Android 天气预报简单记录（一）

主要内容

- 为什么选择ConstraintLayout
- 用dom4j来解析XML
- 利用Map数据结构来实现图片的切换
- 更新动画处理

为什么选择ConstraintLayout

在几年前，布局还是各种嵌套的天下，但无论是 LinearLayout 还是 RelativeLayout 这种传统布局带来的特点是随着布局的复杂度的上升，嵌套的层次就越来越多,会使用很多不必要的 ViewGroup 去进行一个分割，而随之伴随的是性能的下降。

而16年开始发布的 ConstraintLayout 就是为了改善这个而诞生的。

为什么性能会有所提升？简单的说明一下：

> Android 的绘图主要分为三个阶段，测量，布局，绘制
>
> - 在使用 xml 进行布局的实现已经可以发现，其实整个文件就是一个视图树结构，所以在测量的时候,需要自顶之下遍历树结构，去确定每一个 ViewGroup 和 View 的大小
> - 布局时，需要再次遍历树结构，根据测量阶段得到的大小来确定自己以及子对象所在的位置
> - 最后的绘制还是需要执行一个遍历，对视图树的每一个对象，包括 View 还是 ViewGroup，都会创建一个 Canvas 对象，最终传递给 CPU 一个绘制的列表，包含对象的大小和位置。
>
> 所以，在每一个阶段都需要执行一次自顶向下的遍历操作，所以在整个视图层次结构中嵌套的越多，时间和计算的功耗也就越多。因此最理想的Android布局应该体现扁平的层次结构，来得到响应灵敏且简洁的界面

使用 ConstraintLayout 来实现相同的布局，因为引入了 View 与 View，与 ViewGroup 之间的相互约束（constraint）来进行布局的描述，不仅使得对于布局的实现思考变得简单，表现力变得更强，更使得效率大幅度提升。

