9. АРГУМЕНТЫ ФРАГМЕНТОВ

В данном разделе в приложении BookDepository будет реализована совместная работа списка и детализации. Когда пользователь щелкает на элементе списка книг, на экране возникает новый экземпляр BookActivity, который является хостом для экземпляра BookFragment с подробной информацией о конкретном экземпляре Book (рисунок 9.1).

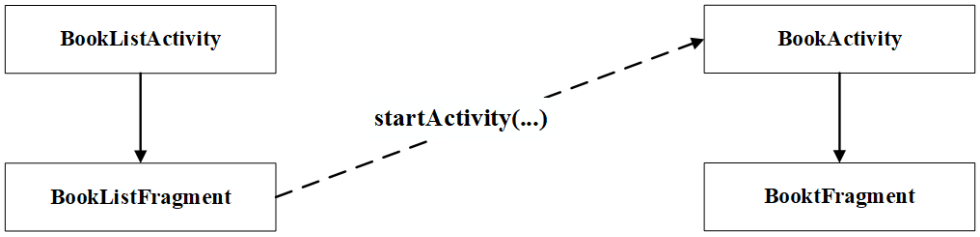

Рисунок 9.1 – Запуск BookActivity из BookListActivity

Запуск активности из фрагмента

Запуск активности из фрагмента осуществляется практически так же, как запуск активности из другой активности. Будет вызван метод `Fragment.startActivity(Intent)`, который вызывает соответствующий метод Activity во внутренней реализации.

В реализации `onListItemClick(…)` из BookListFragment заменить уведомление кодом, запускающим экземпляр BookActivity.

Листинг 9.1 – Запуск BookActivity (BookListFragment.java)

```
private class BookHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
    ...
    @Override
    public void onClick(View v) {
        ~~Toast.makeText(getActivity(),~~
            ~~mBook.getTitle() + " clicked!", Toast.LENGTH_SHORT)~~
            ~~.show();~~

        Intent intent = new Intent(getActivity(), BookActivity.class);
        startActivity(intent);
    }
}
```

Класс BookListFragment создает явный интент с указанием класса BookActivity. BookListFragment использует метод `getActivity()` для передачи активности-хоста как объекта Context, необходимого конструктору Intent.

Запустить приложение BookDepository.

Щёлкнуть на любой строке списка; открывается новый экземпляр BookActivity, управляющий фрагментом BookFragment (рисунок 9.2).

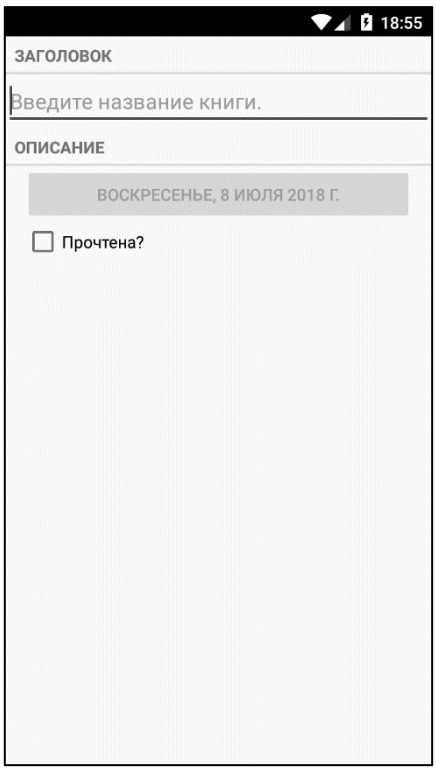

Рисунок 9.2 – Запуск пустого экземпляра BookFragment

Экземпляр BookFragment еще не содержит данных конкретного объекта Book, потому что он не получает сведений, какой именно объект Book следует отображать.

Использование дополнений

Включение дополнения

Чтобы сообщить BookFragment, какой объект Book следует отображать, можно передать идентификатор в дополнении (extra) объекта Intent при запуске BookActivity.

