Факултет по математика и информатика
Софийски университет „Св. Климент Охридски“

ItBooks

Курсов проект по Мобилни приложения

Изготвил:
Моника Шопова, ф.н. 61602, СИ

Contents

Идея

Този проект представлява реализацията на мобилно приложение, което е направено на базата на уеб сайт за сваляне на книги – http://it-ebooks.info/. За целта е използвано предоставеното от сайта API – http://it-ebooks-api.info/. Мобилното приложение(ItBooks) предоставя възможност за търсене на книги, разглеждане на детайлната информация относно книги, и предстои реализацията на модула за сваляне на книги. Най-разпространената мобилна платформа днес е платформата Android, затова е избрана за реализацията.

Реализация

Използвани са помощни библиотеки – android.Volley и google.Gson
Volley е библиотека, използвана за по-лесна комуникация в мрежи.
Gson е библиотека за превръщане на Java обекти в JSON и обратното, използвайки reflection. Може да се дефинират Java обекти, които имат същите имена като съответните им JSON ключове, да се подаде Gson като клас обект, и той сам ще напълни данните.

Следва описание на основните елементи от реализацията на приложението.

package AbstractApp

_____
Тук са нещата, които са необходими на Volley.

Class Web Fetcher:

Volley предоставя метод Volley.newRequestQueue, който създава RequestQueue(опашка със заявки), използвайки стойности по подразбиране, и стартира опашката.

requestQueue = Volley.newRequestQueue(context);

За да се изпрати заявка, първо тя се конструира, а после се добавя към RequestQueue чрез метода add(). След като е добавена, заявката се изпълнява и се доставя някакъв отговор.

Class ImageLoader – помощен клас, който се грижи за зареждането и кеширането на изображение от дадени URLs.
Имаме и вътрешен JsonRequest<T> клас. В него дефинираме и 2 абстрактни метода, които ще бъдат имплементирани в BookDetailsFetcher и BookGridItemsFetcher.

```
public abstract Response.Listener<T> createSuccessListener();
public abstract Response.ErrorListener createErrorListener();
```

Class BitmapLruCache:

Представлява имплементация по подразбиране за LRU Cache. Наследява LruCache<String, Bitmap> и имплементира ImageLoader.ImageCache.

Class GsonRequest:

GsonRequest е имплементация на Volley заявка, която използва Gson за парсване. Наследява Request<T>. parseNetworkResponse() е един от основните методи. Приема NetworkResponse като параметър, който съдържа response payload като byte[], HTTP статус код, и response headers. Връща Response<T>, която съдържа искания тип отговор или error, ако е имало проблем при парсването.

```
protected Response<T> parseNetworkResponse(NetworkResponse response) {
      try {
         String json = new String(response.data,
HttpHeaderParser.parseCharset(response.headers));
         return Response.success(gson.fromJson(json, clazz),
               HttpHeaderParser.parseCacheHeaders(response));
      } catch (UnsupportedEncodingException e) {
         return Response.error(new ParseError(e));
      } catch (JsonSyntaxException e) {
         return Response.error(new ParseError(e));
      }
   }
```

package Model

Class BookDetails:

Представлява детайлния изглед на дадена книга. Полетата са различните елементи, които ще бъдат визуализирани. Сериализирани са. Съдържа и съответните getters и setters.

Class BookGridItem:

Представлява изгледа на елемент в GridView. Съдържа id, заглавие и изображение на книгата, като те се подават още при инициализирането на съответния BookGridItem. Сериализирани са. Съдържа и съответните getters и setters.

package Activity

_____

Class BookDetailsActivity:

Логиката зад екрана, визуализиращ детайлите за дадена книга.
Имаме изображение и детайлите, изобразени чрез ExpandableList.
Зарежда изображението на книгата:

```
imageLoader
.withImageView(bookCoverView)
.withImageUrl(clickedBookCoverUrl)
.configure()
.load();
```

Като се кликне върху даден елемент се изпраща неговото id, за да се изпълни заявката.

```
 bookDetailsFetcher.withTargetView(bookDetailsExpListView)
.withUrl("http://it-ebooks-api.info/v1/book/" + clickedBookId)
.execute();
}
```