这里不再做更加详细的介绍如何使用 ConstraintLayout，直接贴上郭霖大佬的一篇博文

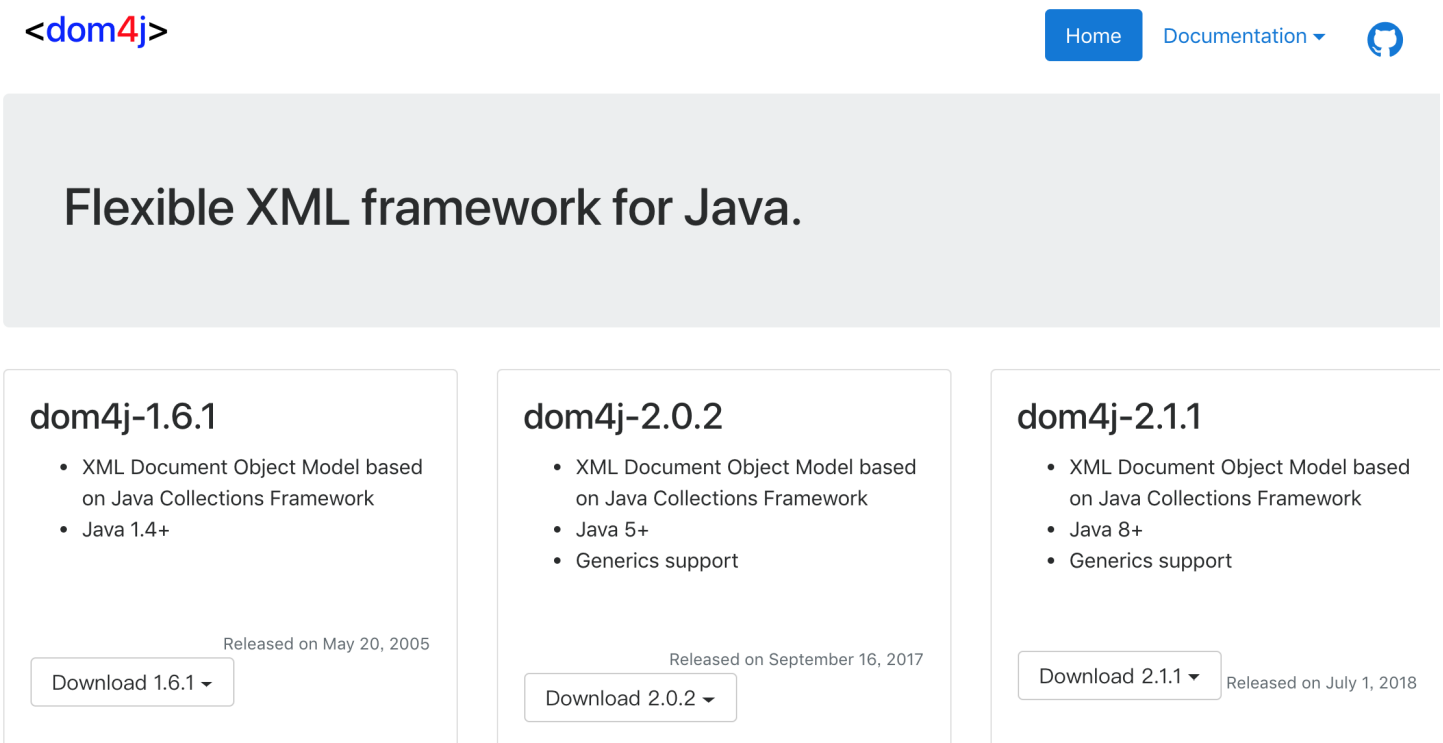
用dom4j来解析XML

XML 是一种通用的数据交换格式,它的平台无关性,语言无关性,系统无关性,给数据集成与交互带来了极大的方便。但现在储存数据的格式种类有很多，其实我更喜欢 JSON 这种数据格式。但是课程提供的 AP 接口返回的就是 XML 数据，所以了解一下如何简洁明了的解析 XML 数据。

XML的解析手法有很多,这里我采用的是 DOM4J，一个开源的优秀 java XML 框架，而且性能优异，使用简单。

下载

首先，我们需要到官网下载对应的 jar，这里我选新不选旧，直接下载的是 dom4j-2.1.1。下载好了之后，在IDE里面直接 Command+; 打开 Project Structure，选择 Modules 的 app 界面最右边 Dependencies 页面+添加即可，如图所示



