


বাংলায় অ্যান্ড্রয়েড সহায়িকা



National
Mobile Application
Awareness Development &
Capacity Building Program

 **GDG** Sonargaon

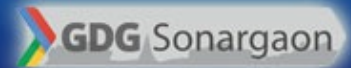
বাংলায় অ্যান্ড্রয়েড সহায়িকা

অ্যান্ড্রয়েডকে বাংলা ভাষাভাষি মানুষের কাছে আরো সহজভাবে উপস্থাপন করার জন্য আমাদের “বাংলায় অ্যান্ড্রয়েড সহায়িকা” পোর্টালটি সবার জন্য উন্মুক্ত করা হল।

জাতীয় মোবাইল অ্যাপ প্রশিক্ষণ কর্মসূচির মাধ্যমে ইতিমধ্যে বাংলাদেশের ৬৪টি জেলায় অ্যান্ড্রয়েড অ্যাপ প্রশিক্ষণ দেয়া হয়েছে। এরই ধারাবাহিকতায় এই বই / পোর্টালটি প্রকাশ করা হল। গুগল নিয়ন্ত্রিত অ্যান্ড্রয়েড এর অফিসিয়াল ডেভলপার সাইটের কনটেন্ট এর বাংলা সংস্করণ হল এই “বাংলায় অ্যান্ড্রয়েড সহায়িকা”।

এই পোর্টালটি এখনও অসম্পূর্ণ এবং এতে প্রতিনিয়ত আপডেটের কাজ চলবে। আপনার যে কোন মন্তব্য/ সংশোধন/ অবদান এর জন্য আমাদের জানান। আমাদের টিম বিষয়টি পর্যালোচনা করে উপযুক্ত ব্যবস্থা গ্রহণ করবে।

বাংলায় অ্যান্ড্রয়েড সহায়িকা



মন্তব্য/ সংশোধন/ অবদান

আপনার যে কোন মন্তব্য/ সংশোধন/ অবদান এর জন্য আমাদের জানান। আমাদের টিম বিষয়টি পর্যালোচনা করে উপযুক্ত ব্যবস্থা গ্রহন করবে।

Loading...

অ্যাপলিকেশন তৈরীর কাজ শুরু করুন

(<http://developer.android.com/training/index.html>)

অ্যান্ড্রয়েড ডেভেলপারদের প্রশিক্ষণে আপনাদের স্বাগতম। এই ক্লাসে এমন সব শিক্ষা পাবেন যেখানে আলোচনা করা হবে কীভাবে কোড স্যাম্পল দিয়ে একটি সুনির্দিষ্ট কাজ আপনি আপনার অ্যাপ এ ব্যবহার করতে পারবেন। ক্লাসগুলো কয়েকটি গ্রুপে শ্রেণীবদ্ধ করা হয়েছে যা উপরের লিংকের বাম পাশের নেভিগেশনের উপরের দিকে দেখতে পাবেন।

প্রথম গ্রুপ, অর্থাৎ এই অধ্যায় আপনাকে অ্যান্ড্রয়েড অ্যাপ ডেভেলপ এর মূল বিষয়গুলো শেখাবে। আপনি যদি নতুন অ্যান্ড্রয়েড অ্যাপ ডেভেলপার হোন, তাহলে আপনার উচিত এই ক্লাসগুলো ধারাবাহিকভাবে সম্পূর্ণ করা।

১. প্রথম অ্যাপ তৈরী করা

অ্যান্ড্রয়েড সফটওয়্যার ডেভেলপ কিট ইনস্টল করার পর অ্যান্ড্রয়েড অ্যাপ ডেভেলপ এর মূল বিষয়গুলো শেখার জন্য এই ক্লাস দিয়ে শুরু করুন।

১. এন্ডরয়েডের প্রজেক্ট তৈরী করা
২. আপনার অ্যাপলিকেশন রান করা
৩. সহজ ইউজার ইন্টারফেস তৈরী করা
৪. অন্যান্য কার্যক্রম শুরু করা

২. একশান বার সংযোজন

অ্যাপ কার্যক্রম বাস্তবায়নের সবচেয়ে গুরুত্বপূর্ণ ডিজাইন উপাদান হচ্ছে একশন বার। যদিও ১১ লেভেলের এপিআই চালু করার সময় অ্যান্ড্রয়েড ২.১ বা এর উপরের সংস্করণের ডিভাইসে একশন বার অন্তর্ভুক্ত করার জন্য আপনি সাপোর্ট লাইব্রেরী ব্যবহার করতে পারেন।

১. একশান বার সেট আপ করা
২. একশান বার সংযোজন করা
৩. একশন বার স্টাইল করা
৪. একশন বার ওভারলে করা

৩. ভিন্ন ভিন্ন ডিভাইসকে সাপোর্ট করা

কীভাবে বিকল্প উপায়ে অ্যাপ তৈরী করা যায় যা বিভিন্ন সংস্করণের ডিভাইসে একটি একক অচক ব্যবহার করে ব্যবহারকারীদের একটি নতুন অভিজ্ঞতা দেওয়া যায়।

১. ভিন্ন ভিন্ন ভাষাকে সাপোর্ট করা

২. ভিন্ন ভিন্ন স্ক্রিন সাপোর্ট করা

৩. ভিন্ন ভিন্ন সংস্করণের প্লাটফর্মকে সাপোর্ট করা

৪. একটিভিটি লাইফসাইকেল ব্যবস্থাপনা করা

কীভাবে অ্যান্ড্রয়েড একাটিভিটি কাজ শুরু করে এবং শেষ করে এবং লাইফসাইকেল কলব্যাক পদ্ধতি বাস্তবায়ন করে ব্যবহারকারীদের একটি নির্ভুল সুন্দর অভিজ্ঞতা দেওয়া যায়।

১. একটিভিটি (কর্মকাল) শুরু করা

২. একটিভিটিতে (কর্মকাল) বিরতি এবং বিরতির পর পূরণায় শুরু

৩. কর্মকাল থামানো (স্টপ) এবং পূরণায় (রিস্টার্ট) শুরু করা

৪. কর্মকাল পূর্ণনির্মান

৫. ফ্র্যাগমেন্ট সহকারে একটি ডায়নামিক ইউজার ইন্টারফেস তৈরী করা

অ্যাপ এর জন্য কীভাবে ইউজার ইন্টারফেস তৈরী করা হয় যা বড় স্ক্রিনে বহুবিধ ইউজার ইন্টারফেস কার্যক্রম পরিবেশনের জন্য যথেষ্ট নমনীয় এবং ছোট স্ক্রিনে কাজ করার মতো উপযোগী করা-ফোন এবং ট্যাবলেট এর জন্য একক অ্যাপলিকেশন প্যাকেজ এর মূল উপাদান গুলো তৈরী করা।

১. একটি ফ্র্যাগমেন্ট তৈরী করা

২. নমনীয় (ফ্লেক্সিবল) ইউজার ইন্টারফেস তৈরী

৩. অন্যান্য ফ্র্যাগমেন্ট সাথে যোগাযোগ করা

৬. তথ্য সেভ করা

কীভাবে তথ্য ডিভাইসে সেভ করতে হয়, সেটাহতে পারে এটা অস্থায়ী ফাইল, ডাউনলোড করা অ্যাপ তথ্য, ব্যবহারকারীর মিডিয়া, কাঠামোবদ্ধ তথ্য অথবা অন্য কিছু।

১. কি-ভ্যালু সেট সেভ করা

২. ফাইল সেভ করা

৩. SQL ডাটাবেজে তথ্য সেভ করা

৭. অন্য এ্যাপের সাথে পারস্পরিক ক্রিয়া (ইন্টারেকশন) করা

উন্নতমানের কর্ম সম্পাদনের লক্ষ্যে ডিভাইসে বিদ্যমান অন্যান্য অ্যাপ এর সর্বোচ্চ সুবিধা নিয়ে ব্যবহারকারীদের একটা ভালো অভিজ্ঞতার ব্যবস্থা করা।

১. ব্যবহারকারীদের (ইউজার) অন্য অ্যাপ এ পাঠানো
২. একটিভিটি (কর্মকান্ড) থেকে ফলাফল প্রাপ্তি
৩. একটিভিটি শুরু করতে অন্য অ্যাপকে অনুমোদন

আপনার প্রথম অ্যাপ তৈরী করুন

(<http://developer.android.com/training/basics/firstapp/index.html>)

অ্যান্ড্রয়েড এ্যাপলিকেশন ডেভেলপমেন্ট এ আপনাকে স্বাগতম!

এই ক্লাসে আপনি শিখবেন কীভাবে আপনার প্রথম অ্যান্ড্রয়েড অ্যাপ তৈরী করবেন। আপনি শিখতে পারবেন কীভাবে অ্যান্ড্রয়েড প্রজেক্ট করা যায় এবং অ্যাপ এর ডিবাগএবল সংস্করন পরিচালনা করা যায়। আপনি আরও শিখতে পারেন অ্যান্ড্রয়েড অ্যাপ ডিজাইন করার কিছু মৌলিক ধারণা যার মধ্যে রয়েছে কীভাবে একটি সরল ইউজার ইন্টারফেস তৈরী করা যায় এবং ইউজার ইনপুট পরিচালনা করা যায়। এই ক্লাস শুরু করার পূর্বে আপনি নিশ্চিত হয়ে নিন যে ডেভেলপের পূর্ণ পরিবেশ তৈরী করা আছে, যেখানে দরকার:

১. অ্যান্ড্রয়েড এসডিকে (SDK) ডাউনলোড করা
২. ইক্লিপস এর জন্য অ্যান্ড্রয়েড ডেভেলপার টুল (ADT) প্লাগিন ইনস্টল করা (আপনি যদি ইক্লিপস IDE ব্যবহার করে থাকেন)
৩. এসডিকে ম্যানেজার ব্যবহার করে সর্বশেষ এসডিকে টুলস এবং প্লাটফর্ম ডাউনলোড করুন

আপনি যদি ইতিমধ্যে এই কাজগুলো শেষ করে না থাকেন, তাহলে অ্যান্ড্রয়েড এসডিকে (Android SDK) (এই <http://developer.android.com/sdk/index.html> লিংক থেকে) ডাউনলোড করুন, নিশ্চিত ইনস্টল এর ধাপগুলো অনুসরণ করুন। আর আপনি যদি এই কাজ শেষ করে থাকেন তাহলে আপনি এই কাজ শুরু করার জন্য প্রস্তুত। এই ক্লাসে একটা টিউটরিয়াল ব্যবহার করা হবে যাতে ধীরে ধীরে একটা অ্যান্ড্রয়েড অ্যাপ তৈরীর করে দেখানো হবে যেখানে অ্যান্ড্রয়েড অ্যাপ ডেভেলপমেন্ট এর মৌলিক ধারণাগুলি সম্পর্কে জানা যাবে, সুতরাং প্রতিটা ধাপ অনুসরণ করা গুরুত্বপূর্ণ।

প্রথম পাঠর শুরু করা

অ্যান্ড্রয়েড প্রজেক্ট তৈরী করা

(<http://developer.android.com/training/basics/firstapp/creating-project.html>)

একটি অ্যান্ড্রয়েড প্রজেক্ট সকল ফাইল ধারণ করে, যা আপনার অ্যান্ড্রয়েড অ্যাপ এর জন্য সোর্স কোড গঠন করে। ডিফল্ট প্রজেক্ট ডিরেক্টরি এবং ফাইলের একটি সেটের সাথে একটি নতুন অ্যান্ড্রয়েড প্রজেক্ট শুরু করাটা অ্যান্ড্রয়েড এসডিকে টুলস সহজ করে দিয়েছে। এই অনুশীলনী আপনাকে দেখাবে কীভাবে একটি কমান্ড লাইন থেকে ইক্লিপস (এডিটিং প্লাগিং সহ) বা এসডিকে টুলস ব্যবহার করে নতুন প্রজেক্ট শুরু করা যায়।

নোট: আপনার অ্যান্ড্রয়েড এসডিকে ইনস্টলড করে ফেলা উচিত, এবং আপনি যদি ইক্লিপস ব্যবহার করে থাকেন তাহলে আপনার ADT Plugin (21.0.0 সংস্করণ বা এর পরবর্তী সংস্করণ) ইনস্টল করে ফেলা উচিত। আপনি যদি না করে থাকেন তাহলে এই অনুশীলনী শুরু করার পূর্বে Installing the Android SDK এই নির্দেশিকা অনুসরণ করা উচিত (লিংক: <http://developer.android.com/sdk/installing/index.html>)।

ইক্লিপস দিয়ে প্রজেক্ট তৈরী করা

১. টুল বারের New  ক্লিক করুন

২. উইন্ডোতে যা দেখা যাবে, সেখানে Android ফোল্ডার ওপেন করুন, Android Application Project নির্বাচন করুন, Next এ ক্লিক করুন

☐ ফিগার ১. ইক্লিপস এ নতুন অ্যান্ড্রয়েড অ্যাপ প্রজেক্ট উইজার্ড

৩. যে ফর্মটি আসবে তা পূরণ করুন:

- Application Name হচ্ছে অ্যাপ এর নাম যেটা হারকারীদের নিকট দৃশ্যমান হবে। এই প্রজেক্টের জন্য এটার নাম দিন
- Package Name হচ্ছে আপনার অ্যাপের প্যাকেজ নেমস্পেস (জাভা প্রোগ্রামিং লেঙ্গুয়েজ প্যাকেজগুলোর মতো একই নিয়ম মেনে চলতে হবে)। আপনার প্যাকেজের নাম অবশ্যই সম্পূর্ণ অ্যান্ড্রয়েড সিস্টেমে যে সকল প্যাকেজ ইনস্টলড হয়েছে তার থেকে স্বতন্ত্র হতে হবে। এই কারনে সবচেয়ে ভালো হয় আপনি যদি আপনার প্রতিষ্ঠানের নাম বা আপনার প্রকাশনী (পাবলিকেশন) পরিচয়ের যে ডোমিয়েন নাম আছে তা উল্টো করে ব্যবহার করেন। এই প্রজেক্টের জন্য আপনি "com.example.myfirstapp" এই ধরনের কিছু একটা ব্যবহার করতে পারেন। কিন্ত এই "com.example" নেমস্পেস দিয়ে আপনার অ্যাপ গুগলে প্রকাশ করতে পারবেন না।
- Minimum Required SDK হচ্ছে অ্যান্ড্রয়েডের সর্বনিম্ন সংস্করণ যা আপনার অ্যাপ সাপোর্ট করে, এপিআই লেভেল ব্যবহার নির্দেশ করে। যত বেশী ডিভাইসকে সাপোর্ট করা সম্ভব তা করতে এটাতে সর্বনিম্ন ভার্সন বা সংস্করণ সেট করা উচিত যা আপনার অ্যাপ অনুমোদন করে এর মূল (কোর) বৈশিষ্ট্যগুলো দিতে পারে। যদি আপনার অ্যাপের কোন ফিচার শুধুমাত্র অ্যান্ড্রয়েডের নতুন সংস্করণে সম্ভব হয় এবং এটা অ্যাপের কোর ফিচার সেটের জন্য ক্রিটিক্যাল না হয়, আপনি তখনই ফিচারটি চালু করতে পারবেন, যখন এই ভার্সনে এটা চালানো হবে যা এটাকে সাপোর্ট করবে। (যেভাবে ভিন্ন ভিন্ন প্ল্যাটফর্ম সংস্করণকে সাপোর্ট করা অধ্যায়ে আলোচনা করা হয়েছে)। এই প্রজেক্টের জন্য এই সেট ডিফল্ট ভ্যালুতে ছেড়ে দিন।
- Target SDK অ্যান্ড্রয়েডের সর্বোচ্চ ভার্সন কে নির্দেশ করে (API Level ও ব্যবহার করে থাকে) যা দিয়ে আপনি আপনার অ্যাপলিকেশন পরীক্ষা করিয়ে নিতে পারেন।
যখন অ্যান্ড্রয়েডের নতুন কোন ভার্সন (সংস্করণ) আসবে, আপনার উচিত হবে নতুন ভার্সনে আপনার অ্যাপ পরীক্ষা করে নেয়া এবং নতুন প্ল্যাটফর্ম বৈশিষ্ট্যের সুবিধা নিতে এই সর্বশেষ এপিআই লেভেলের সাথে ম্যাচ করাতে এই ভ্যালু আপডেট করুন।
- Compile With হচ্ছে প্ল্যাটফর্ম ভার্সন যার বিপরীতে আপনি আপনার অ্যাপ কমপাইল করতে পারবেন। সর্বশেষ অ্যান্ড্রয়েড ভার্সনে এটা বাই ডিফল্ট সেট করা আছে যা

আপনার এসাউকে তে পাবেন (এটা অ্যান্ড্রয়েড ৪.১ অথবা এর চেয়ে বেশী হওয়া উচিত; আপনার যদি এই ভার্সান না থাকে তাহলে অবশ্যই SDK Manager ব্যবহার করে এটা ইনস্টল করতে হবে) লিংক: <http://developer.android.com/sdk/installing/adding-packages.html> । আপনি এখনও আপনার অ্যাপ পুরাতন ভার্সনের জন্য তৈরী করতে পারেন, কিনুত এই সেটিং এর লক্ষ্য হওয়া উচিত যাতে এটা নতুন ভার্সানেও কাজ করে এবং সর্বশেষ ভার্সানের ডিভাইসেও এই অ্যাপ ব্যবহারকারীদের জন্য অন্তর্ভুক্ত করে দেয়া উচিত।

- Theme নির্দেশ করে আপনার অ্যাপে কাজ করানোর জন্য অ্যান্ড্রয়েড ইউজার ইন্টারফেস স্টাইল। আপনি এটাকে নিজের মতো করে কাজ করতে দিতে পারেন।

Next বাটনে ক্লিক করুন

৪. পরের স্ক্রিনে প্রজেক্ট এর আকৃতি দিতে, ডিফল্ট সিলেকশন কে সেভাবেই রেখে Next বাটনে ক্লিক করুন*

৫. পরবর্তী স্ক্রিন আপনার অ্যাপের জন্য লক্ষ্যার আইকন তৈরী করতে সাহায্য করবে।

আপনি বিভিন্ন উপায়ে একটি আইকন এর ধরন নির্ণয় করতে পারেন এবং টুলগুলো সকল স্ক্রিনের ঘনত্ব (ডেনজিটি) অনুসারে একটি আইকন তৈরী করতে পারে। আপনার অ্যাপ পাবলিশ করার আগে আপনার নিশ্চিত হওয়া উচিত যে Iconography ডিজাইন গাইড এ নির্দিষ্টভাবে যে সংজ্ঞা দেয়া আছে তার সাথে আপনার আইকন মেলে কিনা। (লিংক: <http://developer.android.com/design/style/iconography.html>)

Next বাটনে ক্লিক করুন

৬. আপনার অ্যাপ তৈরী করা শুরু করতে আপনি একটা এ্যাক্টিভিটি টেমপ্লেট তৈরী করতে পারেন

এই প্রজেক্ট এর জন্য BlankActivity নির্বাচিত করুন এবং Next বাটনে ক্লিক করুন এখোন যা যা যেভাবে আছে সেভাবেই রাখুন এবং Finish বাটনে ক্লিক করুন

কিছু ডিফল্ট ফাইলের সাথে আপনার অ্যান্ড্রয়েড প্রজেক্টের সেট আপ করা শেষ হয়েছে এখন আপনি আপনার অ্যাপ তৈরির কাজ শুরু করতে প্রস্তুত. পরবর্তী অনুশীলনীতে চলে যান:

কমান্ড লাইন টুলস দিয়ে প্রজেক্ট তৈরী করুন

আপনি যদি ADT দিয়ে Eclipse IDE ব্যবহার না করে থাকেন, তাহলে আপনি এর পরিবর্তে কমান্ড লাইন থেকে SDK tools ব্যবহার করে আপনার প্রজেক্ট তৈরী করতে পারেন:

১. অ্যান্ড্রয়েড SDK's tools এর অভ্যন্তরের ডিরেক্টরিস পরিবর্তন করুন
২. যা করতে হবে:

Android list targets

* এটা অ্যান্ড্রয়েড প্ল্যাটফর্মের একটি তালিকা তৈরী করে দিবে যা আপনি আপনার সফটওয়্যার ডেভেলপমেন্ট টুল (SDK) এর জন্য ডাউনলোড করেছিলেন। প্ল্যাটফর্মটি খুঁজে নিন যার বিপরীতে আপনি আপনার অ্যাপ কম্পাইল করতে চান। টার্গেট আইডির একটা নোট তৈরী করুন। আমাদের পরামর্শ হচ্ছে আপনি যথাসম্ভব সবচেয়ে আপডেট ভার্সন কে নির্বাচিত করুন। আপনি এখনও পুরাতন ভার্সনকে সাপোর্ট করতে পারে এমন অ্যাপ তৈরী করতে পারেন, কিন্ত এই সেটিং এর লক্ষ্য হওয়া উচিত যাতে এটা নতুন ভার্সনেও কাজ করে এবং সর্বশেষ ভার্সনের ডিভাইসেও এই অ্যাপ ব্যবহারকারীদের জন্য অন্তর্ভুক্ত করে দেয়া উচিত।

আপনি যদি দেখেন যে কোন টার্গেট লিস্টেড না, তাহলে আপনাকে অ্যান্ড্রয়েড এসডিকে ম্যানেজার টুল ব্যবহার করে এটা ইনস্টল করতে হবে। দেখুন Adding Platforms and Packages: (<http://developer.android.com/sdk/installing/adding-packages.html>)।

৩. যা করতে হবে:

```
android create project --target --name MyFirstApp \  
--path /MyFirstApp --activity MainActivity \  
--package com.example.myfirstapp
```

টার্গেট তালিকা থেকে একটা আইডির সাথে এর পরিবর্তে প্রতিস্থাপন করে নিন (পূর্ববর্তী ধাপ থেকে) এবং যে লোকেশনে আপনার অ্যান্ড্রয়েড প্রজেক্ট সেভ করতে চান সেই সেই লোকেশন এখানে প্রতিস্থাপন করুন।

আপনার অ্যান্ড্রয়েড প্রজেক্ট এখন কিছু ডিফল্ট কনফিগারেশন দিয়ে সেট আপ হয়েছে এবং আপনি এখন আপনার অ্যাপ তৈরীর জন্য কাজ শুরু করতে পারেন। পরবর্তী অনুশীলনীতে যান।

পরামর্শ*: আপনার PATH এনভায়রনমেন্ট ভেরিয়েবল এ platform-tools/ এবং একই সাথে tools/ যোগ করুন

অ্যাপ রান করা

(<http://developer.android.com/training/basics/firstapp/running-app.html>)

যদি একটি অ্যান্ড্রয়েড প্রজেক্ট তৈরী করতে পূর্ববর্তী অনুশীলনী অনুসরণ করে থাকেন, এরমধ্যে অন্তর্ভুক্ত আছে "Hello World" সোর্স ফাইল এর একটি ডিফল্ট সেট যা অ্যাপ রান করতে তাৎক্ষণিক অনুমোদন দিয়ে থাকে।

আপনি কীভাবে আপনার অ্যাপ রান করাবেন তা দুটো বিষয়ের উপর নির্ভর করে:

আপনার সত্যিকার অ্যান্ড্রয়েড পাওয়ারড ডিভাইস আছে অথবা আপনি ইমুলেটর ব্যবহার করে থাকেন। এই অনুশীলনী আপনাকে শেখাবে আপনার অ্যাপ কীভাবে সত্যিকার ডিভাইসে অথবা অ্যান্ড্রয়েড ইমুলেটরে ইনস্টল ও রান করতে হয়, এবং উভয় ক্ষেত্রেই ইমুলেটর অথবা কমান্ড লাইন টুল দিয়ে ইনস্টল ও রান করতে হয়।

আপনার অ্যাপ রান করার পূর্বে অ্যান্ড্রয়েডে প্রজেক্টের কিছু ডিরেক্টরি এবং ফাইল সম্পর্কে সতর্ক থাকা উচিত:

AndroidManifest.xml

Manifest file অ্যাপের মৌলিক বৈশিষ্ট্য নিয়ে আলোচনা করে এবং এর প্রতিটা উপাদান নিয়েও আলোচনা করে। এই ফাইলে আপনি নানা ধরনের আলোচনা শিখতে পারবেন যদি আপনি আরও ট্রেনিং ক্লাসগুলো পড়েন।

আপনার মেনিফেস্টের উপাদানের অন্যতম উপাদান হিসাবে উপাদান অন্তর্ভুক্ত হওয়া উচিত। এটা বিভিন্ন অ্যান্ড্রয়েড ভার্সনে android:minSdkVersion এবং android:targetSdkVersion এট্রিবিউট ব্যবহার করে আপনার অ্যাপস এর কমপেটিবিলিটি কে ঘোষণা করে। আপনার প্রথম অ্যাপের ক্ষেত্রে এটা এমন দেখানো উচিত:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >
<uses-sdk android:minSdkVersion="8" android:targetSdkVersion="17" />
...
</manifest>
```

আপনার সব সময়ই উচিত সর্বোচ্চ পর্যায়ে android:targetSdkVersion সেট করা এবং অনুরূপ প্ল্যাটফর্ম এ আপনার অ্যাপ টেস্ট করে নেওয়া। আরও তথ্যের জন্যে দেখুন (Supporting Different Platform Versions) ভিন্ন ভিন্ন প্ল্যাটফর্ম সংস্করণকে সাপোর্ট করা অধ্যায়টি দেখুন।

src/

আপনার অ্যাপের প্রধান সোর্স ফাইল এর জন্য ডিরেক্টরি। বাই ডিফল্ট এটা একটি Activity ক্লাসে অন্তর্ভুক্ত যা তখন রান করে যখন অ্যাপ আইকন ব্যবহার করে এটা তার যাত্রা শুরু করে।

src/

app resources এর জন্য কিছু সাব-ডিরেক্টরি ধারণ করে।এখানে অল্প কিছু আছে:

drawable-hdpi/

ড্রয়েবল অবজেক্টের জন্য ডিরেক্টরী (যেমন বিটম্যাপ) যা উচ্চ ঘনত্বমান সম্পন্ন স্ক্রিনের (hdpi) জন্য ডিজাইন করা হয়েছে। অন্য ড্রয়েবল ডিরেক্টরি সেই বৈশিষ্ট্য ধারণ করে যা অন্যান্য স্ক্রিন এর জন্য প্রযোজ্য।

layout/

অ্যাপের ইউজার ইন্টারফেসের আলোচনা করে এমন ফাইলের জন্য ডিরেক্টরী

values/

অন্যান্য বহুবিধ XML ফাইলের জন্য ডিরেক্টরী যা রিসোর্স এর সংগ্রহকে ধারণ করে যেমন স্ট্রিং এবং কালার এর বর্ণনা।

যখন আপনি ডিফল্ট অ্যান্ড্রয়েড অ্যাপ তৈরী এবং রান করবেন, তখন ডিফল্ট Activity ক্লাস একটা লে আউট ফাইল শুরু করবে এবং লোড করা শুরু করবে যা বলবে "Hello World"। এর ফলাফল চমকপ্রদ কিছুই নয় কিনুত এটা গুরুত্বপূর্ণ যে ডেভেলপিং শুরুর পূর্বেই আপনি জানেন আপনার অ্যাপ কীভাবে রান করে।

একটা সত্যিকার ডিভাইসে রান করুন

আপনার যদি একটি সত্যিকার অ্যান্ড্রয়েড পাওয়ারড ডিভাইস থাকে, তাহলে আপনি নিম্নোক্ত পদ্ধতিতে আপনার অ্যাপ ইনস্টল এবং রান করতে পারবেন:


1. আপনার ডিভাইসটা ইউএসবি ক্যাবলের মাধ্যমে আপনার ডেভেলপমেন্ট মেশিনে প্লাগ ইন করুন, আপনার ডিভাইসের সাথে সামঞ্জস্যপূর্ণ ইউএসবি ড্রাইভার ইনস্টল করা লাগতে পারে। ড্রাইভার ইনস্টল করতে সাহায্যের জন্য OEM USB Drivers (<http://developer.android.com/tools/extras/oem-usb.html>) তথ্যটি দেখুন।
2. আপনার ডিভাইসে USB debugging চালু করুন

* অ্যান্ড্রয়েড ৩.২ অথবা এর চেয়ে পুরাতন এর ক্ষেত্রে আপনি অপশনটি খুঁজে পাবেন Settings > Applications > Development এখানে।

* অ্যান্ড্রয়েড ৪.০ এবং এর চেয়ে উন্নত সংস্করণের ক্ষেত্রে Settings > Development options এ খুঁজে পাবেন

নোট: অ্যান্ড্রয়েড ৪.২ এবং এর চেয়ে উন্নত সংস্করণের ক্ষেত্রে Development options বাই ডিফল্ট হিডেন থাকে। এটাকে পেতে হলে Setting > About phone > যান Build number সাতবার টোকা দিন। পূর্ববর্তী স্ক্রিনে ফিরে এসে Development options পাবেন।

ইক্লিপস থেকে অ্যাপ রান করতে:

1. আপনার একটি প্রজেক্ট ফাইল ওপেন করুন এবং টুলবার থেকে **Run**  ক্লিক করুন
2. **Run as** নামে যে উইন্ডো আসবে তার **Android Application** কে নির্বাচিত করে **OK** তে ক্লিক করুন

ইক্লিপস আপনার কানেক্টেড ডিভাইসে অ্যাপ ইনস্টল করে দিবে এবং চালু করে দিবে,

অথবা কমান্ড লাইন থেকে আপনার অ্যাপ রান করা:

1. আপনার অ্যান্ড্রয়েড প্রজেক্টের দিকে ডিরেক্টরি পরিবর্তন করুন এবং করুন:

```
ant debug
```

2. নিশ্চিত করুন আপনার PATH এনভায়রনমেন্ট ভেরিয়েবলে অ্যান্ড্রয়েড এসডিকে platform-tools/ ডিরেক্টরি অন্তর্ভুক্ত আছে। তারপর এটা করুন:

3. আপনার ডিভাইসে *MyFirstActivity* খুজে বের করুন এবং এটা ওপেন করুন

এভাবে আপনি আপনার অ্যান্ড্রয়েড অ্যাপ, একটি ডিভাইসে তৈরী ও রান করাতে পারবেন।
ডেভেলপিং শুরু করতে পরবর্তী অনুশীলনীতে চলে যান।

ইমুলেটরে রান করুন

আপনি ইক্লিপস বা কমান্ড লাইন যেটাই ব্যবহার করে থাকুন না কেন, ইমুলেটরে অ্যাপ রান করতে আপনার প্রথমেই একটি Android Virtual Device (AVD) তৈরী করতে হবে। AVD হচ্ছে অ্যান্ড্রয়েড ইমুলেটরের ডিভাইস কনফিগারেশন যা আপনাকে বিবিধ ডিভাইস কে মডেল করতে অনুমোদন করে।



ফিগার ১. AVD ম্যানেজার কিছু ভার্স্যুয়াল ডিভাইস দেখাচ্ছে।

১. অ্যান্ড্রয়েড ভার্স্যুয়াল ডিভাইস ম্যানেজার:

- ইক্লিপস এ টুলবার থেকে অ্যান্ড্রয়েড ভার্স্যুয়াল ডিভাইস ম্যানেজার এ  ক্লিক করুন

* কমান্ড লাইন থেকে, ডিরেক্টরিকে /tools/ তে পরিবর্তন করুন এবং এটা করুন:

```
android avd
```

f

২. অ্যান্ড্রয়েড ভার্স্যুয়াল ডিভাইস ম্যানেজার প্যানেলে, New ক্লিক করুন।

৩. AVD এর জন্য বিস্তারিত বিবরণগুলো পূরণ করুন। এর একটি নাম, একটা প্ল্যাটফর্ম টার্গেট,

একটি SD কার্ড সাইজ এবং একটি স্ক্রিন দিন (HVGA হচ্ছে ডিফল্ট),

4 . Create AVD তে ক্লিক করুন।

5 . অ্যান্ড্রয়েড ভার্চুয়াল ডিভাইস ম্যানেজার থেকে নতুন AVD নির্বাচিত করুন এবং Start এ ক্লিক করুন

6 . ইমুলেটর বুস্ট আপ হওয়ার পর ইমুলেটর স্ক্রিন আনলক করুন।

ইক্সিপস থেকে অ্যাপ রান করতে:

1. আপনার যে কোন একটি প্রজেক্ট ফাইল ওপেন করুন এবং টুলবার থেকে Run ক্লিক করুন

1. Run as নামে যে উইন্ডো আসবে তার Android Application কে নির্বাচিত করে OK তে ক্লিক করুন

ইক্সিপস আপনার AVD তে অ্যাপ ইনস্টল করে দিবে এবং চালু করে দিবে

অথবা কমান্ড লাইন থেকে আপনার অ্যাপ রান করুন

1 . আপনার অ্যান্ড্রয়েড প্রজেক্ট এর দিকে ডিরেক্টরি পরিবর্তন করুন এবং নিচের কাজটি করুন :

```
ant debug
```

2 . নিশ্চিত করুন আপনার PATH এনভায়রনমেন্ট ভেরিয়েবলে অ্যান্ড্রয়েড এসডিকে platform-tools/ ডিরেক্টরি অন্তর্ভুক্ত আছে। তারপর করুন:

```
adb install bin/MyFirstApp-debug.apk
```

3 . ইমুলেটরে *MyFirstActivity* খুজে বের করুন এবং এটা ওপেন করুন

এভাবে আপনি আপনার অ্যান্ড্রয়েড অ্যাপ ইমুলেটরে তৈরী ও রান করাতে পারবেন। ডেভেলপিং শুরু করতে পরবর্তী অনুশীলনীতে চলে যান।

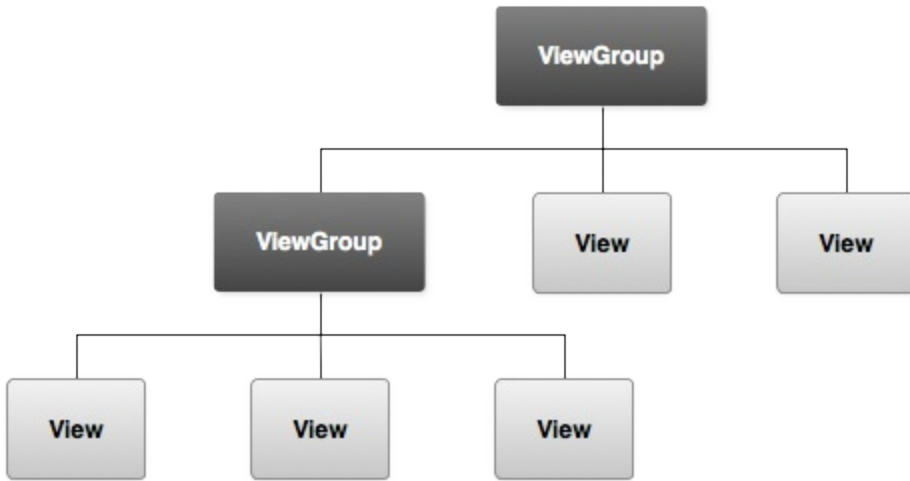
একটি সরল ইউজার ইন্টারফেস তৈরী করা

(<http://developer.android.com/training/basics/firstapp/building-ui.html>)

অ্যান্ড্রয়েড অ্যাপ এর গ্রাফিক্যাল ইউজার ইন্টারফেস, View এবং ViewGroup এর বিষয়বসূত্র হায়ারারকি ব্যবহার করে তৈরী। View অবজেক্ট আসলে ইউজার ইন্টারফেস উপাদান যেমন Buttons বা text fields এবং ViewGroup হচ্ছে অদৃশ্য ভিউ কন্টেইনার যা চাইল্ড ভিউ কেমন হবে তার পরিকল্পনা, যেমন কোন গ্রিড বা কোন ভার্টিক্যাল লিস্ট এ।

অ্যান্ড্রয়েড একটি XML ভোকাবিউলারি সরবরাহ করে থাকে যা View এবং ViewGroup এর সাব ক্লাসগুলোকে ক্রেসপল্ড করে, সুতরাং ইউজার ইন্টারফেসের হায়ারারকি ব্যবহার করে আপনি আপনার ইউজার ইন্টারফেসকে XML এ আলোচনা করতে পারেন

বিকল্প লে আউট : আপনার ইউজার ইন্টারফেস লেআউট রান টাইম কোড এর বদলে XML এ আছে যা জানানোটা বিবিধ কারনে জরুরী, কিন্ত এটা বিশেষ কারনে জরুরী এই জন্য যে আপনি ভিন্ন স্ক্রিন সাইজের জন্য ভিন্ন লেআউট তৈরী করতে পারবেন। উদাহরণ হিসাবে বলা যায় আপনি দুইটা ভার্শনের লে আউট তৈরী করতে পারেন এবং বিদ্যমান সিস্টেম কে জানানো হবে যে একটি ছোট স্ক্রিনের জন্য এবং আরেকটি বড় স্ক্রিনের জন্য। আরও জানতে ভিন্ন ভিন্ন ডিভাইসকে সাপোর্ট করা ক্লাসটি দেখুন।



ফিগার ১. লেআউটের মধ্যে ব্রাঞ্চ থেকে ViewGroup অবজেক্টস এবং অন্য View অবজেক্ট ধারন।

এই অনুশীলনীতে, আপনি XML ল্যাঙ্গুয়েজে একটি লে আউট তৈরী করতে পারবেন যা একটি টেক্সটফিল্ড এবং একটি বাটন অন্তর্ভুক্ত করে। নিম্নোক্ত অনুশীলনীতে, বাটন চেপে টেক্সট ফিল্ড এর কনটেন্ট যখন অন্য একটিভিটিতে পাঠানো হবে তখন আপনি একে স্বগত জানাবেন।

লিনিয়ার লে আউট তৈরী করুন

res/layout/ ডিরেক্টরী থেকে activity_main.xml ফাইল ওপেন করুন।

নোট: ইক্লিপস এ যখন লে আউট ফাইল ওপেন করবেন, প্রথমেই আপনি গ্রাফিক্যাল লে আউট এডিটর দেখতে পারবেন। এই এডিটর দিয়ে আপনি WYSIWYG টুল ব্যবহার করে লেআউট তৈরী করতে পারবেন। এই অনুশীলনীতে আপনাকে সরাসরি XML এর সাথে কাজ করতে হবে, সুতরাং XML এডিটর ওপেন করতে স্ক্রিনের একদম নীচে Activity_main.xml ট্যাবে ক্লিক করুন। যখন আপনি RelativeLayout রুট ভিউ এবং TextView চাইল্ড ভিউ এর সাথে activity_main.xml ফাইল সহ এই প্রজেক্ট তৈরী করবেন, আপনাকে একটি ব্লাক্স এ্যাকটিভিটি টেমপ্লেট তৈরী করতে হবে। প্রথমেই, ি উপাদান ডিলিট করুন এবং কে এ পরিবর্তন করুন। তারপর android:orientation এট্রিবিউট এ্যাড করুন এবং "horizontal" এ এটা সেট করুন। ফলাফলটা এরকম দেখাবে:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >
</LinearLayout>
```

LinearLayout হচ্ছে ভিউ গ্রুপ (ভিউ গ্রুপের সাব ক্লাস) যা চাইল্ড ভিউ লে আউট ভার্টিক্যাল বা হরাইজন্টাল ওরিয়েন্টেশনে তৈরী করে, যেভাবে এট্রিবিউট দ্বারা সুনির্দিষ্ট করা হয়েছে। LinearLayout এর প্রতিটা চাইল্ড ক্রমানুসারে স্ক্রিনে দেখা যাবে যেখানে XML এর মধ্যে দৃশ্যমান হবে।

অন্য দুটি বৈশিষ্ট্য, তাদের আকার সুনির্দিষ্ট করার জন্য android:layout width এবং android:layout height সকল ধরনের ভিউ এর জন্য দরকার।

কারণ LinearLayout হচ্ছে লেআউটের মধ্যে রুট ভিউ, এটার স্ক্রিনের যে আয়তন সম্পূর্ণভাবে পূর্ণ করা উচিত যা অ্যাপে পাওয়া যাবে যা "সধঃপযথত্ধৎবহঃ" এ উচ্চতা এবং প্রস্থ এর সেটিং এর মাধ্যমে করা হয়েছে। এই ভ্যালু এটাই নিশ্চিত করে যে প্যারেন্ট ভিউ এর উচ্চতা ও প্রস্থ এর সাথে ম্যাচ করার জন্য ভিউটির উচিত এর দৈর্ঘ্য এবং প্রস্থের আয়তন বৃদ্ধি করা।

লেআউট প্রপারটিস বিষয়ক আরও তথ্যের জন্য Layout গাইড দেখুন (<http://developer.android.com/guide/topics/ui/declaring-layout.html>)।

টেক্সট ফিল্ড এ্যাড (সংযোজন) করুন

ইউজার কর্তৃক এডিটযোগ্য টেক্সট ফিল্ড তৈরী করতে, এর মধ্যে একটি উপাদান যোগ করুন।

প্রতিটা View অবজেক্ট এর মতোই, EditText অবজেক্ট এর প্রপারটিসকে সুনির্দিষ্ট করতে নির্দিষ্ট XML এট্রিবিউট নির্ধারণ করা উচিত। কীভাবে উপাদানের ভিতরে এটাকে আপনার নিশ্চিত করা উচিত তা এখানে আছে:

```
<EditText android:id="@+id/edit_message"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
```

রিসোর্স অবজেক্টস সম্পর্কে কিছু কথা

সহজ ভাষায় রিসোর্স অবজেক্ট হচ্ছে একটি স্বতন্ত্র ইন্টজার (integer) নাম যা একটা অ্যাপ রিসোর্স এর সঙ্গে যুক্ত, যেমন বিটম্যাপ, লেআউট ফাইল, বা স্ট্রিং। প্রতিটা রিসোর্সে আপনার প্রজেক্টের gen/r.java ফাইলের মধ্যে অনুরূপ রিসোর্স কে নির্ধারণ করে দেয়। আপনার রিসোর্স এ উল্লেখ করতে জ ক্লাসে অবজেক্ট নাম ব্যবহার করতে পারেন, যেমন আপনার যখন android:hint এট্রিবিউট এর জন্য স্ট্রিং ভ্যালু সুনির্দিষ্ট করার প্রয়োজন হবে। আপনি একটি নিয়মবহির্ভূত রিসোর্স আইডি তৈরী করতে android.id এট্রিবিউট ব্যবহার করে একটি ভিউ এর সাথে সংযুক্ত করতে পারেন যা আপনাকে অন্য কোড থেকে ওই ভিউ কে রেফার করতে অনুমোদন করে।

আপনি যখন আপনার অ্যাপ কমপাইল করেন তখন প্রতিবারই এসডিকে টুল R.java তৈরী করে। আপনাকে কখনই নিজ হাতে এই ফাইলকে পরিবর্তন করাতে হবে না।

আরও তথ্যের জন্য Providing Resources গাইড পড়ুন (লিংক: <http://developer.android.com/guide/topics/resources/providing-resources.html>)

এই বিশেষ গুন সমূহ সম্পর্কে:

android.id

এটা ভিউ এর জন্য একটি ইউনিক আইডেন্টিফায়ার সরবরাহ করে, যা আপনি আপনার অ্যাপ কোড থেকে অবজেক্টকে রেফারেন্স হিসাবে ব্যবহার করতে পারেন, যেমন অবজেক্ট পড়তে বা সংশোধন করতে (আপনি এটা পরবর্তী অনুশীলনীতে পারেন)

যখন আপনি XML থেকে যে কোন রিসোর্স অবজেক্ট উল্লেখ করবেন তখন (@) চিহ্নটি দরকার হবে। এটা রিসোর্স টাইপ(এই ক্ষেত্রে id), স্ল্যাশ এবং তারপর রিসোর্স নাম (edit_message) কে অনুসরণ করে।

(+) চিহ্ন টি রিসোর্স টাইপ এর পূর্বে থাকা প্রয়োজন শুধুমাত্র যখন আপনি প্রথমবারের মতো একটি রিসোর্স আইডি নির্ধারন করবেন। আপনি যখন অ্যাপ কম্পাইল করবেন, এসডিকে টুলস আপনার প্রজেক্টের gen/R.java ফাইল এর মধ্যে নতুন রিসোর্স আইডি তৈরী করতে আইডি নাম ব্যবহার করে যা EditText উপাদান কে রেফার করে। একবার যদি এভাবে রিসোর্স আইডি কে ঘোষণা করা হয়, তাহলে আইডির অন্য রেফারেন্স এর জন্য প্লাস সাইন ব্যবহার করতে হবে না। প্লাস সাইন শুধুমাত্র তখনই দরকার শুধুমাত্র যখন একটি নতুন রিসোর্স আইডি কে নির্দিষ্টকরন করা হয় এবং কংক্রিট রিসোর্স যেমন স্ট্রিং বা লেআউট এর জন্য এর দরকার নেই।

```
android:layout width এবং android:layout height
```

উচ্চতা ও প্রস্থের জন্য সুনির্দিষ্ট সাইজ ব্যবহার করার পরিবর্তে, "wrap_content" মান সুনির্দিষ্ট করে ভিউ এর কন্টেন্ট এর সাথে খাপ খেতে যতটুকু বড় হওয়া প্রয়োজন ভিউ এর ঠিক ততটুকু বড় হওয়া উচিত। আপনি যদি এর পরিবর্তে "match_parent", ব্যবহার করে থাকেন, তখন EditText উপাদান স্ক্রিনটা পূর্ণ করবে, কারন এটা প্যারেন্ট LinearLayout এর সাইজের সাথে ম্যাচ করতে পারে। আরও জানতে দেখুন Layouts গাইড (<http://developer.android.com/guide/topics/ui/declaring-layout.html>)।

```
android:hint
```

প্রদর্শন করার জন্য এটা একটি ডিফল্ট স্ট্রিং যখন টেক্সট ফিল্ড খালি থাকে। ভ্যালু হিসাবে হার্ড কোডেড স্ট্রিং এর ব্যবহারের পরিবর্তে, "@string/edit_message" ভ্যালু একটি আলাদা ফাইলে নির্ধারিত স্ট্রিং রিসোর্স রেফার করে। কারন এটা একটা কংক্রিট রিসোর্স রেফার করে (শুধুমাত্র আইডেন্টিফায়ার নয়), এর জন্য কোন প্লাস দরকার হবে না। যাইহোক, যেহেতু আপনি এখন পর্যন্ত স্ট্রিং রিসোর্স কে নির্ধারন করতে পারেন নি, আপনি প্রথমেই একটা কম্পাইল এরর দেখতে পারবেন।

নোট: এই স্ট্রিং রিসোর্স এর এলিমেন্ট আইডি এর মতোই একই নাম: edit_message। যাহোক রিসোর্স রেফারেন্স সবসময়ই রিসোর্স টাইপ দ্বারা সুবিধা প্রাপ্ত হয়(যেমন id অথবা string), সুতরাং একই নাম কোন সংঘাত তৈরী করে না।

স্ট্রিং রিসোর্স এ্যাড(সংযোজন) করুন

যখন আপনার ইউজার ইন্টারফেসে টেক্সট এ্যাড করার দরকার হবে তখন সবসময় প্রতিটা স্ট্রিং কে রিসোর্স এর মতো করে নির্ধারন করা উচিত। স্ট্রিং রিসোর্স একটি সিঙ্গেল লোকেশনে সকল ইউজার ইন্টারফেস কে পরিচালনা করতে অনুমোদন করে। স্ট্রিং এক্সটার্নালাইজিং আপনার অ্যাপকে প্রতিটা স্ট্রিং রিসোর্স এর জন্য বিকল্প সংজ্ঞা প্রদান করে ভিন্ন ভিন্ন ল্যাঙ্গুয়েজে লোকলাইজ করতেও অনুমোদন করে।

বাই ডিফল্ট, আপনার অ্যান্ড্রয়েড প্রজেক্ট res/values/strings.xml এ একটি স্ট্রিং রিসোর্স ফাইল অন্তর্ভুক্ত করে। "edit_message" নামে একটা নতুন স্ট্রিং এ্যাড করুন এবং "Enter a message" এ ভ্যালুটি সেট করুন ("hello_world" স্ট্রিং টি আপনি ডিলিট করে দিতে পারেন)।

আপনি যখন এই ফাইলে অবস্থান করবেন, বাটনের জন্য একটি "Send" স্ট্রিং এ্যাড করতে পারেন যা আপনাকে খুব শিঘ্রই এ্যাড করতে হবে, যার নাম "button_send"। strings.xml এর জন্য ফলাফলটা এমন হবে:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">My First App</string>
    <string name="edit_message">Enter a message</string>
    <string name="button_send">Send</string>
    <string name="action_settings">Settings</string>
    <string name="title_activity_main">MainActivity</string>
</resources>
```

স্ট্রিং রিসোর্স ব্যবহার করে আপনার অ্যাপ অন্যান্য ল্যাঙ্গুয়েজে লোকলাইজ করা সম্পর্কিত আরও তথ্য জানতে, ভিন্ন ভিন্ন ডিভাইস সাপোর্ট করা ক্লাসটি দেখুন।

বাটন এ্যাড (সংযোজন) করুন

এলিমেন্ট কে তাৎক্ষনিক ভাবে অনুসরণ করে এখন লেআউটে < Button > এ্যাড করুন

```
<Button  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/button_send" />
```

উচ্চতা এবং প্রস্থ "wrap_content" তে সেট করা হয় যাতে বাটনটি বাটন টেক্সট এর সাথে খাপ খেতে যততটুকু বড় হওয়া প্রয়োজন ঠিক ততটুকু বড় হয়। এই বাটনের android:id এট্রিবিউটের দরকার নাই, কারন একটিভিটি কোড থেকে রেফারেন্সড হবে না।

স্ক্রিন প্রস্থ ভরাট করার মতো করে ইনপুট বক্স তৈরী করুন

লেআউট ইদানিংকালে ডিজাইন করা হয়েছে যাতে EditText এবং Button উভয় উইডজিট তার উপাদানগুলোর সাথে খাপ খাওয়ার মতো যথেষ্ট বড় হয়. ফিগার ২ তে যেভাবে দেখানো হয়েছে।



ফিগার ২. EditText এবং Button উইজিট এর "wrap_content" এ তাদের প্রস্থ সেট আছে।

এটা বাটনের জন্য ভালো কাজ করে কিনুত টেক্সট ফিল্ড এর জন্য ততটা ভালো নয়, কারন ব্যবহারকারীরা বড় আকারের কিছু টাইপ করতে পারে। সুতরাং অব্যবহৃত স্ক্রিন প্রস্থ কে টেক্সট ফিল্ড দিয়ে পূর্ণ করাটা ভালো হবে। আপনি এটা করতে পারবেন এর LinearLayout মধ্যে *weight* প্রপার্টি দিয়ে, যা android:layout_weight এট্রিবিউট ব্যবহার করে এটাকে সুনির্দিষ্ট করতে পারবেন।

ওয়েট ভ্যালু হচ্ছে একটা সংখ্যা যা প্রতিটা ভিউ এর জন্য যে পরিমান স্পেস আছে তার কতটুকু ব্যয় করা উচিত তার পরিমান সুনির্দিষ্ট করে দেয়, অনুরূপভাবে সিবিং ভিউ এর ক্ষেত্রে কতটা স্পেস ব্যয় করা হবে সেটাও। এই কাজ এভাবে বোঝা যায় যে একটা পানীয়র মিশ্রনে কোনটা কি পরিমান আছে: দুই অংশ ফলের রস এবং এক অংশ পানি, এর অর্থ পানীয়ের দুই-তৃতীয়াংশ ফলের রস আছে। উদাহরন হিসাবে, আপনি যদি একটা ভিউ এর ওয়েট ২ ধরেন এবং আরেকটির জন্য ১ ধরেন তাহলে এর যোগফল হবে ৩, সুতরাং প্রথম ভিউ বিদ্যমান স্পেসের দুই-তৃতীয়াংশ পূর্ণ করে আছে এবং দ্বিতীয় ভিউ বাকী অংশ দখল করে। আপনি যদি তৃতীয় ভিউ যোগ করেন এবং এটার ওয়েট ১ নির্ধারন করেন তখন প্রথম ভিউ (যার ওয়েট ২) বিদ্যমান স্পেসের অর্ধেক দখল করবে, বাকী দুইটা ভিউ এক চতুর্থাংশ করে স্থান দখল করবে।

সকল ভিউ এর ডিফল্ট ওয়েট হচ্ছে ০, সুতরাং আপনি যদি কোন ওয়েট ভ্যালু ০ এর চেয়ে বেশী করে নির্দিষ্ট করে দেন, তখন ওই ভিউ তার যেটুকু দরকার তা দেয়া সত্ত্বেও যতটুকু স্পেস বাকী আছে তার সবটুকুই পূরণ করে নেয়। সুতরাং EditText এলিমেন্ট দিয়ে আপনার লেআউটের বিদ্যমান স্পেস পূরণ করার জন্য এটার ওয়েট ১ দিন এবং বাটনটিকে কোন ওয়েট না দিয়েই ছেড়ে দিন।

```
<EditText
    android:layout_weight="1"
    ... />
```

ওয়েট সুনির্দিষ্ট করার সময় আপনার লেআউটকে কার্যকর করতে, আপনার উচিত EditText এর প্রস্থর আকার শূন্যতে (Zero)পরিবর্তন করা (0dp)। প্রস্থর সেটিং জিরো (0dp) লেআউটের পারফরমেন্সকে উন্নত করে কারন "wrap_content" কে প্রস্থ হিসাবে ব্যবহার করতে সিস্টেমটিকে প্রস্থর হিসাবনিকাশ করার দরকার হয়, যা শেষ পর্যন্ত অপ্রাসঙ্গিক কারন বিদ্যমান স্পেসকে পূরন করতে ওয়েট ভ্যালুকে আরেকটি প্রস্থ হিসাব করতে হয়।

```
<EditText
```



```
android:layout_weight="1"
android:layout_width="0dp"
... />
```

ফিগার ৩ দেখাচ্ছে যে যখন আপনি সকল ওয়েট EditText এলিমেন্টে নিয়োজিত করেন তার ফলাফল কি।



ফিগার ৩. EditText উইজিট সকল লেআউট ওয়েট প্রদান করে, LinearLayout এর মধ্যে বাকী স্পেস পূর্ণ করুন।

এখানে দেখা যাচ্ছে একটি সম্পূর্ণ লেআউট ফাইল দেখতে কেমন হবে:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">
<EditText android:id="@+id/edit_message"
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:hint="@string/edit_message" />
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send" />
</LinearLayout>
```

এই লেআউট ডিফল্ট Activity ক্লাস দ্বারা প্রয়োগ করা হয় যা এসডিকে টুলস তৈরী করেছিল, যখন আপনি এই প্রজেক্ট তৈরী করেছিলেন, সুতরাং আপনি এখন অ্যাপ রান করে এর ফলাফল দেখতে পারেন।

* ইক্লিপস এ, টুলবার থেকে Run  এ ক্লিক করুন।

* অথবা কমান্ড লাইন থেকে আপনার অ্যান্ড্রয়েড প্রজেক্ট রুট ডিরেক্টরী পরিবর্তন করুন এবং এরপর করুন:

```
ant debug
adb install bin/MyFirstApp-debug.apk
```

পরবর্তী অনুশীলনীতে, বাটন প্রেস করতে কীভাবে রেসপন্ড করবেন, টেক্সট ফিল্ড থেকে কীভাবে কন্টেন্ট পড়বেন, অন্য কর্মকাল্ড শুরু করবেন সেই কাজগুলো এবং অন্যান্য কাজ শেখানো হবে।

অন্য একটিভিটি শুরু করা

(<http://developer.android.com/training/basics/firstapp/starting-activity.html#RespondToButton>)

পূর্ববর্তী অনুশীলনী শেষ করার পর আপনি যে অ্যাপ পাবেন যা একটি টেক্সট ফিল্ড এবং একটি বাটন সহ একটি অ্যাকটিভিটি দেখাবে (একটি সিঙ্গেল স্ক্রিন)। এই অনুশীলনীতে MainActivity তে কিছু কোড এ্যাড করতে পারবেন যা নতুন কাজ শুরু করবে যখন ব্যবহারকারী সেন্ড বাটন এ ক্লিক করবে।

সেন্ড বাটনকে রেসপন্স করা

বাটনের অন ক্লিক ইভেন্ট এ রেসপন্স করতে, activity_main.xml লেআউট ফাইল ওপেন করুন এবং < Button > এলিমেন্ট এ android:onClick এট্রিবিউট এ্যাড করুন:

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

android:onClick এট্রিবিউট এর ভ্যালু "sendMessage" হচ্ছে আপনার একটিভিটির একটা পদ্ধতির নাম, ব্যবহারকারী যখন বাটনে ক্লিক করে সিস্টেম তখন এটাকে কল করে।

MainActivity ক্লাস ওপেন করুন (প্রজেক্টের src/ ডিরেক্টরীতে আছে) এবং সমরূপ মেথড এ্যাড করুন।

```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    // Do something in response to button
}
```

এর জন্য দরকার View ক্লাস নিয়ে আসা দরকার।

```
import android.view.View;
```

টিপ: ইক্লিপস এ Ctrl + Shift + O প্রেস করে মিসিং ক্লাস আনতে হয় (Mac এ Cmd + Shift + O)

android:onClick এ যে পদ্ধতির নাম দেওয়া হয়েছে তার সাথে সিস্টেম এর ম্যাচ করার জন্য কিছু বৈশিষ্ট্য যেভাবে দেখানো হয়েছে ঠিক সেভাবে হতে হবে। সুনির্দিষ্টভাবে পদ্ধতিটি অবশ্যই -

- Be public/ উন্মুক্ত হতে হবে
- Have a void return value ঐধাব ধ াড়রফ ৎবঃৎহ াধষঁব/ ভয়েড রিটার্ন ভ্যালু থাকতে হবে
- একমাত্র প্যারামিটার হিসাবে একটা View থাকতে হবে (এটা হবে সেই ভিউ যেটাতে ক্লিক করা হবে)

পরবর্তীতে, আপনি টেক্সট ফিল্ড এর কনটেন্ট পড়তে এবং ওই টেক্সট অন্য কাজে প্রদান করে এই পদ্ধতি পছন্দ করবেন।

ইনটেন্ট তৈরী করা

একটি Intent হচ্ছে একটা অবজেক্ট যা দুটো পৃথক বিষয়ের (যেমন দুটো একটিভিটি) মধ্যে রানটাইম বাইন্ডিং প্রদান করে। Intent একটা অ্যাপের “কোন কিছুর করার পরিকল্পনা” কে চিত্রায়িত করে। আপনি সুবিস্তৃত বহুবিধ কাজে ইনটেন্ট ব্যবহার করতে পারেন তবে বেশীরভাগ সময় এটা অন্য কর্মকান্ড শুরু করার কাজে ব্যবহার করা হয়ে থাকে।

DisplayMessageActivity নামের কর্মকান্ড শুরু করতে sendMessage() পদ্ধতির মধ্যে একটা ইনটেন্ট তৈরী করুন:

```
Intent intent = new Intent(this DisplayMessageActivity.class);
```

কনস্ট্রাক্টর এখানে দুইটা প্যারামিটার ব্যবহার করেছেন:

- একটা Context হচ্ছে এটার প্রথম প্যারামিটার (এটা ব্যবহার করা হয় কারন Activity ক্লাস হচ্ছে Context এর একটি সাব ক্লাস)
- অ্যাপ উপাদানের Class যেখানে সিস্টেমের উচিত Intent ডেলিভারি দেওয়া (এক্ষেত্রে একটিভিটি টির শুরু হওয়া উচিত)

একটি ইনটেন্ট অন্য অ্যাপে পাঠানো

এই অনুশীলনীতে যে ইনটেন্ট তৈরী করা হয়েছে যা এক্সপ্লিসিট ইনটেন্ট (*explicit intent*) হিসাবে বিবেচিত, কারন Intent টি যথাযথ অ্যাপ কম্পোনেন্ট কে সুনির্দিষ্ট করে দেয় যেখানে ইনটেন্ট দেয়া উচিত। যাহোক ইনটেন্ট আবার ইমপ্লিসিট (*implicit*) ও হতে পারে, এখানে কাঙ্ক্ষিত কম্পোনেন্ট কে সুনির্দিষ্ট করে দেয় না, কিনুত ইনটেন্ট রেসপন্স করতে ডিভাইসে যে কোন অ্যাপ ইনস্টল করতে অনুমোদন করে যতক্ষন না এটা এই কাজের জন্য মেটা-ডেটা স্পেসিফিকেশন কে সনুতষ্ট করে যা বিভিন্ন Intent এ সুনির্দিষ্ট। আরও জানতে অন্য অ্যাপের সাথে পারস্পরিক যোগাযোগ করা সম্পর্কিত ক্লাস দেখুন।

নোট: যদি আপনি আইডিই যেমন ইক্লিপস ব্যবহার করে থাকেন তাহলে isplayMessageActivity রেফারেন্স একটি এরর তৈরী করতে পারে, কারন ক্লাসটি এখন পর্যন্ত অস্তিত্বশীল নয়। আপাতত এররটি উপেক্ষা করুন; আপনি শিঘ্রই ক্লাসটি তৈরী করতে পারবেন।

একটি ইনটেন্ট শুধুমাত্র অন্য আরেকটি একটিভিটি শুরু করা কে অনুমোদনই করে না, একটিভিটিতে একগুচ্ছ ডাটাও নিয়ে আসে। sendMessage() পদ্ধতির ভিতরে EditText এলিমেন্ট পেতে findViewById() ব্যবহার করুন এবং ইনটেন্ট এ এর টেক্সট ভ্যালু এ্যাড করুন:

```
Intent intent= new Intent(this, DisplayMessageActivity.class);  
EditText editText = (EditText) findViewById(R.id.edit_message);
```

```
String message = editText.getText().toString();  
intent.putExtra(EXTRA_MESSAGE, message);
```

নোট: android.content.Intent এবং android.widget.EditText এর জন্যে আপনার স্টেটমেন্ট আনার দরকার। আপনি এক মুহুর্তে EXTRA_MESSAGE কনস্ট্যান্স কে নির্ধারন করতে পারবেন।

একটি Intent কি (Key)ভ্যালু পেয়ার হিসাবে বিভিন্ন ডাটা টাইপস নিয়ে আসতে পারে যাকে *extras* বলে। putExtra() পদ্ধতি প্রথম প্যারামিটারে কি (Key) নেম নেয় এবং দ্বিতীয় প্যারামিটারে কি (Key) ভ্যালু নিয়ে থাকে।

পরবর্তী এ্যাক্টিভিটি এর জন্য এক্সট্রা ডাটা অনুসন্ধান করতে পাবলিক কনস্ট্যান্স ব্যবহার করে আপনার ইনটেন্ট এর এক্সট্রার জন্য কি (Key) নির্ধারন করে দেওয়া উচিত। সুতরাং MainActivity ক্লাসের সবচেয়ে উপরে Extra_Message ডেফিনেশন এ্যাড করে দিন:

```
public class MainActivity extends Activity {  
    public final static String EXTRA_MESSAGE = "com.example.myfirstapp.MESSAGE";  
    ...  
}
```

প্রিফিক্স হিসাবে আপনার অ্যাপ এর প্যাকেজ নাম ব্যবহার করে ইনটেন্ট এক্সট্রার জন্য কি (Keys) নির্ধারন করা সাধারনত একটি ভালো চর্চা। এটা নিশ্চিত করে যে অন্যান্য অ্যাপের সাথে পারস্পরিক ক্রিয়ার ক্ষেত্রে তারা স্বতন্ত্র।

দ্বিতীয় কার্যক্রম(একটিভিটি) শুরু করুন

একটা একটিভিটি শুরু কর তে startActivity() কে আহবান করুন এবং আপনার Intent এ প্রবেশ করান। সিস্টেম এই আহবান গ্রহন করবে এবং Intent দ্বারা নির্ধারিত Activity এর একটি ইনস্ট্যান্স শুরু করে।

সেন্ড বাটন কর্তৃক ডেকে আনানতুন কোডের সাথে সম্পূর্ণ sendMessage() টি এখন দেখতে নিম্নরূপ:

```
/** Called when the user clicks the Send button */
public void sendMessage(View view) {
    Intent intent = new Intent(this, DisplayMessageActivity.class);
    EditText editText = (EditText) findViewById(R.id.edit_message);
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);
    startActivity(intent);
}
```


এখন আপনার এই কাজ করতে DisplayMessageActivity ক্লাস তৈরি করা প্রয়োজন

দ্বিতীয় একটিভিটি তৈরী করুন



ফিগার ১. ইক্লিপসে নতুন একটিভিটি উইজার্ড।

ইক্লিপস ব্যবহার করে নতুন একটিভিটি তৈরী করতে:

1. টুলবার থেকে **New**  তে ক্লিক করুন
2. উইন্ডোতে যা দেখা যাবে তার থেকে **Android** ফোল্ডার ওপেন করুন এবং **Android Activity** নির্বাচিত করুন। তারপর **Next** ক্লিক করুন।
3. **Blank Activity** নির্বাচিত করুন এবং **Next** এ ক্লিক করুন।
4. একটিভিটির বিস্তারিত পূরন করুন:

- **Project:** MyFirstApp
- **Activity Name:** DisplayMessageActivity
- **Layout Name:** activity_display_message
- **Title:** My Message
- **Hierarchial Parent:** com.example.myfirstapp.MainActivity
- **Navigation Type:** None

Finish এ ক্লিক করুন.

