد از ترسیم باید اندازه آن‌ها را اصلاح کنیم. قدم به قدم متد BatchSizeOperation را بررسی می‌کنیم:

```
//set textview width depend on screen width
private void BatchSizeOperation()
{
    //get width of next node between cumbs
    Bitmap tinyBmap = BitmapFactory.decodeResource(context.getResources(), tinyNextNodeImage);
    int tinysize=tinyBmap.getWidth();
    //get sum of nodes
    tinysize*=(textViews.size()-1);
    ...
}
```

ابتدا لازم است طول مسیری که همه ویوها یا المان‌های ما را دارند، به دست آوریم. اول از تصویر کوچک شروع می‌کنیم و پهنای آن را می‌گیریم. سپس عدد به دست آمده را در تعداد آن ضرب می‌کنیم تا جمع پهنای آن را داشته باشیم. سپس نوبت به TextViewها می‌رسد.

```
//get width size of screen(layout is screen here)
int screenWidth=GetLayoutWidthSize();

//get sum of arrows and cumbs width
int sumtvs=tinysize;
for (TextView tv : textViews) {

    int width=tv.getWidth();
    sumtvs += width;
}
```

در ادامه‌ی این متد، متد GetLayoutWidthSize را صدا می‌زنیم که وظیفه‌ی آن برگرداندن پهنای layout است و کد آن به شرح زیر است:

```
private int GetLayoutWidthSize()
{
    int width=layout.getWidth();
    int padding=layout.getPaddingLeft()+layout.getPaddingRight();
    width-=padding;
    return width;
}
```

در این متد پهنای paddingها را چپ و راست به دست می‌آید و مقدار آن را به عنوان اندازه‌ی صفحه نمایش، تحویل متد والد می‌دهد. در ادامه هم پهنای هر TextView محاسبه شده و جمع کل آن‌ها را با اندازه‌ی صفحه مقایسه می‌کند. اگر کوچکتر بود، کار این متد در اینجا تمام می‌شود و نیازی به تغییر اندازه نیست. ولی اگر نبود کد ادامه می‌یابد:

```
private void BatchSizeOperation()
{
    ....

    //if sum of cumbs is less than screen size the state is good so return same old textviews
    if(sumtvs<screenWidth)
        return ;

    if(textViews.size()>3)
    {
        //make fake path
        MakeFakePath();

        //clear layout and add textviews again
        AddTextViewsOnLayout();
    }

    //get free space without next nodes -> and spilt rest of space to textviews count to get space
    for each textview
    {
        int freespace =screenWidth-tinysize;
        int each_width=freespace/textViews.size();

        //some elements have less than each_width,so we should leave size them and calculate more space
        again
        {
            int view_count=0;
            for (TextView tv:textViews)
            {
                if (tv.getWidth()<=each_width)
                    freespace=freespace-tv.getWidth();
                else
                    view_count++;
            }
            if (view_count==0) return;

            each_width=freespace/view_count;
            for (TextView tv:textViews)
            {
                if (tv.getWidth()>each_width)
```

```

        tv.setWidth(each_width);
    }

}

```

اگر آیتم‌ها بیشتر از سه عدد باشند، می‌توانیم از حالت مسیر جعلی استفاده کنیم که توسط متد `MakeFakePath` انجام می‌شود. البته بعد از آن هم باید دوباره `view`‌ها را چینش کنیم تا مسیر جدید ترسیم گردد، چون ممکن است بعد از آن باز هم جا نباشد یا آیتم‌ها بیشتر از سه عدد نیستند. در این حالت، حداقل کاری که می‌توانیم انجام دهیم این است که فضای موجود را بین آن‌ها تقسیم کنیم تا همه‌ی کاسه، کوزه‌ها سر آیتم آخر نشکند و متنش به ... تغییر یابد و حداقل از هر آیتم، مقداری از متن اصلی نمایش داده شود. پس میانگین فضای موجود را گرفته و بر تعداد المان‌ها تقسیم می‌کنیم. البته این را هم باید در نظر گرفت که در تقسیم بندی، بعضی آیتم‌ها آن مقدار پهنا را نیاز ندارند و با پهناي کمتر هم می‌شود کل متنشان را نشان داد. پس یک کار اضافه‌تر این است که مقدار پهناي اضافی آن‌ها را هم حساب کنیم و فقط آیتم‌هایی را پهنا دهیم که به مقدار بیشتری از این میانگین احتیاج دارند. در اینجا کار به پایان می‌رسد و مسیر نمایش داده می‌شود.

