

Übung 3: Persistenz – Preferences, Dateisystem, Room & Content Providers

In dieser Übung verwenden Sie die drei Möglichkeiten, um auf Android Daten persistent zu speichern: *Preferences*, das *Dateisystem* und die *Room-Datenbank*. Für die Darstellung der Preferences verwenden wir dazu ein `PreferenceFragment`, welches für die Darstellung von Einstellungs-Bildschirmen praktische Unterstützung bietet. Danach speichern wir Daten in einer Datei und lesen diese wieder von dort ein. Anschliessend speichern wir Notizen in einer Room-DB und lesen diese wieder aus. Zum Abschluss verwenden Sie einen Content Resolver, um via Content-Provider auf Daten einer anderen Applikation zuzugreifen.

1. Neue App: Persistenz

Erstellen Sie ein neues Android-Applikationsprojekt „Persistenz“. Diese Übung besteht aus verschiedenen Teilaufgaben, welche in der `MainActivity` dieses Projekts soweit als möglich resp. sinnvoll dargestellt werden. Die `MainActivity` (Layout-Datei: `activity_main.xml`) sollte am Schluss ungefähr so aussehen wie der Screenshot rechts. Verwenden Sie dazu eine `ScrollView` mit einem `LinearLayout` (siehe Übung 2).

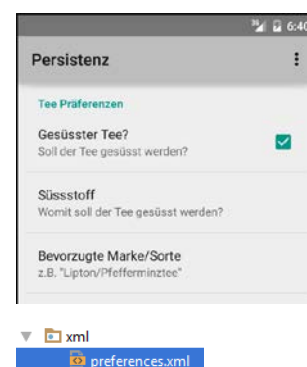


2. Private Preferences: Resume Counter

Zählen Sie auf Ihrer `MainActivity` die Anzahl der Aufrufe der `onResume`-Methode. Verwenden Sie dazu einen Zähler, welchen Sie in den privaten Preferences der `MainActivity` persistent speichern. Geben Sie den aktuellen Zählerstand in einer `TextView` auf dem Bildschirm der Activity an, siehe Screenshot rechts oben. Wenn Sie die App beenden und wieder starten, muss der Counter erhalten bleiben.

3. Tee-Präferenzen

Beim Klick auf den Knopf „Tee-Preferences editieren“ soll ein entsprechender Einstellungs-Bildschirm gemäss Screenshot rechts erscheinen, auf welchem die Tee-Präferenzen erfasst werden können. Verwenden Sie dazu, wie in der Vorlesung gezeigt, eine Ressourcen-Datei `preferences.xml`, in welcher diese Einstellungen spezifiziert sind. Erstellen Sie dazu weiter eine neue Activity `AppPreferenceActivity`, welche eine innere Klasse hat, die von der Klasse `PreferenceFragment` erbt und verwendet wird, um die gewünschten Präferenzen anzuzeigen.



Hinweis: Der gewünschte Süssstoff soll nur abgefragt werden, wenn der Tee überhaupt gesüsst werden soll. Legen Sie für diese Aufgabe ebenfalls die dazu notwendigen String-Arrays in `arrays.xml` an. Die gesetzten Präferenzen sollen beim Verlassen der `AppPreferenceActivity` auf der `MainActivity` aktualisiert werden.

Ich trinke am liebsten Lipton/Pfefferminztee, mit Rohrzucker gesüsst.

4. Tee-Default-Präferenzen

Implementieren Sie einen Knopf, mit welchem ein vordefinierter Satz Tee-Präferenzen im App-Code gesetzt und danach entsprechend angezeigt werden.

5. Dateisystem: Text speichern / laden

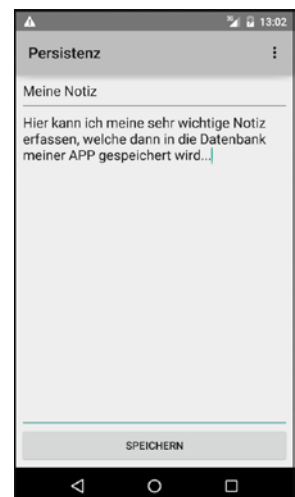
Fügen Sie ihrer App ein Textfeld zum Erfassen und zwei Knöpfe zum Speichern und Laden eines Texts hinzu. Der erfasste Text soll dabei wahlweise intern oder extern (SD-Karte) gespeichert werden. Verwenden Sie immer `MODE_PUBLIC`, wenn sie auf die SD-Karte speichern, damit die Permissions sicher geprüft werden. Statusmeldungen (Speichern ok, Fehler beim Einlesen, usw.) sollen dabei direkt im Textfeld angezeigt werden.

Für das lesen und speichern auf der externen SD-Karte müssen Sie vor der Operation immer die vorhandenen Permissions prüfen und ggf. Permissions beim Benutzer beantragen!

6. [OPTIONAL] SQL Persistenz mit Room: Notiz erfassen & anzeigen

Durch Klick auf „Notiz erzeugen“ soll eine Notiz mit Titel und Text erfasst werden können. Erstellen Sie dazu eine neue Activity `NoteActivity`, welche einerseits verwendet wird, um neue Notizen zu erfassen und andererseits, um bestehende Notizen anzuzeigen. Diese Unterscheidung geschieht mittels eines Long-Parameters `noteId`, welche dem entsprechenden Intent mitgegeben wird. Ist `noteId < 0`, wird ein Formular zum Erfassen einer neuen Notiz angezeigt, sonst die bestehende Notiz mit dieser ID angezeigt. Verwenden Sie dazu zwei verschiedene Layout-Dateien `activity_show_note.xml` und `activity_edit_note.xml`. Bei Klick auf „Notizen zeigen“ soll eine Liste mit allen verfügbaren Notiz-Titeln angezeigt werden (`NoteListActivity` extends `ListActivity`, inkl.

Verwendung eines selber definierten `NotesAdapter` und einer Datei `item_layout.xml` zur Spezifikation des Layouts eines List-Items). Wird eine Notiz angeklickt, wird diese in der der `NoteActivity` (ohne Speichern-Knopf) angezeigt. Hinweise: Erstellen Sie dazu wie in der Vorlesung gezeigt je eine Klasse `Note` (Entity), ein Interface `NoteDao` (Data Access Object) und eine abstrakte Klasse `NotesDatabase`. Für diese Übung ist es zulässig, DB-Abfragen im Main Thread auszuführen.



7. [OPTIONAL] Externen Content Provider nutzen: SMS anzeigen

Durch Klick auf „SMS anzeigen“ sollen alle zurzeit in der Inbox vorhandenen SMS via Content Provider in einem AlertDialog angezeigt werden. Greifen Sie dazu mittels der URI-Konstanten `Telephony.SMS.Inbox.CONTENT_URI` auf den entsprechenden Provider zu.

Hinweis: Vergessen Sie nicht, im Manifest die entsprechenden Permissions einzutragen und beim Benutzer zu erfragen!

Tipp: Sie können dem Emulator SMS schicken, indem sie auf der Emulator-Toolbar ... > Phone > SMS Message > Send Message benutzen