На первом этапе следует создать новый метод newIntent в BookActivity.

Листинг 9.2 – Создание нового метода newIntent (BookActivity.java)

```
public class BookActivity extends SingleFragmentActivity {
    public static final String EXTRA_BOOK_ID =
        "ru.rsue.android.bookdepository.book_id";
    public static Intent newIntent(Context packageContext, UUID bookId)
    {
        Intent intent = new Intent(packageContext, BookActivity.class);
        intent.putExtra(EXTRA_BOOK_ID, bookId);
        return intent;
    }
    ...
}
```

После создания явного интента вызывается метод putExtra(…), передавая строковый ключ и связанное с ним значение (bookId). В данном случае вызывается версия putExtra(String, Serializable), потому что UUID является объектом Serializable.

Затем необходимо обновить класс BookHolder, чтобы он использовал метод newIntent с передачей идентификатора книги.

Листинг 9.3 – Сохранение и передача Book (BookListFragment.java)

```
private class BookHolder extends RecyclerView.ViewHolder
        implements View.OnClickListener {
    ...
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getActivity(), BookActivity.class);
        Intent intent = BookActivity.newIntent(getActivity(),
            mBook.getId());
        startActivity(intent);
    }
}
```

Чтение дополнения

Идентификатор книги сохранен в интенте, принадлежащем BookActivity, однако прочитать и использовать эти данные должен класс BookFragment.

Существуют два способа, которыми фрагмент может обратиться к данным из интента активности: простое и прямолинейное обходное решение и сложная, гибкая полноценная реализация. Сначала будет использован первый способ, а потом реализуовано сложное гибкое решение с *аргументами фрагментов*.

В простом решении BookFragment просто использует метод getActivity() для прямого обращения к интенту BookActivity. Вернуться к классу BookFragment, прочитать дополнение из интента BookActivity и использовать его для получения данных Book.

Листинг 9.4 – Сохранение и передача Book (BookFragment.java)

```
public class BookFragment extends Fragment {
    ...
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mBook = new Book();
        UUID bookId = (UUID) getActivity().getIntent()
            .getSerializableExtra(BookActivity.EXTRA_BOOK_ID);
        mBook = BookLab.get(getActivity()).getBook(bookId);
    }
    ...
```

Если не считать вызова `getActivity()`, листинг 9.4 практически не отличается от кода выборки дополнения из кода активности. Метод `getIntent()` возвращает объект Intent, используемый для запуска BookActivity.

Метод `getSerializableExtra(String)` вызывается для Intent, чтобы извлечь UUID в переменную. После получения идентификатора, он используется для получения объекта Book от BookLab.

Обновление представления BookFragment данными Book

Теперь, когда фрагмент BookFragment получает объект Book, его представление может отобразить данные Book. Обновите метод `onCreateView(…)`, чтобы он выводил краткое описание книги и признак прочтения (код вывода даты уже имеется).

Листинг 9.5 – Обновление объектов представления (BookFragment.java)

