Adaptive User Interface Ausarbeitung

Alexander Halbarth, 5CHIF, 21.11.2014

Inhalt

Einführung

In dieser Ausarbeitung wird die Möglichkeit der dynamischen Anpassung einer Navigation Drawer Activity erklärt. Es wird eine Hitliste angelegt, welche automatisch sortiert wird, das meistverwendet Element befindet sich an der ersten Position. Diese Lösung ist völlig dynamisch und kann leicht mit weiteren Fragments erweitert werden. Es müssen dafür nur die Fragments geschrieben werden und in das XML File hinzugefügt werden. Die Hitlisten sind an User gebunden welche bei der Verwendung einfach gewechselt und angelegt werden können. Die Applikation kann weiters die Sprache für sich selbst ändern und speichert alle Informationen beim Beenden. Im folgenden wird der Source Code ausführlich erklärt. Somit bietet diese Applikation eine automatische Personalisierung für jeden User an. Dieser Code kann sehr einfach in andere Projekte integriert werden. Die Hit Listen Daten können in einem weiteren Schritt auch ausgewertet werden.

Das Gesamte Projekt kann unter angesehen github.com/alexhalbi/AdaptveUI und heruntergeladen werden.

Source Code erklärt

NavigationDrawerFragment.java

Die hinzugefügte Methode erstellt alle Elemente im Navigation Drawer neu. Sie wird verwendet um die Beschriftungen zu aktualisieren. Die Reihenfolge des Parameter Arrays wird übernommen.

```
public void updateElements(String[] items) {
      mDrawerListView.setAdapter(new ArrayAdapter<String>(

   getActionBar().getThemedContext(),
         android.R.layout.simple_list_item_1,
         android.R.id.text1,
         items));
}
```

Der restliche Code dieser Klasse ist genauso wie von Eclipse generiert.

FavoriteMap

Das Programm verwendet HashMaps die Hits (pro User) und Namen der Fragments zu speichern.

Die Maps sind folgendermaßen aufgebaut:

```
private Map<String username, Map<Class fragment, Integer hitCounter>> user;
private Map<Class fragment, String fragmentName> names;
private Class[] fragments;
```

Das Class Array ist gefüllt mit allen Fragments so wie sie aus dem Konfigurations-XML eingelesen werden. Es wird dafür verwendet neue User in den Hashmaps anzulegen um Daten eintragen zu können. Dieses Array wird im Konstruktor aus einem String Array mit dem Classpath erzeugt. Der Code erzeugt aus jedem Classpath ein Class Objekt und speichert dieses dann. Wenn das Class Objekt nicht erzeugt werden kann wird dies im Log File vermerkt und der Eintrag übersprungen.

```
public FavoriteMap(String[] fragments, String[] fragmentNames) {
      this.fragments = new Class[fragments.length];
      for (int i = 0; i < fragments.length; i++) {
            try {
                  this.fragments[i] = Class.forName(fragments[i]);
            } catch (ClassNotFoundException e) {
                  Log.e("FragmentMap", "Fragment " + fragments[i]
                              + " not found, skipped", e);
            }
      }
      names = new HashMap<Class, String>();
      updateNames(fragments, fragmentNames);
}
```

Weiters wird auch die oben erwähnte names HashMap instanziert und die Methode updateNames aufgerufen welche Werte in names einträgt.

```
public void updateNames(String[] fragments, String[] fragmentNames) {
	if (fragments.length != fragmentNames.length) {
		throw new InvalidParameterException(
				"Resource Name and Fragment Length do not match!");
	}
	for (int i = 0; i < fragments.length; i++) {
		try {
			this.names.put(Class
			  .forName(fragments[i]), fragmentNames[i]);
		} catch (ClassNotFoundException e) {
			Log.e("FragmentNameMap", "Fragment " + fragments[i]
					+ " not found, skipped", e);
		}
	}
}
```

Diese Methode überprüft zuerst ob die Array Größen zusammenpassen und erzeugt danach für jeden Namen einen Eintrag in der Hashmap. Diese Methode muss bei jeder Namensänderung aufgerufen werden, da keine Ressourcewerte gespeichert werden können. Dies liegt daran, dass ein Objekt nicht serialisiert werden kann wenn es den Application Context als lokale Variable hat. Deswegen wird diese Methode bei jedem onCreate aufgerufen um inkonsistente Daten zu vermeiden.

Um den Namen eines Fragments zu bekommen gibt es mehrere Methoden:

```
public String getNameFrgmt(Fragment frgmnt) {
	return names.get(frgmnt.getClass());
}

public String getNamePos(String user, int position) {
	return getNameFrgmt(getPosition(user, position, false));
}

public String[] getSortedNames(String user) {
	String[] list = new String[names.size()];
	for (int i = 0; list.length > i; i++) {
		list[i] = getNamePos(user, i);
	}
	return list;
}
```