আপনি যদি ভিন্ন আইডিই বা কমান্ড লাইন টুলস ব্যবহার করে থাকেন তাহলে প্রজেক্ট এর src/ ডিরেক্টরীতে DisplayMessageActivity.java নামে একটি নতুন ফাইল তৈরী করুন, যেটা MainActivity.java আসল ফাইলের কাছাকাছি

এই displayMessageActivity.java ফাইলটি ওপেন করুন। আপনি যদি এই একটিভিটি তৈরী করতে ইক্লিপস ব্যবহার করে থাকেন:

* ক্লাস ইতিমধ্যে কাঙ্ক্ষিত onCreate() পদ্ধতির একটি বাস্তবায়ন অন্তর্ভুক্ত করেছে

* এখানে onCreateOptionsMenu() পদ্ধতির একটি বাস্তবায়নও অন্তর্ভুক্ত করেছে। কিন্ত এটা এই

অ্যাপের জন্য আপনার দরকার হবে না সুতরাং আপনি এটা মুছে ফেলতে পারেন

* এখানে `onOptionsItemSelected()` পদ্ধতির একটি বাস্তবায়নও অন্তর্ভুক্ত করেছে যা একশন বারের আপ (টু) বিহেভিয়ার এর জন্য আচরন নিয়ন্ত্রণ করে। এটাকে ধরে রাখার এটা একটা উপায়।

কারণ ActionBar APIs শুধুমাত্র HONEYCOMB (API level 11) এবং এর চেয়ে উন্নত সংস্করণে আছে, বর্তমান প্ল্যাটফর্ম ভার্সন চেক করার জন্য আপনাকে `getActionBar()` পদ্ধতির চারপাশে অবশ্যই একটি পরিস্থিতি এ্যাড করতে হবে। lint এরর এড়াতে আপনাকে অবশ্যই `onCreate()` পদ্ধতিতে `@SuppressWarnings("NewApi")` ট্যাগ এ্যাড করতে হবে। `DisplayMessageActivity` ক্লাসটি এখন দেখতে এই রকম হওয়া উচিত:

```
public class DisplayMessageActivity extends Activity {

    @SuppressWarnings("NewApi")
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_display_message);

        // Make sure we're running on Honeycomb or higher to use ActionBar APIs
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
            // Show the Up button in the action bar.
            getActionBar().setDisplayHomeAsUpEnabled(true);
        }
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case android.R.id.home:
                NavUtils.navigateUpFromSameTask(this);
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

আপনি যদি ইক্লিপস ছাড়া অন্য আইডিই ব্যবহার করে থাকেন তাহলে উপরের কোড দ্বারা আপনার `DisplayMessageActivity` ক্লাস আপডেট করে নিন।

সকল Activity এর সাবক্লাসের `onCreate()` পদ্ধতিটি অবশ্যই বাস্তবায়ন করা উচিত। সিস্টেম এটাকে কল করে যখন একটিভিটি এর নতুন একটা উদাহরণ তৈরী করবে। এই মেথডে আপনাকে অবশ্যই পদ্ধতি দিয়ে একটিভিটি লেআউট নির্ধারন করতে হবে এবং যেখানে আপনাকে একটিভিটি কম্পোনেন্ট এর জন্য প্রারম্ভিক সেটআপ দিতে হবে।

নোট: আপনি যদি ইক্লিপস ছাড়া অন্য আইডিই ব্যবহার করে থাকেন তাহলে আপনার প্রজেক্টে `activity_display_message` লেআউট থাকবে না যা `setContentView()` খুজে থাকে। এটা ঠিক আছে কারণ আপনি এই মেথড/পদ্ধতি পরবর্তীতে আপডেট করবেন এবং আপনি ওই লেআউট ব্যবহার করছেন না।

টাইটেল স্ট্রিং এ্যাড করুন

আপনি যদি ইক্লিপস ব্যবহার করে থাকেন, পরবর্তী সেকশন এড়িয়ে চলে যেতে পারেন, কারণ টেমপ্লেটটি নতুন একটিভিটির জন্য টাইটেল স্ট্রিং সরবরাহ করে থাকে।

আপনি যদি ইক্লিপস ছাড়া অন্য আইডিই ব্যবহার করে থাকেন তাহলে strings.xml ফাইলে নতুন একটিভিটির টাইটেল এ্যাড করে দিন

```
<resources>
...
    <string name="title_activity_display_message">My Message</string>
</resources>
```

এটাকে মেনিফেস্ট এ্যাড করুন

একটিভিটি এলিমেন্ট ব্যবহার করে অবশ্যই আপনার মেনিফেস্ট ফাইলে `AndroidManifest.xml`, সকল ফাইল ডিক্লেয়ার হওয়া উচিত।

একটিভিটি তৈরী করতে আপনি যখন ইক্লিপস টুলস ব্যবহার করবেন তখন এটা একটা ডিফল্ট এন্ট্রি তৈরী করবে। আপনি যদি ভিন্ন আইডিই ব্যবহার করে থাকেন, আপনার নিজেকেই মেনিফেস্ট এন্ট্রি এ্যাড করতে হবে। এটা দেখতে এরকম হবে:

```
<application ... >
...
<activity
    android:name="com.example.myfirstapp.DisplayMessageActivity"
    android:label="@string/title_activity_display_message"
    android:parentActivityName="com.example.myfirstapp.MainActivity" >
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.myfirstapp.MainActivity" />
</activity>
</application>
```

`android:parentActivityName` এট্রিবিউট টি অ্যাপের লজিক্যাল হায়ারার্কি এর মধ্যে থেকে এই এক্টিভিটির প্যারেন্ট এক্টিভিটির নাম ঘোষণা করে। সিস্টেম ডিফল্ট নেভিগেশন আচরন বাস্তবায়ন করতে এই ভ্যালু ব্যবহার করে থাকে, যেমন অ্যান্ড্রয়েড ৪.১ (API level 16) বা এর পরবর্তী ভার্সনে `Up navigation`। আপনি একই নেভিগেশন বিহেভিয়ার অ্যান্ড্রয়েডের পুরানো সংস্করণে দিতে পারেন, `Support Library` ব্যবহার করে এবং এলিমেন্ট এ্যাড করে, যেভাবে এখানে দেখানো হয়েছে।

নোট: আপনার অ্যান্ড্রয়েড এসডিকে ইতিমধ্যে সর্বশেষ অ্যান্ড্রয়েড সাপোর্ট লাইব্রেরী তে অন্তর্ভুক্ত হওয়ার কথা। এটা এডিটি বান্ডল এর সাথে অন্তর্ভুক্ত কিনুত আপনি যদি ভিন্ন আইডিই ব্যবহার করে থাকেন, তাহলে `adding Platforms and Packages` ধাপেই আপনাকে এটা ইনস্টল করতে হতো। যখন ইক্লিপস এ টেমপ্লেট ব্যবহার করা হয় তখন সাপোর্ট লাইব্রেরী সয়ংক্রিয়ভাবে আপনার অ্যাপ প্রজেক্টে এ্যাড হয়ে যায় (আপনি *android Dependencies* তালিকায় লাইব্রেরীর JAR ফাইল দেখতে পারেন)। আপনি যদি ইক্লিপস ব্যবহার না করে থাকেন, আপনাকে ম্যানুয়ালী আপনার প্রজেক্টে লাইব্রেরী এ্যাড করতে হবে - `setting up the Support Library` (<http://developer.android.com/tools/support-library/setup.html>) গাইড অনুসরণ করুন তারপর আবার এখানে ফিরে আসুন।

আপনি যদি ইক্লিপস দিয়ে ডেভেলপ করে থাকেন তাহলে আপনি এখন অ্যাপ রান করতে পারেন, কিনুত বেশী কিছু হবে না। সেল্ফ বাটনে ক্লিক করে দ্বিতীয় এক্টিভিটি শুরু করুন কিনুত এটা টেমপ্লেট কর্তৃক প্রদত্ত ডিফল্ট "Hello World" লেআউট ব্যবহার করে। আপনি শিঘ্রই ডিসপ্লে কে কাস্টম টেক্সট ভিউ এ পরিবর্তন করে এক্টিভিটিকে আপডেট করবেন, সুতরাং আপনি যদি ভিন্ন আইডিই ব্যবহার করে থাকেন তাহলে চিন্তিত হবেন না যে আপনার অ্যাপ এখনো কম্পাইল হয় নাই।

ইনটেন্ট গ্রহণ করুন

প্রতিটা Activity কেই Intent আহ্বান করে থাকে, ব্যবহারকারী এটাকে কীভাবে পরিচালনা করে সেটা কোন বিষয় নয়। আপনি Intent টা পেতে পারেন যা getIntent()কে কল করে আপনার একটিভিটিকে শুরু করতে পারে এবং তথ্যকে পূরণরুদ্ধার করে এটার মধ্যে রেখে দেয়।

DisplayMessageActivity ক্লাসের onCreate() পদ্ধতির মধ্যে ইনটেন্ট খুজে নিন এবং MainActivity কর্তৃক প্রদত্ত মেসেজ টা সরিয়ে নিন।

```
Intent intent = getIntent();  
String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);
```

মেসেজ ডিসপ্লে করুন

স্ক্রিনে মেসেজ শো করতে একটা TextView উইডজিট (widget) তৈরী করুন এবং setText() ব্যবহার করে টেক্সটটি সেট করুন। setContentView() দিকে পাস করে দিয়ে একটিভিটি লেআউট এর রুট ভিউ হিসাবে TextView এ্যাড করুন।

DisplayMessageActivity এর জন্য সম্পূর্ণ onCreate() পদ্ধতিটি দেখতে এরকম হবে:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Get the message from the intent
    Intent intent = getIntent();
    String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

    // Create the text view
    TextView textView = new TextView(this);
    textView.setTextSize(40);
    textView.setText(message);

    // Set the text view as the activity layout
    setContentView(textView);
}
```

আপনি এখন অ্যাপ রান করতে পারবেন। যখন এটা ওপেন হবে, টেক্সট ফিল্ডে একটা মেসেজ টাইপ করুন, সেভ বাটনে ক্লিক করুন এবং মেসেজটি দ্বিতীয় একটিভিটিতে দৃশ্যমান হবে।



ফিগার ২. চূড়ান্ত অ্যাপের মধ্যে উভয় একটিভিটি অ্যান্ড্রয়েড ৪.০ তে রান করে।

আপনি আপনার প্রথম যে অ্যান্ড্রয়েড অ্যাপটি তৈরী করলেন এটাই সেটা!

একশন বার এ্যাড (সংযোজন) করা

(<http://developer.android.com/training/basics/actionbar/index.html>)

একশন বার হচ্ছে আপনার অ্যাপ এর একটিভিটি বাস্তবায়নের জন্য অন্যতম গুরুত্বপূর্ণ ডিজাইন উপাদান। এটা বেশ কিছু ইউজার ইন্টারফেস বৈশিষ্ট্য প্রদান করে যা অন্যান্য অ্যান্ড্রয়েড অ্যাপস এর মধ্যে সমন্বয় করে আপনার অ্যাপকে ব্যবহারকারীদের কাছে তাৎক্ষনিকভাবে জনপ্রিয় করে তুলবে। এর মূল কাজ সমূহ:

- আপনার অ্যাপকে একটা পরিচয় দেওয়ার জন্য একটি ডেডিকেটেড স্পেস এবং অ্যাপের মধ্যে ইউজার এর লোকেশন নির্দেশ করে।
- অনুমেয় উপায়ে গুরুত্বপূর্ণ কর্মকাণ্ডে অনুপ্রবেশ করতে পারা (যেমন সার্চ)
- নেভিগেশন এবং ভিউ পরিবর্তন করতে সহায়তা করে (ট্যাব বা ড্রপ-ডাউন লিস্ট সহ)



এই প্রশিক্ষণ ক্লাসে একশন বারের মৌলিক বিষয় দ্রুততার সাথে শেখার গাইড এর পরামর্শ দেয়। একশন বারের বিভিন্ন বৈশিষ্ট্য সম্পর্কে আরও জানতে দেখুন Action Bar (<http://developer.android.com/guide/topics/ui/actionbar.html>) গাইড।

এই অধ্যায়ের অনুশীলনী সমূহ

একশনবার সেটআপ করা

শিখুন আপনার একটিভিটিতে কীভাবে মৌলিক একশন বার এ্যাড করতে হয়, হতে পারে আপনার অ্যাপ শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চে উন্নত সংস্করণকে সাপোর্ট করে অথবা অ্যান্ড্রয়েড ২.১ এর মতো নীচের সংস্করণকেও সাপোর্ট করতে পারে (অ্যান্ড্রয়েড সাপোর্ট লাইব্রেরী ব্যবহার করে)

একশন বাটন এ্যাড (সংযোজন) করা

শিখুন, কীভাবে একশন বারের মধ্যে ইউজার একশন এ্যাড এবং রেসপন্স করতে হয়

একশন বার এর স্টাইল ঠিক করা

শিখুন কীভাবে একশন বার এর দৃশ্যমানতাকে কাস্টোমাইজ করতে হয়

একশন বার ওভারলে করা

শিখুন কীভাবে আপনার লেআউটের সামনে একশন বারকে ওভারলে করতে ত্রুটিহীন পরিবর্তন কে অনুমোদন করবে যখন একশন বার হাইড করা থাকে।

একশন বার সেট আপ করা

(<http://developer.android.com/training/basics/actionbar/setting-up.html>)

এর সবচেয়ে মৌলিক ধরন, একশন বার একটিভিটির জন্য টাইটেল এবং অ্যাপ আইকন বাম পাশে ডিসপ্লে করে। এমনকি এই সরল চেহারা, একশন বার সকল একটিভিটির জন্য উপকারী কেননা এটা ব্যবহারকারীকে জানায় যে তারা এখন কোন অবস্থায় আছে এবং নিয়মিত আপনার অ্যাপের পরিচয় বহন করার জন্য।



ফিগার ১. অ্যাপ আইকন এবং একটিভিটি টাইটেল সহকারে একটি একশন বার।

একটা মৌলিক একশন বার সেটআপ করতে দরকার কারন, আপনার অ্যাপ একটা একটিভিটি থিম ব্যবহার করবে যা একশন বার কে সক্ষম করবে। কীভাবে এই ধরনের থিম চাওয়া হয় তা নির্ভর করে আপনার অ্যাপ কর্তৃক অ্যান্ড্রয়েডের সর্বনিম্ন কোন ভার্সনকে সাপোর্ট করবে। সুতরাং এই অনুশীলনী দুইটা অধ্যায়ে ভাগ করা হয়েছে যে অ্যান্ড্রয়েডের সর্বনিম্ন কোন সংস্করণ পর্যন্ত সাপোর্ট করে।

শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণকে সাপোর্ট করা

অ্যান্ড্রয়েড ৩.০ (API level 11) দিয়ে শুরু করলে, একশন বার সকল একটিভিটির মধ্যে অন্তর্ভুক্ত থাকে যা Theme.Holo থিম (অথবা এর যে কোন একটা শাখা) ব্যবহার করে, যেটা একটা ডিফল্ট থিম, যখন targetSdkVersion অথবা minSdkVersion এটিবিউটের যে কোন একটা "১১" অথবা এর চেয়ে বেশীতে সেট করা হয়।

সুতরাং আপনার একটিভিটিতে একশন বার এ্যাড করতে, সোজাসুজি ১১ বা এর চেয়ে বেশীতে এটিবিউট সেট করুন। উদাহরন:

```
<manifest ... >
    <uses-sdk android:minSdkVersion="11" ... />
    ...
</manifest>
```

নোট: আপনি যদি কাস্টম থিম সেট করে থাকেন, নিশ্চিত হোন যে এটা যে কোন একটা Theme.Holo থিম এর প্যারেন্ট হিসাবে ব্যবহার করে।

এখন Theme.Holo থিম আপনার অ্যাপ এ ব্যবহৃত হচ্ছে এবং সকল একটিভিটি একশন বার দেখাচ্ছে।

অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণকে সাপোর্ট করে

যখন অ্যান্ড্রয়েড ৩.০ এর চেয়ে পুরাতন সংস্করণে (অ্যান্ড্রয়েড ২.১ পর্যন্ত) এটা রান করে তখন একশন বার এ্যাড করতে আপনার অ্যাপলিকেশনে অ্যান্ড্রয়েড সাপোর্ট লাইব্রেরী অন্তর্ভুক্ত করা দরকার।

শুরু করার সময় Support Library Setup তথ্য পড়ুন এবং v7 **appcompat** লাইব্রেরী সেটআপ করুন (কখনও যদি আপনি লাইব্রেরী প্যাকেজ ডাউনলোড করে থাকেন, তাহলে Adding libraries with resources নির্দেশিকা অনুসরণ করুন)

কখনও আপনার অ্যাপ প্রজেক্টের সাথে যদি অন্তর্ভুক্ত সাপোর্ট লাইব্রেরী থেকে থাকে:

1. আপনার একটিভিটি আপডেট করুন যাতে এটা ActionBarActivity কে বিসৃত করে।
উদাহরন:

```
public class MainActivity extends ActionBarActivity { ... }
```

2. আপনার মেনিফেস্ট ফাইলে যে কোন একটা Theme.AppCompat থিম ব্যবহার করার জন্য হয় উপাদান বা স্বতন্ত্র উপাদান আপডেট করুন। উদাহরন:

```
<activity android:theme="@style/Theme.AppCompat.Light" ... >
```

নোট: আপনি যদি কাস্টম থিম তৈরী করে থাকেন, নিশ্চিত হোন যে এটা যে কোন একটা Theme.AppCompat থিম এর প্যারেন্ট হিসাবে ব্যবহার করে। একশন বার স্টাইল করা অধ্যায়টি দেখুন।

এখন আপনার একটিভিটি একশনবার কে অন্তর্ভুক্ত করে নিয়েছে যখন অ্যান্ড্রয়েড ২.১ (APi level 7) এবং এর পরবর্তী সংস্করণে রান করে।

মেনিফেস্টে আপনার অ্যাপের এপিআই লেভেল সঠিকভাবে সেট করার কথা স্মরণ রাখবেন:

```
<manifest ... >  
    <uses-sdk android:minSdkVersion="7" android:targetSdkVersion="18" />  
    ...  
</manifest>
```

একশন বাটন এ্যাড (সংযোজন) করা

(<http://developer.android.com/training/basics/actionbar/adding-buttons.html>)

একশন বার আপনাকে অ্যাপসের বর্তমান প্রেক্ষাপটে সবচেয়ে গুরুত্বপূর্ণ একশন আইটেম এর জন্য বাটন এ্যাড করতে দেয়। ওইগুলো একশন বারে সরাসরি আইকন বা/এবং টেক্সট এর সাথে দৃশ্যমান হয় যাকে একশন বাটন (*action buttons*) বলা হয়। যে একশন, একশন বার এর সাথে থাপ খায় না বা তেমন গুরুত্বপূর্ণ নয় সেটা একশন ওভারফ্লোতে হিডেন থাকে।



ফিগার ১, সার্চ এবং একশন ছড়িয়ে দেয়ার জন্য একশন বাটন সহকারে একশন বার, যা অতিরিক্ত একশন উন্মোচন করে।

তগথ এ একশনকে সুনির্দিষ্ট করুন

একশন ওভারফ্লোর মধ্যে সকল একশন বাটন এবং অন্যান্য আইটেম একটি এক্সএমএল menu resource কর্তৃক নির্ধারিত হয়ে থাকে। একশন বারে একশন এ্যাড করতে আপনার প্রজেক্টের res/menu ডিরেক্টরীতে একটা নতুন এক্সএমএল ফাইল তৈরী করুন।

আপনি একশন বারে যে আইটেমগুলো অন্তর্ভুক্ত করতে চান তার প্রতিটার জন্য একটি এলিমেন্ট এ্যাড করুন। উদাহরন:

res/menu/main_activity_actions.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android" >
<!-- Search, should appear as action button -->
<item android:id="@+id/action_search"
      android:icon="@drawable/ic_action_search"
      android:title="@string/action_search"
      android:showAsAction="ifRoom" />
<!-- Settings, should always be in the overflow -->
<item android:id="@+id/action_settings"
      android:title="@string/action_settings"
      android:showAsAction="never" />
</menu>
```

একশন বার আইকন ডাউনলোড করুন

অ্যাক্সিয়েড iconography গাইডলাইন কে ভালোভাবে ম্যাচ করতে, আপনার Action Bar Icon Pack কর্তৃক প্রদেয় আইকন ব্যবহার করা উচিত।

(<http://developer.android.com/design/downloads/index.html#action-bar-icon-pack>)

এটা জানায় যে সার্চ একশনকে একশন বাটন হিসাবে দৃশ্যমান হওয়া উচিত, যখন একশন বারে যথেষ্ট জায়গা থাকে, কিনুত সেটিং একশন সবসময় ওভারফ্লোতে দৃশ্যমান হওয়া উচিত। (বাই ডিফল্ট, সকল একশন ওভারফ্লোতে দৃশ্যমান হয়, কিনুত প্রতিটা একশনের জন্যে আপনার ডিজাইনের উদ্দেশ্য স্পষ্টত জানানো হচ্ছে ভালো চর্চা)

একটি ইমেজের জন্য icon (আইকন) এট্রিবিউট এর রিসোর্স আইডি দরকার। আপনার প্রজেক্টের res/drawable/ ডিরেক্টরীতে @drawable/ নামে একটা বিটম্যাপ ইমেজ সেভ করেছেন। উদাহরণ হিসাবে, "@drawable/ic_action_search" G+K ic_action_search.png কে নির্দেশ করে। একই ভাবে Title এট্রিবিউট স্ট্রিং রিসোর্স ব্যবহার করে যা আপনার প্রজেক্টের res/values/ ডিরেক্টরীর একটা এক্সএমএল কর্তৃক নির্ধারিত, যা একটি সহজ ইউজার ইন্টারফেস তৈরী করা অধ্যায়ে আলোচিত হয়েছে।

নোট: যখন আপনার অ্যাপের জন্য আইকন এবং অন্যান্য বিটম্যাপ ইমেজ তৈরী করবেন, তখন এটা গুরুত্বপূর্ণ যে আপনি মাল্টিপল ভার্সন (ভিন্ন ভিন্ন সংস্করণ) প্রদান করবেন যা বিভিন্ন স্ক্রিন ঘনত্ব অনুসারে তৈরী করা। এটা ভিন্ন ভিন্ন স্ক্রিনকে সাপোর্ট করা অনুশীলনীতে বিস্তারিত আলোচনা করা হয়েছে।

আপনার অ্যাপ যদি অ্যান্ড্রয়েড ২.১ এর মতো কম ভার্সনে কাজ করার মতো উপযুক্ত হতে সাপোর্ট লাইব্রেরী ব্যবহার করে থাকে, তাহলে android: নেমস্পেস থেকে showAsAction এট্রিবিউট পাওয়া যাবে না। এর পরিবর্তে এই এট্রিবিউট সাপোর্ট লাইব্রেরী কর্তৃক প্রদত্ত হয় এবং আপনাকে অবশ্যই আপনার নিজস্ব নেমস্পেসকে নির্ধারণ করতে হবে এবং ওই নেমস্পেসকে এট্রিবিউট প্রিফিক্স হিসাবে ব্যবহার করতে হবে। (একটি কাস্টম এক্সএমএল নেমস্পেস আপনার অ্যাপ নেমের উপর ভিত্তি করে হওয়া উচিত, কিনুত এটা যে কোন নাম হতে পারে যেটা আপনি চান এবং শুধুমাত্র ফাইলের সুবিধার মধ্যে প্রবেশযোগ্য যেখানে আপনি এটা ঘোষণা করবেন)। উদাহরন:

res/menu/main_activity_actions.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:yourapp="http://schemas.android.com/apk/res-auto" >
<!-- Search, should appear as action button -->
<item android:id="@+id/action_search"
      android:icon="@drawable/ic_action_search"
      android:title="@string/action_search"
      yourapp:showAsAction="ifRoom" />

...
</menu>
```

একশন বারে একশন এ্যাড করুন

একশন বারের মধ্যে মেনু আইটেম রাখতে, নির্দিষ্ট Menu অবজেক্টের মধ্যে মেনু রিসোর্স বিস্তার করতে আপনার একটিভিটির মধ্যে onCreateOptionsMenu() কলব্যাক মেথডটি বাস্তবায়ন করুন।
উদাহরন:

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu items for use in the action bar
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.main_activity_actions, menu);
    return super.onCreateOptionsMenu(menu);
}
```

একশন বাটন রেসপন্স করা

যখন একজন ব্যবহারকারী একশন ওভারফ্লোর মধ্যে কোন একশন বাটনে বা অন্য কোন আইটেমে চাপ দিবে, সিস্টেম আপনার এক্টিভিটির `onOptionsItemSelected()` কলব্যাক মেথডকে কল করবে। আপনার এই পদ্ধতি বাস্তবায়নে কোন আইটেমটিতে চাপ দেওয়া হয়েছে তা ঠিক করতে নির্দিষ্ট MenuItem এ `getItemId()` কে কল করে- ফিরতি আইডি ভ্যালু যা আপনি কনসপন্ডিং এলিমেন্ট এর `android:id` এট্রিবিউটকে ঘোষণা করেছিলেন তার সাথে ম্যাচ করে।

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle presses on the action bar items
    switch (item.getItemId()) {
        case R.id.action_search:
            openSearch();
            return true;
        case R.id.action_settings:
            openSettings();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

লো-লেভেল একটিভিটির জন্যে আপ বাটন এ্যাড (সংযোজন) করা



ফিগার 8. Gmail এ আপ বাটন

আপনার অ্যাপের মধ্যে যে সকল স্ক্রিন আছে তা আপনার অ্যাপের প্রধান প্রবেশ পথ নয় (একটিভিটি যা হোম স্ক্রিন নয়) একশন বারের *Up* বাটন প্রেস করে, ব্যবহারকারীদের অ্যাপ এর হয়ারারকিতে লজিক্যাল প্যারেন্ট স্ক্রিনে নেভিগেট করার জন্যে একটা উপায় দেওয়া উচিত।

যখন অ্যান্ড্রয়েড 8.1 (API level 16) বা এর চেয়ে উন্নত সংস্করণে রান করে অথবা সাপোর্ট লাইব্রেরী থেকে `ActionBarActivity` ব্যবহার করা হয়, পারফর্ম করা *Up* নেভিগেশন শুধুমাত্র চায় যে আপনি মেনিফেস্ট ফাইলে প্যারেন্ট একটিভিটি ডিক্লেয়ার করবেন এবং একশন বারের জন্যে *Up* বাটন সক্রিয় করবেন।

উদাহরন: এখানে আপনি মেনিফেস্টে একটি একটিভিটির প্যারেন্ট ডিক্লেয়ার করতে পারেন:

```
For example, here's how you can declare an activity's parent in the manifest:<application ... >
...
<!-- The main/home activity (it has no parent activity) -->
<activity
    android:name="com.example.myfirstapp.MainActivity" ...>
    ...
</activity>
<!-- A child of the main activity -->
<activity
    android:name="com.example.myfirstapp.DisplayMessageActivity"
    android:label="@string/title_activity_display_message"
    android:parentActivityName="com.example.myfirstapp.MainActivity" >
    <!-- Parent activity meta-data to support 4.0 and lower -->
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value="com.example.myfirstapp.MainActivity" />
</activity>
</application>
```

তারপর `setDisplayHomeAsUpEnabled()`কে কল করার মাধ্যমে *Up* বাটন হিসাবে অ্যাপ আইকনকে সক্রিয় করতে পারেন:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_displaymessage);

getSupportActionBar().setDisplayHomeAsUpEnabled(true);
// If your minSdkVersion is 11 or higher, instead use:
// getSupportActionBar().setDisplayHomeAsUpEnabled(true);
}
```

কারণ সিস্টেমটি এখন জানে যে DisplayMessageActivity জন্য MainActivity হচ্ছে প্যারেন্ট একটিভিটি, যখন ব্যবহারকারী *Up* বাটন প্রেস করবে, সিস্টেমটি প্যারেন্ট একটিভিটিতে সঠিকভাবে নেভিগেশন করবে-আপনার *Up* বাটনের ইভেন্টে কোন হস্তক্ষেপ করার দরকার নেই।

আপ নেভিগেশন সম্পর্কে আরও তথ্য জানতে , নেভিগেশন প্রদান করা অধ্যায়টি দেখুন।

একশন বারটিকে স্টাইল করা

(<http://developer.android.com/training/basics/actionbar/styling.html>)

একশন বার আপনার ব্যবহারকারীদের দিচ্ছে আপনার অ্যাপে কাজ করতে এবং নেভিগেট করার পরিচিত এক উপায়, কিন্ত এর মানে এই নয় যে অন্যান্য অ্যাপে যেভাবে থাকে আপনার অ্যাপেও ঠিক সেভাবে থাকবে। আপনি যদি একশন বারকে স্টাইল করে আপনার ব্র্যান্ড পনের সাথে ভালো ভাবে ফিট করতে চান তাহলে আপনি খুব সহজেই অ্যান্ড্রয়েডের style and theme রিসোর্স ব্যবহার করে তা করতে পারেন (লিংক: <http://developer.android.com/guide/topics/ui/themes.html>)।

অ্যান্ড্রয়েড বেশ কিছু বিল্ট ইন একটিভিটি থিম অন্তর্ভুক্ত করেছে যার মধ্যে রয়েছে “ডার্ক” বা “লাইট” একশন বার স্টাইল। আপনি আপনার একশন বারের দৃশ্যমানতাকে পরিবর্তন করতে চাইলে এই থিমগুলোকে আরও প্রসারিত/বিস্তৃত করতে পারবেন।

নোট: আপনি যদি একশন বারের জন্যে সাপোর্ট লাইব্রেরী এপিআই (অচওং) ব্যবহার করে থাকেন, অবশ্যই আপনাকে স্টাইলের Theme.AppCompat ফ্যামিলি ব্যবহার (অথবা ওভাররাইড করতে হবে) করতে হবে (এপিআই ১১ বা এর উপরের লেভেল যে Thme.Helo ফ্যামিলি আছে তার পরিবর্তে ব্যবহার করতে হবে)। আপনি যদি তাই করে থাকেন, তাহলে প্রতিটা স্টাইল প্রপারটি যা আপনি ডিক্লেয়ার করেছেন অবশ্যই দুইবার ডিক্লেয়ার করতে হবে, একবার প্লাটফর্মের স্টাইল প্রপারটি (android:প্রপারটি) ব্যবহার করে এবং একবার সাপোর্ট লাইব্রেরীতে অন্তর্ভুক্ত স্টাইল প্রপারটি ব্যবহার করে (appcompat.R.attr প্রপারটি)। বিস্তারিত জানতে নিচের উদাহরন দেখুন।

অ্যান্ড্রয়েড থীম ব্যবহার করুন



অ্যান্ড্রয়েড দুইটা বেজলাইন একটিভিটি থিম অন্তর্ভুক্ত করে যা একশন বারের জন্য কালার নির্দেশ করে:

- “ডার্ক” থিম এর জন্য Theme.Holo
- “লাইট” থিমের জন্য Theme.Holo.Light

আপনি এই থিম আপনার পূর্ণাঙ্গ অ্যাপে প্রয়োগ করতে পারেন অথবা কোন স্বতন্ত্র একটিভিটি তে এলিমেন্ট বা একক এলিমেন্ট এর জন্য ধহফৎড়রফঃঃযবসব এট্রিবিউট এর সাথে আপনার মেনিফেস্ট ফাইল এ তাদের ডিক্লেয়ার করার মাধ্যমে।

উদাহরনের জন্য:

```
<application android:theme="@android:style/Theme.Holo.Light" ... />
```



আপনি ডার্ক একশন বারও ব্যবহার করতে পারেন যেখানে একটিভিটির বাকী অংশ Theme.Holo.Light.DarkActionBar থিম ডিক্লেয়ার করার মাধ্যমে লাইট কালার স্কিম ব্যবহার করে।

যখন সাপোর্ট লাইব্রেরী ব্যবহার করবেন, আপনি অবশ্যই পরিবর্তে Theme.AppCompat থিম ব্যবহার করবেন

- Theme.AppCompat : “ডার্ক” থিম এর জন্য
- Theme.AppCompat.Light: “লাইট” থিম এর জন্য
- Theme.AppCompat.Light.DarkActionBar “লাইট” থিম সাথে ডার্ক একশন বার এর জন্য

নিশ্চিত হোন যে আপনি যে একশন বার আইকন ব্যবহার করেছেন তা যেন আপনার একশন বারের কালার থেকে সম্পূর্ণভাবে ভিন্ন হয়। আপনার সাহায্যের জন্য, Action Bar Icon Pack হোলো লাইট এবং হোলো ডার্ক একশন বার উভয় এর সাথেই ব্যবহার করার জন্য স্ট্যান্ডার্ড একশন আইকন অন্তর্ভুক্ত করে।

ব্যাকগ্রাউন্ড কাস্টমাইজ করুন



একশন বার ব্যাকগ্রাউন্ড পরিবর্তন করতে, আপনার একটিভিটির জন্য একটা কাস্টম থিম তৈরী করুন যা actionBarStyle প্রপারটি কে ওভাররাইড করে। এই প্রপারটি অন্য স্টাইল নির্দেশ করে যার মধ্যে আপনি একশন বার ব্যাকগ্রাউন্ড এর জন্য ড্রয়েবল রিসোর্স নির্দিষ্ট করতে background প্রপারটি ওভাররাইড করতে পারবেন।

যদি আপনার অ্যাপস navigation tabs বা split action bar ব্যবহার করে থাকে, তাহলে আপনি এই বারের জন্য আলাদা আলাদা ভাবে backgroundStacked এবং backgroundSplit প্রপারটি ব্যবহার করে ব্যাকগ্রাউন্ড নির্দিষ্ট করতে পারেন।

সতর্কতা: এটা গুরুত্বপূর্ণ যে আপনি একটা যথাযথ প্যারেন্ট থিম ডিক্লেয়ার করতে পারেন যা থেকে আপনার কাস্টম থিম এবং স্টাইল তাদের স্টাইল পেয়ে থাকে। একটা প্যারেন্ট স্টাইল ছাড়া, আপনার একশন বার অনেক স্টাইল প্রপারটি ছাড়াই একশন বার হবে যদিনা আপনি স্বয়ং নিজেই তাদেরকে স্পষ্টভাবে ডিক্লেয়ার না করেন।

শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যখন অ্যান্ড্রয়েড এবং এর চেয়ে উন্নত সংস্করণকে সাপোর্ট করবে, আপনি একশন বার এর ব্যাকগ্রাউন্ড কে এভাবে নির্ধারন করতে পারেন:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@style/Theme.Holo.Light.DarkActionBar">
    <item name="android:actionBarStyle">@style/MyActionBar</item>
</style>

<!-- ActionBar styles -->
<style name="MyActionBar"
    parent="@style/Widget.Holo.Light.ActionBar.Solid.Inverse">
    <item name="android:background">@drawable/actionbar_background</item>
</style>
</resources>
```

এরপর আপনার থিম আপনার পূর্ণাঙ্গ অ্যাপে বা স্বতন্ত্র একটিভিটিতে প্রয়োগ করতে পারেন:

```
<application android:theme="@style/CustomActionBarTheme" ... />
```

অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যখন আপনি সাপোর্ট লাইব্রেরী ব্যবহার করেন, উপরের একই থিম পরিবর্তন হয়ে অবশ্যই এরকম দেখাবে:

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@style/Theme.AppCompat.Light.DarkActionBar">
    <item name="android:actionBarStyle">@style/MyActionBar</item>

    <!-- Support library compatibility -->
    <item name="actionBarStyle">@style/MyActionBar</item>
</style>

<!-- ActionBar styles -->
<style name="MyActionBar"
    parent="@style/Widget.AppCompat.Light.ActionBar.Solid.Inverse">
    <item name="android:background">@drawable/actionbar_background</item>

    <!-- Support library compatibility -->
    <item name="background">@drawable/actionbar_background</item>
</style>
</resources>
```

এরপর আপনার থিম আপনার পূর্ণাঙ্গ অ্যাপে বা স্বতন্ত্র একটিভিটিতে প্রয়োগ করতে পারেন:

```
<application android:theme="@style/CustomActionBarTheme" ... />
```

টেক্সট কালার কাস্টোমাইজ

আপনার একশন বার পরিবর্ধন বা পরিমার্জন করতে, প্রতিটা টেক্সট এলিমেন্ট এর জন্য ভিন্ন প্রপারটিজ ওভাররাইড করা দরকার:

- একশন বার টাইটেল: একটি কাস্টম স্টাইল তৈরী করুন যা `textColor` প্রপারটি কে সুনির্দিষ্ট করে এবং আপনার কাস্টম `actionBarStyle` এর মধ্যে `titleTextStyle` এর জন্য ওই স্টাইলকে সুনির্দিষ্ট করে।

নোট: `TitleTextStyle` তে কাস্টম স্টাইল প্রয়োগ করতে `TextAppearance.Holo.Widget.ActionBar.Title` কে প্যারেন্ট স্টাইল হিসাবে ব্যবহার করা উচিত।

- একশন বার ট্যাবস: আপনার একটিভিটি থিম এর মধ্যে `actionBarTabTextStyle` কে ওভাররাইড করুন
- একশন বাটন: একটিভিটি থিম এর মধ্যে `actionMenuTextColor` কে ওভাররাইড করুন

শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যখন অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণ কে সাপোর্ট করা হয়, তখন আপনার স্টাইল এক্সএমএল ফাইল এই রকম দেখাবে:

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@style/Theme.Holo">
    <item name="android:actionBarStyle">@style/MyActionBar</item>
    <item name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>
    <item name="android:actionMenuTextColor">@color/actionbar_text</item>
</style>

<!-- ActionBar styles -->
<style name="MyActionBar"
    parent="@style/Widget.Holo.ActionBar">
    <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>
</style>

<!-- ActionBar title text -->
<style name="MyActionBarTitleText"
    parent="@style/TextAppearance.Holo.Widget.ActionBar.Title">
    <item name="android:textColor">@color/actionbar_text</item>
</style>

<!-- ActionBar tabs text styles -->
<style name="MyActionBarTabText"
    parent="@style/Widget.Holo.ActionBar.TabText">
    <item name="android:textColor">@color/actionbar_text</item>
</style>
</resources>
```

অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যখন সাপোর্ট লাইব্রেরী ব্যবহার করা হবে, তখন আপনার স্টাইল এক্সএমএল ফাইল এই রকম দেখাবে:

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@style/Theme.AppCompat">
    <item name="android:actionBarStyle">@style/MyActionBar</item>
    <item name="android:actionBarTabTextStyle">@style/MyActionBarTabText</item>
    <item name="android:actionMenuTextColor">@color/actionbar_text</item>

    <!-- Support library compatibility -->
    <item name="actionBarStyle">@style/MyActionBar</item>
    <item name="actionBarTabTextStyle">@style/MyActionBarTabText</item>
    <item name="actionMenuTextColor">@color/actionbar_text</item>
</style>

<!-- ActionBar styles -->
<style name="MyActionBar"
    parent="@style/Widget.AppCompat.ActionBar">
    <item name="android:titleTextStyle">@style/MyActionBarTitleText</item>

    <!-- Support library compatibility -->
    <item name="titleTextStyle">@style/MyActionBarTitleText</item>
</style>

<!-- ActionBar title text -->
<style name="MyActionBarTitleText"
    parent="@style/TextAppearance.AppCompat.Widget.ActionBar.Title">
    <item name="android:textColor">@color/actionbar_text</item>
    <!-- The textColor property is backward compatible with the Support Library -->
</style>

<!-- ActionBar tabs text -->
<style name="MyActionBarTabText"
    parent="@style/Widget.AppCompat.ActionBar.TabText">
    <item name="android:textColor">@color/actionbar_text</item>
    <!-- The textColor property is backward compatible with the Support Library -->
</style>
</resources>
```


ট্যাব ইন্ডিকেটর কাস্টমাইজ করা



Navigation tabs এর জন্য ব্যবহার করা ইন্ডিকেটর পরিবর্তন করতে, একটা একটিভিটি থিম তৈরী করুন যা actionBarTabStyle প্রপার্টিকে ওভাররাইড করে। এই প্রপারটি অন্য স্টাইল রিসোর্স কে নির্দেশ করে যার মধ্যে আপনি background প্রপার্টিকে ওভাররাইড করতে পারবেন যাতে একটি স্টেট-লিস্ট ড্রয়েবল নির্দিষ্ট করা উচিত।

নোট: একটি স্টেট-লিস্ট ড্রয়েবল গুরুত্বপূর্ণ যাতে বর্তমানে ট্যাব ব্যাকগ্রাউন্ড দিয়ে এর স্টেট সিলেক্ট করে যা অন্যান্য ট্যাবের চেয়ে ভিন্ন। একটি ড্রয়েবল রিসোর্স যা মাল্টিপল বাটন স্টেট ধারণ করে তা কীভাবে তৈরী করা যায় সে সম্পর্কে জানতে পড়ুন State List (<http://developer.android.com/guide/topics/resources/drawable-resource.html#StateList>) ডকুমেন্টেশন।

উদাহরণ এর জন্য, এখানে একটি স্টেট-লিস্ট ড্রয়েবল দেয়া হচ্ছে, যা একটি একশন বার ট্যাবের কতিপয় বিভিন্ন স্টেটস এর জন্য সুনির্দিষ্ট ব্যাকগ্রাউন্ড ইমেজকে ডিক্লেয়ার করে:

res/drawable/actionbar_tab_indicator.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Non focused states -->
    <item android:state_focused="false" android:state_selected="false"
        android:state_pressed="false"
        android:drawable="@drawable/tab_unselected" />
    <item android:state_focused="false" android:state_selected="true"
        android:state_pressed="false"
        android:drawable="@drawable/tab_selected" />

    <!-- Focused states (such as when focused with a d-pad or mouse hover) -->
    <item android:state_focused="true" android:state_selected="false"
        android:state_pressed="false"
        android:drawable="@drawable/tab_unselected_focused" />
    <item android:state_focused="true" android:state_selected="true"
        android:state_pressed="false"
        android:drawable="@drawable/tab_selected_focused" />

    <!-- STATES WHEN BUTTON IS PRESSED -->

    <!-- Non focused states -->
    <item android:state_focused="false" android:state_selected="false"
        android:state_pressed="true"
        android:drawable="@drawable/tab_unselected_pressed" />
    <item android:state_focused="false" android:state_selected="true"
        android:state_pressed="true"
        android:drawable="@drawable/tab_selected_pressed" />
```

```
<!-- Focused states (such as when focused with a d-pad or mouse hover) -->
  <item android:state_focused="true" android:state_selected="false"
    android:state_pressed="true"
    android:drawable="@drawable/tab_unselected_pressed" />
  <item android:state_focused="true" android:state_selected="true"
    android:state_pressed="true"
    android:drawable="@drawable/tab_selected_pressed" />
</selector>
```

শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যখন অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণ কে সাপোর্ট করে, তখন আপনার স্টাইল এক্সএমএল ফাইল এই রকম দেখাবে:

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@style/Theme.Holo">
    <item name="android:actionBarTabStyle">@style/MyActionBarTabs</item>
</style>

<!-- ActionBar tabs styles -->
<style name="MyActionBarTabs"
    parent="@style/Widget.Holo.ActionBar.TabView">
<!-- tab indicator -->
    <item name="android:background">@drawable/actionbar_tab_indicator</item>
</style>
</resources>
```

অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যখন সাপোর্ট লাইব্রেরী ব্যবহার করা হবে, তখন আপনার স্টাইল এক্সএমএল ফাইল এই রকম দেখাবে:

res/values/themes.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@style/Theme.AppCompat">
    <item name="android:actionBarTabStyle">@style/MyActionBarTabs</item>

    <!-- Support library compatibility -->
    <item name="actionBarTabStyle">@style/MyActionBarTabs</item>
</style>

<!-- ActionBar tabs styles -->
<style name="MyActionBarTabs"
    parent="@style/Widget.AppCompat.ActionBar.TabView">
    <!-- tab indicator -->
    <item name="android:background">@drawable/actionbar_tab_indicator</item>

    <!-- Support library compatibility -->
    <item name="background">@drawable/actionbar_tab_indicator</item>
</style>
</resources>
```

আরও কিছু তথ্য

- Action Bar গাইডের তালিকা থেকে একশন বার এর জন্য আরও স্টাইল প্রপারটি দেখুন
- Style and Themes গাইড থেকে আরও শিখুন থিম কীভাবে কাজ করে
- এমন কি একশন বারের জন্য পূর্ণাঙ্গ স্টাইল সম্পর্কে জানতে Android Action Bar Style Generator এ চেষ্টা করুন

একশন বার ওভারলে করা

(<http://developer.android.com/training/basics/actionbar/overlaying.html>)

বাই ডিফল্ট, একশন বার আপনার একটিভিটি উইন্ডোর উপরের দিকে দৃশ্যমান হয়, আপনার একটিভিটির অবশিষ্ট লেআউট এর জন্য নির্ধারিত স্পেসের পরিমাণ সামান্য কমিয়ে দেয়। ইউজারের যোগাযোগের সময় যদি আপনি একশন বার হাইড বা শো করতে চান, আপনি এটা ActionBar এ `hide()` এবং `show()` দ্বারা করতে পারেন। যা হোক, এটা আপনার একটিভিটিকে লেআউট এর নতুন সাইজের উপর ভিত্তি করে রিকম্পিউট বা রিড্র করাবে।



ফিগার ১. ওভারলে মোডে গ্যালারির একশন বার।

যখন একশন বার হাইড করে এবং শো করে, আপনার লেআউট রিসাইজ করা পরিহার করতে, আপনি একশন বারের জন্য *overlay mode* সক্রিয় করতে পারেন। যখন ওভারলে মোড এ থাকবে, আপনার একটিভিটি লেআউট বিদ্যমান স্পেসের সবটুকু ব্যবহার করে নেয়, যেন একশন বার সেখানে নেই এবং সিস্টেম আপনার লেআউটের সামনে একশন বার ড্র করে। এটা উপরের লেআউটের কিছু অংশ অস্পষ্ট করে দেয়, কিনুত এখন একশন বার যখন হাইড বা শো করে, সিস্টেমটির লেআউট রিসাইজ করার কোন প্রয়োজন নেই এবং এর পরিবর্তন নিখুত।

পরামর্শ: আপনি যদি চান আপনার লেআউট একশন বারের পেছনে আংশিকভাবে দৃশ্যমান হবে, একশন বারের জন্য আংশিক স্বচ্ছ ব্যাকগ্রাউন্ড সহ একটি কাস্টম স্টাইল তৈরী করুন, যেমন ফিগার ১ দেখানো হয়েছে। একশন বার ব্যাকগ্রাউন্ড কীভাবে নির্ধারন করা হয় তা জানতে Styling the Action Bar পড়ুন।

ওভারলে মোড সক্রিয় করুন

একশন বারের জন্য ওভারলে মোড সক্রিয় করতে, আপনার একটি কাস্টম থিম তৈরী করা প্রয়োজন যা বিদ্যমান একশন বার থিম কে বিস্তৃত করে এবং true তে android.windowActionBarOverlay প্রপারটি সেট করুন।

ত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য

যদি আপনার minSdkVersion ১১ বা এর উপরের লেভেলে সেট হয়ে থাকে, আপনার কাস্টম থিম এর উচিত আপনার প্যারেন্ট থিম হিসাবে Theme.Holo থিম (অথবা এই গোত্রের অন্য থিম) ব্যবহার করা। উদাহরনের জন্য:

```
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@android:style/Theme.Holo">
    <item name="android:windowActionBarOverlay">true</item>
</style>
</resources>
```

অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য

অ্যান্ড্রয়েড ৩.০ এর চেয়ে নীচের সংস্করণে সামঞ্জস্যপূর্ণ করার জন্য যদি আপনার অ্যাপ সাপোর্ট লাইব্রেরী ব্যবহার করে থাকে, আপনার কাস্টম থিম এর উচিত আপনার প্যারেন্ট থিম হিসাবে Theme.AppCompat থিম (অথবা একই গোত্রের কোন একটি) ব্যবহার করা। উদাহরণের জন্য:

```
<resources>
<!-- the theme applied to the application or activity -->
<style name="CustomActionBarTheme"
    parent="@android:style/Theme.AppCompat">
    <item name="android:windowActionBarOverlay">true</item>

    <!-- Support library compatibility -->
    <item name="windowActionBarOverlay">true</item>
</style>
</resources>
```

এছাড়াও এটা নোটিশ করে যে, এই থিম windowActionBarOverlay স্টাইলের জন্যে দুইটা অর্থ বহন করে, একটি android: প্রিফিক্স সহ এবং অন্যটি android: প্রিফিক্স ছাড়া। android: প্রিফিক্স সহটি অ্যান্ড্রয়েডের এর সংস্করণগুলোর জন্য যা প্ল্যাটফর্মে স্টাইল অন্তর্ভুক্ত করে এবং android: প্রিফিক্স ছাড়া অন্যটি পুরাতন সংস্করণের জন্যে যা সাপোর্ট লাইব্রেরী থেকে স্টাইল রিড করে।

লেআউট টপ মার্জিন নির্দিষ্ট করুন

যখন একশন বার ওভারলে মোডে থাকে, এটা আপনার লেআউটের কিছু অংশকে অস্পষ্ট করে দিতে পারে যা দৃশ্যমান থাকার প্রয়োজন। এটা নিশ্চিত করতে, এই আইটেমগুলোকে সব সময় একশন বারের নীচে রাখতে, actionBarSize দ্বারা নির্ধারিত উচ্চতা ব্যবহার করে ভিউ এর উপরে হয় মার্জিন নাহয় প্যাডিং এড করুন। উদাহরনের জন্য:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="?android:attr/actionBarSize">
    ...
</RelativeLayout>
```

আপনি যদি একশন বার এর জন্য সাপোর্ট লাইব্রেরী ব্যবহার করে থাকেন, আপনার android:prifিক্সটি সরিয়ে ফেলা দরকার। উদাহরনের জন্য:

```
<!-- Support library compatibility -->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingTop="?attr/actionBarSize">
    ...
</RelativeLayout>
```

এই ক্ষেত্রে, প্রিফিক্স ছাড়া ?attr/actionBarSize ভ্যালু টি সকল সংস্করনে কাজ করে, এমনকি অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করনেও।

বিভিন্ন ডিভাইসকে সাপোর্ট করানো

(<http://developer.android.com/training/basics/supporting-devices/index.html>)

সারা বিশ্বে অ্যান্ড্রয়েড ডিভাইস বিভিন্ন আকার আকৃতি নিয়ে আসে। এই যে বহুবিধ ধরনের ডিভাইসের বিস্তৃত এলাকা, এখানে আপনি পাবেন আপনার অ্যাপকে একটা বিশাল সংখ্যক গ্রাহকের কাছে পৌঁছানোর সুযোগ। অ্যান্ড্রয়েডে যতটুকু সফল হওয়া সম্ভব তা হওয়ার জন্য আপনাকে আপনার অ্যাপকে এমন ভাবে তৈরী করতে হবে যা ডিভাইসের বিভিন্ন কনফিগারেশনে মানান সই হয়। কিন্ত বৈচিত্রের বিষয়টা আপনাকে বিবেচনায় রাখতে হবে যেমন, বিভিন্ন ল্যান্ডস্কেপ, স্ক্রিন এর সাইজ, এবং অ্যান্ড্রয়েড প্ল্যাটফর্ম বিভিন্ন সংস্করণ ইত্যাদী।

এই ক্লাস আপনাকে শেখাবে, কীভাবে মৌলিক প্ল্যাটফর্ম বৈশিষ্ট্যগুলো ব্যবহার করা যায় যা বিকল্প রিসোর্স এবং অন্যান্য বৈশিষ্ট্যগুলোকে নিয়ন্ত্রণ করে যাতে একটা সিঙ্গেল অ্যাপলিকেশন প্যাকেজ (APK) ব্যবহার করে, বিভিন্ন অ্যান্ড্রয়েড-উপযুক্ত ডিভাইসে আপনার অ্যাপ অপটিমাইজড ইউজার এক্সপেরিয়েন্স সরবরাহ করতে পারবে।

এই অধ্যায়ের অনুশীলনী সমূহ

ভিন্ন ভিন্ন ভাষাকে সাপোর্ট করা

শিখুন কীভাবে বিকল্প স্ট্রিং রিসোর্স দিয়ে অনেকগুলো ভাষাকে সাপোর্ট করা যায়।

ভিন্ন ভিন্ন স্ক্রিনকে সাপোর্ট করা

শিখুন বিভিন্ন স্ক্রিন সাইজ এবং ঘণত্ব এর জন্য কীভাবে ইউজার এক্সপেরিয়েন্সকে অপটিমাইজ করা যায়।

ভিন্ন ভিন্ন প্লাটফর্ম সংস্করণকে সাপোর্ট করা

শিখুন অ্যান্ড্রয়েড এর নতুন সংস্করণে যে এপিআই গুলো(অচওং) পাওয়া যায় তা কীভাবে অ্যান্ড্রয়েডের পুরাতন সংস্করণে ব্যবহার করা যায়।

ভিন্ন ভিন্ন ভাষাকে সাপোর্ট করা

(<http://developer.android.com/training/basics/supporting-devices/languages.html>)

আপনার অ্যাপ কোড থেকে ইউজার ইন্টারফেস স্ট্রিংকে বের করে ফেলা এবং একটা বাইরের ফাইলে রাখা, সবসময়ই একটা ভালো চর্চা। প্রতিটা অ্যান্ড্রয়েড প্রজেক্টের রিসোর্স ডিরেক্টরী দিয়ে অ্যান্ড্রয়েড এটাকে সহজ করে দেয়।

আপনি যদি আপনার প্রজেক্ট অ্যান্ড্রয়েড এসডিকে টুলস ব্যবহার করে তৈরী করে থাকেন, টুলসটি প্রজেক্টের একদম উপরের স্তরে একটা `res/` ডিরেক্টরী তৈরী করে। এই `res/` ডিরেক্টরীর মধ্যে বিভিন্ন রিসোর্স টাইপের জন্য সাবডিরেক্টরী আছে। এখানে আরও কিছু ডিফল্ট ফাইল আছে যেমন `res/values/strings.xml` যা আপনার স্ট্রিং ভ্যালু কে ধারণ করে।

লোকাল ডিরেক্টরী এবং স্ট্রিং ফাইল তৈরী করা

আরও ভাষার জন্য সাপোর্ট এ্যাড করতে, `res/` এর ভিতরে অতিরিক্ত `values` ডিরেক্টরী তৈরী করুন যা ডিরেক্টরীর একদম শেষে একটি হাইফেন এবং আইএসও কান্ট্রি কোড অন্তর্ভুক্ত করে।

উদাহরণস্বরূপ, `values-es/` হচ্ছে ল্যাঙ্গুয়েজ কোড "es" দিয়ে লোকালের জন্য সাধারণ রিসোর্সগুলো কে ধরে রাখার ডিরেক্টরী। অ্যান্ড্রয়েড ডিভাইস চলাকালীন সময়ে এর লোকাল সেটিং অনুসারে যথাযথ রিসোর্স লোড করে।

আপনি যদি ঠিক করে থাকেন কোন ভাষা কে সাপোর্ট করবেন, রিসোর্স সাব ডিরেক্টরী এবং স্ট্রিং রিসোর্স ফাইল তৈরী করুন। উদাহরণস্বরূপ:

```
MyProject/  
res/  
  values/  
    strings.xml  
  values-es/  
    strings.xml  
  values-fr/  
    strings.xml
```

প্রতিটা লোকালের জন্য স্ট্রিং ভ্যালু যথাযথ ফাইলে এ্যাড করুন।

কাজ চলাকালীন সময়ে, অ্যান্ড্রয়েড সিস্টেম চলতি সময়ে ইউজারের ডিভাইসে যে লোকাল সেট করা হয়েছে তার উপর ভিত্তি করে সামঞ্জস্যপূর্ণ স্ট্রিং রিসোর্স এর সেট ব্যবহার করে।

উদাহরণস্বরূপ, বিভিন্ন ভাষার জন্য কিছু ভিন্ন স্ট্রিং রিসোর্স নীচে দেওয়া আছে।

ইংরেজী (ডিফল্ট লোকাল), `/values/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>  
  <resources>  
    <string name="title">My Application</string>  
    <string name="hello_world">Hello World!</string>  
  </resources>
```

স্প্যানিশ, `/values-es/strings.xml`:

```
<?xml version="1.0" encoding="utf-8"?>  
  <resources>  
    <string name="title">Mi Aplicación</string>  
    <string name="hello_world">Hola Mundo!</string>  
  </resources>
```

ফরাসী , /values-fr/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mon Application</string>
    <string name="hello_world">Bonjour le monde !</string>
</resources>
```

নোট: আপনি যে কোন রিসোর্স টাইপের উপরে লোকাল কোয়ালিফায়ার (অথবা যে কোন কনফিগারেশন কোয়ালিফায়ার) ব্যবহার করতে পারেন, যেমন আপনি যদি আপনার বিটম্যাপ ড্রয়েবল এর লোকলাইজড সংস্করণ সরবরাহ করতে চান। আরও তথ্যের জন্য Localization (<http://developer.android.com/guide/topics/resources/localization.html>) দেখুন।

স্ট্রিং রিসোর্স ব্যবহার করুন

আপনি আপনার সোর্স কোডে এবং অন্যান্য এক্সএমএল ফাইলে এ `< string>` এলিমেন্টের name এট্রিবিউট দ্বারা নির্ধারিত রিসোর্স নাম ব্যবহার করে আপনার স্ট্রিং রিসোর্স উল্লেখ করতে পারেন।

আপনার সোর্স কোডে আপনি সিনট্যাক্স `R.string.< string name>` এর সাথে একটি স্ট্রিং রিসোর্স উল্লেখ করতে পারেন। এখানে নান ধরনের পদ্ধতি আছে যা স্ট্রিং রিসোর্স কে এই উপায়ে গ্রহন করে।

উদাহরণস্বরূপ:

```
// Get a string resource from your app's Resources
String hello = getResources().getString(R.string.hello_world);

// Or supply a string resource to a method that requires a string
TextView textView = new TextView(this);
textView.setText(R.string.hello_world);
```

অন্য এক্সএমএল ফাইলে আপনি সিনট্যাক্স `@string/< string name>` এর সাথে একটি স্ট্রিং রিসোর্স উল্লেখ করতে পারেন যখনই এক্সএমএল এট্রিবিউট একটি স্ট্রিং ভ্যালু গ্রহণ করে।