使用

接下里就是如何使用 dom4j 来解析 XML 数据，首先，你得知道返回的 XML 格式的标签，这里随便运行一下你的程序，在 Logcat 里面复制上 log 出来的 XML 数据，然后去美化网站（我这里随便百度的一个），粘贴观察发现，

```
<?xml version="1.0" encoding="utf-8"?>
<resp>
    <city>海淀</city>
    <updatetime>14:21</updatetime>
    <wendu>11</wendu>
    <fengli>
        <![CDATA[2级]]>
    </fengli>
    <shidu>26%</shidu>
    <fengxiang>东南风</fengxiang>
    <sunrise_1>06:49</sunrise_1>
    <sunset_1>17:07</sunset_1>
    <sunrise_2></sunrise_2>
    <sunset_2></sunset_2>
    <yesterday>
        <date_1>5日星期一</date_1>
        <high_1>高温 11℃</high_1>
        <low_1>低温 1℃</low_1>
        <day_1>
            <type_1>多云</type_1>
            <fx_1>南风</fx_1>
            <fl_1>
                <![CDATA[<3级]]>
            </fl_1>
        </day_1>
        <night_1>
            <type_1>多云</type_1>
            <fx_1>北风</fx_1>
            <fl_1>
                <![CDATA[<3级]]>
            </fl_1>
        </night_1>
    </yesterday>
    <forecast>
        <weather>
            <date>6日星期二</date>
            <high>高温 11℃</high>
            <low>低温 0℃</low>
            <day>
                <type>多云</type>
                <fengxiang>东北风</fengxiang>
                <fengli>
                    <![CDATA[<3级]]>
                </fengli>
            </day>
            <night>
                <type>多云</type>
                <fengxiang>西北风</fengxiang>
                <fengli>
                    <![CDATA[<3级]]>
                </fengli>
            </night>
        </weather>
        <weather>
```

```
        <date>7日星期三</date>
        <high>高温 13℃</high>
        <low>低温 3℃</low>
        <day>
            <type>多云</type>
            <fengxiang>南风</fengxiang>
            <fengli>
                <![CDATA[<3级]]>
            </fengli>
        </day>
        <night>
            <type>多云</type>
            <fengxiang>西南风</fengxiang>
            <fengli>
                <![CDATA[<3级]]>
            </fengli>
        </night>
    </weather>
    <weather>
        <date>8日星期四</date>
        <high>高温 14℃</high>
        <low>低温 1℃</low>
        <day>
            <type>多云</type>
            <fengxiang>南风</fengxiang>
            <fengli>
                <![CDATA[<3级]]>
            </fengli>
        </day>
        <night>
            <type>晴</type>
            <fengxiang>西北风</fengxiang>
            <fengli>
                <![CDATA[3-4级]]>
            </fengli>
        </night>
    </weather>
    <weather>
        <date>9日星期五</date>
        <high>高温 14℃</high>
        <low>低温 0℃</low>
        <day>
            <type>晴</type>
            <fengxiang>西北风</fengxiang>
            <fengli>
                <![CDATA[3-4级]]>
            </fengli>
        </day>
        <night>
            <type>晴</type>
            <fengxiang>北风</fengxiang>
            <fengli>
                <![CDATA[<3级]]>
            </fengli>
        </night>
```

```
    </weather>
    <weather>
        <date>10日星期六</date>
        <high>高温 13℃</high>
        <low>低温 0℃</low>
        <day>
            <type>晴</type>
            <fengxiang>北风</fengxiang>
            <fengli>
                <![CDATA[<3级]]>
            </fengli>
        </day>
        <night>
            <type>多云</type>
            <fengxiang>西北风</fengxiang>
            <fengli>
                <![CDATA[<3级]]>
            </fengli>
        </night>
    </weather>
</forecast>
<zhishus>
    <zhishu>
        <name>晨练指数</name>
        <value>不宜</value>
        <detail>早晨天气寒冷，请尽量避免户外晨练，若坚持室外晨练请注意保暖防冻，建议年老体弱人群
    </zhishu>
    <zhishu>
        <name>舒适度</name>
        <value>较舒适</value>
        <detail>白天天气晴好，早晚会感觉偏凉，午后舒适、宜人。</detail>
    </zhishu>
    <zhishu>
        <name>穿衣指数</name>
        <value>较冷</value>
        <detail>建议着厚外套加毛衣等服装。年老体弱者宜着大衣、呢外套加羊毛衫。</detail>
    </zhishu>
    <zhishu>
        <name>感冒指数</name>
        <value>易发</value>
        <detail>昼夜温差很大，易发生感冒，请注意适当增减衣服，加强自我防护避免感冒。</detail>
    </zhishu>
    <zhishu>
        <name>晾晒指数</name>
        <value>基本适宜</value>
        <detail>天气不错，午后温暖的阳光仍能满足你驱潮消霉杀菌的晾晒需求。</detail>
    </zhishu>
    <zhishu>
        <name>旅游指数</name>
        <value>适宜</value>
        <detail>天气较好，但丝毫不会影响您出行的心情。温度适宜又有微风相伴，适宜旅游。</detail>
    </zhishu>
    <zhishu>
        <name>紫外线强度</name>
        <value>最弱</value>
```

```
            <detail>属弱紫外线辐射天气，无需特别防护。若长期在户外，建议涂擦SPF在8-12之间的防晒护肤
        </zhishu>
        <zhishu>
            <name>洗车指数</name>
            <value>较适宜</value>
            <detail>较适宜洗车，未来一天无雨，风力较小，擦洗一新的汽车至少能保持一天。</detail>
        </zhishu>
        <zhishu>
            <name>运动指数</name>
            <value>较不宜</value>
            <detail>天气较好，但考虑天气寒冷，推荐您进行室内运动，户外运动时请注意保暖并做好准备活动
        </zhishu>
        <zhishu>
            <name>约会指数</name>
            <value>较适宜</value>
            <detail>虽然有点风，但情侣们可以放心外出，不用担心天气来调皮捣乱而影响了兴致。</detail>
        </zhishu>
        <zhishu>
            <name>雨伞指数</name>
            <value>不带伞</value>
            <detail>天气较好，不会降水，因此您可放心出门，无须带雨伞。</detail>
        </zhishu>
    </zhishus>
</resp>
```

