**ДОДАТОК 4**

Розробка інформаційного та програмного забезпечення підсистеми Електронного Кампусу "Розклад" з підтримкою мобільних платформ. Розробка додатку для Android платформ

Лістинг коду програми

ІК11.05 0414. 05 13

Аркушів 13

Київ – 2015

/\*\*

\* A login screen that offers login via login/password.

\*/

public class LoginActivity extends Activity implements LoaderCallbacks<Cursor> {

public static final String LOGIN\_URL = "http://campus-api.azurewebsites.net/User/Auth";

public static final String CURRENT\_USER\_URL = "http://campus-api.azurewebsites.net/User/GetCurrentUser";

/\*\*

\* A dummy authentication store containing known user names and passwords.

\* TODO: remove after connecting to a real authentication system.

\*/

private static final String[] DUMMY\_CREDENTIALS = new String[]{

"test:test", "bar@example.com:world"

};

JSONParser jsonParser = new JSONParser();

/\*\*

\* Keep track of the login task to ensure we can cancel it if requested.

\*/

private UserLoginTask mAuthTask = null;

// UI references.

private MaterialAutoCompleteTextView mLoginView;

private EditText mPasswordView;

private View mProgressView;

private View mLoginFormView;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_login);

MainActivity.prefs = new ObscuredSharedPreferences(

this, this.getSharedPreferences("LOCAL\_DATA", Context.MODE\_PRIVATE));

// Set up the login form.

mLoginView = (MaterialAutoCompleteTextView) findViewById(R.id.login);

populateAutoComplete();

mPasswordView = (MaterialEditText) findViewById(R.id.password);

mPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener() {

@Override

public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {

Log.v("lol", id + "");

if (id == 6 || id == EditorInfo.IME\_NULL) {

attemptLogin();

return true;

}

return false;

}

});

ButtonFlat mLoginSignInButton = (ButtonFlat) findViewById(R.id.login\_sign\_in\_button);

mLoginSignInButton.setOnClickListener(new OnClickListener() {

@Override

public void onClick(View view) {

attemptLogin();

}

});

mLoginFormView = findViewById(R.id.login\_form);

mProgressView = findViewById(R.id.login\_progress);

if ((MainActivity.prefs.getString("login", null) != null) && (MainActivity.prefs.getString("password", null) != null))

if ((!MainActivity.prefs.getString("login", null).isEmpty()) && (!MainActivity.prefs.getString("password", null).isEmpty())) {

Log.v("lol", MainActivity.prefs.getString("login", null) + MainActivity.prefs.getString("password", null));

showProgress(true);

/\*mAuthTask = new UserLoginTask(MainActivity.prefs.getString("login", null), MainActivity.prefs.getString("password", null));

mAuthTask.execute((Void) null);\*/

mAuthTask = new UserLoginTask(MainActivity.prefs.getString("login", null), MainActivity.prefs.getString("password", null));

mAuthTask.onPostExecute(true);

}

}

private void populateAutoComplete() {

getLoaderManager().initLoader(0, null, this);

}

/\*\*

\* Attempts to sign in or register the account specified by the login form.

\* If there are form errors (invalid login, missing fields, etc.), the

\* errors are presented and no actual login attempt is made.

\*/

public void attemptLogin() {

if (mAuthTask != null) {

return;

}

// Reset errors.

mLoginView.setError(null);

mPasswordView.setError(null);

// Store values at the time of the login attempt.