Class BooksGridActivity:

Логиката зад екрана, визуализиращ книгите, подредени в GridView.
Като кликнем на някой item, това ни препраща към BookDetailsActivity, като подава id-то на елемента и адреса на изображението.

```
 gridView.setOnItemClickListener(new GridView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int position, long
id) {
            final String clickedItemId =
bookGridItemsFetcher.bookList.get(position).getId();
            final String clickedItemBookCoverImageUrl =
bookGridItemsFetcher.bookList.get(position).getBookCoverImageUrl();
            Intent intent = new Intent(BooksGridActivity.this,
BookDetailsActivity.class);
```

```
            intent.putExtra(BOOK_COVER_ID, clickedItemId);
            intent.putExtra(BOOK_COVER_IMAGE_URL,
clickedItemBookCoverImageUrl);
            startActivity(intent);
        }
    });
```

Class BookSearchActivity:

Представлява началния екран. в onCreateOptionsMenu се създава searchManager, който всъщност представлява търсачката. Когато заявката е готова(като се натисне enter) се изпълнява метод doSearch - изпраща заявка към API-то, после изпращаме получената информация към BookGridActivity.

```
private void doSearch(String query) {
String bookSearchUrl = IT_EBOOKS_API_BASE_URL + query;
Intent intent = new Intent(BookSearchActivity.this, BooksGridActivity.class);
intent.putExtra(SEARCH_URL, bookSearchUrl);
startActivity(intent);
}
```

package Adapter

_____

Class BookCoverGridAdapter:

getView метода визуализира gridview елементите;
На адаптера се подава списък с елементи. Методът getView съдържа логиката за това как трябва да изглежда един елемент.

Class BookDetailsExpandableListAdapter:

Отговаря за ExpandableList-овете в BookDetailsActivity.
getGroupView() връща първоначалния изглед, а getChildView() връща изгледа, след като е избран някой от елементите.

package Net

_____

Class BookArrayJsonDeserializer:

Имплементира JsonDeserializer<BookGridItem[]>

Правим custom реализация, за да не match-ва целия JSON, а само нещата, които са нужни. В конкретния случай – за GridView са нужни само id, title и image на книгите.

Class BookGridItemsFetcher:

Визуализира получените JSON данни в определен вид (елементи в GridView)
Наследява WebFetcher.JsonRequest<BookGridItem[]>.
Първо се инициализира Gson обект, като му се подава списък от елементи (BookGridItem-s), и custom десериализацията.

```
private final Gson bookListCustomGson = new GsonBuilder()
.registerTypeAdapter(BookGridItem[].class, new BookArrayJsonDeserializer())
.create();
```

Имплементира createSuccessListener() и createErrorListener().
При onResponse() в createSuccessListener() преобразува масива с елементите, който получава в списък(за по-лесна работа по-натам), създава new BookCoverGridAdapter и му подава информацията.