Es ist möglich mit dem Fragment Objekt den Namen zu bekommen. Mit dem User Namen und der Position im Drawer und es ist auch möglich ein sortiertes Array aller Fragments zu bekommen. Dieses Array wird dem Navigation Drawer 1:1 übergeben und ist für die Darstellung der Menüeinträge verantwortlich.

```
private Fragment getPosition(String user, int position, boolean incr) {
	Log.d("FavoriteMap", "getPosition" + position + '(' + user + ')');
	Map<Class, Integer> frgmntCnt = null;
	if (this.user != null) {
		frgmntCnt = this.user.get(user);
	} else {
		this.user = new HashMap<String, Map<Class, Integer>>();
	}

	if (frgmntCnt == null) {
		frgmntCnt = new HashMap<Class, Integer>(fragments.length);
		for (Class c : fragments) {
			frgmntCnt.put(c, 0);
		}
	}

	FragmentComparator bvc = new FragmentComparator(frgmntCnt);
	TreeMap<Class, Integer> sorted_map =
		new TreeMap<Class, Integer>(bvc);
	sorted_map.putAll(frgmntCnt);

	Entry<Class, Integer> entr = getNEntrie(position, sorted_map);

	if (incr) {
		frgmntCnt.put(entr.getKey(), entr.getValue() + 1);
	}

	this.user.put(user, frgmntCnt);

	try {
		Log.d("FavoriteMap", "newInstance");
		Fragment f = (Fragment) entr.getKey().newInstance();
		Log.d("FavoriteMap", "newInstance Class=" +
			f.getClass().getName());
		return f;
	} catch (IllegalAccessException e) {
		Log.e("FavoriteMap", "newInstance", e);
	} catch (IllegalArgumentException e) {
		Log.e("FavoriteMap", "newInstance", e);
	} catch (InstantiationException e) {
		Log.e("FavoriteMap", "newInstance", e);
	}
	return null;
}
```

Diese Methode gibt einem die für den User und die Position passende Fragment zurück. Es sucht zuerst ob dieser User bereits in der Hashmap angelegt ist, wenn nicht dann wird er angelegt und eine HashMap mit Einträgen für jedes Fragment wird hinzugefügt. Danach wird die HashMap mit einem Comperator sortiert (Quelle: http://stackoverflow.com/questions/109383/how-to-sort-a-mapkey-value-on-the-values-in-java, 18.11.2014). Danach wird die Mathode getNEntrie aufgerufen die aus einer HashMap den n. Eintrag ausgeben kann (Quelle: http://stackoverflow.com/questions/1509391/how-to-get-the-one-entry-from-hashmap-without-iterating, 18.11.2014). Danach wird, wenn das Inkrementier Bit 1 ist, der Couter erhöht. Darauf wird der Eintrag in der globalen HashMap mit dem User als Key gespeichert.

Rückgegeben wird eine neue Instanz des Fragments. Dieses wird aus dem Class Objekt mit der newInstance Methode erzeugt. Diese setzt voraus, dass es einen Konstruktor ohne Attribute gibt. Ist dies nicht der Fall kann das Fragment nicht geöffnet werden.

Die getPosition Methode ist absichtlich privat, da von außen kein Fragment aufgerufen werden darf, dass den Count nicht erhöht. Dafür gibt es eine Überladung der Methode ohne incr Attribut welche public ist und die getPosition Methode mit incr = false aufruft.

Dies ist die Methode, welche aus einer HashMap den Eintrag auf einer bestimmten Position ausgeben kann:

```
public static Map.Entry<Class, Integer> getNEntrie(int n,
                  Map<Class, Integer> source) {
      int count = 0;
      for (Map.Entry<Class, Integer> entry : source.entrySet()) {
            if (count == n) {
                  return entry;
            }
            count++;
      }
      return null;
}
```

Dies ist der Comperator, welcher die HashMap nach dem HitCounter sortiert:

```
@SuppressWarnings("rawtypes")
class FragmentComparator implements Comparator<Class> {

      Map<Class, Integer> base;

      public FragmentComparator(Map<Class, Integer> frgmntCnt) {
            this.base = frgmntCnt;
      }

      @Override
      public int compare(Class a, Class b) {
            if (base.get(a) >= base.get(b)) {
                  return -1;
            } else {
                  return 1;
            }
      }
}
```

Fragments.xml

Dieses XML File ist dafür da die verwendeten Fragments mit Namen zu versehen und alle Fragments aufzulisten welche über den Navigation Drawer erreichbar sein sollen. Diese Daten werden von der Klasse FavoriteMap benötigt. Um ein Fragment in den Navigation Drawer hinzuzufügen muss nur ein weiteres Element in beiden Listen erstellt werden. Den Rest erledigt der Code.

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This Resource File is for the Fragment Configuration of the
Application. -->
<resources>

    <!-- Here must be all Fragments which should appear in the Navigation
Drawer. -->
    <string-array name="fragments">
        <item>at.halbarth.alexander.adaptiveui.fragments.ExamplePic</item>
        <item>at.halbarth.alexander.adaptiveui.fragments.ExampleTxt</item>
        <item>at.halbarth.alexander.adaptiveui.fragments.LinksWIFI</item>

<item>at.halbarth.alexander.adaptiveui.fragments.ChangeLocale</item>

<item>at.halbarth.alexander.adaptiveui.fragments.ExampleEditTxt</item>
    </string-array>
    <!-- For every Fragment above must be a Name. -->
    <string-array name="fragment_names">
        <item>@string/title_section1</item>
        <item>@string/title_section2</item>
        <item>@string/title_section3</item>
        <item>@string/title_section4</item>
        <item>@string/title_section5</item>
    </string-array>

</resources>
```

Im ersten String Array werden die Class Paths aller Fragments aufgelistet. Im zweiten String Array befinden sich Referezen zu den Namens Strings der Fragments. Es ist nicht notwendig die Namen der Fagments zu Referenzieren, sie können auch statisch eingetragen werden, jedoch verkompliziert es dies wenn der Selbe String noch an anderen Stellen benutzt werden soll.

Die hier eingetragenen Namen werden automatisch im Navigation Drawer und in der Application Bar eingetragen. Der Code dafür befindet sich in der MainActivity.

User.xml

Diese XML File ist beinhaltet eine Auflistung aller Dafault Users. Es wird dafür benötigt, wenn die Applikation zum 1. Mal gestartet wird um User automatisch hinzuzufügen. Es bietet auch eine Fallback Lösung an falls die SharedPreferences nicht mehr gelesen werden können. Es ist jedoch aus dem Code heraus nicht möglich in dieses XML File weitere user einzutragen. Aufgrund dessen wurde der Code geändert um die user in den SharedPreferences zu speichern.

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This Resource File is for the Default Users of the Application.
If SharedPref cannot be read this UserList will be used instead.
-->
<resources>

    <string-array name="default_user_array">
        <item>Red</item>
        <item>Green</item>
        <item>Blue</item>
    </string-array>

</resources>
```

Jedes Element in diesem String Array ist ein User der eine Hitliste hat und der ausgewählt werden kann.