String login = mLoginView.getText().toString();

String password = mPasswordView.getText().toString();

boolean cancel = false;

View focusView = null;

// Check for a valid password, if the user entered one.

if (!TextUtils.isEmpty(password) && !isPasswordValid(password)) {

mPasswordView.setError(getString(R.string.error\_invalid\_password));

focusView = mPasswordView;

cancel = true;

}

// Check for a valid login address.

if (TextUtils.isEmpty(login)) {

mLoginView.setError(getString(R.string.error\_field\_required));

focusView = mLoginView;

cancel = true;

} else if (!isLoginValid(login)) {

mLoginView.setError(getString(R.string.error\_invalid\_login));

focusView = mLoginView;

cancel = true;

}

if (cancel) {

// There was an error; don't attempt login and focus the first

// form field with an error.

focusView.requestFocus();

} else {

// Show a progress spinner, and kick off a background task to

// perform the user login attempt.

showProgress(true);

mAuthTask = new UserLoginTask(login, password);

mAuthTask.execute((Void) null);

}

}

private boolean isLoginValid(String login) {

//TODO: Replace this with your own logic

return login.length() > 0;

}

private boolean isPasswordValid(String password) {

//TODO: Replace this with your own logic

return password.length() > 0;

}

/\*\*

\* Shows the progress UI and hides the login form.

\*/

@TargetApi(Build.VERSION\_CODES.HONEYCOMB\_MR2)

public void showProgress(final boolean show) {

if (this.getCurrentFocus() != null)

((InputMethodManager) getSystemService(Activity.INPUT\_METHOD\_SERVICE)).hideSoftInputFromWindow(this.getCurrentFocus().getWindowToken(), 0);

// On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow

// for very easy animations. If available, use these APIs to fade-in

// the progress spinner.

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.HONEYCOMB\_MR2) {

int shortAnimTime = getResources().getInteger(android.R.integer.config\_shortAnimTime);

mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);

mLoginFormView.animate().setDuration(shortAnimTime).alpha(

show ? 0 : 1).setListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);

}

});

mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);

mProgressView.animate().setDuration(shortAnimTime).alpha(

show ? 1 : 0).setListener(new AnimatorListenerAdapter() {

@Override

public void onAnimationEnd(Animator animation) {

mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);

}

});

} else {

// The ViewPropertyAnimator APIs are not available, so simply show

// and hide the relevant UI components.

mProgressView.setVisibility(show ? View.VISIBLE : View.GONE);

mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);

}

}

@Override

public Loader<Cursor> onCreateLoader(int i, Bundle bundle) {

return new CursorLoader(this,

// Retrieve data rows for the device user's 'profile' contact.

Uri.withAppendedPath(ContactsContract.Profile.CONTENT\_URI,

ContactsContract.Contacts.Data.CONTENT\_DIRECTORY), ProfileQuery.PROJECTION,

// Select only login addresses.

ContactsContract.Contacts.Data.MIMETYPE +

" = ?", new String[]{ContactsContract.CommonDataKinds.Nickname

.CONTENT\_ITEM\_TYPE},

// Show primary login first. Note that there won't be

// a primary login if the user hasn't specified one.

ContactsContract.Contacts.Data.IS\_PRIMARY + " DESC");

}

@Override

public void onLoadFinished(Loader<Cursor> cursorLoader, Cursor cursor) {

List<String> logins = new ArrayList<String>();

cursor.moveToFirst();

while (!cursor.isAfterLast()) {

logins.add(cursor.getString(ProfileQuery.ADDRESS));

cursor.moveToNext();

}

addLoginsToAutoComplete(logins);

}

@Override

public void onLoaderReset(Loader<Cursor> cursorLoader) {

}

private void addLoginsToAutoComplete(List<String> loginAddressCollection) {

//Create adapter to tell the AutoCompleteTextView what to show in its dropdown list.

ArrayAdapter<String> adapter =

new ArrayAdapter<String>(LoginActivity.this,

android.R.layout.simple\_dropdown\_item\_1line, loginAddressCollection);

mLoginView.setAdapter(adapter);

}

private interface ProfileQuery {

String[] PROJECTION = {

ContactsContract.CommonDataKinds.Nickname.IS\_PRIMARY,

};

int ADDRESS = 0;

int IS\_PRIMARY = 1;

}

/\*\*

\* Represents an asynchronous login/registration task used to authenticate

\* the user.