可以看到我们需要的信息在哪个节点里面一目了然，包括有一个 Forecast 下面的 weather 就是包括今天和未来几天的天气信息。同时可以发现这里并没有 pm2.5 和空气质量的数据返回，这两个只有在省会城市才会返回（比如北京，上海，杭州），所以在代码里要进行判断。

为了解释方便，我直接贴上代码，注释都在里面。

```
private TodayWeatherInfo dom4jPaeseXML(String xml) throws DocumentException {
        Boolean today = false; //因为今日天气和未来天气都在Forecast节点下
        List<ForecastWeather> dayOfWeekForecastWeather = new ArrayList<>();//ForecastWe
        TodayWeatherInfo info = new TodayWeatherInfo();//TodayWeatherInfo是返回天气数据的
        //传入xml文本，构建document对象
        Document document = DocumentHelper.parseText(xml);
        //获取文档document对象的根节点对象，这里就是resp
        Element root = document.getRootElement();
        Element node = root.element("city");//找到子节点city
        info.setCity(node.getText());//取得city节点的内容传入对应的setter函数
        info.setUpdateTime(root.element("updatetime").getText());//当然上面的步骤也可合二为
        info.setWendu(root.elementText("wendu"));//或者采用更简单的方法，直接取得对应节点的内容
        info.setFengli(root.elementText("fengli"));
        info.setShidu(root.elementText("shidu"));
        info.setFengxiang(root.elementText("fengxiang"));
        node = root.element("environment");//省会城市有该节点，而普通地区没有，这里我没有pm2.5
        if (node!=null){
            info.setPm25(node.elementText("pm25"));
            info.setQuality(node.elementText("quality"));
        }
        node = root.element("forecast");//获得forecast节点，遍历获取天气信息
        //迭代得到forecast节点下面所有的weather节点，进行下一步处理
        for (Iterator<Element> it = node.elementIterator("weather");it.hasNext();){
            Element weather = it.next();//得到weather节点
            if (!today){//如果是第一个weather节点，today为false，那么是今日天气
                today = true;//设置true，说明今日信息已经取到。下面同理
                info.setDate(weather.elementText("date"));
                info.setHigh(weather.elementText("high"));
                info.setLow(weather.elementText("low"));
                info.setType(weather.element("day").elementText("type"));
            }else {
                //未来几天天气情况
                ForecastWeather eachDayForecastWeather  = new ForecastWeather();
                eachDayForecastWeather.setDate(weather.elementText("date"));
                eachDayForecastWeather.setHigh(weather.elementText("high"));
                eachDayForecastWeather.setLow(weather.elementText("low"));
                eachDayForecastWeather.setType(weather.element("day").elementText("type
                eachDayForecastWeather.setFengxiang(weather.element("day").elementText(
                //取得未来每一天的天气信息，add到List中
                dayOfWeekForecastWeather.add(eachDayForecastWeather);
            }
        }
        info.setDayOfWeekForcastWeather(dayOfWeekForecastWeather);
        return info;//返回我们需要的所有XML信息
    }
```

更新

得到上面的 TodayWeatherInfo 的对象之后，它包含了我们需要的所有天气信息，以及一个List 包含未来4天的天气数据。因为未来天气的信息较少，只需要天气，气温，风向，日期，所有额外创建一个数据Bean类 ForecastWeather。

接下来就是异步机制，传递给消息队列，等待主线程给UI界面进行更新，再次提醒普通地区是没有pm2.5以及空气质量的信息返回，所有需要额外的处理。

具体的处理思路有两种，第一种就是不知道，直接在逻辑分支上更新为未知，不提供信息，第二种换一个天气API接口或者再次查询此城市所在省的省会城市的pm2.5和空气质量信息（如果觉得可以用省会城市的数据来笼统代表整个省份的空气情况的话）。这里我是采用第一种方法，直接设置为未知，如果没有返回相关信息的话。

利用Map数据结构来实现图片的切换

因为返回的天气状态有很多种, 相应的对应很多图片。同理，根据pm2.5的不同数值也对应不同的图片对应。那该怎么较为有效且简单的处理呢?

