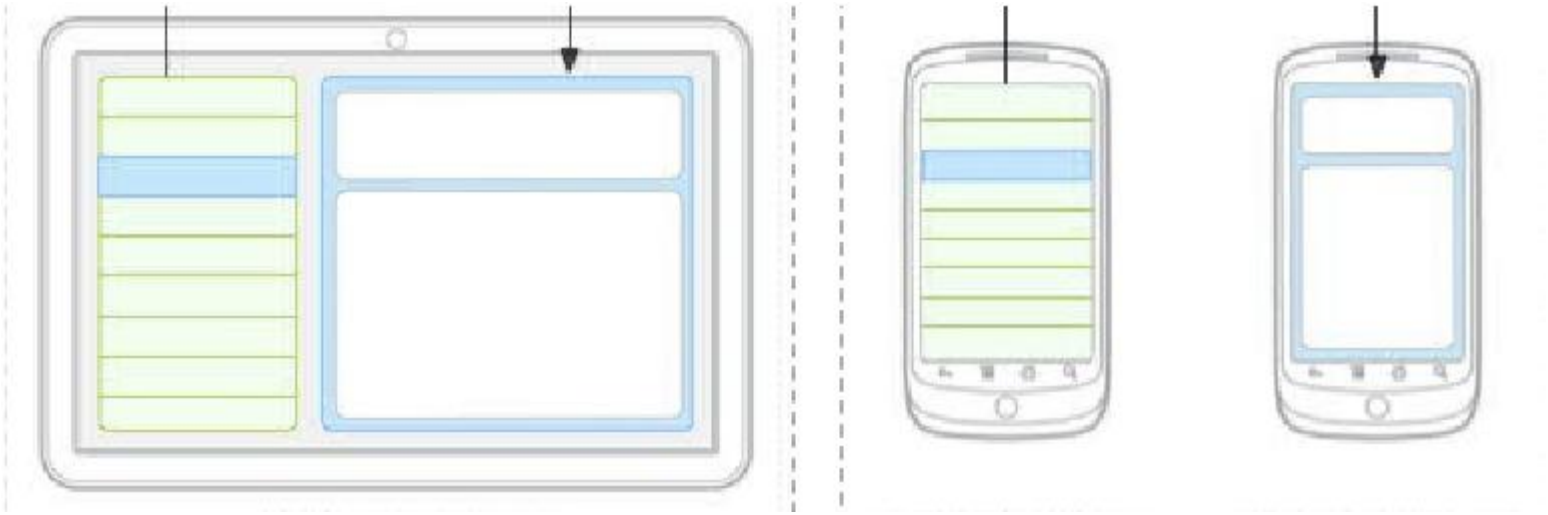
Mobile Development
Lecture 10: Fragments

Mahmoud El-Gayyar
elgayyar@ci.suez.edu.eg
Elgayyar.weebly.com

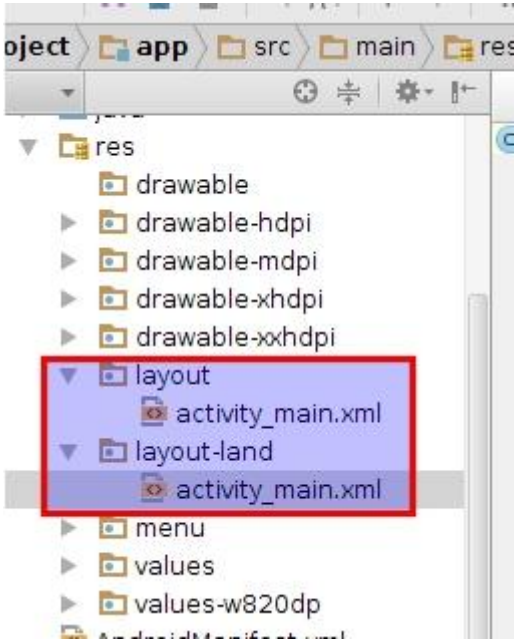
Situational Layouts

- *Your app can use different layout in different situations:*
  - different device type (tablet vs phone vs watch)
  - different screen size
  - different orientation (portrait vs. landscape)
  - different country or locale (language, etc.)