\*/

public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {

private final String mLogin;

private final String mPassword;

UserLoginTask(String login, String password) {

mLogin = login;

mPassword = password;

}

@Override

protected Boolean doInBackground(Void... params) {

try {

List<NameValuePair> reqString = new ArrayList<NameValuePair>();

reqString.add(new BasicNameValuePair("login", mLogin));

reqString.add(new BasicNameValuePair("password", mPassword));

Log.d("lol", "starting");

// getting product details by making HTTP request

JSONObject json = jsonParser.makeHttpRequest(

LOGIN\_URL, "GET", reqString);

// check your log for json response

Log.d("lol", json.toString());

if (json.getInt("StatusCode") == 200) {

MainActivity.sessionId = json.getString("Data");

reqString.clear();

reqString.add(new BasicNameValuePair("sessionId", MainActivity.sessionId));

JSONObject currentUser = jsonParser.makeHttpRequest(

CURRENT\_USER\_URL, "GET", reqString).getJSONObject("Data");

MainActivity.prefs.edit().putString("login", mLogin).commit();

MainActivity.prefs.edit().putString("password", mPassword).commit();

MainActivity.prefs.edit().putString("userId", currentUser.getString("UserAccountId")).commit();

if (MainActivity.prefs.getString(MainActivity.prefs.getString("userId", null), null) == null) {

MainActivity.prefs.edit().putString(MainActivity.prefs.getString("userId", null), currentUser.toString()).commit();

}

return true;

}

} catch (Exception e) {

Log.d("lol", e.toString());

return false;

}

for (String credential : DUMMY\_CREDENTIALS) {

String[] pieces = credential.split(":");

if (pieces[0].equals(mLogin)) {

// Account exists, return true if the password matches.

return pieces[1].equals(mPassword);

}

}

// TODO: register the new account here.

return false;

}

@Override

protected void onPostExecute(final Boolean success) {

mAuthTask = null;

if (success) {

finish();

LoginActivity.this.startActivity(new Intent(LoginActivity.this, MainActivity.class));

} else {

mPasswordView.setError(getString(R.string.error\_incorrect\_password));

mPasswordView.requestFocus();

}

showProgress(false);

}

@Override

protected void onCancelled() {

mAuthTask = null;

showProgress(false);

}

}

}

public class MainActivity extends ActionBarActivity {

public static Toolbar mToolbar;

public static ActionBarDrawerToggle mDrawerToggle;

public static Fragment currentFragment;

private static DrawerLayout mDrawerLayout;

private static DrawerLayout mInfoDrawerLayout;

private static ActionBarDrawerToggle mInfoDrawerToggle;

private static String sessionId;

private static SharedPreferences prefs;

private static JSONObject currentUser;

private ArrayList<NavMenuItem> menuList;

private RecyclerView mRecyclerView;

private MenuAdapter mAdapter;

private LinearLayoutManager mLayoutManager;

private LinearLayout mInfoDrawerView;

private JSONParser jsonParser = new JSONParser();

private GetSessionIdTask mSessionIdTask;

private View mLogout;

private View mToolbarContainer;

public void showToolbar() {

mToolbarContainer.animate().cancel();

mToolbarContainer.animate().translationY(0).setDuration(100);

//mFragmentContainer.setPadding(0, mToolbar.getHeight(), 0, 0);

}

public void hideToolbar() {

mToolbarContainer.animate().cancel();

mToolbarContainer.animate().translationY(-mToolbarContainer.getHeight()).setDuration(100);

//mFragmentContainer.setPadding(0,0,0,0);

}

public void finishMoveToolbar() {

if (mToolbarContainer.getTranslationY() > -mToolbarContainer.getHeight() / 2) {

showToolbar();

} else {

hideToolbar();

}

}

public void moveToolbar(int dy) {

mToolbarContainer.animate().cancel();

if ((dy > 0) && (mToolbarContainer.getTranslationY() + mToolbarContainer.getHeight() > 0)) {

if (mToolbarContainer.getTranslationY() + mToolbarContainer.getHeight() - dy < 0) {

mToolbarContainer.setTranslationY(-mToolbarContainer.getHeight());

} else {

mToolbarContainer.setTranslationY(mToolbarContainer.getTranslationY() - dy);

