**FRAGMENT**

**Si descriva il meccanismo dei Frammenti. Si faccia un esempio per illustrarne l’utilità.**

I frammenti sono porzioni di interfaccia utente che hanno una vita propria nell’activity che li ospita: per questo, talvolta sono chiamati anche sub-activity. Essi hanno un differente ciclo di vita dall’activity, ma esso dipende da quello dell’activity che lo ospita: se l’activity ospitante viene distrutta, vengono distrutti anche i vari frammenti.

Un frammento può essere creato staticamente utilizzando un ***tag <fragment>*** da inserire nel layout dell’activity ospitante. Il layout del frammento, tuttavia, dev’essere specificato in un file XML separato. Nel momento in cui viene invocato il metodo ***onCreateView()*** del frammento, occorre effettuare ***l’inflate*** del file di layout corrispondente nel ***tag <fragment>*** del layout principale: dopo questa operazione, il frammento è ***Created*** nell’activity che lo ospita. Dopo che l’activity viene stoppata e prima che essa viene distrutta, viene effettuato il detatch del frammento (metodo ***onDetatch()***).

Una classe ***Fragment*** cura l’utilizzo dinamico del frammento nel corso dell’interazione con l’utente.

Ad esempio, un frammento può essere anche inserito dinamicamente nel layout a runtime. Occorre eseguire il codice posto di seguito:

|  |
| --- |
| *FragmentManager fm = getFragmentManager();*  *FragmentTransaction ft = fm.beginTransaction();*  *ExampleFragment fragment = new ExampleFragment();*  *ft.add(R.id.fragment\_container, fragment); /\* R.id.fragment\_container è un ViewGroup nel layout dell’activity che individua la porzione dello schermo da dedicare a questo frammento \*/* |

Normalmente, un frammento non viene aggiunto nel backstack: si usa il metodo ***addToBackstack()*** per inserire i cambiamenti nel backstack; questo perché il backstack considera solo le activity, pertanto dobbiamo gestire manualmente i frammenti. Se non chiamiamo questo metodo, quando premiamo Back salteremo i cambiamenti fatti con i frammenti: non è quello che l’utente si aspetta.

Durante il corso è stato analizzato un esempio in cui occorreva far comunicare due frammenti. In tal senso, è bene specificare che un frammento dovrebbe evitare la comunicazione diretta con un altro frammento della stessa activity, in quanto ciò riduce la riusabilità; al contrario, è opportuno utilizzare l’activity ospitante come “comunicatore” tra il frammento mittente ed il frammento destinatario.

**Per cosa sono utili i Frammenti?**

Vengono usati i frammenti per costruire interfacce utenti dinamiche che permettono di adattarsi a diverse dimensioni degli schermi. Sono utili per incapsulare la logica dell’applicazione, per migliorare la gestione del ciclo di vita e per riutilizzarli in altre attività. Ad esempio mediante i frammenti si può decidere di inserire un ***listView*** che occupa metà dello schermo e un ***webView*** che occupa l’altra metà in modo tale che quando si clicca su un elemento del frammento A, viene passata l’informazione alla ***webView*** nel frammentoB e immediatamente si vede cosa viene ricercato senza dover effettuare un cambio di attività.

**Se due frammenti di un activity devono comunicare è buona prassi di programmazione implementare tale comunicazione non in modo diretto da frammento a frammento ma passando attraverso l’activity che ospita i frammenti (quindi il frammento che vuole inviare la comunicazione lo fa interagendo con l’activity ospitante e poi questa interagisce con il frammento che deve ricevere la comunicazione). Perché è una buona prassi di programmazione? Si descriva un modo per implementare la comunicazione fra due frammenti attraverso l’activity ospitante.**

È una buona prassi di programmazione perché favorisce il riuso del codice.

Si assuma di avere un’applicazione che, selezionato un nome di un autore, visualizzerà la sua citazione. Per quest’azione vengono utilizzati 2 frammenti (FragmentAutori, FragmentCitazioni). La prima operazione che deve essere effettuata è quella della visualizzazione dei nomi degli autori. Dunque nell’*onCreate()* della ***MainActivity*** istanzieremo un riferimento alla classe FragmentAutori mediante un ***FragmentManager***. Per non far comunicare direttamente i 2 frammenti viene settato nella classe FragmentAutori, un metodo che setta in una variabile della classe FragmentAutori, un riferimento alla classe ***MainActivity***, in maniera tale che, quando verrà selezionato un autore, la classe FragmentAutori, mediante un ***onItemClick***, invocherà tramite il riferimento alla ***MainActivity***, il metodo della classe ***MainActivity*** (respond) che gestirà la richiesta e invocherà il metodo del fragmentCitazione che mostrerà la citazione.