这里我选择采用Map，也就是映射这种数据结构进行处理。它的核心价值就是利用键值对和hash函数，所以查找是O(1)量级的，也就是查找数据很快。所以我们直接把返回的天气状态与相应的天气图片作为键值对直接存入Map中，在返回的XML数据中得到天气信息 x ，直接get(x)得到相对应的天气图片资源。

具体实现如下：

首先得到一个HashMap（Map接口的一个具体实现）来存储键：天气状态(String)和值：对应的图片资源(Integer)。因为图片资源，例如R.drawable.biz_plugin_weather_qing都是数值。

```
private Map<String,Integer> imgSrcForWeather = new HashMap<>();
```

然后在将所有的键值对写入：

```
private void initImgWithWeather(){
        imgSrcForWeather.put("晴", R.drawable.biz_plugin_weather_qing);
        imgSrcForWeather.put("多云", R.drawable.biz_plugin_weather_duoyun);
        imgSrcForWeather.put("暴雪", R.drawable.biz_plugin_weather_baoxue);
        imgSrcForWeather.put("暴雨", R.drawable.biz_plugin_weather_baoyu);
        imgSrcForWeather.put("大暴雨", R.drawable.biz_plugin_weather_dabaoyu);
        imgSrcForWeather.put("大雪", R.drawable.biz_plugin_weather_daxue);
        imgSrcForWeather.put("大雨", R.drawable.biz_plugin_weather_dayu);
        imgSrcForWeather.put("雷阵雨", R.drawable.biz_plugin_weather_leizhenyu);
        imgSrcForWeather.put("雷阵雨与冰雹", R.drawable.biz_plugin_weather_leizhenyubingb
        imgSrcForWeather.put("沙尘暴", R.drawable.biz_plugin_weather_shachenbao);
        imgSrcForWeather.put("特大暴雨", R.drawable.biz_plugin_weather_tedabaoyu);
        imgSrcForWeather.put("雾", R.drawable.biz_plugin_weather_wu);
        imgSrcForWeather.put("小雪", R.drawable.biz_plugin_weather_xiaoxue);
        imgSrcForWeather.put("小雨", R.drawable.biz_plugin_weather_xiaoyu);
        imgSrcForWeather.put("阴", R.drawable.biz_plugin_weather_yin);
        imgSrcForWeather.put("雨夹雪", R.drawable.biz_plugin_weather_yujiaxue);
        imgSrcForWeather.put("阵雪", R.drawable.biz_plugin_weather_zhenxue);
        imgSrcForWeather.put("阵雨", R.drawable.biz_plugin_weather_zhenyu);
        imgSrcForWeather.put("中雪", R.drawable.biz_plugin_weather_zhongxue);
        imgSrcForWeather.put("中雨", R.drawable.biz_plugin_weather_zhongyu);
}
```

这样，我们就已经建立了天气状态到图片的一个映射关系了，下面就是如何利用它进行图片的更新了：

```
// weatherImg是天气图片的对象实例的引用
// setImageResource()方法需要一个图片资源的参数
// info.getType()返回天气状态(String)
// imgSrcForWeather.get()传入一个键：天气状态，返回值：天气图片资源
weatherImg.setImageResource(imgSrcForWeather.get(info.getType()));
```