}

}

if ((dy < 0) && (mToolbarContainer.getTranslationY() < 0)) {

if (mToolbarContainer.getTranslationY() - dy > 0) {

mToolbarContainer.setTranslationY(0);

} else {

mToolbarContainer.setTranslationY(mToolbarContainer.getTranslationY() - dy);

}

}

//mFragmentContainer.setPadding(0, (int)( mToolbar.getHeight() + mToolbar.getTranslationY()), 0, 0);

// mFragmentContainer.setTranslationY(mToolbar.getTranslationY());

// FrameLayout.LayoutParams lp = (FrameLayout.LayoutParams) mFragmentContainer.getLayoutParams();

// Display display = getWindowManager().getDefaultDisplay();

// Point size = new Point();

// display.getSize(size);

// lp.height = (int) -mToolbar.getTranslationY() + size.y - lp.topMargin;

// mFragmentContainer.requestLayout();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

DisplayImageOptions options = new DisplayImageOptions.Builder()

.cacheOnDisk(true)

.build();

ImageLoaderConfiguration config = new ImageLoaderConfiguration.Builder(this).defaultDisplayImageOptions(options).build();

ImageLoader.getInstance().init(config);

MainActivity.prefs = new ObscuredSharedPreferences(

this, this.getSharedPreferences("LOCAL\_DATA", Context.MODE\_PRIVATE));

try {

currentUser = new JSONObject(prefs.getString(prefs.getString("userId", null), null));

} catch (JSONException e) {

e.printStackTrace();

} catch (Exception e) {

e.printStackTrace();

}

setContentView(R.layout.activity\_main);

mToolbar = (Toolbar) findViewById(R.id.app\_bar);

setSupportActionBar(mToolbar);

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_layout);

mInfoDrawerLayout = (DrawerLayout) findViewById(R.id.drawer\_sidebar);

mInfoDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_CLOSED);

mInfoDrawerView = (LinearLayout) findViewById(R.id.info\_drawer);

int width = getResources().getDisplayMetrics().widthPixels;

int height = getResources().getDisplayMetrics().heightPixels;

DrawerLayout.LayoutParams params = (android.support.v4.widget.DrawerLayout.LayoutParams) mInfoDrawerView.getLayoutParams();

int display\_mode = getResources().getConfiguration().orientation;

if (display\_mode == 1) {

params.width = width;

} else {

params.width = height;

}

mInfoDrawerView.setLayoutParams(params);

mInfoDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout, R.string.drawer\_open, R.string.drawer\_close) {

/\*\* Called when a drawer has settled in a completely closed state. \*/

public void onDrawerClosed(View view) {

super.onDrawerClosed(view);

mInfoDrawerLayout.setDrawerLockMode(DrawerLayout.LOCK\_MODE\_LOCKED\_CLOSED);

invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()

}

/\*\* Called when a drawer has settled in a completely open state. \*/

public void onDrawerOpened(View drawerView) {

super.onDrawerOpened(drawerView);

invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()

}

};

mDrawerToggle = new ActionBarDrawerToggle(this, mDrawerLayout, mToolbar, R.string.drawer\_open, R.string.drawer\_close) {

/\*\* Called when a drawer has settled in a completely closed state. \*/

public void onDrawerClosed(View view) {

super.onDrawerClosed(view);

invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()

}

/\*\* Called when a drawer has settled in a completely open state. \*/

public void onDrawerOpened(View drawerView) {

super.onDrawerOpened(drawerView);

invalidateOptionsMenu(); // creates call to onPrepareOptionsMenu()

}

};

mDrawerLayout.setDrawerListener(mDrawerToggle);

mInfoDrawerLayout.setDrawerListener(mInfoDrawerToggle);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);

getSupportActionBar().setHomeButtonEnabled(true);

menuList = new ArrayList<NavMenuItem>();

menuList.add(new NavMenuItem(getResources().getString(R.string.title\_main\_page\_fragment), R.drawable.ic\_school\_grey600\_24dp, new MainPageFragment()));