**Scrivere il metodo onCreate di un’app che utilizza due frammenti e che all’avvio faccia partire i due frammenti. I due frammenti dividono lo schermo in due parti uguali. Mostrare anche i file xml per il layout. Si assuma di avere a disposizione le classi FrammentoA e FrammentoB dei due frammenti.**

|  |  |
| --- | --- |
| *protected void onCreate(Bundle savedInstanceState) {*  *super.onCreate(savedInstanceState);*  *setContentView(R.layout.activity\_main);*  *FragmentManager fm= getSupportFragmentManager();*  *FragmentTransaction ft = fm.beginTransaction();*  *FrammentoA fragmentA = new FrammentoA();*  *FrammentoB fragmentB = new FrammentoB();*  *ft.add(R.id.fragment\_container, fragmentA);*  *ft.add(R.id.fragment\_container, fragmentB);*  *ft.commit();*  *}* | *<FrameLayout*  *xmlns:tools="http://schemas.android.com/tools"*  *android:layout\_width="match\_parent"*  *android:layout\_height="match\_parent"*  *tools:context="it.unisa.di.mp.myfragmentstatic.FragmentA"*  ***android:layout\_weight="1"****>*  *//TO - DO*  *</FrameLayout>* |

**Scrivere il frammento di codice da usare nel metodo onCreate() di un’activity che permette di inserire un solo frammento (Frammento1) se la larghezza del display è minore di 300dp e di inserire due frammenti (Frammento1 e Frammento2) se la larghezza del display è uguale o maggiore di 300dp. Si assume che il layout preveda un LinearLayout orizzontale con due frame per inserire i frammenti con id, rispettivamente, frame1 e frame2.**

|  |
| --- |
| *@Override*  *protected void onCreate(Bundle savedInstanceState) {*  *super.onCreate(savedInstanceState);*  *setContentView(R.layout.activity\_main);*  *FragmentManager fm= getSupportFragmentManager();*  *FragmentTransaction ft = fm.beginTransaction();*  *Frammento1 fragmentA = new Frammento1();*  *Frammento2 fragmentB = new Frammento2();*  *ft.add(R.id.frame1, fragmentA);*  *float dpWidth = calcolaLarghezzaDispayInDp();*  *if(dpWidth >= 300) ft.add(R.id.frame2, fragmentB);*  *ft.commit();*  *}* |

**State progettando un’app che utilizza dei frammenti. Avete già scritto la classe MioFrammento che implementa un singolo frammento. Adesso dovete scrivere nell’activity principale il codice che inserisce 4 frammenti MioFrammento. Il file di layout prevede 4 frame per ospitare i 4 frammenti e gli id dei 4 frame sono frameFrammento1, frameFrammento2, frameFrammento3, frameFrammento4. Mostrare il codice che serve a creare ed inserire dinamicamente i frammenti. Il costruttore della classe MioFrammento prende in input una stringa che servirà per mostrare il “nome” del frammento; pertanto dovete passare una stringa “Frammento i”, dove i=1,2,3,4, per ognuno dei frammenti da creare ed inserire.**

|  |
| --- |
| *protected void onCreate(Bundle savedInstanceState) {*  *super.onCreate(savedInstanceState);*  *setContentView(R.layout.activity\_main);*  *FragmentManager fm= getSupportFragmentManager();*  *FragmentTransaction ft= fm.beginTransaction();*  *MioFrammento mf1= new MioFrammento("MioFrammento1");*  *MioFrammento mf2= new MioFrammento("MioFrammento2");*  *MioFrammento mf3= new MioFrammento("MioFrammento3");*  *MioFrammento mf4= new MioFrammento("MioFrammento4");*  *ft.add(R.id.frameFrammento1, mf1);*  *ft.add(R.id.frameFrammento2, mf2);*  *ft.add(R.id.frameFrammento3, mf3);*  *ft.add(R.id.frameFrammento4, mf4);*  *ft.commit();*  *}* |