উদাহরণস্বরূপ:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

ভিন্ন ভিন্ন স্ক্রিনকে সাপোর্ট করা

(<http://developer.android.com/training/basics/supporting-devices/screens.html>)

অ্যান্ড্রয়েড দুইটা সাধারণ প্রপার্টি ব্যবহার করে ডিভাইস স্ক্রিনকে শ্রেণীবিভাজন করে: তা হচ্ছে এর সাইজ এবং ঘনত্ব। আপনার আশা করা উচিত যে আপনার অ্যাপ সাইজ এবং ঘনত্ব উভয়ের রেঞ্জের মধ্যে থেকে স্ক্রিন সহ ডিভাইসে ইনস্টল হবে। এর জন্য আপনার উচিত কিছু বিকল্প রিসোর্স অন্তর্ভুক্ত করা যা বিভিন্ন মাপের স্ক্রিন সাইজ এবং ঘনত্বের জন্য আপনার অ্যাপের দৃশ্যমানতাকে অপটিমাইজ করবে।

- এখানে চারধরনের সার্বজনীন সাইজ রয়েছে: স্মল, নরমাল, লার্জ এবং এক্সট্রা লার্জ
- এবং চার ধরনের সার্বজনীন ঘনত্ব রয়েছে: লো (ldpi), মিডিয়াম (mdpi), হাই (hdpi) এবং এক্সট্রা হাই (xhdpi)

বিভিন্ন লেআউট এবং বিটম্যাপ ডিক্লেয়ার করে বিভিন্ন স্ক্রিনের জন্য ব্যবহার করতে চাইলে, আপনাকে অবশ্যই এই বিকল্প রিসোর্স গুলোকে আলাদা ডিরেক্টরীতে রাখতে হবে, বিভিন্ন ভাষার স্ট্রিং এর ক্ষেত্রে যেভাবে করেছেন ঠিক সেভাবেই করতে হবে।

আরও সতর্ক থাকতে হবে যে স্ক্রিন ওরিয়েন্টেশন (ল্যান্ডস্কেপ অথবা পোর্টেইট) এর বিষয়টাও স্ক্রিন সাইজের বৈচিত্রের মধ্যে বিবেচনা করতে হবে, প্রতিটা ওরিয়েন্টেশনে ইউজার এক্সপেরিয়েন্স অপটিমাইজ করতে অনেক অ্যাপের লেআউট পূর্ণবিবেচনা করা উচিত।

ভিন্ন ভিন্ন লেআউট তৈরী করুন

ভিন্ন ভিন্ন স্ক্রিন সাইজে আপনার ইউজার এক্সপেরিয়েন্স সক্রিয় করতে, আপনার উচিত প্রতিটা স্ক্রিন সাইজের জন্যে একটি স্বতন্ত্র লেআউট এক্সএমএল ফাইল তৈরী করা যা আপনি সাপোর্ট করতে চান। প্রতিটা লেআউট যথাযথ রিসোর্স ডিরেক্টরীতে সেভ করা উচিত, একটি-`< screen_size>` সাফিক্স সহ দেয়া নাম। উদাহরণস্বরূপ, একটি লার্জ স্ক্রিনের জন্য স্বতন্ত্র লেআউট `res/layout-large/` এর অধীনে সেভ করা উচিত।

নোট: স্ক্রিনকে সম্পূর্ণভাবে মানানসই করার জন্য অ্যান্ড্রয়েড স্বয়ংক্রিয়ভাবে আপনার লেআউটের সমতা রক্ষা করে। তাই ভিন্ন ভিন্ন স্ক্রিন সাইজের জন্য আপনার লেআউটের ইউজার ইন্টারফেস এলিমেন্টের পরিপূর্ণ সাইজ নিয়ে চিন্তিত হওয়ার কোন কারন নেই কিনুত পরিবর্তে লেআউট কাঠামোর উপরে ফোকাস করা ইউজার এক্সপেরিয়েন্সে প্রভাব ফেলে (যেমন গুরুত্বপূর্ণ ভিউ এর সাইজ অথবা অবস্থান সিবলিং ভিউ এর সাথে সম্পর্কিত)।

উদাহরণস্বরূপ, এই প্রজেক্ট লার্জ স্ক্রিনের জন্য একটা ডিফল্ট লেআউট এবং একটি বিকল্প লেআউট অন্তর্ভুক্ত করে:

```
MyProject/  
  res/  
    layout/  
      main.xml  
    layout-large/  
      main.xml
```

ফাইলটির নাম অবশ্যই একই হতে হবে, কিন্তু অনুরূপ স্ক্রিন সাইজের জন্য অপটিমাইজড ইউজার ইন্টারফেস সরবরাহ করার জন্য তাদের বিষয়বসূত ভিন্ন থাকে।

আপনার অ্যাপে সাধারণভাবে শুধুমাত্র লেআউট ফাইলটি উল্লেখ করুন:

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
}
```

সিস্টেমটি ডিভাইসের স্ক্রিনের সাইজ যেখানে আপনার অ্যাপ রান করছে তার উপর ভিত্তি করে যথাযথ লেআউট ডিরেক্টরী থেকে লেআউট ফাইলটি লোড করে। কীভাবে অ্যান্ড্রয়েড যথাযথ রিসোর্স বাছাই করে এ সম্পর্কে আরও তথ্য Providing Resources গাইডে আছে।

ভিন্ন একটা উদাহরণ হিসাবে, এখানে দেয়া হলো ল্যান্ডস্কেপ ওরিয়েন্টেশনের জন্য বিকল্প লেআউট সহ একটি প্রজেক্ট:

```
MyProject/  
  res/  
    layout/  
      main.xml  
    layout-land/  
      main.xml
```

বাই ডিফল্ট, layout/main.xml ফাইলটি পোর্টেইট ওরিয়েন্টেশন এর জন্য ব্যবহার করা হয়।

আপনি যদি ল্যান্ডস্কেপের জন্য একটি স্পেশাল লেআউট দিতে চান, কোন লার্জ স্ক্রিনের ক্ষেত্রেও, তখন আপনার large এবং land উভয় কোয়ালিফায়ার ব্যবহার করা উচিত:

```
MyProject/  
  res/  
    layout/                # default (portrait)  
      main.xml  
    layout-land/           # landscape  
      main.xml  
    layout-large/          # large (portrait)  
      main.xml  
    layout-large-land/     # large landscape  
      main.xml
```

নোট: অ্যান্ড্রয়েড ৩.২ এবং এর চেয়ে উন্নত সংস্করণ স্ক্রিন সাইজ নির্ধারণে উন্নত পদ্ধতিকে সাপোর্ট করে, যা আপনাকে ডেনজিটি -স্বাধীন পিক্সেল এর শর্তে সর্বনিম্ন প্রস্থ এবং উচ্চতার ভিত্তিতে স্ক্রিন সাইজের জন্য রিসোর্স কে সুনির্দিষ্ট করতে অনুমোদন দেয়। এই অনুশীলনী এই নতুন কৌশল কে ধারণ করে না। আরও তথ্যের জন্য, বিভিন্ন ধরনের স্ক্রিনের জন্য ডিজাইন করা অধ্যায়টি পড়ুন।

বিভিন্ন ধরনের বিটম্যাপ তৈরী করুন

আপনাকে অবশ্যই সবসময় বিটম্যাপ রিসোর্স দেয়া উচিত যা সাধারণ ঘণত্ব মাপকাঠি অনুসারে যথাযথভাবে পরিমাপ হয়: যেমন লো, মিডিয়াম, হাই এবং এক্সট্রা হাই। এটা আপনাকে ভালো গ্রাফিক্যাল কোয়ালিটি অর্জনে এবং সকল স্ক্রিনে কাজ করতে সহায়তা করবে।

এই ইমেজ তৈরী করতে, আপনার উচিত ভেক্টর ফর্মেট এ 'র' রিসোর্স নিয়ে শুরু করা এবং নিম্নোক্ত সাইজ স্কেল ব্যবহার করে প্রতিটা ঘণত্বের জন্য ইমেজ তৈরী করা:

- xhdpi: 2.0
- hdpi: 1.5
- mdpi: 1.0 (baseline)
- ldpi: 0.75

এর মানে আপনি যদি xhdpi ডিভাইসের জন্য ২০০ x ২০০ ইমেজ তৈরী করেন, তাহলে আপনার একই রিসোর্স hdpi ডিভাইসের জন্য ১৫০ x ১৫০, mdpi ডিভাইসের জন্য ১০০ x ১০০ এবং ldpi ডিভাইসের জন্য ৭৫ x ৭৫ এ তৈরী করা উচিত।

এরপর যথাযথ ড্রয়েবল রিসোর্স ডিরেক্টরীতে ফাইলটি রেখে দিন:

```
MyProject/  
  res/  
    drawable-xhdpi/  
      awesomeimage.png  
    drawable-hdpi/  
      awesomeimage.png  
    drawable-mdpi/  
      awesomeimage.png  
    drawable-ldpi/  
      awesomeimage.png
```

যেকোন সময় আপনি @drawable/awesomeimage উল্লেখ করতে পারেন, সিস্টেমটি স্ক্রিনের ঘনত্বের উপর ভিত্তি করে যথাযথ বিটম্যাপ বেছে নেয়।

নোট: লো ডেনজিটি/ঘণত্ব (ldpi) রিসোর্স সব সময় প্রয়োজনীয় নয়। যখন আপনি hdpi এসেট প্রদান করেন, সিস্টেমটি যথাযথভাবে ldpi স্ক্রিনের সাথে মানানসই করতে একে পরিমাপ করে অর্ধেক নামিয়ে নিয়ে আসে।

আপনার অ্যাপের জন্য আইকন এসেট তৈরী করার বিষয়ে আরও জানতে দেখুন Iconography design guide (<http://developer.android.com/design/style/iconography.html>)

ভিন্ন প্লাটফর্ম সংস্করনকে সাপোর্ট করা

(<http://developer.android.com/training/basics/supporting-devices/platforms.html>)

যখন অ্যান্ড্রয়েডের সর্বশেষ সংস্করনে আপনার অ্যাপের জন্য এপিআই (APIs) প্রদান করে, আপনার উচিত অ্যান্ড্রয়েডের পুরাতন সংস্করনকে সাপোর্ট করতে পারে সেভাবে কাজ করা যতক্ষণ না আরও ডিভাইস আপডেট না হচ্ছে। এই অনূশীলনী আপনাকে দেখাবে কীভাবে সর্বশেষ এপিআই (APIs) থেকে সুবিধা নিয়ে পুরাতন সংস্করনকে সাপোর্ট করতে পারে সেভাবে কাজ করা।

অ্যান্ড্রয়েডের প্রতিটা সংস্করনে রান করা সক্রিয় ডিভাইস এর বন্টন দেখাতে Platform Versions এর জন্য ড্যাশবোর্ডটি নিয়মিতভাবে আপডেট হচ্ছে, এটা ডিভাইসের সংখ্যার উপর ভিত্তি করে করা, যা গুগল স্টোর পরিদর্শন করে। সাধারনভাবে, ৯০% সক্রিয় ডিভাইসকে সাপোর্ট করাটা একটা ভালো চর্চা, যখন আপনার অ্যাপ সর্বশেষ সংস্করনের প্রতি লক্ষ্য নির্ধারন করে থাকে।

পরামর্শ: কিছু অ্যান্ড্রয়েড সংস্করনকে সবচেয়ে ভালো বৈশিষ্ট্য এবং কার্যক্ষমতা প্রদান করার জন্য আপনার অ্যাপে Android Support Library ব্যবহার করা উচিত, যা আপনাকে পুরাতন সংস্করনের উপর কিছু নতুন প্লাটফর্ম এপিআই (APIs) ব্যবহার অনুমোদন করে।

মিনিমাম (ন্যূনতম) এবং টার্গেট এপিআই লেভেল সুনির্দিষ্ট করা

AndroidManifest.xml ফাইলটি আপনার অ্যাপ সম্পর্কে বিস্তারিত আলোচনা করে থাকে এবং কোন সংস্করণের অ্যান্ড্রয়েড এটাকে সাপোর্ট করবে তা চিহ্নিত করে। বিশেষ করে, <uses-sdk এলিমেন্ট এর জন্য minSdkVersion এবং targetSdkVersion এট্রিবিউটটি সর্বনিম্ন এপিআই লেভেল চিহ্নিত করে যেটা সহ আপনার অ্যাপ সব কিছুর সাথে এবং সর্বোচ্চ এপিআই লেভেল খাপ খাওয়ানোর মতো যার বিপরীতে আপনি আপনার অ্যাপ ডিজাইন করেছেন এবং টেস্ট করেছেন।

উদাহরণস্বরূপ:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android" ... >
    <uses-sdk android:minSdkVersion="4" android:targetSdkVersion="15" />
    ...
</manifest>
```

যেহেতু অ্যান্ড্রয়েডের নতুন সংস্করণ এসেছে, কিছু স্টাইল এবং আচরণ পরিবর্তন হতে পারে। আপনার অ্যাপকে এই পরিবর্তনগুলোর সুবিধা নেয়ার বিষয়টা করতে দিতে এবং নিশ্চিত করতে যে আপনার অ্যাপ সকল ডিভাইসে মানানসই হবে, সর্বশেষ সংস্করণের সাথে খাপ খাওয়াতে targetSdkVersion ভ্যালু সেট করা উচিত।

রানটাইমে সিস্টেম সংস্করনকে চেক করুন

অ্যান্ড্রয়েড Build কনস্ট্যান্ট ক্লাসের মধ্যে, প্রতিটা প্ল্যাটফর্ম সংস্করনের জন্য দিচ্ছে একটি ইউনিক (স্বতন্ত্র) কোড। আপনার অ্যাপের মধ্যে থেকে এই কোডগুলো ব্যবহার করে কনডিশন তৈরী করা যা কোডটিকে নিশ্চিত করে এবং যা উচ্চতর এপিআই লেভেল সংঘটিত হওয়ার উপর নির্ভর করে, শুধুমাত্র তখন যখন ওই এপিআই সিস্টেমে থাকে।

```
private void setUpActionBar() {  
    // Make sure we're running on Honeycomb or higher to use ActionBar APIs  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {  
        ActionBar actionBar = getActionBar();  
        actionBar.setDisplayHomeAsUpEnabled(true);  
    }  
}
```

নোট: যখন এক্সএমএল রিসোর্স এর পার্সিং করা হয়, অ্যান্ড্রয়েড এক্সএমএল এট্রিবিউটকে পরিহার করে যেটাকে বর্তমান ডিভাইস সাপোর্ট করে না। সুতরাং আপনি নিরাপদভাবে এক্সএমএল এট্রিবিউট ব্যবহার করতে পারেন যেটাকে শুধুমাত্র নতুন সংস্করন সাপোর্ট করে থাকে পুরাতন সংস্করনের ব্রেকিং সম্পর্কে কোন ধরনের দুর্শ্চিন্তা ছাড়াই, যখন এটা ওই কোড কে এনকাউন্টার করে। উদাহরণস্বরূপ, আপনি যদি `targetSdkVersion="11"` টি করে থাকেন, আপনার অ্যাপ অ্যান্ড্রয়েড ৩.০ বা এর চেয়ে উন্নত সংস্করনের ক্ষেত্রে বাই ডিফল্ট ActionBar অন্তর্ভুক্ত করে। তখন একশন বারে মেনু আইটেম সেট করতে, আপনার মেনু রিসোর্স এক্সএমএল এ `android:showAsAction="ifRoom"` সেট করা দরকার। এটা ক্রস-ভার্সন এক্সএমএল ফাইলে করাটা নিরাপদ, কারন পুরাতন সংস্করনের অ্যান্ড্রয়েড স্বভাবতই `showAsAction` এট্রিবিউট পরিহার করে (তাই, `res/menu-v11/` এ আপনার কোন পৃথক সংস্করনের দরকার নেই)।

প্লাটফর্ম স্টাইল এবং থিম ব্যবহার

অ্যান্ড্রয়েড ইউজার এক্সপেরিয়েন্স থিম প্রদান করে যা অ্যাপকে মৌলিক/আসল অপারেটিং সিস্টেম এর লুক এবং ফিল দেয়। এই থিমগুলো মেনিফেস্ট ফাইলের মধ্যে থেকে আপনার অ্যাপে প্রয়োগ হতে পারে। এই বিল্ট ইন থিম এবং স্টাইল ব্যবহার করে আপনার অ্যাপ স্বাভাবিকভাবে নতুন রিলিজ হওয়া প্রতিটা অ্যান্ড্রয়েডের সর্বশেষ লুক এবং ফিল কে অনুসরণ করবে।

আপনার একটিভিটিকে ডায়ালগ বক্স এর মতো করে তৈরী করতে:

```
<activity android:theme="@android:style/Theme.Dialog">
```

আপনার একটিভিটিকে স্বচ্ছ ব্যাকগ্রাউন্ড সহ তৈরী করতে:

```
<activity android:theme="@android:style/Theme.Translucent">
```

/res/values/styles.xml তে নির্ধারিত আপনার নিজস্ব কাস্টম থিম প্রয়োগ করতে:

```
<activity android:theme="@style/CustomTheme">
```

আপনার সম্পূর্ণ অ্যাপে (সকল একটিভিটি) একটি থিম প্রয়োগ, < application> এলিমেন্টে android:theme এট্রিবিউট এ্যাড করুন :

```
<application android:theme="@style/CustomTheme">
```

থিম তৈরী এবং ব্যবহার সম্পর্কিত আরও তথ্যের জন্য Styles and Themes গাইডটি পড়ুন। .

একটিভিটি লাইফসাইকেল ব্যবস্থাপনা

(<http://developer.android.com/training/basics/activity-lifecycle/index.html>)

প্লাটফর্ম স্টাইল এবং থিম ব্যবহারযেহেতু একজন ইউজার আপনার অ্যাপের মধ্যে দিয়ে, বাইরে বা অ্যাপে ফিরে আসার জন্য নেভিগেট করে, আপনার অ্যাপের Activity ইনস্ট্যান্স তার লাইফসাইকেলে বিভিন্ন পরিবর্তিত অবস্থানের মধ্যে থাকে। উদাহরণস্বরূপ যখন আপনার একটিভিটি প্রথমবারের মতো শুরু হয়, এটা সিস্টেমের সম্মুখভাগে চলে আসে এবং ইউজার ফোকাস গ্রহণ করে। এই প্রক্রিয়ার সময়, অ্যান্ড্রয়েড সিস্টেম একটিভিটির উপরে একটি ধারাবাহিক লাইফসাইকেল পদ্ধতিকে কল করে যেখানে আপনি ইউজার ইন্টারফেস এবং অন্যান্য উপাদান সেট করেন। যদি ইউজার একটি একশন সম্পাদন করে থাকে যা অন্য একটিভিটি শুরু করে বা অন্য অ্যাপে পরিবর্তন করে চলে যায়, সিস্টেম আরেকটি লাইফসাইকেল পদ্ধতির সেটকে কল করে যেহেতু এটি ব্যাকগ্রাউন্ডের দিকে ধাবিত হয় (যেখানে একটিভিটি আর দৃশ্যমান নয়, কিনুত ইনস্ট্যান্স এবং এর ধরণ অবিকৃত থাকে)।

লাইফসাইকেল কলব্যাক পদ্ধতির মধ্যে থেকে আপনি ডিক্লেয়ার করতে পারেন যে আপনার একটিভিটি কেমন আচরণ করবে যখন ইউজার একটিভিটি থেকে বের হবে এবং পুনঃপ্রবেশ করবে। উদাহরণস্বরূপ, আপনি যদি একটা স্ট্রিমিং ভিডিও প্লেয়ার বানাতে থাকেন, যখন ইউজার আরেকটি অ্যাপে চলে যায় তখন আপনার ভিডিওতে একটা পজ দেয়া লাগতে পারে এবং নেটওয়ার্ক কানেকশন কে বন্ধ করে দেয়া লাগতে পারে। যখন ইউজার ফিরে আসে, আপনি নেটওয়ার্ককে পূর্ণসংযোগ করতে পারেন এবং ইউজারকে ভিডিওটিকে যে অবস্থায় রেখে যাওয়া হয়েছিল সে অবস্থা থেকে শুরু করাকে অনুমোদন করতে পারে।

এই ক্লাস গুরুত্বপূর্ণ লাইফসাইকেল কলব্যাক মেথড কে বিশ্লেষণ করে যা প্রতিটা Activity ইনস্ট্যান্স গ্রহণ করে এবং কীভাবে এদের ব্যবহার করতে পারেন যাতে ইউজার যেভাবে চায় সেভাবে কাজ করতে পারে এবং সিস্টেম রিসোর্সকে ব্যবহার করে না যখন আপনার একটিভিটির জন্য এর প্রয়োজন হয় না।

এই অধ্যায়ের অনুশীলনী সমূহ

একটিভিটি শুরু করা

একটিভিটি লাইফসাইকেল এর মৌলিক বিষয় সম্পর্কে শিখুন, কীভাবে ইউজার আপনার অ্যাপ শুরু করবে, এবং কীভাবে মৌলিক একটিভিটি তৈরী করার বিষয়টি সম্পাদন করা যায়।

একটিভিটি পজিং এবং রিজিউমিং

যখন আপনার একটিভিটিতে পজ দেয়া হবে (আংশিকভাবে হিডেন থাকে) এবং রিজিউম করা হবে তখন কী হবে এবং যখন এই অবস্থানের পরিবর্তন হবে তখন আপনার কী করা উচিত সে বিষয়ে শিখুন।

একটিভিটি স্টপিং এবং রিস্টার্ট

যখন ইউজার সম্পূর্ণভাবে আপনার একটিভিটি ত্যাগ করবে এবং এটাতে আবার ফিরে আসবে তখন কী হবে সে বিষয়ে শিখুন।

একটিভিটির পূর্ণনির্মাণ

যখন আপনার একটিভিটি সম্পূর্ণভাবে ধ্বংস হবে তখন কী হবে এবং কীভাবে আপনি একটিভিটি স্টেট পূর্ণনির্মাণ করতে পারবেন যখন এর প্রয়োজন হবে সে বিষয়ে শিখুন।

একটিভিটি শুরু করা

(<http://developer.android.com/training/basics/activity-lifecycle/starting.html>)

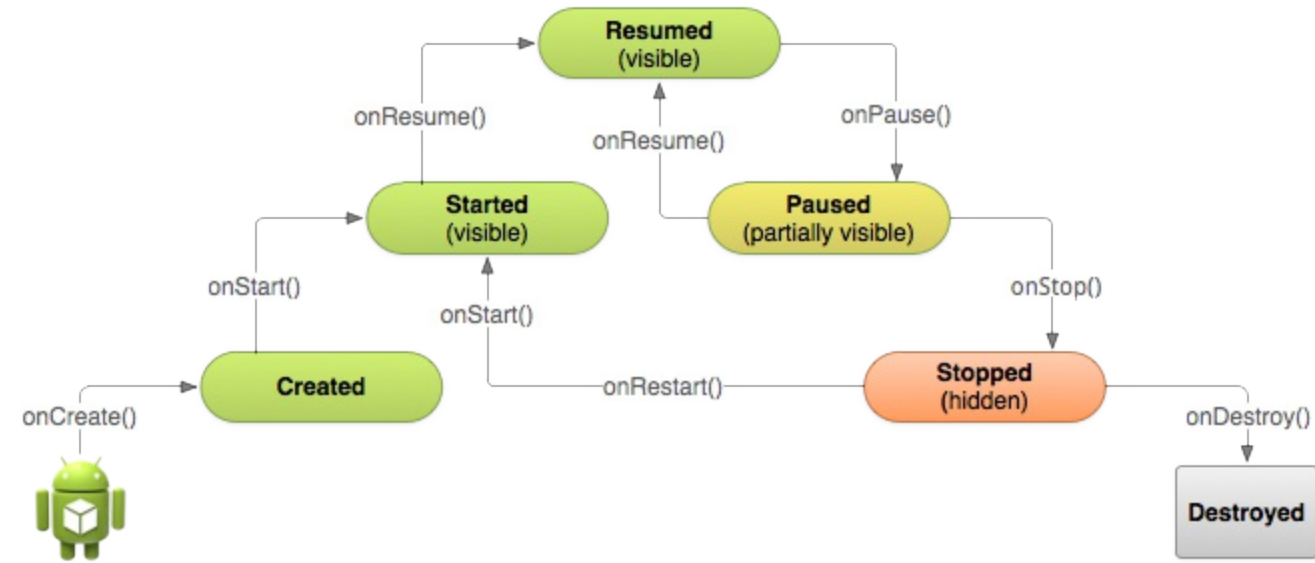
অন্য প্রোগ্রামিং প্যারাডাইম এর মতো নয় যেখানে অ্যাপস একটি `main()` পদ্ধতি কর্তৃক তার যাত্রা শুরু করে, অ্যান্ড্রয়েড সিস্টেম সুনির্দিষ্ট কলব্যাক পদ্ধতি আবাহনের মাধ্যমে একটি Activity ইনস্ট্যান্স এর মধ্যে কোড তৈরী করে যা এর লাইফসাইকেলের সুনির্দিষ্ট ধাপে যোগাযোগ করে। এখানে কলব্যাক পদ্ধতির একটা অনুক্রম (সিকোয়েন্স) আছে যা একটি একটিভিটি তৈরী করে এবং কলব্যাক পদ্ধতির একটি অনুক্রম (সিকোয়েন্স) আছে যা একটিভিটির বিমোচন ঘটায়।

এই অনুশীলনী সবচেয়ে গুরুত্বপূর্ণ লাইফসাইকেল পদ্ধতির একটা সাধারণ ধারণা প্রদান করে এবং দেখায় যে কীভাবে প্রথম লাইফ সাইকেল কলব্যাক পরিচালনা করতে হয় যা আপনার একটিভিটির জন্য একটি নতুন ইনস্ট্যান্স তৈরী করে

লাইফসাইকেল কলব্যাক সম্পর্কে ধারণা

একটি একটিভিটির জীবদশাকালীন সময়ে, সিস্টেম লাইফসাইকেল পদ্ধতির একটি মূল সেট কে ক্রমানুসারে কল করে যা একটা পিরামিডের ধাপের মতো। তাই একটিভিটি লাইফসাইকেল এর প্রতিটা ধাপ পিরামিডের পৃথক ধাপ। যখন সিস্টেম একটি নতুন একটিভিটি ইনস্ট্যান্স তৈরী করে, প্রতিটা কলব্যাক পদ্ধতি একটিভিটি স্টেটকে উপরের দিকে একধাপ এগিয়ে নেয়। পিরামিডের সবচেয়ে উপরের দিক হচ্ছে সেই পয়েন্ট যাতে একটিভিটি সম্মুখভাগে রান করে এবং ইউজার এর সাথে যোগাযোগ করতে পারে।

যখন ইউজার একটিভিটি ত্যাগ করা শুরু করে, সিস্টেম আরেকটি পদ্ধতি কল করে যা একটিভিটিকে বিচ্ছিন্ন করার জন্য একটিভিটি স্টেট কে পিরামিডের নীচের দিকে নামিয়ে আনে। কিছু কিছু ক্ষেত্রে, একটিভিটি শুধুমাত্র একটা অংশ নামিয়ে নিয়ে আসে এবং অপেক্ষা করে (যেমন, যখন ইউজার অন্য অ্যাপে চলে যায়), যে পয়েন্ট থেকে একটিভিটি আবার উপরে উঠে যেতে পারে (যদি ইউজার একটিভিটিতে ফিরে আসে) এবং যেখানে ইউজার এটা রেখে যায় সেখান থেকে শুরু করে।



ফিগার ১. একটিভিটি লাইফসাইকেলের একটি সরলীকৃত ডায়াগ্রাম, একটি পিরামিড স্টেপের প্রদর্শন। এটা দেখায় কীভাবে একটিভিটি রিজিউম স্টেট এর উপরের দিকে নিয়ে যেতে প্রতিটা কলব্যাক ব্যবহৃত হয়, এখানে আরও একটি কলব্যাক পদ্ধতি আছে যা একটিভিটিকে নীচের ধাপে নিয়ে যায়। একটিভিটি পজড স্টেট (অবস্থা) এবং স্টপড স্টেট থেকে রিজিউম স্টেটে ফিরেও আসতে পারে।

আপনার একটিভিটির জটিলতার উপর নির্ভর করে, সম্ভবত আপনার সকল লাইফসাইকেল মেথড বাস্তবায়ন করার দরকার নেই। যাহোক এটা গুরুত্বপূর্ণ যে আপনি প্রতিটা বিষয় বুঝতে পেরেছেন এবং বাস্তবায়ন করছেন যা ইউজার যা আশা করছে আপনার অ্যাপ সেভাবে কাজ করবে। আপনার একটিভিটি লাইফসাইকেল পদ্ধতির যথাযথ বাস্তবায়ন আপনার অ্যাপকে বিভিন্ন উপায়ে সুন্দরভাবে কাজ করাটাকে নিশ্চিত করবে, যার মধ্যে রয়েছে:

- আপনার অ্যাপ ব্যবহার করার সময় যদি ইউজারের কোন ফোন কল আসে বা অন্য কোন অ্যাপে চলে যায় তখন এটা একেবারে বন্ধ হয়ে যায় না।
- ইউজার এটাকে সক্রিয়ভাবে ব্যবহার না করলে মূল্যবান সিস্টেম রিসোর্স ব্যয় করে না
- যখন ইউজার অ্যাপ ছেড়ে চলে যায় এবং পড়ে আবার ফিরে আসে সেক্ষেত্রে এটা ইউজারের অগ্রগতিকে হারিয়ে যেতে দেয় না
- স্ক্রিন যখন ল্যান্ডস্কেপ এবং পোর্ট্রেইট এর মধ্যে রোটেট করে তখন এটা ইউজারের অগ্রগতিকে হারিয়ে যেতে অথবা বন্ধ হতে দেয় না

যখন আপনি নিম্নোক্ত অনুশীলনী শিখবেন, সেখানে বেশ কিছু অবস্থা আছে যেখানে একটি একটিভিটি বিভিন্ন স্টেটের (অবস্থানের) মধ্যে পরিবর্তন হয় যা ফিগার ১ এ দেখানো হয়েছে। কিনুত এই স্টেটগুলোর মধ্যে শুধুমাত্র তিনটা অনড় থাকে। তাই, একটিভিটি একটি বর্ধিত সময়ের জন্য তিনটার মধ্যে একটিতে অবস্থান করতে পারে:

Resumed/রিজিউমড

এই স্টেটে (অবস্থানে), একটিভিটি সামনে থাকে এবং ইউজার এটার সাথে যোগাযোগ করতে পারে। (মাঝে এটাকে “রানিং” স্টেট হিসাবে উল্লেখ করা হয়)

Paused/পজড

এই স্টেটে, একটিভিটি অন্য একটি দ্বারা আংশিকভাবে ঢাকা থাকে-অন্য একটিভিটি যেটা সামনে থাকে সেটা অর্ধস্বচ্ছ বা সম্পূর্ণ স্ক্রিন ঢেকে দেয় না। পজড একটিভিটিটি ইউজার ইনপুট গ্রহণ করে না এবং কোন কোড সম্পাদন করে না।

Stopped/স্টপড

এই স্টেটে, একটিভিটি সম্পূর্ণভাবে ঢাকা থাকে এবং ইউজার এটা দেখতে পায় না; এটা পেছনভাগে থাকার বিষয়টা বিবেচনা করে। যখন একটিভিটি বন্ধ হয় তখন একটিভিটি ইনসটেন্স এবং এর সকল স্টেট তথ্য যেমন মেম্বার ভেরিয়েবলসকে ধরে রাখা হয় কিনুত এটা কোন কোড সম্পাদন করতে পারে না।

অন্য স্টেট (ক্রিয়েটেড এবং স্টার্টেড) হচ্ছে অস্থায়ী এবং সিস্টেমটি অতিদ্রুত তাদের থেকে পরবর্তী স্টেটে চলে যায় পরবর্তী লাইফসাইকেল কলব্যাক মেথডকে কল করার মাধ্যমে। তাই সিস্টেমটি onCreate()কে কল করার পর, এটা দ্রুত onStart()কে কল করে, যা দ্রুততার সাথে onResume()কে অনুসরণ করে।

এটা মৌলিক একটিভিটি লাইফসাইকেল এর জন্য। এখন আপনি কিছু সুনির্দিষ্ট লাইফ সাইকেল আচরন সম্পর্কে শিখবেন।

আপনার অ্যাপের লঞ্চার একটিভিটি নির্দিষ্ট করুন

যখন হোম স্ক্রিন থেকে ইউজার আপনার অ্যাপ আইকন বাছাই করে, তখন সিস্টেম আপনার অ্যাপের Activity এর জন্য onCreate() পদ্ধতিকে কল করে, যাকে আপনি “লঞ্চার” (অথবা প্রধান/“main”) একটিভিটি হিসাবে ঘোষণা করে। এটা সেই একটিভিটি যা আপনার অ্যাপের ইউজার ইন্টারফেসের প্রধান প্রবেশ পথ হিসাবে কাজ করে।

আপনি নির্ধারণ করতে পারেন যে অ্যান্ড্রয়েড মেনিফেস্ট ফাইলে কোন একটিভিটিটি প্রধান একটিভিটি হিসাবে ব্যবহৃত হবে, AndroidManifest.xml, যা আপনার প্রজেক্ট ডিরেক্টরীর উৎসে আছে।

আপনার অ্যাপের জন্য প্রধান একটিভিটি অবশ্যই < intent-filter> সহ মেনিফেস্টে ডিক্লেয়ার করতে হবে যা MAIN একশন এবং LAUNCHER ক্যাটাগরি কে অন্তর্ভুক্ত করে। উদাহরণস্বরূপ:

```
<activity android:name=".MainActivity" android:label="@string/app_name">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

নোট: যখন আপনি অ্যান্ড্রয়েড এসডিকে টুলস দিয়ে একটি নতুন অ্যান্ড্রয়েড প্রজেক্ট তৈরি করেন, Activity ক্লাস সহ ডিফল্ট প্রজেক্ট ফাইলটি এই ফিল্টার দিয়ে মেনিফেস্টে ডিক্লেয়ার হয়।

যদি আপনার কোন একটা একটিভিটির জন্য হয় MAIN একশন বা LAUNCHER ক্যাটাগরি ডিক্লেয়ার করা না হয়ে থাকে, তখন হোম স্ক্রিনের অ্যাপের তালিকায় আপনার অ্যাপ আইকন দৃশ্যমান হবে না।

17.3 একটি নতুন ইনসটেন্স তৈরী করুন

অধিকাংশ অ্যাপ কিছু ভিন্ন ভিন্ন একটিভিটি অন্তর্ভুক্ত করে যা ইউজার কর্তৃক বিভিন্ন একশন সম্পাদন করাকে অনুমোদন করে। হতে পারে একটি একটিভিটি প্রধান একটিভিটি, যেটা তৈরী হয় যখন ইউজার আপনার অ্যাপ আইকনে ক্লিক করে অথবা একটি স্বতন্ত্র একটিভিটি যা আপনার অ্যাপ একটা ইউজার একশন রেসপন্স এ শুরু করে, সিস্টেমটি Activity এর প্রতিটা নতুন ইনসট্যান্স onCreate() পদ্ধতিকে কল করার মাধ্যমে তৈরী করে থাকে।

আপনাকে অবশ্যই বেসিক অ্যাপলিকেশন স্টার্টআপ লজিক সম্পাদন করতে onCreate() পদ্ধতি বাস্তবায়ন করতে হবে। উদাহরণস্বরূপ, আপনার onCreate()-এর বাস্তবায়নে ইউজার ইন্টারফেসকে নির্ধারণ করে দেওয়া উচিত এবং সম্ভাব্য ক্লাস-স্কোপ ভেরিয়েবলসকে ইনসটেনশিয়েট করা।

উদাহরণস্বরূপ, নিম্নোক্ত onCreate() পদ্ধতির উদাহরণ দেখায় কিছু কোড যা একটিভিটির জন্য কিছু মৌলিক সেটআপ সম্পাদন করে থাকে, যেমন, ইউজার ইন্টারফেস ডিক্লেয়ার করা (একটি এক্সএমএল লেআউট ফাইলে নির্ধারিত), মেম্বার ভেরিয়েবল নির্ধারণ, এবং কিছু ইউজার ইন্টারফেস কনফিগার করা।

```
TextView mTextView; // Member variable for text view in the layout

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Set the user interface layout for this Activity
    // The layout file is defined in the project res/layout/main_activity.xml file
    setContentView(R.layout.main_activity);

    // Initialize member TextView so we can manipulate it later
    mTextView = (TextView) findViewById(R.id.text_message);

    // Make sure we're running on Honeycomb or higher to use ActionBar APIs
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        // For the main activity, make sure the app icon in the action bar
        // does not behave as a button
        ActionBar actionBar = getActionBar();
        actionBar.setHomeButtonEnabled(false);
    }
}
```

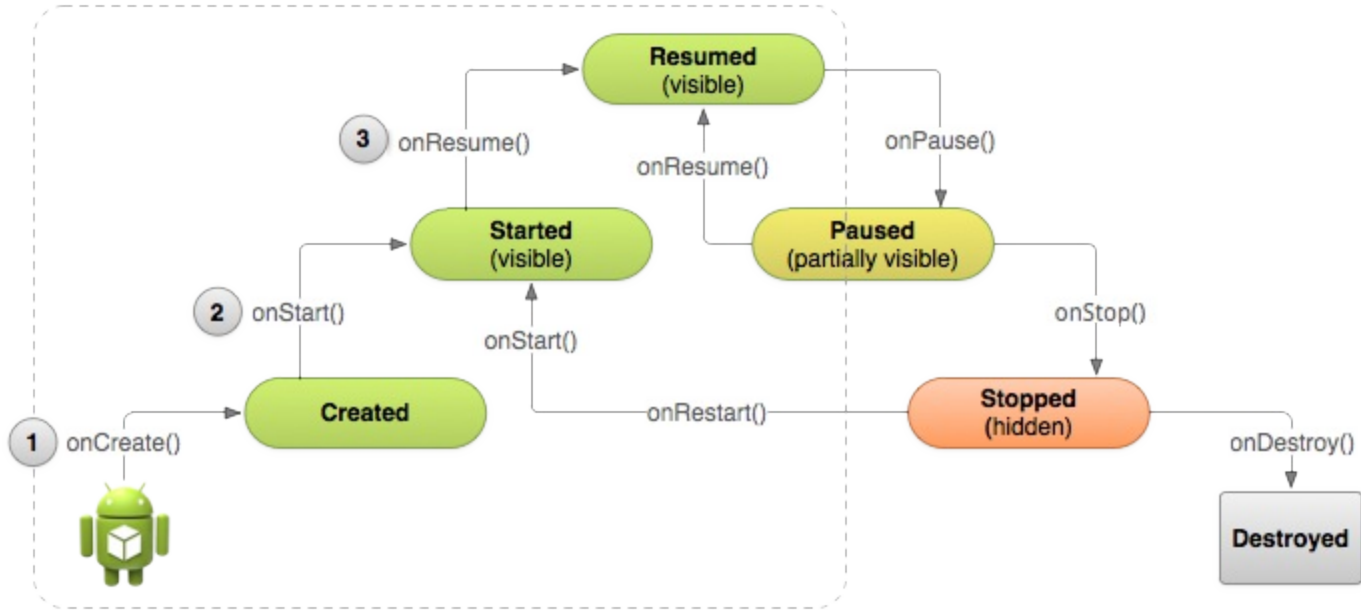
সতর্কতা: নতুন এপিআই (APIs) থেকে পুরাতন সিস্টেমকে পরিহার করতে SDK INT ব্যবহার করে অ্যান্ড্রয়েড ২.০ বা এর চেয়ে উন্নত সংস্করণে এই উপায়ে কাজ করে। পুরাতন সংস্করণ রানটাইম এক্সেপশন কে এনকাউন্টার করে।

যখন onCreate() বাস্তবায়ন শেষ করে, সিস্টেম onStart() এবং onResume() পদ্ধতিকে কে দ্রুত ক্রমানুসারে কল করে। আপনার একটিভিটি কখনই ক্রিয়েটেড এবং স্টাটেড স্টেটে থাকে না। কৌশলগতভাবে, একটিভিটি ইউজারের কাছে দৃশ্যমান হয় যখন onStart() কে কল করা হয়, কিনুত onResume() দ্রুত অনুসরণ করে এবং একটিভিটি রিজিউম স্টেটে থাকে যতক্ষণ পর্যন্ত না এটাকে পরিবর্তন করার জন্য কোন কিছু না ঘটে থাকে, যেমন যখন একটা ফোন কল রিসিভ করা হয়,

ইউজার আরেকটি একাউন্টতে নোভগেট করে অথবা ডিভাইস স্ক্রিন বন্ধ করা হয়।

অন্য অনুশীলনীতে যা অনুসরণ করে, আপনি দেখতে পাবেন যে স্টার্টআপ পদ্ধতি `onStart()` এবং `onResume()` কীভাবে আপনার একটিভিটির লাইফসাইকেলের জন্য প্রয়োজন যখন পজড এবং স্টপড স্টেট থেকে একটিভিটি পুনরায় শুরু করতে ব্যবহৃত হয়।

নোট: `onCreate()` পদ্ধতিটি `savedInstanceState` নামে একটি প্যারামিটার অন্তর্ভুক্ত করে যা পরবর্তী অনুশীলনীতে আলোচনা করা হবে।



ফিগার ২. তিনটা প্রধান কলব্যাক এর উপর জোর দেয়া একটিভিটি লাইফসাইকেল কাঠামোর আরেকটি চিত্র যাতে সিস্টেম ক্রমানুসারে কল করে যখন একটিভিটির একটা নতুন ইনস্ট্যান্স তৈরী করা হয়: `onCreate()`, `onStart()`, এবং `onResume()`। যখন এই কলব্যাকের ক্রম শেষ হয়, একটিভিটি রিজিউম স্টেট এ পৌছায় যেখানে ইউজার একটিভিটির সাথে যোগাযোগ করতে পারে যতক্ষণ না তারা অন্য একটিভিটিতে চলে যায়।

একটিভিটির বিলোপ

onCreate() যখন একটিভিটির প্রথম লাইফসাইকেল কলব্যাক হয়, onDestroy()টা তখন শেষ কলব্যাক হয়। সিস্টেমটি ফাইনাল সিগন্যাল হিসাবে আপনার একটিভিটির উপর এই পদ্ধতি কল করে যা আপনার একটিভিটি ইনসট্যান্সকে সম্পূর্ণরূপে সিস্টেম মেমরি থেকে সরিয়ে ফেলে।

অধিকাংশ অ্যাপস এর এই পদ্ধতি বাস্তবায়ন করার দরকার নেই কারণ লোকাল ক্লাস রেফারেন্স একটিভিটির সাথে বিলোপ হয়ে যায় এবং onPause() এবং onStop() এর সময় আপনার একটিভিটির সবচেয়ে পরিষ্কার করা উচিত। যাহোক যদি আপনার একটিভিটি ব্যাকগ্রাউন্ড থ্রেড অন্তর্ভুক্ত করে থাকে যা আপনি onCreate() অথবা অন্য লংরানিং রিসোর্স এর সময় তৈরী করেছেন যা মেমরিকে বের করে দিতে পারে (leak) যদি না তা যথাযথভাবে বন্ধ করা হয়, onDestroy() এর সময় আপনার এগুলোকে ধ্বংস করা উচিত।

```
@Override
public void onDestroy() {
    super.onDestroy(); // Always call the superclass

    // Stop method tracing that the activity started during onCreate()
    android.os.Debug.stopMethodTracing();
}
```

নোট: সিস্টেমটি ইতিমধ্যে onPause() এবং onStop() কে কল করার পর onDestroy() কে সব সময় কল করে একটা ব্যতীক্রম ছাড়া: যখন আপনি onCreate() পদ্ধতির মধ্যে থেকে finish() কে কল করবেন। কিছু কিছু ক্ষেত্রে যখন আপনার একটিভিটি আরেকটি একটিভিটি লঞ্চ করতে অস্থায়ী সিদ্ধান্ত গ্রহণকারী হিসাবে কাজ পরিচালনা করে, আপনি একটিভিটি বিলোপ (destroy) করতে onCreate() এর মধ্যে থেকে finish() কে কল করতে পারেন। এই ক্ষেত্রে সিস্টেমটি অন্য কোন লাইফসাইকেল পদ্ধতি গ্রহণ না করে তাৎক্ষনিকভাবে onDestroy() কে কল করে।

একটি একটিভিটির পজ এবং রিজিউম করা

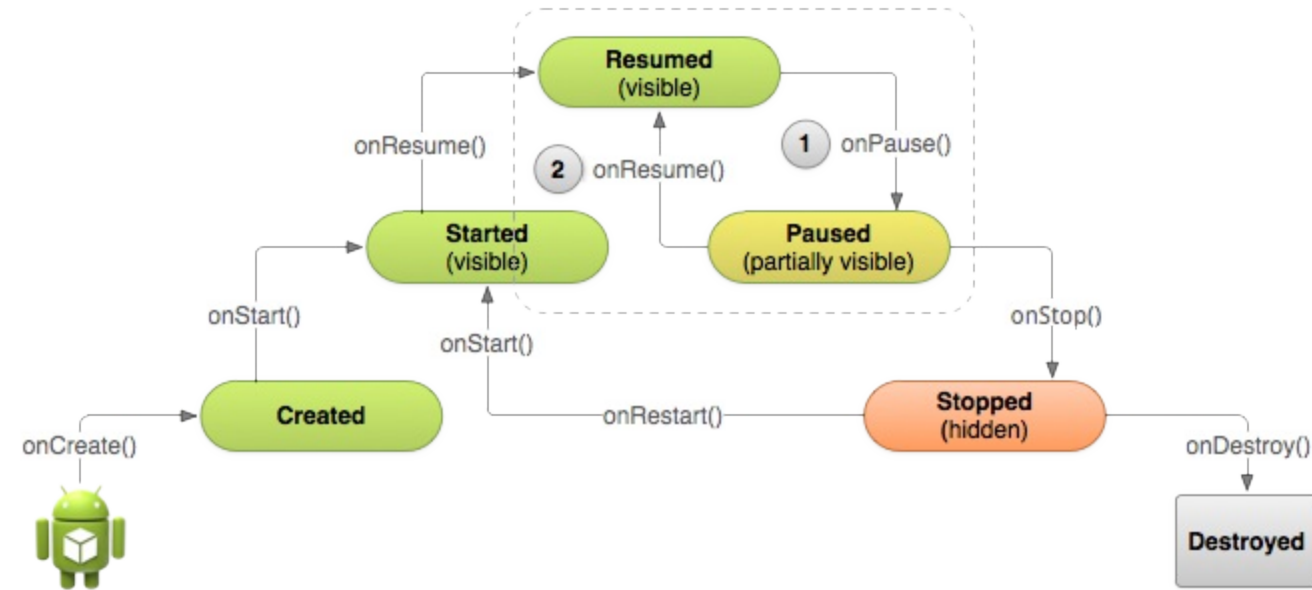
(<http://developer.android.com/training/basics/activity-lifecycle/pausing.html>)

অ্যাপের স্বাভাবিক ব্যবহারকালীন সময়ে সামনের একটিভিটিটি অনেক সময় অন্য কোন ভিজ্যুয়াল উপদান কর্তৃক বাধাপ্রাপ্ত হয় যা একটিভিটির পজ এর কারন। উদাহরণস্বরূপ, যখন একটি অর্ধ স্বচ্ছ একটিভিটি ওপেন করা হয় (যেমন একটি ডায়লগের কোন একটি স্টাইলে) তখন পূর্ববর্তী একটিভিটিটি পজ হয়ে যায়। যতক্ষণ এই একটিভিটি আংশিকভাবে দৃশ্যমান থাকবে কিনুত ফোকাসে থাকবে না, এটা পজ হিসাবে থাকবে।

যাহোক একটিভিটিটি যখন সম্পূর্ণভাবে বাধাপ্রাপ্ত হবে এবং দৃশ্যমান হবে না তখন এটা স্টপ (বন্ধ) হয়ে যাবে(এ বিষয়ে পরবর্তী অনুশীলনীতে আলোচনা করা হবে)।

যেহেতু আপনার একটিভিটি পজ অবস্থায় প্রবেশ করেছে, সিস্টেম আপনার Activity র উপর onPause()কে কল করে, যেটা আপনাকে একটি চলমান একশন/কর্মকান্ড কে থামিয়ে দিতে দেয়, যা পজ অবস্থায় চালিয়ে যাওয়া উচিত নয় (যেমন ভিডিও) অথবা কোন তথ্য ধরে রাখা যা একজন ইউজারের আপনার অ্যাপ ত্যাগ করার সময় স্থায়ীভাবে সেভ করে রাখা উচিত। যদি ইউজার পজ অবস্থান থেকে আপনার একটিভিটিতে ফিরে আসে, সিস্টেমটি এটাকে পূরণায় শুরু করে এবং onResume()মেথডকে কল করে।

নোট: যখন আপনার একটিভিটি onPause()কল গ্রহণ করবে, এটা একটা নির্দেশনা হবে যে একটিভিটি কিছু সময়ের জন্য পজ হবে এবং ইউজার আপনার একটিভিটিতে আবার ফিরে আসতে পারে। যাহোক, এটা সাধারণত প্রথম দিকনির্দেশনা যে ইউজার আপনার একটিভিটি ত্যাগ করেছে।



ফিগার ১. যখন একটি অর্ধস্বচ্ছ একটিভিটি আপনার একটিভিটিকে ঢেকে দিবে, সিস্টেম onPause()কে কল করবে এবং একটিভিটি পজ অবস্থায় অপেক্ষা করে (১)। যদি পজ থাকা অবস্থায় ইউজার একটিভিটিতে ফিরে আসে, সিস্টেমটি onResume()কে কল করে (২)।

একটিভিটিতে পজ দেওয়া

যখন সিস্টেমটি আপনার একটিভিটির জন্য onPause()কল করে, এটা কৌশলগতভাবে বোঝায় যে আপনার একটিভিটি এখনও আংশিকভাবে দৃশ্যমান, কিনুত প্রায়শই এরকম নির্দেশনা থাকে যে ইউজার একটিভিটি ত্যাগ করছে এবং এটা খুব শিঘ্রই স্টপ অবস্থায় প্রবেশ করে। নিম্নোক্ত ক্ষেত্রে আপনার onPause()কলব্যাক ব্যবহার করা উচিত:

- এনিমেশন বা অন্যান্য চলমান একশন/কর্মকান্ড বন্ধ করা যা সিপিইউ কে ব্যবহার করছে
- সেভ করা হয় নাই এমন পরিবর্তন গুলো সেভ করা, কিনুত শুধুমাত্র তখনই যখন ইউজার একটিভিটি ত্যাগ করে তখন স্থায়ীভাবে এই চেঞ্জগুলোকে সেভ হতে আশা করে। (যেমন, একটা ড্রাফট ইমেইল)
- যখন ইউজারের জন্য এটার প্রয়োজন নাই কিনুত এটা পজ অবস্থায় থেকে ব্যাটারীর আয়ুষ্কালের উপর প্রভাব ফেলে তখন ব্রডকাস্ট রিসিভার, সেন্সর করতে জিপিএস (GPS) বা অন্য কোন সিস্টেম রিসোর্স গুলোকে রিলিজ করে দিন।

উদাহরণস্বরূপ, যদি আপনার অ্যাপলিকেশন Camera ধব্যবহার করে, এটাকে রিলিজ করবে onPause()পদ্ধতি একটা ভালো জায়গা

```
@Override
public void onPause() {
    super.onPause(); // Always call the superclass method first

    // Release the Camera because we don't need it when paused
    // and other activities might need to use it.
    if (mCamera != null) {
        mCamera.release();
        mCamera = null;
    }
}
```

সাধারনভাবে, ইউজার চেঞ্জগুলোকে স্থায়ী স্টোরে স্টোর করতে onPause()ব্যবহার করা উচিত না (যেমন, একটি ফর্মে ব্যক্তিগত তথ্য প্রবেশ করানো)। আপনি শুধুমাত্র তখনই ইউজার চেঞ্জগুলোকে স্থায়ী স্টোরে onPause()এর মধ্যে থেকে ধরে রাখতে পারেন যখন আপনি নিশ্চিত হবেন যে ইউজার আশা করবে যে এই চেঞ্জগুলো অটো সেভ হবে (যেমন, যখন একটা ইমেইল ড্রাফট করা হয়ে থাকে)। যাহোক, onPause()এর সময় আপনার সিপিইউ-ইন্টেনসিভ কাজ করা পরিহার কার উচিত, যেমন একটি ডাটাবেজে লেখার কাজ, কারন এটা পরবর্তী একটিবিটিতে দৃশ্যমান পরিবর্তন কে ধীর গতির করে দেয় (পরিবর্তে, onStop()এর সময় আপনার হেভি লোড শাটডাউন অপারেশন করা উচিত)।

যদি আপনপর একটিভিটি আসলেই স্টপ/বন্ধ হয় তখন ইউজারের পরবর্তী লক্ষ্যে দ্রুত চলে যাওয়ার বিষয়টা (a speedy transition) সমর্থন করতে onPause()পদ্ধতির মধ্যে অপেক্ষাকৃত সহজভাবে যে

পারমান অপারেশন গুলো সম্পাদিত হয়েছে তা আপনার ধরে রাখা উচিত।

নোট: যখন আপনার একটিভিটি পজ করা হয়, তখন Activity ইনসটেন্স কে মেমরিতে রাখা হয় এবং এটাকে রিকল করা হয় যখন এটা আবার পূরণায় শুরু করা হয়। আপনার উপাদানগুলোকে পূরণায় শুরু করার দরকার নেই যা রিজিউম অবস্থায় নিয়ে যেতে যেকোন কলব্যাক মেথড এর সময় তৈরী করা হয়।

আপনার একটিভিটি রিজিউম (পুনরায় শুরু) করা

যখন ইউজার পজ অবস্থা থেকে আপনার একটিভিটি পুনরায় শুরু করবে (রিজিউম), তখন সিস্টেম `onResume()` পদ্ধতিকে কল করে।

এ বিষয়ে সতর্ক থাকা উচিত যে, আপনার একটিভিটি সম্মুখভাগে (ফোরগ্রাউন্ড) আসার ক্ষেত্রে সব সময়ই সময়ই সিস্টেমটি এই পদ্ধতিকে কল করে। যথা, কম্পোনেন্ট শুরু করতে আপনার `onResume()` বাস্তবায়ন করা উচিত যা আপনি `onResume()` এর সময় মুক্ত করে দিয়েছিলেন এবং অন্য যে কোন কিছু শুরুর করার সময় যা একটিভিটি রিজিউম অবস্থায় প্রবেশের প্রতিটা সময়েই ঘটে থাকে (যেমন, যখন শুধুমাত্র একটিভিটির ইউজার ফোকাস থাকে তখন এনিমেশন শুরু করা এবং কম্পোনেন্ট শুরু করা)।

নিম্নোক্ত `onResume()` এর উদাহরণটি উপরের `onResume()` এর উদাহরণের প্রতিক্রম, এটা ক্যামেরাকে শুরু করায় যা একটিভিটি পজের সময় মুক্ত করা হয়েছিল।

```
@Override
public void onResume() {
    super.onResume(); // Always call the superclass method first

    // Get the Camera instance as the activity achieves full user focus
    if (mCamera == null) {
        initializeCamera(); // Local method to handle camera init
    }
}
```

একটিভিটি স্টপ এবং রিস্টার্ট করা

(<http://developer.android.com/training/basics/activity-lifecycle/stopping.html>)

একটিভিটি লাইফসাইকেলের মধ্যে সার্বিকভাবে আপনার একটিভিটি স্টপ এবং রিস্টার্ট হওয়া একটা গুরুত্বপূর্ণ প্রক্রিয়া যা ইউজারের এ বিষয়ে উপলব্ধিকে নিশ্চিত করবে যাতে আপনার অ্যাপ সবসময় জীবন্ত থাকবে এবং তাদের যে অগ্রগতি তা ধরে রাখবে। নিচে কিছু চিত্র তুলে ধরা হলো যাতে আপনার একটিভিটি স্টপ এবং রিস্টার্ট করবে:

- যখন ইউজার আপনার সাম্প্রতিক অ্যাপ ওপেন করবে এবং আপনার অ্যাপ থেকে অন্য অ্যাপে চলে যাবে। আপনার অ্যাপের একটিভিটি যেটা বর্তমানে সম্মুখভাগে (ফোরগ্রাউন্ড) আছে তা বন্ধ হলে। যদি হোম স্ক্রিন লঞ্চার আইকন অথবা সাম্প্রতিক অ্যাপ উইন্ডোস থেকে ইউজার আপনার অ্যাপে ফিরে আসে, একটিভিটি পুনরায় শুরু করবে।
- ইউজার আপনার অ্যাপে এমন একটা কাজ করে, যা আরেকটি নতুন একটিভিটি শুরু করিয়ে দেয়। যখন দ্বিতীয় একটিভিটি তৈরী হয় তখন চলতি একটিভিটি বন্ধ (স্টপ) হয়ে যায়। এই সময় ইউজার যদি *Back button চাপে তাহলে প্রথম একটিভিটি আবার শুরু হবে (রিস্টার্টেড)।
- যখন ইউজার আপনার অ্যাপ ব্যবহার করার সময় একটি ফোন কল রিসিভ করে।

Activity ক্লাস `onStop()` এবং `onRestart()` এই দুই ধরনের লাইফসাইকেল পদ্ধতি প্রদান করে, যা আপনার একটিভিটি কীভাবে স্টপ এবং রিস্টার্টকে ধারণ করবে তা আপনাকে সুনির্দিষ্টভাবে করতে দেয়। এটা পজ অবস্থানের মতো নয় যা একটি আংশিক ইউজার ইন্টারফেস বাধাকে চিহ্নিত করে বরং স্টপ অবস্থা এর নিশ্চয়তা দেয় যে ইউজার ইন্টারফেস আর দৃশ্যমান হবে না এবং ইউজারের ফোকাস একটি ভিন্ন একটিভিটিতে থাকবে (অথবা একটা সম্পূর্ণ ভিন্ন অ্যাপে)।

নোট: কারন সিস্টেমটি আপনার Activity ইনসটেন্সকে সিস্টেম মেমরিতে ধরে রাখে যখন এটা বন্ধ (স্টপড) হয়, এটা সম্ভব যে আপনার `onStop()` এবং `onRestart()` অথবা এমনকি `onStart()` পদ্ধতিকেও বাস্তবায়নের দরকার নেই। অধিকাংশ একটিভিটি যা অপেক্ষাকৃতভাবে সহজ, একটিভিটি সুন্দরভাবে স্টপ এবং রিস্টার্ট করতে পারবে এবং চলতি কাজকে পজ দিতে এবং সিস্টেম রিসোর্স থেকে সংযোগ বিচ্ছিন্ন করতে আপনার শুধুমাত্র `onPause()` ব্যবহার করতে হতে পারে।

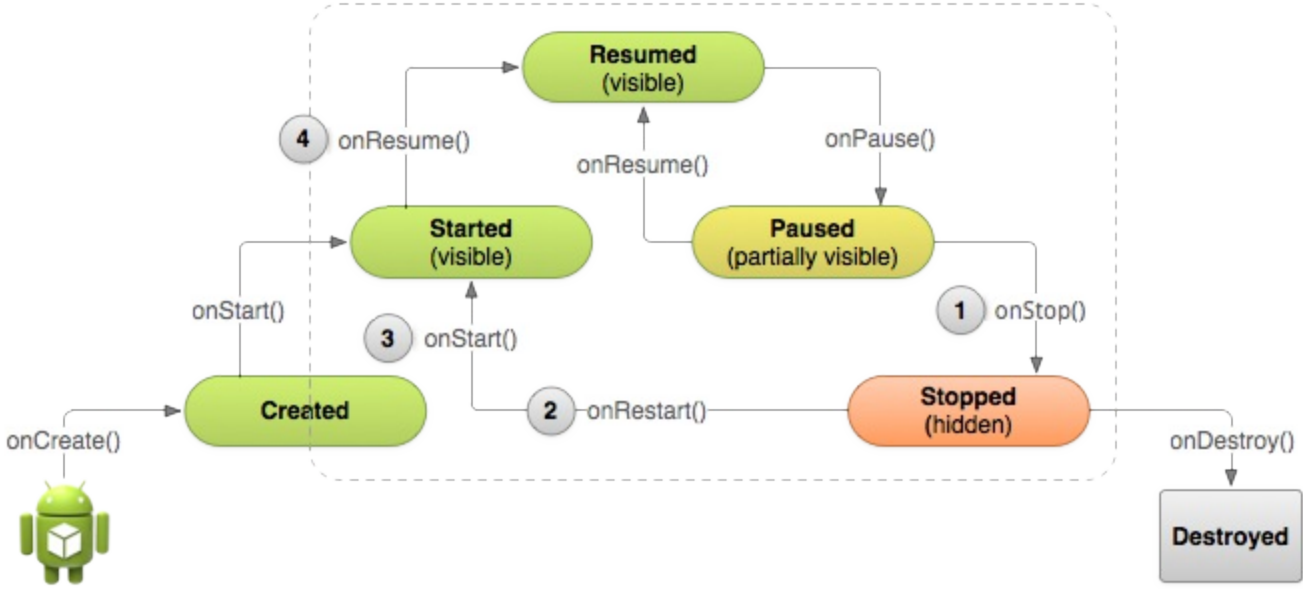


Figure 1. যখন ইউজার আপনার একটিভিটি ত্যাগ করে, সিস্টেমটি একটিভিটি বন্ধ (স্টপ) করতে `onStop()`কে কল করে (১)। যদি ইউজার একটিভিটি বন্ধ (স্টপড) থাকা অবস্থায় যখন আবার ফিরে আসে, সিস্টেমটি দ্রুত `onStart()` (৩) এবং `onResume()` (৪) কে অনুসরণ করে `onRestart()`কে কল করে (২)। উল্লেখ্য যে কোন বিষয়টা একটিভিটিকে স্টপ করার কারন এটা কোন বিষয় নয়, সিস্টেম সবসময়ই `onStop()`কে কল করার পূর্বে `onPause()`কে কল করে।

আপনার একটিভিটি বন্ধ (স্টপ) করুন

যখন আপনার একটিভিটি `onStop()` পদ্ধতি এ একটি কল রিসিভ করে, এটা আর দৃশ্যমান থাকে না এবং অধিকাংশ রিসোর্সই যার প্রয়োজন নাই যখন ইউজার এটা ব্যবহার করছে না তা মুক্ত (রিলিজ) করে দেয়া উচিত। যখন আপনার একটিভিটি বন্ধ (স্টপ) হবে, সিস্টেম ইনসটেন্সকে ধ্বংস করে দিবে যদি এর সিস্টেম মেমরির উন্নতি করার প্রয়োজন হয়। চরম পর্যায়ে সিস্টেমটি একটিভিটির চূড়ান্ত `onDestroy()` কলব্যাক কে কল না করেই আপনার অ্যাপ প্রসেস কে সরাসরি মেরে ফেলতে পারে, সুবরাং এটা গুরুত্বপূর্ণ যে মেমরিকে লিক করতে পারে এমন রিসোর্সগুলোকে রিলিজ করতে `onStop()` ব্যবহার করতে পারেন।

যদিও `onPause()` কে পদ্ধতি `onStop()` এর পূর্বে কল করা হয়, আরও ভালোভাবে কার্যসম্পাদন করতে, আরও সিপিইউ-ইন্টেনসিভ শাট-ডাউন অপারেশন করতে আপনার `onStop()` ব্যবহার করা উচিত, যেমন, একটি ডাটাবেজে তথ্য লেখার কাজ করতে।

উদাহরণস্বরূপ, এখানে `onStop()` এর একটি বাস্তবায়ন দেয়া আছে যা একটি ড্রাফট নোটের কনটেন্ট অনড় স্টোরেজে সেভ করে:

```
@Override
protected void onStop() {
    super.onStop(); // Always call the superclass method first

    // Save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    ContentValues values = new ContentValues();
    values.put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText());
    values.put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle());

    getResolver().update(
        mUri, // The URI for the note to update.
        values, // The map of column names and new values to apply to them.
        null, // No SELECT criteria are used.
        null // No WHERE columns are used.
    );
}
```

যখন আপনার একটিভিটি বন্ধ (স্টপ) হবে, Activity অবজেক্টটি মেমরিতে রাখা হয় এবং যখন একটিভিটি পুনরায় শুরু (রিজিউমড) হয় তখন এটাকে আবার রিকল করা হয়। ক্রমে ক্রমে রিজিউম অবস্থায় নিয়ে যেতে যেকোন কলব্যাক মেথড এর সময় তৈরী করা কম্পোনেন্ট এর পুনঃপ্রবর্তনের প্রয়োজন নেই। সিস্টেমটি লেআউটের মধ্যে প্রতিটা View এর জন্য চলতি অবস্থার ট্র্যাকগুলোকে ধরেও রাখে, সুতরাং ইউজার যদি EditText উইজিটের (widget) মধ্যে টেক্সট প্রবেশ করায়, ওই কনটেন্ট যথাস্থানে রাখা থাকে তাই আপনার এটাকে সেভ কার এবং রিস্টোর করার প্রয়োজন নেই।

নোট: এমনকি যদি সিস্টেমটি আপনার একটিভিটি কে ধ্বংস করে দেয় যখন এটা বন্ধ থাকে, এটা তখনও একটি Bundle এ (কি ভ্যালু পেয়ার এর একটি বিন্দু) View অবজেক্টের (যেমন, একটি EditText এর মধ্যে টেক্সট) স্টেটকে ধরে রাখে এবং তাদেরকে রিস্টোর করে যদি ইউজার আপনার

একাটাভাটর একই ইনসটেন্স এ নোভিগেট করে ফিরে আসে (পরবর্তী অনুশীলনীতে (next lesson) আপনার একটিভিটি ধ্বংস হওয়া বা পুননির্মাণের ক্ষেত্রে অন্যান্য সেট ডাটা সেভ করতে Bundle ব্যবহার করার বিষয় সম্পর্কে বিস্তারিত আলোচনা করা হবে)।

আপনার একটিভিটি স্টার্ট/রিস্টার্ট করুন

যখন আনপর একটিভিটি স্টপ অবস্থা থেকে সম্মুখভাগে (ফোরগ্রাউন্ড) ফিরে আসে, এটা `onRestart()`এ একটা কল রিসিভ করে। সিস্টেমটি `onStart()` পদ্ধতিকেও কল করে যা আপনার একটিভিটির প্রতিবার দৃশ্যমান হওয়া ক্ষেত্রেই কল করে (যখন প্রথমবারের মতো রিস্টার্ট করা বা তৈরী করা হয়)। যাহোক `onRestart()` পদ্ধতিকে শুধুমাত্র তখনই কল করা হয় যখন একটিভিটি স্টপ অবস্থা থেকে আবার পূরণায় কাজ শুরু করে, সুতরাং এটা আপনি বিশেষ রিস্টোরেশন কাজে ব্যবহার করতে পারবেন যা শুধুমাত্র তখনই প্রয়োজনীয় হতে পারে একটিভিটিটি যদি পূর্বেই বন্ধ (স্টপড) হয়ে থাকে, কিনুত ধ্বংস না হয়।

এটা কমই দেখা যায় যে একটি অ্যাপের একটিভিটির অবস্থা রিস্টোর করতে `onRestart()` ব্যবহার করার দরকার হয়, সুতরাং এখানে এই পদ্ধতির জন্য কোন গাইডলাইন নেই যা অ্যাপের সাধারণ পপুলেশনে প্রয়োগ করা হয়। যাহোক যেহেতু আপনার `onStop()` পদ্ধতির উচিত আপনার একটিভিটির সকল রিসোর্স পরিস্কার করে ফেলা, যখন একটিভিটিটি রিস্টার্ট করবে তখন আপনার এগুলোকে রি-ইনসটেনসিয়েট করা দরকার। যখন আপনার একটিভিটি প্রথমবারের মতো তৈরী করবেন (যখন একটিভিটির কোন অস্তিত্ব ছিল না) আপনার তখনও এটাকে ইনসটেনসিয়েট করতে হবে। এই কারনে, আপনার `onStop()` পদ্ধতি এর প্রতিরূপ হিসাবে `onStart()` কলব্যাক পদ্ধতি স্বাভাবিকভাবে ব্যবহার করা উচিত, কারন সিস্টেম `onStart()` উভয়ই কল করে যখন এটা আপনার একটিভিটি তৈরী করে এবং যখন এটা বন্ধ (স্টপড) অবস্থা থেকে একটিভিটিকে রিস্টার্ট করে।

উদাহরণস্বরূপ, যেহেতু ইউজার এখানে ফিরে আসার আগে দীর্ঘ একটা সময়ের জন্য আপনার অ্যাপ থেকে চলে যাচ্ছে, `onStart()` পদ্ধতি এটা প্রমান করার একটা ভালো জায়গা যে প্রয়োজনীয় সিস্টেম ফিচার সক্রিয় আছে:

```
@Override
protected void onStart() {
    super.onStart(); // Always call the superclass method first

    // The activity is either being restarted or started for the first time
    // so this is where we should make sure that GPS is enabled
    LocationManager locationManager =
        (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    boolean gpsEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);

    if (!gpsEnabled) {
        // Create a dialog here that requests the user to enable GPS, and use an intent
        // with the android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS action
        // to take the user to the Settings screen to enable GPS when they click "OK"
    }
}

@Override
protected void onRestart() {
    super.onRestart(); // Always call the superclass method first

    // Activity being restarted from stopped state
}
```

যখন সিস্টেমাট আপনার একাটাভাট /ধ্বংস করে দেয়, এটা আপনার Activity জন্য onDestroy() পদ্ধতি কল করে। কারন স্বাভাবিকভাবে আপনি onStop()এর সাথে আপনার অধিকাংশ রিসোর্স রিলিজ করে দিয়েছেন, ঠিক এই সময়ে আপনি onDestroy()এ একটা কল রিসিভ করেছেন, অধিকাংশ অ্যাপের এখানে খুব বেশী কিছু করার দরকার নেই। এই পদ্ধতি রিসোর্স পরিষ্কার করে ফেলার একটা শেষ সুযোগ যা মেমরি রিকের দিকে নিয়ে যেতে পাও, সুতরাং আপনার নিশ্চিত হওয়া উচিত যে অতিরিক্ত থ্রেড ধ্বংস হয়েছে এবং মেথড ট্রেসিং এর মতো দীর্ঘ মেয়াদী কর্মকান্ডও বন্ধ (স্টপড) হয়েছে।

একটিভিটি পুনরায় তৈরী করা

(<http://developer.android.com/training/basics/activity-lifecycle/recreating.html>)

এমন অনেক অবস্থা আছে যেখানে আপনার একটিভিটি স্বাভাবিক অ্যাপ কর্মকালন্ডের কারনে ধ্বংস হয়, যেমন ইউজার যখন **Back** button চাপে অথবা আপনার একটিভিটি finish()কে কল করে এটা নিজেকেই ধ্বংস করে। সিস্টেমটি আপনার একটিভিটিকে ধ্বংস করতে পারে যদি এটা বর্তমানে বন্ধ (স্টপড) থাকে এবং দীর্ঘ সময় এর কোন ব্যবহার করা না হয় বা ফোরগ্রাউন্ড একটিভিটি আরও রিসোর্স দাবী করে যার ফলে সিস্টেম অবশ্যই মেমরি রিকভার করতে ব্যাকগ্রাউন্ড প্রসেসকে শাটডাউন করে দেয়।

ইউজারের **Back** চাপার কারনে অথবা একটিভিটি নিজেই এটাকে শেষ করে দেওয়ার কারনে যখন আপনার একটিভিটি ধ্বংসপ্রাপ্ত হয়, ওই Activity ইনসটেন্স এর সিস্টেমের ধারণা চিরতরে চলে যায় কারন এই কর্মকালন্ড নির্দেশ করে যে একটিভিটিটির আর কোন প্রয়োজন নাই। যাহোক, সিস্টেমের সীমাবদ্ধতার কারনে (স্বাভাবিক কর্মকালন্ড না হয়ে) যদি সিস্টেম একটিভিটিটি ধ্বংস করে দেয়, তখনও আসল Activity ইনসটেন্স চলে যাবে, সিস্টেম এটাকে স্মরন করবে যে এটার অস্তিত্ব আছে যদি ইউজার নেভিগেট করে ফিরে আসে, সিস্টেমটি এক সেট সেভ করা ডাটা ব্যবহার করে একটিভিটির একটা নতুন ইনসটেন্স তৈরী করে যা একটিভিটির অবস্থার বর্ণনা করে যখন এটা ধ্বংস হয়েছিল। সেভ করা ডাটা যেটা সিস্টেম পূর্ববর্তী অবস্থা রিস্টোর করতে ব্যবহার করে তাকে "instance state" বলে এবং এটা হচ্ছে Bundle অবজেক্টে সেটা হওয়া কি ভ্যালু পেয়ারস এর কালেকশন।

সতর্কতা: ইউজার স্ক্রিন রোট্টে করলে প্রতিবারই আপনার একটিভিটি ধ্বংস এবং নতুন করে তৈরী হবে। যখন স্ক্রিন এর ওরিয়েন্টেশন বদলাবে তখন সিস্টেম ফোরগ্রাউন্ড একটিভিটি ধ্বংস করবে এবং নতুন কওে তৈরী করবে কারন স্ক্রিন কনফিগারেশন পরিবর্তন হয়েছে এবং আপনার একটিভিটির বিকল্প রিসোর্স লোড করার দরকার হতে পারে (যেমন, লেআউটটি)।

বাই ডিফল্ট, আপনার একটিভিটি লেআউটের প্রতিটা View অবজেক্ট সম্পর্কিত তথ্য সেভ করতে সিস্টেমটি Bundle ইনসটেন্স অবস্থা ব্যবহার করে (যেমন, টেক্সট ভ্যালু EditText অবজেক্টে প্রবেশ করে)। সুতরাং, যদি আপনার একটিভিটি ইনসটেন্স ধ্বংস হয় বা নতুন করে তৈরী হয়, আপনার চাওয়া কোন কোড ছাড়াই লেআউটের অবস্থা এর পূর্বস্থায় রিস্টোর হয়। যাহোক আপনার একটিভিটির সম্ভবত আরও অবস্থাগত তথ্য আছে যা আপনি রিস্টোর করতে চান, যেমন মেম্বার ভেরিয়েবল যা একটিভিটিতে ইউজারের অগ্রগতিকে অনুসরণ করে।

নোট: অ্যান্ড্রয়েড সিস্টেমের জন্য আপনার একটিভিটির ভিউয়ের অবস্থা রিস্টোর করতে, android:id এট্রিবিউট কর্তৃক প্রদত্ত প্রতিটা ভিউয়ের অবশ্যই স্বতন্ত্র আইডি থাকা উচিত।

একটিভিটি অবস্থা সম্পর্কে অতিরিক্ত ডাটা সেভ করতে, আপনাকে অবশ্যই onSaveInstanceState()কলব্যাক মেথডকে অগ্রাহ্য করতে হবে। যখন ইউজার আপনার একটিভিটি ত্যাগ করছে তখন সিস্টেম এটাকে কল করে এবং এটাকে Bundle অবজেক্টে পাস করে দেয় যা ইভেন্টে সেভ হয়ে যাবে যাতে আপনার একটিভিটি অনাকাঙ্ক্ষিতভাবে ধ্বংস হয়ে যায়। যদি সিস্টেম অবশ্যই পরবর্তীতে একটিভিটি ইনসটেন্স নতুন করে তৈরী করে, এটা একই Bundle অবজেক্ট onRestoreInstanceState()এবং onCreate()উভয় পদ্ধতির দিকে পাস করে দেয়।



ফিগার ২. যেহেতু সিস্টেম আপনার একটিভিটি ষ্টপ করতে শুরু হয়েছে, এটা `onSaveInstanceState()` কে কল করে (১) যাতে ইনসটেন্স অবশ্যই নতুন করে তৈরী করতে হবে এমন ক্ষেত্রে আপনি অতিরিক্ত স্টেট ডাটাকে সুনির্দিষ্ট করতে পারেন যা আপনি সেভ করতে চান। যদি একটিভিটি ধ্বংস হয়ে থাকে এবং একই ইনসটেন্স অবশ্যই নতুন করে তৈরী করতে হবে, সিস্টেম স্টেট ডাটা যা (১) এ নির্ধারন করা হয়েছে তাকে `onCreate()` পদ্ধতি (২) এবং `onRestoreInstanceState()` উভয় (৩) পদ্ধতির দিকে পাস করে।

আপনার একটিভিটি স্টেট (অবস্থান) সেভ করা

যেহেতু সিস্টেম আপনার একটিভিটি স্টপ করতে শুরু হয়েছে, এটা onSaveInstanceState()কে কল করে (১) যাতে আপনার একটিভিটি একটি কী ভ্যালু কালেকশনের সাথে স্টেট তথ্যকে সেভ করতে পারে। এই পদ্ধতির ডিফল্ট বাস্তবায়ন একটিভিটির ভিউ হায়ারার্কির স্টেট সম্পর্কিত তথ্য সেভ কওে, যেমন EditText উইজিট (widget) এর মধ্যে টেক্সট অথবা ListView এর স্ক্রলের অবস্থান।

আপনার একটিভিটির জন্য অতিরিক্ত স্টেট তথ্য সেভ করতে, আপনাকে অবশ্যই ডুহ্লাধাবওহঃধহপবঝঃধঃব()এবং ইঁহফষবঅবজেক্টে কী ভ্যালু পেয়ারস বাস্তবায়ন করতে হবে।
উদাহরণস্বরূপ:

```
static final String STATE_SCORE = "playerScore"; static final String STATE_LEVEL = "playerLevel"; ...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save the user's current game state
    savedInstanceState.putInt(STATE_SCORE, mCurrentScore);
    savedInstanceState.putInt(STATE_LEVEL, mCurrentLevel);

    // Always call the superclass so it can save the view hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

সতর্কতা: সবসময় onSaveInstanceState()এর সুপারক্লাস ইমপ্লিমেন্টেশনকে কল করুন যাতে ডিফল্ট ইমপ্লিমেন্টেশন ভিউ হায়ারার্কির স্টেট কে সেভ করতে পারে।

আপনার একটিভিটি স্টেটকে (অবস্থান) রিস্টোর করুন

আপনার একটিভিটি যখন পূর্বে ধ্বংস হয়ে আবার নতুন করে তৈরী হয়, আপনি Bundle থেকে আপনার সেভ স্টেট কে উদ্ধার করতে পারেন যা সিস্টেম আপনার একটিভিটিতে পাস করে। onCreate() এবং onRestoreInstanceState() উভয় কলব্যাক মেথডই একই Bundle গ্রহণ করে যা ইনসটেন্স স্টেট তথ্য ধারণ করে।

কারণ onCreate() পদ্ধতিকে কল করা হয় যখন সিস্টেম আপনার একটিভিটির একটা নতুন ইনসটেন্স তৈরী করেছে অথবা পূর্ববর্তী একটাকে নতুন করে তৈরী করেছে, এটা পড়তে শুরু করার পূর্বেই আপনাকে অবশ্যই চেক করতে হবে যে স্টেট Bundle অগ্রহণযোগ্য কিনা। যদি এটা অগ্রহণযোগ্য হয়, তখন সিস্টেম একটিভিটির নতুন ইনসটেন্স তৈরী করতে থাকে, পূর্ববর্তী কোন একটা যা ধ্বংস হয়েছে তাকে রিস্টোর না করে।

উদাহরণস্বরূপ, এখানে দেখানো হচ্ছে কীভাবে আপনি onCreate() এর মধ্যে স্টেট ডাটা রিস্টোর করতে পারেন:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass first

    // Check whether we're recreating a previously destroyed instance
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
        mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
    } else {
        // Probably initialize members with default values for a new instance
    }
    ...
}
```

onCreate() এর সময় স্টেট রিস্টোর না করে আপনি onRestoreInstanceState() কে বাস্তবায়ন করাকে বেছে নিতে পারেন, যা সিস্টেম onStart() পদ্ধতির পণ্ডে কল করে। সিস্টেম শুধুমাত্র onRestoreInstanceState() কল করে যদি সেখানে রিস্টোর করতে একটি সেভ করা স্টেট থাকে, সুতরাং আপনাকে আর পরীক্ষা করতে হবে না যে Bundle গ্রহণযোগ্য:

```
public void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);

    // Restore state members from saved instance
    mCurrentScore = savedInstanceState.getInt(STATE_SCORE);
    mCurrentLevel = savedInstanceState.getInt(STATE_LEVEL);
}
```

সতর্কতা: সবসময়ই `onRestoreInstanceState()` এর সুপারক্লাস ইমপ্লিমেন্টেশন কল করুন যাতে ডিফল্ট ইমপ্লিমেন্টেশন ভিউ হায়ারকারির স্টেটটিকে রিস্টোর করতে পারে।

রান টাইমের সময় ইভেন্ট রিস্টার্ট করার জন্য আপনার একটিভিটি নতুন করে শুরু করা (যেমন, যখন স্ক্রিন রোটের করে) সম্পর্কিত আরও তথ্য জানতে `Handling Runtime Changes` (<http://developer.android.com/guide/topics/resources/runtime-changes.html>) পড়ুন।

ফ্র্যাগমেন্ট সহ ডাইনামিক ইউজার ইন্টারফেস তৈরী করা

(<http://developer.android.com/training/basics/fragments/index.html>)

অ্যান্ড্রয়েডে একটি ডাইনামিক এবং মাল্টি-পেন ইউজার ইন্টারফেস তৈরী করতে, আপনার ইউজার ইন্টারফেস কম্পোনেন্ট এবং একটিভিটি আচরণকে মডিউল এ সংযুক্ত করা দরকার যাতে আপনি আপনার একটিভিটির ভিতরে বা বাইরে অদল বদল করতে পারেন। আপনি এই মডিউল Fragment ক্লাস দিয়ে তৈরী করতে পারবেন, যা যে কোনভাবেই হোক একটি নেস্টেড একটিভিটির মতো আচরণ করে যাতে আপনি এর নিজস্ব লেআউট নির্ধারন করতে পারেন এবং এর নিজস্ব লাইফসাইকেল ব্যবস্থাপনা করতে পারেন।

যখন একটি ফ্রাগমেন্ট এর নিজস্ব লেআউট সুনির্দিষ্ট করে, বিভিন্ন স্ক্রিন সাইজের জন্য আপনার লেআউট কনফিগারেশন পরিবর্তন করতে একটি একটিভিটির মধ্যে অন্যান্য ফ্রাগমেন্টের সাথে বিভিন্ন কন্টেনশনের মধ্যে কনফিগার হতে পারে (একটি ছোট স্ক্রিন এক সাথে একটি ফ্রাগমেন্ট দেখাতে পারে, কিনুত একটি বড় স্ক্রিন দু ইবা ততোধিক ফ্রাগমেন্ট দেখাতে পারে)।

এই ক্লাস আপনাকে দেখাবে কীভাবে ফ্রাগমেন্ট সহ একটি ডাইনামিক ইউজার এক্সপেরিয়েন্স তৈরী করতে পারে এবং বিভিন্ন স্ক্রিন সাইজ সহ ডিভাইসের জন্য আপনার অ্যাপসের ইউজার এক্সপেরিয়েন্স কে অপটিমাইজ করে, যা সারাক্ষণ অ্যান্ড্রয়েড ১.৬ এর মতো পুরাতন ভার্সনে রান করা ডিভাইসকে সমর্থন করে।

এই অধ্যায়ের অনুশীলনী সমূহ

একটি ফ্রাগমেন্ট তৈরী করা

শিখুন কীভাবে একটি ফ্রাগমেন্ট তৈরী করতে হয় এবং এটার কলব্যাক মেথডের মধ্যে থেকে মৌলিক আচরণ বাস্তবায়ন করতে হয়।

একটি নমনীয় (ফ্লেক্সিবল) ইউজার ইন্টারফেস তৈরী করা

শিখুন কীভাবে লেআউটসহ আপনার অ্যাপস তৈরী করবেন যা বিভিন্ন স্ক্রিনের জন্য বিভিন্ন ফ্রাগমেন্ট কনফিগারেশন প্রদান করে।

অন্যান্য ফ্রাগমেন্টের সাথে যোগাযোগ স্থাপন

শিখুন কীভাবে একটি ফ্রাগমেন্ট থেকে একটিভিটিতে বা অন্য ফ্রাগমেন্টে কমিউনিকেশন/যোগাযোগের উপায় সেট করতে হয়।

একটি ফ্রাগমেন্ট তৈরী করা

(<http://developer.android.com/training/basics/fragments/creating.html>)

আপনি একটি একটিভিটির মডিউলার সেকশন হিসাবে একটি ফ্রাগমেন্টের চিন্তা করতে পারেন, এর নিজস্ব লাইফসাইকেল আছে, এটা নিজস্ব ইনপুট ইভেন্ট গ্রহণ করে এবং একটিভিটি যখন রান করে তখন আপনি এটা সংযোজন বা রিযোজন করতে পারেন (কতকাংশে “সাবএকটিভিটির” মতো যা আপনি বিভিন্ন একটিভিটিতে পূর্ণব্যবহার করতে পারেন)। এই অনুশীলনী আপনাকে শেখাবে কীভাবে Support Library ব্যবহার করে Fragment টিকে বর্ধিত করা যায় যাতে আপনার অ্যাপ অ্যান্ড্রয়েড ১.৬ এর মতো পুরাতন ভার্সন ডিভাইসে ব্যবহার করার মতো উপযুক্ত রাখে।

নোট: আপনি যদি ঠিক করে থাকেন যে আপনার অ্যাপের জন্য এপিআই লেভেল কমপক্ষে ১১ বা এর চেয়ে বেশী হবে, আপনার সাপোর্ট লাইব্রেরী ব্যবহার করার দরকার নেই এবং পরিবর্তে আপনি ফ্রেমওয়ার্কের বিল্ট ইন Fragment ক্লাস এবং এ সম্পর্কিত এপিআই ব্যবহার করতে পারেন। শুধু সতর্ক থাকবেন যে এই অনুশীলনী সাপোর্ট লাইব্রেরী থেকে এপিআই ব্যবহার করাটাকে ফোকাস করে, যা প্লাটফর্মে অন্তর্ভুক্ত ভার্সনের চেয়ে একটি স্পেসিফিক প্যাকেজ সিগনেচার এবং মাঝে মাঝে সামান্য ভিন্ন এপিআই নাম ব্যবহার করে।

এই অনুশীলনী শুরু করার পূর্বে, সাপোর্ট লাইব্রেরী ব্যবহার করার জন্য অবশ্যই আপনার অ্যান্ড্রয়েড প্রজেক্ট সেটআপ করতে হবে। আপনি যদি ইতিপূর্বে সাপোর্ট লাইব্রেরী ব্যবহার করে না থাকেন, Support Library Setup (<http://developer.android.com/tools/support-library/setup.html>) ডকুমেন্ট অনুসরণ করে v4 লাইব্রেরী করতে আপনার প্রজেক্ট সেটআপ করুন। আবার v7 appcompat লাইব্রেরী ব্যবহার করার পরিবর্তে আপনি আপনার একটিভিটিতে action bar (<http://developer.android.com/guide/topics/ui/actionbar.html>) অন্তর্ভুক্তও করতে পারেন, যা অ্যান্ড্রয়েড ২.১ (এপিআই লেভেল ৭) এর উপযুক্ত এবং Fragment এপিআইও অন্তর্ভুক্ত করে।

একটি ফ্রাগমেন্ট ক্লাস তৈরী করুন

ফ্রাগমেন্ট তৈরী করতে Fragment ক্লাসটি বর্ধিত করুন, এরপর আপনার অ্যাপ লজিকে কি লাইফ সাইকেল মেথড ওভাররাইড করুন, ঠিক একই রকম ভাবে যেমন একটি Activity ক্লাসের সাথে থাকবেন।

যখন Fragment তৈরী করা হয় তখন একটা ভিন্নতা থাকে তা হলো লেআউট নির্ধারন করতে অবশ্যই আপনাকে onCreateView() কলব্যাক ব্যভহার করতে হবে। মূলত, এটা একমাত্র কলব্যাক একটা ফ্রাগমেন্ট রানিং অবস্থায় পাওয়ার জন্য যেটা আপনার প্রয়োজন। উদাহরনস্বরূপ, এখানে একটি সহজ ফ্রাগমেন্ট আছে যা এর নিজস্ব লেআউট কে নির্ধারন করে:

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```

ঠিক একটি একটিভিটির মতো, একটি ফ্রাগমেন্টের অন্যান্য লাইফসাইকেল কলব্যাক বাস্তবায়ন করা উচিত যা আপনাকে এর স্টেট এর ব্যবস্থাপনা করতে দিবে যেহেতু এটা একটিভিটি থেকে সংযোজিত বা বিয়োজিত হবে এবং যেহেতু একটিভিটি এর লাইফসাইকেল স্টেটের মধ্যে পরিবর্তন হবে। উদাহরনস্বরূপ, যখন একটিভিটির onPause() মেথড কে কল করা হবে, একটিভিটির যেকোন ফ্রাগমেন্টও onPause() এ একটা কল রিসিভ করে।

ফ্রাগমেন্ট লাইফসাইকেল এবং কলব্যাক মেথডের জন্য আরও তথ্য Fragments ডেভেলপ গাইডে আছে।

XML ব্যবহার করে একটিভিটিতে ফ্র্যাগমেন্ট সংযোজন করুন

যখন ফ্র্যাগমেন্ট পূর্ণব্যবহারযোগ্য, মডিউলার ইউজার ইন্টারফেস কম্পোনেন্ট, একটি Fragment ক্লাসের প্রতিটা ইনস্টেন্স অবশ্যই একটি প্যারেন্ট FragmentActivity এর সাথে সম্পৃক্ত থাকতে হবে। আপনার একটিভিটি লেআউট এক্সএমএল ফাইলের মধ্যে প্রতিটা ফ্র্যাগমেন্ট নির্ধারন করে আপনি এই সম্পৃক্ততা অর্জন করতে পারেন।

নোট: এপিআই লেভেল ১১ এর চেয়ে পুরাতন সিস্টেম ভার্সনে ফ্র্যাগমেন্ট কে ধারন করতে সাপোর্ট রাইব্রেরীতে প্রদেয় FragmentActivity একটা বিশেষ একটিভিটি। যদি নুন্যতম সিস্টেম ভার্সন এপিআই লেভেল ১১ বা এর চেয়ে বেশী হয় যাকে আপনি সাপোর্ট করছেন, তখন আপনি নিয়মিত Activity ব্যবহার করতে পারেন।

এখানে লেআউট ফাইলের একটা উদাহরন দেয়া হলো যা একটি একটিভিটিতে দুটি ফ্র্যাগমেন্ট যুক্ত করে যখন ডিভাইস স্ক্রিন “লার্জ” (বড়) হিসাবে বিবেচ্য হয় (ডিবেক্টরি নেম এ large কোয়ালিফায়ার দ্বারা নির্ধারিত)

res/layout-large/news_articles.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <fragment android:name="com.example.android.fragments.HeadlinesFragment"
        android:id="@+id/headlines_fragment"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

    <fragment android:name="com.example.android.fragments.ArticleFragment"
        android:id="@+id/article_fragment"
        android:layout_weight="2"
        android:layout_width="0dp"
        android:layout_height="match_parent" />

</LinearLayout>
```

টিপস: ভিন্ন ভিন্ন স্ক্রিনের জন্য লেআউট তৈরী করা সম্পর্কিত আরও তথ্য জানতে বিভিন্ন ধরনের স্ক্রিন সাইজ সাপোর্ট করা অধ্যায়টি পড়ুন।

এরপর আপনার একটিভিটিতে লেআউটটি প্রয়োগ করুন:

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;
```

```
public class MainActivity extends FragmentActivity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.news_articles);  
    }  
}
```

যদি আপনি v7 appcompat library ব্যবহার করে থাকেন, আপনার একটিভিটির পরিবর্তে actionBarActivity ব্যবহার করা উচিত, যা FragmentActivity এর সাব ক্লাস (আরও জানতে অফফরহম ঃযব অপঃরড়হ ইধৎ পড়ুন)।

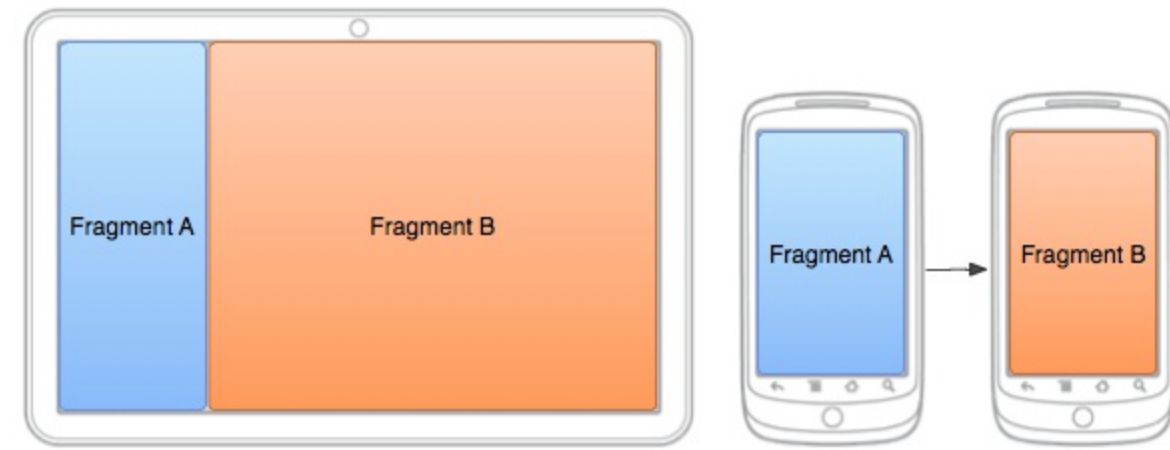
নোট: লেআউট এক্সএমএল ফাইলে ফ্রাগমেন্ট নির্ধারন করার মাধ্যমে যখন আপনি একটিভিটি লেআউটে ফ্রাগমেন্ট সংযোজন করবেন, রানটাইমে এই ফ্রাগমেন্ট কে বিযুক্ত করতে পারবেন না। আপনি যদি ইউজার ইন্টারেকশনের সময় আপনার ভিত্তেও এবং বাইরে অদল বদল করার পরিকল্পনা করে থাকেন, আপনাকে অবশ্যই একটিভিটিতে ফ্রাগমেন্ট যুক্ত করতে হবে যখন একটিভিটি প্রথম শুরু করে, পরবর্তী অনুশীলনীতে যেভাবে দেখানো হয়েছে সেভাবে।

একটি নমনীয় (ফ্লেক্সিবল) ইউজার ইন্টারফেস তৈরী করুন

(<http://developer.android.com/training/basics/fragments/fragment-ui.html>)

যখন আপনার অ্যাপলিকেশনকে স্ক্রিন সাইজের ব্যাপক বিস্তৃতিতে সাপোর্ট করতে তৈরী করা হবে, বিদ্যমান স্ক্রিন সাইজের উপর ভিত্তি করে ইউজার এক্সপেরিয়েন্স অপটিমাইজ করতে আপনি বিভিন্ন লেআউট কনফিগারেশনে আপনার ফ্রাগমেন্টকে পূর্ণব্যবহার করতে পারেন।

উদাহরণস্বরূপ, একটি হ্যান্ডসেট ডিভাইসে একটি সিঙ্গেল-পেন ইউজার ইন্টারফেসের জন্য একবারে শুধু একটা ফ্রাগমেন্ট প্রদর্শন করাটা উপযুক্ত হবে। একইভাবে আপনি একটি ট্যাবলেট যাতে আছে ইউজারকে আরও তথ্য প্রদর্শন করার একটি প্রশস্ত স্ক্রিন তাতে ফ্রাগমেন্টগুলোকে পাশাপাশি রাখতে চাইতে পারেন



ফিগার ১. দুইটা ফ্রাগমেন্ট ভিন্ন ভিন্ন স্ক্রিন সাইজে একই একটিভিটির জন্য ভিন্ন ভিন্ন কনফিগারেশন প্রদর্শন করছে। বড় স্ক্রিনে দুইটা ফ্রাগমেন্টই পাশাপাশি ফিট করেছে কিনুত একটি হ্যান্ডসেট ডিভাইসে একবারে শুধু একটা ফ্রাগমেন্ট ফিট করে সুতরাং ইউজার যখন নেভিগেট করে তখন একটা ফ্রাগমেন্টের বদলে আরেকটি ফ্রাগমেন্ট আসে।

FragmentManager ক্লাসটি মেথড সরবরাহ করে যা একটি ডাইনামিক এক্সপেরিয়েন্স তৈরী করার জন্য একটি একটিভিটিতে ফ্রাগমেন্ট সংযোজন, বিয়োজন এবং প্রতিস্থাপন করতে অনুমোদন করে।

রানটাইমে একটি একটিভিটিতে ফ্রাগমেন্ট সংযোজন (এ্যাড) করুন

যেভাবে < Fragment> এলিমেন্ট সহ পূর্ববর্তী অনুশীলনীতে (previous lesson) দেখানো হয়েছে, লেআউটের মধ্যে একটা একটিভিটির জন্য ফ্রাগমেন্ট নির্ধারণ করার বদলে আপনি একটিভিটি রানটাইমে একটিভিটিতে একটি ফ্রাগমেন্ট সংযোজন করে দিতে পারেন। এটা প্রয়োজনীয় যদি আপনি একটিভিটি অস্তিত্বশীল থাকাকালীন সময়ে ফ্রাগমেন্ট পরিবর্তন করার পরিকল্পনা করে থাকেন।

একটি ফ্রাগমেন্ট সংযোজন বা বিয়োজন করার মতো বিনিময় কর্ষ সম্পাদন করতে, একটি FragmentTransaction তৈরী করতে আপনাকে অবশ্যই FragmentManager ব্যবহার করতে হবে, যা সংযোজন, বিয়োজন, প্রতিস্থাপন এবং অন্যান্য ফ্রাগমেন্ট বিনিময় করতে এপিআই সরবারহ করে থাকে।

ঐদি আপনার একটিভিটি ফ্রাগমেন্টকে বিয়োজিত বা প্রতিস্থাপিত হতে অনুমোদন করে, আপনার উচিত একটিভিটির onCreate() মেথডের সময় একটিভিটিতে প্রাথমিক ফ্রাগমেন্ট সংযোজন করা।

ফ্রাগমেন্ট নিয়ে কাজ করার সময় একটি গুরুত্বপূর্ণ নিয়ম-বিশেষত ঐ সকল সকল যা রান টাইমে সংযোজন করা হয়েছে-ঐ নিয়মটি হচ্ছে যে ফ্রাগমেন্টের অবশ্যই লেআউটের মধ্যে একটা কনটেইনার View থাকতে হবে যার মধ্যে ফ্রাগমেন্টের লেআউট অবস্থান করে।

নিম্নোক্ত লেআউট পূর্ববর্তী অনুশীলনীতে দেখানো লেআউটের বিকল্প যা একবারে শুধু একটা ফ্রাগমেন্ট প্রদর্শন করে। একটা ফ্রাগমেন্টকে আর একটা ফ্রাগমেন্টে প্রতিস্থাপন করতে একটিভিটির লেআউট একটা খালি FrameLayout অন্তর্ভুক্ত করে যা ফ্রাগমেন্ট কনটেইনার হিসাবে কাজ করে।

দেখা যায় যে, পূর্ববর্তী অনুশীলনীর লেআউট ফাইলের মতো ফাইলনেমটি একই রকম, কিনুত লেআউট ডিরেক্টরীতে ষষ্ঠমব কোয়ালিফায়ার নেই, তাই ডিভাইস স্ক্রিন বড় হওয়ার চেয়ে যখন ছোট হয় তখন এই লেআউটটি ব্যবহৃত হয় কারন স্ক্রিন একই সাথে উভয় ফ্রাগমেন্টের সাথে ফিট করে না।

res/layout/news_articles.xml:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

আপনার একটিভিটির মধ্যে সাপোর্ট লাইব্রেরী এপিআই ব্যবহার করে একটি FragmentManager পেতে getSupportFragmentManager() কল করুন। তারপর FragmentTransaction তৈরী করতে beginTransaction() কল করুন এবং একটি ফ্রাগমেন্ট সংযোজন করতে add() কল করুন।

আপনি একই `FragmentManager` ব্যবহার করে একাধিকটির জন্য মাল্টিপল ফ্রাগমেন্ট ট্রানজেকশন সম্পাদন করতে পারেন। যখন আপনি এই পরিবর্তন করতে প্রস্তুত হবেন, আপনাকে অবশ্যই `commit()` কল করতে হবে।

উদাহরণস্বরূপ, পূর্ববর্তী লেআউটে কীভাবে একটি ফ্রাগমেন্ট সংযোজন করা যায় তা দেয়া হলো:

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);

        // Check that the activity is using the layout version with
        // the fragment_container FrameLayout
        if (findViewById(R.id.fragment_container) != null) {

            // However, if we're being restored from a previous state,
            // then we don't need to do anything and should return or else
            // we could end up with overlapping fragments.
            if (savedInstanceState != null) {
                return;
            }

            // Create a new Fragment to be placed in the activity layout
            HeadlinesFragment firstFragment = new HeadlinesFragment();

            // In case this activity was started with special instructions from an
            // Intent, pass the Intent's extras to the fragment as arguments
            firstFragment.setArguments(getIntent().getExtras());

            // Add the fragment to the 'fragment_container' FrameLayout
            getSupportFragmentManager().beginTransaction()
                .add(R.id.fragment_container, firstFragment).commit();
        }
    }
}
```

কারণ ফ্রাগমেন্ট রানটাইমে `FrameLayout` কনটেইনারে সংযুক্ত হয়েছে- একটি `< fragment>` এলিমেন্ট দিয়ে একটিভিটির লেআউটে এটা নির্ধারণ করার পরিবর্তে- একটিভিটিটি ফ্রাগমেন্টটিকে বিয়োজিত করতে পারে এবং ভিন্ন একটার সাথে এটাকে প্রতিস্থাপন করতে পারে।

একটা ফ্রাগমেন্টের সাথে আরেকটি ফ্রাগমেন্টের প্রতিস্থাপন (রিপ্লেস)

একটি ফ্রাগমেন্ট প্রতিস্থাপনের প্রক্রিয়া সংযোজন প্রক্রিয়ার মতই, কিন্ত `add()`এর পরিবর্তে `replace()` মেথড প্রয়োজন হয়।

মনে রাখবেন যে যখন আপনি ফ্রাগমেন্ট ট্রানজেকশন সম্পাদন করবেন, যেমন কোনটার বিয়োজন বা প্রতিস্থাপন, ইউজারকে পরিবর্তনটিকে "undo" করতে নেভিগেট করে পেছনে নিয়ে আসাটা অনুমোদন করতে এটা বরাবরই যথাযথ। ফ্রাগমেন্ট ট্রানজেকশনের মাধ্যমে ইউজারকে পেছন দিকে নেভিগেট করা অনুমোদন করতে, `FragmentManager` করার পূর্বে আপনাকে অবশ্যই `addToBackStack()` কল করতে হবে।

নোট: যখন আপনি একটি ফ্রাগমেন্ট প্রতিস্থাপন বা বিয়োজন করবেন এবং ব্যাক স্ট্যাকে ট্রানজেকশনটি সংযুক্ত করবেন, ফ্রাগমেন্টটি যা বিয়োজিত হয়েছে তা থেমে (স্টপ) যায় (ধ্বংস হয় না)। যদি ইউজার ফ্রাগমেন্টটা রিস্টোর করতে নেভিগেট করে পেছন দিকে আসে, এটা রিস্টার্ট করে। যদি আপনি ব্যাক স্ট্যাক এ ট্রানজেকশন সংযোজন না করে থাকেন, যখন ফ্রাগমেন্ট প্রতিস্থাপন বা বিয়োজন করা হয় তখন এটা ধ্বংস হয়ে যায়।

একটি ফ্রাগমেন্টের সাথে আরেকটির প্রতিস্থাপনের উদাহরণ:

```
// Create fragment and give it an argument specifying the article it should show
ArticleFragment newFragment = new ArticleFragment();
Bundle args = new Bundle();
args.putInt(ArticleFragment.ARG_POSITION, position);
newFragment.setArguments(args);

FragmentManager transaction = getSupportFragmentManager().beginTransaction();

// Replace whatever is in the fragment_container view with this fragment,
// and add the transaction to the back stack so the user can navigate back
transaction.replace(R.id.fragment_container, newFragment);
transaction.addToBackStack(null);

// Commit the transaction
transaction.commit();
```

`addToBackStack()` মেথডটি একটি ঐচ্ছিক স্ট্রিং প্যারামিটার নেয় যা ট্রানজেকশনের জন্য একটা ইউনিক নাম সুনির্দিষ্ট করে। নামটির কোন প্রয়োজন নেই যদি না আপনি `FragmentManager.BackStackEntry` এপিআই ব্যবহার করে উন্নত ফ্রাগমেন্ট অপারেশন সম্পাদন করার পরিকল্পনা করে থাকেন।

অন্য ফ্রাগমেন্টের সাথে যোগাযোগ

(<http://developer.android.com/training/basics/fragments/communicating.html>)

ফ্রাগমেন্ট ইউজার ইন্টারফেস কম্পোনেন্ট ব্যবহারের জন্য, আপনাকে সম্পূর্ণ স্বয়ংসম্পূর্ণভাবে প্রতিটা মডিউলার কম্পোনেন্ট তৈরী করা উচিত যা এর নিজস্ব লেআউট এবং আচরন নির্ধারণ করে। পূর্বে আপনি যদি এই পূণব্যবহারযোগ্য ফ্রাগমেন্টগুলোকে নিরূপন করে থাকেন, আপনি তাদেরকে একটি একটিভিটির সাথে একাত্ম করে দিতে পারেন এবং সার্বিক মিশ্র ইউজার ইন্টারফেসকে অনুধাবন করতে অ্যাপলিকেশন লজিকের সাথে এদের সংযুক্ত করে দিন।

মাঝে মাঝে আপনি চাবেন একটি ফ্রাগমেন্ট আরেকটি ফ্রাগমেন্টের সাথে যোগাযোগ করুক, উদাহরণস্বরূপ একটা ইউজার ইন্টারফেসের উপর ভিত্তি করে কন্টেন্ট পরিবর্তন করা। সকল ফ্রাগমেন্ট থেকে ফ্রাগমেন্টের যোগাযোগটা সংযুক্ত একটিভিটির মাধ্যমে ঘটে থাকে। দুইটা ফ্রাগমেন্টের সরাসরি যোগাযোগ করা উচিত নয়।

একটি ইন্টারফেস নির্ধারন করা

একটি ফ্রাগমেন্টকে এর একটিভিটি পর্যন্ত যোগাযোগ করাটা অনুমোদন করতে আপনি ফ্রাগমেন্ট ক্লাসে একটা ইন্টারপেস নির্ধারণ করতে পারেন এবং একটিবিটির মধ্যে এর বাস্তবায়ন করুন। ফ্রাগমেন্টটি এর `onAttach()` লাইফসাইকেল মেথড সময়ে ইন্টারফেস বাস্তবায়নকে ক্যাপচার কওে এবং তারপর একটিভিটির সাথে যোগাযোগ স্থাপনের জন্য ইন্টারফেস মেথডকে কল করে।

এখানে একটিভিটি কমিউনিকেশনে ফ্রাগমেন্টের একটি উদাহরণ দেয়া হলো:

```
public class HeadlinesFragment extends ListFragment {
    OnHeadlineSelectedListener mCallback;

    // Container Activity must implement this interface
    public interface OnHeadlineSelectedListener {
        public void onArticleSelected(int position);
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);

        // This makes sure that the container activity has implemented
        // the callback interface. If not, it throws an exception
        try {
            mCallback = (OnHeadlineSelectedListener) activity;
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString()
                + " must implement OnHeadlineSelectedListener");
        }

        ...
    }
}
```

এখন ফ্রাগমেন্ট `OnHeadlineSelectedListener` ইন্টারফেসের `mCallback` ইনসটান্সে ব্যবহার করে `onArticleSelected()` কে কল করার মাধ্যমে একটিভিটিতে ম্যাসেজ/বার্তা পাঠাতে পারে।

উদাহরণস্বরূপ, নিচেরোক্ত ফ্রাগমেন্টের মধ্যকার মেথডটিকে করা হয় যখন ইউজার লিস্ট আইটেমে ক্লিক করে। ফ্রাগমেন্টটি প্যারেন্ট একটিভিটিতে ইভেন্টটি পৌছে দিতে কলব্যাক ইন্টারফেস ব্যবহার করে।

```
@Override
public void onListItemClick(ListView l, View v, int position, long id) {
    // Send the event to the host activity
    mCallback.onArticleSelected(position);
}
```

ইন্টারফেস বাস্তবায়ন করা

ফ্রাগমেন্ট থেকে ইভেন্ট কলব্যাক গ্রহণ করতে, একটিভিটি যা একে আশ্রয় দেয় তার উচিত ফ্রাগমেন্ট ক্লাসে নির্ধারিত ইন্টারফেস অবশ্যই বাস্তবায়ন করা।

উদাহরণস্বরূপ, নিম্নোক্ত একটিভিটি উপরোক্ত উদাহরণ থেকে ইন্টারফেস বাস্তবায়ন করছে।

```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article
    }
}
```

ফ্রাগমেন্টে বার্তা (মেসেজ) পৌঁছে দিন

হোস্ট একটিভিটি `findFragmentById()` দিয়ে `Fragment` কে ক্যাপচার করার মাধ্যমে ফ্রাগমেন্টে মেসেজ পৌঁছে দিতে পারে, তারপর সরাসরি ফ্রাগমেন্টের পাবলিক মেথড কে কল করে।

উদাহরণস্বরূপ, চিন্তা করুন যে একটিভিটি যেটা উপরে দেখানো হয়েছে আরেকটি ফ্রাগমেন্টকে ধারণ করতে পারে যা উপরোক্ত কলব্যাক মেথডের মধ্যে ফিরে আসা ডাটা দ্বারা নির্ধারিত আইটেমকে প্রদর্শন করতো। এক্ষেত্রে একটিভিটি কলব্যাক মেথডে রিসিভ করা তথ্য অন্য ফ্রাগমেন্টে পাস করে দিতে পারে যা আইটেমকে প্রদর্শিত করবে।

```
public static class MainActivity extends Activity
    implements HeadlinesFragment.OnHeadlineSelectedListener{
    ...

    public void onArticleSelected(int position) {
        // The user selected the headline of an article from the HeadlinesFragment
        // Do something here to display that article

        ArticleFragment articleFrag = (ArticleFragment)
            getSupportFragmentManager().findFragmentById(R.id.article_fragment);

        if (articleFrag != null) {
            // If article frag is available, we're in two-pane layout...

            // Call a method in the ArticleFragment to update its content
            articleFrag.updateArticleView(position);
        } else {
            // Otherwise, we're in the one-pane layout and must swap frags...

            // Create fragment and give it an argument for the selected article
            ArticleFragment newFragment = new ArticleFragment();
            Bundle args = new Bundle();
            args.putInt(ArticleFragment.ARG_POSITION, position);
            newFragment.setArguments(args);

            FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();

            // Replace whatever is in the fragment_container view with this fragment,
            // and add the transaction to the back stack so the user can navigate back
            transaction.replace(R.id.fragment_container, newFragment);
            transaction.addToBackStack(null);

            // Commit the transaction
            transaction.commit();
        }
    }
}
```

ডাটা সেভ করা

(<http://developer.android.com/training/basics/data-storage/index.html>)

অধিকাংশ অ্যান্ড্রয়েড অ্যাপের ডাটা সেভ করার দরকার হয়, এমনকি শুধুমাত্র onPause()এর সময় অ্যাপ স্টেট সম্পর্কিত তথ্য সেভ করা যাতে ইউজারের অগ্রগতি হারিয়ে না যায়। অধিকাংশ নন ট্রাইভ্যাল অ্যাপেরও ইউজার সেটিং সেভ করতে হয়, এবং কিছু অ্যাপকে অবশ্যই ফাইলে এবং ডাটাবেজে অনেক তথ্য ব্যবস্থাপনা করতে হয়। এই ক্লাস আপনাকে অ্যান্ড্রয়েডের প্রিন্সিপাল ডাটা স্টোরেজ অপশন এর সাথে পরিচয় করিয়ে দিবে, যার মধ্যে রয়েছে:

- একটি শেয়ারড প্রিফারেন্স ফাইলে সিম্পল ডাটা টাইপ এর কি-ভ্যালু পেয়ার সেভ করা
- অ্যান্ড্রয়েডের ফাইল সিস্টেমে আরবিট্রারি ফাইল সেভ
- SQLite কর্তৃক পরিচালিত ডাটাবেজ ব্যবহার

এই অধ্যায়ের অনুশীলনীসমূহ

কি-ভ্যালু সেট সেভ

কি-ভ্যালু পেয়ারে ছোট আকারের তথ্য স্টোর করার জন্য শেয়ারড প্রিফারেন্স ফাইল ব্যবহার শিখুন।

ফাইল সেভ করা

বেসিক ফাইল সেভ করা শিখুন, যেমন লম্বা ক্রমের ডাটা স্টোর করা যা সাধারনত ক্রমানুসারে পড়তে হয়।

SQL ডাটাবেজে ডাটা সেভ করা

কার্ঠামোবদ্ধ ডাটা পড়তে এবং লিখতে SQLite ডাটাবেজ ব্যবহার করতে শিখুন।

কি-ভ্যালু সেট সেভ করা

(<http://developer.android.com/training/basics/data-storage/shared-preferences.html>)

যদি আপনার কি-ভ্যালুর অপেক্ষাকৃত ছোট সংগ্রহ থাকে যা আপনি সেভ করতে চান, আপনার SharedPreferences এপিআই ব্যবহার করা উচিত। একটি SharedPreferences অবজেক্ট ফাইল কনটেইনিং কি-ভ্যালু পেয়ারস কে নির্দেশ করে এবং তাদের পড়তে এবং লিখতে সরল মেথড সরবরাহ করে। প্রতিটা SharedPreferences ফাইল ফ্রেমওয়ার্ক কর্তৃক পরিচালিত হয় এবং এটা প্রাইভেট বা শেয়ারড হতে পারে।

এই ক্লাস আপনাকে দেখাবে সহজ সরল ভ্যালু স্টোর এবং তা উদ্ধার করতে কীভাবে SharedPreferences এপিআই ব্যবহার করতে হয়।

নোট: SharedPreferences এপিআই শুধুমাত্র কি-ভ্যালু পেয়ার পড়া এবং লেখার জন্য এবং আপনি Preferences এর সাথে এটাকে মিলিয়ে ফেলে দ্বিধাশ্রিত হবেন না, কারণ Preferences আপনার অ্যাপ সেটিং এর জন্য ইউজার ইন্টারফেস তৈরীতে সহায়তা করে (যদিও তারা অ্যাপ সেটিং সেভ করতে তাদের বাস্তবায়ন হিসাবে SharedPreferences ব্যবহার করে)। Preferences এপিআই ব্যবহার করা সম্পর্কিত তথ্যের জন্য Settings (<http://developer.android.com/guide/topics/ui/settings.html>) গাইড দেখুন।

একটি Shared Preferences এ হ্যান্ডেল লাভ করা

দুইটার মধ্যে একটা মেথড কল করার মাধ্যমে আপনি একটি নতুন শেয়ারড প্রিফারেন্স তৈরী করতে পারেন বা বিদ্যমান একটিতে প্রবেশযোগ্যতা তৈরী করতে পারেন:

- `getSharePreferences()` ব্যবহার করুন, আপনার যদি নাম দ্বারা চিহ্নিত মাল্টিপল শেয়ারড প্রিফারেন্স দরকার হয়, যা আপনি প্রথম প্যারামিটার দিয়ে সুনির্দিষ্ট করেন। আপনি এটা আপনার অ্যাপের যে কোন Context থেকে কল করতে পারেন।
- `getPreferences()`-একটা একটিভিটি থেকে এটা ব্যবহার করুন যদি আপনার একটিভিটির জন্য শুধুমাত্র একটি শেয়ারড প্রিফারেন্স ফাইল ব্যবহারের প্রয়োজন হয়। কারন এটা একটা ডিফল্ট শেয়ারড প্রিফারেন্স উদ্ধার করে যা একটিভিটির সাথে থাকে, আপনার একটা নাম প্রদান করার প্রয়োজন নেই।

উদাহরণস্বরূপ, নিম্নোক্ত কোড একটি Fragment এর মধ্যে সংঘটিত হয়েছে। এটা শেয়ারড প্রিফারেন্স ফাইল অনুপ্রবেশ করায় যা রিসোর্স স্ট্রিং `R.string.preference_file_key` কর্তৃক চিহ্নিত এবং প্রাইভেট মোড ব্যবহার করে এটা ওপেন করে যাতে ফাইলটি শুধুমাত্র আপনার অ্যাপ কর্তৃক প্রবেশযোগ্য হয়।

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
    getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

যখন আপনার শেয়ারড প্রিফারেন্স ফাইলের চিহ্নিতকরন করা হয়, একটা স্বতন্ত্র এবং আপনার অ্যাপে সহজে চিহ্নিত করা যায় আপনার উচিত তেমন একটা নাম ব্যবহার করা, যেমন `"com.example.myapp.PREFERENCE_FILE_KEY"`

অপরদিকে যদি আপনার একটিভিটির জন্য শুধু একটি শেয়ারড প্রিফারেন্স ফাইল দরকার হয়, আপনি `getPreferences()` মেথড ব্যবহার করতে পারেন:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
```

সতর্কতা: আপনি যদি `MODE_WORLD_READABLE` বা `MODE_WORLD_WRITEABLE` দিয়ে একটি শেয়ারড প্রিফারেন্স ফাইল তৈরী করেন, তাহলে অন্য যে কোন অ্যাপ যা জানে যে ফাইল আইডেনটিফায়ার আপনার ডাটাতে অনুপ্রবেশ করতে পারবে।

শেয়ারড প্রিফারেন্স লেখা (রাইট)

একটি শেয়ারড প্রিফারেন্স ফাইলে লিখতে, আপনার SharedPreferences উপরে edit()কে কল করার মাধ্যমে একটি SharedPreferences.Editor তৈরী করতে পারেন।

কি এবং ভ্যালু টা পাস করুন যা আপনি putInt()এবং putString()এর মতো মেথড দিয়ে লিখতে চান। তারপর পরিবর্তনটাকে সেভ করতে commit() কল করুন। উদাহরণস্বরূপ:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);  
SharedPreferences.Editor editor = sharedPref.edit();  
editor.putInt(getString(R.string.saved_high_score), newHighScore);  
editor.commit();
```

শেয়ারড প্রিফারেন্স থেকে পাঠ করা

একটি শেয়ারড প্রিফারেন্স ফাইল থেকে ভ্যালু উদ্ধার করতে, `getInt()` এবং `getString()` এর মতো মেথড কল করুন, যে ভ্যালুটি আপনি চান তার জন্য কি সরবরাহ করুন এবং যদি কি বিদ্যমান না থাকে ফিরে আসতে ঐচ্ছিকভাবে একটি ডিফল্ট ভ্যালু। উদাহরণস্বরূপ:

```
SharedPreferences sharedPref = getActivity().getPreferences(Context.MODE_PRIVATE);
int defaultValue = getResources().getInteger(R.string.saved_high_score_default);
long highScore = sharedPref.getInt(getString(R.string.saved_high_score), defaultValue);
```

ফাইল সেভ করা

(<http://developer.android.com/training/basics/data-storage/files.html>)

অ্যান্ড্রয়েড একটা ফাইল সিস্টেম ব্যবহার করে যা অন্যান্য প্লাটফর্মের ডিস্ক বেজড ফাইল সিস্টেমের মতো। এই অনুশীলনী আলোচনা করবে File এপিআই দিয়ে লিখতে বা পড়তে কীভাবে অ্যান্ড্রয়েড ফাইল সিস্টেম দিয়ে কাজ করতে হয়।

একটি File অবজেক্ট কোন রকম স্কিপ করা ছাড়াই শুরু থেকে শেষ পর্যন্ত ক্রমানুসারে ব্যাপক সংখক ডাটা লিখতে বা পড়তে পারার জন্য যথাপোযুক্ত। উদাহরণস্বরূপ, এটা ইমেজ ফাইল বা একটি নেটওয়ার্কের ওপরে পরিবর্তিত কোন কিছুর জন্য ভালো।

এই অনুশীলনী দেখাবে কীভাবে আপনার অ্যান্ড্রয়েডে বেসিক ফাইল সম্পর্কিত কাজ সম্পাদন করা হয়। এই অনুশীলনী অনুমান করে যে আপনি বেসিক লিনাক্স ফাইল সিস্টেম এবং java.io এর মধ্যে স্ট্যান্ডার্ড ফাইল ইনপুট/আউটপুট এপিআই এর সাথে পরিচিত।

ইন্টারনাল বা এক্সটার্নাল স্টোরেজ পছন্দ করা

সকল অ্যান্ড্রয়েড ডিভাইসের দুইটা ফাইল স্টোরেজ এলাকা আছে: "internal" এবং "external" স্টোরেজ। এই নাম গুলো অ্যান্ড্রয়েডের প্রথমাদিককার দিন গুলো থেকেই চলে আসছে, যখন অধিকাংশ ডিভাইস বিল্ট-ইন নন-ভল্যাটাইল মেমরী (ইন্টারনাল স্টোরেজ) দিত, এর সাথে রিমুভ্যাবল স্টোরেজ মিডিয়াম যেমন একটি মাইক্রো এসডি কার্ড (এক্সটার্নাল স্টোরেজ) দিত। কিছু ডিভাইস "internal" এবং "external" পার্টিশান এই স্থায়ী স্টোরেজ স্পেসে বিভক্ত ছিল, সুতরাং একটি রিমুভ্যাবল স্টোরেজ মিডিয়াম থাকা সত্ত্বেও এখানে সবসময় দুইটা স্টোরেজ স্পেস থাকে এবং এপিআই আচরন একই থাকে, এক্সটার্নাল স্টোরেজ রিমুভ্যাবল হোক বা না হোক। নিম্নোক্ত তালিকাগুলো প্রতিটা স্টোরেজ স্পেস সম্পর্কিত ফ্যাক্টস গুলোর সংক্ষিপ্ত বর্ণনা।

ইন্টারনাল স্টোরেজ:

- এটা সবসময় পাওয়া যায়
- ফাইলটা এখানে সেভ হয় যা শুধুমাত্র বাই ডিফল্ট আপনার অ্যাপ কর্তৃক প্রবেশযোগ্য
- যখন ইউজার আপনার অ্যাপ আনইনস্টল করে, সিস্টেমটি ইন্টারনাল স্টোরেজ থেকে আপনার অ্যাপের সমস্ত ফাইল অপসারণ করে

যখন আপনি নিশ্চিত হতে চাইবেন যে ইউজার বা অন্য অ্যাপ আপনার ফাইলে প্রবেশ করতে পারবে না সেক্ষেত্রে ইন্টারনাল স্টোরেজই উত্তম।

এক্সটার্নাল স্টোরেজ:

- এটা সবসময় পাওয়া যায় না, কারন ইউজার ইউএসবি স্টোরেজ হিসাবে এক্সটার্নাল স্টোরেজকে প্রসারিত করতে পারে এবং কিছু ক্ষেত্রে এটাকে ডিভাইস থেকে রিমুভ করে ফেলতে পারে।
- এটা ওয়ার্ল্ড-রিডেবল, তাই ফাইলগুলো এখানে সেভ হয় আপনার নিয়ন্ত্রনের বাইরে থেকে পঠিত হতে পারে
- যখন ইউজার আপনার অ্যাপ আনইনস্টল করে, সিস্টেমটি আপনার অ্যাপের সমস্ত ফাইল অপসারণ করে শুধুমাত্র তখনই আপনি যখন `getExternalFilesDir()` থেকে ডিরেক্টরীতে তাদের সেভ করে থাকেন।