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup parent,
Bundle savedInstanceState) {
  ...
  mTitleField = (EditText)v.findViewById(R.id.book_title);
  mTitleField.setText(mBook.getTitle());
  mTitleField.addTextChangedListener(new TextWatcher() {
      ...
  });
  ...
  mReadedCheckBox = (CheckBox)v.findViewById(R.id.book_readed);
  mReadedCheckBox.setChecked(mBook.isReaded());
  mReadedCheckBox.setOnCheckedChangeListener(new
          OnCheckedChangeListener() {
      ...
  });
  ...
  return v;
}
```

Запустить приложение BookDepository. Выбрать строку Book #4 и убедиться в том, что на экране появится экземпляр BookFragment с правильными данными книги (рисунок 9.3).

Недостаток прямой выборки

Обращение из фрагмента к интенту, принадлежащему активности-хосту, упрощает код. С другой стороны, оно нарушает инкапсуляцию фрагмента. Класс BookFragment уже не является структурным элементом, пригодным для повторного использования, потому что он предполагает, что

его хостом всегда будет активность с объектом Intent, определяющим дополнение с именем ru.rsue.android.bookdepository.book_id.

Рисунок 9.3 – Книга, выбранная в списке

Возможно, для BookFragment такое предположение разумно, но оно означает, что класс BookFragment в своей текущей реализации не может использоваться с произвольной активностью.

Другое, более правильное решение — сохранение идентификатора в месте, принадлежащем BookFragment (вместо хранения его в личном пространстве BookActivity). В этом случае объект BookFragment может прочитать данные, не полагаясь на присутствие конкретного дополнения в интенте активности. Такое «место», принадлежащее фрагменту, называется *пакетом аргументов* (arguments bundle).

Присоединение аргументов к фрагменту

Чтобы присоединить пакет аргументов к фрагменту, надо вызвать метод `Fragment.setArguments(Bundle)`. Присоединение должно быть выполнено после создания фрагмента, но до его добавления в активность.

Для этого программисты Android используют схему с добавлением в класс Fragment статического метода с именем `newInstance()`. Этот метод создает экземпляр фрагмента, упаковывает и задает его аргументы.

Когда активности-хосту потребуется экземпляр этого фрагмента, она вместо прямого вызова конструктора вызывает метод `newInstance()`.

Активность может передать newInstance(…) любые параметры, необходимые фрагменту для создания аргументов.

Включить в BookFragment метод newInstance(UUID), который получает UUID, создает пакет аргументов, создает экземпляр фрагмента, а затем присоединяет аргументы к фрагменту.

Листинг 9.6 – Метод newInstance(UUID) (BookFragment.java)

```
public class BookFragment extends Fragment {
    private static final String ARG_BOOK_ID = "book_id";
    private Book mBook;
    private EditText mTitleField;
    private Button mDateButton;
    private CheckBox mReadedCheckbox;

    public static BookFragment newInstance(UUID bookId) {
        Bundle args = new Bundle();
        args.putSerializable(ARG_BOOK_ID, bookId);
        BookFragment fragment = new BookFragment();
        fragment.setArguments(args);
        return fragment;
    }
    ...
}
```

Теперь класс BookActivity должен вызывать BookFragment.newInstance(UUID) каждый раз, когда ему потребуется создать BookFragment. При вызове передается значение UUID, полученное из дополнения. Вернуться к классу BookActivity, в методе createFragment() получить дополнение из интента BookActivity и передать его BookFragment.newInstance(UUID).

Константу EXTRA_BOOK_ID также можно сделать закрытой, потому что ни одному другому классу не потребуется работать с этим дополнением.

Листинг 9.7 – Использование newInstance(UUID) (BookActivity.java)

```
public class BookActivity extends SingleFragmentActivity {
    private static final String EXTRA_BOOK_ID =
        "ru.rsue.android.bookdepository.book_id";
    ...
    @Override
    protected Fragment createFragment() {
        return new BookFragment();
        UUID bookId = (UUID) getIntent()
            .getSerializableExtra(EXTRA_BOOK_ID);
        return BookFragment.newInstance(bookId);
    }
}
```

Потребность в независимости не является двусторонней. Класс BookActivity должен многое знать о классе BookFragment — например, то, что он содержит метод `newInstance(UUID)`. Это нормально; активность-хост должна располагать конкретной информацией о том, как управлять фрагментами, но фрагментам такая информация об их активности не нужна (по крайней мере, если необходимо сохранить гибкость независимых фрагментов).

Получение аргументов

Когда фрагменту требуется получить доступ к его аргументам, он вызывает метод `getArguments()` класса Fragment, а затем один из get-методов Bundle для конкретного типа.

В методе `BookFragment.onCreate(…)` заменить код упрощенного решения выборкой UUID из аргументов фрагмента.

Листинг 9.8 – Получение идентификатора книги из аргументов (BookFragment.java)

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    UUID bookId = (UUID) getActivity().getIntent()
        .getSerializableExtra(BookActivity.EXTRA_BOOK_ID);
    UUID bookId = (UUID) getArguments().getSerializable(ARG_BOOK_ID);
    mBook = BookLab.get(getActivity()).getBook(bookId);
}
```

Запустить приложение BookDepository. Оно работает точно так же, но реализует архитектуру с независимостью BookFragment.