Menü: Main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="at.halbarth.alexander.adaptiveui.MainActivity" >

    <item
        android:id="@+id/user_indicator"
        android:enabled="false"
        android:title="User Indicator"
        app:showAsAction="withText|ifRoom"/>
    <item
        android:id="@+id/change_user"
        android:orderInCategory="1"
        android:title="@string/change_user"
        app:showAsAction="never"/>

</menu>
```

Dieses XML ist dafür zuständig die Elemente in das Menü einzutragen.

Der 1. Eintrag ist der User Indicator: dieser Zeigt, in der ActionBar, an welcher User derzeit eingeloggt ist. Der andere Eintrag ist der Button um den User zu wechseln, dieser ist nur über den Menü Button (⋮) erreichbar.

MainActivity.java

Die MainActivity ist die Haupt-Activity welche die Fragments beinhaltet und die User wechseln kann.

Sie speichert auch alle erforderlichen Objekte beim Beenden und kann diese beim Starten wieder auslesen, damit die Daten nicht verloren gehen.

```
private static final String USERS = "userList";
private static final String USERLOGGEDIN = "userLoggedIn";
private static final String FAVORITEMAP = "favoriteMap";
private static final String DEF_STRING_ERR = "DEF_STRING_ERR";

private String[] users;
private int userLoggedIn;
private FavoriteMap favoriteMap;
private Locale locale = null;
```

Die Konstanten werden verwendet um alle Elemente die gespeichert werden mit deren Speicherplatz zu versehen. Dies vereinfacht den Aufruf von Methoden und verhindert Tippfehler. Konstanten zu verwenden um Speicher in den SharedPreferences zu adressieren zu verwenden ist Best-Practice.

Das users Array ist ein String Array mit allen Usern die existieren und der Integer userLoggedIn zeigt an welcher User derzeit eingeloggt ist. Die favoriteMap ist eine Instanz der Klasse favoriteMap welche benötigt wird um die Hits auf die Fragments im Navigation Drawer zu zählen.

Das locale Attribut spiegelt die derzeit eingestellte Sprache der Applikation wieder.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
	super.onCreate(savedInstanceState);
	SharedPreferences settings = PreferenceManager
			.getDefaultSharedPreferences(this);
	loadUsers();

	try {
		favoriteMap = getFavoriteMap();
		favoriteMap.updateNames(
			getResources().getStringArray(R.array.fragments),
			getResources().getStringArray(R.array.fragment_names));
	} catch (Exception e) {
		Log.e("FavoriteMap", "NewFavoriteMap", e);
	}

	if (favoriteMap == null) {
		Log.d("FavoriteMap", "NewFavoriteMap");
		favoriteMap = new FavoriteMap(getResources().getStringArray(
				R.array.fragments), getResources().getStringArray(
				R.array.fragment_names));
	}

	setContentView(R.layout.activity_main);
	mTitle = getTitle();

	createNavigationDrawer();
	Configuration config = getBaseContext().getResources()
				.getConfiguration();

	String lang = settings.getString(getString(R.string.pref_locale),"");
	if (!"".equals(lang) && !config.locale.getLanguage().equals(lang)) {
		locale = new Locale(lang);
		Locale.setDefault(locale);
		config.locale = locale;
		getBaseContext().getResources().updateConfiguration(config,
			getBaseContext().getResources().getDisplayMetrics());
	}
	mNavigationDrawerFragment.updateElements(favoriteMap
		.getSortedNames(users[userLoggedIn]));
}
```

Als Erstes wird die zuletzt eingestellte Sprache aus den SharedPreferences ausgelesen und wider gesetzt. Zunächst wird das Users Array und der zuletzt eingeloggt User mit der Methode loadUsers ausgelesen, denn beim erstellen des Inhalts der Activity wird der Username des derzeit eingeloggten User benötigt.

Danach wird versucht die favoriteMap mit der getFavoriteMap Methode einzulesen und die Namen in der Methode zu aktualisieren. Wenn dies nicht gelingt (Es gibt sehr viele Exceptions die entstehen können in dieser Methode) dann wird eine neue FavoriteMap angelegt, da davon auszugehen ist, dass die Map noch nicht existiert hat oder die gespeicherten Informationen nicht mehr einlesbar sind. Dies wird alles im Log ausgegeben.

Danach wird der Content der Activity eingelesen und der Navigation Drawer erzeugt. Danach wird die die Methode updateElements des navigation Drawers wird danach aufgerufen um die Elemente aus der Favorite map in der Reihenfolge der Präferenzen des Users einzufügen.

```
@Override
public void onNavigationDrawerItemSelected(int position) {
	// update the main content by replacing fragments
	FragmentManager fragmentManager = getSupportFragmentManager();
	Fragment frgmt = favoriteMap.getPosition(users[userLoggedIn],
			position);
	Log.d("FavoriteMap", "frgmt=" + frgmt.toString());
	fragmentManager.beginTransaction().replace(R.id.container, frgmt)
			.commit();
	if (mNavigationDrawerFragment != null)
		mNavigationDrawerFragment.updateElements(favoriteMap
				.getSortedNames(users[userLoggedIn]));
}

public void onSectionAttached(int number) {
	mTitle = favoriteMap.getNamePos(users[userLoggedIn], number);
}
```

Diese Beiden Methoden werden von der navigation Drawer Activity verwendet um Fragments zu öffnen wenn auf einen Menüeintrag geklickt wird und dann den Titel der Activity zu ändern.