এক্সটার্নাল স্টোরেজ হচ্ছে সেই সকল ফাইলের জন্য সবচেয়ে ভালো জায়গা যা প্রবেশের ক্ষেত্রে বিধিনিষেধ দাবী করে না এবং সেই সকল ফাইলের জন্য যা আপনি অন্য অ্যাপের সাথে শেয়ার করতে চান অথবা ইউজারকে একটি কম্পিউটার দিয়ে এখানে প্রবেশ করাটা অনুমোদন করেন।

টিপ: যদিও অ্যাপ ইন্টারনাল স্টোরেজে বাই ডিফল্ট ইনস্টলড হয়, আপনি আপনার মনিফেস্টে `android:installLocation` এট্রিবিউটটি সুনির্দিষ্ট করতে পারেন যাতে আপনার অ্যাপ এক্সটার্নাল স্টোরেজে ইনস্টলড হতে পারে। ইউজার এই অপশনকে সাদরে গ্রহণ করে যখন এপিকে সাইজ অনেক বড় হয় এবং যাদের একটা এক্সটার্নাল স্টোরেজ থাকে যা ইন্টারনাল স্টোরেজ থেকে বড় হয়।

আরও	তথ্যের	জন্য	App	Install	Location	(link:
http://developer.android.com/guide/topics/data/install-location.html) দেখুন।						

এক্সটার্নাল স্টোরেজের জন্য অনুমতি গ্রহণ

এক্সটার্নাল স্টোরেজে লিখতে, আপনার মেনিফেস্টের মধ্যে WRITE_EXTERNAL_STORAGE পারমিশন কে অবশ্যই আবেদন করতে হয়:

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
```

সতর্কতা: বর্তমানে সকল অ্যাপের কোন বিশেষ অনুমোদন ছাড়াই এক্সটার্নাল স্টোরেজ রিড করার সামর্থ আছে। যাহোক, এটা ভবিষ্যতে পরিবর্তন হবে। যদি আপনার অ্যাপের এক্সটার্নাল স্টোরেজ রিড করার প্রয়োজন হয় (কিন্তু এটাতে লেখা নয়), তখন আপনার READ_EXTERNAL_STORAGE অনুমোদনকে/পারমিশন ডিক্লেয়ার করা প্রয়োজন। আপনার অ্যাপ প্রত্যাশা অনুযায়ী কাজ করছে এটা নিশ্চিত করতে আপনার উচিত এই পারমিশন ডিক্লেয়ার করা, পরিবর্তনটা ফলাফল গ্রহণ করার পূর্বেই।

```
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

যাহোক, যদি আপনার অ্যাপ WRITE_EXTERNAL_STORAGE পারমিশন ব্যবহার করে, তখন এটাতে সম্পূর্ণভাবে এক্সটার্নাল স্টোরেজ রিড করারও পারমিশন থাকে।

ইন্টারনাল স্টোরেজে ফাইল সেভ করতে আপনার কোন পারমিশন প্রয়োজন নেই। আপনার অ্যাপের সবসময়ই এর ইন্টারনাল ডিরেক্টরির মধ্যকার ফাইল লেখা বা পড়ার পারমিশন থাকে।

ইন্টারনাল স্টোরেজে ফাইল সেভ করা

যখন ইন্টারনাল স্টোরেজে একটি ফাইল সেভ করবেন, দুইটা মেথডের একটি কল করে আপনি একটি File হিসাবে একটি যথাযথ ডিরেক্টরী পেতে পারেন:

getFilesDir()

আপনার অ্যাপের জন্য একটা ইন্টারনাল ডিরেক্টরীর প্রতিনিধিত্বকারী একটা File ফিরিয়ে আনে

getCacheDir()

আপনার অ্যাপের অস্থায়ী পধপষব ফাইলের জন্য একটা ইন্টারনাল ডিরেক্টরীর প্রতিনিধিত্বকারী একটা ফাইল ফিরিয়ে আনে। যখন আর প্রয়োজন হবে না তখন প্রতিটা ফাইল ডিলিট করাটা নিশ্চিত করুন এবং যে কোন নির্দিষ্ট সময়ে যে পরিমান মেমরী আপনি ব্যবহার করেন তার সাথে সামঞ্জস্যপূর্ণ সাইজ লিমিট বাস্তবায়ন করুন, যেমন ১ মেগাবাইট (MB)। সিস্টেমটি যদি খুব কম স্টোরেজ এ রান করা শুরু করে, এটা কোন পূর্ব সতর্কতা ছাড়াই আপনার পধপষব ফাইল ডিলিট করে দিতে পারে।

এই ডিরেক্টরীর যে কোন একটাতে একটা নতুন ফাইল তৈরী করতে, উপরোক্ত মেথড যা আপনার ইন্টারনাল স্টোরেজ ডিরেক্টরী কে সুনির্দিষ্ট করে সেটা কর্তৃক প্রদত্ত File পাস করতে আপনি File() কনস্ট্রাকটর ব্যবহার করতে পারেন। উদাহরণস্বরূপ:

```
File file = new File(context.getFilesDir()  
filename);
```

অপরপক্ষে, আপনি একটি FileOutputStream পেতে openFileOutput() কল করতে পারেন যা আপনার ইন্টারনাল ডিরেক্টরীর একটি ফাইলে লিখে। উদাহরণস্বরূপ, এখানে দেয়া হলো কীভাবে কিছু টেক্সট ফাইলে লিখতে হয়:

```
String filename = "myfile";  
String string = "Hello world!";  
FileOutputStream outputStream;  
  
try {  
    outputStream = openFileOutput(filename, Context.MODE_PRIVATE);  
    outputStream.write(string.getBytes());  
    outputStream.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

অথবা যদি আপনার কিছু ফাইল গুপ্ত অবস্থায় সঞ্চয় করার প্রয়োজন হয়, পরিবর্তে আপনার createTempFile() ব্যবহার করা উচিত। উদাহরণস্বরূপ, নিম্নোক্ত মেথড URL থেকে ফাইল নামকে পৃথক করে এবং আপনার অ্যাপের ইন্টারনাল ক্যাশে ডিরেক্টরীতে ওই নামে একটি ফাইল তৈরী

করে:

```
public File getTempFile(Context context, String url) {
    File file;
    try {
        String fileName = Uri.parse(url).getLastPathSegment();
        file = File.createTempFile(fileName, null, context.getCacheDir());
    } catch (IOException e) {
        // Error while creating file
    }
    return file;
}
```

নোট: আপনার অ্যাপের ইন্টারনাল স্টোরেজ ডিরেক্টরী অ্যান্ড্রয়েড ফাইল সিস্টেমের একটি বিশেষ লোকেশনে আপনার অ্যাপের প্যাকেজ নাম নির্দিষ্ট করে। কৌশলগতভাবে, অন্য অ্যাপ আপনার ইন্টারনাল ফাইল পড়তে পারে যদি আপনি ফাইল মোডকে রিডেবল হতে সেট করেন। কিন্ত, অন্য অ্যাপকেও আপনার অ্যাপ প্যাকেজ নেম এবং ফাইল নেম জানতে হবে। অন্য অ্যাপ আপনার ইন্টারনাল ডিরেক্টরী ব্রাউস করতে পারবে না এবং লেখার বা পড়ার প্রবেশাধিকার থাকবে না যদি না আপনি পরিক্ষারভাবে ফাইলটি রিডেবল বা রাইটেবল হতে সেট না করেন। সুতরাং যতক্ষণ আপনি ইন্টারনাল স্টোরেজের উপর আপনার ফাইলের জন্য MODE PRIVATE ব্যবহার করবেন, তারা অন্য অ্যাপের জন্য প্রবেশযোগ্য হবে না।

এক্সটার্নাল স্টোরেজে ফাইল সেভ করা

কারণ এক্সটার্নাল স্টোরেজ সহজপ্রাপ্য নাও হতে পারে-যেমন যখন ইউজার একটা পিসিতে স্টোরেজ প্রসারিত করে অথবা এসডি কার্ড যা এক্সটার্নাল স্টোরেজ সরবরাহ করে তা রিমুভ/অপসারণ করে-আপনার সবসময় যাচাই করা উচিত যে এটাতে প্রবেশ করার পূর্বে ভলিউমটি বিদ্যমান থাকে। আপনি `getExternalStorageState()` কল করে এক্সটার্নাল স্টোরেজ অবস্থা সম্পর্কে অনুসন্ধান করতে পারেন। যদি ফিরতি স্টেট `MEDIA_MOUNTED` এর মতো হয় তাহলে আপনি আপনার ফাইল লিখতে বা পড়তে পারেন। উদাহরণস্বরূপ, নিম্নোক্ত মেথড স্টোরেজ এর বিদ্যমানতা নির্ধারনে গুরুত্বপূর্ণ:

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

যদিও এক্সটার্নাল স্টোরেজ ইউজার এবং অন্য অ্যাপ কর্তৃক পরিবর্তনযোগ্য, সেখানে দুই ধরনের ফাইল আছে যা আপনি এখানে সেভ করতে পারেন:

পাবলিক ফাইল

ফাইল যা অন্য অ্যাপে এবং ইউজারের জন্য মুক্তভাবে সহজলভ্য হওয়া উচিত। যখন ইউজার আপনার অ্যাপ আনইনস্টল করবে, এই সকল ফাইল ইউজারের কাছে থাকা উচিত।

উদাহরণ হিসাবে, আপনার অ্যাপ কর্তৃক ক্যাপচার করা ফটো অথবা অন্যান্য ডাউনলোড করা ফাইল।

প্রাইভেট ফাইল

ফাইল যা আইনগতভাবে আপনার অ্যাপের সাথে থাকা উচিত এবং যখন ইউজার আপনার অ্যাপ আনইনস্টল করে তখন এটা ডিলিট হয়ে যাওয়া উচিত। যদিও এই সকল ফাইল কৌশলগতভাবে ইউজার বা অন্য অ্যাপ কর্তৃক প্রবেশগম্য কারণ তারা এক্সটার্নাল স্টোরেজ এ আছে, এই ফাইল যৌক্তিকভাবে আপনার অ্যাপের বাইরে কোন ইউজারকে ভ্যালু প্রদান করে না। যখন ইউজার আপনার অ্যাপ আনইনস্টল করে, সিস্টেমটি আপনার অ্যাপের এক্সটার্নাল প্রাইভেট ডিরেক্টরীর

সকল ফাইল ডিলিট করে দেয়।

উদাহরণস্বরূপ, আপনার অ্যাপ কর্তৃক ডাউনলোড করা অতিরিক্ত রিসোর্স অথবা অস্থায়ী মিডিয়া ফাইল।

যদি আপনি পাবলিক ফাইল এক্সটার্নাল স্টোরেজে সেভ করতে চান, এক্সটার্নাল স্টোরেজে যথাপোযুক্ত ডিরেক্টরির প্রতিনিধিত্বকারী একটি File পেতে `getExternalStoragePublicDirectory()` মেথড ব্যবহার করুন। এই মেথড যে ধরনের ফাইল আপনি সেভ করতে চান তা সুনির্দিষ্ট করে একটা আলোচনা নিয়ে আসে যাতে তারা অন্যান্য পাবলিক ফাইলের সাথে যৌক্তিকভাবে সংগঠিত হতে পারে, যেমন, DIRECTORY MUSIC বা DIRECTORY PICTURES। উদাহরণস্বরূপ:

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

আপনি যদি ফাইল সেভ করতে চান যা আপনার অ্যাপের কাছে প্রাইভেট, আপনি `getExternalFilesDir()` কল করে যথাযথ ডিরেক্টরী পেতে পারেন এবং ডিরেক্টরীর ধরন যা আপনি পছন্দ করেন তা নির্দেশ করে এটা একটা নেমে পাস করে দিন। এই এই উপায়ে তৈরী করে প্রতিটা ডিরেক্টরী প্যারেন্ট ডিরেক্টরীতে যুক্ত হয় যা আপনার অ্যাপের এক্সটার্নাল স্টোরেজের সকল ফাইলকে একত্রিত করে রাখে যা সিস্টেম ডিলিট করে দেয় যখন ইউজার আপনার অ্যাপ আনইনস্টল করে দেয়।

উদাহরণস্বরূপ, এখানে একটা মেথড দেয়া আছে যা আপনি একটি একক ফটো এলবামের জন্য একটি ডিরেক্টরী তৈরীতে ব্যবহার করতে পারেন:

```
public File getAlbumStorageDir(Context context, String albumName) {
    // Get the directory for the app's private pictures directory.
    File file = new File(context.getExternalFilesDir(
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

যদি পূর্ব নির্ধারিত সাব ডিরেক্টরীর কোন নামই আপনার ফাইলের সাথে না যায়, পরীবার্তে আপনি `getExternalFilesDir()` কল করতে পারেন এবং ইঁষষ পাস করতে পারেন। এটা এক্সটার্নাল স্টোরেজে আপনার অ্যাপের প্রাইভেট ডিরেক্টরীর জন্য রুট ডিরেক্টরী ফেরত নিয়ে আসে।

যদি মনে রাখেন যে `getExternalFilesDir()` একটি ডিরেক্টরীর মধ্যে একটি ডিরেক্টরী তৈরী করে, যা ডিলিট হয় যখন ইউজার আপনার অ্যাপ আনইনস্টল করে। যদি ফাইলটি যা আপনি সেভ করছেন তা ইউজার কর্তৃক আপনার অ্যাপ আনইনস্টল করার পরও বিদ্যমান থাকতে হয়-যেমন আপনার

অ্যাপ হচ্ছে একটা ক্যামেরা এবং ইউজার ফটোটা রাখতে চায়- পারবর্তে আপনার `getExternalStoragePublicDirectory()` ব্যবহার করা উচিত।

যদি কোন কারনে আপনি ফাইলের জন্য `getExternalStoragePublicDirectory()` ব্যবহার করেন যা শেয়ারড বা ফাইলের জন্য `getExternalFilesDir()` যা আপনার অ্যাপের কাছে প্রাইভেট, এটা গুরুত্বপূর্ণ যে আপনি এপিআই কনসটেন্ট কর্তৃক প্রদত্ত DIRECTORY PICTURE এর মতো ডিরেক্টরী নেম ব্যবহার করছেন। এই ডিরেক্টরী নেম নিশ্চিত করে যে ফাইলটি সিস্টেম কর্তৃক যথাযথভাবে যত্ন পাবে, উদাহরণস্বরূপ, DIRECTORY RINGTONES সেভ হওয়া ফাইল সিস্টেম মিডিয়া স্ক্যানার কর্তৃক মিউজিকের পরিবর্তে রিংটোন হিসাবে শ্রেণীবদ্ধ হয়।

ফ্রি স্পেস অনুসন্ধান

যদি আপনি আগে থেকে জানেন কি পরিমান ডাটা আপনি সেভ করছেন, `getFreeSpace()` বা `getTotalSspace()` কল করে একটি `IOException` না ঘটিয়েই আপনি খুজে বের করতে পারেন যে কোথায় পর্যাপ্ত স্পেস আছে। এই মেথডগুলো বিদ্যমান প্রাপ্তিসাধ্য স্পেস এবং স্টোরেজ ভলিউমের সর্বমোট স্পেস আলাদাভাবে সরবরাহ করে। এই তথ্য নির্দিষ্ট সীমার বাইরে স্টোরেজ ভলিউম পরিপূর্ণ করা পরিহার করতে উপকারী।

যাহোক সিস্টেম নিশ্চয়তা দেয় না যে আপনি `getFreeSpace()` কর্তৃক নির্দেশিত যত খুশি তত বাইটস লিখতে পারেন। আপনি যে সাইজের ডাটা সেভ করতে চান তার চাইতে কয়েক মেগাবাইট বেশী যদি নাম্বারটি ফেরত দেয়, বা ফাইল সিস্টেম ৯০% এর চেয়ে কম পূর্ণ হয়, তাহলে এটা এগিয়ে নেয়ার জন্য নিরাপদ। অন্যথায় আপনার সম্ভবত স্টোরেজে লেখা উচিত নয়।

নোট: আপনার ফাইল সেভ করার পূর্বে বিদ্যমান স্পেসের পরিমান চেক করা আপনার জন্য প্রয়োজনীয় নয়। আপনি পরিবর্তে সঙ্গে সঙ্গে তা লিখতে চেষ্টা করতে পারেন, তারপর একটা `IOException` ক্যাচ করুন, যদি একটা ঘটে থাকে। আপনার হয়তো এটা করার দরকার হতে পারে যদি আপনি সঠিকভাবে না জানেন আপনার কি পরিমান স্পেসের প্রয়োজন। উদাহরণস্বরূপ, PNG ইমেজ JPEG তে কনভার্ট করার মাধ্যমে এটা সেভ করার পূর্বে আপনি যদি ফাইলের এনকোডিং পরিবর্তন করেন, আপনি পূর্বেই ফাইলের সাইজ জানতে পারবেন না।

ফাইল ডিলিট করা

যে সকল ফাইলের আর দরকার নেই তা আপনার সবসময় ডিলিট করে ফেলা উচিত। একটা ফাইল ডিলিট করার সবচেয়ে সোজাসাপ্টা রাস্তা হচ্ছে ওপেনড ফাইল রেফারেন্স যা নিজেতে delete() কল করে।

```
myFile.delete();
```

যদি ফাইল ইন্টারনাল ফাইলে সেভ হয়, আপনি deleteFile() কল করে একটি ফাইলের স্থান চিহ্নিত করা এবং ডিলিট করতে Context কে জিজ্ঞাসাও করতে পারেন:

```
myContext.deleteFile (fileName);
```

নোট: যখন ইউজার আপনার অ্যাপ আনইনস্টল করে, অ্যান্ড্রয়েড সিস্টেম নিজে থেকে বিধ্বস্ত বিষয়গুলো ডিলিট করে:

- যে সকল ফাইল আপনি ইন্টারনাল স্টোরেজে সেভ করেছেন
- getExternalFilesDir() ব্যবহার করে যে সকল ফাইল আপনি এক্সটারনাল স্টোরেজে সেভ করেছেন

যাহোক, আপনার উচিত ম্যানুয়ালি getCacheDir() দ্বারা তৈরী সকল ক্যাশে ফাইল নিয়মিতভাবে ডিলিট করা এবং অন্যান্য ফাইল যা আপনার আর দরকার নেই তাও নিয়মিতভাবে ডিলিট করুন।

SQL ডাটাবেজে ডাটা সেভ করা

(<http://developer.android.com/training/basics/data-storage/databases.html>)

পুনরাবৃত্তি হওয়া ডাটা বা কাঠামোবদ্ধ ডাটার জন্য ডাটাবেজে ডাটা সেভ করাটা ভালো, যেমন কনটাক্ট ইনফর্মেশন। এই ক্লাস অনুমান করে যে আপনি মোটামুটিভাবে SQL ডাটাবেজে এর সাথে পরিচিত এবং আপনাকে অ্যান্ড্রয়েডে SQLite ডাটাবেজ সহ কাজ শুরু করতে সাহায্য করে। এপিআই যা অ্যান্ড্রয়েডে ডাটাবেজে ব্যবহার করতে আপনার প্রয়োজন হবে তা `android.database.sqlite` প্যাকেজে আছে।

স্কিমা (Schema) এবং কনট্রাক্ট নির্ধারণ

SQL ডাটাবেজের অন্যতম প্রধান নীতি হচ্ছে স্কিমা: কীভাবে ডাটাবেজ বিন্যাস্ত হবে তার একটা আনুষ্ঠানিক ঘোষণা। স্কিমা SQL স্টেটমেন্টে প্রতিফলন হয় যা আপনি আপনার ডাটাবেজ তৈরীতে ব্যবহার করেন। আপনি দেখবেন যে একটা সঙ্গি ক্লাস তৈরীতে এটা সাহাজ্য করে, যা কনট্রাক্ট ক্লাস হিসাবে পরিচিত, যা স্পষ্টভাবে সিস্টেমেটিক এবং সেন্স-ডকুমেন্টিং উপায়ে আপনার স্কিমার লেআউট সুনির্দিষ্ট করে।

একটি কনট্রাক্ট ক্লাস কনসটেন্ট এর জন্য কনটেইনার যা URLs, টেবল এবং কলাম এর জন্য নাম নির্ধারণ করে। কনট্রাক্ট ক্লাস আপনার একই প্যাকেজের মধ্যে অন্য সকল ক্লাসে একই কনসটেন্ট ব্যবহার করাটাকে অনুমোদন করে। এটা আপনাকে একটা স্থানে কলাম নাম পরিবর্তন করতে দেয় এবং আপনার কোডের সর্বত্র সঞ্চারিত হতে দেয়।

একটি কনট্রাক্ট ক্লাস বিন্যাস করতে ভালো উপায় হচ্ছে সংজ্ঞা আরোপ করা যা ক্লাসের রুট লেভেলে সম্পূর্ণ ডাটাবেজে বৈশ্বিক হয়। তারপর প্রতিটা টেবলের জন্য একটা ইনার ক্লাস তৈরী করুন যা এর কলামকে গণনা করে।

নোট: BaseColumns ইন্টারফেসটি বাস্তবায়ন করার মাধ্যমে, আপনার ইনার ক্লাস ID যা কিছু অ্যান্ড্রয়েড ক্লাস নামে একটা প্রাথমিক কি ফিল্ড পেতে পারে যেমন কারসর অ্যাডাপটর এটার থাকাকে প্রত্যাশা করে। এটার প্রয়োজন নেই তবে এটা অ্যান্ড্রয়েড ফ্রেমওয়ার্ক দিয়ে আপনার ডাটাবেজ চমৎকারভাবে কাজ করতে পারবে।

উদাহরণস্বরূপ, এই স্নিপেট একটি সিঙ্গেল টেবিলের জন্য টেবিল নাম এবং কলাম নাম নির্ধারণ করে দেয়:

```
public final class FeedReaderContract {
    // To prevent someone from accidentally instantiating the contract class,
    // give it an empty constructor.
    public FeedReaderContract() {}

    /* Inner class that defines the table contents */
    public static abstract class FeedEntry implements BaseColumns {
        public static final String TABLE_NAME = "entry";
        public static final String COLUMN_NAME_ENTRY_ID = "entryid";
        public static final String COLUMN_NAME_TITLE = "title";
        public static final String COLUMN_NAME_SUBTITLE = "subtitle";
        ...
    }
}
```


SQL হেলপার ব্যবহার করে ডাটাবেজ তৈরী

পূর্বে আপনি যদি ঠিক করে থাকেন আপনার ডাটাবেজ দেখতে কেমন হবে, আপনাকে মেথড বাস্তবায়ন করতে হবে যা ডাটাবেজ এবং টেবিল তৈরী এবং দেখাশোনা করে। এখানে কিছু সাধারণ বিবৃতি আছে যা একটি টেবিল তৈরী এবং ডিলিট করে:

```
private static final String TEXT_TYPE = " TEXT";
private static final String COMMA_SEP = ",";
private static final String SQL_CREATE_ENTRIES =
    "CREATE TABLE " + FeedEntry.TABLE_NAME + " (" +
    FeedEntry._ID + " INTEGER PRIMARY KEY," +
    FeedEntry.COLUMN_NAME_ENTRY_ID + TEXT_TYPE + COMMA_SEP +
    FeedEntry.COLUMN_NAME_TITLE + TEXT_TYPE + COMMA_SEP +
    ... // Any other options for the CREATE command
    ")";

private static final String SQL_DELETE_ENTRIES =
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE_NAME;
```

ঠিক ডিভাইসের ইন্টারনাল স্টোরেজে (internal storage) সেভ করা ফাইলের মতো, অ্যান্ড্রয়েড আপনার ডাটাবেজ প্রাইভেট ডিস্ক স্পেসে জমা করে যা সংযুক্ত অ্যাপলিকেশন। আপনার ডাটা নিরাপদ, কারন বাই ডিফল্ট এই এলাকা অন্য অ্যাপলিকেশনের জন্য উন্মুক্ত নয়।

SQLiteOpenHelper ক্লাসে উপকারী এক সেট এপিআই পাওয়া যায়। যখন আপনি আপনার ডাটাবেজে রেফারেন্স পেতে এই ক্লাস ব্যবহার করেন, সিস্টেমটি ডাটাবেজ তৈরী এবং আপডেট এর সম্ভাব্য লং-রানিং অপারেশন সম্পাদন করে শুধুমাত্র যখন প্রয়োজন হয় তখন, এবং অ্যাপ শুরুর সময়ে নয়। আপনার এই সব করতে getWritableDatabase()এবং getReadableDatabase()কল করা।

নোট: কারন তারা লং-রানিং হতে পারে, নিশ্চিত হোন যে আপনি একটি ব্যাকগ্রাউন্ড থ্রেড এ getWritableDatabase()বা getReadableDatabase()কল করেছেন যেমন AsyncTask বা IntentService দিয়ে।

SQLiteOpenHelper ব্যবহার করতে একটি সাবক্লাস তৈরী করুন যা onCreate(), onUpgrade()এবং onOpen()কলব্যাক মেথডকে ওভাররাইড করে। আপনি হয়তো onDowngrade()বাস্তবায়ন করতে চান, কিনুত এটার প্রয়োজন নেই।

উদাহরণস্বরূপ, এখানে SQLiteOpenHelperএর একটি বাস্তবায়ন আছে যা উপরে দেখানো কমান্ডের কিছু ব্যবহার করে:

```
public class FeedReaderDbHelper extends SQLiteOpenHelper {
    // If you change the database schema, you must increment the database version.
    public static final int DATABASE_VERSION = 1;
    public static final String DATABASE_NAME = "FeedReader.db";

    public FeedReaderDbHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
}
```

```

public void onCreate(SQLiteDatabase db) {
    db.execSQL(SQL_CREATE_ENTRIES);
}
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    // This database is only a cache for online data, so its upgrade policy is
    // to simply to discard the data and start over
    db.execSQL(SQL_DELETE_ENTRIES);
    onCreate(db);
}
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    onUpgrade(db, oldVersion, newVersion);
}
}

```

আপনার ডাটাবেজে প্রবেশ করতে, আপনার SPLiteOpenHelperএর সাবক্লাস শুরু করুন:

```

FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getApplicationContext());

```

তথ্য একটা ডাটাবেজে রাখুন

insert()মেথডে ContentValues অবজেক্ট পাস করে ডাটাবেজে ডাটা প্রবেশ করান:

```
// Gets the data repository in write mode
SQLiteDatabase db = mDbHelper.getWritableDatabase();

// Create a new map of values, where column names are the keys
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_ENTRY_ID, id);
values.put(FeedEntry.COLUMN_NAME_TITLE, title);
values.put(FeedEntry.COLUMN_NAME_CONTENT, content);

// Insert the new row, returning the primary key value of the new row
long newRowId;
newRowId = db.insert(
    FeedEntry.TABLE_NAME,
    FeedEntry.COLUMN_NAME_NULLABLE,
    values);
```

insert()এর প্রথম আলোচনা হচ্ছে শুধুমাত্র টেবিল নাম। দ্বিতীয় আলোচনা (আর্গুমেন্ট) একটা কলামের নাম প্রদান করে যেখানে ফ্রেমওয়ার্ক ইভেন্টে NULL প্রবেশ করাতে পারে যাতে ContentValues খালি হয় (আপনি যদি পরিবর্তে "null" এ এটা সেট করেন, যখন এখানে কোন ভ্যালু থাকবে না তখন ফ্রেমওয়ার্ক একটি রো প্রবেশ করাবে না)

ডাটাবেজ থেকে তথ্য পাঠ করা

ডাটাবেজ থেকে পড়তে, `query()` মেথড ব্যবহার করুন, আপনার বাছাই প্রক্রিয়া এবং কাঙ্ক্ষিত কলামে এটা পাস করে। মেথডটি `insert()` এবং `update()` এর এলিমেন্টগুলো একত্রিত করে, ব্যতিক্রম শুধু কলাম লিস্ট, আপনি যে ডাটা পেতে চান তা নির্ধারণ করে, ডাটা প্রবেশ কারনোর পূর্বে। অনুসন্ধানের ফলাফল আপনার কাছে একটি `Cursor` অবজেক্টের মধ্যে ফিরে আসবে।

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// Define a projection that specifies which columns from the database
// you will actually use after this query.
String[] projection = {
    FeedEntry._ID,
    FeedEntry.COLUMN_NAME_TITLE,
    FeedEntry.COLUMN_NAME_UPDATED,
    ...
};

// How you want the results sorted in the resulting Cursor
String sortOrder =
    FeedEntry.COLUMN_NAME_UPDATED + " DESC";

Cursor c = db.query(
    FeedEntry.TABLE_NAME, // The table to query
    projection,            // The columns to return
    selection,             // The columns for the WHERE clause
    selectionArgs,         // The values for the WHERE clause
    null,                 // don't group the rows
    null,                 // don't filter by row groups
    sortOrder             // The sort order
);
```

কোর্সরের মধ্যে একটা রো দেখতে, যে কোন একটা `Cursor` মুভ মেথড ব্যবহার করুন, রিডিং ভ্যালু শুরু করার পূর্বে আপনাকে অবশ্যই সবসময় যাকে কল করতে হবে। সাধারনত, `moveToFirst()` কল করার মাধ্যমে আপনার শুরু করা উচিত, যা "read position" কে রেজাল্টের প্রথম প্রবেশের উপর প্লেস করে। প্রতিটা রো এর জন্য, আপনি একটি কলামের ভ্যালু পড়তে পারেন যেকোন একটি `Cursor` গেট মেথড কল করার মাধ্যমে, যেমন, `getString()` বা `getLong()`। প্রতিটা গেট মেথডের জন্য আপনাকে অবশ্যই আপনার কাঙ্ক্ষিত কলামের ইনডেক্স পজিশন পাস করতে হবে, যা আপনি `getColumnIndex()` বা `getColumnIndexOrThrow()` কল করে পেতে পারেন। উদাহরণস্বরূপ:

```
cursor.moveToFirst();
long itemId = cursor.getLong(
    cursor.getColumnIndexOrThrow(FeedEntry._ID)
);
```

ডাটাবেজ থেকে তথ্য ডিলিট করা

একটি টেবিল থেকে রো ডিলিট করতে, আপনার সিলেকশন ক্রাইটেরিয়া প্রদান করা উচিত যা রো কে চিহ্নিত করে। ডাটাবেজ এপিআই সিলেকশন ক্রাইটেরিয়ার জন্য একটি মেকানিজম প্রদান করে যা SQL ইনজেকশন এর কাছ থেকে প্রতিরোধ করে। মেকানিজমটি সিলেকশন স্পেসিফিকেশনকে সিলেকশন ক্লজ এবং সিলেকশন আর্গুমেন্ট এ ভাগ করে। ক্লজ কলামকে নির্ধারণ করে দেখতে এবং কলাম টেস্ট একত্রিত করতে আপনাকে অনুমোদনও করতে দেয়। আর্গুমেন্ট হচ্ছে ভ্যালু বিপরীত পরিষ্কা যা ক্লজের মধ্যে আবদ্ধ হয়। কারন নিয়মিত SQL স্টেটমেন্টের মতো একইভাবে রেজাল্ট পরিচালিত হয় না, এটা SQL ইনজেকশন থেকে নিরাপদ।

```
// Define 'where' part of query.  
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";  
// Specify arguments in placeholder order.  
String[] selectionArgs = { String.valueOf(rowId) };  
// Issue SQL statement.  
db.delete(table_name, selection, selectionArgs);
```

ডাটাবেজ আপডেট করা

যখন আপনার ডাটাবেজ ভ্যালুর একটি সাবসেট পরিবর্তন করতে হবে, তখন update() মেথড ব্যবহার করুন।

টেবিল আপডেট করে update() এর যিৎব সিনট্যাক্স দিয়ে insert() এর কনটেন্ট ভ্যালু সিনট্যাক্স একত্রিত করে।

```
SQLiteDatabase db = mDbHelper.getReadableDatabase();

// New value for one column
ContentValues values = new ContentValues();
values.put(FeedEntry.COLUMN_NAME_TITLE, title);

// Which row to update, based on the ID
String selection = FeedEntry.COLUMN_NAME_ENTRY_ID + " LIKE ?";
String[] selectionArgs = { String.valueOf(rowId) };

int count = db.update(
    FeedReaderDbHelper.FeedEntry.TABLE_NAME,
    values,
    selection,
    selectionArgs);
```

অন্য অ্যাপের সাথে সম্পর্ক স্থাপন

(<http://developer.android.com/training/basics/intents/index.html>)

একটি অ্যান্ড্রয়েডের স্বাভাবিকভাবে কিছু একটিভিটি থাকে (activities)। প্রতিটা একটিভিটি একটা ইউজার ইন্টারফেস প্রদর্শন করে যা ইউজারকে একটি সুনির্দিষ্ট কাজ করতে দেয় (যেমন একটা ম্যাপ দেখা বা একটা ফটো নেওয়া)। ইউজারকে একটা একটিভিটি থেকে আরেকটি একটিভিটিতে নিয়ে যেতে আপনার অ্যাপকে অবশ্যই কিছু করতে, অ্যাপের "intent"কে নির্ধারণ করতে একটি Intent ব্যবহার করা উচিত। যখন আপনি startActivity()এর মতো একটা মেথড দিয়ে সিস্টেমে একটা Intentপাস করেন, সিস্টেম যথাযথ অ্যাপ উপাদান খুঁজে বের করতে এবং শুরু করতে Intent ব্যবহার করে। ইনটেন্ট ব্যবহার আপনার অ্যাপকে অন্য অ্যাপে থাকা একটি একটিভিটি শুরু করতে অনুমোদন করে।

একটি নির্দিষ্ট কম্পোনেন্ট (একটি নির্দিষ্ট Activity ইনসটেন্স) শুরু করার জন্য একটি Intent *explicit* হতে পারে অথবা কোন কম্পোনেন্ট যা পরিকল্পিত কাজ (যেমন একটা ফটো ধারণ করা) করে তা শুরু করতে ইনটেন্ট *implicit* ও হতে পারে।

এই ক্লাস আপনাকে দেখাবে কীভাবে অন্য অ্যাপের সাথে কিছু মৌলিক মিথস্ক্রিয়ার কাজ সম্পাদন করতে একটি Intent ব্যবহার করা হয়, যেমন অন্য অ্যাপ শুরু করতে, ওই অ্যাপ থেকে রেজাল্ট গ্রহণ করতে এবং আপনার অ্যাপকে অন্য অ্যাপের ইনটেন্টে প্রতিক্রিয়া জানানোর মতো সক্ষম করা।

এই অধ্যায়ের অনুশীলনী সমূহ

ইউজারকে অন্য অ্যাপে পাঠানো

দেখানো হবে অন্য অ্যাপ যা একটি কাজ করতে পারবে তা শুরু করতে আপনি কীভাবে ইমপ্লিসিট ইনটেন্ট তৈরী করতে পারবেন

একটি একটিভিটি থেকে রেজাল্ট পাওয়া

দেখানো হবে কীভাবে অন্য একটিভিটি শুরু করতে হয় এবং একটিভিটিটি থেকে কীভাবে একটি রেজাল্ট গ্রহণ করতে হয়

অন্য অ্যাপেকে আপনার অ্যাপ শুরু করতে দেয়া

দেখানো হবে ইনটেন্ট ফিল্টার যা আপনার অ্যাপ কর্তৃক গ্রহণকৃত ইমপ্লিসিট ইনটেন্টকে ডিক্লেয়ার করে তা নির্ধারণ করার মাধ্যমে অন্য অ্যাপ কর্তৃক ব্যবহার করার জন্য কীভাবে আপনার অ্যাপের একটিভিটি তৈরী করতে হয়।

ইউজারকে অন্য অ্যাপে পাঠানো

(<http://developer.android.com/training/basics/intents/sending.html>)

একটি অ্যান্ড্রয়েডের সবচেয়ে গুরুত্ব বৈশিষ্ট্য হচ্ছে একটা কাজের (যা করতে চাওয়া হয়) উপর ভিত্তি করে ইউজারকে অন্য অ্যাপে পাঠানোর ক্ষেত্রে একটা অ্যাপের সামর্থ্য। উদাহরণস্বরূপ, যদি আপনার অ্যাপের একটা ব্যবসার ঠিকানা থাকে যা আপনি প্রদর্শন করতে চান, ম্যাপ প্রদর্শনের জন্য আপনার অ্যাপে একটা একটিভিটি তৈরী করার দরকার নেই। পরিবর্তে আপনি একটি Intent ব্যবহার করে ঠিকানা দেখার জন্য একটি রিকোয়েস্ট তৈরী করতে পারেন। তখন অ্যান্ড্রয়েড সিস্টেম একটি অ্যাপ শুরু করবে যা ম্যাপের উপর ঠিকানা শো করতে সামর্থ্য হবে।

প্রথম ক্লাস (Building Your First App) যেভাবে ব্যাখ্যা করা হয়েছে, আপনার অ্যাপের একটিভিটির মধ্যে নেভিগেট করতে আপনাকে অবশ্যই ইনটেন্ট ব্যবহার করতে হবে। আপনি সাধারণভাবে তা এক্সপ্লিসিট ইনটেন্ট দিয়ে করতে পারবেন, যা কম্পোনেন্টের সঠিক ক্লাস নাম নির্ধারণ করে যা আপনি শুরু করতে চান। কিন্ত যখন আপনি চান একটি কাজ করতে একটা আলাদা অ্যাপ থকুক, যেমন “একটি ম্যাপ দেখা” আপনাকে তখন অবশ্যই একটি ইমপ্লিসিট ইনটেন্ট ব্যবহার করতে হবে।

এই অনুশীলনী আপনাকে শেখাবে কীভাবে একটি সুনির্দিষ্ট কাজের জন্য একটি ইমপ্লিসিট ইনটেন্ট তৈরী করা যায় এবং কীভাবে এটাকে ব্যবহার করে একটা একটিভিটি শুরু করা যায় যা অন্য অ্যাপে কাজটি সম্পাদন করে।

একটি ইমপ্লিসিট ইনটেন্ট তৈরী করা

ইমপ্লিসিট ইনটেন্ট শুরু করতে কম্পোনেন্ট এর ক্লাস নেম ডিক্লেয়ার করে না কিনুত পরিবর্তে সম্পাদন করতে একটি একশন ডিক্লেয়ার করে। একশনটি যে বিষয়গুলো আপনি করতে চান তা নির্দিষ্ট করে, যেমন, কোন কিছু ভিউ, এডিট, সেন্ড, গেট ইত্যাদি। ইনটেন্ট মাঝে মাঝে একশনের সাথে সম্পৃক্ত ডাটা অন্তর্ভুক্ত করে, যেমন ঠিকানাটি যা আপনি দেখতে চান অথবা ইমেইল মেসেজ যা আপনি সেন্ড করতে চান। আপনি কী ধরনের ইনটেন্ট তৈরী করতে চান তার উপর নির্ভর করে, ডাটা একটি Uri হতে পারে, অন্য আরও ডাটার একটি অথবা ইনটেন্টের আদৌ ডাটার দরকার নেই।

আপনার ডাটা যদি Uri হয়, একটা সরল Intent() আছে যা আপনি একশন এবং ডাটা নির্ধারন করতে ব্যবহার করতে পারেন।

উদাহরণস্বরূপ, এখানে দেখানো হয়েছে টেলিফোন নাম্বার সুনির্দিষ্ট করতে Uri ডাটা ব্যবহার করে একটা ফোন কল আরম্ভ করতে কীভাবে একটি ইনটেন্ট তৈরী করতে হয়:

```
Uri number = Uri.parse("tel:5551234");  
Intent callIntent = new Intent(Intent.ACTION_DIAL, number);
```

যখন আপনার অ্যাপ startActivity()কল করে এই ইনটেন্টকে আবাহন করে, ফোন অ্যাপটি প্রদত্ত ফোন নাম্বারে একটি কল করে।

এখানে আছে একজোড়া অন্য ইনটেন্ট এবং তাদের একশন এবং Uri ডাটার জোড়া:

- View a map:/ ম্যাপ ভিউ একটি:

```
// Map point based on address Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California"); // Or map point based on latitude/longitude // Uri location = Uri.parse("geo:37.422219,-122.08364?z=14"); // z param is zoom level Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);
```

- View a web page:/ ওয়েব পেজ ভিউ:

```
Uri webpage = Uri.parse("http://www.android.com");  
Intent webIntent = new Intent(Intent.ACTION_VIEW, webpage);
```

ইমপ্লিসিট ইনটেন্টের অন্য ধরনে “এক্সট্রা” ডাটার প্রয়োজন যা ভিন্ন ডাটা টাউপস প্রদান করে, যেমন একটি স্ট্রিং। আপনি নানা ধরনের putExtra()মেথড ব্যবহার করে এক্সট্রা ডাটার এক বা একাধিক অংশ সংযুক্ত করতে পারেন।

বাই ডিফল্ট, সিস্টেমাট Uri ডাটা যা অন্তর্ভুক্ত আছে তার উপর ভিত্তি করে ইনটেন্ট কর্তৃক চাওয়া যথাযথ MIME টাইপটি নির্ধারণ করে। যদি আপনি একটি Uri ইনটেন্টে অন্তর্ভুক্ত না করেন, স্বাভাবিকভাবে ইনটেন্টের সাথে সম্পৃক্ত ডাটার টাইপ সুনির্দিষ্ট করতে আপনার উচিত setType()ব্যবহার করা। MIME টাইপটি সেটিং করে আরও সুনির্দিষ্টি করার এই ধরনের একটিভিটিগুলোর ইনটেন্ট গ্রহণ করা উচিত।

এখানে আরও কিছু ইনটেন্ট আছে যা কাঙ্ক্ষিত একশন সুনির্দিষ্ট করতে এক্সট্রা ডাটা এড করে:

- এটাচমেন্টসহ একটা ইমেইল পাঠান:

```
Intent emailIntent = new Intent(Intent.ACTION_SEND);
// The intent does not have a URI, so declare the "text/plain" MIME type
emailIntent.setType(HTTP.PLAIN_TEXT_TYPE);
emailIntent.putExtra(Intent.EXTRA_EMAIL, new String[] {"jon@example.com"}); // recipients
emailIntent.putExtra(Intent.EXTRA_SUBJECT, "Email subject");
emailIntent.putExtra(Intent.EXTRA_TEXT, "Email message text");
emailIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse("content://path/to/email/attachment"));
// You can also attach multiple items by passing an ArrayList of Uris
```

- একটি ক্যালেন্ডার ইভেন্ট তৈরী করুন:

```
Intent calendarIntent = new Intent(Intent.ACTION_INSERT, Events.CONTENT_URI);
Calendar beginTime = Calendar.getInstance().set(2012, 0, 19, 7, 30);
Calendar endTime = Calendar.getInstance().set(2012, 0, 19, 10, 30);
calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, beginTime.getTimeInMillis());
calendarIntent.putExtra(CalendarContract.EXTRA_EVENT_END_TIME, endTime.getTimeInMillis());
calendarIntent.putExtra(Events.TITLE, "Ninja class");
calendarIntent.putExtra(Events.EVENT_LOCATION, "Secret dojo");
```

নোট: ক্যালান্ডার ইভেন্টের জন্য এই ইনটেন্ট শুধুমাত্র এপিআই লেভেল ১৪ এবং এর চেয়ে উপরের লেভেল সাপোর্ট করে।

নোট: এটা গুরুত্বপূর্ণ যে আপনি যতটুকু সুনির্দিষ্ট করা সম্ভব ঠিক সেভাবে আপনার Intentনির্ধারণ করেছেন। উদাহরণস্বরূপ, যদি আপনি ACTION_VIEW ইনটেন্ট ব্যবহার করে একটি ইমেজ প্রদর্শন করতে চান, আপনার image/*এর একটি MIME টাইপ সুনির্দিষ্ট করা উচিত। এটা অ্যাপস পরিহার করে যা ইনটেন্ট কর্তৃক শুরু হওয়া থেকে অন্য ধরনের ডাটা (যেমন একটি ম্যাপ) কে “ভিউ” করতে পারে।

ইনটেন্টটি গ্রহণ করতে একটি অ্যাপ আছে কিনা যাচাই করুন

যদিও অ্যান্ড্রয়েড প্ল্যাটফর্ম নিশ্চয়তা দেয় যে কিছু ইনটেন্ট যেকোন একটি বিল্ট-ইন অ্যাপস এ সুনির্দিষ্ট করা থাকবে (যেমন, ফোন, ইমেইল বা ক্যালেন্ডার অ্যাপ), একটি ইনটেন্ট কে আবাহন করার পূর্বে আপনার সব সময় একটি যাচাইকরন পদক্ষেপ থাকা উচিত।

সতর্কতা: আপনি যদি একটা ইনটেন্ট আবাহন করেন এবং ডিভাইসে ইনটেন্টটিকে নিয়ে কাজ করার মতো কোন অ্যাপ না থাকে, আপনার অ্যাপ ক্রাশ করবে।

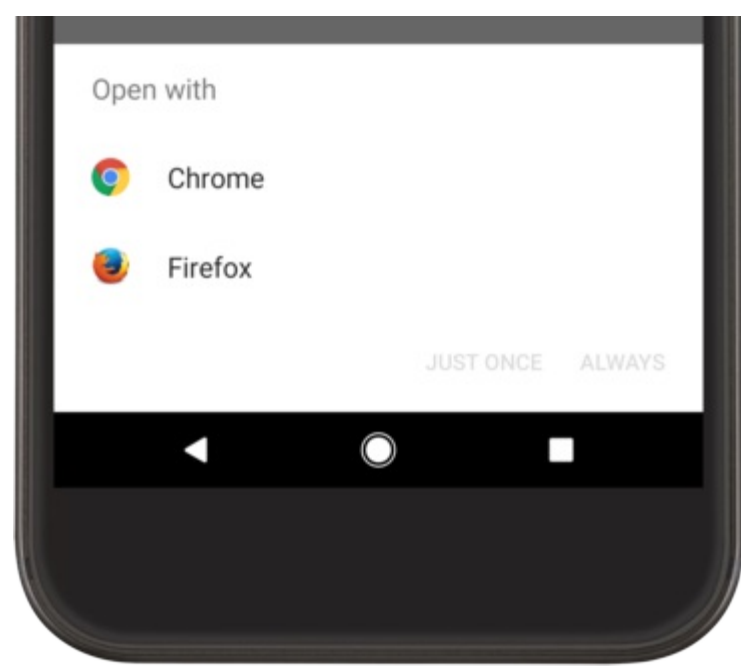
যাচাই করার জন্য একটা একটিভিটি আছে যা ইনটেন্টের রেসপন্স করে, আপনার Intent কে নিয়ে কাজ করতে পারে এমন একটিভিটির তালিকা পেতে QueryIntentActivities()কে কল করুন। যদি ফিরতি List খালি না থাকে, আপনি নিরাপদে কাজ শুরু করতে পারেন। উদাহরণস্বরূপ:

```
PackageManager packageManager = getPackageManager();  
List<ResolveInfo> activities = packageManager.queryIntentActivities(intent, 0);  
boolean isIntentSafe = activities.size() > 0;
```

যদি isIntentSafe, true হয়, তখন কমপক্ষে একটি অ্যাপ ইনটেন্টের প্রতি রেসপন্স করবে। যদি এটা false হয়, তখন ইনটেন্টকে নিয়ে কাজ করার মতো কোন অ্যাপ থাকে না।

নোট: আপনার উচিত এই যাচাই কাজ সম্পাদন করা যখন আপনার একটিভিটি প্রথমে শুরু করে, যদি আপনার ফিচারটি নিষ্ক্রিয় করার দরকার হয় যে ফিচার ইনটেন্টটি ইউজার কর্তৃক ব্যবহারের প্রয়াসের পূর্বেই ব্যবহার করে থাকে। আপনি যদি কোন নির্দিষ্ট অ্যাপ সম্পর্কে জানেন যেটা ইনটেন্টকে নিয়ে কাজ করতে পারবে, আপনি অ্যাপটি ইউজার কর্তৃক ডাউনলোড করার জন্য একটি লিংক সরবরাহ করতে পারেন।(দেখুন গুগল প্লেতে কীভাবে আপনার পণ্যে লিংক যুক্ত করবেন) (<http://developer.android.com/distribute/googleplay/promote/linking.html>)

ইনটেন্ট দিয়ে একটি একটিভিটি শুরু করুন



ফিগার ১. সিলেকশন ডায়ালগের উদাহরণ যা দৃশ্যমান হয় যখন একাধিক অ্যাপ একটি একটিভিটি নিয়ে কাজ করতে পারে।

পূর্বে আপনি যদি আপনার Intent তৈরী করে থাকেন এবং এক্সট্রা ইনফো সেট করে থাকেন, এটা সিস্টেমে সেল্ড করতে `startActivity()` কল করুন। যদি সিস্টেমটি একের অধিক একটিভিটি চিহ্নিত করে থাকে যা ইনটেন্টটিকে নিয়ে কাজ করতে পারবে, এটা ইউজারের জন্য কোন অ্যাপটি ব্যবহার করবে তা বেছে নিতে একটি ডায়ালগ প্রদর্শন করবে, ফিগার ১ এ যেভাবে দেখানো হয়েছে। যদি শুধু একটি একটিভিটি থাকে যা ইনটেন্ট নিয়ে কাজ করে, সিস্টেমটি তাৎক্ষনিকভাবে এটা শুরু করে।

`startActivity(intent);`

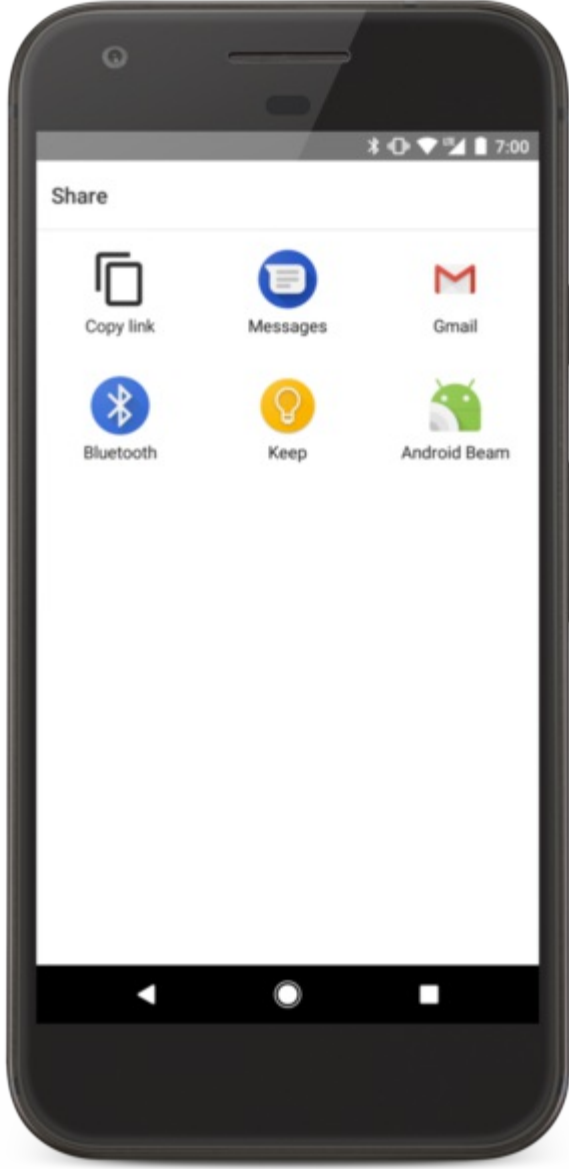
এখানে একটা সম্পূর্ণ উদাহরণ যা দেখাবে কীভাবে একটা ম্যাপ দেখার জন্য একটি ইনটেন্ট তৈরী করতে হয়, ইনটেন্ট নিয়ে কাজ করতে একটা অ্যাপ বিদ্যমান আছে তা যাচাই করা, তারপর এটা শুরু করা:

```
// Build the intent
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California");
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe) {
    startActivity(mapIntent);
}
```

একটি অ্যাপ চুজার প্রদর্শন করা



ফিগার ২. একটি চুজার ডায়ালগ

লক্ষ্য করুন যে যখন `startActivity()`তে `Intent` পাস করার মাধ্যমে আপনি একটি একটিভিটি শুরু করেন এবং সেখানে একের অধিক অ্যাপ আছে যা ইনটেন্টের প্রতি রেসপন্স করে তখন ইউজার বাছাই করতে পারবে যে কোন অ্যাপ বাই ডিফল্ট ব্যবহার করতে হবে(ডায়ালগের একদম নীচে একটি চেকবক্স সিলেক্ট করে, ফিগার ১. দেখুন)। এটা ভালো, যখন একটা একশন সম্পাদন করা হয় যার জন্য ইউজার সাধারণত প্রতিবার একই অ্যাপ ব্যবহার করতে চায়, যেমন যখন একটি ওয়েব পেজ ওপেন করা হয় (ইউজার সম্ভবত মাত্র একটা ওয়েব ব্রাউজার ব্যবহার করে) বা একটা ফটো নেওয়া হয় (ইউজার সম্ভবত একটা ক্যামেরা পছন্দ করে)।

কিন্তু, যদি সম্পাদিত একশন বহুবিধ অ্যাপস কর্তৃক নিয়ন্ত্রিত হয় এবং ইউজার প্রতিবার ভিন্ন ভিন্ন অ্যাপ পছন্দ করতে পারে-যেমন একটি “শেয়ার” একশন, যার জন্য ইউজারের অনেকগুলো অ্যাপস থাকতে পারে যার মাধ্যমে তারা একটি আইটেম শেয়ার করতে পারে- আপনার উচিত

স্পষ্টভাবে একটা চুজার ডায়লগ প্রদর্শন করা যেভাবে ফিগার ২ তে দেখানো হয়েছে। চুজার ডায়লগ প্রতিবার একশনের জন্য কোন অ্যাপ ব্যবহার করতে হবে তা বেছে নিতে ইউজারকে বাধ্য করে (ইউজার একশনের জন্য ডিফল্ট অ্যাপ বেছে নিতে পারে না)।

একটি চুজার প্রদর্শন করতে `createChooser()` ব্যবহার করে একটি `Intent` তৈরী করুন এবং `startActivity()` প্রতি এটা পাস করে দিন। উদাহরণস্বরূপ:

```
Intent intent = new Intent(Intent.ACTION_SEND);
...