Перезагрузка списка

Запустить приложение BookDepository, щёлкнуть на элементе списка и внести изменения в подробную информацию о книге. Эти изменения сохраняются в модели, но при возвращении к списку содержимое RecyclerView остается неизменным.

Адаптеру RecyclerView необходимо сообщить, что набор данных изменился (или мог измениться), чтобы тот мог заново получить данные и повторно загрузить список. Работая со стеком возврата ActivityManager, можно перезагрузить список в нужный момент.

Когда BookListFragment запускает экземпляр BookActivity, последний помещается на вершину стека. При этом экземпляр BookActivity, который до этого находился на вершине, приостанавливается и останавливается.

Когда пользователь нажимает кнопку Back для возвращения к списку, экземпляр BookActivity извлекается из стека и уничтожается. В этот момент BookListActivity запускается и продолжает выполнение (рисунок 9.4).

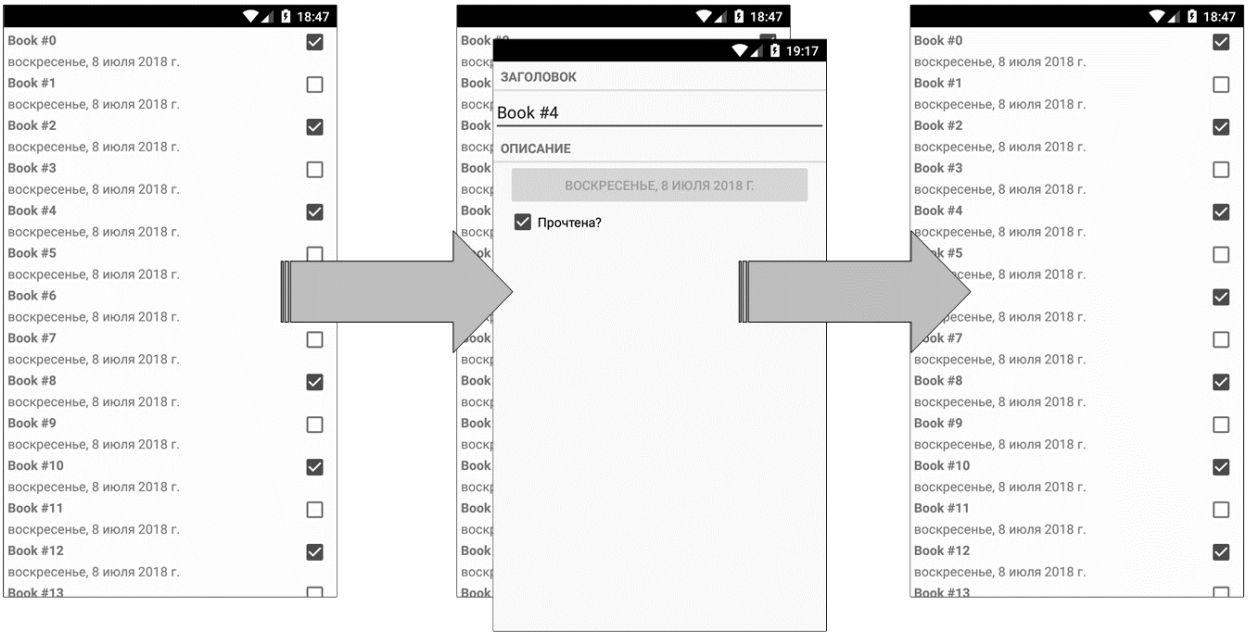

Рисунок 9.4 – Стек возврата BookDepository

Когда экземпляр BookListActivity продолжает выполнение, он получает вызов `onResume()` от ОС. При получении этого вызова BookListActivity его экземпляр FragmentManager вызывает `onResume()` для фрагментов, хостом которых в настоящее время является активность. В данном случае это единственный фрагмент BookListFragment.