Der Code wurde verändert um das richtige fragment aus der favoriteMap zu verwenden welches für den user an der gewählten Stelle ist. Danach werden, falls das NavigationDrawerFragment bereits erstellt ist, dies ist beim ersten Aufruf der Mathode in der OnCreate nicht der Fall, alle Elemente im Navigation Drawer neu eingelesen, weil sich die Reihenfolge geändert haben könnte.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
	if (!mNavigationDrawerFragment.isDrawerOpen()) {
		// Only show items in the action bar relevant to this screen
		// if the drawer is not showing. Otherwise, let the drawer
		// decide what to show in the action bar.
		getMenuInflater().inflate(R.menu.main, menu);
		restoreActionBar();
		try {
			MenuItem user_indicator = menu.getItem(0);
			for (int i = 1; user_indicator != null; i++) {
				if (user_indicator.getItemId() ==
						R.id.user_indicator) {
					user_indicator.setTitle(users[userLoggedIn]);
					break;
				}
				user_indicator = menu.getItem(i);
			}
		} catch (IndexOutOfBoundsException e) {
			Log.d("Exeption in onCreateOptionsMenu",
				"IndexOutOfBounds", e);
		}
		return true;
	}
	return super.onCreateOptionsMenu(menu);
}
```

Dieser Code kümmert sich um das Setzen des Users in der Application Bar. Er sucht das user_indicator Element und setzt es danach auf den Namen des derzeit eingeloggten Users.

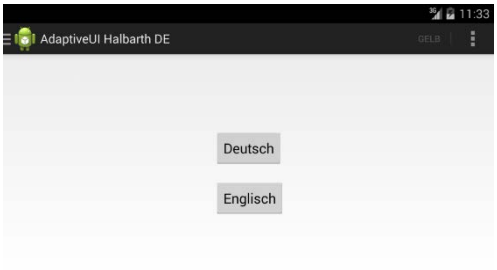
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
	// Handle action bar item clicks here. The action bar will
	// automatically handle clicks on the Home/Up button, so long
	// as you specify a parent activity in AndroidManifest.xml.
	int id = item.getItemId();
	if (id == R.id.change_user) {
		changeUser();
	}
	return super.onOptionsItemSelected(item);
}
```

Diese Methode ruft bei einem Klick auf den Change User Button automatisch die changeUser Methode auf.

Die folgende changeUser Methode öffnet ein Dialogfenster um einen user auszuwählen und loggt automatisch diesen User ein. Der Userliste die ausgegeben wird wird am Ende ein eintrag hinzugefügt um einen neuen User zu erstellen. Wird dieser ausgewählt wird die newUser Methode aufgerufen, diese bittet einen um die Eingabe eines Benutzernamens und bietet auch einen Abbrechen Button an. Wird ein Username eingegeben und die Eingabe bestätigt wird dieser User automatisch eingeloggt. (Quelle: http://www.androidsnippets.com/prompt-user-input-with-an-alertdialog, 21.11.2014)

```
public void changeUser() {
	String[] u = new String[users.length + 1];
	for (int i = 0; i < users.length; i++) {
		u[i] = users[i];
	}
	u[users.length] = getString(R.string.action_create_user);

	AlertDialog.Builder builder = new AlertDialog.Builder(this);
	builder.setTitle(R.string.choose_user);
	builder.setItems(u, new DialogInterface.OnClickListener() {
		@SuppressLint("NewApi")
		@Override
		public void onClick(DialogInterface dialog, int id) {
			if (id == users.length) {
				newUser();
			} else {
				changeUser(id);
			}
		}
	});
	builder.show();
}

@SuppressLint("NewApi")
public void changeUser(int userId) {
	userLoggedIn = userId;
	invalidateOptionsMenu();
	onNavigationDrawerItemSelected(0);
}

public void newUser() {
	final EditText input = new EditText(this);

	new AlertDialog.Builder(this)
	    .setTitle(R.string.create_user)
	    .setView(input)
	    .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
		public void onClick(DialogInterface dialog, int whichButton) {
			String[] u = new String[users.length + 1];
			for (int i = 0; i < users.length; i++) {
				u[i] = users[i];
			}
			u[users.length] = input.getText().toString();
			users = u;
			saveUsers();
			changeUser(users.length - 1);
			}
		})
		.setNegativeButton("Cancel",
			new DialogInterface.OnClickListener() {
				public void onClick(DialogInterface dialog,
					int whichButton) {
					// Do nothing.
				}
	}).show();
}
```

Locale in der Applikation ändern



Diese Methoden werden benötigt um die Sprache in der Applikation zu ändern. Die Methoden localeGER und localeENG sind OnClickListener der Buttons auf der Change Locle Seite in der Appplikation. In der on ConfigurationChanged Methode wird die Locale gesetzt.

```
@Override
public void onConfigurationChanged(Configuration newConfig) {
	super.onConfigurationChanged(newConfig);
	if (locale != null) {
		newConfig.locale = locale;
		Locale.setDefault(locale);
		getBaseContext().getResources().updateConfiguration(newConfig,
			getBaseContext().getResources().getDisplayMetrics());
	}
}

public void localeGER(View v) {
	Locale locale = new Locale("de");
	Locale.setDefault(locale);
	Configuration config = new Configuration();
	config.locale = locale;
	getBaseContext().getResources().updateConfiguration(config,
			getBaseContext().getResources().getDisplayMetrics());
	createNavigationDrawer();
	Toast.makeText(getApplicationContext(), getString(R.string.restart),
			Toast.LENGTH_LONG).show();
}

public void localeENG(View v) {
	Locale locale = new Locale("en");
	Locale.setDefault(locale);
	Configuration config = new Configuration();
	config.locale = locale;
	getBaseContext().getResources().updateConfiguration(config,
			getBaseContext().getResources().getDisplayMetrics());
	createNavigationDrawer();
	Toast.makeText(getApplicationContext(), getString(R.string.restart),
			Toast.LENGTH_LONG).show();
}
```