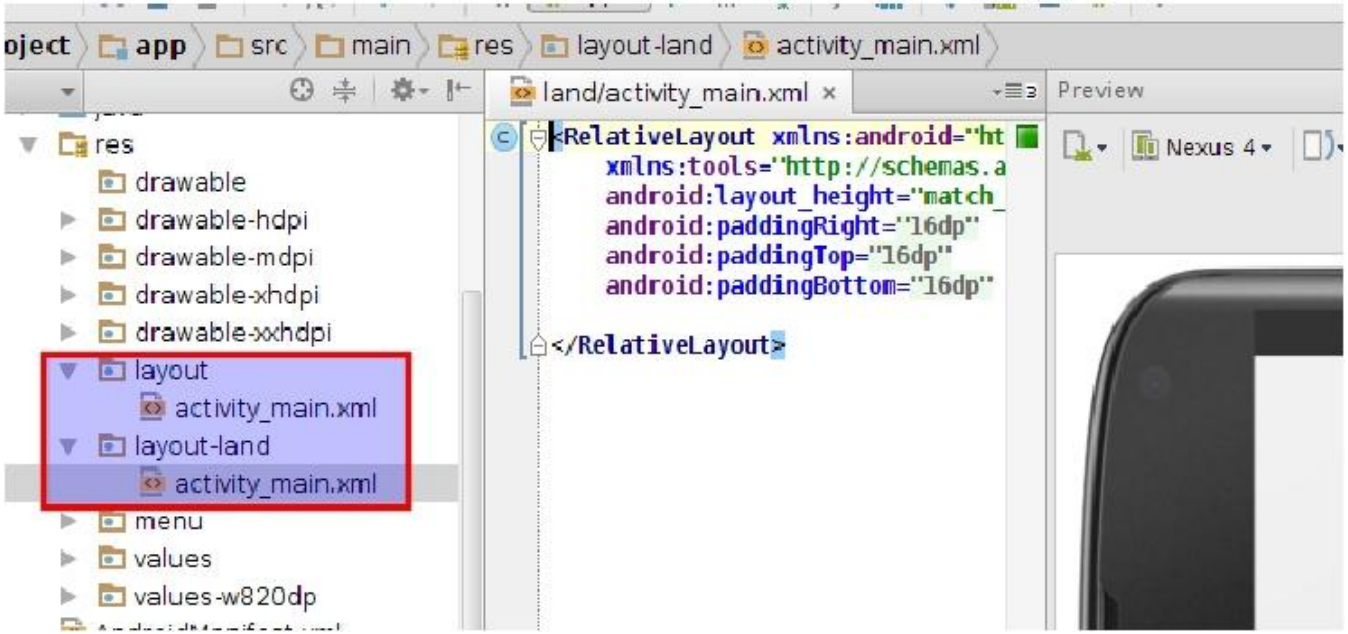
Situation-Specific Folders

- *Your app will look for resource folder names with suffixes:*
  - screen density (e.g. drawable-hdpi)
    - *xhdpi: 2.0 (twice as many pixels/dots per inch)*
    - *hdpi: 1.5*
    - *mdpi: 1.0 (baseline)*
    - *ldpi: 0.75*
  - orientation (e.g. layout-land)
    - *portrait (), land (landscape)*



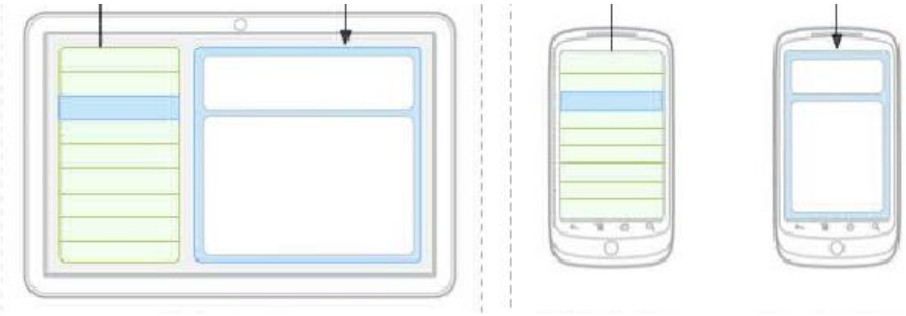
Portrait vs Landscape Layout

- *To create a different layout in landscape mode:*
  - create a folder in your project called **res/layout-land**
  - place another copy of your activity's **layout XML file** there
  - modify it as needed to represent the differences