//menuList.add(new NavMenuItem(getResources().getString(R.string.title\_messenger\_fragment), R.drawable.ic\_messenger\_grey600\_24dp, new MessengerFragment()));

menuList.add(new NavMenuItem(getResources().getString(R.string.title\_schedule\_fragment), R.drawable.ic\_event\_note\_grey600\_24dp, new ScheduleFragment()));

menuList.add(new NavMenuItem(getResources().getString(R.string.title\_disciplines\_fragment), R.drawable.ic\_book\_grey600\_24dp, new DisciplinesFragment()));

//menuList.add(new NavMenuItem(getResources().getString(R.string.title\_control\_fragment), R.drawable.ic\_check\_grey600\_24dp, new ControlFragment()));

mRecyclerView = (RecyclerView) findViewById(R.id.menu\_recycler\_view);

mLayoutManager = new LinearLayoutManager(this);

mRecyclerView.setLayoutManager(mLayoutManager);

mAdapter = new MenuAdapter(menuList, getSupportFragmentManager(), this);

mRecyclerView.setAdapter(mAdapter);

mLogout = findViewById(R.id.logout\_container);

mLogout.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

finish();

sessionId = null;

MainActivity.prefs.edit().clear().commit();

MainActivity.this.startActivity(new Intent(MainActivity.this, LoginActivity.class));

}

});

getSessionId();

mToolbarContainer = findViewById(R.id.toolbar\_conatiner);

showToolbar();

}

@Override

public void onBackPressed() {

if (mInfoDrawerLayout.isDrawerOpen(GravityCompat.END)) { //replace this with actual function which returns if the drawer is open

mInfoDrawerLayout.closeDrawer(GravityCompat.END); // replace this with actual function which closes drawer

} else {

super.onBackPressed();

}

}

@Override

public View onCreateView(View parent, String name, Context context, AttributeSet attrs) {

return super.onCreateView(parent, name, context, attrs);

}

@Override

protected void onPostCreate(Bundle savedInstanceState) {

super.onPostCreate(savedInstanceState);

// Sync the toggle state after onRestoreInstanceState has occurred.

mDrawerToggle.syncState();

}

@Override

public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig);

mDrawerToggle.onConfigurationChanged(newConfig);

}

public void getSessionId() {

mSessionIdTask = new GetSessionIdTask(MainActivity.prefs.getString("login", null), MainActivity.prefs.getString("password", null));

mSessionIdTask.execute();

}

public class GetSessionIdTask extends AsyncTask<Void, Void, Boolean> {

private final String mLogin;

private final String mPassword;

GetSessionIdTask(String login, String password) {

mLogin = login;

mPassword = password;

}

@Override

protected Boolean doInBackground(Void... params) {

try {

List<NameValuePair> reqString = new ArrayList<NameValuePair>();

reqString.add(new BasicNameValuePair("login", mLogin));

reqString.add(new BasicNameValuePair("password", mPassword));

Log.d("lol", "starting");

// getting product details by making HTTP request

JSONObject json = jsonParser.makeHttpRequest(

LoginActivity.LOGIN\_URL, "GET", reqString);

// check your log for json response

Log.d("lol", json.toString());

if (json.getInt("StatusCode") == 200) {

MainActivity.sessionId = json.getString("Data");

reqString.clear();

reqString.add(new BasicNameValuePair("sessionId", MainActivity.sessionId));

return true;

}

} catch (Exception e) {

Log.d("lol", e.toString());

return false;

}

return false;

}

}

}

public class SchedulePagerAdapter extends PagerAdapter {

List<String> mDaysOfWeek = new ArrayList<>();

private RecyclerView mRecyclerView;

private LinearLayoutManager mLayoutManager;

private ScheduleAdapter mAdapter;

private View[] mPages;

private int mWeek;

private Context mContext;

private Calendar mCalendar;

private int mCurrentWeek;

private int mCurrentWeekDay;

public SchedulePagerAdapter(int week, Context context) {

mWeek = week;

mCalendar = Calendar.getInstance();

mCurrentWeek = mCalendar.get(Calendar.WEEK\_OF\_YEAR) % 2;

mCurrentWeekDay = mCalendar.get(Calendar.DAY\_OF\_WEEK) - 2;

try {

if (mCurrentWeekDay < 0) mCurrentWeekDay = 7 - mCurrentWeekDay;