(Quelle: http://manojprasaddevelopers.blogspot.co.at/2014/04/changing-locale-programmatically.html, 02.11.2014)

(Quelle: http://stackoverflow.com/questions/8049207/how-to-refresh-activity-after-changing-language-locale-inside-application, 02.11.2014)

Speichern und Auslesen

In der Applikation werden verschiedene Arten verwendet um Einstellungen und auch Objekte zu speichern. Eine davon ist die Folgende. Das FavoriteMap Objekt wird serialisiert und in eine Datei gespeichert um die Hitlisten beim nächsten öffnen der Applikation nicht zu verlieren. Zum einlesen wird die Datei einfach geöffnet und das Objekt deserialisiert. Um Dateien schreiben und lesen zu können ohne weiteren Berechtigungen wird die Methode openFileOutput und openFileInput verwendet (Quelle: http://developer.android.com/reference/android/content/Context.html#openFileOutput(java.lang.String, int), 18.11.2014)

```
public void saveFavoriteMap(FavoriteMap favoriteMap) {
	Log.d("saveFavoriteMap", "saveFavoriteMap");
	ObjectOutputStream oos = null;
	FileOutputStream fout = null;
	try {
		fout = openFileOutput(FAVORITEMAP, Context.MODE_PRIVATE);
		oos = new ObjectOutputStream(fout);
		oos.writeObject(favoriteMap);
		Log.d("saveFavoriteMap", "written=" + favoriteMap.toString());
	} catch (Exception e) {
		Log.e("saveFavoriteMap", "saveFavoriteMap", e);
	} finally {
		if (oos != null) {
			try {
				oos.close();
			} catch (IOException e) {
				Log.e("saveFavoriteMap", "saveFavoriteMap", e);
			}
		}
	}
}

public FavoriteMap getFavoriteMap() {
	ObjectInputStream objectinputstream = null;
	FavoriteMap favoriteMap = null;
	try {
		FileInputStream streamIn = openFileInput("favoriteMap");
		objectinputstream = new ObjectInputStream(streamIn);
		favoriteMap = (FavoriteMap) objectinputstream.readObject();
		Log.d("FavoriteMap", "read=" + favoriteMap.toString());

	} catch (Exception e) {
		Log.e("FavoriteMap", "getFavoriteMap", e);
	} finally {
		if (objectinputstream != null) {
			try {
				objectinputstream.close();
			} catch (IOException e) {
				Log.e("FavoriteMap", "getFavoriteMap", e);
			}
		}
	}
	return favoriteMap;
}
```

Um die User Liste und auch den derzeit eingeloggten User zu speichern werden die Daten in die SharedPreferences geschrieben. Um das Array zu schreiben wird es in eine durch Komma getrennte Form gebracht und um es zu lesen wird es an den Kommas wieder geteilt. Wenn das User Array falsch gelesen wird oder noch nicht existiert wird das User Array aus dem user.xml eingelesen.

```
private void saveUsers() {
	SharedPreferences settings = PreferenceManager
			.getDefaultSharedPreferences(this);
	Editor e = settings.edit();
	e.putInt(USERLOGGEDIN, userLoggedIn);

	StringBuilder sb = new StringBuilder();
	for (String u : users) {
		sb.append(u).append(",");
	}
	e.putString(USERS, sb.toString());

	e.commit();
}

@SuppressLint("NewApi")
private void loadUsers() {
	SharedPreferences settings = PreferenceManager
			.getDefaultSharedPreferences(this);
	userLoggedIn = settings.getInt(USERLOGGEDIN, 0);

	String s = settings.getString(USERS, DEF_STRING_ERR);
	if (s == null || s.trim().equalsIgnoreCase("") || s.isEmpty())
		s = DEF_STRING_ERR;

	if (!s.equals(DEF_STRING_ERR)) {
		users = s.split(",");
	} else {
		Log.d("loadUsers", "loading Default Users");
		users = getResources()
			.getStringArray(R.array.default_user_array);
	}
}
```

