07

Аппаратные возможности

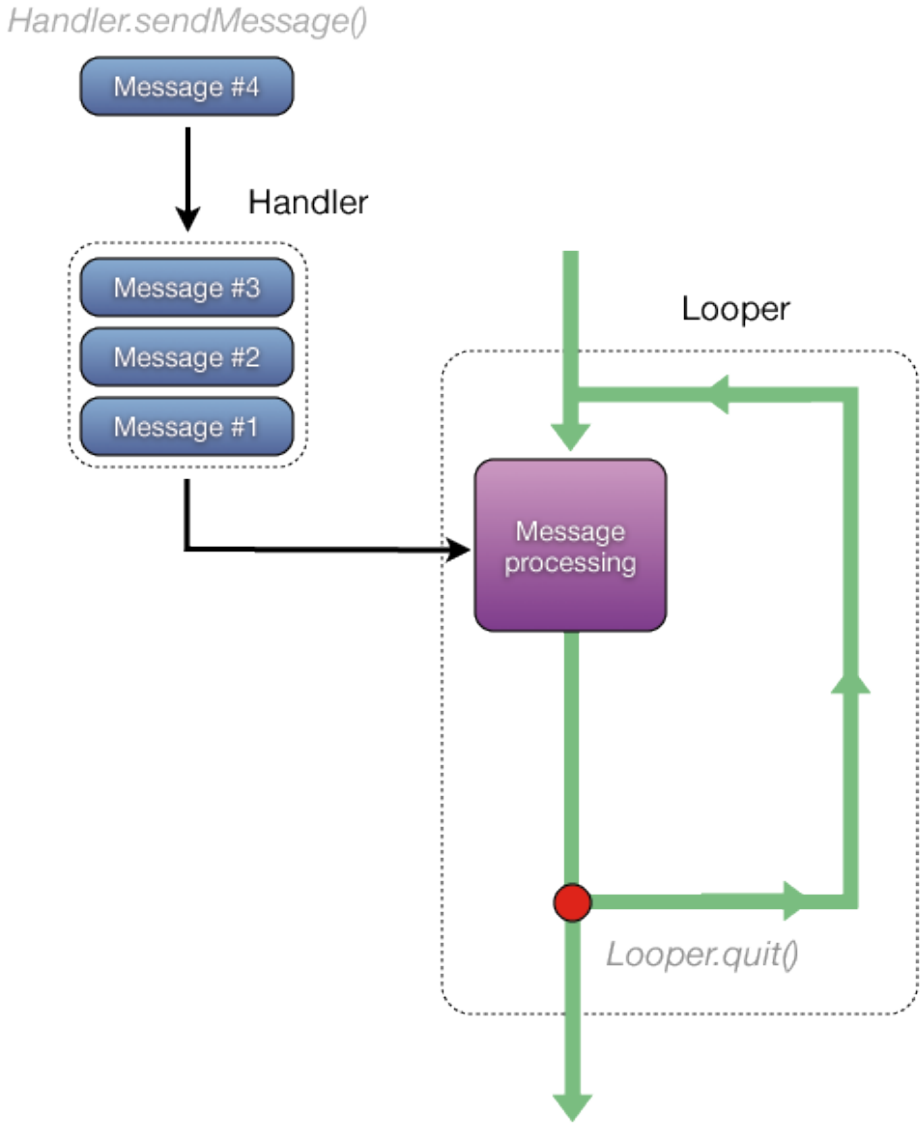
Nikolay Moskvin
moskvin@sibext.com

7 ноября 2013 г.

Содержание

Петля

- Это цикл обработки событий

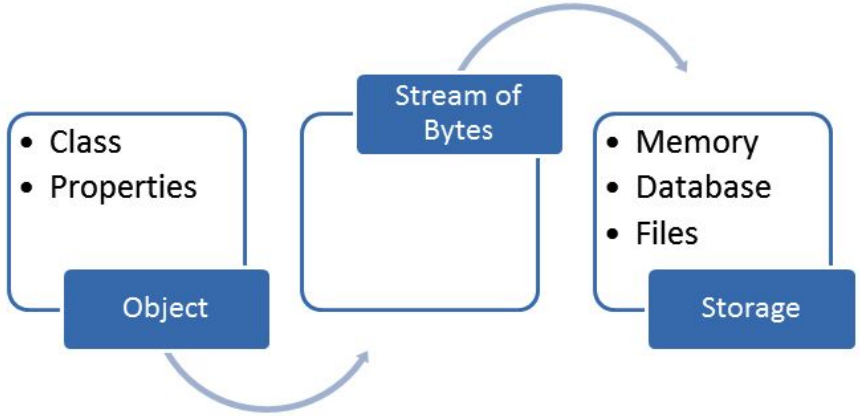


JSON

sibext company

- Простой формат для обмена данными
- Объект {} и Массив []
  `http://developer.android.com/reference/org/json`
- GSON – хорошая библиотека от google
  `http://code.google.com/p/google-gson/`
- Jackson – быстрая и легковесная
  `http://wiki.fasterxml.com/JacksonHome`