В классе BookListFragment переопределить `onResume()` и инициировать вызов updateUI() для перезагрузки списка. Изменить метод `updateUI()` для вызова `notifyDataSetChanged()`, если объект BookAdapter уже создан.

Листинг 9.9 – Перезагрузка списка в onResume() (BookListFragment.java)

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
    Bundle savedInstanceState) {
    ...
}

@Override
public void onResume() {
    super.onResume();
    updateUI();
}

private void updateUI() {
    BookLab bookLab = BookLab.get(getActivity());
    List<Book> books = bookLab.getBooks();
    if (mAdapter == null) {
```

```
            mAdapter = new BookAdapter(books);
            mBookRecyclerView.setAdapter(mAdapter);
        } else {
            mAdapter.notifyDataSetChanged();
        }
    }
```

Запустить приложение BookDepository. Выбрать книгу в списке и изменить её подробную информацию. Вернувшись к списку, пользователь немедленно увидит свои изменения.

Самостоятельные задания.

Задание. Эффективная перезагрузка RecyclerView

Метод `notifyDataSetChanged` адаптера хорошо подходит, для того чтобы приказать RecyclerView перезагрузить все элементы, видимые в настоящее время.

В BookDepository этот метод неэффективен, потому что при возвращении к BookListFragment заведомо изменилось не более одного объекта Book.

Использовать метод `notifyItemChanged(int)` объекта `RecyclerView.Adapter`, чтобы перезагрузить один элемент в списке. Изменить код для вызова этого метода несложно; труднее обнаружить, в какой позиции произошло изменение, и перезагрузить правильный элемент. Для этого использовать метод `getAdapterPosition()`.