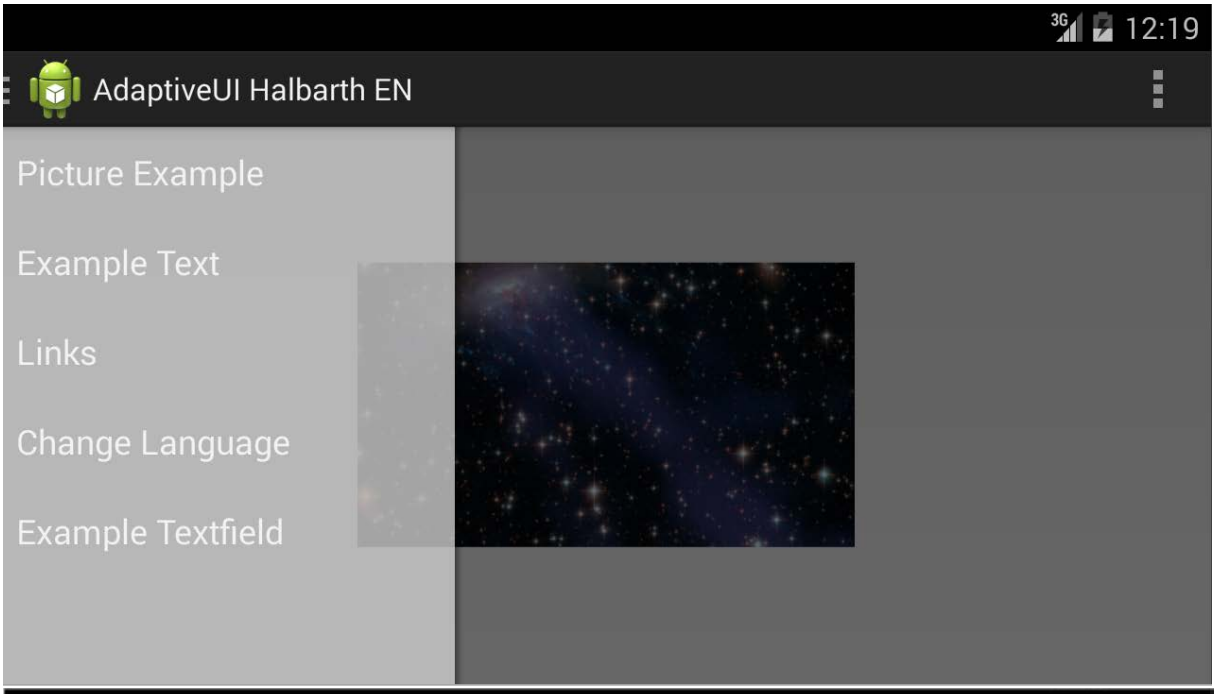
Gespeichert wird in diesen beiden Methoden und an wichtigen Stellen im Code.

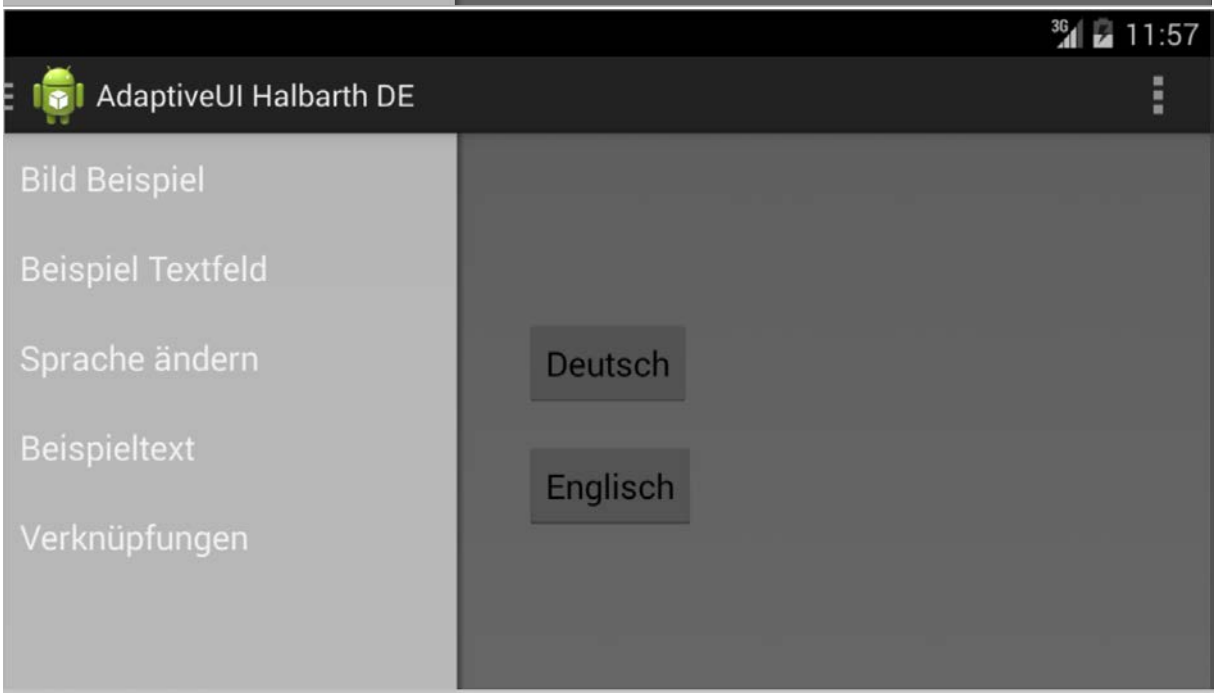
```
@Override
protected void onSaveInstanceState(Bundle outState) {
	saveFavoriteMap(favoriteMap);
	saveUsers();
	super.onSaveInstanceState(outState);
}

@Override
protected void onPause() {
	saveFavoriteMap(favoriteMap);
	saveUsers();
	super.onPause();
}
```