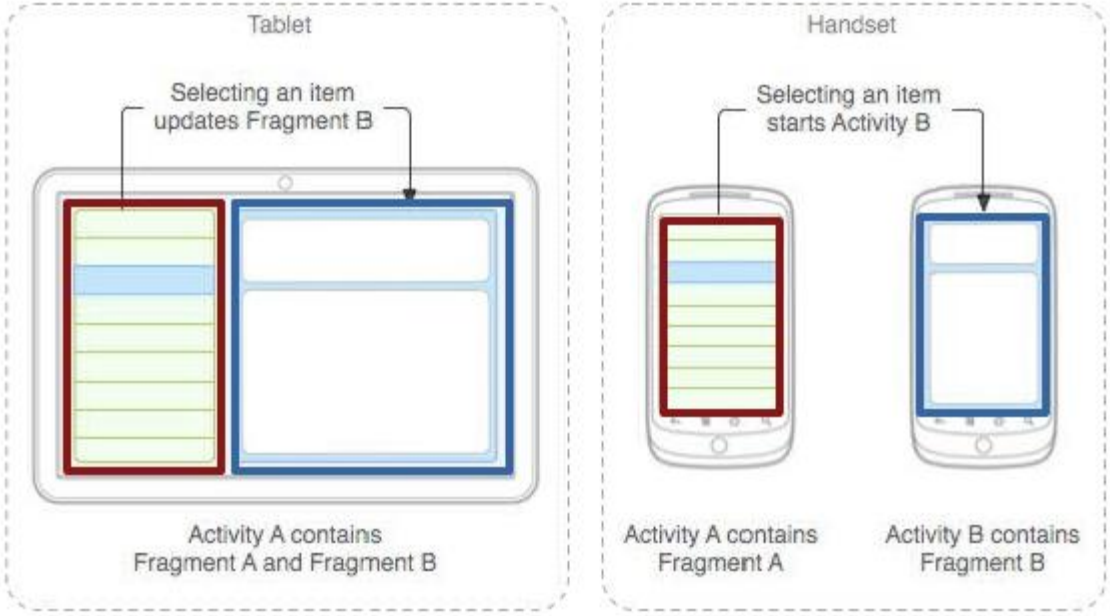
Problem: Redundant Layouts

- *With situational layout you begin to encounter redundancy.*
  - The layout in one case (e.g. portrait or medium) is very similar to the layout in another case (e.g. landscape or large).
  - You don't want to represent the same XML or Java code multiple times in multiple places.
- *You sometimes want your code to behave situationally.*
  - In portrait mode, clicking a button should launch a new activity.
  - In landscape mode, clicking a button should launch a new view.



Fragments

- ***Fragment**: A reusable segment of Android UI that can appear in an activity.*
  - can help handle different devices and screen sizes
  - fragments can be swapped into and out of activities without stopping them
  - can reuse a common fragment across multiple activities
  - first added in Android 3.0