Сериализация объектов



Process of Serialization

- Процесс перевода объектов в другие структуры данных, которые можно передавать.
- Необходим, если мы хотим передавать свои данные через **Intent**
- Также позволяет формировать данные запроса на сервер или сервис

Как серализовать?

sibext company

- **Serializable** – стандартный подход
- **Parcelable** – android подход `http://developer.android.com/reference/android/os/Parcelable.html`
- сторонние библиотеки, обычно используют аннотации или xml для описания правил сериализации

Пример Pracelable

sibext company

```
public class DataParcelable implements Parcelable {
    private int data;
    private String name;
    @Override
    public int describeContents() {
        return 0;
    }
    @Override
    public void writeToParcel(Parcel out, int flags) {
        out.writeInt(data);
        out.writeString(name);
    }

    public static final
      Parcelable.Creator<DataParcelable> CREATOR
            = new Parcelable.Creator<DataParcelable>() {
        public DataParcelable createFromParcel(Parcel in) {
            return new DataParcelable(in);
        }

        public DataParcelable[] newArray(int size) {
            return new DataParcelable[size];
        }
    };

    private DataParcelable(Parcel in) {
        data = in.readInt();
        name = in.readString();
    }
}
```

Пример сериализации

sibext company

- Модель

```
public final class Response {
      @SerializedName("orders")
      @JsonProperty("orders")
      private Order[] orders;
}
public final class Order {
      @SerializedName("address")
      @JsonProperty("address")
      private Address address;
}
public final class Address {
      @SerializedName("city")
      @JsonProperty("city")
      private String city;
}
```

- Jackson

```
JsonFactory f = new JsonFactory();
ObjectMapper mapper = new ObjectMapper();
JsonParser parser = f.createJsonParser(reader);
parser.nextToken();
Response response = mapper.readValue(parser, Response.class);
```

- Данные

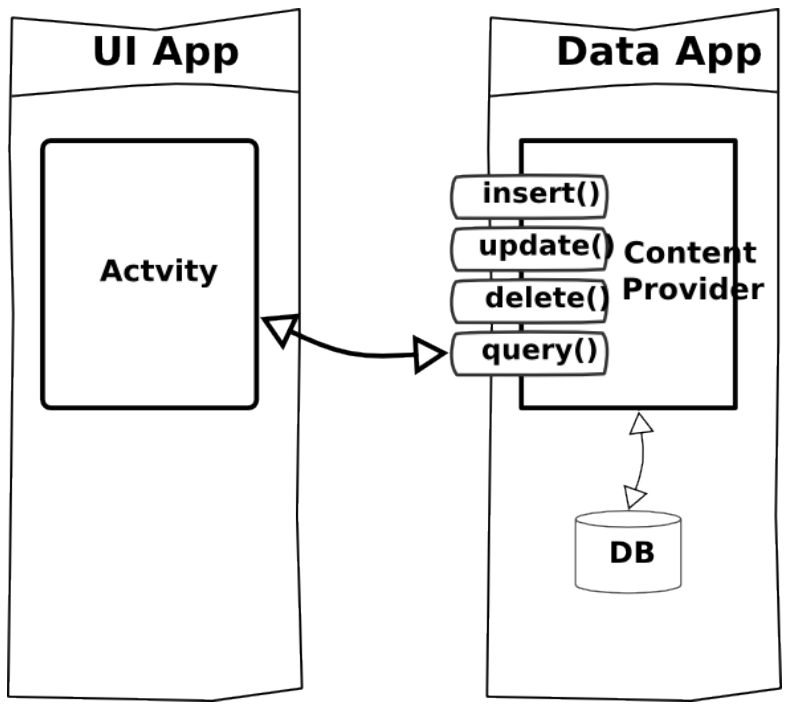
```
{"orders": [
  {"address": {"city": "Novosibirsk"}},
  {"address": {"city": "New York"}},
]}
```

Content Provider

Проблема:

- Приложения запущены в разных процессах (PC)
- Нужен способ передачи данных

Решение:



Методы

sibext company

- Основные методы:
  - **query** – получение данных
  - **insert** – добавление данных
  - **delete** – удаление данных
  - **getType** – получение типа данных
  - **openFile** – открытие файла
- Во всех методах присутствует **Uri** – идентификатор ресурса в провайдере

Uri

- Клиенты будут работать с провайдером формируя необходимые URI
- Общий вид:

```
content://com.<CompanyName>.provider.<ApplicationName>
```

- Связь authorities провайдера в манифесте:

```
android:authorities="com.<CompanyName>.provider.<ApplicationName>"
```

- Доступ к набору:

```
content://com.<CompanyName>.provider.<ApplicationName>/<Set>
```

- Доступ к элементу из набора:

```
content://com.<CompanyName>.provider.<ApplicationName>/<Set>/<Element>
```

