Volley接入HTTPDNS SDK

Volley接入HTTPDNS SDK

以下代码片段摘自SDK使用Sample（HttpDnsSample目录），完整代码请参考使用Sample

Volley的默认实现，在Android2.3以下，使用HttpClient作为底层网络库，在Android2.3及以上，使用HttpURLConnection作为底层网络库。因此，Volley接入HTTPDNS SDK（基于Volley的默认实现），主要是基于HttpClient及HttpURLConnection接入HTTPDNS SDK的方式进行兼容。HttpClient及HttpURLConnection如何接入HTTPDNS SDK请参考对应的接入文档（当前目录下）及使用Sample（HttpDnsSample目录），本文档示例代码仅演示如何将HttpClient及HttpURLConnection的兼容实现引入到Volley中

示例代码如下：

```
// VolleyHelper.kt
internal object VolleyHelper {

    lateinit var requestQueue: RequestQueue

    fun init(context: Context) {
        requestQueue = Volley.newRequestQueue(context.applicationContext,
CompatHttpStack())
    }
}

// CompatHttpStack.kt
internal class CompatHttpStack : BaseHttpStack() {

    @SuppressLint("ObsoleteSdkInt")
    private val realHttpStack =
        // NOTE: Sample的minSdk是14，这里是为了演示minSdk 9以下的情况
        // NOTE: HurlStackWrapper基于Volley源代码中对于HttpURLConnection的封
装，同时引入接入HTTPDNS SDK的兼容代码
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.GINGERBREAD)
HurlStackWrapper()
        // NOTE: HttpClientStackWrapper基于Volley源代码中对于HttpClient的封装，
同时引入接入HTTPDNS SDK的兼容代码
        else
AdaptedHttpStack(HttpClientStackWrapper(HttpClientHelper.httpClient))

    override fun executeRequest(request: Request<*>?, additionalHeaders:
MutableMap<String, String>?) =
            realHttpStack.executeRequest(request, additionalHeaders)
}
```

```
// NOTE: HurlStackWrapper代码拷贝自com.android.volley.toolbox.HurlStack, 并基
于HTTPDNS需求做少量修改
// HurlStackWrapper.java
final class HurlStackWrapper extends BaseHttpStack {

    // ...

    @Override
    public HttpResponse executeRequest(Request<?> request, Map<String,
String> additionalHeaders)
            throws IOException, AuthFailureError {

        // ...

        URL parsedUrl = new URL(url);
        // NOTE: BEGIN HTTPDNS-added
        String hostname = parsedUrl.getHost();
        if (DnsServiceWrapper.INSTANCE.getUseHttpDns()) {
            DnsLog.INSTANCE.d("HurlStackWrapper lookup for %s", hostname);
            InetAddress[] inetAddrs =
DnsServiceWrapper.INSTANCE.getAddrsByName(hostname);
            if (0 < inetAddrs.length) {
                parsedUrl = new URL(url.replace(hostname,
InetAddressExtKt.toUriFormat(inetAddrs[0])));
            }
        }
        // NOTE: END HTTPDNS-added
        HttpURLConnection connection = openConnection(parsedUrl, request);
        // NOTE: BEGIN HTTPDNS-added
        HttpUrlConnectionHelper.INSTANCE.compat4ChangeHost(connection,
hostname);
        // NOTE: END HTTPDNS-added

        // ...

    }

    // ...

}
```

```
// NOTE: HttpClientStackWrapper代码拷贝自
com.android.volley.toolbox.HttpClientStack, 并基于HTTPDNS需求做少量修改
// HttpClientStackWrapper.java
final class HttpClientStackWrapper implements HttpStack {

    // ...

    /** Creates the appropriate subclass of HttpUriRequest for passed in
request. */
    @SuppressWarnings("deprecation")
    /* protected */ static HttpUriRequest createHttpRequest(
            Request<?> request, Map<String, String> additionalHeaders)
throws AuthFailureError {
        // NOTE: BEGIN HTTPDNS-added
        Pair<String, String> url2HostPair =
HttpClientHelper.INSTANCE.url2HostPair(request.getUrl());
        // NOTE: END HTTPDNS-added
        switch (request.getMethod()) {
            // NOTE: BEGIN HTTPDNS-changed
            case Method.DEPRECATED_GET_OR_POST:
            {
                // This is the deprecated way that needs to be handled for
backwards
                // compatibility.
                // If the request's post body is null, then the assumption
is that the request
                // is
                // GET.  Otherwise, it is assumed that the request is a
POST.
                byte[] postBody = request.getPostBody();
                if (postBody != null) {
                    HttpPost postRequest = new
HttpPost(url2HostPair.getFirst());
                    postRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                    postRequest.addHeader(
                            HEADER_CONTENT_TYPE,
request.getPostBodyContentType());
                    HttpEntity entity;
                    entity = new ByteArrayEntity(postBody);
                    postRequest.setEntity(entity);
                    return postRequest;
                } else {
                    HttpGet getRequest = new
HttpGet(url2HostPair.getFirst());
                    getRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                    return getRequest;
                }
            }
            case Method.GET:
            {
                HttpGet getRequest = new HttpGet(url2HostPair.getFirst());
                getRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                return getRequest;
```

```java
            }
            case Method.DELETE:
            {
                HttpDelete deleteRequest = new
HttpDelete(url2HostPair.getFirst());
                deleteRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                return deleteRequest;
            }
            case Method.POST:
            {
                HttpPost postRequest = new
HttpPost(url2HostPair.getFirst());
                postRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                postRequest.addHeader(HEADER_CONTENT_TYPE,
request.getBodyContentType());
                setEntityIfNonEmptyBody(postRequest, request);
                return postRequest;
            }
            case Method.PUT:
            {
                HttpPut putRequest = new HttpPut(url2HostPair.getFirst());
                putRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                putRequest.addHeader(HEADER_CONTENT_TYPE,
request.getBodyContentType());
                setEntityIfNonEmptyBody(putRequest, request);
                return putRequest;
            }
            case Method.HEAD:
            {
                HttpHead headRequest = new
HttpHead(url2HostPair.getFirst());
                headRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                return headRequest;
            }
            case Method.OPTIONS:
            {
                HttpOptions optionsRequest = new
HttpOptions(url2HostPair.getFirst());
                optionsRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                return optionsRequest;
            }
            case Method.TRACE:
            {
                HttpTrace traceRequest = new
HttpTrace(url2HostPair.getFirst());
                traceRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                return traceRequest;
            }
            case Method.PATCH:
            {
                HttpPatch patchRequest = new
HttpPatch(url2HostPair.getFirst());
                patchRequest.addHeader(HTTP.TARGET_HOST,
url2HostPair.getSecond());
                patchRequest.addHeader(HEADER_CONTENT_TYPE,
request.getBodyContentType());
                setEntityIfNonEmptyBody(patchRequest, request);
                return patchRequest;
            }
            // NOTE: END HTTPDNS-changed
            default:
                throw new IllegalStateException("Unknown request method.");
        }
    }

    // ...

}
```