نحوه‌ی کارکرد متد `MakeFakePath` بدین صورت است که 4 عدد `TextView` را ایجاد کرده که المان‌های با اندیس 0 و 2 و 3 به صورت نرمال و عادی ایجاد شده و همان کارکرد سابق را دارند. ولی المان شماره دو با اندیس 1 با متن ... نماینده‌ی آیتم‌های میانی است و رویداد کلیک آن به شکل زیر تحریف یافته است:

```

//if elements are so much(mor than 3),we make a fake path to decrease elements
private void MakeFakePath()
{
    //we make 4 new elements that index 1 is fake element and has a rest of real path in its heart
    //when user click on it,path would be opened
    textViews=new ArrayList<>(4);
    TextView[] tvs=new TextView[4];
    int[] positions= {0,items.size()-3,items.size()-2,items.size()-1};

    for (int i=0;i<4;i++)
    {
        //request for new textviews
        tvs[i]=MakeTextView(positions[i],items.get(positions[i]).getId());

        if(i!=1)
            tvs[i].setText(items.get(positions[i]).getDisplayText());
        else {
            tvs[i].setText("...");
            //override click event and change it to part of code to open real path by call
            //setposition method and redraw path
            tvs[i].setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    int pos = items.size() - 3;
                    int id = items.get(pos).getId();
                    SetPosition(items.size() - 3);
                    clickListener.onClick(pos, id);
                }
            });
        }
        textViews.add(tvs[i]);
    }
}

```

این رویداد با استفاده از `setPosition` به آیتم `index-3` بازگشته و مجدداً المان‌ها رسم می‌گردند و سپس رویداد کلیک این آیتم را هم اجرا می‌کند و المان‌های با اندیس 2 و 3 را به ترتیب به رویدادهای `index-1` و `index-2` متصل می‌کنیم.

یکی از روش‌هایی که امروزه مورد استقبال برنامه نویسان اندروید و جاوا قرار گرفته‌است، استفاده از یک سیستم [DSL](#) به نام [Gradle](#) (+) است. ابتدا در سیستم‌های [Apache Ant](#) (+) و [Maven](#) (+) مورد استفاده قرار می‌گرفت، ولی با جمع کردن نقاط ضعف آن دو سیستم، و رفع عیوب آن‌ها و افزودن مزیت‌های جدید، [Gradle](#) ایجاد شد. یکی از استفاده‌هایی که به شدت مورد استفاده‌ی برنامه نویسان اندروید قرار می‌گیرد، استفاده از یک سیستم توزیع برای کلاس‌های اندرویدی است. اگر امروزه به خیلی از سورس‌های قرار گرفته بر روی گیت هاب، نگاه کنید، به غیر از روش افزودن آن سورس به پروژه به عنوان ماژول، روش دیگری نیز وجود دارد که آن، استفاده از سیستم توزیع [Gradle](#) است. استفاده از این روش محبوبیت زیادی دارد و بسیار هم راحت‌تر است از افزودن یک سورس به پروژه.

برای افزودن یک ماژول به پروژه از طریق گریدل، به صورت زیر اقدام می‌کنیم:

هر ماژول شامل یک فایل به نام `build.gradle` است که تنظیمات سطح آن ماژول را به عهده دارد و پروژه نیز یک `build.gradle` دارد که تنظیمات آن در سطح پروژه صورت می‌گیرد. برای افزودن سورس در سطح یک ماژول لازم است که تعدادی خط کد را که معرف و آدرس آن سورس را دارد، به فایل `build.gradle` اضافه کنیم. به عنوان مثال برای سورس [Active Android ORM](#) را که یک [ORM](#) [عالی در سطح اندروید](#) به شمار می‌آید، به ماژولمان اضافه می‌کنیم:

```
dependencies {
    compile 'com.michaelpardo:activeandroid:3.1.0-SNAPSHOT'
}
```

ولی یک سوال پدید می‌آید که این خط کوتاه که تنها شامل نام و نسخه‌ی کتابخانه است، چگونه می‌تواند آدرس آن سورس را به دست بیاورد؟ در این نوشتار قصد داریم این مساله را بررسی کرده و بدانیم که این سورس‌ها چگونه به این سیستم توزیع اضافه شده‌اند.