Content Resolver

sibext company

- Клиенты должны получить объект класса **ContentResolver**
- ContentResolver позволяет выполнять операции: query, update, delete для указанных Uri
- ContentResolver обеспечивает связь между клиентом и провайдером, который связан с данным Uri
- `http://developer.android.com/guide/topics/providers/content-provider-basics.html`

Widgets

- Появилась в android 1.5
- В Android имеется некоторый набор стандартных: Analog Clock, Music, и другие
- Шаблон подложки для виджетов Android 4: `http://developer.android.com/shareables/app_widget_templates-v4.0.zip`

Widgets

sibext company

Чтобы создать виджет необходимо знать:

- AppWidgetProviderInfo – описание метаданные для виджета: layout, частата обновления, итд. (XML)
- AppWidgetProvider – описание базовых методов, которые позволяют наладить взаимодействия виджета с другими компонентами `http://developer.android.com/reference/android/appwidget/AppWidgetProvider.html`

В манифесте проекта должны быть определены:

- receiver с action.APPWIDGET_UPDATE
- тег meta-data с android.appwidget.provider

AppWidgetProvider

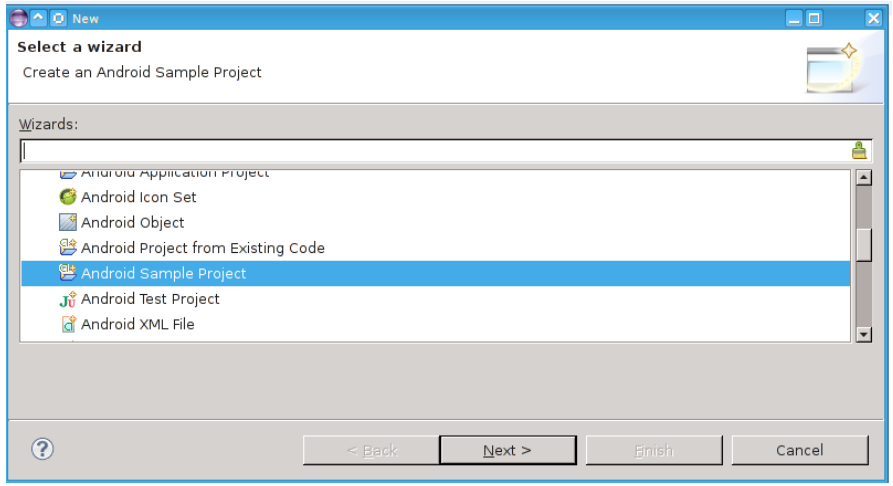
sibext company

Основные методы:

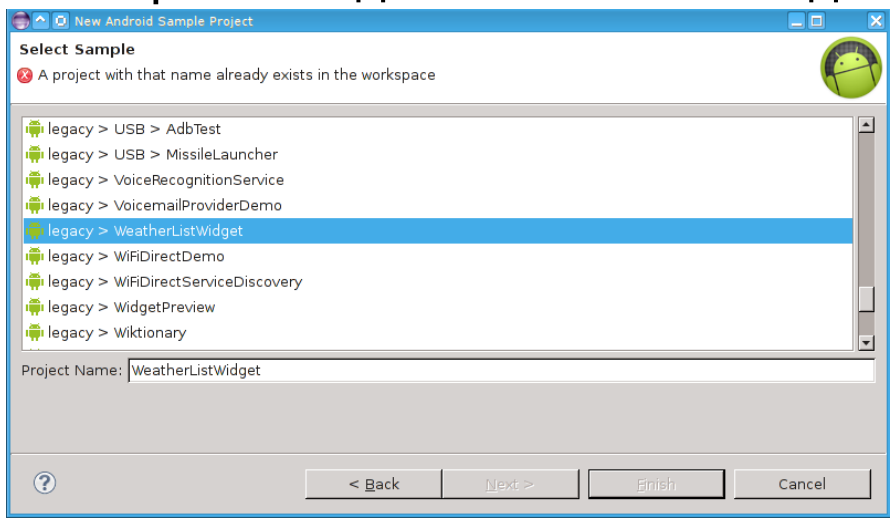
- **onUpdate()** – периодически вызывается с интервалом updatePeriodMillis после того, как пользователь поместил себе виджет
- **onAppWidgetOptionsChanged()** – будет вызван первый раз и после изменения размеров виджета
- **onDeleted()** – вызывается каждый раз, когда виджет был удален со стола
- **onEnabled()** – будет вызвано когда виджет создан впервые. При последующих добавлениях на рабочий стол вызван не будет
- **onDisabled()** – будет вызвано в момент, когда последний экземпляр виджета был удален
- **onReceive()** – не рекомендуется переопределять, вызывается для каждого broadcast сообщения

Создаем виджет из примеров

- Eclipse -> File -> New -> Other -> Android Sample Project



- Выбираем виджет списка погоды



Создаем виджет из примеров

- Загружаем на устройство и добавляем виджет на рабочий стол