Log.v("lol", mCurrentWeek + " " + mCurrentWeekDay);

if (MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").opt(1) != null)

mDaysOfWeek.add("ПН".toUpperCase());

if (MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").opt(2) != null)

mDaysOfWeek.add("ВТ".toUpperCase());

if (MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").opt(3) != null)

mDaysOfWeek.add("СР".toUpperCase());

if (MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").opt(4) != null)

mDaysOfWeek.add("ЧТ".toUpperCase());

if (MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").opt(5) != null)

mDaysOfWeek.add("ПТ".toUpperCase());

if (MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").opt(6) != null)

mDaysOfWeek.add("СБ".toUpperCase());

} catch (JSONException e) {

e.printStackTrace();

}

mPages = new View[getCount()];

mContext = context;

}

/\*

\* @return the number of pages to display

\*/

@Override

public int getCount() {

return mDaysOfWeek.size();

}

/\*\*

\* @return true if the value returned from {@link #instantiateItem(android.view.ViewGroup, int)} is the

\* same object as the {@link View} added to the {@link android.support.v4.view.ViewPager}.

\*/

@Override

public boolean isViewFromObject(View view, Object o) {

return o == view;

}

// BEGIN\_INCLUDE (pageradapter\_getpagetitle)

/\*\*

\* Return the title of the item at {@code position}. This is important as what this method

\* returns is what is displayed in the {@link SlidingTabLayout}.

\* <p/>

\* Here we construct one using the position value, but for real application the title should

\* refer to the item's contents.

\*/

public View getPage(int position) {

return mPages[position];

}

@Override

public CharSequence getPageTitle(int position) {

return mDaysOfWeek.get(position);

}

// END\_INCLUDE (pageradapter\_getpagetitle)

/\*\*

\* Instantiate the {@link View} which should be displayed at {@code position}. Here we

\* inflate a layout from the apps resources and then change the text view to signify the position.

\*/

@Override

public Object instantiateItem(ViewGroup container, int position) {

// Inflate a new layout from our resources

View view = ((MainActivity) mContext).getLayoutInflater().inflate(R.layout.pager\_item,

container, false);

// Add the newly created View to the ViewPager

container.addView(view);

mRecyclerView = (RecyclerView) view.findViewById(R.id.schedule\_recycler\_view);

mRecyclerView.setHasFixedSize(true);

mLayoutManager = new LinearLayoutManager(mContext);

mLayoutManager.setOrientation(LinearLayoutManager.VERTICAL);

mRecyclerView.setBackgroundColor(mContext.getResources().getColor(R.color.background\_material\_light));

mRecyclerView.setLayoutManager(mLayoutManager);

try {

mAdapter = new ScheduleAdapter(MainActivity.currentUser.getJSONObject("schedule").getJSONArray(mWeek + 1 + "").getJSONArray((position + 1)), mContext);

} catch (JSONException e) {

e.printStackTrace();

}

mRecyclerView.setAdapter(mAdapter);

mPages[position] = view;

return view;

}

@Override

public void destroyItem(ViewGroup container, int position, Object object) {

container.removeView((View) object);

}

public String getPageDate(int i) {

mCalendar = Calendar.getInstance();

mCalendar.add(Calendar.WEEK\_OF\_YEAR, mCurrentWeek != mWeek ? 1 : 0);

mCalendar.add(Calendar.DAY\_OF\_WEEK, i - mCurrentWeekDay);

return mCalendar.get(Calendar.DAY\_OF\_MONTH) + " " + mCalendar.getDisplayName(Calendar.MONTH, Calendar.LONG, Locale.getDefault());

}

}