// Always use string resources for UI text.
// This says something like "Share this photo with"
String title = getResources().getString(R.string.chooser_title);
// Create and start the chooser
Intent chooser = Intent.createChooser(intent, title);
startActivity(chooser);
```

এটা একটি অ্যাপসের তালিকা সহ একটি ডায়লগ প্রদর্শন করে যা `createChooser()` মেথডকে পাস করে ইনটেন্টের প্রতি রেসপন্স করে এবং ডায়লগ টাইটেল হিসাবে সরবরাহকৃত টেক্সটটি ব্যবহার করে।

একটি একটিভিটি থেকে রেজাল্ট পাওয়া

(<http://developer.android.com/training/basics/intents/result.html>)

অন্য একটিভিটি শুরু করাটা এখনুখি হতে হবে না। আপনি একটা একটিভিটি শুরু করতে পারেন এবং একটা ফিরতি রেজাল্ট গ্রহন করতে পারেন। একটা রেজাল্ট পেতে `startActivityForResult()` কল করুন (`startActivity()` এর পরিবর্তে)।

উদাহরণস্বরূপ, আপনার অ্যাপ একটা ক্যামেরা শুরু করতে পারে এবং ফলাফল হিসাবে ক্যাপচার করা ফটো পাবে। অথবা ইউজার কর্তৃক একটি কনট্যাক্ট বেছে নেওয়ার জন্য আপনি পিপল অ্যাপ শুরু করতে পারেন এবং ফলাফল হিসাবে আপনি কনট্যাক্ট ডিটেইলস পেতে পারেন।

অবশ্যই, একটিভিটিটি যা রেসপন্স করে তাকে এমনভাবে ডিজাইন করতে হবে যাতে একটা ফিরতি রেজাল্ট আসে। যখন এটা হবে, এটা অন্য Intent অবজেক্ট হিসাবে রেজাল্টটিকে সেন্ড করবে। আপনার একটিভিটি এটা `onActivityResult()` কলব্যাকে রিসিভ করবে।

নোট: যখন আপনি `startActivityForResult()` কল করবেন আপনি এক্সপ্লিসিট বা ইমপ্লিসিট ইনটেন্ট ব্যবহার করতে পারেন। একটা রেজাল্ট রিসিভ করতে আপনার নিজস্ব একটিভিটির মধ্যে যখন একটি শুরু করবেন, আপনার উচিত একটা এক্সপ্লিসিট ইনটেন্ট ব্যবহার করা এটা নিশ্চিত করতে যে আপনি একটা কান্ট্রিভ রেজাল্ট গ্রহণ করবেন।

একটিভিটি শুরু করা

আপনার Intent অবজেক্ট ব্যবহার করাটা তেমন বিশেষ কিছু নয় যখন একটা রেজাল্টের জন্য একটা একটিভিটি শুরু করা হয়, কিন্ত আপনার `startActivityForResult()` মেথডে একটি অতিরিক্ত ইন্টিজার আরগুমেন্ট পাস করা দরকার।

ইন্টিজার আরগুমেন্ট হচ্ছে একটি “রিকোয়েস্ট কোড” যা আপনার রিকোয়েস্ট চিহ্নিত করবে। যখন আপনি রেজাল্ট Intent গ্রহণ করবেন, কলব্যাকটি একই রিকোয়েস্ট কোড সরবরাহ করে যাতে আপনার অ্যাপ যথাযথভাবে রেজাল্টটিকে চিহ্নিত করতে পারে এবং নির্ধারণ করতে পারে কীভাবে এটা নিয়ন্ত্রণ করতে হয়।

উদাহরণস্বরূপ, এখানে দেয়া আছে কীভাবে একটি একটিভিটি শুরু করতে হয় যা ইউজারকে একটি কন্ট্যাক্ট বাছাই করতে অনুমোদন করে:

```
static final int PICK_CONTACT_REQUEST = 1; // The request code
...
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contacts"));
    pickContactIntent.setType(Phone.CONTENT_TYPE); // Show user only contacts w/ phone numbers
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
}
```

রেজাল্টটি গ্রহণ (রিসিভ) করা

যখন ইউজার পরবর্তী একটিভিটি দিয়ে শেষ করবেন এবং ফিরবেন, সিস্টেমটি আপনার একটিভিটির `onActivityResult()` মেথডকে কল করে। এই মেথডে তিনটা আরগুমেন্ট অন্তর্ভুক্ত করে:

- রিকোয়েস্ট কোডটি আপনি `startActivityForResult()` এ পাস করেছেন
- দ্বিতীয় একটিভিটি কর্তৃক সুনির্দিষ্ট একটি রেজাল্ট কোড। এটা `RESULT_OK` হয় যদি অপারেশন কৃতকার্য হয়ে থাকে অথবা `RESULT_CANCELED` হয় যদি ইউজার ফিরে আসে বা কোন কারনে অপারেশন ব্যর্থ হয়ে থাকে।
- একটি `Intent` যা রেজাল্ট ডাটা বহন করে।

উদাহরণস্বরূপ, এখানে দেয়া আছে আপনি কীভাবে “পিক কনটেন্ট” ইনটেন্ট এর জন্য রেজাল্টকে নিয়ন্ত্রণ করতে পারেন:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

এই উদাহরণে, রেজাল্ট `Intent` অ্যান্ড্রয়েডের কনট্যাক্ট কর্তৃক ফিরে আসে অথবা পিপল অ্যাপ একটি কনটেন্ট `Uri` সরবরাহ করে যা ইউজার যে কনট্যাক্ট বাছাই করে তা চিহ্নিত করে।

রেজাল্টকে সফলভাবে হ্যান্ডল করতে, আপনাকে অবশ্যই বুঝতে হবে রেজাল্ট `Intent` এর ফরমেট কি হতে পারে। এমনটি করা সহজ যখন একটিভিটিটি একটি রেজাল্ট ফেরত নিয়ে এসে আপনার নিজস্ব একটিভিটির একটি হয়। অ্যান্ড্রয়েড প্ল্যাটফর্মের সাথে অন্তর্ভুক্ত অ্যাপস তাদের নিজস্ব এপিআইগুলো প্রস্তাব করে যাতে আপনি নির্দিষ্ট রেজাল্ট ডাটার উপর নির্ভর করতে পারেন। উদাহরণস্বরূপ পিপল অ্যাপ (কিছু পুরাতন সংস্করণের উপর কনট্যাক্ট অ্যাপ) সবসময় কনটেন্ট ইউআরআই দিয়ে একটা রেজাল্ট ফেরত আনে যা বাছাইকৃত কনট্যাক্ট চিহ্নিত করে, এবং ক্যামেরা অ্যাপ “ডাটা” এক্সট্রাতে একটি `Bitmap` ফেরত নিয়ে আসে (Capturing Photos সম্পর্কিত ক্লাসটি দেখুন)।

বোনাস: কনট্যাক্ট ডাটা পড়া

উপরোক্ত কোডটি দেখায় যে পিপল অ্যাপ থেকে কীভাবে রেজাল্ট পেতে হয় যা আসলে কীভাবে রেজাল্ট থেকে ডাটা পড়তে হয় সেসম্পর্কে বিস্তারিত আলোচনায় যায় না, কারন এটা content providers সম্পর্কে আরও উন্নত আলোচনার দাবী করে। কিনুত আপনি যদি উৎসাহী হোন, এখানে আরও কিছু কোড আছে যা আপনাকে দেখাবে বাছাই করা কনট্যাক্ট থেকে ফোন নাম্বার পেতে কীভাবে রেজাল্ট ডাটাকে অনুসন্ধান করতে হয়:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request it is that we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // Get the URI that points to the selected contact
            Uri contactUri = data.getData();
            // We only need the NUMBER column, because there will be only one row in the result
            String[] projection = {Phone.NUMBER};

            // Perform the query on the contact to get the NUMBER column
            // We don't need a selection or sort order (there's only one result for the given URI)
            // CAUTION: The query() method should be called from a separate thread to avoid blocking
            // your app's UI thread. (For simplicity of the sample, this code doesn't do that.)
            // Consider using CursorLoader to perform the query.
            Cursor cursor = getContentResolver()
                .query(contactUri, projection, null, null, null);
            cursor.moveToFirst();

            // Retrieve the phone number from the NUMBER column
            int column = cursor.getColumnIndex(Phone.NUMBER);
            String number = cursor.getString(column);

            // Do something with the phone number...
        }
    }
}
```

নোট: অ্যান্ড্রয়েড ২.৩ (এপিআই লেভেল ৯) এর পূর্বে, Contacts Provider এর উপর অনুসন্ধান করতে (উপরে যেমনটি দেখানো হয়েছে) প্রয়োজন যে আপনার অ্যাপ READ CONTACTS পারমিশন কে ডিক্লেয়ার করে (Security and Permissions দেখুন)। কিনুত অ্যান্ড্রয়েড ২.৩ দিয়ে শুরু করলে, কনট্যাক্ট/পিপল অ্যাপ কনট্যাক্ট প্রভাইডার থেকে পড়তে আপনার অ্যাপকে একটি অস্থায়ী অনুমতি প্রদান করে যখন এটা আপনাকে একটা রেজাল্ট ফেরত দিবে। অস্থায়ী অনুমতি শুধুমাত্র নির্দিষ্ট কনট্যাক্ট রিকোয়েস্ট এ প্রয়োগ হয়, যার ফলে যেটা ইনটেন্টের Uri কর্তৃক সুনির্দিষ্ট সেটা ছাড়া আপনি একটি কনট্যাক্টকে অনুসন্ধান করতে পারবেন না, যদি না আপনি READ CONTACTS পারমিশন ডিক্লেয়ার করেন

অন্য অ্যাপকে আপনার অ্যাপ শুরু করতে দেয়া

(<http://developer.android.com/training/basics/intents/filters.html>)

পূর্ববর্তী দুইটা অনুশীলনী গল্পের একটা দিক ফোকাস করে: আপনার অ্যাপ থেকে অন্য অ্যাপের একটিভিটি শুরু করা। কিন্ত যদি আপনার অ্যাপ একটা কাজ সম্পাদন করতে পারে যা অন্য অ্যাপের জন্য উপকারী হতে পারে, আপনার অ্যাপকে অন্য অ্যাপ থেকে একশন রিকোয়েস্ট এ রেসপন্স করার মতো করে প্রস্তুত করা উচিত। উদাহরণস্বরূপ, আপনি যদি একটা সামাজিক অ্যাপ তৈরী করেন যা ইউজারের বন্ধুদের সাথে মেসেজ বা ফটো শেয়ার করতে পারে, এটা আপনার আগ্রহের কেন্দ্রে থাকবে ACTION_SEND ইন্টেন্টকে সাপোর্ট করাটা, ফলে ইউজার অন্য অ্যাপ থেকে একটি “শেয়ার” একশন চালু করতে পারবে এবং একশনটি সম্পাদন করতে আপনার অ্যাপ শুরু করবে।

আপনার একটিভিটি শুরু করতে অন্য অ্যাপকে অনুমোদন দিতে, আপনার প্রয়োজন অনুরূপ `<activity>` এলিমেন্ট এর জন্য আপনার মেনিফেস্ট ফাইলে `<intent-filter>` এলিমেন্ট সংযুক্ত করা।

যখন আপনার অ্যাপ একটি ডিভাইসে ইনস্টল করবেন, সিস্টেম আপনার ইন্টেন্ট ফিল্টারস চিহ্নিত করে এবং ইনস্টল করা সকল অ্যাপ কর্তৃক সাপোর্ট করা ইন্টেন্টের ইন্টার্নাল ক্যাটালোগে ইনফরমেশনগুলো সংযুক্ত করে। যখন একটি অ্যাপ একটা ইমপ্লিসিট ইন্টেন্ট দিয়ে `startActivity()` বা `startActivityForResult()` কল করে, সিস্টেমটি কোন একটিভিটি (বা একটিভিটিগুলো) ইন্টেন্টে রেসপন্স করতে পারে।

ইনটেন্ট ফিল্টার সংযোজন (এ্যাড) করা

কোন ইনটেন্ট আপনার একটিভিটি ধারণ করতে পারবে তা যথাযথভাবে নির্ধারণ করার জন্য, আপনি সংযুক্ত করেছেন এমন প্রতিটা ইনটেন্ট ফিল্টারের একটিভিটি গ্রহণ করা একশন এবং ডাটার টাইপের মতো করে যতদূর সম্ভব নির্দিষ্ট হওয়া উচিত।

সিস্টেমটি একটি একটিভিটিতে একটি নির্দিষ্ট Intent সেন্ড করতে পারে যদি ওই একটিভিটির একটি ইনটেন্ট ফিল্টার থাকে যা Intent অবজেক্টের নিম্নোক্ত বৈশিষ্ট্যগুলো পূর্ণ করে:

একশন

একটি স্ট্রিং পরিচালনা করতে একশনের নামকরন করে। সাধারনত প্লাটফর্ম-নির্ধারিত ভ্যালু যেমন ACTION SEND বা ACTION VIEW ।

< action> এলিমেন্ট দিয়ে আপনার ইনটেন্ট ফিল্টারে এটা সুনির্দিষ্ট করুন। ভ্যালুটি যেটা আপনি এই এলিমেন্টে সুনির্দিষ্ট করেছেন তা অবশ্যই একশনের জন্য পূর্ণ স্ট্রিং নেম হবে, এপিআই কনসটেন্ট এর পরিবর্তে (নীচের উদাহরণ দেখুন)

ডাটা

ইনটেন্টের সাথে সংযুক্ত ডাটার একটি বর্ণনা।

এলিমেন্ট দিয়ে আপনার ইনটেন্ট ফিল্টারে এটা সুনির্দিষ্ট করুন। এই এলিমেন্টে এক বা একাধিক এট্রিবিউট ব্যবহার করতে, আপনি শুধু একটি MIME টাইপ, শুধু একটি ইউআরআই প্রিফিক্স, শুধু একটি ইউআরআই স্কিম সুনির্দিষ্ট করতে পারেন বা এগুলার সমন্বয় এবং অন্যান্য যা গ্রহন করা ডাটা টাইপ নির্দেশ করে।

নোট: যদি ডাটা Uri সম্পর্কে আপনার সুনির্দিষ্ট ডিক্লেয়ারের দরকার না হয় (যেমন যখন আপনার একটিভিটি অন্য ধরনের “এক্সট্রা” ডাটা হ্যান্ডল করে, একটি ইউআরআই এর পরিবর্তে), আপনার একটিভিটির হ্যান্ডেল করা ডাটার ধরনের ডিক্লেয়ার করতে আপনার শুধু android:mimeType এট্রিবিউটটি সুনির্দিষ্ট করা উচিত, যেমন *text/plain* or *image/jpeg*

ক্যাটাগরি

ইনটেন্ট পরিচালনা করা একটিভিটির বর্ণনা করতে একটা অতিরিক্ত উপায় সরবরাহ করুন, সাধারনত ইউজার চিহ্ন বা লোকেশন যেখান থেকে এটা শুরু করেছিল। সেখানে সিস্টেম দ্বারা সাপোর্ট পাওয়া বিভিন্ন ক্যাটাগরি আছে, কিন্তু অধিকাংশ খুব কমই ব্যবহার হয়। যাহোক সকল ইমপ্লিসিট ইনটেন্ট বাই ডিফল্ট CATEGORY DEFAULT দিয়ে সংজ্ঞায়িত।

এটা < category>এলিমেন্ট দিয়ে আপনার ইনটেন্ট ফিল্টারে চিহ্নিত করুন।

আপনার ইনটেন্ট ফিল্টারে, < intent-filter>এলিমেন্টে থাকা সমগোত্রিয় এক্সএমএল এলিমেন্ট দিয়ে তাদের প্রতিটা ডিক্লেয়ার করার মাধ্যমে আপনার একটিভিটি কোন মানদণ্ড গ্রহণ করে তা আপনি

ডিপ্লোয়ার করতে পারেন।

উদাহরণস্বরূপ, এখানে একটি ইনটেন্ট ফিল্টার সহ একটি একটিভিটি যা ACTION SEND ইনটেন্ট পরিচালিত করে যখন ডাটা টাইপ হয়, টেক্সট বা ইমেজ হয়:

```
<activity android:name="ShareActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
    <data android:mimeType="image/*"/>
  </intent-filter>
</activity>
```

প্রতিটা প্রবেশোদ্যত ইনটেন্ট শুধুমাত্র একটি একশন এবং একটি ডাটা টাইপ সুনির্দিষ্ট করে, কিন্ত প্রতিটা < intent-filter> এর মধ্যে < action>, < category> এবং < data>এলিমেন্টের মাল্টিপল ইনসটেন্স করতে এটা ঠিক আছে।

যদি একশন এবং ডাটার কোন জোড়া তাদের আচরনে পারস্পরিকভাবে খাপ না খায়, কোন একশন গ্রহণযোগ্য যখন কোন ডাটা টাইপের সাথে জোড়া হবে সেটা কোনটা হবে তা নির্দিষ্ট করতে আপনার আলাদা ইনটেন্ট তৈরী করা উচিত।

উদাহরণস্বরূপ, মনে করুন আপনার একটিভিটি ACTION SEND এবং ACTION SENDTO ইনটেন্টের উভয়ের জন্য টেক্সট এবং ইমেজ উভয়ই পরিচালনা করে। এই ক্ষেত্রে আপনাকে অবশ্যই দুইটা একশনের জন্যে দুইটা আলাদা ইনটেন্ট নির্ধারন করতে হবে কারন send বা sendto ইউআরআই স্কিম ব্যবহার করে গ্রাহকের ঠিকানা সুনির্দিষ্ট করতে একটি ACTION SENDTO ইনটেন্টের অবশ্যই ডাটা Uriব্যবহার করা। উদাহরণস্বরূপ:

```
<activity android:name="ShareActivity">
  <!-- filter for sending text; accepts SENDTO action with sms URI schemes -->
  <intent-filter>
    <action android:name="android.intent.action.SENDTO"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:scheme="sms" />
    <data android:scheme="smsto" />
  </intent-filter>
  <!-- filter for sending text or images; accepts SEND action and text or image data -->
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="image/*"/>
    <data android:mimeType="text/plain"/>
  </intent-filter>
</activity>
```

নোট: ইমপ্লিসিট ইনটেন্ট গ্রহণ করার জন্য, আপনাকে অবশ্যই ইনটেন্ট ফিল্টারে CATEGORY DEFAULT ক্যাটাগরি অন্তর্ভুক্ত করতে হবে। মেথড startActivity()এবং startActivityForResult() সকল ইনটেন্ট এর ব্যবহার করে যেন তারা CATEGORY DEFAULT ক্যাটাগরি ধারণ করে। আপনি যদি এটা ডিপ্লোয়ার না করেন, কোন ইমপ্লিসিট ইনটেন্ট আপনার একটিভিটি নির্ধারন করবে না।

ACTION SEND ইনটেন্ট যা সামাজিক শেয়ার আচরণ সেন্ড এবং রিসিভ করে সে সম্পর্কে আরও

তথ্যের জন্য, Receiving Simple Data from Other Apps সম্পর্কে অনুশীলনীট দেখুন।

আপনার একটিভিটিতে ইনটেন্টটি ব্যবস্থাপনা করা

আপনার একটিভিটিতে কি একশন নেয়া যায় ঠিক করার জন্য, আপনি Intent পড়তে পারেন যা এটা শুরু করতে ব্যবহৃত হয়।

যখন আপনার একটিভিটি শুরু করে, Intent কে উদ্ধার করতে getIntent() কল করুন যা একটিভিটি শুরু করবে। একটিভিটির লাইফসাইকেল কালের যে কোন সময়ে আপনি এটা করতে পারেন, কিন্ত সাধারনত আপনার এটা করা উচিত কলব্যাকের প্রথম দিকটায় যেমন onCreate() বা onStart()।

উদাহরণস্বরূপ:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.main);

    // Get the intent that started this activity
    Intent intent = getIntent();
    Uri data = intent.getData();

    // Figure out what to do based on the intent type
    if (intent.getType().indexOf("image/") != -1) {
        // Handle intents with image data ...
    } else if (intent.getType().equals("text/plain")) {
        // Handle intents with text ...
    }
}
```


একটি রেজাল্ট ফেরত নিয়ে আসা

যদি আপনি একটিভিটিতে একটি রেজাল্ট ফেরত চান যা আপনারটা আবাহন করে, রেজাল্ট কোড এবং রেজাল্ট Intent সুনির্দিষ্ট করতে শুধুমাত্র setResult()কল করুন। যখন আপনার অপারেশন শেষ হবে এবং ইউজারকে মূল একটিভিটিতে ফিরে যেতে হবে, আপনার একটিভিটি বন্ধ করতে (এবং ধ্বংস করতে) finish()কল করুন। উদাহরণস্বরূপ: // Create intent to deliver some kind of result data
Intent result = new Intent("com.example.RESULT_ACTION", Uri.parse("content://result_uri");
setResult(Activity.RESULT_OK, result); finish();

আপনাকে অবশ্যই সবসময় রেজাল্ট দিয়ে রেজাল্ট কোড সুনির্দিষ্ট করতে হবে। সাধারণত, হয় এটা RESULT OK বা RESULT CANCELED। এরপর আপনি প্রয়োজন অনুসারে একটা Intent দিয়ে অতিরিক্ত ডাটা সরবরাহ করতে পারেন।

নোট: রেজাল্টটি RESULT CANCELED এ বাই ডিফল্ট সেট হয়েছে। তাই, একশন শেষ করার পূর্বে বা আপনি রেজাল্ট সেট করার পূর্বে যদি ইউজার ব্যাক বাটন চাপে, মূল একটিভিটি “ক্যানসেল” রেজাল্ট রিসিভ করে।

আপনার যদি কেবল একটি ইনটিজার ফেরত আনার দরকার হয় যা কিছু রেজাল্ট অপশনের একটি নির্দেশ করে, আপনি ০ এর চেয়ে বেশী যে কোন ভ্যালুতে রেজাল্ট কোড সেট করতে পারেন। যদি আপনি একটি ইনটিজার ডেলিভারি দিতে রেজাল্ট কোড ব্যবহার করেন এবং আপনার Intent অন্তর্ভুক্ত করার কোন দরকার না থাকে, আপনি setResult()কল করতে পারেন এবং শুধু একটি রেজাল্ট কোড পাস করতে পারেন। উদাহরণস্বরূপ:

```
setResult(RESULT_COLOR_RED);  
finish();
```

এক্ষেত্রে, এখানে শুধু অল্প কিছু সম্ভাব্য রেজাল্ট হতে পারে, ফলে রেজাল্ট কোডটি হচ্ছে একটি কাছাকাছি সুনির্দিষ্ট ইনটিজার (০ এর চেয়ে বেশী)। এটা ভালোবাবে কাজ করবে যখন আপনার নিজস্ব অ্যাপে একটি একটিভিটিতে রেজাল্ট ফেরত আনছেন, কারন একটিভিটিটা যেটা রেজাল্ট রিসিভ করছে রেজাল্ট কোডের ভ্যালু নির্ধারন করতে পাবলিক কনসট্যান্টকে রেফারেন্স করতে পারে।

নোট: আপনার একটিভিটি কি startActivity()নাকি startActivityForResult()দিয়ে চলছে তা চেক প্রয়োজন নেই। শুধু setResult()কল করুন যদি ইনটেন্ট যা আপনার একটিভিটি শুরু করেছে একটা রেজাল্ট আশা করতে পারে। যদি প্রথমিকভাবে শুরু করা একটিভিটি startActivityForResult()কল করে থাকে, তখন সিস্টেমটি এটা ডেলিভারি দেয়, রেজাল্ট যেটা আপনি setResult()এ সাপ্লাই করেছেন; অন্যথায় রেজাল্ট উপেক্ষিত হয়।

কনটেন্ট শেয়ারিং সহ অ্যাপস তৈরী করুন

(<http://developer.android.com/training/building-content-sharing.html>)

এই অনুশীলনী আপনাকে শেখাবে কীভাবে একটি অ্যাপ তৈরী করা যায় যা অ্যাপস এবং ডিভাইসের মধ্যে ডাটা শেয়ার করবে।

১. সাধারণ ডাটা শেয়ার করা

অন্য অ্যাপের সাথে তথ্য আদান প্রদান করার মাধ্যমে আপনার অ্যাপকে পরবর্তী লেভেলে কীভাবে যোগাযোগ করতে হয়, কীভাবে ফিরতি তথ্য গ্রহণ করতে হয়, এবং কীভাবে ইউজার কনটেন্ট দিয়ে শেয়ার একশন সম্পাদন করতে একটি সহজ এবং মাপযোগ্য উপায় সরবরাহ করা যায়।

১. অন্য অ্যাপে সাধারণ ডাটা সেভ করা
২. অন্য অ্যাপ থেকে সাধারণ ডাটা রিসিভ করা
৩. সহজ শেয়ার একশন এ্যাড করুন

২. ফাইল শেয়ার করা

একটি কনটেন্ট ইউআরআই এবং অস্থায়ী প্রবেশ অনুমতি ব্যবহার করে আপনার অ্যাপের সাথে যুক্ত ফাইলে কীভাবে একটা নিরাপদ প্রবেশ সরবরাহ করা যায়।

১. ফাইল শেয়ার করা সেটআপ করা
২. একটি ফাইল শেয়ার
৩. শেয়ার করা ফাইল রিকোয়েস্ট করা
৪. ফাইল তথ্য পুনরুদ্ধার

৩. NFC দিয়ে ফাইল শেয়ার

NFC অ্যান্ড্রয়েড বিম ফিচার ব্যবহার করে কীভাবে ডিভাইসের মধ্যে ফাইল স্থানান্তর করা যায়।

১. অন্য ডিভাইসে ফাইল সেভ করা
২. অন্য ডিভাইস থেকে ফাইল গ্রহণ

সাধারণ ডাটা শেয়ার করা

(<http://developer.android.com/training/sharing/index.html>)

অ্যান্ড্রয়েড অ্যাপলিকেশনের একটি বড় বৈশিষ্ট্য হচ্ছে তাদের একে অপরের সাথে যোগাযোগ করতে পারা এবং একীভূত হওয়ার ক্ষমতা। কেন একটা কার্যকারিতার পুনরুদ্ধার করবেন যা আপনার অ্যাপলিকেশনের মধ্যে নেই যখন এটা অন্য একটা অ্যাপলিকেশনের মধ্যে ইতিমধ্যে বিদ্য রয়েছে?

এই ক্লাস Intent এপিআই এবং ActionProvider অবজেক্ট ব্যবহার করে অ্যাপলিকেশনের মধ্যে সাধারণ ডাটা সেন্ড এবং রিসিভ করার কিছু প্রচলিত উপায় অন্তর্ভুক্ত করে।

অনুশীলনীসমূহ

অন্য অ্যাপে সাধারণ ডাটা সেভ করা

শিখুন ইনটেন্ট দিয়ে অন্য অ্যাপলিকেশনে টেক্সট এবং বাইনারি সেভ করতে সমর্থ করতে কীভাবে আপনার অ্যাপলিকেশন সেট করতে হয়।

অন্য অ্যাপ থেকে সাধারণ ডাটা রিসিভ করা

শিখুন ইনটেন্ট থেকে টেক্সট এবং বাইনারি ডাটা রিসিভ করতে কীভাবে আপনার অ্যাপলিকেশন সেট করতে হয়।

সহজ শেয়ার একশন এ্যাড করুন

শিখুন আপনার একশন বারে কীভাবে একটি “শেয়ার” একশন আইটেম এ্যাড করতে হয়।

অন্য অ্যাপে সাধারণ ডাটা সেন্ড করা

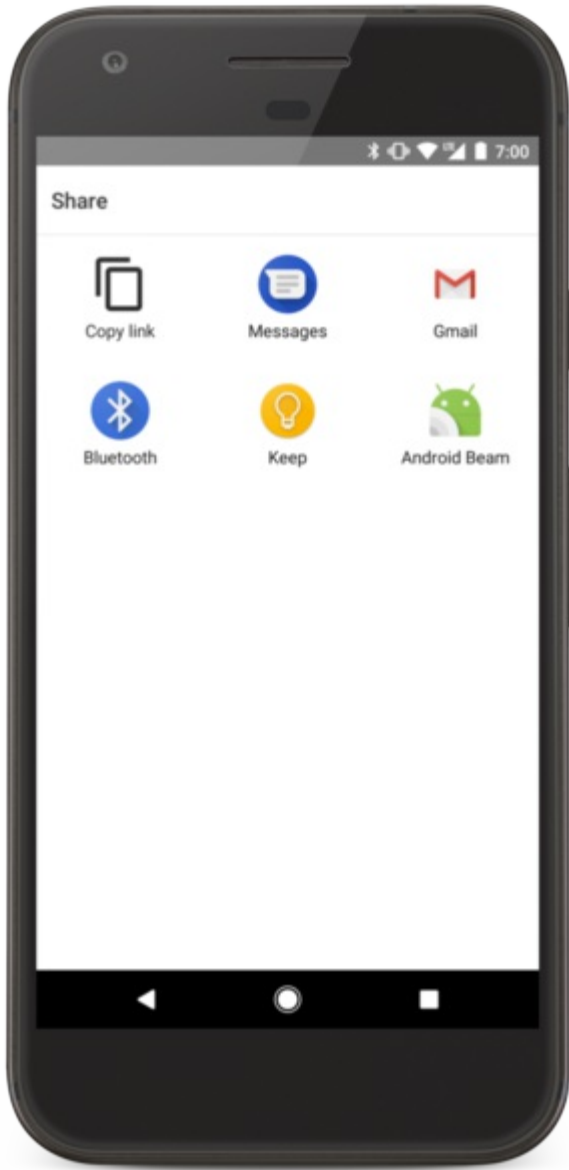
(<http://developer.android.com/training/sharing/send.html>)

যখন আপনি একটি ইনটেন্ট গঠন করেন, আপনাকে অবশ্যই একশনকে সুনির্দিষ্ট করতে হবে আপনি যদি ইনটেন্টটি সক্রিয় করতে চান ("trigger")। অ্যান্ড্রয়েড কয়েকটি একশনকে নির্ধারণ করে, যার মধ্যে রয়েছে ACTION_SEND যা নির্দেশ করে যে ইনটেন্ট একটি একটিভিটি থেকে আরেকটিতে ডাটা সেন্ড করছে। অন্য একটিভিটিতে ডাটা সেন্ড করতে, আপনাদের সকলকে ডাটা এবং এর টাইপকে সুনির্দিষ্ট করা দরকার, সিস্টেম সর্বগ্রহী রিসিভিং একটিভিটি নির্ধারণ করে এবং ইউজারের কাছে তা প্রদর্শন করে (যদি সেখানে বহুমুখি অপশন থাকে) অথবা তাৎক্ষণিকভাবে শুরু করে (যদি শুধু একটা অপশন থাকে)। একইভাবে, আপনি ডাটা টাইপ প্রচার করতে পারেন যে আপনার একটিভিটি আপনার মেসেজবক্স তাদের সুনির্দিষ্ট করার মাধ্যমে অন্য অ্যাপ থেকে রিসিভ করাকে সাপোর্ট করে।

ইনটেন্ট দিয়ে অ্যাপলিকেশনের মধ্যে ডাটা সেন্ড এবং রিসিভ করাটা হচ্ছে কন্টেন্ট এর সামাজিক শেয়ারিং এর জন্য সবচেয়ে প্রচলিত ব্যবহার। ইনটেন্ট ইউজারকে দ্রুত এবং সহজে তথ্য শেয়ার করতে দেয়, তাদের পছন্দের অ্যাপলিকেশন ব্যবহার করে।

নোট: একটি ActionBar এ একটি শেয়ার একশন আইটেম যুক্ত করার সবচেয়ে ভালো উপায় হচ্ছে ShareActionProvider ব্যবহার করা যা এপিআই লেভেল ১৪ তে পাওয়া যাবে। Adding an Easy Share Action সম্পর্কিত অনুশীলনীতে ShareActionProvider আলোচিত হয়েছে।

টেক্সট কনটেন্ট সেন্ড করা



ফিগার ১. একটি হ্যান্ডসেটে ACTION SEND ইনটেন্টে চুজারের একটি স্ক্রিনশট।

ACTION SEND একশনের সবচেয়ে সোজাসাপটা এবং সবচেয়ে বেশী ব্যবহার হচ্ছে একটি একটিভিটি থেকে আরেকটিতে টেক্সট কনটেন্ট সেন্ড করা। উদাহরণস্বরূপ, বিল্ট-ইন ব্রউজার অ্যাপ যেকোন অ্যাপলিকেশন দিয়ে টেক্সট হিসাবে চলতি-প্রদর্শিত পেজের ইউআরএল শেয়ার করতে পারে। এটা ইমাইল বা সামাজিক যোগাযোগের মাধ্যমে বন্ধুদের সাথে একটি আর্টিকেল বা ওয়েবসাইট শেয়ার করার জন্য উপকারী। এই ধরনের শেয়ার বাস্তায়ন করতে এখানে কোডটি আছে:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");
startActivity(sendIntent);
```

যদি একটা ফিল্টার সহ একটা ইনস্টল করা অ্যাপলিকেশন থাকে যা ACTION_SEND এবং MIME টাইপ টেক্সট/প্লেইন ম্যাচ করে, অ্যান্ড্রয়েড সিস্টেম এটা রান করে; যদি একের অধিক অ্যাপলিকেশন ম্যাচ করে, সিস্টেম একাধিক বোধক ডায়ালগ(একটি চুজার) প্রদর্শন করে যা ইউজারকে একটি অ্যাপ পছন্দ করতে দেয়। যদি আপনি ইনটেন্টের জন্য Intent.createChooser() কল করেন, অ্যান্ড্রয়েড সবসময় চুজারটি প্রদর্শন করে। এটার কিছু সুবিধা আছে:

- এমনকি যদি ইউজার পূর্বেই ইনটেন্টের জন্য একটি ডিফল্ট একশন নির্বাচিত করে, চুজার তখনও প্রদর্শিত করে।
- যদি কোন অ্যাপলিকেশন ম্যাচ না করে, অ্যান্ড্রয়েড একটি সিস্টেম ইমেজ প্রদর্শন করে।
- আপনি চুজার ডায়ালগের জন্য একটা টাইটেল নির্দিষ্ট করে দিতে পারেন।

এটা একটা আপডেট কোড:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");
startActivity(Intent.createChooser(sendIntent, getResources().getText(R.string.send_to)));
```

ফলাফল ডায়ালগ ফিগার ১ এ দেখানো হয়েছে।

ঐচ্ছিকভাবে, আপনি ইনটেন্টের জন্য কিছু স্ট্যান্ডার্ড এক্সট্রা সেট করতে পারেন: EXTRA_EMAIL, EXTRA_CC, EXTRA_BCC, EXTRA_SUBJECT। কিনুত যদি রিসিভ করা অ্যাপলিকেশন তাদের ব্যবহার করার মতো করে ডিজাইন করা না থাকে, তাহলে কিছুই ঘটবে না। আপনি কাস্টম এক্সট্রাও ব্যবহার করতে পারেন, কিনুত এখানে কোন প্রভাব পরবে না যদি না রিসিভ করা অ্যাপলিকেশন এটা বুঝতে না পারে। স্বাভাবিকভাবে আপনি রিসিভ করা অ্যাপলিকেশনের নিজের দ্বারা নির্ধারিত কাস্টম এক্সট্রা ব্যবহার করতে পারেন।

নোট: কিছু ই-মেইল অ্যাপলিকেশন, যেমন জিমেইল, এক্সট্রা এর জন্য EXTRA_EMAIL এবং EXTRA_CC এর মতো একটি String[] এর প্রত্যাশা করে, আপনার ইনটেন্ট এটা যুক্ত করতে putExtra (String, String[]) ব্যবহার করে।

বাইনারি কনটেন্ট সেন্ড করুন

বাইনারি ডাটা শেয়ার হয় যথাযথ MIME টাইপ সেটিং দ্বারা একত্রিত ACTION SEND একশন ব্যবহার করে এবং EXTRA STREAM নামে একটি এক্সট্রার মধ্যে ডাটাতে ইউআরআই সন্নিবেশ করে। এটা সাধারনত একটি ইমেজ শেয়ার করার কাজে ব্যবহার করা হয় কিনুত যে কোন ধরনের বাইনারি কনটেন্ট শেয়ার করতে ব্যবহার করা যেতে পারে:

```
Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND);
shareIntent.putExtra(Intent.EXTRA_STREAM, uriToImage);
shareIntent.setType("image/jpeg");
startActivity(Intent.createChooser(shareIntent, getResources().getText(R.string.send_to)));
```

নিম্নোক্ত বিষয়গুলো নোট করুন:

- আপনি "/" এর MIME টাইপ ব্যবহার করতে পারেন, কিনুত এটা শুধু একটিভিটি ম্যচ করবে যা জেনেরিক ডাটা স্ট্রিম পরিচালনা করতে সক্ষম।
- রিসিভ করা অ্যাপের ডাটাতে প্রবেশ করতে অনুমতি দরকার যা Uri নির্দেশ করে। এটা করার প্রস্তাবিত উপায় হচ্ছে:
 - আপনার নিজস্ব ContentProvider এ ডাটা স্টোর করুন, এটা নিশ্চিত করতে যে আপনার প্রভাইডারে অন্য অ্যাপের প্রবেশের সঠিক অনুমতি আছে। প্রবেশগম্যতা প্রদানের পছন্দসই মেকানিজম হচ্ছে per-URI permissions ব্যবহার করা যা অস্থায়ী এবং শুধুমাত্র রিসিভ করা অ্যাপলিকেশনে প্রবেশগম্যতাকে অনুমোদন করে। এটার মতো করে একটি ContentProvider তৈরী করার একটা সহজ উপায় হচ্ছে FileProvider হেল্পার ক্লাস ব্যবহার করা।
 - সিস্টেম MediaStore ব্যবহার করুন। MediaStore প্রাথমিকভাবে ভিডিও, অডিও এবং ইমেজ MIME টাইপ লক্ষ্য করে করা, কিনুত অ্যান্ড্রয়েড ৩.০ (এপিআই লেভেল ১১) দিয়ে শুরু করলে এটা নন-মিডিয়া টাইপ (আরও জানতে দেখুন MediaStore.Files) স্টোর করতে পারে। ফাইল scanFile() ব্যবহার করে MediaStore এ প্রবেশ করতে পারে, যেটার পরে শেয়ারের জন্য উপযুক্ত একটি content:// স্টাইল Uri কে প্রভাইডকৃত onScanCompleted() কলব্যাকে পাস করে দেয়। উল্লেখ্য যে একবার সিস্টেম MediaStore যুক্ত হলে কনটেন্টটি ডিভাইসের যে কোন অ্যাপে প্রবেশগম্য হবে।

কনটেন্ট এর মাল্টিপল অংশ সেন্ড করুন

কনটেন্ট এর মাল্টিপল অংশ শেয়ার করতে, কনটেন্টে নির্দেশ করা ইউআরআই (URIs) এর একটি তালিকা সহ ACTION_SEND_MULTIPLE একশন ব্যবহার করুন। কনটেন্টের মিশ্রণ যা আপনি ব্যবহার করছেন সেই অনুসারে MIME টাইপের তারতম্য হয়। উদাহরণস্বরূপ, যদি আপনি ৩ JPEG ইমেজ শেয়ার করতে চান, টাইপ তখনও "image/jpeg"। একটি ইমেজ টাইপের মিক্সচারের জন্য, একটি একটিভিটি যা যে কোন ধরনের ইমেজ পরিচালিত করে ঐ একটিভিটি ম্যাচ করতে এটা [image/*] হওয়া উচিত।

আপনার শুধুমাত্র [/*/*] ব্যবহার করা উচিত যদি আপনি অনেকগুলো ভিন্ন টাইপ শেয়ার করে থাকেন। যেমনভাবে পূর্বে বর্ণনা করা হয়েছে , আপনার ডাটাকে বিশ্লেষণ এবং প্রক্রিয়া করাটা, রিসিভ করা অ্যাপলিকেশনের উপর বর্তায়। উদাহরণস্বরূপ:

```
ArrayList<Uri> imageUris = new ArrayList<Uri>();
imageUris.add(imageUri1); // Add your image URIs here
imageUris.add(imageUri2);

Intent shareIntent = new Intent();
shareIntent.setAction(Intent.ACTION_SEND_MULTIPLE);
shareIntent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, imageUris);
shareIntent.setType("image/*");
startActivity(Intent.createChooser(shareIntent, "Share images to.."));
```

পূর্বের মতো, নিশ্চিত করুন যে প্রদত্ত URIs ডাটার প্রতি নির্দেশ করে যাতে একটি রিসিভ করা অ্যাপলিকেশন প্রবেশগম্য হতে পারে।

অন্য অ্যাপ থেকে সাধারণ ডাটা রিসিভ করা

(<http://developer.android.com/training/sharing/receive.html>)

ঠিক যেমন আপনার অ্যাপলিকেশন অন্য অ্যাপলিকেশনে ডাটা সেভ করতে পারে, তেমনিভাবে এটা সহজেই অন্য অ্যাপলিকেশন থেকে পাঠানো ডাটা রিসিভও করতে পারে। কীভাবে ইউজার আপনার অ্যাপলিকেশনের সাথে পারস্পরিক ক্রিয়া করবে এবং অন্য অ্যাপলিকেশন থেকে কি ধরনের ডাটা আপনি রিসিভ করতে পারেন সে সম্পর্কে চিন্তা করুন। উদাহরণস্বরূপ, একটি সামাজিক যোগাযোগের অ্যাপলিকেশন অন্য অ্যাপলিকেশন থেকে টেক্সট কনটেন্ট, একটা কৌতুহলোদ্দীপক ওয়েব ইউআরএল রিসিভ করতে আগ্রহী হতে পারে। গুগল প্লাস অ্যান্ড্রয়েড অ্যাপলিকেশন (Google+ Android application: <https://play.google.com/store/apps/details?id=com.google.android.apps.plus>) টেক্সট এবং একক বা বহুমুখি ইমেজ উভয়ই গ্রহণ করে।

আপনার মেনিফেস্ট আপডেট করা

ইনটেন্ট ফিল্টার সিস্টেমকে অবহিত করে যে একটি ইনটেন্ট অ্যাপলিকেশন কম্পোনেন্ট গ্রহণ করতে ইচ্ছুক। একইভাবে অন্য অ্যাপে সাধারণ ডাটা সেন্ড করুন (Sending Simple Data to Other Apps) অনুশীলনীর মতো ACTION SEND দিয়ে কীভাবে আপনি একটি ইনটেন্ট তৈরী করেন, এই একশন দিয়ে ইনটেন্ট গ্রহণ করতে পারার জন্য আপনি ইনটেন্ট ফিল্টার তৈরী করেন। < intent-filter>এলিমেন্ট ব্যবহার করে আপনি আপনার মেনিফেস্ট একটি ইনটেন্ট ফিল্টার নির্ধারণ করেন। উদাহরণস্বরূপ, যদি আপনার অ্যাপলিকেশন টেক্সট কনটেন্ট রিসিভ করা চালিত করে, যে কোন ধরনের একটি একক ইমেজ বা যে কোন ধরনের বহুমুখি ইমেজ, আপনার মেনিফেস্ট এরকম দেখাবে:

```
<activity android:name=".ui.MyActivity" >
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="image/*" />
  </intent-filter>
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
  </intent-filter>
  <intent-filter>
    <action android:name="android.intent.action.SEND_MULTIPLE" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="image/*" />
  </intent-filter>
</activity>
```

নোট: ইনটেন্ট ফিল্টার এবং ইনটেন্ট রেজ্যুলেশনের সম্পর্কে আরও জানতে Intents and Intent Filters (<http://developer.android.com/guide/components/intents-filters.html#ifs>) দেখুন।

যখন অন্য অ্যাপলিকেশন একটি ইনটেন্ট গঠন করে এবং startActivity()র প্রতি এটা পাস করার মাধ্যমে এই বিষয়গুলোর কোন কিছু শেয়ার করার চেষ্টা করে, আপনার অ্যাপলিকেশন ইনটেন্ট চুজারের মধ্যে একটি অপশন হিসাবে তালিকাভুক্ত হবে। যদি ইউজার আপনার অ্যাপলিকেশন বেছে নেয়, সংশ্লিষ্ট একটিভিটি (উপরোক্ত উদাহরনের মধ্যে .ui.MyActivity) শুরু হবে। তারপর আপনার কোড এবং ইউজার ইন্টারফেসের মধ্যে থেকে কনটেন্টটি যথাযথভাবে চালিত করাটা আপনার উপর নির্ভর করে।

ইনকামিং কনটেন্ট ব্যবস্থাপনা করা

একটি Intent দ্বারা ডেলিভারি দেওয়া কনটেন্ট চালনা করতে, Intent অবজেক্ট পেতে getIntent() কল করে শুরু করুন। আপনার যদি পূর্বেই অবজেক্টটি থেকে থাকে, পরবর্তীতে কি করতে হবে তা নির্ধারণ করতে এটার কনটেন্ট পরীক্ষা করতে পারেন। মনে রাখবেন যে যদি এই একটিভিটি সিস্টেমটির অন্য অংশ থেকে শুরু করতে পারে, যেমন লঞ্চার, তখন এটাকে আপনার বিবেচনায় আনার দরকার হতে পারে যখন ইনটেন্টটি পরীক্ষা করা হয়।

```
void onCreate (Bundle savedInstanceState) {  
    ...  
    // Get intent, action and MIME type  
    Intent intent = getIntent();  
    String action = intent.getAction();  
    String type = intent.getType();  
  
    if (Intent.ACTION_SEND.equals(action) && type != null) {  
        if ("text/plain".equals(type)) {  
            handleSendText(intent); // Handle text being sent  
        } else if (type.startsWith("image/")) {  
            handleSendImage(intent); // Handle single image being sent  
        }  
    } else if (Intent.ACTION_SEND_MULTIPLE.equals(action) && type != null) {  
        if (type.startsWith("image/")) {  
            handleSendMultipleImages(intent); // Handle multiple images being sent  
        }  
    } else {  
        // Handle other intents, such as being started from the home screen  
    }  
    ...  
}  
  
void handleSendText(Intent intent) {  
    String sharedText = intent.getStringExtra(Intent.EXTRA_TEXT);  
    if (sharedText != null) {  
        // Update UI to reflect text being shared  
    }  
}  
  
void handleSendImage(Intent intent) {  
    Uri imageUri = (Uri) intent.getParcelableExtra(Intent.EXTRA_STREAM);  
    if (imageUri != null) {  
        // Update UI to reflect image being shared  
    }  
}  
  
void handleSendMultipleImages(Intent intent) {  
    ArrayList<Uri> imageUris = intent.getParcelableArrayListExtra(Intent.EXTRA_STREAM);  
    if (imageUris != null) {  
        // Update UI to reflect multiple images being shared  
    }  
}
```

সতর্কতা: ইনকামিং ডাটা চেক করতে অতিরিক্ত যত্নবান হোন, আপনি কখনই জানতে পারবেন না অন্য অ্যাপলিকেশন আপনাকে কি সেন্ড করতে পারে। উদাহরণস্বরূপ, ভুল MIME টাইপ সেট হতে পারে, অথবা অতিরিক্ত বড় আকারে ইমেজ সেট হতে পারে। প্রধান ("UI") থ্রেডের মধ্যে করার

চেয়ে একটি আলাদা থ্রেডের মধ্যে বাইনারি ডাটা প্রসেস করার বিষয়টাও মনে রাখবেন।

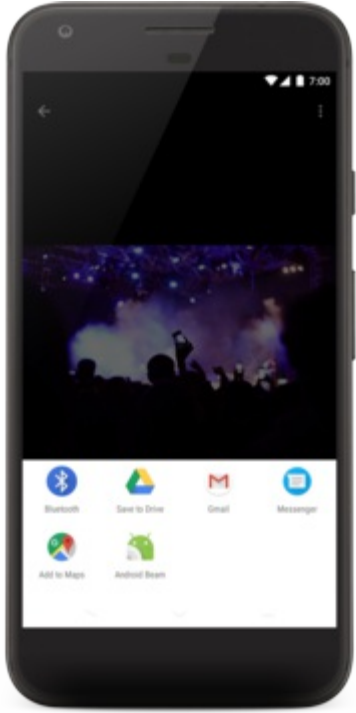
ইউজার ইন্টারফেস আপডেট করাটা একটি EditText পূর্ণ করার মতোই সহজ হতে পারে, অথবা এটা একটি ইমেজে একটা আকর্ষণীয় ফটো ফিল্টার প্রয়োগ করার মতো আরও জটিলও হতে পারে। এটা আসলেই আপনার অ্যাপলিকেশনে সুনির্দিষ্ট যে পরবর্তীতে কি হবে।

একটি সহজ শেয়ার একশন যুক্ত করা

(<http://developer.android.com/training/sharing/shareaction.html>)

অ্যান্ড্রয়েড ৪.০ (এপিআই লেভেল ১৪) এর মধ্যে ActionProvider এর প্রবর্তন দ্বারা আপনার ActionBar এর মধ্যে কার্যকরী এবং ইউজার বান্ধব শেয়ার একশন বাস্তবায়ন করাটা এমনকি আরও সহজ করে তৈরী। একটি ActionProvider, একবার একশনবারের মধ্যে মেনু আইটেমে সংযুক্ত হলে, ওই আইটেমের উপস্থিতি এবং আচরণ উভয়কেই চালনা করে। ShareActionProvider এর ক্ষেত্রে, আপনি একটা শেয়ার ইনটেন্ট সরবরাহ করেন এবং এটা বাকীটা করে।

নোট: এপিআই লেভেল ১৪ এবং এর চেয়ে উন্নত সংস্করণ দিয়ে শুরু করা কাজে ShareActionProvider বিদ্যমান আছে



ফিগার ১. গ্যালারি অ্যাপে ShareActionProvider ।

মেনু ডিক্লেয়ারেশন আপডেট করা

ShareActionProviders দিয়ে শুরু করতে, আপনার menu resource ফাইলের মধ্যে সংশ্লিষ্ট < item> এর জন্য android:actionProviderClass এট্রিবিউট নির্ধারণ করুন:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/menu_item_share"
        android:showAsAction="ifRoom"
        android:title="Share"
        android:actionProviderClass=
            "android.widget.ShareActionProvider" />
    ...
</menu>
```

এটা ShareActionProvider এর প্রতি আইটেমের উপস্থিতি এবং কার্যাবলীর দায়িত্ব হস্তান্তর করে। কিনুত আপনার সরবরাহকারীকে বলা দরকার যে আপনি কী শেয়ার করতে চান।

শেয়ার ইনটেন্ট সেট করা

ShareActionProvider কে কাজ করাতে, আপনাকে অবশ্যই এটাকে একটি শেয়ার ইনটেন্ট সরবরাহ করতে হবে। EXTRA TEXT এবং EXTRA STREAM এর মতো এক্সট্রার সাহায্যে একশন ACTION SEND এবং অতিরিক্ত ডাটা সেট দিয়ে (Sending Simple Data to Other Apps) অনুশীলনী তে যেভাবে আলোচনা করা হয়েছে, এই শেয়ার ইনটেন্টের সেরকম হওয়া উচিত। একটি শেয়ার ইনটেন্টকে নিযুক্ত করতে, প্রথমে সংশ্লিষ্ট MenuItem খুঁজে বের করুন যখন আপনার Activity বা Fragment এর মধ্যে আপনার মেনু রিসোর্স প্রসারিত করে। পরবর্তীতে ShareActionProvider এর একটি ইনসটেন্স উদ্ধার করতে MenuItem.getActionProvider() কল করুন। ওই একশন আইটেমের সাথে যুক্ত শেয়ার আইটেম আপডেট করতে setShareIntent() ব্যবহার করুন। এখানে একটি উদাহরণ আছে:

```
private ShareActionProvider mShareActionProvider;
...

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate menu resource file.
    getMenuInflater().inflate(R.menu.share_menu, menu);

    // Locate MenuItem with ShareActionProvider
    MenuItem item = menu.findItem(R.id.menu_item_share);

    // Fetch and store ShareActionProvider
    mShareActionProvider = (ShareActionProvider) item.getActionProvider();

    // Return true to display menu
    return true;
}

// Call to update the share intent
private void setShareIntent(Intent shareIntent) {
    if (mShareActionProvider != null) {
        mShareActionProvider.setShareIntent(shareIntent);
    }
}
```

আপনার শুধু আপনার মেনু তৈরীর সময় একবার শেয়ার ইনটেন্ট সেট করার প্রয়োজন হতে পারে, অথবা আপনি এটা সেট করতে চাইতে পারেন এবং ইউজার ইন্টারফেস পরিবর্তন হিসাবে এটাকে আপডেট করুন। উদাহরণস্বরূপ, যখন আপনি গ্যালারী অ্যাপে ফটো পূর্ণ স্ক্রিনে ভিউ করেন, শেয়ারিং ইনটেন্ট পরিবর্তন করে যাতে আপনি ফটোর মধ্যে ঘুরতে পারেন।

ShareActionProvider অবজেক্ট সম্পর্কিত আরও আলোচনার জন্য Action Bar গাইড দেখুন।

ফাইল শেয়ার করা

(<http://developer.android.com/training/secure-file-sharing/index.html>)

অ্যাপের প্রতিনিয়ত অন্য অ্যাপে তাদের এক বা একাধিক ফাইল দেয়ার প্রস্তাব করার দরকার হয়। উদাহরণস্বরূপ, একটি ইমেজ গ্যালারি ইমেজ এডিটরে ফাইল পাঠানোর প্রস্তাব করতে পারে, অথবা একটি ফাইল ব্যবস্থপনা অ্যাপ এক্সটার্নাল স্টোরেজের এলাকার মধ্যে ইউজারকে ফাইল কপি এবং পেস্ট করতে দিতে চাইতে পারে। একমুখি রাস্তা একটি সেন্ডিং অ্যাপ একটি ফাইল শেয়ার করতে পারে সেটা হচ্ছে প্রদত্ত রিকোয়েস্টের প্রতি রিসিভিং অ্যাপ থেকে সাড়া প্রদান।

সকল ক্ষেত্রে আপনার অ্যাপ থেকে অন্য অ্যাপে ফাইল পাঠানোর প্রস্তাবের একমাত্র নিরাপদ উপায় হচ্ছে রিসিভিং অ্যাপে ফাইলের কনটেন্ট ইউআরআই পাঠানো এবং ঐ ইউআরআই এ অস্থায়ী প্রবেশ অনুমতি দেওয়া। ইউআরআই অস্থায়ী প্রবেশ অনুমতি সহ কনটেন্ট ইউআরআই নিরাপদ কারন তারা শুধু অ্যাপে প্রয়োগ করে যা ইউআরআই রিসিভ করে, এবং তার স্বয়ংক্রিয়ভাবে শেষ হয়। একটি ফাইলের কনটেন্ট ইউআরআই উৎপাদন করার জন্য অ্যান্ড্রয়েড FileProvider কম্পোনেন্ট মেথড `getUriForFile()` সরবারহ করে।

আপনি যদি অ্যাপসের মধ্যে অল্প পরিমানে টেক্সট বা সংখ্যাচক (নিউমেরিক) ডাটা শেয়ার করতে চান, আপনার উচিত একটা Intent সেন্ড করা যা ডাটাকে ধারন করে। Intent দিয়ে কীভাবে সাধারন ডাটা সেন্ড করা যায় তা শিখতে (Sharing Simple Data) ক্লাসটি দেখুন।

এই অনুশীলনী ব্যাখ্যা করবে অ্যান্ড্রয়েড FileProvider কম্পোনেন্ট কর্তৃক উৎপাদিত কনটেন্ট ইউআরআই ব্যবহার করে আপনার অ্যাপ থেকে অন্য অ্যাপে কীভাবে নিরাপদে ফাইল শেয়ার করা যায় এবং অস্থায়ী অনুমতি যা কনটেন্ট ইউআরআই এর জন্য রিসিভিং অ্যাপকে দেয়া হয়।

অনুশীলনীসমূহ

ফাইল শেয়ার করতে অ্যাপ সেট আপ করা

শিখুন ফাইল শেয়ার করতে কীভাবে আপনার অ্যাপ সেটআপ দিতে হয়

একটি ফাইল শেয়ার করা

শিখুন ফাইলের জন্য একটি কনটেন্ট ইউআরআই উৎপাদন করার মাধ্যমে কীভাবে একটি ফাইল অন্য অ্যাপে পাঠানোর প্রস্তাব করা হয়।

শেয়ার করা ফাইল রিকোয়েস্ট করা

শিখুন কীভাবে অন্য অ্যাপ কর্তৃক শেয়ার করা একটি ফাইলকে রিকোয়েস্ট করতে হয়, ফাইলের জন্য কনটেন্ট ইউআরআই রিসিভ করতে হয় এবং ফাইলটি ওপেন করতে কনটেন্ট ইউআরআই ব্যবহার করতে হয়।

ফাইল তথ্য পুনরুদ্ধার করা

শিখুন কীভাবে একটি অ্যাপ একটি FileProvider দ্বারা উৎপাদিত কনটেন্ট ইউআরআই ব্যবহার করে ফাইল তথ্য পুনরুদ্ধার করে যার মধ্যে MIME টাইপ এবং ফাইল সাইজ অন্তর্ভুক্ত।

ফাইল শেয়ার করতে অ্যাপ সেট আপ করা

(<http://developer.android.com/training/secure-file-sharing/setup-sharing.html>)

আপনার অ্যাপ থেকে অন্য অ্যাপে একটি ফাইল নিরাপদে পাঠানোর প্রস্তাব করতে, কনটেন্টের ইউআরআই এর ফর্মে ফাইলের প্রতি একটি নিরাপদ ব্যত্য়স্থাপনা প্রস্তাব করতে আপনাকে আপনার অ্যাপকে কনফিগারেশন করতে হবে। অ্যান্ড্রয়েড FileProvider কম্পোনেন্ট ফাইলের জন্য কনটেন্ট ইউআরআই উৎপাদন করে, আপনার দ্বারা এক্সএমএল এ প্রদত্ত স্পেসিফিকেশন এ উপর ভিত্তি করে। এই অনুশীলনী আপনাকে দেখাবে কীভাবে আপনার অ্যাপে FileProvider এর ডিফল্ট বাস্তবায়ন এড করতে হয় এবং আপনার অ্যাপ থেকে অন্য অ্যাপে যে ফাইলটি প্রস্তাব করতে চান তা কীভাবে সুনির্দিষ্ট করতে হয়।

নোট: FileProvider ক্লাসটি v4 Support Library এর অংশ। আপনার অ্যাপের মধ্যে এই লাইব্রেরী অন্তর্ভুক্ত করা সম্পর্কিত তথ্যের জন্য (Support Library Setup) দেখুন।

ফাইল প্রভাইডার (প্রদানকারী) নির্দিষ্ট করুন

আপনার অ্যাপের জন্য একটি FileProvider নির্ধারন করা আপনার মেনিফেস্টের মধ্যে একটা এন্ট্রি চায়। এই এন্ট্রি উৎপন্ন কনটেন্ট ইউআরআই এ ব্যবহার করতে অথরিটি নির্দিষ্ট করে, একইভাবে একটি এক্সএমএল ফাইলের নাম যা আপনার অ্যাপ শেয়ার করতে পারা ডিরেক্টরী সুনির্দিষ্ট করে।

নিম্নোক্ত চিত্রটি আপনাকে দেখাবে কীভাবে আপনার মেনিফেস্ট `< provider>`এলিমেন্ট এ্যাড করতে হয় যা FileProvider ক্লাস, অথরিটি, এবং এক্সএমএল ফাইল নাম সুনির্দিষ্ট করে:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication"
    <application
        ...>
        <provider
            android:name="android.support.v4.content.FileProvider"
            android:authorities="com.example.myapplication.fileprovider"
            android:grantUriPermissions="true"
            android:exported="false">
            <meta-data
                android:name="android.support.FILE_PROVIDER_PATHS"
                android:resource="@xml/filepaths" />
        </provider>
        ...
    </application>
</manifest>
```

এই উদাহরনে, `android:authorities` এট্রিবিউট ইউআরআই অথরিটি সুনির্দিষ্ট করে যা আপনি FileProvider কর্তৃক উৎপন্ন কনটেন্ট ইউআরআই এর জন্য ব্যবহার করতে চাইতে পারেন। এই উদাহরনে, অথরিটি হচ্ছে `com.example.myapplication.fileprovider`। আপনার নিজস্ব অ্যাপের জন্য, অ্যাপের `android:package` ভ্যালুর অংশীভূত অথরিটি সুনির্দিষ্ট করা সাথে এটাতে স্ট্রিং `"fileprovider"` যুক্ত করে। অথরিটি ভ্যালু সম্পর্কে আরও জানতে টপিক (Content URIs) এবং `android:authorities` এট্রিবিউটের জন্য ডকুমেন্টেশন দেখুন।

`< provider>` এর `< meta-data>`চাইল্ড এলিমেন্ট এক্সএমএল ফাইলের প্রতি নির্দেশ করে যা আপনার শেয়ার করতে চাওয়া ডিরেক্টরী নির্দিষ্ট করে। `android:resource` এট্রিবিউট .ীসষ এক্সটেনশন ব্যতিত ফাইলের নাম এবং পথ। এই ফাইলের কনটেন্ট পরবর্তী শেসনে আলোচনা করা হবে।

শেয়ারযোগ্য ডিরেক্টরী নির্দিষ্টকরণ

পূর্বে আপনি যদি আপনার মেনিফেস্ট FileProvider যুক্ত করে থাকেন, আপনাকে ডিরেক্টরী নির্দিষ্ট করতে হবে যা সেই ফাইলকে ধরে রাখে যা আপনি শেয়ার করতে চান। ডিরেক্টরী নির্দিষ্ট করতে, আপনার প্রজেক্টের res/xml/ সাবডিরেক্টরীর মধ্যে ফাইল filepaths.xml তৈরী করার মাধ্যমে শুরু করুন। এই ফাইলের মধ্যে, প্রতিটা ডিরেক্টরীর জন্য একটি এক্সএমএল এলিমেন্ট যুক্ত করার মাধ্যমে ডিরেক্টরী নির্দিষ্ট করুন। নিম্নোক্ত অংশটি আপনাকে res/xml/filepaths.xml এর কনটেন্টের উদাহরণ দেখাবে। এই অংশটি আরও প্রমাণ করবে আপনার ইন্টারনাল স্টোরেজের এলাকার মধ্যে files/ ডিরেক্টরীর একটি সাবডিরেক্টরী কীভাবে শেয়ার করতে হয়:

```
<paths>
  <files-path path="images/" name="myimages" />
</paths>
```

এই উদাহরনের মধ্যে, < files-path> ট্যাগ আপনার অ্যাপের ইন্টারনাল স্টোরেজের files/ ডিরেক্টরীর মধ্যে থেকে ডিরেক্টরী শেয়ার করতে পারে। path এট্রিবিউট files/ এর images/ সাবডিরেক্টরী শেয়ার করে। name এট্রিবিউট FileProvider কে files/images/ সাবডিরেক্টরীতে ফাইলের জন্য কনটেন্ট ইউআরআই এ পাথ সেগমেন্ট myimages যুক্ত করতে বলে।

< paths> এলিমেন্টের মাল্টিপল চিলড্রেন থাকতে পারে, শেয়ার করতে এর প্রতিটা ভিন্ন ডিরেক্টরী নির্দিষ্ট করে। উপরনুত এলিমেন্ট আপনি এক্সটার্নাল স্টোরেজের ডিরেক্টরী শেয়ার করতে এলিমেন্ট ব্যবহার করতে পারেন। চাইল্ড এলিমেন্ট যা শেয়ার করা ডিরেক্টরী নির্দিষ্ট করে সে সম্পর্কে আরও জানতে, FileProvider রেফারেন্স ডকুমেন্টেশন দেখুন।

নোট: এক্সএমএল ফাইল হচ্ছে আপনি যে ডিরেক্টরী শেয়ার করতে চান তা নির্দিষ্ট করার একমাত্র উপায়; আপনি প্রোগ্রামেটিকভাবে একটি ডিরেক্টরী যুক্ত করতে পারেন না।

এখন আপনার একটি FileProvider এর একটি পূর্ণাঙ্গ স্পেসিফিকেশন আছে যা আপনার অ্যাপের ইন্টারনাল স্টোরেজের files/ ডিরেক্টরীর ফাইলের জন্য অথবা files/এর সাবডিরেক্টরীর ফাইলের জন্য কনটেন্ট ইউআরআই উৎপন্ন করে। যখন আপনার অ্যাপ একটি ফাইলের জন্য কনটেন্ট ইউআরআই উৎপন্ন করে এটা ফাইলের < provider> এলিমেন্ট (com.example.myapplication.fileprovider), পাথ myimages/, এবং নেম এর মধ্যে নির্দিষ্ট অথরিটিকে ধারণ করে।

উদাহরণস্বরূপ, এই অনুশীলনের এই চিত্রটি অনুসারে আপনি যদি একটি FileProvider নির্ধারন করেন এবং আপনি যদি ফাইল ফবভর্ষঃথরসধমবলঢ়ম এর জন্য একটি কনটেন্ট ইউআরআই রিকোয়েস্ট করেন, FileProvider নিম্নোক্ত ইউআরআই ফেরত দেয়:

```
content://com.example.myapplication.fileprovider/myimages/default_image.jpg
```

একটি ফাইল শেয়ার করা

(<http://developer.android.com/training/secure-file-sharing/share-file.html>)

পূর্বে কখনও কনটেন্ট ইউআরআই ব্যবহার করে ফাইল শেয়ার করতে যদি আপনার অ্যাপ সেটআপ করে থাকেন, ওই ফাইলের জন্য অন্য অ্যাপের পাঠানো রিকোয়েস্ট রেসপন্স করতে পারেন। এই রিকোয়েস্ট রেসপন্স করার একটি উপায়, সার্ভার অ্যাপ থেকে একটি ফাইল সিলেকশন ইন্টারফেস সরবরাহ করা যাতে অন্য অ্যাপলিকেশন আবাহন করতে পারে। এই অ্যাপ্রোচ একটি ক্লায়েন্ট (ভোক্তা) অ্যাপলিকেশনকে সার্ভার অ্যাপ থেকে একটি ফাইল ইউজারকে বাছাই করতে দেওয়ার বিষয়টা অনুমোদন করে এবং তারপর নির্বাচিত ফাইলের কনটেন্ট ইউআরআই রিসিভ করে।

এই অনুশীলনী দেখাবে আপনার অ্যাপে কীভাবে একটি ফাইল নির্বাচনী Activity তৈরী করতে হয় যা ফাইলের জন্য রিকোয়েস্ট রেসপন্স করে।

ফাইল রিকোয়েস্ট রিসিভ করা

ক্লায়েন্ট অ্যাপ থেকে ফাইলের জন্য রিকোয়েস্ট গ্রহণ করতে এবং কনটেন্ট ইউআরআই দিয়ে রেসপন্স করতে, আপনার অ্যাপের উচিত একটি ফাইল নির্বাচনী Activity সরবরাহ করা। ক্লায়েন্ট অ্যাপ একশন ACTION_PICK ধারণ করা একটি Intent দিয়ে startActivityForResult() কল করার মাধ্যমে এই Activity শুরু করে। যখন ক্লায়েন্ট অ্যাপ startActivityForResult() কল করে, আপনার অ্যাপ ক্লায়েন্ট অ্যাপে একটা রেজাল্ট ফেরত দিতে পারে, ইউজার যে ফাইল সিলেক্ট করেছে তার জন্য একটি কনটেন্ট ইউআরআই আকারে।

একটি ক্লায়েন্ট অ্যাপে একটি ফাইলের জন্য রিকোয়েস্ট বাস্তবায়ন করতে হয় কীভাবে তা শিখতে অনুশীলনী Requesting a Shared File দেখুন।

ফাইল সিলেকশন একটিভিটি তৈরী করা

ফাইল সিলেকশন একটিভিটি সেটআপ করতে, আপনার মেনিফেস্ট Activity নির্দিষ্ট করার মাধ্যমে শুরু করুন, একটি ইনটেন্ট ফিল্টার সাথে নিয়ে যা একশন ACTION PICK, ক্যাটাগরি CATEGORY DEFAULT, এবং CATEGORY OPENABLE ম্যাচ করে। আপনার অ্যাপ অন্য অ্যাপে যে ফাইল পরিবেশন করে তার জন্য MIME টাইপ ফিল্টার যুক্তও করে। নীচের অংশটি আপনাকে দেখাবে নতুন একটিভিটি এবং ইনটেন্ট ফিল্টার কীভাবে সুনির্দিষ্ট করতে হয়:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    ...
    <application>
        ...
        <activity
            android:name=".FileSelectActivity"
            android:label="@\"File Selector\" >
            <intent-filter>
                <action
                    android:name="android.intent.action.PICK"/>
                <category
                    android:name="android.intent.category.DEFAULT"/>
                <category
                    android:name="android.intent.category.OPENABLE"/>
                <data android:mimeType="text/plain"/>
                <data android:mimeType="image/*"/>
            </intent-filter>
        </activity>
```