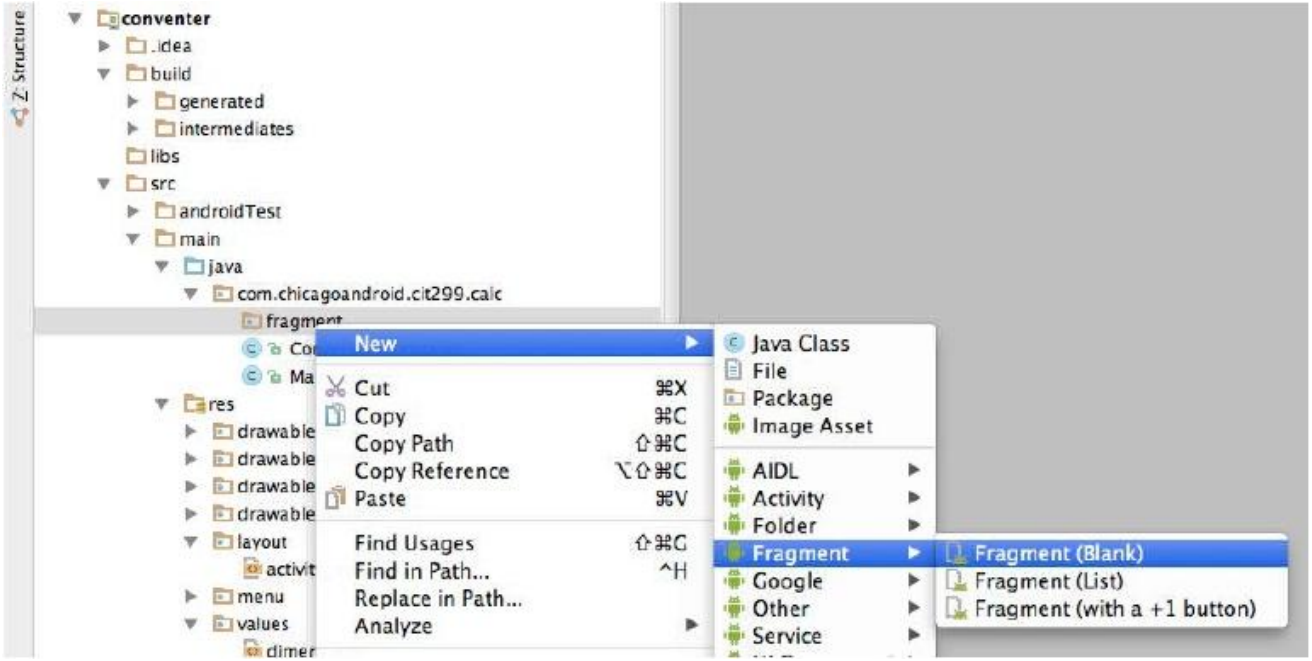


Creating a Fragment

- *In Android Studio, right-click app, click:*

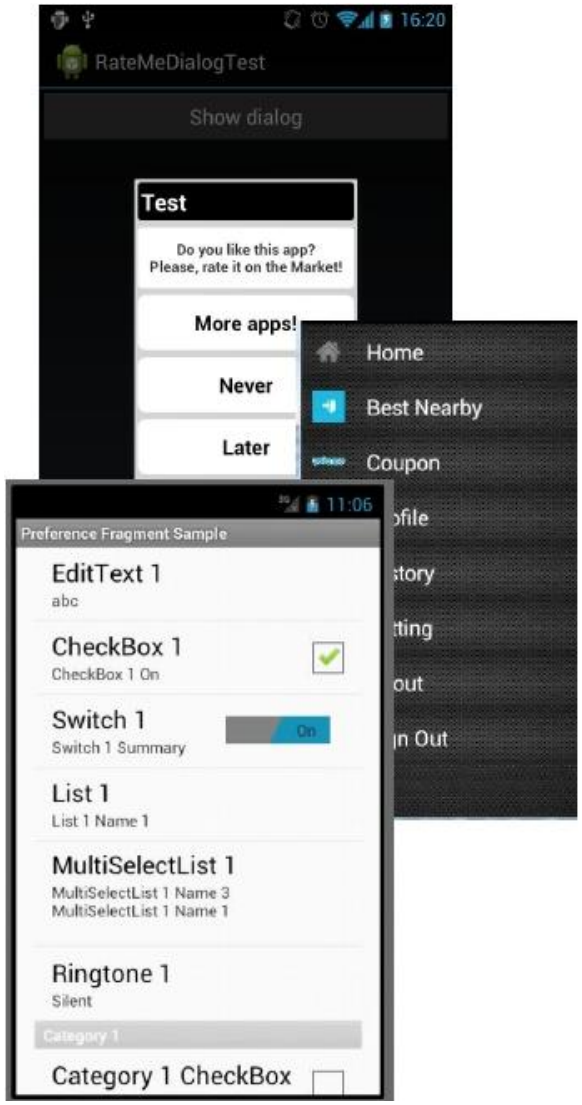
  New → Fragment → Fragment (blank)

  - now create layout XML and Java event code as in an Activity



Fragment Subclasses

- ***DialogFragment*** *- a fragment meant to be shown as a dialog box that pops up on top of the current activity.*
- ***ListFragment*** *- a fragment that shows a list of items as its main content.*
- ***PreferenceFragment*** *- a fragment whose main content is meant to allow the user to change settings for the app.*