10. VIEWPAGER

В данном разделе будет создана новая активность, которая станет хостом для BookFragment. Макет активности будет состоять из экземпляра ViewPager. Включение виджета ViewPager в пользовательский интерфейс позволит «листать» элементы списка, проводя пальцем по экрану (рисунок 10.1).

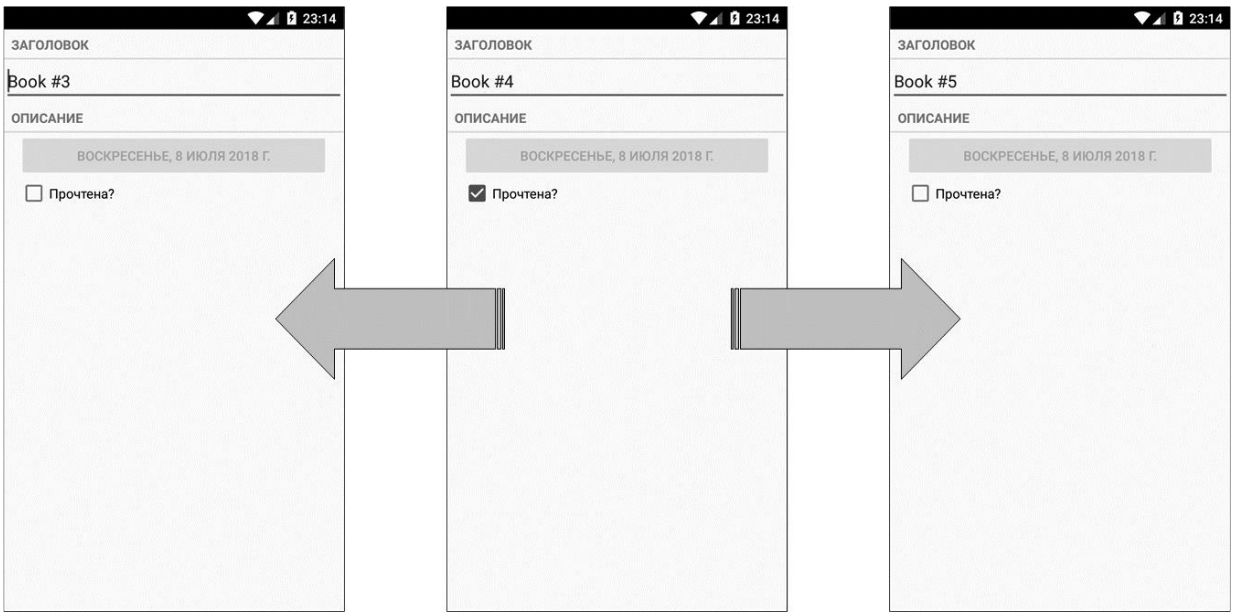

Рисунок 10.1 – Листание страниц

На рисунке 10.2 представлена обновленная диаграмма BookDepository.

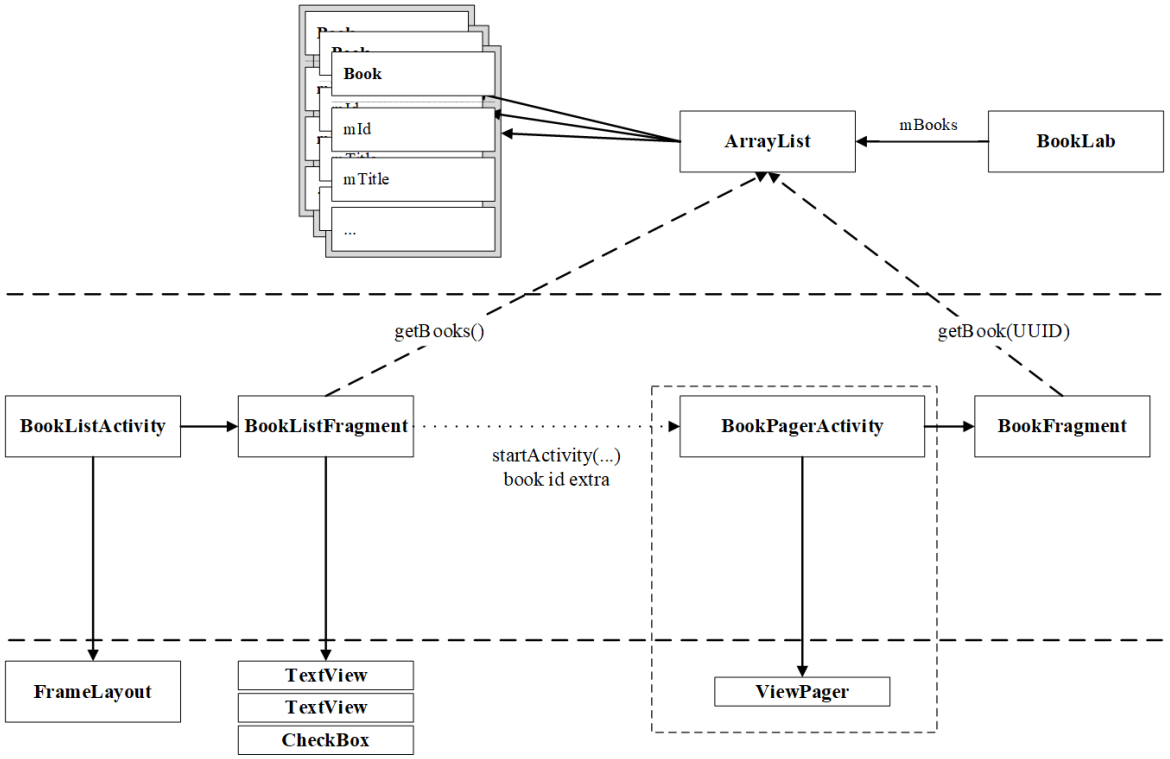

Рисунок 10.2 – Диаграмма объектов BookPagerActivity

Новая активность с именем BookPagerActivity займет место BookActivity. Ее макет состоит из экземпляра ViewPager. Все новые объекты, которые необходимо создать, находятся в пунктирном прямоугольнике на приведенной диаграмме. Для реализации листания страничных представлений в BookDepository ничего другого менять не придется. В частности, класс BookFragment останется неизменным благодаря той работе по обеспечению независимости BookFragment, которая была проведена в предыдущем разделе.

Создание BookPagerActivity

Класс BookPagerActivity будет субклассом FragmentActivity. Он создает экземпляр и управляет ViewPager. Создайте новый класс с именем BookPagerActivity. Назначьте его суперклассом FragmentActivity и создать представление для активности.

Листинг 10.1 – Создание ViewPager (BookPagerActivity.java)