ফাইল সিলেকশন একটিভিটি কোডে নির্ধারণ করুন

পরবর্তীতে, একটি Activity সাবক্লাস নির্ধারণ করুন যা ইন্টারনাল স্টোরেজে আপনার অ্যাপের files/images/ ডিরেক্টরী থেকে সহজপ্রাপ্য ফাইল প্রদর্শন করে এবং ইউজারকে কাঙ্ক্ষিত ফাইল তুলে নিতে অনুমোদন করে। নিম্নোক্ত অংশটি আপনাকে দেখায় কীভাবে এই Activity নির্ধারণ করতে হয় এবং ইউজারের সিলেকশনকে রেসপন্স করতে হয়:

```
public class MainActivity extends Activity {
    // The path to the root of this app's internal storage
    private File mPrivateRootDir;
    // The path to the "images" subdirectory
    private File mImagesDir;
    // Array of files in the images subdirectory
    File[] mImageFiles;
    // Array of filenames corresponding to mImageFiles
    String[] mImageFilenames;
    // Initialize the Activity
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // Set up an Intent to send back to apps that request a file
        mResultIntent =
            new Intent("com.example.myapp.ACTION_RETURN_FILE");
        // Get the files/ subdirectory of internal storage
        mPrivateRootDir = getFilesDir();
        // Get the files/images subdirectory;
        mImagesDir = new File(mPrivateRootDir, "images");
        // Get the files in the images subdirectory
        mImageFiles = mImagesDir.listFiles();
        // Set the Activity's result to null to begin with
        setResult(Activity.RESULT_CANCELED, null);
        /*
         * Display the file names in the ListView mFileListView.
         * Back the ListView with the array mImageFilenames, which
         * you can create by iterating through mImageFiles and
         * calling File.getAbsolutePath() for each File
         */
        ...
    }
    ...
}
```

ফাইল সিলেকশনকে রেসপন্স করা

কখনও একজন ইউজার একটি শেয়ার করা ফাইল নির্বাচন (সিলেক্ট) করে, আপনার অ্যাপলিকেশনকে অবশ্যই কোন ফাইল সিলেক্ট করা হয়েছে তা নির্ধারণ করা এবং তারপর ফাইলের জন্য একটি কনটেন্ট ইউআরআই তৈরী করতে হবে। যেহেতু Activity একটি ListView সহজপ্রাপ্য ফাইলের তালিকা প্রদর্শন করে, যখন ইউজার একটি ফাইল নাম ক্লিক করে সিস্টেম মেথড `onItemClick()` কল করে, যার মধ্যে আপনি একটি নির্বাচিত ফাইল পেয়ে থাকে।

`onItemClick()` এর মধ্যে, নির্বাচিত ফাইলের ফাইল নামের জন্য একটি File অবজেক্ট পেতে এবং `getUriForFile()` এ একটি আরগুমেন্ট হিসাবে পাস করতে, অথরিটি সহ যা আপনি FileProvider এর জন্য `<provider>` এলিমেন্টে সুনির্দিষ্ট করেছেন। উদুভত কনটেন্ট ইউআরআই অথরিটি, ফাইলের ডিরেক্টরীর (এক্সএমএল মেটা-ডাটা তে নির্দিষ্টকরনের মতো) সাথে সংশ্লিষ্ট একটি পাথ সেগমেন্ট, এবং এক্সটেনশন সহ ফাইলের নাম ধারণ করে। কীভাবে FileProvider এক্সএমএল মেটা-ডাটা এর উপর ভিত্তি করে পাথ সেগমেন্টে ডিরেক্টরীগুলো ম্যাপ করে তা (Specify Sharable Directories) অধ্যয়ে আলোচনা করা হয়েছে।

নিম্নোক্ত অংশটি আপনাকে দেখায় কীভাবে নির্বাচিত ফাইল সনাক্ত করতে হয় এবং এটার জন্য একটি কনটেন্ট ইউআরআই পেতে হয়:

```
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Define a listener that responds to clicks on a file in the ListView
    mFileListView.setOnItemClickListener(
        new AdapterView.OnItemClickListener() {
            @Override
            /*
             * When a filename in the ListView is clicked, get its
             * content URI and send it to the requesting app
             */
            public void onItemClick(AdapterView<?> adapterView,
                                    View view,
                                    int position,
                                    long rowId) {
                /*
                 * Get a File for the selected file name.
                 * Assume that the file names are in the
                 * mImageFilename array.
                 */
                File requestFile = new File(mImageFilename[position]);
                /*
                 * Most file-related method calls need to be in
                 * try-catch blocks.
                 */
                // Use the FileProvider to get a content URI
                try {
                    fileUri = FileProvider.getUriForFile(
                        MainActivity.this,
                        "com.example.myapp.fileprovider",
                        requestFile);
                } catch (IllegalArgumentException e) {
                    Log.e("File Selector",
                        "The selected file can't be shared: " +
                        clickedFilename);
                }
            }
        }
    );
}
```

```
}  
    ...  
}  
});  
...  
}
```

মনে রাখবেন যে আপনি শুধুমাত্র ফাইলের জন্য কনটেন্ট ইউআরআই তৈরী করতে পারবেন, যে ফাইল একটি ডিরেক্টরীর মধ্যে থাকে যা আপনি মেটা-ডাটা ফাইলে নির্দিষ্ট করেছেন, যা < paths> এলিমেন্ট ধারণ করে, যেভাবে পূর্ববর্তী অধ্যায়ে (Specify Sharable Directories) আলোচিত হয়েছে। একটি পথ এ একটি File এর জন্য যদি আপনি `getUriForFile()` কল করেন যা আপনি নির্দিষ্ট করেন নি, আপনি একটি `IllegalArgumentException` রিসিভ করবেন।

ফাইলের জন্য পারমিশন (অনুমতি) প্রদান

এখন অন্য অ্যাপের সাথে শেয়ার করতে চাওয়া ফাইলের জন্য আপনার একটি কনটেন্ট ইউআরআই আছে, আপনার দরকার ক্লায়েন্ট অ্যাপকে ফাইলে প্রবেশ করতে দেয়া। প্রবেশগম্যতা অনুমোদন করতে, একটি Intent এ কনটেন্ট ইউআরআই যুক্ত করার মাধ্যমে ক্লায়েন্ট অ্যাপে পারমিশন দিতে পারেন এবং Intent এর উপর পারমিশন ফ্লাগ সেট করতে পারেন। যে পারমিশন আপনি দিয়েছেন তা অস্থায়ী এবং স্বয়ংক্রিয়ভাবে সমাপ্ত হবে যখন রিসিভ করা অ্যাপের কার্যক্রম শেষ হবে।

নিম্নোক্ত কোড অংশটি দেখায় আপনি কীভাবে ফাইলের জন্য রিড পারমিশন সেট করবেন:

```
protected void onCreate(Bundle savedInstanceState) {  
    ...  
    // Define a listener that responds to clicks in the ListView  
    mFileListView.setOnItemClickListener(  
        new AdapterView.OnItemClickListener() {  
            @Override  
            public void onItemClick(AdapterView<?> adapterView,  
                View view,  
                int position,  
                long rowId) {  
                ...  
                if (fileUri != null) {  
                    // Grant temporary read permission to the content URI  
                    mResultIntent.addFlags(  
                        Intent.FLAG_GRANT_READ_URI_PERMISSION);  
                }  
                ...  
            }  
        }  
    );  
    ...  
}
```

সতর্কতা: অস্থায়ী প্রবেশ পারমিশন ব্যবহার করে আপনার ফাইলে নিরপদভাবে প্রবেশগম্যতা অনুমোদন করতে setFlags() কল করা হচ্ছে একমাত্র উপায়। একটি ফাইলের কনটেন্ট ইউআরআই এর জন্য Context.grantUriPermission() মেথড কল করা পরিহার করুন, কারণ এই মেথড প্রবেশগম্যতা প্রদান করে যাতে আপনি শুধুমাত্র Context.revokeUriPermission() কল করার মাধ্যমে আবাহন করতে পারেন।

রিকোয়েস্ট করা অ্যাপের সাথে ফাইল শেয়ার

অ্যাপ যা ফাইল রিকোয়েস্ট করে এটাকে দিয়ে ফাইল শেয়ার করতে, কনটেন্ট ধারণ করা Intent পাস করুন এবং setResult() এ পারমিশন দিন। যখন Activity যা আপনি মাত্রই নির্ধারণ করেছেন তা শেষ হলে, সিস্টেম ক্লায়েন্ট অ্যাপে কনটেন্ট ইউআইআই ধারণ করা Intent সেন্ড করে। নিম্নোক্ত কোড অংশটি দেখায় কীভাবে এটা করতে হয়:

```
protected void onCreate(Bundle savedInstanceState) {  
    ...  
    // Define a listener that responds to clicks on a file in the ListView  
    mFileListView.setOnItemClickListener(  
        new AdapterView.OnItemClickListener() {  
            @Override  
            public void onItemClick(AdapterView<?> adapterView,  
                View view,  
                int position,  
                long rowId) {  
                ...  
                if (fileUri != null) {  
                    ...  
                    // Put the Uri and MIME type in the result Intent  
                    mResultIntent.setDataAndType(  
                        fileUri,  
                        getContentResolver().getType(fileUri));  
                    // Set the result  
                    MainActivity.this.setResult(Activity.RESULT_OK,  
                        mResultIntent);  
                } else {  
                    mResultIntent.setDataAndType(null, "");  
                    MainActivity.this.setResult(RESULT_CANCELED,  
                        mResultIntent);  
                }  
            }  
        });  
};
```

ইউজারকে ক্লায়েন্ট অ্যাপে তাৎক্ষণিকভাবে ফিরে আসার জন্য একটা উপায় প্রদান করুন যদি তারা ইতিমধ্যে একটি ফাইল বেছে নেয়। এটা করার একটা উপায় একটি চেকমার্ক বা **Done** বাটন সরবরাহ করা। বাটনের android:onClick এট্রিবিউট ব্যবহার করে বাটনের সাথে একটা মেথড যুক্ত করুন। উদাহরণস্বরূপ:

```
public void onDoneClick(View v) {  
    // Associate a method with the Done button  
    finish();  
}
```

একটি শেয়ার করা ফাইল রিকোয়েস্ট করা

(<http://developer.android.com/training/secure-file-sharing/request-file.html>)

যখন একটি অ্যাপ অন্য অ্যাপ দ্বারা শেয়ার করা একটি ফাইলে প্রবেশ করতে চায়, রিকোয়েস্ট করা (ক্লায়েন্ট) অ্যাপ সাধারণভাবে ফাইল শেয়ার করা অ্যাপকে একটি রিকোয়েস্ট সেন্ড করে (সারভার)। অধিকাংশ ক্ষেত্রে, সারভার অ্যাপে রিকোয়েস্টটি একটি একটিভিটি শুরু করে যা ফাইল প্রদর্শন করে এটা শেয়ার করতে পারে। ইউজার একটি ফাইল তুলে নেয়, যার পরে সারভার অ্যাপ ক্লায়েন্ট অ্যাপে ফাইলের কনটেন্ট ইউআরআই ফেরত পাঠায়।

এই অনুশীলনী দেখায় কীভাবে একটি ক্লায়েন্ট অ্যাপ একটি সারভার অ্যাপ থেকে একটি ফাইলকে রিকোয়েস্ট করে, সারভার অ্যাপ থেকে ফাইলের কনটেন্ট ইউআরআই গ্রহণ করে, এবং কনটেন্ট ইউআরআই ব্যবহার করে ফাইলটি ওপেন করে।

ফাইলের জন্য একটি রিকোয়েস্ট সেন্ড (পাঠানো) করা

সারভার অ্যাপ থেকে একটি ফাইল রিকোয়েস্ট করতে, ক্লায়েন্ট অ্যাপ ACTION PICK এবং একটি MIME টাইপের মতো একশন ধারণ করা Intent দিয়ে startActivityForResult কল করে যা ক্লায়েন্ট অ্যাপ চালিত করতে পারে।

উদাহরণস্বরূপ, নিম্নোক্ত কোড অংশটি দেখায় Activity শুরু করার জন্য একটি সারভার অ্যাপে কীভাবে একটি Intent সেন্ড করতে হয় যা Sharing a File এ আলোচনা করা হয়েছে:

```
public class MainActivity extends Activity {
    private Intent mRequestFileIntent;
    private ParcelFileDescriptor mInputPFD;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mRequestFileIntent = new Intent(Intent.ACTION_PICK);
        mRequestFileIntent.setType("image/jpg");
        ...
    }
    ...
    protected void requestFile() {
        /**
         * When the user requests a file, send an Intent to the
         * server app.
         * files.
         */
        startActivityForResult(mRequestFileIntent, 0);
        ...
    }
    ...
}
```

রিকোয়েস্ট করা ফাইলে প্রবেশ

সারভার অ্যাপ ফাইলের কনটেন্ট ইউআরআই একটি Intent এর মধ্যে ক্লায়েন্ট অ্যাপে ফেরত পাঠায়। এই Intent ক্লায়েন্ট অ্যাপে এটার onActivityResult()-এর ওভাররাইডে পাস হয়। কখনও ক্লায়েন্ট অ্যাপের ফাইলের কনটেন্ট ইউআরআই থাকে, এটা ফাইলে প্রবেশ করতে পারে এটার FileDescriptor পাওয়ার মাধ্যমে।

ফাইল নিরাপত্তা এই পক্রিয়ার মধ্যে সংরক্ষিত আছে কারন কনটেন্ট ইউআরআই হচ্ছে ডাটার একমাত্র অংশ যা ক্লায়েন্ট অ্যাপ গ্রহণ করে। যেহেতু এই ইউআরআই একটি ডিরেক্টরী পাথ ধারন না করে, ক্লায়েন্ট অ্যাপ সারভার অ্যাপের অন্য ফাইল আবিষ্কার এবং ওপেন করতে পারে না। শুধুমাত্র ক্লায়েন্ট অ্যাপ ফাইলে প্রবেশগম্যতা লাভ করে, এবং শুধুমাত্র সারভার অ্যাপ কর্তৃক পারমিশন প্রদান করা হয়। পারমিশনটা অস্থায়ী, তাই একসময় ক্লায়েন্ট অ্যাপের কার্যক্রম শেষ হলে, সারভার অ্যাপের বাইরে থেকে ফাইলের প্রবেশগম্যতা বিদ্যমান থাকে না। নিচের অংশটি দেখায় সারভার অ্যাপ থেকে পাঠানো Intent কে ক্লায়েন্ট অ্যাপ কীভাবে চালিত করে, এবং ইউআরআই ব্যবহার করে কীভাবে ক্লায়েন্ট অ্যাপ FileDescriptor পেয়ে থাকে:

```
/*
 * When the Activity of the app that hosts files sets a result and calls
 * finish(), this method is invoked. The returned Intent contains the
 * content URI of a selected file. The result code indicates if the
 * selection worked or not.
 */
@Override
public void onActivityResult(int requestCode, int resultCode,
    Intent returnIntent) {
    // If the selection didn't work
    if (resultCode != RESULT_OK) {
        // Exit without doing anything else
        return;
    } else {
        // Get the file's content URI from the incoming Intent
        Uri returnUrl = returnIntent.getData();
        /*
         * Try to open the file for "read" access using the
         * returned URI. If the file isn't found, write to the
         * error log and return.
         */
        try {
            /*
             * Get the content resolver instance for this context, and use it
             * to get a ParcelFileDescriptor for the file.
             */
            mInputPFD = getContentResolver().openFileDescriptor(returnUrl, "r");
        } catch (FileNotFoundException e) {
            e.printStackTrace();
            Log.e("MainActivity", "File not found.");
            return;
        }
        // Get a regular file descriptor for the file
        FileDescriptor fd = mInputPFD.getFileDescriptor();
        ...
    }
}
```


মেথড `openFileDescriptor()` ফাইলের জন্য একটি `ParcelFileDescriptor` ফেরত পাঠায়। এই অবজেক্ট থেকে, ক্লায়েন্ট অ্যাপ একটি `FileDescriptor` অবজেক্ট লাভ করে, এটা পরবর্তীতে ফাইল রিড করতে ব্যবহৃত হয়।

ফাইল তথ্য উদ্ধার

(<http://developer.android.com/training/secure-file-sharing/retrieve-info.html>)

একটি ফাইলে যার জন্য এর একটি কনটেন্ট ইউআরআই আছে তার সাথে একটি ক্লায়েন্ট অ্যাপের কাজ করার চেষ্টা করার পূর্বে, অ্যাপটি সারভার অ্যাপ থেকে তথ্য সম্পর্কে রিকোয়েস্ট করতে পারে, যার মধ্যে রয়েছে ডাটা টাইপ এবং ফাইল সাইজ। ডাটা টাইপ ক্লায়েন্ট অ্যাপকে নির্ধারন করতে সাহায্য করে যাতে এটা ফাইলকে চালিত করতে পারে এবং ফাইল সাইজ ক্লায়েন্ট অ্যাপকে ফাইলের জন্য বাফারিং এবং ক্যাচিং সেটআপ করতে সহায়তা করে।

এই অনুশীলনী দেখায় একটি ফাইলের MIME টাইপ এবং সাইজ উদ্ধার করতে কীভাবে সারভার অ্যাপের FileProvider অনুসন্ধান করতে হয়।

একটি ফাইলের MIME টাইপ উদ্ধার

একটা ফাইলের ডাটা টাইপ ক্লায়েন্ট অ্যাপে নির্দেশ করে কীভাবে এটার ফাইলের কনটেন্ট চালিত করা উচিত। একটা শেয়ার করা ফাইলের ডাটা টাইপ পেতে এটার কনটেন্ট ইউআরআই প্রদান করে, ক্লায়েন্ট অ্যাপ `ContentResolver.getType()` কল করে। এই মেথড ফাইলের MIME টাইপ ফেরত দেয়। বাই ডিফল্ট, একটি `FileProvider` ফাইলের MIME টাইপ নিরূপন করে এর ফাইল নেম এক্সটেনশন থেকে।

নিম্নোক্ত কোড চিত্রটি দেখায় ক্লায়েন্ট অ্যাপ একটি ফাইলের MIME টাইপ উদ্ধার কওে, কোন সময় যদি সারভার অ্যাপ ক্লায়েন্টে কনটেন্ট ইউআরআই ফেরত পাঠায়:

```
...
/*
 * Get the file's content URI from the incoming Intent, then
 * get the file's MIME type
 */
Uri returnUrl = returnIntent.getData();
String mimeType = getContentResolver().getType(returnUrl);
...
```

একটি ফাইলের নাম ও সাইজ উদ্ধার

FileProvider ক্লাসের query() মেথডের একটি ডিফল্ট বাস্তবায়ন আছে যা একটি Cursor এ একটি কনটেন্ট ইউআরআই এর সাথে যুক্ত ফাইলের নাম ও সাইজকে ফেরত পাঠায়। ডিফল্ট বাস্তবায়ন দুইটা কলাম ফেরত পাঠায়:

ডিসপ্লে নেম (DISPLAY NAME)

ফাইলের নাম, একটি String হিসাবে। এই ভ্যালুটা File.getName() দ্বারা ফেরত আসে ভ্যালুর মতো একইভাবে।

সাইজ (SIZE)

ফাইলের সাইজ বাইটে হয়, একটি ষড়হম হিসাবে। এই ভ্যালুটা File.length() দ্বারা ফেরত আসে ভ্যালুর মতো একইভাবে।

কনটেন্ট ইউআরআই এর জন্য ছাড়া null এ query() এর আরগুমেন্টের সব কিছু সেট করার মাধ্যমে ক্লায়েন্ট অ্যাপ একটি ফাইলের জন্য DISPLAY NAME এবং SIZE উভয়ই পেতে পারে। উদাহরণস্বরূপ, এই কোড চিত্রটি একটি ফাইলের DISPLAY NAME এবং SIZE উদ্ধার করে এবং প্রতিটা পৃথক TextView এ প্রদর্শন করে:

```
...
/*
 * Get the file's content URI from the incoming Intent,
 * then query the server app to get the file's display name
 * and size.
 */
Uri returnUrl = returnIntent.getData();
Cursor returnCursor =
    getContentResolver().query(returnUri, null, null, null, null);
/*
 * Get the column indexes of the data in the Cursor,
 * move to the first row in the Cursor, get the data,
 * and display it.
 */
int nameIndex = returnCursor.getColumnIndex(OpenableColumns.DISPLAY_NAME);
int sizeIndex = returnCursor.getColumnIndex(OpenableColumns.SIZE);
returnCursor.moveToFirst();
TextView nameView = (TextView) findViewById(R.id.filename_text);
TextView sizeView = (TextView) findViewById(R.id.filesize_text);
nameView.setText(returnCursor.getString(nameIndex));
sizeView.setText(Long.toString(returnCursor.getLong(sizeIndex)));
...
```

এনএফসি দিয়ে ফাইল শেয়ার করা

(<http://developer.android.com/training/beam-files/index.html>)

অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার বৈশিষ্ট্য ব্যবহার করে ডিভাইসের মধ্যে বড় ফাইল বদলি করতে অ্যান্ড্রয়েড আপনাকে অনুমোদন দেয়। এই বৈশিষ্ট্যের একটি সরল এপিআই আছে এবং শুধুমাত্র ডিভাইস টাচ করার মাধ্যমে ইউজারকে বদলি প্রক্রিয়াটি শুরু করতে দেয়। জবাবে, অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার একটি ডিভাইস থেকে আরেকটিতে স্বয়ংক্রিয়ভাবে ফাইল কপি করে এবং ইউজারকে এটা অবগত করে যখন এটা শেষ হয়।

যখন অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার এপিআই ব্যাপক পরিমান ডাটা নিয়ে কাজ করে, অ্যান্ড্রয়েড ৪.০ (এপিআই লেভেল ১৪) এ চালিত অ্যান্ড্রয়েড বিম NDEF ট্রান্সফার এপিআই অল্প পরিমান ডাটা নিয়ে কাজ করে যেমন ইউআইআই বা অন্য ছোট বার্তা। উপরনুত, অ্যান্ড্রয়েড বিম একমাত্র বৈশিষ্ট্য যা অ্যান্ড্রয়েড NFC ফ্রেমওয়ার্কে পাওয়া যায়, যেটা আপনাকে ট্যাগস থেকে NDEF মেসেজ পড়তে (রিড) অনুমোদন করে। অ্যান্ড্রয়েড বিম সম্পর্কে আরও জানতে Beaming NDEF Messages to Other Devices দেখুন এবং NFC ফ্রেমওয়ার্কে আরও জানতে Near Field Communication এপিআই গাইড দেখুন

অনুশীলনীসমূহ

অন্য ডিভাইসে ফাইল সেল্ড করা

অন্য ডিভাইসে ফাইল সেল্ড করতে শিখুন কীভাবে আপনার অ্যাপ সেটআপ করতে হয়।

অন্য ডিভাইস থেকে ফাইল রিসিভ করা

অন্য ডিভাইস থেকে পাঠানো ফাইল রিসিভ করতে কীভাবে আপনার অ্যাপ সেটআপ করতে হয়

মেনিফেস্টে ফিচারগুলো (বৈশিষ্ট্যগুলো) ডিক্লেয়ার করা

প্রথমে, আপনার অ্যাপের যে পারমিশন এবং ফিচার প্রয়োজন তা ডিক্লেয়ার করতে আপনার অ্যাপ মেনিফেস্ট এডিট করুন।

পারমিশন (অনুমতি) রিকোয়েস্ট (অনুরোধ) করা

NFC ব্যবহার করে এক্সটার্নাল স্টোরেজ থেকে ফাইল সেন্ড করতে আপনার অ্যাপকে অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার ব্যবহার করতে দিতে, আপনাকে অবশ্যই আপনার অ্যাপ মেনিফেস্টে নিম্নোক্ত পারমিশনগুলো রিকোয়েস্ট করতে হবে: NFC আপনার অ্যাপকে NFC এর উপরে ডাটা সেন্ড করতে দিন। এই পারমিশন নির্দিষ্ট করতে < manifest>এলিমেন্টের একটি চাইল্ড হিসাবে নিম্নোক্ত এলিমেন্ট যুক্ত করুন:

```
<uses-permission  
    android:name="android.permission.NFC" />
```

READ_EXTERNAL_STORAGE

আপনার অ্যাপকে এক্সটার্নাল স্টোরেজ থেকে রিড (পড়তে) করতে দিন। এই পারমিশন নির্দিষ্ট করতে < manifest>এলিমেন্টের একটি চাইল্ড হিসাবে নিম্নোক্ত এলিমেন্ট যুক্ত করুন:

```
<uses-permission  
    android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

নোট: অ্যান্ড্রয়েড ৪.২.২ (এপিআই লেভেল ১৭) শুরু করতে এই পারমিশন বাধ্যতামূলক নয়। প্লাটফর্মের ভবিষ্যত সংস্করণে অ্যাপসের জন্য এটা চাইতে পারে যা এক্সটার্নাল স্টোরেজ থেকে রিড করতে (পড়তে) চাইতে পারে। অগ্রসর সর্বগ্রহনযোগ্যতা নিশ্চিত করতে, এখন পারমিশন রিকোয়েস্ট করুন, এটা প্রয়োজনীয় হওয়ার পূর্বেই

NFC বৈশিষ্ট্য গুলো (ফিচারগুলো) নির্দিষ্ট করুন

নির্দিষ্ট করুন যে আপনার অ্যাপ NFC ব্যবহার করছে, < manifest>এলিমেন্টের একটি চাইল্ড হিসাবে একটি < uses-feature> element যুক্ত করার মধ্যে দিয়ে। আপনার অ্যাপ NFC উপস্থিতি ছাড়া কাজ করতে পারে না এটা নির্দেশ করতে true এ android:required এট্রিবিউট সেট করে।

নিম্নোক্ত খন্ডিত অংশটি দেখায় কীভাবে < uses-feature>এলিমেন্ট নির্দিষ্ট করতে হয়:

```
<uses-feature  
    android:name="android.hardware.nfc"  
    android:required="true" />
```

উল্লেখ্য শুধুমাত্র NFC ব্যবহার করা যদি আপনার অ্যাপের একমাত্র অপশন হয়, যদি NFC উপস্থিত না থাকে কিনুত এখনও কাজ করছে, আপনার উচিত false এ android:required সেট করা এবং কোডে NFC এর জন্য পরীক্ষা করা।

অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার নির্দিষ্ট করুন

যেহেতু অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার শুধুমাত্র অ্যান্ড্রয়েড ৪.১ (এপিআই লেভেল ১৬) এবং পরবর্তী সংস্করণে পাওয়া যায়, যদি আপনার অ্যাপ এর কার্যক্রমের একটি মূল অংশের জন্য অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার এ উপরে নির্ভরশীল হয় আপনাকে অবশ্যই `android:minSdkVersion="16"` এট্রিবিউট সহকারে `<uses-sdk>` এলিমেন্ট নির্দিষ্ট করতে হবে। অন্যপ্রকারে আপনি অন্য অ্যাপে প্রয়োজন অনুসারে `android:minSdkVersion` সেট করতে পারেন এবং কোডে প্ল্যাটফর্ম সংস্করণ পরীক্ষা করুন, যেভাবে নিচের সেকশনে আলোচনা করা হয়েছে।

অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার সাপোর্টের জন্য পরীক্ষা

আপনার অ্যাপ মেনিফেস্ট নির্দিষ্ট করতে NFC হচ্ছে অপশনাল, আপনি নিম্নোক্ত এলিমেন্ট ব্যবহার করতে পারেন:

```
<uses feature android:name="android.hardware.nfc" android:required="false" />
```

আপনি যদি `android:required="false"` সেট করেন, আপনাকে অবশ্যই কোডে NFC সাপোর্ট এবং অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার সাপোর্টের জন্য পরীক্ষা করতে হবে।

কোডে অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার সাপোর্টের জন্য পরীক্ষা করতে, পরীক্ষা দ্বারা শুরু করুন যাতে আরগুমেন্ট `FEATURE NFC` সহকারে `PackageManager.hasSystemFeature()` কল করার মাধ্যমে ডিভাইস NFC কে সাপোর্ট করে। পরে, চেক করুন যে অ্যান্ড্রয়েড ভার্সন `SDK_INT` এর ভ্যালু পরীক্ষা করার মাধ্যমে অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফারকে সাপোর্ট করে। যদি অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার সাপোর্ট পায়, NFC কন্ট্রোলারের একটি ইনসটেন্স লাভ করে যা আপনাকে NFC হার্ডওয়ারের সাথে যোগাযোগ করতে দিবে। উদাহরণস্বরূপ:

```
public class MainActivity extends Activity {
    ...
    NfcAdapter mNfcAdapter;
    // Flag to indicate that Android Beam is available
    boolean mAndroidBeamAvailable = false;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        ...
        // NFC isn't available on the device
        if (!PackageManager.hasSystemFeature(PackageManager.FEATURE_NFC)) {
            /*
             * Disable NFC features here.
             * For example, disable menu items or buttons that activate
             * NFC-related features
             */
            ...
        }
        // Android Beam file transfer isn't supported
    } else if (Build.VERSION.SDK_INT <
        Build.VERSION_CODES.JELLY_BEAN_MR1) {
        // If Android Beam isn't available, don't continue.
        mAndroidBeamAvailable = false;
        /*
         * Disable Android Beam file transfer features here.
         */
        ...
    }
    // Android Beam file transfer is available, continue
} else {
    mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
    ...
}
```


একটি কলব্যাক মেথড তৈরী করুন যা ফাইল সরবরাহ করে

ইতিপূর্বে আপনি যদি নিশ্চিত করেন যে ডিভাইসটি অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার সাপোর্ট করে, একটি কলব্যাক মেথড যুক্ত করুন যা সিস্টেমটি আবাহন করে যখন অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার চিহ্নিত করে যে ইউজার অন্য একটি NFC -এনাবলড ডিভাইসে একটি ফাইল পাঠাতে চায়। এই কলব্যাক মেথডে, Uri অবজেক্টের একটি বিন্যাস ফেরত দিন। অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার রিসিভিং ডিভাইসে এই সব ইউআরআই দ্বারা পরিবেশিত ফাইলটিকে কপি করে।

কলব্যাক মেথড যুক্ত করতে, NfcAdapter.CreateBeamUriCallback ইন্টারফেস এবং এর মেথড createBeamUri() বাস্তবায়ন করুন। নিম্নোক্ত খন্ডাংশটি দেখায় এটা কীভাবে করতে হয়:

```
public class MainActivity extends Activity {  
    ...  
    // List of URIs to provide to Android Beam  
    private Uri[] mFileUri = new Uri[10];  
    ...  
    /**  
     * Callback that Android Beam file transfer calls to get  
     * files to share  
     */  
    private class FileUriCallback implements  
        NfcAdapter.CreateBeamUriCallback {  
        public FileUriCallback() {  
        }  
        /**  
         * Create content URIs as needed to share with another device  
         */  
        @Override  
        public Uri[] createBeamUri(NfcEvent event) {  
            return mFileUri;  
        }  
    }  
    ...  
}
```

আপনি কখনও ইন্টারফেসটি বাস্তবায়ন করলে, setBeamPushUriCallback() কল করার মাধ্যমে অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফারে কলব্যাক সরবরাহ করুন। নিম্নোক্ত খন্ডাংশ টি আপনাকে দেখাবে কীভাবে এটা করতে হয়:

```
public class MainActivity extends Activity {  
    ...  
    // Instance that returns available files from this app  
    private FileUriCallback mFileUriCallback;  
    ...  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        ...  
        // Android Beam file transfer is available, continue  
        ...  
    }  
}
```

```

mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
/*
 * Instantiate a new FileUriCallback to handle requests for
 * URIs
 */
mFileUriCallback = new FileUriCallback();
// Set the dynamic callback for URI requests.
mNfcAdapter.setBeamPushUrisCallback(mFileUriCallback, this);
...
}
...
}

```

নোট: আপনার অ্যাপের NfcAdapter ইনসটেন্স এর মাধ্যমে সরাসরি NFC ফ্রেমওয়ার্কে আপনি Uri অবজেক্টের বিন্যাস প্রদানও করতে পারেন। এই অ্যাপ্রোচ বেছে নিন, আপনি যদি NFC টাচ ইভেন্ট ঘটার পূর্বেই স্থানান্তর করতে ইউআরআই নির্ধারণ করতে পারেন। এই অ্যাপ্রোচ সম্পর্কে আরও জানতে, NfcAdapter.setBeamPushUris() দেখুন।

সেভ করতে ফাইল নির্দিষ্ট করুন

অন্য NFC- এনাবলড ডিভাইসে এক বা একাধিক ফাইল স্থানান্তর করতে, প্রতিটা ফাইলের জন্য একটি ফাইল ইউআরআই (একটি file সহকারে একটি URI) সংগ্রহ করুন এবং তারপর Uri অবজেক্টের একটি বিন্যাসে এটা যুক্ত করুন। একটি ফাইল স্থানান্তর করতে, আপনার অবশ্যই ফাইলের জন্য রিড করার স্থায়ী প্রবেশগম্যতা থাকতে হবে। উদাহরণস্বরূপ, নিচের কোড খন্ডাংশটি দেখাবে একটি ফাইল নেম থেকে কীভাবে আপনি একটি ফাইল ইউআরআই পাবেন এবং তারপর ইউআরআইটি বিন্যাসে (অ্যারে) যুক্ত করবেন:

```
/*
 * Create a list of URIs, get a File,
 * and set its permissions
 */
private Uri[] mFileUris = new Uri[10];
String transferFile = "transferimage.jpg";
File extDir = getExternalFilesDir(null);
File requestFile = new File(extDir, transferFile);
requestFile.setReadable(true, false);
// Get a URI for the File and add it to the list of URIs
fileUri = Uri.fromFile(requestFile);
if (fileUri != null) {
    mFileUris[0] = fileUri;
} else {
    Log.e("My Activity", "No File URI available for file.");
}
```

অন্য ডিভাইস থেকে ফাইল রিসিভ করা

(<http://developer.android.com/training/beam-files/receive-files.html>)

অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার রিসিভিং ডিভাইসের একটি বিশেষ ডিরেক্টরীতে ফাইল কপি করে। এটা অ্যান্ড্রয়েড মিডিয়া স্ক্যানার দিয়ে কপি করা ফাইলের স্ক্যান করে এবং MediaStore প্রভাইডারে মিডিয়া ফাইলের জন্য এন্ট্রি যুক্ত করে। এই অনুশীলনী দেখায় কীভাবে আপনি রেসপন্স করবেন যখন ফাইল কপি সম্পূর্ণ হয় এবং রিসিভিং ডিভাইসে কপি করা ফাইলকে স্থাপন করুন।

ডাটা প্রদর্শন করতে একটি রিকোয়েস্টের প্রতি রেসপন্স করা

অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার যখন রিসিভিং ডিভাইসে ফাইল কপি করার কাজ শেষ করে, এটা একশন ACTION VIEW সহকারে একটি Intent ধারণ করা নোটিফিকেশন, প্রথম ফাইলের MIME টাইপ যা স্থানান্তরিত হয়েছে, এবং একটি ইউআরআই যা প্রথম ফাইলের প্রতি নির্দেশ করে তা পোস্ট করে। যখন ইউজার নোটিফিকেশন ক্লিক করে, এই ইনটেন্ট সিস্টেমের বাইরে পঠানো হয়। আপনার অ্যাপের এই ইনটেন্টে রেসপন্স করতে, Activity এর `<activity>` এলিমেন্টের জন্য একটি `<intent-filter>` যুক্ত করুন যার রেসপন্স করা উচিত। `<intent-filter>` এলিমেন্টের মধ্যে নিম্নোক্ত চাইল্ড এলিমেন্ট যুক্ত করুন:

```
<action android:name="android.intent.action.VIEW" />
```

নোটিফিকেশন থেকে পাঠানো ACTION VIEW ইনটেন্ট ম্যাচ করুন।

```
<category android:name="android.intent.category.CATEGORY_DEFAULT" />
```

একটি Intent ম্যাচ করুন যার সুস্পষ্ট কোন ক্যাটাগরি থাকে না।

```
<data android:mimeType="mime-type" />
```

একটি MIME টাইপ ম্যাচ করুন। শুধুমাত্র ঐ সকল MIME টাইপ সুনির্দিষ্ট করুন যা আনার অ্যাপ সামলাতে পারবে

উদাহরণস্বরূপ, নিম্নোক্ত খন্ডাংশটি আপনাকে দেখাবে কীভাবে একটি ইনটেন্ট ফিল্টার যুক্ত করতে হয় যা একটি ভিডিও `com.example.android.nfctransfer.ViewActivity` চালু করে।

```
<activity
    android:name="com.example.android.nfctransfer.ViewActivity"
    android:label="Android Beam Viewer" >
    ...
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        ...
    </intent-filter>
</activity>
```

নোট: অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার একটি ACTION VIEW ইনটেন্টের একমাত্র উৎস নয়। রিসিভিং ডিভাইসের অন্য অ্যাপও এই একশন সহ একটি Intent পাঠাতে পারে। এই অবস্থা সামলানোর বিষয়টা Get the directory from a content URI সেকশনে আলোচনা করা হয়েছে।

ফাইল পারমিশন রিকোয়েস্ট

ফাইল পড়তে যা অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার ডিভাইসে কপি করেছে, পারমিশন `READ_EXTERNAL_STORAGE` রিকোয়েস্ট করুন। উদাহরণস্বরূপ:

```
<uses permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

আপনি যদি আপনার অ্যাপের নিজস্ব স্টোরেজ এলাকাতে স্থানান্তরিত ফাইল কপি করতে চান, পরিবর্তে পারমিশন `WRITE_EXTERNAL_STORAGE` রিকোয়েস্ট করুন। `WRITE_EXTERNAL_STORAGE` `READ_EXTERNAL_STORAGE` কে অন্তর্ভুক্ত করে।

নোট: অ্যান্ড্রয়েড ৪.২.২ (এপিআই লেভেল ১৭) এর মতো, একমাত্র পারমিশন `READ_EXTERNAL_STORAGE` জারি রাখা হয় যদি ইউজার এটা করতে পছন্দ করে। প্ল্যাটফর্মের ভবিষ্যত সংস্করণ সকল ক্ষেত্রে এই পারমিশন চাইতে পারে। অগ্রসর সর্বগ্রহনযোগ্যতা নিশ্চিত করতে, এখন পারমিশন রিকোয়েস্ট করুন, এটা প্রয়োজনীয় হওয়ার পূর্বেই।

যেহেতু আপনার অ্যাপের এর ইন্টারনাল স্টোরেজ এলাকার উপর নিয়ন্ত্রন আছে, আপনার ইন্টারনাল স্টোরেজ এলাকায় একটি স্থানান্তরিত ফাইল কপি করতে আপনাকে রাইট পারমিশন রিকোয়েস্ট করার দরকার হবে না।

কপি কৃত ফাইলের জন্য ডিরেক্টরী পাওয়া

অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার রিসিভিং ডিভাইসের একটি ডিরেক্টরীতে একটি একক ট্রান্সফারে সকল ফাইল কপি করে। অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার নোটিফিকেশন কর্তৃক পাঠানো কনটেন্ট Intent এর মধ্যকার ইউআরআই প্রথম স্থানান্তরিত ফাইলকে নির্দেশ করে। কিন্ত আপনার অ্যাপ অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার ছাড়া অন্য কোন উৎস থেকেও একটি ACTION VIEW ইনটেন্ট রিসিভ করতে পারে। ইনকামিং Intent কীভাবে চালিত করতে হয় তা নির্ধারন করতে, আপনার দরকার এর স্কিম এবং অথরিটি পরীক্ষা করা।

ইউআরআই এর জন্য স্কিম পেতে Uri.getScheme() কল করুন। নিম্নোক্ত কোড খন্ডাংশটা আপনাকে দেখায় কীভাবে স্কিম নির্ধারণ করতে হয় এবং সেইমতে ইউআরআই চালিত করে:

```
public class MainActivity extends Activity {  
    ...  
    // A File object containing the path to the transferred files  
    private File mParentPath;  
    // Incoming Intent  
    private Intent mIntent;  
    ...  
    /*  
     * Called from onNewIntent() for a SINGLE_TOP Activity  
     * or onCreate() for a new Activity. For onNewIntent(),  
     * remember to call setIntent() to store the most  
     * current Intent  
     */  
    private void handleViewIntent() {  
        ...  
        // Get the Intent action  
        mIntent = getIntent();  
        String action = mIntent.getAction();  
        /*  
         * For ACTION_VIEW, the Activity is being asked to display data.  
         * Get the URI.  
         */  
        if (TextUtils.equals(action, Intent.ACTION_VIEW)) {  
            // Get the URI from the Intent  
            Uri beamUri = mIntent.getData();  
            /*  
             * Test for the type of URI, by getting its scheme value  
             */  
            if (TextUtils.equals(beamUri.getScheme(), "file")) {  
                mParentPath = handleFileUri(beamUri);  
            } else if (TextUtils.equals(  
                beamUri.getScheme(), "content")) {  
                mParentPath = handleContentUri(beamUri);  
            }  
        }  
        ...  
    }  
    ...  
}
```

একটি ফাইল ইউআরআই থেকে ডিরেক্টরী লাভ

যদি ইনকামিং Intent একটি ফাইল ইউআরআই ধারণ করে, ইউআরআই ফাইলের একটি সম্পূর্ণ ফাইল নেম ধারণ করে, যার মধ্যে পূর্ণ ডিরেক্টরী পাথ এবং ফাইল নেম। অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার এর জন্য, ডিরেক্টরী পাথ অন্য স্থানান্তরিত ফাইলের অবস্থান নির্দেশ করে, যদি কোন একটা থেকে থাকে। ডিরেক্টরী পাথ পেতে, ইউআরআই এর পাথ অংশ নিন যা এই file: প্রিফিক্স ছাড়া ইউআরআই এর সকল কিছু ধারণ করে। পাথ অংশ থেকে একটি File তৈরী করুন, তারপর File এর প্যারেন্ট পাথ অর্জন করুন:

```
...
public String handleFileUri(Uri beamUri) {
    // Get the path part of the URI
    String fileName = beamUri.getPath();
    // Create a File object for this filename
    File copiedFile = new File(fileName);
    // Get a string containing the file's parent directory
    return copiedFile.getParent();
}
...
```

একটি কনটেন্ট ইউআরআই থেকে ডিরেক্টরী লাভ করা

যদি ইনকামিং Intent একটি কনটেন্ট ইউআরআই ধারণ করে, ইউআরআই একটি ডিরেক্টরীর এবং ফাইল নেমে যা MediaStore কনটেন্ট প্রভাইডারে স্টোর করা আছে তার প্রতি নির্দেশ করে। ইউআরআই এর অথরিটি ভ্যালু পরীক্ষা করার মাধ্যমে এর MediaStore জন্য আপনি একটি কনটেন্ট সনাক্ত করতে পারেন। MediaStore এর জন্য একটি কনটেন্ট ইউআরআই অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার বা অন্য অ্যাপ থেকে আসতে পারে, কিন্তু উভয় ক্ষেত্রেই কনটেন্ট ইউআরআই এর জন্য আপনি একটি ডিরেক্টরী এবং ফাইল নেম উদ্ধার করতে পারেন।

MediaStore এর চেয়ে একটি কনটেন্ট প্রভাইডারের জন্য আপনি একটি কনটেন্ট ইউআরআই ধারণ করা একটি ইনকামিং ACTION VIEW ইনটেন্ট গ্রহণ করতে পারেন। এক্ষেত্রে, কনটেন্ট ইউআরআই MediaStore অথরিটি ভ্যালু ধারণ করে না এবং কনটেন্ট ইউআরআই সাধারণত একটি ডিরেক্টরীর প্রতি নির্দেশ করে না।

নোট: অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার এর জন্য, ACTION VIEW ইনটেন্ট এর মধ্যে আপনি একটি কনটেন্ট ইউআরআই গ্রহণ করেন যদি প্রথম ইনকামিং ফাইলের "audio/*", "image/*", বা "video/*" এর একটি MIME টাইপ থাকে, এটা নির্দেশ করে যে এটা মিডিয়া-সম্পৃক্ত। অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার মিডিয়া ফাইলের ইনডেক্স করে এটা ডিরেক্টরীর রানিং মিডিয়া স্ক্যানার দ্বারা স্থানান্তর করে যেখানে এটা স্থানান্তরিত ফাইল স্টোর করে। মিডিয়া স্ক্যানার এর রেজাল্ট (ফলাফল) MediaStore কনটেন্ট প্রভাইডারে রাইট করে এটা তারপর প্রথম ফাইলের জন্য একটি কনটেন্ট ইউআরআই অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফারে ফেরত পাঠায়। এই কনটেন্ট ইউআরআই হচ্ছে সেটা যেটা আপনি নোটিফিকেশন Intent এ গ্রহণ করেছেন। প্রথম ফাইলের ডিরেক্টরী পেতে, কনটেন্ট ইউআরআই ব্যবহার করে MediaStore থেকে এটা আপনি উদ্ধার করতে পারেন।

কনটেন্ট প্রভাইডার নির্ণয় করা

নিরূপন করতে, আপনি যদি কনটেন্ট ইউআরআই থেকে একটি ফাইল ডিরেক্টরী উদ্ধার করতে চান, ইউআরআই এর অথরিটি পেতে Uri.getAuthority() কল করে ইউআরআই এর সাথে সংশ্লিষ্ট কনটেন্ট প্রভাইডার নিরূপন করুন। ফলাফলের দুইটা সম্ভাব্য ভ্যালু আছে:

MediaStore.AUTHORITY

ইউআরআই একটি ফাইলের জন্য বা ফাইল MediaStore দ্বারা অনুসন্ধান। MediaStore থেকে পূর্ণ ফাইল নেম উদ্ধার করুন এবং ফাইল নেম থেকে ডিরেক্টরী অর্জন করুন।

অন্য যেকোন অথরিটি ভ্যালু

অন্য কনটেন্ট প্রভাইডার থেকে একটি কনটেন্ট ইউআরআই। কনটেন্ট ইউআরআই এর সাথে সংশ্লিষ্ট ডাটা প্রদর্শন করে কিনুত ফাইল ডিরেক্টরী পায় না।

একটি MediaStore কনটেন্ট ইউআরআই এর জন্য ডিরেক্টরী পেতে, একটি অনুসন্ধান রান করুন যা Uri আর্গুমেন্ট এর জন্য ইনকামিং কনটেন্ট ইউআরআই এবং প্রজেকশন এর জন্য কলাম MediaColumns.DATA নির্দিষ্ট করে। ইউআরআই কর্তৃক পরিবেশিত ফাইলের জন্য ফেরত আসা Cursor পূর্ণ পথ এবং নাম ধারণ করে। এই পথ সকল অন্য ফাইল ধারণ করে যা অ্যান্ড্রয়েড বিম ফাইল ট্রান্সফার মাত্রই ডিভাইসে কপি করেছে।

নিম্নোক্ত খন্ডাংশ টি আপনাকে দেখায় কীভাবে কনটেন্ট ইউআরআই এর অথরিটি পরীক্ষা করতে হয় এবং স্থানান্তরিত ফাইলের জন্য পথ এবং নেম উদ্ধার করে:

```
...
public String handleContentUri(Uri beamUri) {
    // Position of the filename in the query Cursor
    int filenameIndex;
    // File object for the filename
    File copiedFile;
    // The filename stored in MediaStore
    String fileName;
    // Test the authority of the URI
    if (!TextUtils.equals(beamUri.getAuthority(), MediaStore.AUTHORITY)) {
        /*
         * Handle content URIs for other content providers
         */
        // For a MediaStore content URI
    } else {
        // Get the column that contains the file name
        String[] projection = { MediaStore.MediaColumns.DATA };
        Cursor pathCursor =
            getContentResolver().query(beamUri, projection,
                null, null, null);
        // Check for a valid cursor
        if (pathCursor != null &&
            pathCursor.moveToFirst()) {
            // Get the column index in the Cursor
            filenameIndex = pathCursor.getColumnIndex(
                MediaStore.MediaColumns.DATA);
```

```
        // Get the full file name including path
        fileName = pathCursor.getString(filenameIndex);
        // Create a File object for the filename
        copiedFile = new File(fileName);
        // Return the parent directory of the file
        return new File(copiedFile.getParent());
    } else {
        // The query didn't work; return null
        return null;
    }
}
}
...
```

একটি কনটেন্ট প্রভাইডার থেকে ডাটা উদ্ধার করা সম্পর্কে আরও জানতে, Retrieving Data from the Provider সেকশনটি দেখুন।

মাল্টিমিডিয়া সহ অ্যাপ তৈরী

(<http://developer.android.com/training/building-multimedia.html>)

এই ক্লাস আপনাকে আপনাকে শেখাবে কীভাবে সমৃদ্ধ মাল্টিমিডিয়া অ্যাপ তৈরী করতে হয় এবং ইউজার যা চায় তার মতো করে আচরণ করে।

১. অডিও প্লেব্যাক ব্যবস্থাপনা

কীভাবে হার্ডওয়ার অডিও কি প্রেস রেসপন্স করতে হয়, যখন অডিও প্লে করা হয় তখন অডিও ফোকাস রিকোয়েস্ট করা এবং অডিও ফোকাসের পরিবর্তনে যথাযথ রেসপন্স করা।

1. আপনার অ্যাপের ভলিউম এবং প্লেব্যাক নিয়ন্ত্রণ
2. অডিও ফোকাস ব্যবস্থাপনা করা
3. অডিও আউটপুট হার্ডওয়ারের সাথে আদান প্রদান করা

২. ফটো ক্যাপচার করা

ইউজারের ডিভাইসে ফটো তুলতে বা সরাসরি ক্যামেরা হার্ডওয়ার নিয়ন্ত্রণ এবং আপনার নিজস্ব ক্যামেরা অ্যাপ তৈরী করার ক্ষেত্রে কীভাবে বিদ্যমান ক্যামেরা অ্যাপকে কাজে লাগানো যায়।

1. ফটো নেয়া
2. ভিডিও রেকর্ড করা
3. ক্যামেরা নিয়ন্ত্রণ করা

৩. কনটেন্ট প্রিন্ট করা

কীভাবে আপনার অ্যাপ থেকে ফটো, HTML ডকুমেন্ট, এবং কাস্টম ডকুমেন্ট প্রিন্ট করতে হয়।

1. ফটো
2. HTML ডকুমেন্ট
3. কাস্টম ডকুমেন্ট

অডিও প্লেব্যাক ব্যবস্থাপনা

(<http://developer.android.com/training/managing-audio/index.html>)

যদি আপনার অ্যাপ অডিও প্লে করে, এটা গুরুত্বপূর্ণ যে আপনার ইউজার অডিওকে কান্ট্রীতে নিয়ন্ত্রণ করতে পারে। একটি অসাধারণ ইউজার এক্সপেরিয়েন্স নিশ্চিত করতে, এটাও গুরুত্বপূর্ণ যে একাধিক অ্যাপ একই সময়ে অডিও প্লে না করাটা নিশ্চিত করতে আপনার অ্যাপ অডিও ফোকাস পরিচালনা করে।

এই ক্লাসের শেষে আপনি অ্যাপস তৈরীতে সমর্থ হবেন যা হার্ডওয়ার অডিও কি প্রেসেস এ রেসপন্স করে, যা অডিও প্লে করার সময় অডিও ফোকাস রিকোয়েস্ট করে এবং সিস্টেম বা অন্য অ্যাপলিকেশন কর্তৃক অডিওর কোন পরিবর্তনকে যথাযথভাবে রেসপন্স করে।

অনুশীলনীসমূহ

আপনার অ্যাপের ভলিউম এবং প্লেব্যাক নিয়ন্ত্রণ

শিখুন হার্ডওয়ার বা সফটওয়ার ভলিউম কন্ট্রোল ব্যবহার করে আপনি কীভাবে নিশ্চিত করবেন আপনার ইউজার অ্যাপের ভলিউম নিয়ন্ত্রণ করতে পারবে এবং প্লে, স্টপ, পজ, স্কিপ এবং প্রিভিয়াস প্লেব্যাক কি কোথায় আছে।

অডিও ফোকাস ব্যবস্থাপনা করা

একাধিক অ্যাপের সম্ভাব্য অডিও প্লেয় স্কেলে এটা চিন্তা করাটা গুরুত্বপূর্ণ তাদের কীভাবে পারস্পরিকভাবে কাজ করা উচিত। একই সময়ে প্রত্যেকটা মিউজিক অ্যাপের এক সাথে প্লে করা পরিহার করতে, অ্যান্ড্রয়েড অডিও প্লেব্যাককে পরিমিত করতে অডিও ফোকাস ব্যবহার করে। শিখুন কীভাবে অডিও ফোকাস রিকোয়েস্ট করতে হয়, অডিও ফোকাসের একটি ক্ষতিকে নজর দিতে হয়, এবং কীভাবে রেসপন্স করতে হয় যখন এটা ঘটে।

অডিও আউটপুট হার্ডওয়ারের সাথে কাজ করা অডিও বিভিন্ন উৎস থেকে প্লে হতে পারে। শিখুন কীভাবে খুঁজে বের করতে হয় কোথা থেকে অডিও প্লে হচ্ছে এবং প্লেব্যাকের সময় বিচ্ছিন্ন হওয়া হ্যান্ডসেট কীভাবে সামলাতে হয়।

আপনার অ্যাপের ভলিউম এবং প্লেব্যাক নিয়ন্ত্রণ

(<http://developer.android.com/training/managing-audio/volume-playback.html>)

একটি ভালো ইউজার এক্সপেরিয়েন্স একটি অনুমেয় বিষয়। আপনার অ্যাপ যদি মিডিয়া প্লে করে এটা গুরুত্বপূর্ণ যে আপনার ইউজার তাদের ডিভাইস, ব্লুটুথ, বা হেডফোনের হার্ডওয়া বা সফটওয়্যার ভলিউম কন্ট্রোল ব্যবহার করে আপনার অ্যাপের ভলিউম নিয়ন্ত্রণ করতে পারে।

একইভাবে, আপনার অ্যাপ কর্তৃক ব্যবহৃত অডিও স্ট্রিমের কোথায় প্লে, স্টপ, পজ, স্কিপ এবং প্রিভিয়াস মিডিয়া প্লেব্যাক কিগুলো তাদের নিজ নিজ কার্য সম্পাদন করতে পারে এবং এগুলো কোথায় থাকে।

কোন অডিও স্ট্রিম ব্যবহার করতে হবে তা চিহ্নিত করা

একটি অনুমেয় অডিও এক্সপেরিয়েন্স তৈরীর প্রথম ধাপ হচ্ছে আপনার অ্যাপ কোন অডিও স্ট্রিম ব্যবহার করবে তা বোঝা।

অ্যাম্বেড মিউজিক, এলার্ম, নোটিফিকেশন, ইনকামিং কল রিংগার, সিস্টেম সাউন্ড, ইন-কল ভলিউম এবং DTMF টোন প্লে করার জন্য একটি আলাদা অডিও স্ট্রিম বজায় রাখে। এটা করা হয় প্রাথমিকভাবে প্রতিটা স্ট্রিম এর ভলিউম স্বাধীনভাবে ইউজারকে নিয়ন্ত্রণ করতে দিতে।

এই অধিকাংশ স্ট্রিম সিস্টেম ইভেন্টে সীমাবদ্ধ, সুতরাং যদি না আপনার অ্যাপ একটি বদলি এলার্ম ক্লক না হয়, আপনি প্রায় নিশ্চিতভাবে STREAM MUSIC স্ট্রিম ব্যবহার করে আপনার অডিও প্লে করবেন

আপনার অ্যাপের অডিও ভলিউম নিয়ন্ত্রণ করতে হার্ডওয়ার ভলিউম কি ব্যবহার করুন

বাই ডিফল্ট, ভলিউম কন্ট্রোল প্রেস করা সক্রিয় অডিও স্ট্রিমকে পরিমিত করে। যদি আপনার অ্যাপ এখন কোন কিছু প্লে করে না থাকে, ভলিউম কি তে চাপ দিয়ে রিংগার ভলিউম সমন্বয় করুন।

আপনি যদি একটি গেম বা মিউজিক অ্যাপ পেয়ে থাকেন, তখন ভালো সম্ভাবনা থাকে যে যখন ইউজার ভলিউম কি হিট করে তারা গেম বা মিউজিকের ভলিউম নিয়ন্ত্রণ করতে চায়, এমনকি যদি তারা বর্তমানে গান বা চলতি গেম লোকেশনে মিউজিক নেই এমন হয়।

আপনি ভলিউম কি প্রেস এর জন্য চেষ্টা করতে এবং শুনতে প্রলুব্ধ হতে পারেন এবং আপনার অডিও স্ট্রিমের ভলিউম এই উপায়ে পরিবর্তন করতে পারেন। এটা করবেন না। অ্যান্ড্রয়েড অডিও স্ট্রিম যেটা আপনি নির্দিষ্ট করেছেন তাতে ভলিউম কি প্রেসেস পরিচালনা করতে উপকারী `setVolumeControlStream()` মেথড প্রদান করে।

অডিও স্ট্রিম চিহ্নিত থাকতে যা আপনার অ্যাপলিকেশন ব্যবহার করতে থাকবে, আপনার এটাকে ভলিউম স্ট্রিম টার্গেট হিসাবে সেট করা উচিত। আপনার উচিত আপনার অ্যাপের লাইফসাইকেলের প্রথমদিকে এই কল করে ফেলা-কারণ আপনার শুধু এটাকে একবার একটিভিটি লাইফসাইকেল সময়কালে কল করা দরকার, আপনার উচিত `onCreate()` মেথডের (Activity বা Fragment এর যা আপনার মিডিয়া নিয়ন্ত্রণন করে) মধ্যে সাধারণভাবে এটাকে কল করা। এটা নিশ্চিত করে যে যখনই আপনার অ্যাপ দৃশ্যমান হবে, ভলিউম কার্যক্রমকে কে নিয়ন্ত্রণ করবে ইউজারের আকাঙ্ক্ষা অনুসারে।

```
setVolumeControlStream(AudioManager.STREAM_MUSIC);
```

এই বিন্দু থেকে, ডিভাইসের ভলিউম কি প্রেস করা অডিও স্ট্রিম কে প্রভাবিত করে যা আপনি নির্দিষ্ট করেছেন (এক্ষেত্রে “মিউজিক”) যখনই টার্গেট একটিভিটি বা ফ্রাগমেন্ট দৃশ্যমান হয়।

আপনার অ্যাপের অডিও প্লেব্যাক নিয়ন্ত্রণ করতে হার্ডওয়ার প্লেব্যাক নিয়ন্ত্রণ কি ব্যবহার করুন

মিডিয়া প্লেব্যাক বাটন, যেমন প্লে, পজ, স্টপ, স্কিপ এবং প্রিভিয়াস কিছ' হ্যান্ডসেটে আছে এবং অনেক সংযুক্ত বা বেতার হেডসেটেও আছে। যখন একজন ইউজার এই হার্ডওয়ার কি এর একটি প্রেস করে, সিস্টেমটি ACTION MEDIA BUTTON একশন সহকারে একটি ইনটেন্ট সম্প্রচার করে।

মিডিয়া বাটন ক্লিক এ রেসপন্স করতে, আপনার প্রয়োজন আপনার মেনিফেস্টে একটি BroadcastReceiver রেজিস্টার করা যা একশন ব্রডকাস্ট শুনে থাকে যা নিম্নে দেয়া আছে।

```
<receiver android:name=".RemoteControlReceiver">
    <intent-filter>
        <action android:name="android.intent.action.MEDIA_BUTTON" />
    </intent-filter>
</receiver>
```

রিসিভার বাস্তবায়ন তার নীজেরই বের হওয়া দরকার যে কি ব্রডকাস্ট ঘটতে প্রেস করা হয়। Intent এটা EXTRA KEY EVENT কি'র অধীনে অন্তর্ভুক্ত করে, যখন KeyEvent ক্লাস একটি তালিকা KEYCODE MEDIA স্টাটিক কনস্টেন্ট অন্তর্ভুক্ত করে যা প্রতিটা সম্ভাব্য মিডিয়া বাটন পরিবেশন করে, যেমন KEYCODE MEDIA PLAY PAUSE এবং KEYCODE MEDIA NEXT।

নিম্নের খন্ডাংশটি দেখায় যে কীভাবে মিডিয়া বাটন প্রেসড সরিয়ে ফেলতে হয় এবং সে অনুসারে মিডিয়া প্লেব্যাক কে প্রভাবিত করতে হয়।

```
public class RemoteControlReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (Intent.ACTION_MEDIA_BUTTON.equals(intent.getAction())) {
            KeyEvent event = (KeyEvent)intent.getParcelableExtra(Intent.EXTRA_KEY_EVENT);
            if (KeyEvent.KEYCODE_MEDIA_PLAY == event.getKeyCode()) {
                // Handle key press.
            }
        }
    }
}
```

কারণ বহুমুখি (মাল্টিপল) অ্যাপলিকেশন মিডিয়া বাটন প্রেসেস এরজন্য শুনতে চাইতে পারে, আপনাকে অবশ্যই প্রোগ্রামেটিক্যালি নিয়ন্ত্রণ করতে হবে যখন আপনার অ্যাপের মিডিয়া বাটন প্রেস ইভেন্ট গ্রহন করা উচিত।

AudioManager ব্যবহার করে আপনার মিডিয়া বাটন ইভেন্ট রিসিভার রেজিস্টার এবং ডিরেজিস্টার করতে নীচের কোডটি আপনার অ্যাপের মধ্যে ব্যবহৃত হতে পারে। যখন রেজিস্টার করা হয়,

আপনার ব্রডকাস্ট রিসিভার হচ্ছে সকল মিডিয়া বাটন ব্রডকাস্টের এক্সক্লুসিভ রিসিভার।

```
AudioManager am = mContext.getSystemService(Context.AUDIO_SERVICE);  
...  
  
// Start listening for button presses  
am.registerMediaButtonEventReceiver(RemoteControlReceiver);  
...  
  
// Stop listening for button presses  
am.unregisterMediaButtonEventReceiver(RemoteControlReceiver);
```

সাধারণভাবে, অ্যাপের উচিত তাদের অধিকাংশ রিসিভার আনরেজিস্টার করা যখনই তারা নিষ্ক্রিয় বা অদৃশ্য হয় (যেমন, onStop() কলব্যাকের সময়কালে)। কিন্ত এটা মিডিয়া প্লেব্যাক অ্যাপসের জন্য সহজ নয়-কার্যত, মিডিয়া প্লেব্যাক বাটনে রেসপন্স করা হচ্ছে সবচেয়ে গুরুত্বপূর্ণ যখন আপনার অ্যাপলিকেশন দৃশ্যমান নয় এবং সেইজন্য অন-স্ক্রিন ইউজার ইন্টারফেস কর্তৃক নিয়ন্ত্রিত হতে পারে না।