سوال : اندروید استودیو، کتابخانه‌های اندرویدی را از کجا دانلود می‌کند؟

[Apache Maven](#) یک سیستم آزاد است که برای توزیع کتابخانه‌ها مورد استفاده قرار می‌گیرد. سرورهای این سیستم شامل یک مخزن `maven` هستند که کتابخانه‌ها در آن قرار می‌گیرند و شناسه‌ی دسترسی به آن کتابخانه از طریق همان شناسه‌ای است که شما در `build.gradle` تعریف می‌کنید. به طور عادی دو سرور استاندارد برای اینکار وجود دارند که یکی از آن‌ها `jcenter` و دیگری `mavenCentral` است. البته سرورهای دیگری نیز وجود دارند، یا اینکه حتی خودتان هم می‌توانید میزبانی را به عهده بگیرید و یا بعضی از شرکت‌ها برای خود مخزنی جداگانه دارند.

JCenter

این سرور که یک مخزن `maven` است، توسط [Bintray](#) میزبانی می‌شود که می‌توانید آن را در [این آدرس](#) ببابید. برای اینکه شناسه‌های `gradle` مربوط به این سرور در اندروید استودیو دانلود شود، نیاز است خط زیر را به `build.gradle` سطح پروژه اضافه کنید:

```
allprojects {
    repositories {
        jcenter()
    }
}
```

MavenCentral

این مخزن توسط [Sonatype.org](#) میزبانی می‌شود که کل مخزن آن را می‌توانید در [این آدرس](#) ببابید. برای دسترسی به مخازن این

سرور نیاز است خطوط زیر را به gradle سطح پروژه اضافه کنید:

```
allprojects {
    repositories {
        mavenCentral()
    }
}
```

موقعی که شما شناسه‌ی گریدل را اضافه می‌کنید، حتما باید دقت کنید مخزن آن کجا قرار گرفته است؟ آیا در یکی از آدرس‌های بالاست یا حتی می‌تواند در هر دو آدرس بالا قرار گرفته باشد؛ یا مخزنی غیر از مخازن بالاست.

به عنوان مثال Twitter's Fabric.io خودش کتابخانه‌ی خودش را میزبانی می‌کند و مخزن آن در این آدرس قرار گرفته است و برای افزودن این کتابخانه به پروژه نیاز است مسیر زیر طی شود:

```
//project build.gradle
repositories {
    maven { url 'https://maven.fabric.io/public' }
}

//module build.gradle
dependencies {
    compile 'com.crashlytics.sdk.android:crashlytics:2.2.4@aar'
}
```

سوال : کدامیک از مخازن بالا را انتخاب کنیم؟

اینکه کدامیک را انتخاب کنیم مساله‌ای است که به خودتان مرتبط است و هر دو برای یک هدف ایجاد شده‌اند و آن میزبانی کتابخانه‌های جاوا و اندرویدی است. بسیاری از توسعه دهندگان از هر دو استفاده می‌کنند؛ ولی بعضی‌ها هم تنها یکی از آن دو را بر می‌گزینند. اندروید استودیو در نسخه‌های اولیه‌ی خود به طور پیش فرض mavenCentral را صدا می‌زد و به طور پیش فرض در build.gradle پروژه، آن را معرفی کرده بود. ولی مساله‌ای که در این بین بود، این بود که این مخزن چندان به مذاق توسعه دهندگان خوش نمی‌آمد و کمی کار با آن دشوار و سخت بود. لذا تیم اندروید بنا به دلایلی مثل آن و موارد امنیتی و ... و اینکه توسعه دهندگان بیرونی بیشتر از jcenter استفاده می‌کردند، آن هم به سمت jcenter رفتند که در ورژن‌های فعلی اندروید استودیو می‌توانید ببینید که کتابخانه‌ی پیش فرض تغییر یافته است و jcenter() به جای mavenCentral() صدا زده می‌شود.

دلایل مهاجرت از mavenCentral به jcenter:

سیستم jcenter از طریق یک CDN عمل می‌کند که در این صورت می‌تواند تجربه‌ی خوبی از سرعت بهتر را برای توسعه دهندگان به همراه داشته باشد. کتابخانه‌های jcenter بسیار بیشتر از mavenCentral هستند؛ تا جایی که می‌توان گفت اکثر کتابخانه‌هایی که روی mavenCentral پیدا می‌شوند، روی jcenter هم هست و jcenter بزرگترین مخزن به شمار می‌آید. آپلود کتابخانه بر روی jcenter بسیار راحت‌تر است و نیاز به کار پیچیده‌ای ندارد. در این نوشتار سعی داریم ابتدا کتابخانه‌ی AndroidBreadCrumb را بر روی jcenter آپلود کنیم و سپس با استفاده از روش آسانتری آن را به سمت mavenCentral بفرستیم.