```
public class BookPagerActivity extends FragmentActivity {
      @Override
      protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_book_pager);
      }
}
```

Файл макета еще не существует. Создать новый файл макета в res/layout/ и присвойте ему имя activity_book_pager. Назначить его корневым представлением ViewPager и присвойть ему атрибуты, показанные на рисунке 10.3. Обратить внимание на необходимость использования полного имени пакета ViewPager (`android.support.v4.view.ViewPager`).

```
android.support.v4.view.ViewPager
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/activity_book_pager_view_pager"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

Рисунок 10.3 – Определение ViewPage в BookPagerActivity (activity_book_pager.xml)

Полное имя пакета используется при добавлении в файл макета, потому что класс ViewPager определен в библиотеке поддержки. В отличие от Fragment, класс ViewPager доступен *только* в библиотеке поддержки; в более поздних версиях SDK так и не появилось «стандартного» класса ViewPager.

ViewPager и PagerAdapter

Класс ViewPager в чем-то похож на RecyclerView. Чтобы класс RecyclerView мог выдавать представления, ему необходим экземпляр Adapter. Классу ViewPager также необходим адаптер PagerAdapter.

Однако взаимодействие между ViewPager и PagerAdapter намного сложнее взаимодействия между RecyclerView и Adapter. Но можно использовать FragmentStatePagerAdapter — субкласс PagerAdapter, который берет на себя многие технические подробности.

FragmentStatePagerAdapter сводит взаимодействие к двум простым методам: `getCount()` и `getItem(int)`. При вызове метода `getItem(int)` для позиции в массиве книг следует вернуть объект BookFragment, настроенный для вывода информации объекта в заданной позиции.

В классе BookPagerActivity Добавить следующий код для назначения PagerAdapter класса ViewPager и реализации его методов `getCount()` и `getItem(int)`.

Листинг 10.2 – Назначение PagerAdapter (BookPagerActivity.java)

```
public class BookPagerActivity extends FragmentActivity {
    private ViewPager mViewPager;
    private List<Book> mBooks;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_book_pager);

        mViewPager = (ViewPager) findViewById(
                R.id.activity_book_pager_view_pager);
        mBooks = BookLab.get(this).getBooks();
        FragmentManager fragmentManager = getSupportFragmentManager();
        mViewPager.setAdapter(new
                        FragmentStatePagerAdapter(fragmentManager) {
            @Override
            public Fragment getItem(int position) {
                Book book = mBooks.get(position);
                return BookFragment.newInstance(book.getId());
            }
            @Override
            public int getCount() {
                return mBooks.size();
            }
        });
    }
}
```

После поиска ViewPager в представлении активности получается от BookLab набор данных — контейнер List объектов Book. Затем получается экземпляр FragmentManager для активности.

На следующем шаге адаптером назначается безымянный экземпляр FragmentStatePagerAdapter. Для создания FragmentStatePagerAdapter необходим объект FragmentManager. FragmentStatePagerAdapter — агент, управляющий взаимодействием с ViewPager. Чтобы агент мог выполнить свою работу с фрагментами, возвращаемыми в getItem(int), он должен быть способен добавить их в активность. Для этого необходим экземпляр FragmentManager.