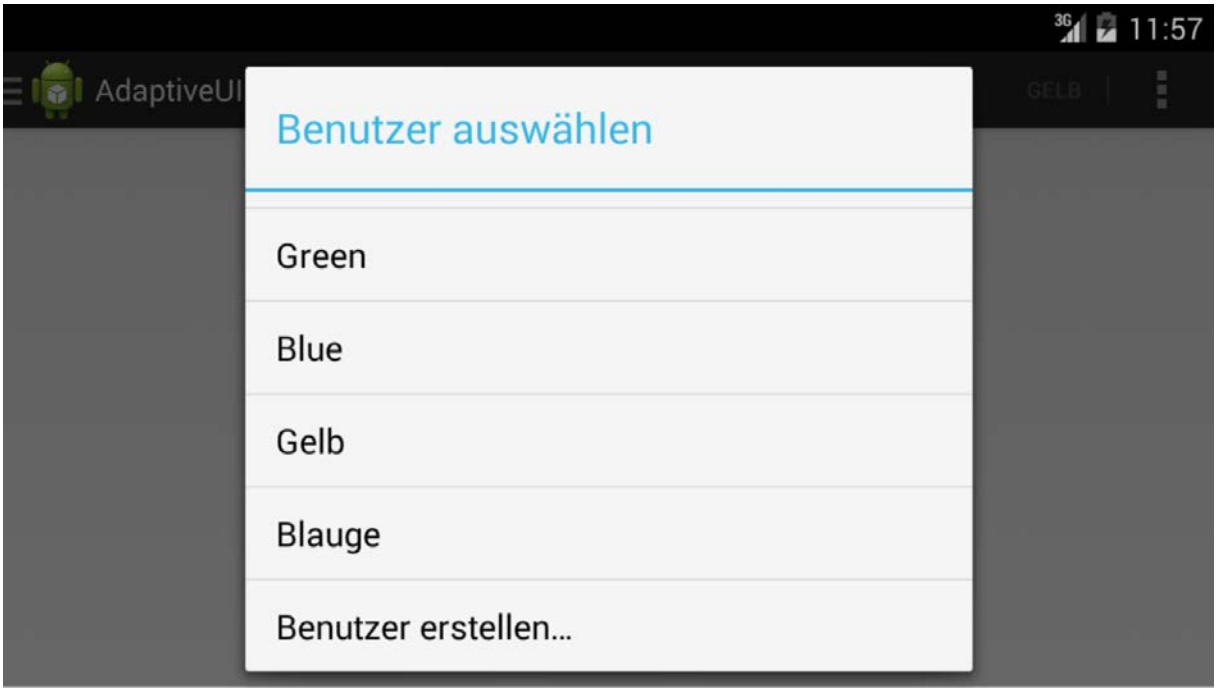
Einschränkungen dieser Lösung

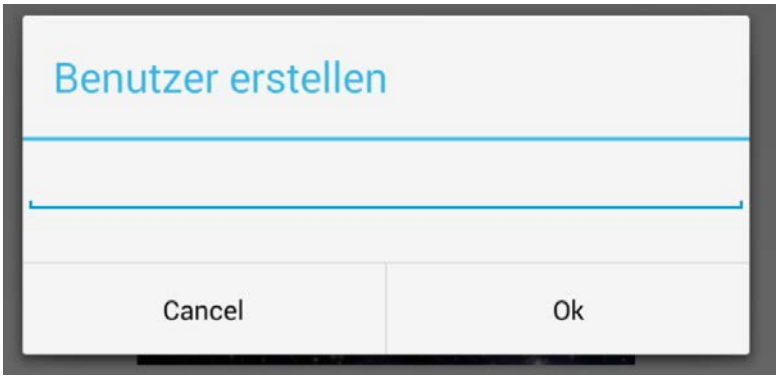
- Alle Fragments müssen in einem XML File eingetragen werden und können nicht automatisch hinzugefügt werden
- Es können nur Fragments verwendet werden, welche einen Konstruktor ohne Attribute haben.