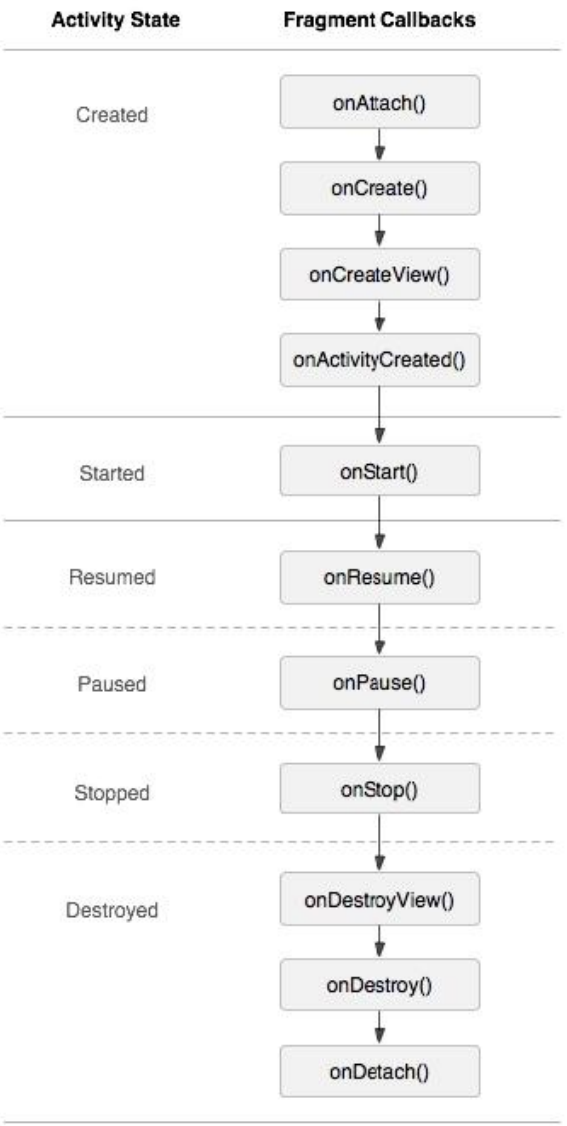
Using Fragments in Activity XML

- *Activity layout XML can include fragments.*

```
<!-- activity_name.xml -->
<LinearLayout ...>
      <fragment ...
            android:id="@+id/id1"
            android:name="ClassName1"
            tools:layout="@layout/name1" />
      <fragment ...
            android:id="@+id/id2"
            android:name="ClassName2"
            tools:layout="@layout/name2" />
</LinearLayout>
```

Fragment Life Cycle

- *Fragments have a similar life cycle and events as activities.*
- *Important methods:*
  - **onAttach:** to glue fragment to its surrounding activity
  - **onCreate:** when fragment is loading
  - **onCreateView:** method that must return fragment's root UI view
  - **onActivityCreated:** method that indicates the enclosing activity is ready
  - **onPause:** when fragment is being left/exited
  - **onDetach:** just as fragment is being deleted



Fragment Template

```
public class Name extends Fragment {
      @Override
      public View onCreateView(LayoutInflater inflater,
                  ViewGroup vg, Bundle bundle) {
            // load the GUI layout from the XML
            return inflater.inflate(R.layout.id, vg, false);
      }

      public void onActivityCreated(Bundle savedState) {
            super.onActivityCreated(savedState);
            // ... any other GUI initialization needed
      }

      // any other code (e.g. event-handling)
}
```

Fragment vs. Activity

- *Fragment code is similar to activity code, with a few changes:*
  - Many activity methods aren't present in the fragment, but you can call **getActivity** to access the activity the fragment is inside of.
  - Button b = (Button) findViewById(R.id.but);
  - Button b = (Button) getActivity().findViewById(R.id.but);
  - Sometimes also use **getView** to refer to the activity's layout
  - Event handlers cannot be attached in the XML any more. :-(
    - *Must be attached in Java code instead.*
  - Passing information to a fragment (via Intents) is trickier.
    - *The fragment must ask its enclosing activity for the information.*

Fragment onClick Listener

- *Activity:*

```
<Button android:id="@+id/b1"
        android:onClick="onClickB1" ... />
```

- *Fragment:*

```
<Button android:id="@+id/b1" ... />

// in fragment's Java file
Button b = (Button) getActivity().findViewById(r.id.b1);
b.setOnClickListener(new View.OnClickListener() {
     @Override public void onClick(View view) {
          // whatever code would have been in onClickB1
     }
});
```

Activity that accepts Parameters

```
public class Name extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.name);

        // extract parameters passed to activity from intent
        Intent intent = getIntent();
        int name1 = intent.getIntExtra("id1", default);
        String name2 = intent.getStringExtra("id2", "default");

        // use parameters to set up the initial state
        ...
    }
    ...
}
```

Fragment that accepts Parameters

```
public class Name extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
            ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.name, container, false);
    }

    @Override
    public void onActivityCreated(Bundle savedState) {
        super.onActivityCreated(savedState);

        // extract parameters passed to activity from intent
        Intent intent = getActivity().getIntent();
        int name1 = intent.getIntExtra("id1", default);
        String name2 = intent.getStringExtra("id2", "default");

        // use parameters to set up the initial state
        ...
    }
```