Интеграция контроллера

Теперь можно переходить к устранению класса BookActivity и замене его классом BookPagerActivity.

Начать нужно с добавления метода newIntent в BookPagerActivity вместе с дополнением для идентификатора книги.

Листинг 10.3 – Создание newIntent (BookPagerActivity.java)

```
public class BookPagerActivity extends FragmentActivity {
    private static final String EXTRA_BOOK_ID =
            "ru.rsue.android.bookdepository.book_id";

    private ViewPager mViewPager;
    private List<Book> mBooks;

    public static Intent newIntent(Context packageContext, UUID bookId)
    {
        Intent intent = new Intent(packageContext,
                BookPagerActivity.class);
        intent.putExtra(EXTRA_BOOK_ID, bookId);
        return intent;
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_book_pager);
        UUID bookId = (UUID) getIntent()
                .getSerializableExtra(EXTRA_BOOK_ID);
        ...
    }
}
```

Теперь нужно сделать так, чтобы при выборе элемента списка в BookListFragment запускался экземпляр BookPagerActivity вместо BookActivity.

Вернуться к файлу BookListFragment.java и изменить метод BookHolder.onClick(…), чтобы он запускал BookPagerActivity.

Листинг 10.4 – Запуск активности (BookListFragment.java)

```
private class BookHolder extends RecyclerView.ViewHolder
implements View.OnClickListener {
    ...
    @Override
    public void onClick(View v) {
        Intent intent = BookActivity.newIntent(getActivity(),
                        mBook.getId());
        Intent intent = BookPagerActivity.newIntent(getActivity(),
                        mBook.getId());
        startActivity(intent);
    }
}
```

Необходимо добавить BookPagerActivity в манифест, чтобы ОС могла запустить эту активность. Для этого достаточно заменить в манифесте BookActivity на BookPagerActivity.

Листинг 10.5 – Добавление BookPagerActivity в манифест (AndroidManifest.xml)

```
<application ...>
    ...
    <activity
        android:name=".BookActivity"
        android:name=".BookPagerActivity"
        android:label="@string/app_name" >
    </activity>
    ...
</application>
```

Наконец, чтобы не загромождать проект, удалить BookActivity.java в окне инструментов Project.

Запустить приложение BookDepository. Нажать на строке Book #0, чтобы просмотреть подробную информацию. Провести по экрану влево или вправо, чтобы просмотреть другие элементы списка. Переключение страниц происходит плавно и без задержек. По умолчанию ViewPager загружает элемент, находящийся на экране, а также по одному соседнему элементу в каждом направлении, чтобы отклик на жест прокрутки был немедленным. Количество загружаемых соседних страниц можно настроить вызовом `setOffscreenPageLimit(int)`.

По умолчанию ViewPager отображает в своем экземпляре PagerAdapter первый элемент. Чтобы вместо него отображался элемент, выбранный

пользователем, назначьте текущим элементом ViewPager элемент с указанным индексом.

В конце `BookPagerActivity.onCreate(…)` найти индекс отображаемой книги; для этого перебрать и проверить идентификаторы всех книг. Когда будет найден экземпляр Book, у которого поле mId совпадает с bookId в дополнении интента, изменить текущий элемент по индексу найденного объекта Book.

Листинг 10.6 – Назначение исходного элемента (BookPagerActivity.java)

```
public class BookPagerActivity extends FragmentActivity {
   @Override
   public void onCreate(Bundle savedInstanceState) {
      ...
      FragmentManager fragmentManager = getSupportFragmentManager();
      mViewPager.setAdapter(new
               FragmentStatePagerAdapter(fragmentManager) {
         ...
      });
      for (int i = 0; i < mBooks.size(); i++) {
            if (mBooks.get(i).getId().equals(bookId)) {
            mViewPager.setCurrentItem(i);
            break;
         }
      }
   }
}
```

Запустить приложение BookDepository. При выборе любого элемента списка должна отображаться подробная информация правильного объекта Book. Теперь экземпляр ViewPager полностью готов к работе.