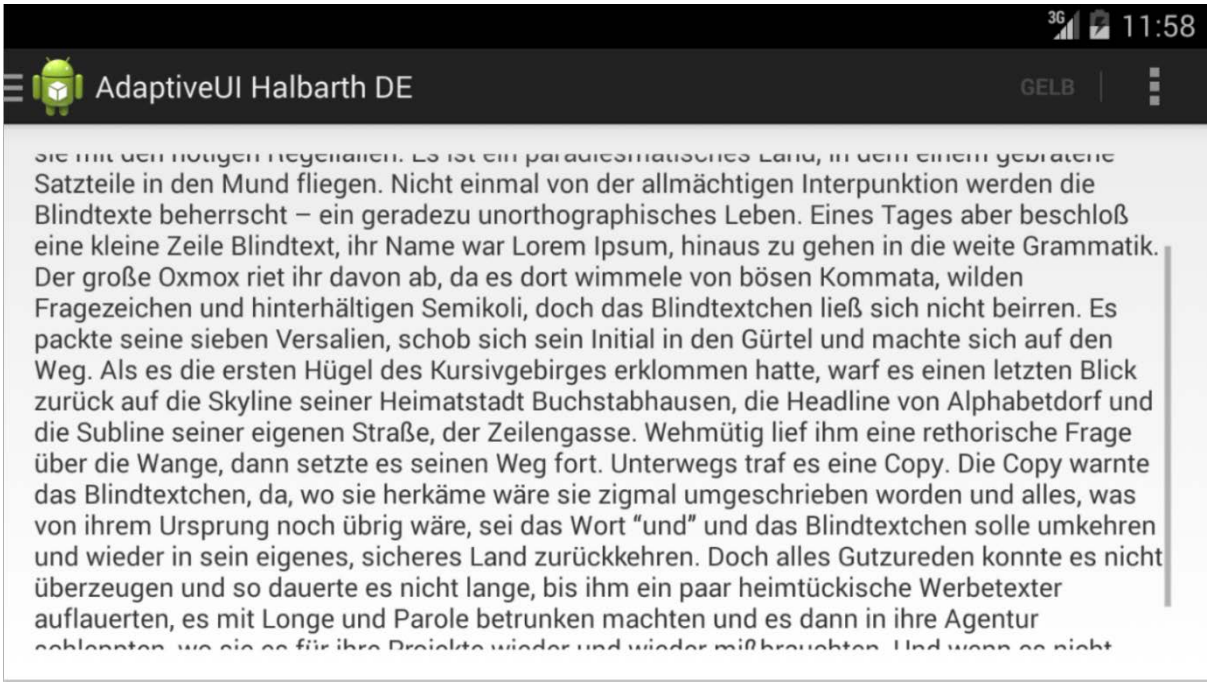
Screenshots

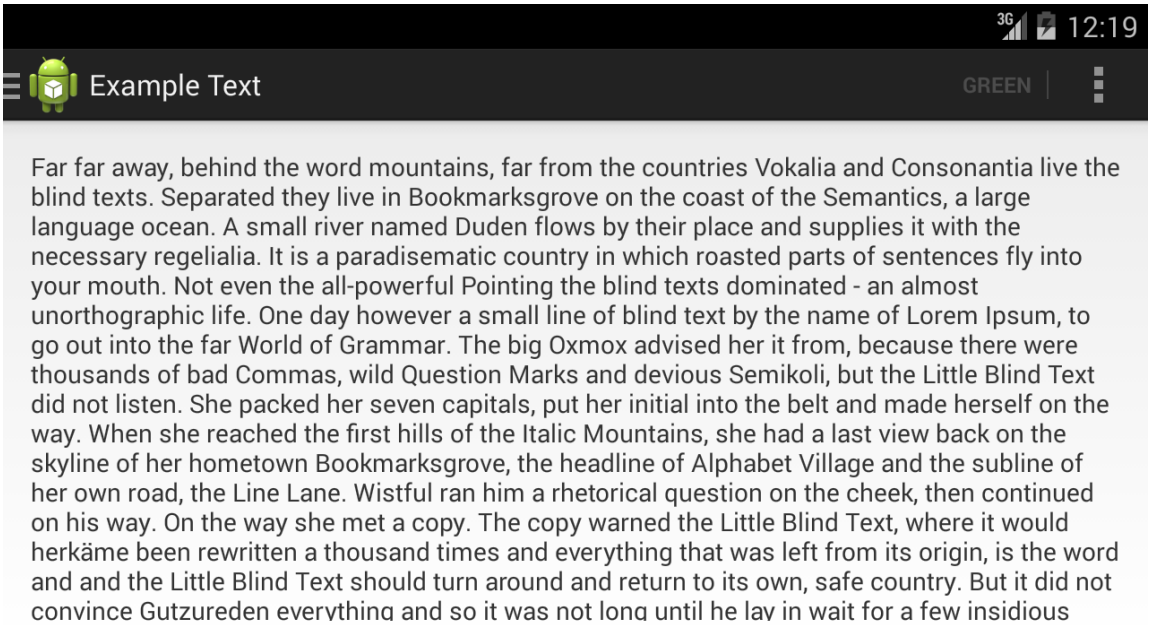





11:57
AdaptiveUI
GELB
Benutzer auswählen
Green
Blue
Gelb
Blauge
Benutzer erstellen...


Benutzer erstellen
Cancel
Ok


11:58
AdaptiveUI Halbarth DE
GELB
Satzteile in den Mund fliegen. Nicht einmal von der allmächtigen Interpunktion werden die Blindtexte beherrscht – ein geradezu unorthographisches Leben. Eines Tages aber beschloß eine kleine Zeile Blindtext, ihr Name war Lorem Ipsum, hinaus zu gehen in die weite Grammatik. Der große Oxmox riet ihr davon ab, da es dort wimmele von bösen Kommata, wilden Fragezeichen und hinterhältigen Semikoli, doch das Blindtextchen ließ sich nicht beirren. Es packte seine sieben Versalien, schob sich sein Initial in den Gürtel und machte sich auf den Weg. Als es die ersten Hügel des Kursivgebirges erklommen hatte, warf es einen letzten Blick zurück auf die Skyline seiner Heimatstadt Buchstabhausen, die Headline von Alphabetdorf und die Subline seiner eigenen Straße, der Zeilengasse. Wehmütig lief ihm eine rethorische Frage über die Wange, dann setzte es seinen Weg fort. Unterwegs traf es eine Copy. Die Copy warnte das Blindtextchen, da, wo sie herkäme wäre sie zigmal umgeschrieben worden und alles, was von ihrem Ursprung noch übrig wäre, sei das Wort "und" und das Blindtextchen solle umkehren und wieder in sein eigenes, sicheres Land zurückkehren. Doch alles Gutzureden konnte es nicht überzeugen und so dauerte es nicht lange, bis ihm ein paar heimtückische Werbetexter auflauerten, es mit Longe und Parole betrunken machten und es dann in ihre Agentur


12:19
Example Text
GREEN
Far far away, behind the word mountains, far from the countries Vokalia and Consonantia live the blind texts. Separated they live in Bookmarksgrove on the coast of the Semantics, a large language ocean. A small river named Duden flows by their place and supplies it with the necessary regelialia. It is a paradisematic country in which roasted parts of sentences fly into your mouth. Not even the all-powerful Pointing the blind texts dominated - an almost unorthographic life. One day however a small line of blind text by the name of Lorem Ipsum, to go out into the far World of Grammar. The big Oxmox advised her it from, because there were thousands of bad Commas, wild Question Marks and devious Semikoli, but the Little Blind Text did not listen. She packed her seven capitals, put her initial into the belt and made herself on the way. When she reached the first hills of the Italic Mountains, she had a last view back on the skyline of her hometown Bookmarksgrove, the headline of Alphabet Village and the subline of her own road, the Line Lane. Wistful ran him a rhetorical question on the cheek, then continued on his way. On the way she met a copy. The copy warned the Little Blind Text, where it would herkäme been rewritten a thousand times and everything that was left from its origin, is the word and and the Little Blind Text should turn around and return to its own, safe country. But it did not convince Gutzureden everything and so it was not long until he lay in wait for a few insidious