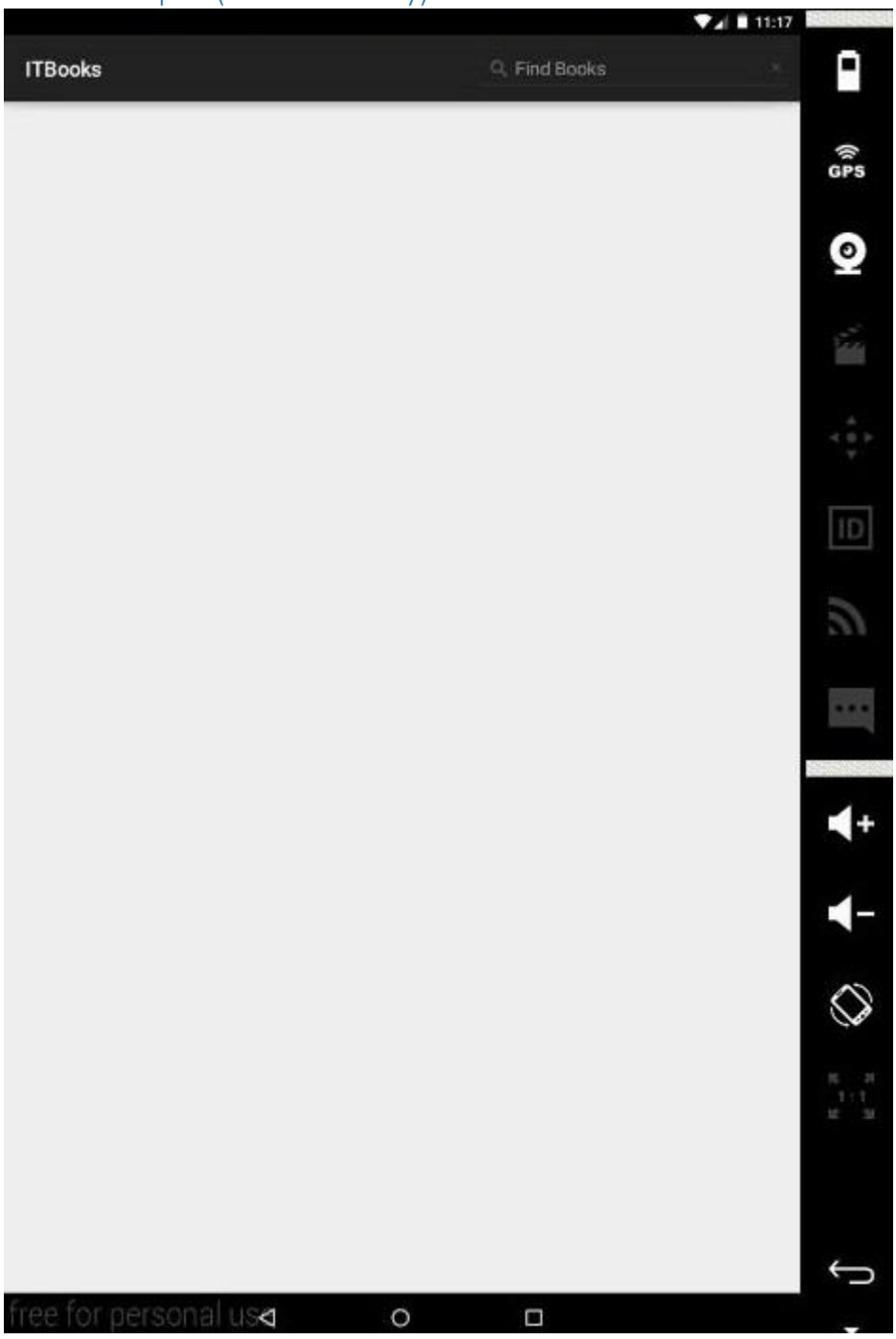
```
public Response.Listener<BookGridItem[]> createSuccessListener() {
      return new Response.Listener<BookGridItem[]>() {
         @Override
         public void onResponse(BookGridItem[] bookGridItems) {
            bookList = Arrays.asList(bookGridItems);
            gridAdapter = new BookCoverGridAdapter(bookList, context);
            gridView.setAdapter(gridAdapter);
         }
      };
   }

   public Response.ErrorListener createErrorListener() {
      return new Response.ErrorListener() {
         @Override
         public void onErrorResponse(VolleyError volleyError) {
            if(volleyError != null)
               Log.e("BooksGridActivity", volleyError.getMessage());
         }
      };
   }
```

Class BookDetailsFetcher:

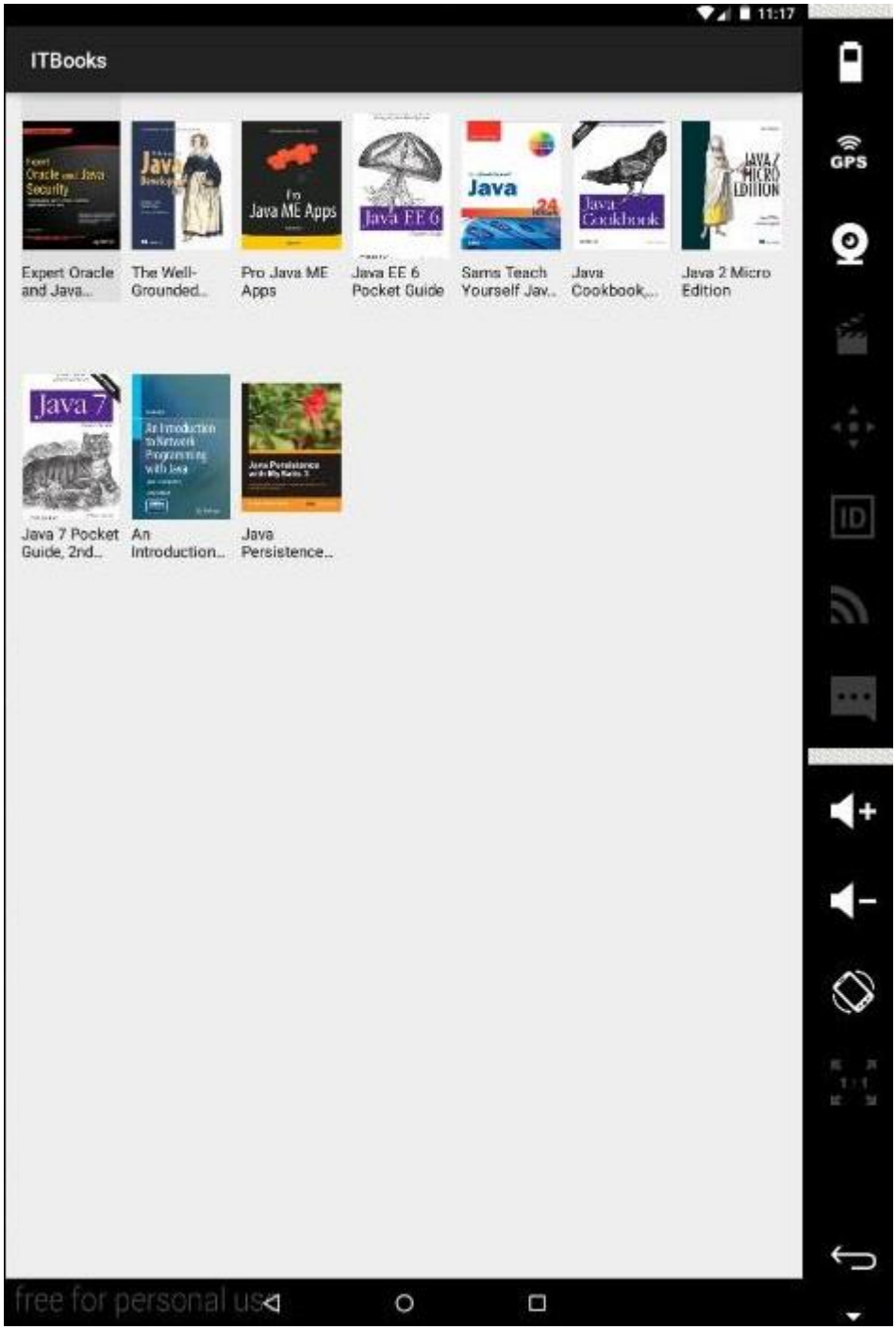
Наследява WebFetcher.JsonRequest<BookDetails> и имплементира методите createSuccessListener() и createErrorListener(). Методите populateBookDetailsData() и addBookDetailsPair() се грижат за напасването на заглавията (Title, Description …) и съответните получени данни за тях, и всичко се изпраща към BookDetailsExpandableListAdapter-а.