在更新天气的方法里面，只需要这一行代码即可根据返回的天气自动完成对图片的切换。

同理，也可以完成对pm2.5的数值更新头像图片的功能：

```
private Map<Integer,Integer> imgSrcForPm = new HashMap<>();
// 键0~5 代表 (返回的pm2.5数值/50)的值
// 0就代表返回的是0~49的数值 基本符合0~50的设想
private void initImgWithPm(){
        imgSrcForPm.put(0, R.drawable.biz_plugin_weather_0_50);
        imgSrcForPm.put(1, R.drawable.biz_plugin_weather_51_100);
        imgSrcForPm.put(2, R.drawable.biz_plugin_weather_101_150);
        imgSrcForPm.put(3, R.drawable.biz_plugin_weather_151_200);
        imgSrcForPm.put(4, R.drawable.biz_plugin_weather_201_300);
        imgSrcForPm.put(5, R.drawable.biz_plugin_weather_201_300);
}
```

在更新天气的方法里面，需要对是否返回pm2.5信息进行判断：

```
// 如果返回的pm2.5不为null，说明有相关信息，进行更新
if (info.getPm25() != null){
            pmImg.setImageResource(imgSrcForPm.get(Integer.parseInt(info.getPm25())/50)
            pmDataTv.setText(info.getPm25());
            pmQualityTv.setText(info.getQuality());
        }else {
            //若为null，直接将数据设置为 未知，图片默认为0_50
            pmImg.setImageResource(R.drawable.biz_plugin_weather_0_50);
            pmDataTv.setText("未知");
            pmQualityTv.setText("未知");
}
```

总结

其实这已经是一个很好的处理方法，逻辑清晰，且效率高效。其实之前学长给出的拼音库和反射技术，本质上也是相同的。文字转拼音，本质上也是一个文字和拼音相对应的映射关系，包括UTF-8之类的编码，也是相同的道理，背后都是有相对应的映射关系支撑。

更新动画处理

目的是在刷新过程中的对刷新图片做一个动态的展示，一是为了给用户直观的说明刷新操作正在进行，二是在动画过程中禁用 imageView 的一个点击事件，防止出现多个线程操作出错且浪费计算资源。

这里没有采用上课所使用的进度条控件替代 imageView 的方式，直接对其加上旋转的动画效果。因为只需要进行旋转，所以使用Animation(补间动画支持平移，缩放，旋转，透明度)对 View 进行操作即可。

思路十分简单

1. 设置动画文件，说明动画的效果
2. 给刷新的imageView进行绑定，并注销点击事件
3. 更新界面最后重新恢复点击事件

在res/anim/anim_round_rotate.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <rotate
        android:fromDegrees="0"
        android:toDegrees="359"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="1000"
        android:repeatCount="-1" />
        <!--设置不断旋转  设置角度，旋转的中心-->
</set>
```

然后在 MainActivity 里面写上一个绑定动画的函数：

```
//传入上下文context，和待绑定控件update
private void updateAnimation(Context context, ImageView update){
        Animation circle_anim = AnimationUtils.loadAnimation(context, R.anim.anim_rounc
        LinearInterpolator interpolator = new LinearInterpolator();  //设置匀速旋转，在xm
        circle_anim.setInterpolator(interpolator);
        if (circle_anim != null) {
            update.startAnimation(circle_anim);  //开始动画
        }
    }
```

动画何时执行呢？在 update 对象的点击事件监听函数里面进行描述：

```
myUpdate.setClickable(false);//注销点击事件
updateAnimation(MainActivity.this, myUpdate);//绑定动画效果
requestWeatherByCode(cityCode);//发送请求进行更新操作
```

最后需要在 updateWeather 函数，就是更新页面天气信息的函数末尾，恢复刷新图片的点击效果：

```
myUpdate.clearAnimation();//取消动画
myUpdate.setClickable(true);//恢复点击事件
```

到这里，要求基本完成。

> 点击更新图片-->触发点击事件-->注销点击事件监听，开启动画效果-->
> 发起更新方法（创建一个线程获取XML数据）-->
> 解析XML数据得到相关数据放入消息队列，异步通知-->
> 主线程得到数据，更新UI，取消动画效果，恢复监听

总结

时间不够，先写一点，接下来会继续更新搜索框搜索（支持中文，拼音，首字母，并按序排列），RecyclerView来显示可选择城市以及未来几天天气信息等等。也可以直接去博客浏览。