بررسی قسمت‌های یک شناسه Gradle

هر شناسه شامل سه قسمت می‌شود:

```
GROUP_ID:ARTIFACT_ID:VERSION
```

قسمت اول نام پکیج است که به آن Group_ID می‌گویند و می‌تواند خانواده‌ای از یک پکیج را نیز مشخص کند. سپس قسمت Artifact، نامی است که بر روی پروژه‌ی خود گذاشته‌اید و سپس ورژن است که در قالب x.y.z معرفی می‌شود و در صورت

اختیار می‌توانید عباراتی مثل beta- و snapshot- را هم داشته باشید.
کتابخانه‌های زیر، از یک خانواده هستند که به راحتی می‌توانید آن‌ها را از هم تشخیص دهید:

```
dependencies {  
    compile 'com.squareup:otto:1.3.7'  
    compile 'com.squareup.picasso:picasso:2.5.2'  
    compile 'com.squareup.okhttp:okhttp:2.4.0'  
    compile 'com.squareup.retrofit:retrofit:1.9.0'  
}
```

پس نحوه‌ی دریافت کتابخانه‌ها به این شکل است که ابتدا اندروید استودیو به ترتیب مخازن معرفی شده‌ی در سطح پروژه را چک می‌کند و از طریق شناسه‌ی آن‌ها بررسی می‌کند که آیا این کتابخانه اینجا موجود است، یا خیر و اگر موجود بود آن را دانلود می‌کند.

شناخت فایل‌های AAR

همانطور که می‌دانید فرمت فایل‌های بایت کدی جاوا JAR می‌باشد که هم توسط جاوا و هم اندروید پشتیبانی می‌شود. ولی در صورتیکه کلاس شما یک پروژه‌ی اندرویدی باشد، نمی‌توانید آن را در قالب یک فایل JAR منتشر کنید. چرا که که کلاس اندرویدی می‌تواند شامل فایل مانیفست، منابع و ... باشد که در فایل JAR جایی برای آن‌ها مهیا نشده است. به همین علت فایل‌های نوع AAR برای اینکار مهیا شده‌اند که این فایل در واقع یک فایل زیپ است که محتویات مورد نظر داخل آن قرار گرفته است و یکی از آن فایل Classes.jar برای کدهاست و مابقی آن به شرح زیر است:

```
- /AndroidManifest.xml (الزامی)  
- /classes.jar (الزامی)  
- /res/ (الزامی)  
- /R.txt (الزامی)  
- /assets/ (اختیاری)  
- /libs/*.jar (اختیاری)  
- /jni/<abi>/*.so (اختیاری)  
- /proguard.txt (اختیاری)  
- /lint.jar (اختیاری)
```

در مقاله‌ی بعدی کار را با jcenter آغاز می‌کنیم.

نظرات خوانندگان

نویسنده: مرتضی حاتمی
تاریخ: ۲۲:۵۸ ۱۳۹۴/۰۸/۰۹

سلام

اگر میشه کمی در مورد نسخه beta- و snapshot- بیشتر توضیح دهید.

نویسنده: علی یگانه مقدم
تاریخ: ۲:۱۹ ۱۳۹۴/۰۸/۱۰

snapshot بدین معنی است که این نسخه در حال حاضر در حال توسعه است و به نسخه حقیقی آن نرسیده است و ممکن است نسخه ای که دیروز از گریدل گرفتید با نسخه ای که امروز از گریدل دانلود می‌کنید متفاوت باشد و در هر مرحله زمانی به روزرسانی شود و بیشتر جهت بررسی آن نسخه ارائه گردیده است و این احتمال می‌رود هنوز به مرحله پایداری نرسیده است. در حالی که نسخه بتا یک نسخه تست شده در محیط توسعه است و یک سری امکاناتی را ارائه داده است و جهت تست و رفع عیب در محیط عملیاتی ارائه گشته است تا نسخه بتای بعد یا نهایی و ... ارائه گردد