Fragment that accepts Parameters

- *Fragments can have a Bundle object attached to them*
- *referred to as **arguments***
- *Create Bundle and attach after fragment created, but before fragment added to Activity*
- ***Convention**: create static method newInstancethat creates Fragment and bundles up arguments*

getArguments

- *Activity:*

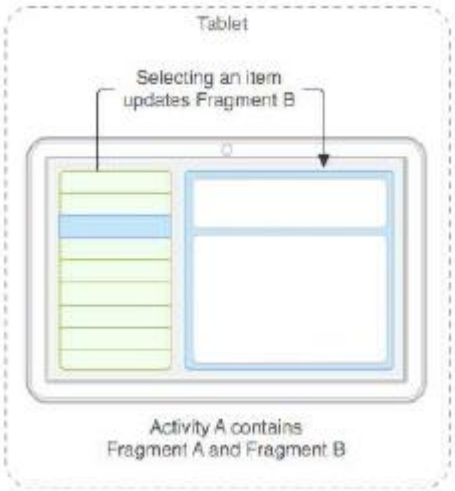
```
/**
 * Create a new instance of DetailsFragment, initialize
 * show the text at 'index'.
 */
public static DetailsFragment newInstance(int index) {
    DetailsFragment f = new DetailsFragment();

    // Supply index input as an argument.
    Bundle args = new Bundle();
    args.putInt("index", index);
    f.setArguments(args);

    return f;
}
```

- *Fragment:*

```
public int getShownIndex() {
    return getArguments().getInt("index", 0);
}
```

Communication between Fragments

- *One activity might contain multiple fragments.*
- *The fragments may want to talk to each other.*
  - Use activity's getFragmentManager method.
  - its findFragmentById method can access any fragment that has an id.




```
Activity act = getActivity();
if (act.getResources().getConfiguration().orientation ==
        Configuration.ORIENTATION_LANDSCAPE) {
    // update other fragment within this same activity
    FragmentClass fragment = (FragmentClass)
        act.getFragmentManager().findFragmentById(R.id.id);
    fragment.methodName(parameters);
}
```

Add/Replace Fragments Programmatically

```
//create a new instance of your fragment
details= DetailsFragment.newInstance(index);
//use a fragment transaction object
FragmentTransaction ft= getFragmentManager().beginTransaction();
//add a new fragment
ft.add(R.id.details,details);        //first parameter is layout id
//or replace an existing fragment
ft.replace(R.id.details,details);
//set transition animation
ft.setTransition(FragmentTransaction.TRANSIT_FRAGMENT_FADE);
//do the action
ft.commit();
```

Use of Fragments

- *Android development documents recommend ALWAYS using Fragments*
- *Provide for flexibility of UIs*
- *Activity tightly coupled with its View*
- *Fragments provide flexibility, looser coupling between Activity and UI Views*
  - fragment becomes a building block
- *downside, more complexity in code, more moving parts*

Assignment

- *Go to SDK → Samples → android-? (17) → ApiDemos app*
- *Displays Shakespeare play titles in a List*
- *Clicking on a title displays a sample from the play*
- *com.example.android.apis.app*
  - FragmentLayout.java
  - FragmentDialog.java
  - …

Assignment


9:04
App/Fragment/Layout
Henry IV (1)
Henry V
Henry VIII
Richard II
Richard III
Merchant of Venice
Othello
King Lear
10:23
API Demos
Hear him but reason in divinity,And all-admiring with an inward wishYou would desire the king were made a prelate:Hear him debate of commonwealth affairs,You would say it hath been all in all his study:List his discourse of war, and you shall hearA fearful battle render'd you in music:Turn him to any cause of policy,The Gordian knot of it he will unloose,Familiar as his garter: that, when he speaks,The air, a charter'd libertine, is still,And the mute wonder lurketh in men's ears,To steal his sweet and honey'd sentences;So that the art and practic part of lifeMust be the mistress to this theoric:Which
Details Fragment
10:23
App/Fragment/Layout
Henry IV (1)
Henry V
Henry VIII
Richard II
Richard III
Merchant of Venice
Hear him but reason in divinity,And all-admiring with an inward wishYou would desire the king were made a prelate:Hear him debate of commonwealth affairs,You would say it hath been all in all his study:List his discourse of war, and you shall hearA fearful battle render'd you in music:Turn him to any cause of policy,The Gordian knot of it he will unloose,Familiar as his garter: that, when he speaks,The air, a charter'd libertine, is
Titles Fragment