Screenshots

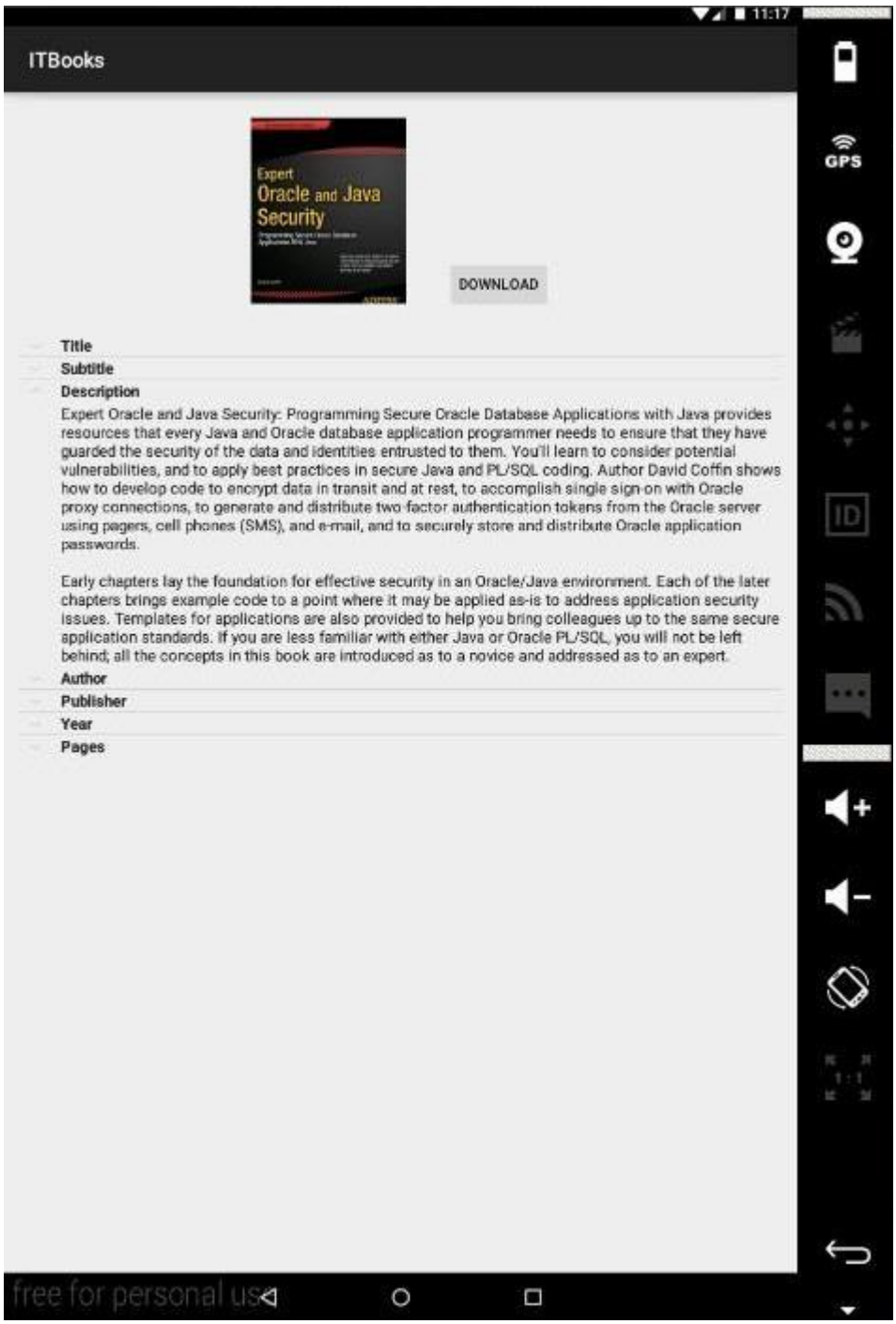
Начален екран (Search Activity)



GridView Activity



Details Activity


11:17
ITBooks
Expert
Oracle and Java
Security
DOWNLOAD
Title
Subtitle
Description
Expert Oracle and Java Security: Programming Secure Oracle Database Applications with Java provides resources that every Java and Oracle database application programmer needs to ensure that they have guarded the security of the data and identities entrusted to them. You'll learn to consider potential vulnerabilities, and to apply best practices in secure Java and PL/SQL coding. Author David Coffin shows how to develop code to encrypt data in transit and at rest, to accomplish single sign-on with Oracle proxy connections, to generate and distribute two-factor authentication tokens from the Oracle server using pagers, cell phones (SMS), and e-mail, and to securely store and distribute Oracle application passwords.
Early chapters lay the foundation for effective security in an Oracle/Java environment. Each of the later chapters brings example code to a point where it may be applied as-is to address application security issues. Templates for applications are also provided to help you bring colleagues up to the same secure application standards. If you are less familiar with either Java or Oracle PL/SQL, you will not be left behind; all the concepts in this book are introduced as to a novice and addressed as to an expert.
Author
Publisher
Year
Pages
GPS
free for personal use