একটি ভালো অ্যাপ্রোচ হচ্ছে মিডিয়া বাটন ইভেন্ট রিসিভার রেজিস্টার এবং আনরেজিস্টার করা যখন আপনার অ্যাপলিকেশন অডিও ফোকাসটি অর্জন করে বা হারিয়ে ফেলে। এটা পরবর্তী অনুশীলনীতে বিস্তারিতভাবে আলোচনা করা হয়েছে।

অডিও ফোকাস ব্যবস্থাপনা

(<http://developer.android.com/training/managing-audio/audio-focus.html>)

বহুমুখি (মাল্টিপল) অ্যাপস দিয়ে সম্ভাব্য অডিও প্লে এর ক্ষেত্রে তারা কীভাবে পারস্পরিকভাবে কাজ করে সেই বিষয়ে চিন্তা করাটা গুরুত্বপূর্ণ। প্রতিটা মিউজিক অ্যাপ একই সময়ে প্লে করা পরিহার করতে, অ্যান্ড্রয়েড অডিও প্লেব্যাককে পরিমিত করতে অডিও ফোকাস ব্যবহার করে- শুধুমাত্র সেই অ্যাপ যা অডিও ফোকাস ধারণ করে তার অডিও প্লে করা উচিত।

আপনার অ্যাপের অডিও প্লে শুরু করার পূর্বেই এর অডিও ফোকাস রিকোয়েস্ট এবং রিসিভ করা উচিত। একই ভাবে এটার জানা উচিত অডিও ফোকাস লস হলে কীভাবে তা শোনা উচিত এবং এটা ঘটলে কীভাবে যথাযথভাবে রেসপন্স করতে হয়।

অডিও ফোকাস রিকোয়েস্ট

আপনার অ্যাপের কোন অডিও প্লে শুরু করার পূর্বেই এটার উচিত স্ট্রিমে যেটা ব্যবহার হবে তার জন্য জন্য অডিও ফোকাস ধারন করা উচিত। `requestAudioFocus()`এ কল দিয়ে এটা সম্পূর্ণ হবে যা `AUDIOFOCUS_REQUEST_GRANTED` ফেরত দেয় যদি আপনার রিকোয়েস্ট সফল হয়।

আপনাকে অবশ্যই নির্দিষ্ট করতে হবে কোন স্ট্রিম আপনি ব্যবহার করতে যাচ্ছেন এবং আপনি কোনটা চান, অস্থায়ী বা স্থায়ী অডিও ফোকাস। অস্থায়ী ফোকাস রিকোয়েস্ট করা হয় যখন আপনি অল্প সময়ের জন্য অডিও প্লে করতে চান (উদাহরনস্বরূপ যখন নেভিগেশন ইনস্ট্রাকশন প্লে করা হয়)। স্থায়ী অডিও ফোকাস রিকোয়েস্ট করা হয় যখন আপনি অনুমেয় ভবিষ্যতের জন্য অডিও প্লে করার পরিকল্পনা করবেন (উদাহরনস্বরূপ, যখন মিউজিক প্লে করা হয়)।

নিম্নোক্ত খন্ড চিত্রটি মিউজিক অডিও স্ট্রিমে স্থায়ী অডিও ফোকাস রিকোয়েস্ট করে। আপনার প্লেব্যাক শুরু করার পূর্বে আপনাকে অবশ্যই তাৎক্ষনিকভাবে অডিও ফোকাস রিকোয়েস্ট করতে হবে, যেমন যখন ইউজার পরবর্তী গেম লেভেল শুরু করার জন্য প্লে বা ব্যাকগ্রাউন্ড মিউজিক প্রেস করে।

```
AudioManager am = mContext.getSystemService(Context.AUDIO_SERVICE);
...

// Request audio focus for playback
int result = am.requestAudioFocus(afChangeListener,
                                // Use the music stream.
                                AudioManager.STREAM_MUSIC,
                                // Request permanent focus.
                                AudioManager.AUDIOFOCUS_GAIN);

if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
    am.unregisterMediaButtonEventReceiver(RemoteControlReceiver);
    // Start playback.
}
```

ইতিমধ্যে আপনি যদি প্লেব্যাক সম্পূর্ণ করেন নিশ্চিত হতে `abandonAudioFocus()` কল করুন। এটা সিস্টেমটিকে জ্ঞাপন করে যে আপনি আর ফোকাস চান না এবং সংশ্লিষ্ট `AudioManager.OnAudioFocusChangeListener` কে আনরেজিস্টার করে। অস্থায়ী ফোকাস পরিত্যাগের ক্ষেত্রে, এটা যে কোন বাধাগ্রস্থ অ্যাপকে প্লেব্যাক জারি রাখতে অনুমোদন দেয়।

```
// Abandon audio focus when playback complete
am.abandonAudioFocus(afChangeListener);
```

যখন রিকোয়েস্ট করা অস্থায়ী অডিও ফোকাস আপনার কাছে অতিরিক্ত অপশন হিসাবে থাকে: যেটাই হোক আপনি "ducking" সক্রিয় করতে চাইবেন। স্বাভাবিকভাবে, যখন সদাচরন করা অডিও অ্যাপ অডিও ফোকাস লস করে এটা তাৎক্ষনিকভাবে এটার প্লেব্যাক নিরব করে দেয়। একটি অস্থায়ী অডিও ফোকাস যা ডাকিং করতে দেয় সেটা রিকোয়েস্ট করার মাধ্যমে আপনি অন্য অডিও

অ্যাপসকে বলেন যে প্লে বজায় রাখতে এটা তাদের জন্য গ্রহনযোগ্য, যতক্ষণ না ফোকাসাট তাদের কাছে ফেরত আসে তারা তাদের ভলিউম নীচু রাখে।

```
// Request audio focus for playback
int result = am.requestAudioFocus(afChangeListener,
                                   // Use the music stream.
                                   AudioManager.STREAM_MUSIC,
                                   // Request permanent focus.
                                   AudioManager.AUDIOFOCUS_GAIN_TRANSIENT_MAY_DUCK);

if (result == AudioManager.AUDIOFOCUS_REQUEST_GRANTED) {
    // Start playback.
}
```

ডাকিং বিশেষভাবে অ্যাপসের জন্য উপযোগী যা থেকে থেকে অডিও স্ট্রিম ব্যবহার করে, যেমন শ্রবণগোচর ড্রাইভিং ডিরেকশন।

যখন অন্য অ্যাপ অডিও ফোকাস রিকোয়েস্ট করে যেভাবে উপরে আলোচনা করা হয়েছে, এটা যখন ফোকাস রিকোয়েস্ট করা হয় তখন আপনি যে শ্রোতা রেজিস্টার করেছেন তাদের দ্বারা গ্রহণ করা স্থায়ী এবং অস্থায়ী () অডিও ফোকাসের মধ্যে একটি পছন্দ করে।

অডিও ফোকাসের ক্ষতি/হারিয়ে ফেলা কে ব্যবস্থাপনা করা

যদি আপনার অ্যাপ অডিও ফোকাস রিকোয়েস্ট করতে পারে, এটা অনুসরণ করে যে এটা পালান্ক্রমে লস করে যখন অন্য অ্যাপ এটাকে রিকোয়েস্ট করে। আপনার অ্যাপ অডিও ফোকাসের লস কীভাবে রেসপন্স করে তা নির্ভর করে ঐ লসের আচরনের উপর।

অডিও ফোকাসের `onAudioFocusChange()` কলব্যাক মেথড শ্রোতা যা আপনি রেজিস্টার করেছেন তাকে পরিবর্তন করে যখন রিকোয়েস্টিং অডিও ফোকাস একটি মানদণ্ড ব্যবহার করে যা ফোকাস চেঞ্জ ইভেন্ট আলোচনা করে। নির্দিষ্টভাবে, সম্ভাব্য ফোকাস লস ইভেন্ট ফোকাস রিকোয়েস্ট টাইপ কে প্রতিফলিত করে-পূর্ববর্তী সেকশন থেকে- স্থায়ী লস, অস্থায়ী লস এবং অস্থায়ী সাথে ডাকিং পারমিটেড।

সাধারনভাবে বলতে, অডিও ফোকাসের একটি অস্থায়ী লসের উচিত আপনার অ্যাপের অডিও স্ট্রিম নিশুচপ করিয়ে দিয়ে এর উপর প্রভাব ফেলা, কিন্তু অন্যথায় একই অবস্থা বহার রেখে। আপনার উচিত অডিও ফোকাসের পরিবর্তন মনিটর করা অব্যাহত রাখা এবং প্লেব্যাক পূণরায় শুরু করতে প্রসুত হন যেখানে এটা পজ করা হয়েছিল আপনি পূনরায় ফোকাস অর্জন করবেন।

যদি অডিও ফোকাস লস স্থায়ী হয়, এটা গৃহিত হয় যে অন্য অ্যাপলিকেশন এখন অডিও শুনতে শুরু করবে এবং আপনার অ্যাপের কার্যকরীভাবে নিজেই এটার শেষ করা উচিত। বাস্তবিকপক্ষে, ওটার মানে প্লেব্যাক স্টপ করা, মিডিয়া বাটন লিসেনার রিম্যুভ করা-নতুন অডিও প্লেয়ারকে এক্সক্লুসিভভাবে ওই ইভেন্ট চালিত করতে অনুমোদন করে-এবং আপনার অডিও ফোকাস ত্যাগ করে। যে সময়ে, অডিও প্লে পূণরায় শুরু করার পূর্বেই প্রয়োজন হবে এমন একটি ইউজার একশন আপনি আশা করতে পারেন(আপনার অ্যাপে প্লে প্রেস করা)।

নিচের কোড খন্ডচিত্রটিতে, আমরা প্লেব্যাক বা আমাদের মিডিয়া প্লেয়ার অবজেক্ট পজ দেয় যদি অডিও লস অস্থায়ী হয় এবং আমরা এটা পূণরায় শুরু করি যখন আমরা ফোকাস পূণরায় অর্জন করি। যদি লস স্থায়ী হয়, এটা আমাদের মিডিয়া বাটন ইভেন্ট রিসিভার আনরেজিস্টার করে এবং অডিও ফোকাস এর পরিবর্তন মনিটরিং করা বন্ধ করে।

```
OnAudioFocusChangeListener afChangeListener = new OnAudioFocusChangeListener() {
    public void onAudioFocusChange(int focusChange) {
        if (focusChange == AudioManager.AUDIOFOCUS_LOSS_TRANSIENT
            // Pause playback
        ) else if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {
            // Resume playback
        } else if (focusChange == AudioManager.AUDIOFOCUS_LOSS) {
            am.unregisterMediaButtonEventReceiver(RemoteControlReceiver);
            am.abandonAudioFocus(afChangeListener);
            // Stop playback
        }
    }
};
```

আউও ফোকাসের একাট অস্থায়ী লসের ক্ষেত্রে ডাকিং অনুমতি প্রাপ্ত, প্লেব্যাক পজ না করে আপান পরিবর্তে “ডাক” করতে ডারেন।

Duck! / ডাক !

ডাকিং হচ্ছে অন্য অ্যাপ থেকে অস্থায়ী অডিও বানাতে আপনার অডিও স্ট্রিম আউটপুট ভলিউম নীচু করার একটি প্রক্রিয়া, আপনার নিজস্ব অ্যাপলিকেশন থেকে কোন রকম ব্যহত ছাড়াই অডিও সহজভাবে শুনতে।

নীচের কোডের খন্ডিত চিত্রটি আমাদের মিডিয়া প্লেয়ার অবজেক্টে ভলিউম কমিয়ে দেয় যখন আমরা অস্থায়ী ভাবে ফোকাস লস করি, তারপর এটাকে এর পূর্ববর্তী লেভেলে ফেরত আনি যখন আমরা ফোকাস পূরণায় অর্জন করি।

```
OnAudioFocusChangeListener afChangeListener = new OnAudioFocusChangeListener() {  
    public void onAudioFocusChange(int focusChange) {  
        if (focusChange == AUDIOFOCUS_LOSS_TRANSIENT_CAN_DUCK) {  
            // Lower the volume  
        } else if (focusChange == AudioManager.AUDIOFOCUS_GAIN) {  
            // Raise it back to normal  
        }  
    }  
};
```

অডিও ফোকাসের লস হচ্ছে প্রতিক্রিয়া করতে সবচেয়ে গুরুত্বপূর্ণ ব্রডকাস্ট, কিন্তু একমাত্র নয়। সিস্টেম ইউজারের অডিও এক্সপেরিয়েন্সের পরিবর্তনের প্রতি সতর্ক করতে কয়েকটি ইনটেন্ট ব্রডকাস্ট করে। পরবর্তী অনুশীলনী দেখাবে ইউজারের সার্বিক এক্সপেরিয়েন্স উন্নত করতে কীভাবে তাদের মনিটর করা হয়।

অডিও আউটপুট হার্ডওয়ার ব্যবস্থা করা

(<http://developer.android.com/training/managing-audio/audio-output.html>)

ইউজারের কিছু বিকল্প আছে যখন এটা তাদের অ্যান্ড্রয়েড ডিভাইস থেকে অডিও উপভোগ করতে আসে। অধিকাংশ ডিভাইসের বিল্ট-ইন স্পিকার, তারসহ হেডসেটের জন্য হেডফোন জ্যাক এবং আরও অনেক কিছু আছে, এছাড়াও ব্লুটুথ কানেকটিভিটি এবং A2DP অডিও সমর্থন বৈশিষ্ট্যও দেয়।

চেক করুন কোন হার্ডওয়ার ব্যবহৃত হচ্ছে

কীভাবে আপনার অ্যাপ আচরণ করবে তা প্রভাবিত হতে পারে কিসে হার্ডওয়ার এর আউটপুট রুট ঠিক করা হয়েছে।

নির্ধারন করতে পারেন যে যদি বর্তমানে অডিওটি ডিভাইস স্পিকারের, তার সহ হেডসেট, বা সংযুক্ত বুলটুথ ডিভাইস এর দিকে রুট করা হয়েছে আপনি এক্ষেত্রে AudioManager টি অনুসন্ধান করতে পারেন যেভাবে নীচের খন্ড চিত্রটিতে দেখানো হয়েছে:

```
if (isBluetoothA2dpOn()) {  
    // Adjust output for Bluetooth.  
} else if (isSpeakerphoneOn()) {  
    // Adjust output for Speakerphone.  
} else if (isWiredHeadsetOn()) {  
    // Adjust output for headsets  
} else {  
    // If audio plays and noone can hear it, is it still playing?  
}
```

অডিও আউটপুট হার্ডওয়ার এর পরিবর্তন সামলানো

যখন একটি হেডসেটকে আনপ্লাগড করা হয়, অথবা একটি বুলটুথ ডিভাইস বিচ্ছিন্ন করা হয় অডিও স্ট্রিম স্বয়ংক্রিয়ভাবে বিল্ট-ইন স্পিকারের দিকে রুট ঠিক করে। আপনি যদি আপনার মিউজিক শুনে থাকেন আমি যত জোরে শুনছি ঠিক ততো জোরে, এটা একটি শব্দময় বিশ্বয় হতে পারে।

সৌবাগ্যবশত সিস্টেমটি একটি ACTION_AUDIO_BECOMING_NOISY ইনটেন্ট ব্রডকাস্ট করে যখন এটা ঘটে। একটি BroadcastReceiver রেজিস্টার করা একটা ভালো চর্চা যা এই ইনটেন্টের জন্য শুনে থাকে যখন আপনি এটা প্লে করছেন। মিউজিক প্লেয়ারের ক্ষেত্রে, ইউজার চিরাচিরতভাবে আশা করে প্লেব্যাক স্টপ হবে- যখন গেমের জন্য আপনি উল্লেখযোগ্যভাবে ভলিউম নীচু করতে পছন্দ করতে পারেন

```
private class NoisyAudioStreamReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        if (AudioManager.ACTION_AUDIO_BECOMING_NOISY.equals(intent.getAction())) {
            // Pause the playback
        }
    }
}

private IntentFilter intentFilter = new IntentFilter(AudioManager.ACTION_AUDIO_BECOMING_NOISY);

private void startPlayback() {
    registerReceiver(myNoisyAudioStreamReceiver(), intentFilter);
}

private void stopPlayback() {
    unregisterReceiver(myNoisyAudioStreamReceiver);
}
```


ফটো ক্যাপচার করা

(<http://developer.android.com/training/camera/index.html>)

সমৃদ্ধ মিডিয়া সহজলভ্য হওয়ার আগে পৃথিবী নিরস এবং বৈশিষ্টহীন ছিল। Gopher কে স্মরণ করতে পারেন? আমরা এখন আর পারি না। আপনার অ্যাপকে আপনার ইউজারের জীবনের একটি অংশ করতে, তাদের কে একটি সুযোগ দিন যাতে তারা তাদের জীবন কিছ' অংশ এটার মধ্যে দিতে পারে। অন-বোর্ড ক্যামেরা ব্যবহার করে, আপনার অ্যাপলিকেশন ইউজার তার চার পাশে যা দেখে তা ছড়িয়ে দিতে সামর্থ্য তৈরী করে, একটি স্বতন্ত্র অ্যাভাটার তৈরী, চারপাশে জম্বি (ভূত-প্রেত) খুঁজে বেরানো, অথবা শুধুমাত্র অভিজ্ঞতা বিনিময়।

এই ক্লাস আপনাকে বিদ্যমান ক্যামেরা অ্যাপলিকেশনের সর্বোচ্চ সুবিধা নেয়ার অতি-সহজ উপায় সহ দ্রুত ক্লিক করা অবস্থায় পাবে। পরবর্তী অনুশীলনীতে, আপনি আরও গভীরে যাবেন এবং ক্যামেরা হার্ডওয়ার সরাসরি কীভাবে নিয়ন্ত্রণ করা তা শিখতে পাবেন।

অনুশীলনীসমূহ

ফটো নেয়া

অন্য অ্যাপলিকেশনকে প্রভাবিত করা এবং শুধুমাত্র কিছু কোডের লাইন সহ ফটো ক্যাপচার করা

ভিডিও রেকর্ড করা

অন্য অ্যাপলিকেশনকে প্রভাবিত করা এবং শুধুমাত্র কিছু কোডের লাইন সহ ভিডিও রেকর্ড করা

ক্যামেরা নিয়ন্ত্রণ

ক্যামেরা হার্ডওয়ার সরাসরি নিয়ন্ত্রণ করা এবং আপনার নিজস্ব ক্যামেরা বাস্তায়ন করা ক্যাপচার করা

ছবি নেয়া

(<http://developer.android.com/training/camera/photobasics.html>)

এই অনুশীলনী ব্যাখ্যা করবে কীভাবে একটি বিদ্যমান ক্যামেরা ব্যবহার করে একটি ফটো ধারণ করা যায়।

ধরুন আপনি একটি ক্লাউড-সোর্সড (জন-সম্পৃক্ত) আবহাওয়া সেবা দিয়ে থাকেন যা আপনার ক্লায়েন্ট অ্যাপ রান করা ডিভাইস কর্তৃক তোলা আকাশের ছবি একত্রিত করার মাধ্যমে একটি বৈশ্বিক আবহাওয়া ম্যাপ তৈরী করে। ফটো একত্রিত করা আপনার অ্যাপলিকেশনের খুব ক্ষুদ্র একটা অংশ। আপনি অতি ব্যস্ততা না দেখিয়ে ফটো তুলতে চান, ক্যামেরায় অতি পরিবর্তন না করে। খুশির খবর হচ্ছে অধিকাংশ অ্যান্ড্রয়েড ডিভাইস কমপক্ষে একটি করে ক্যাসেরা অ্যাপলিকেশন ইনস্টল করেছে। এই অনুশীলনীতে, আপনি শিখবেন কীভাবে এটাকে আপনার ছবি তোলার জন্য বানাবেন।

ক্যামেরা পারমিশন রিকোয়েস্ট করা

যদি আপনার অ্যাপের একটি গুরুত্বপূর্ণ কাজ হয় ছবি তোলা, তাহলে গুগল প্লেতে ডিভাইসে যার ক্যামেরা আছে তাতে এর দৃশ্যমানতা সীমাবদ্ধ করে দিন। প্রচারিত করা যে আপনার অ্যাপলিকেশন ক্যামেরা থাকার উপর নির্ভরশীল, আপনার মেনিফেস্ট ফাইলে একটি `<uses-feature>` ট্যাগ রাখুন:

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera" />
    ...
</manifest ... >
```

যদি আপনার অ্যাপলিকেশন ব্যবহার করে, কিনুত কাজ করার জন্য কোন ক্যামেরা না চায়, ট্যাগে `android:required="false"` যুক্ত করুন। এটা করলে, গুগল প্লে ক্যামেরা ছাড়া ডিভাইসে আপনার অ্যাপলিকেশন ডাউনলোড করতে দেয়। তারপর `hasSystemFeature(PackageManager.FEATURE_CAMERA)` কল করে রান টাইমে ক্যামেরা আছে কিনা তা চেক করা আপনার দায়িত্ব। যদি ক্যামেরা না থাকে, আপনার উচিত তখন ক্যামেরা বৈশিষ্ট্য সক্রিয় করা।

ক্যামেরা অ্যাপ দিয়ে ফটো তুলুন

অ্যান্ড্রয়েড ডেলিগেটিং উপায় অন্য অ্যাপলিকেশনে কাজ করা হচ্ছে একটি Intentকে আহবান করা যেটা আপনি যা করতে চান তা বর্ণনা করে। এই প্রক্রিয়া তিনটা অংশ কে যুক্ত করে: Intent নিজে, এক্সটার্নাল Activity শুরু করতে একটি কল এবং যখন ফোকাস আপনার একটিভিটিতে ফিরে আসে তখন ইমেজ ডাটা চালিত করার জন্য কিছু কোড।

এখানে একটি ফাংশন আছে যেটা একটি ছবি তুলতে একটি ইনটেন্ট কে আহবান করে।

```
private void dispatchTakePictureIntent(int actionCode) {  
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    startActivityForResult(takePictureIntent, actionCode);  
}
```

অভিনন্দন: এই কোড দিয়ে আপনার অ্যাপলিকেশন অন্য ক্যামেরা অ্যাপলিকেশনকে এট বিড করানোর মেতা সামর্থ অর্জন করেছে! অবশ্যই, যদি কোন সর্বউপযোগী অ্যাপলিকেশন ইনটেন্ট ধরার জন্য প্রসূত না থাকে, তখন আপনার অ্যাপ ব্যর্থতায় পর্যবসিত হবে। এখানে একটি ফাংশন দেয়া হলো যার মাধ্যমে আপনি চেক করতে পারবেন আপনার অ্যাপটা আপনার ইনটেন্টটা সামলাতে পারবে কিনা:

```
public static boolean isIntentAvailable(Context context, String action) {  
    final PackageManager packageManager = context.getPackageManager();  
    final Intent intent = new Intent(action);  
    List<ResolveInfo> list =  
        packageManager.queryIntentActivities(intent, PackageManager.MATCH_DEFAULT_ONLY);  
    return list.size() > 0;  
}
```

ফটো ভিউ করা

যদি কৃতিত্বের সাথে ছবি তোলাটাই আপনার অ্যাপের চূড়ান্ত আকাঙ্ক্ষা না হয়, তাহলে আপনি তখন সম্ভবত ক্যামেরা অ্যাপ থেকে ছবিটা ফেরত চাইবেন এবং এটা দিয়ে অন্য কিছু করবেন।

অ্যান্ড্রয়েড ক্যামেরা অ্যাপলিকেশন ফিরতি Intent এ ফটোটি এনকোড করে, কি "data"র অধীনে এক্সট্রার মধ্যে একটি Bitmap বিটম্যাপ হিসাবে onActivityResult()এ ডেলিভারি দেয়। নিম্নোক্ত কোড এই ইমেজ উদ্ধার করে এবং ImageView এ প্রদর্শন করে।

```
private void handleSmallCameraPhoto(Intent intent) {  
    Bundle extras = intent.getExtras();  
    mImageBitmap = (Bitmap) extras.get("data");  
    mImageView.setImageBitmap(mImageBitmap);  
}
```

নোট: "data" থেকে এই থামবনেইল ইমেজ একটি আইকনের জন্য ভালো হতে পারে, কিনুত এর চেয়ে বেশী কিছু না। একটি পূর্ণ সাইজের ইমেজ নিয়ে কাজ করতে আরেকটু বেশী কাজ করতে হয়।

ফটো সেভ করা

অ্যান্ড্রয়েড ক্যামেরা অ্যাপলিকেশন একটি পূর্ণ সাইজের ফটো সেভ করে যদি আপনি এটা সেভ করতে দেন। আপনাকে অবশ্যই একটি পথ সরবরাহ করতে হবে যা স্টোরেজ ভলিউম, ফোল্ডার এবং ফাইল নেম অন্তর্ভুক্ত করে।

এখানে ফটোর জন্য পথ পাওয়ার একটি সহজ উপায় আছে, কিন্ত এটা শুধু অ্যান্ড্রয়েড ২.২ (এপিআই লেভেল ৮) এবং এর পরবর্তী সংস্করণে কাজ করে:

```
storageDir = new File(  
    Environment.getExternalStoragePublicDirectory(  
        Environment.DIRECTORY_PICTURES  
    ),  
    getAlbumName()  
);
```

প্রথমদিককার এপিআই লেভেলের জন্য, আপনার নিজেকে ফটো ডিরেক্টরীর নাম সরবরাহ করতে হবে।

```
storageDir = new File (  
    Environment.getExternalStorageDirectory()  
        + PICTURES_DIR  
        + getAlbumName()  
);
```

নোট: পথ কম্পোনেন্ট PICTURE DIR হচ্ছে শুধু Pictures/, শেয়ার ফটোর জন্য যথাযথ লোকেশন (স্থান) হচ্ছে এক্সটার্নাল/ শেয়ারড স্টোরেজ।

ফাইল নেম সেট করুন

পূর্ববর্তী অধ্যায়ে যেভাবে দেখানো হয়েছে, একটি ইমেজের জন্য ফাইল লোকেশন অবশ্যই ডিভাইস এনভায়রনমেন্ট দ্বারা চালিত হওয়া উচিত। যার জন্য আপনার নিজেকে একটি সংঘর্ষ-রোধক ফাইল-নাম করা স্কিম (a collision-resistant file-naming scheme) বেছে নিতে হবে। পরবর্তী ব্যবহারের জন্য আপনি একটি মেম্বর ভেরিয়েবলের মধ্যে পথ সেভ করতে চাইতেও পারেন। এখানে একটি উদারণ সমাধান দেয়া হলো:

```
private File createImageFile() throws IOException {
    // Create an image file name
    String timeStamp =
        new SimpleDateFormat("yyyyMMdd_HHmmss").format(new Date());
    String imageFileName = JPEG_FILE_PREFIX + timeStamp + "_";
    File image = File.createTempFile(
        imageFileName,
        JPEG_FILE_SUFFIX,
        getAlbumDir()
    );
    mCurrentPhotoPath = image.getAbsolutePath();
    return image;
}
```


ইনটেন্টের উপর ফাইলনেম যুক্ত করুন

যদি কখনও আপনার ইমেজ সেভ করার একটি জায়গা থাকে, Intent এর মাধ্যমে ঐ লোকেশনকে ক্যামেরা অ্যাপলিকেশনে পাস করে দিন।

```
File f = createImageFile();  
takePictureIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(f));
```

একটি গ্যালারিতে ফটো সংযোজন করুন

যখন ইন্টেনেটের মাধ্যমে আপনি একটি ফটো তৈরী করবেন, আপনার জানা উচিত ইমেজ কোথায় আছে, কারন আপনি বলেছেন প্রথম স্থানের মধ্যে এটা কোথায় সেভ করতে হবে। বাকীদের জন্য, সম্ভবত সহজ উপায়ে আপনার ফটোকে প্রবেশ যোগ্য করা হচ্ছে সিস্টেমের মিডিয়া প্রভাইডার থেকে এটাকে প্রবেশযোগ্য করা।

নিম্নোক্ত উদাহরণ মেথড দেখায় মিডিয়া প্রভাইডার ডাটাবেজ আপনার ফটো যুক্ত করে কীভাবে সিস্টেমের মিডিয়া স্ক্যানারকে আহবান করে, অ্যান্ড্রয়েড গ্যালারী অ্যাপলিকেশন এবং অন্য অ্যাপে এটাকে সহজপ্রপ্য করা যায়।

```
private void galleryAddPic() {  
    Intent mediaScanIntent = new Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE);  
    File f = new File(mCurrentPhotoPath);  
    Uri contentUri = Uri.fromFile(f);  
    mediaScanIntent.setData(contentUri);  
    this.sendBroadcast(mediaScanIntent);  
}
```

একটি স্কেলড ইমেজ ডিকোড করা

সীমিত আকারের মেমরীতে বহুমুখি ইমেজ ব্যবস্থনা জটিল হতে পারে। আপনি যদি দেখেন যে আপনার অ্যাপলিকেশন অল্প কিছু ছবি দেখিয়েই মেমরী শেষ করে ফেলে, আপনি নাটকীয়ভাবে একটি মেমরী বিন্যাসের মধ্যে বর্ধিত করার দ্বারা ব্যবহৃত ডায়নামিক হিপ গুলো কমিয়ে ফেলতে পারেন যা ইতিমধ্যে গন্তব্য ভিউ এর সাইজের সাথে ম্যাচ করানোর জন্য মাপা হয়েছে। নিম্নোক্ত উদাহরণ মেথড এই কৌশল দেখায়।

```
private void setPic() {
    // Get the dimensions of the View
    int targetW = mImageView.getWidth();
    int targetH = mImageView.getHeight();

    // Get the dimensions of the bitmap
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    bmOptions.inJustDecodeBounds = true;
    BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    int photoW = bmOptions.outWidth;
    int photoH = bmOptions.outHeight;

    // Determine how much to scale down the image
    int scaleFactor = Math.min(photoW/targetW, photoH/targetH);

    // Decode the image file into a Bitmap sized to fill the View
    bmOptions.inJustDecodeBounds = false;
    bmOptions.inSampleSize = scaleFactor;
    bmOptions.inPurgeable = true;

    Bitmap bitmap = BitmapFactory.decodeFile(mCurrentPhotoPath, bmOptions);
    mImageView.setImageBitmap(bitmap);
}
```

ভিডিও রেকর্ড করা

(<http://developer.android.com/training/camera/videobasics.html>)

এই অনুশীলনী দেখাবে কীভাবে বিদ্যমান ক্যামেরা অ্যাপলিকেশন দিয়ে ভিডিও ক্যাপচার করতে হয়।

আপনার অ্যাপলিকেশনের করার জন্য কাজ আছে, এবং ভিডিও সমন্বয় করা তার একটি ক্ষুদ্র অংশ। আপনি অতি ব্যস্ততা না দেখিয়ে ভিডিও করতে চান, ক্যামকর্ডারে অতি পরিবর্তন না করে। খুশির খবর হচ্ছে অধিকাংশ অ্যান্ড্রয়েড ডিভাইসের একটি ক্যামেরা অ্যাপলিকেশন আছে যা ভিডিও রেকর্ড করে। এই অনুশীলনীতে, আপনি এটা আপনার জন্য তা করার জন্য বানাবেন।

ক্যামেরা পারমিশন রিকোয়েস্ট

প্রচার করতে যে, আপনার অ্যাপলিকেশন একটি ক্যামেরা থাকার উপর নির্ভর করে, মেনিফেস্টে ফাইলে একটি `<uses-feature>` ট্যাগ রাখুন :

```
<manifest ... >
    <uses-feature android:name="android.hardware.camera" />
    ...
</manifest ... >
```

যদি আপনার অ্যাপলিকেশন ব্যবহার করে, কিনুত কাজ করার জন্য কোন ক্যামেরা না চায়, ট্যাগে `android:required="false"` যুক্ত করুন। এটা করলে, গুগল প্লে ক্যামেরা ছাড়া ডিভাইসে আপনার অ্যাপলিকেশন ডাউনলোড করতে দেয়। তারপর `hasSystemFeature(PackageManager.FEATURE_CAMERA)` কল করে রান টাইমে ক্যামেরা আছে কিনা তা চেক করা আপনার দায়িত্ব। যদি ক্যামেরা না থাকে, আপনার উচিত তখন ক্যামেরা বৈশিষ্ট সক্রিয় করা।

ক্যামেরা অ্যাপ দিয়ে ভিডিও রেকর্ড করা

অ্যান্ড্রয়েড ডেলিগেটিং উপায় অন্য অ্যাপলিকেশনে কাজ করা হচ্ছে একটি Intent কে আহবান করা যেটা আপনি যা করতে চান তা বর্ণনা করে। এই প্রক্রিয়া তিনটা অংশ কে যুক্ত করে: Intent নিজে, এক্সটার্নাল Activity শুরু করতে একটি কল এবং যখন ফোকাস আপনার একটিভিটিতে ফিরে আসে তখন ভিডিও চালিত করার জন্য কিছু কোড।

এখানে একটি ফাংশন আছে যেটা একটি ভিডিও করতে একটি ইনটেন্ট কে আহবান করে।

```
private void dispatchTakeVideoIntent() {  
    Intent takeVideoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);  
    startActivityForResult(takeVideoIntent, ACTION_TAKE_VIDEO);  
}
```

এটা একটা ভালো চিন্তা, একটি অ্যাপ একটি ইনটেন্টকে আহবান করার পূর্বে সামলানোর জন্য অস্তিত্বশীল থাকে। এখানে একটি ফাংশন দেয়া আছে যা অ্যাপের জন্য চেক করে যা আপনার ইনটেন্ট সামলাতে পারবে:

```
public static boolean isIntentAvailable(Context context, String action) {  
    final PackageManager packageManager = context.getPackageManager();  
    final Intent intent = new Intent(action);  
    List<ResolveInfo> list =  
        packageManager.queryIntentActivities(intent,  
            PackageManager.MATCH_DEFAULT_ONLY);  
    return list.size() > 0;  
}
```

ভিডিও ভিউ

অ্যান্ড্রয়েড ক্যামেরা অ্যাপলিকেশন Intent এর মধ্যে ভিডিও ফেরত দেয় Uri হিসাবে onActivityResult() এর প্রতি ডেলিভারি করে স্টোরেজে ভিডিও লোকেশন নির্দেশ করে। নিচের কোড এই ভিডিও উদ্ধার করে এবং VideoView এ এটা প্রদর্শন করে।

```
private void handleCameraVideo(Intent intent) {  
    mVideoUri = intent.getData();  
    mVideoView.setVideoURI(mVideoUri);  
}
```

ক্যামেরা নিয়ন্ত্রণ করা

(<http://developer.android.com/training/camera/cameradirect.html>)

এই অনুশীলনীতে আমরা আলোচনা করব ফ্রেমওয়ার্ক এপিআই ব্যবহার করে সরাসরি কীভাবে ক্যামেরা হার্ডওয়ার নিয়ন্ত্রণ করা যায়।

সরাসরি একটি ডিভাইস ক্যামেরা নিয়ন্ত্রণ বিদ্যমান ক্যামেরা অ্যাপলিকেশন থেকে ছবি বা ভিডিও রিকোর্ড করা যায়। কিন্তু যদি আপনি বিশেষ ক্যামেরা অ্যাপলিকেশন বা আপনার অ্যাপে সমন্বিত অবস্থায় থাকা অন্য কিছু তৈরী করতে চান, এই অনুশীলনী আপনাকে এটা করতে শেখাবে।

ক্যামেরা অবজেক্ট ওপেন করা

Camera অবজেক্ট এর একটি ইনসটেন্স পাওয়া হচ্ছে সরাসরি ক্যামেরা নিয়ন্ত্রন প্রক্রিয়ার প্রথম ধাপ। যেহেতু অ্যান্ড্রয়েডের নিজস্ব ক্যামেরা অ্যাপলিকেশন করে থাকে, ক্যামেরাতে প্রবেশ করার প্রস্তাবিত উপায় হচ্ছে একটি পৃথক থ্রেড যা onCreate()থেকে শুরু হয় তার উপর Camera ওপেন করা। এই অ্যাপ্রোচ একটি ভালো ধারণা যেহেতু এটা কিছু সময় ব্যয় করতে পারে এবং ইউজার ইন্টারফেস থ্রেড কে অগ্রসর হতে বাধা দিতে পারে। একটি আরও মৌলিক বাস্তায়নের মধ্যে, কোড পূর্বব্যবহার করতে সঞ্চালন করতে এবং নিয়ন্ত্রণ প্রবাহ ধরে রাখতে ক্যামেরা ওপেন করা onResume()মেথডে বিলম্ব করতে পারে।

Camera.open()কল করা একটি ব্যতিক্রম করতে পারে যদি ক্যামেরা ইতিমধ্যে অন্য অ্যাপ কর্তৃক একটি ব্যবহারের মধ্যে থাকে, তাই আমরা এটাকে ংতুল্লকে যুক্ত করে রেখে দিতে পারি।

```
private boolean safeCameraOpen(int id) {
    boolean qOpened = false;

    try {
        releaseCameraAndPreview();
        mCamera = Camera.open(id);
        qOpened = (mCamera != null);
    } catch (Exception e) {
        Log.e(getString(R.string.app_name), "failed to open Camera");
        e.printStackTrace();
    }

    return qOpened;
}

private void releaseCameraAndPreview() {
    mPreview.setCamera(null);
    if (mCamera != null) {
        mCamera.release();
        mCamera = null;
    }
}
```

যদি এপিআই লেভেল ৯ হয়, ক্যামেরা ফ্রেমওয়ার্ক বহুমুখি ক্যামেরা সাপোর্ট করে। যদি আপনি লিগ্যাসি এপিআই ব্যবহার করেন এবং একটি আরগুমেন্ট ছাড়াই open()কল করেন, আপনি প্রথম রিয়ার ফেসিং ক্যামেরা পাবেন।

ক্যামেরা প্রিভিউ তৈরী করা

একটি ছবি নেয়ার জন্য সাধারনত প্রয়োজন যে আপনার ইউজার শাটার টেপার আগে তার সাবজেক্টের প্রিভিউ দেখতে পারে। এটা করতে, আপনি যা ক্যামেরা সেন্সরে তুলবেন তার প্রিভিউ ড্র করতে SurfaceView ব্যবহার করতে পারেন।

প্রিভিউ ক্লাস

প্রিভিউ প্রদর্শন করে (ডিসপ্লে) শুরু করতে, আপনার প্রিভিউ ক্লাস দরকার। প্রিভিউ `android.view.SurfaceHolder.Callback` ইন্টারফেসের বাস্তবায়ন চায়, যা ক্যামেরা হার্ডওয়ার থেকে অ্যাপলিকেশনে ইমেজ ডাটা পাস করার কাজে ব্যবহার করা হয়।

```
class Preview extends ViewGroup implements SurfaceHolder.Callback {

    SurfaceView mSurfaceView;
    SurfaceHolder mHolder;

    Preview(Context context) {
        super(context);

        mSurfaceView = new SurfaceView(context);
        addView(mSurfaceView);

        // Install a SurfaceHolder.Callback so we get notified when the
        // underlying surface is created and destroyed.
        mHolder = mSurfaceView.getHolder();
        mHolder.addCallback(this);
        mHolder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }
    ...
}
```

লাইভ ইমেজ প্রিভিউ শুরু হওয়ার পূর্বেই প্রিভিউ ক্লাস অবশ্যই Cameraতে পাস করতে হবে, যেভাবে পরবর্তী অধ্যায়ে দেখানো হয়েছে।

প্রিভিউ সেট করুন এবং শুরু করা

একটি ক্যামেরা ইনস্টেন্স এবং এর সাথে সম্পর্কিত প্রিভিউ অবশ্যই একটি নির্দিষ্ট ক্রমানুসারে তৈরী হতে হবে, ক্যামেরা অবজেক্ট দিয়ে প্রথমে হবে। নীচের খন্ডচিত্রটিতে ক্যামেরা আরম্ভের প্রক্রিয়া সংযুক্ত করা হয়েছে যাতে `setCamera()` মেথড দ্বারা `Camera.startPreview()` কল করা হয়, যখনই ইউজার ক্যামেরার পরিবর্তনের জন্য কিছু করে। প্রিভিউকে অবশ্যই প্রিভিউ ক্লাস `surfaceChanged()` কলব্যাক মেথডে রিস্টার্ট করতে হবে।

```
public void setCamera(Camera camera) {
    if (mCamera == camera) { return; }

    stopPreviewAndFreeCamera();

    mCamera = camera;

    if (mCamera != null) {
        List<Size> localSizes = mCamera.getParameters().getSupportedPreviewSizes();
        mSupportedPreviewSizes = localSizes;
        requestLayout();

        try {
            mCamera.setPreviewDisplay(mHolder);
        } catch (IOException e) {
            e.printStackTrace();
        }

        /*
         Important: Call startPreview() to start updating the preview surface. Preview must
         be started before you can take a picture.
        */
        mCamera.startPreview();
    }
}
```

ক্যামেরা সেটিং পরিবর্তন করা

ক্যামেরা সেটিং যে উপায়ে ক্যামেরা ছবি নেয় তা পরিবর্তন করে, জুম লেভেল থেকে শুরু করে এক্সপোজার কম্পেনসেশন পর্যন্ত। এই উদাহরণ শুধুমাত্র প্রিভিউ সাইজ পরিবর্তন করে; আর কোন কিছুর জন্য ক্যামেরা অ্যাপলিকেশনের সোর্স কোড দেখুন।

```
public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {  
    // Now that the size is known, set up the camera parameters and begin  
    // the preview.  
    Camera.Parameters parameters = mCamera.getParameters();  
    parameters.setPreviewSize(mPreviewSize.width, mPreviewSize.height);  
    requestLayout();  
    mCamera.setParameters(parameters);  
  
    /*  
     Important: Call startPreview() to start updating the preview surface. Preview must be  
     started before you can take a picture.  
    */  
    mCamera.startPreview();  
}
```

প্রিভিউ ওরিয়েন্টেশন সেট করুন

অধিকাংশ ক্যামেরা অ্যাপলিকেশন ডিসপ্লেকে ল্যান্ডস্কেপ পদ্ধতিতে আটকে ফেলে কারন ঐটা ক্যামেরা সেন্সরের স্বাভাবিক ওরিয়েন্টেশন। এই সেটিং আপনাকে পোর্ট্রেইট-মোড ফটো নিতে বাধা দিবে না কারন ডিভাইসের ওরিয়েন্টেশন EXIF হেডারে রেকর্ড করা আছে। কীভাবে ইমেজ রেকর্ড করা হচ্ছে তাকে প্রভাবান্বিত না করেই কীভাবে প্রিভিউ প্রদর্শিত (ডিসপ্লে) হবে তা `setCameraDisplayOrientation()` মেথড আপনাকে পরিবর্তন করতে দিবে। এপিআই লেভেল ১৪ এর আগের অ্যান্ড্রয়েডে আপনাকে অবশ্যই ওরিয়েন্টেশন পরিবর্তন করার পূর্বে আপনার প্রিভিউ বন্ধ করতে হবে এবং তারপর এটা রিস্টার্ট করতে হবে।

একটি ছবি নেয়া

যখনই প্রিভিউ শুরু হবে একটি ছবি তুলতে `Camera.takePicture()` ব্যবহার করুন। আপনি `Camera.PictureCallback` এবং `Camera.ShutterCallback` অবজেক্ট তৈরী করতে পারেন এবং তাদের `Camera.takePicture()` এ পাস করে দিতে পারেন।

যদি আপনি ক্রমাগতভাবে ইমেজ নিতে চান, আপনি একটি `Camera.PreviewCallback` তৈরী করতে পারেন যা `onPreviewFrame()` বাস্তবায়ন করে। এর মধ্যে কিছু করতে, আপনি শুধু বাছাই করা প্রিভিউ ফ্রেমস ক্যাপচার করতে পারেন, অথবা `takePicture()` কল করে একটি দেরী করা একশন সেটআপ করতে পারেন।

রিভিউ পূরণায় শুরু করনি (রিস্টার্ট)

একটি ছবি নেয়া শেষ হলে, ইউজার আরেকটি ছবি নেয়ার পূর্বেই আপনাকে অবশ্যই প্রিভিউ রিস্টার্ট করতে হবে। এই উদাহরণে, শাটার বাটন ওভারলোড করার মাধ্যমে রিস্টার্ট করা হয়েছে।

```
@Override
public void onClick(View v) {
    switch(mPreviewState) {
        case K_STATE_FROZEN:
            mCamera.startPreview();
            mPreviewState = K_STATE_PREVIEW;
            break;

        default:
            mCamera.takePicture( null, rawCallback, null);
            mPreviewState = K_STATE_BUSY;
    } // switch
    shutterBtnConfig();
}
```


প্রিভিউ বন্ধ করুন এবং ক্যামেরা রিলিজ (ছেড়ে দেয়া) করুন

একসময় আপনার অ্যাপলিকেশন ক্যামেরা ব্যবহার করা শেষ করবে, এটা এখন পরিষ্কার করার সময়। বিশেষ করে, আপনাকে অবশ্যই Camera অবজেক্ট রিলিজ করতে হবে, অথবা আপনি অন্য অ্যাপলিকেশন ক্র্যাশ করার ঝুঁকি নিবেন, যার মধ্যে আপনার নিজস্ব অ্যাপলিকেশনের নতুন ইনসটেন্স রয়েছে।

আপনাকে কখন প্রিভিউ বন্ধ এবং ক্যামেরা রিলিজ করতে হবে? আপনার প্রিভিউ সারফেস ধ্বংস হওয়া একটি ভালো নিদর্শন যে প্রিভিউ বন্ধ এবং ক্যামেরা রিলিজ করতে এটাই উপযুক্ত সময়, Preview ক্লাস থেকে এই মেথডের মধ্যে যে ভাবে দেখানো হয়েছে।

```
public void surfaceDestroyed(SurfaceHolder holder) {
    // Surface will be destroyed when we return, so stop the preview.
    if (mCamera != null) {
        /*
         * Call stopPreview() to stop updating the preview surface.
         */
        mCamera.stopPreview();
    }
}

/**
 * When this function returns, mCamera will be null.
 */
private void stopPreviewAndFreeCamera() {

    if (mCamera != null) {
        /*
         * Call stopPreview() to stop updating the preview surface.
         */
        mCamera.stopPreview();

        /*
         * Important: Call release() to release the camera for use by other applications.
         * Applications should release the camera immediately in onPause() (and re-open() it in
         * onResume()).
         */
        mCamera.release();

        mCamera = null;
    }
}
```

এই অনুশীলনীর প্রথম দিকে, এই প্রক্রিয়াও setCamera() এর একটি অংশ ছিল, তাই একটি ক্যামেরা শুরু করাটা সবসময় প্রিভিউ বন্ধ করার মাধ্যমে শুরু হয়।

কনটেন্ট প্রিন্ট করা

(<http://developer.android.com/training/printing/index.html>)

অ্যান্ড্রয়েড ইউজার ক্রমাগতভাবে তাদের ডিভাইসে শুধুমাত্র কনটেন্ট ভিউ করে (দেখে)। কিন্ত একটি সময় আসতে পারে কোন তথ্য শেয়ার করার জন্য কাউকে শুধুমাত্র স্ক্রিনে দেখানোটা পর্যাপ্ত উপায় নাও হতে পারে। আপনার অ্যাপলিকেশন থেকে তথ্য প্রিন্ট দিতে পারাটা একজন ইউজারকে আপনার অ্যাপ থেকে একটি বড় আকারের কনটেন্ট দেখার বা অন্য ব্যক্তি যিনি আপনার অ্যাপ ব্যবহার করেন না তার সাথে এই তথ্য শেয়ার করার একটি উপায় প্রদান করে। প্রিন্টিং তাদের তথ্যর (ইনফর্মেশন) একটি -œযাপশট তৈরী করতে দেয় যা ডিভাইস থাকা বা না থাকা, পর্যাপ্ত ব্যাটারী পাওয়ার, বা ওয়ারলেস নেটওয়ার্ক কানেকশন এর উপর নির্ভর করে না।

অ্যান্ড্রয়েড ৪.৪ (এপিআই লেভেল ১৯) এবং এর চেয়ে উপরের সংস্করণের মধ্যে, ফ্রেমওয়ার্ক সরাসরি অ্যান্ড্রয়েড অ্যাপলিকেশন থেকে প্রিন্ট করা ইমেজে এবং ডকুমেন্টস এর জন্য সার্ভিস প্রদান করে। এই প্রশিক্ষন আলোচনা করবে আপনার অ্যাপলিকেশনে কীভাবে প্রিন্টিং কে সক্রিয় করা যায় যার মধ্যে রয়েছে প্রিন্টিং ইমেজ, HTML পেজেস এবং প্রিন্ট করার জন্য কাস্টম ডকুমেন্টস তৈরী করা।

অনুশীলনীসমূহ

ফটো প্রিন্ট

এই অনুশীলনী দেখাবে কীভাবে ইমেজ প্রিন্ট করতে হয়

HTML ডকুমেন্ট প্রিন্ট

এই অনুশীলনী দেখাবে কীভাবে HTML ডকুমেন্ট প্রিন্ট করতে হয়

কাস্টম ডকুমেন্টস প্রিন্ট

এই অনুশীলনী দেখাবে কীভাবে আপনি অ্যান্ড্রয়েড প্রিন্ট ম্যানেজারের সাথে সংযোগ করবেন, একটি প্রিন্ট এ্যাডাপটর তৈরী করবেন এবং প্রিন্টের জন্য কনটেন্ট তৈরী করবেন

ফটো প্রিন্ট

(<http://developer.android.com/training/printing/photos.html>)

মোবাইল ডিভাইসের জন্য একটি ফটো তোলা এবং শেয়ার করাটা সবচেয়ে জনপ্রিয় ব্যবহার। আপনার অ্যাপলিকেশন যদি ফটো নেয়, তাদের প্রদর্শন করে এবং ইউজারকে তা শেয়ার করতে অনুমোদন করে, সেক্ষেত্রে আপনাকে আপনার অ্যাপলিকেশন থেকে ঐ ইমেজ প্রিন্টকে সক্রিয় করার বিষয়টিকে বিবেচনা করা উচিত। Android Support Library (<http://developer.android.com/tools/support-library/index.html>) একটি ন্যূনতম পরিমাণ কোড এবং প্রিন্ট লেআউট অপশনের একটি সহজ সেট ব্যবহার করে ইমেজ প্রিন্টিংকে সক্রিয় করার জন্য একটি যথাযথ ফাংশন সরবরাহ করে।

এই অনুশীলনী দেখাবে v4 সাপোর্ট লাইব্রেরী PrintHelper ক্লাস ব্যবহার করে কীভাবে ইমেজ প্রিন্ট করতে হয়।

ইমেজ প্রিন্ট

অ্যান্ড্রয়েড সাপোর্ট লাইব্রেরী PrintHelper ক্লাস ইমেজের প্রিন্ট করতে একটি সহজ উপায় প্রদান করে। ক্লাসটির একটি একক লেআউট অপশন আছে, setScaleMode(), যা দুইয়ের মধ্যে একটি অপশন দিয়ে আপনাকে প্রিন্ট করতে দেয়:

- SCALE MODE FIT – এই অপশন ইমেজকে আকার প্রদান করে যাতে সম্পূর্ণ ইমেজ পেজটির প্রিন্টযোগ্য এলাকার মধ্যে দেখা যায়।
- SCALE MODE FIT - এই অপশন ইমেজকে পরিমাপ করে যাতে এটা পেজটির সম্পূর্ণ প্রিন্টযোগ্য এলাকা পূর্ণ করে। এই সেটিং পছন্দ করলে বোঝায় যে উপরের বা নীচের কিছু অংশ বা ডান অথবা বাম পাশের কিনারার কিছু অংশ প্রিন্ট হবে না। এই অপশন হচ্ছে ডিফল্ট ভ্যালু যদি না আপনি স্কেল মোড পরিবর্তন করে দেন।

setScaleMode() এর জন্য উভয় স্কেলিং অপশনই ইমেজ ইনট্যাক্ট এর বিদ্যমান অনুপাত ধরে রাখে। নীচের কোড উদাহরণ দেখায় কীভাবে PrintHelper ক্লাসের একটি ইনসটেন্স তৈরী করতে হয়, কীভাবে স্কেলিং অপশন সেট করতে হয় এবং প্রিন্টিং প্রক্রিয়া শুরু করতে হয়:

```
private void doPhotoPrint() {  
    PrintHelper photoPrinter = new PrintHelper(getActivity());  
    photoPrinter.setScaleMode(PrintHelper.SCALE_MODE_FIT);  
    Bitmap bitmap = BitmapFactory.decodeResource(getResources(),  
        R.drawable.droids);  
    photoPrinter.printBitmap("droids.jpg - test print", bitmap);  
}
```

এই মেথডটি একটি মেনু আইটেমের জন্য একশন হিসাবে বলা যেতে পারে। উল্লেখ্য, একশনের জন্য মেনু আইটেম যা সবসময় সাপোর্ট লাভ করে না (যেমন প্রিন্টিং) ওভারলোয়া মেনুতে প্লেস করা উচিত। আরও জানতে Action Bar (<http://developer.android.com/design/patterns/actionbar.html>) ডিজাইন গাইড দেখুন।

printBitmap() কল করার পর, আপনার অ্যাপ থেকে আর কোন একশনের প্রয়োজন হয় না। অ্যান্ড্রয়েড প্রিন্ট ইউজার ইন্টারফেস আবির্ভূত হয়, ইউজারকে একটি প্রিন্টার এবং প্রিন্টিং অপশন বেছে নিতে দেয়। ইউজার এরপর ইমেজটি প্রিন্ট করতে পারে বা একশনটি ক্যানসেল করতে পারে। যদি ইউজার ইমেজ প্রিন্ট করাটা বেছে নেয়, একটি প্রিন্ট জব তৈরী হবে এবং সিস্টেম বারে একটি প্রিন্টিং নোটিফিকেশন দৃশ্যমান হবে।

আপনি যদি শুধু ইমেজ ছাড়াও অতিরিক্ত কোন কনটেন্ট আপনার প্রিন্টআউটে অন্তর্ভুক্ত করতে চান, আপনাকে অবশ্যই একটি প্রিন্ট ডকুমেন্ট তৈরী করতে হবে। প্রিন্টিং এর জন্য ডকুমেন্ট তৈরীর বিষয়ে তথ্য পেতে Printing an HTML Document বা Printing a Custom Document অনুশীলনী দেখুন

HTML ডকুমেন্ট প্রিন্ট

(<http://developer.android.com/training/printing/html-docs.html>)

অ্যান্ড্রয়েডে শুধু একটি ফটোর বাইরেও কনটেন্ট প্রিন্টআউটের জন্য প্রয়োজন একটি প্রিন্ট ডকুমেন্টের ভিতরে টেক্সট এবং গ্রাফিক্স কম্পোজ করা। একটি ডকুমেন্ট কম্পোজ করতে এবং একটি ন্যূনতম কোড দিয়ে এটা প্রিন্ট করতে অ্যান্ড্রয়েড ফ্রেমওয়ার্ক HTML ব্যবহার করার একটি উপায় সরবরাহ করে।

অ্যান্ড্রয়েড ৪.৪ (এপিআই লেভেল ১৯) এর মধ্যে, WebView ক্লাস HTML কনটেন্ট প্রিন্টিং সক্রিয় করতে আপডেট হচ্ছে। ক্লাস একটি স্থানীয় HTML রিসোর্স লোড করতে বা ওয়েব থেকে একটি পেজ ডাউনলোড করতে, একটি প্রিন্ট জব তৈরী করতে এবং অ্যান্ড্রয়েডের প্রিন্ট সার্ভিসের প্রতি এটা হস্তান্তর করতে আপনাকে অনুমোদন করে।

এই অনুশীলনী আপনাকে দেখাবে কীভাবে টেক্সট এবং গ্রাফিক্স ধারণ করা HTML ডকুমেন্ট অতি দ্রুত তৈরী করা যায় এবং এটা প্রিন্ট করতে WebView ব্যবহার করা যায়।

HTML ডকুমেন্ট লোড করুন

WebView দিয়ে একটি HTML ডকুমেন্ট প্রিন্টিং HTML রিসোর্স লোড করা বা একটি স্ট্রিং হিসাবে একটি HTML ডকুমেন্ট তৈরী করার বিষয়গুলোকে অন্তর্ভুক্ত করে। এই অধ্যায় আলোচনা করে কীভাবে একটি HTML স্ট্রিং তৈরী করতে হয় এবং প্রিন্টিং এর জন্য WebView এর মধ্যে এটাকে লোড করে।

এই ভিউ অবজেক্ট একটি একটিভিটি লেআউটের অংশ হিসাবে সাধারণভাবে ব্যবহার হয়ে থাকে। যদি আপনার অ্যাপলিকেশন WebView ব্যবহার করে না থাকে, আপনি ক্লাসের একটি ইনস্টেন্স তৈরী করতে পারেন সুনির্দিষ্টভাবে প্রিন্টিং এর উদ্দেশ্যে। এই কাস্টম প্রিন্ট ভিউ তৈরী করার প্রধান ধাপগুলো হচ্ছে:

1. একটি WebViewClient তৈরী করুন যা HTML রিসোর্স লোড হওয়ার পর একটি প্রিন্ট জব শুরু করে।
2. WebView অবজেক্টের মধ্যে HTML রিসোর্স লোড করা

নিম্নোক্ত কোড নমুনা দেখায় কীভাবে একটি সহজ WebViewClient তৈরী করা যায় এবং অতি দ্রুত চলমান অবস্থায় তৈরী HTML ডকুমেন্ট লোড করা হয়:

```
private WebView mWebView;

private void doWebViewPrint() {
    // Create a WebView object specifically for printing
    WebView webView = new WebView(getActivity());
    webView.setWebViewClient(new WebViewClient() {

        public boolean shouldOverrideUrlLoading(WebView view, String url) {
            return false;
        }

        @Override
        public void onPageFinished(WebView view, String url) {
            Log.i(TAG, "page finished loading " + url);
            createWebPrintJob(view);
            mWebView = null;
        }

    });

    // Generate an HTML document on the fly:
    String htmlDocument = "<html><body><h1>Test Content</h1><p>Testing, " +
        "testing, testing...</p></body></html>";
    webView.loadDataWithBaseURL(null, htmlDocument, "text/HTML", "UTF-8", null);

    // Keep a reference to WebView object until you pass the PrintDocumentAdapter
    // to the PrintManager
    mWebView = webView;
}
```

নোট: পূর্ববর্তী অধ্যায়ে আপনি তৈরী করেছিলেন, সেই WebViewClient এর

onPageFinished() মেথডের মধ্যে হওয়া একটি প্রিন্ট জব তৈরী করার জন্য আপনার কল নিশ্চিত করুন। যতক্ষণ পেজ লোডিং শেষ না হচ্ছে ততক্ষণ যদি আপনি অপেক্ষা করতে না পারেন, প্রিন্ট এর আউটপুট অসম্পূর্ণ বা খালি হবে বা সম্পূর্ণভাবে ব্যর্থ হবে।

নোট: উপরের উদাহরণ কোড WebView একটি ইনসটেন্স ধরে রাখে যাতে প্রিন্ট জব তৈরী হওয়ার পূর্বে এটা যাতে গারবেজ কালেকশন না হয়। এটা নিশ্চিত করুন যে আপনি আপনার নিজস্ব বাস্তবায়ন একই ভাবে করছেন অন্যথায় প্রিন্ট প্রক্রিয়া ব্যর্থ হতে পারে।

আপনি যদি পেজে গ্রাফিক্স অন্তর্ভুক্ত করতে চান, আপনার প্রজেক্টের assets/ ডিরেক্টরীর মধ্যে গ্রাফিক ফাইল প্লেস করুন এবং loadDataWithBaseUrl() মেথডের প্রথম প্যারামিটারের মধ্যে একটি বেজ URL সুনির্দিষ্ট করুন, যেভাবে নীচের কোড উদাহরণে দেখানো হয়েছে:

```
webView.loadDataWithBaseUrl("file:///android_asset/images/"
    htmlBody
        "text/HTML"
        "UTF-8"
        null);
```

আপনি loadDataWithBaseUrl() এর মেথডের সাথে loadUrl() বদল করার মাধ্যমে প্রিন্টিং এর জন্য একটি ওয়েব পেজ লোডও করতে পারেন, যেভাবে নীচে দেখানো হলো।

```
// Print an existing web page (remember to request INTERNET permission!):
webView.loadUrl("http://developer.android.com/about/index.html");
```

যখন আপনি প্রিন্ট ডকুমেন্ট তৈরী করার জন্য WebView ব্যবহার করে থাকেন, আপনাকে নীচের সীমাবদ্ধতা বিষয়ে সতর্ক থাকতে হবে:

1. আপনি পেজে হেডার, ফুটার যুক্ত করা বা পেজ নাম্বার যুক্ত করতে পারবেন না
2. HTML ডকুমেন্টের জন্য প্রিন্টিং অপশনগুলো প্রিন্ট পেজ রেঞ্জকে অন্তর্ভুক্ত করে না, যেমন: ১০ পেজের HTML ডকুমেন্টের ২ থেকে ৪ পর্যন্ত প্রিন্ট করা হবে, এই বিষয়গুলো সাপোর্ট করা হয় না।
3. WebView একটি ইনসটেন্স এক সাথে শুধুমাত্র একটি প্রিন্ট জব প্রক্রিয়া করতে পারে
4. HTML ডকুমেন্ট CSS প্রিন্ট এট্রিবিউট ধারণ করে, যেমন ল্যান্ডস্কেপ প্রপারটি সাপোর্ট করা হয় না
5. প্রিন্টিং শুরু করতে একটি HTML ডকুমেন্ট এর মধ্যে JavaScript ব্যবহার করতে পারবেন না

নোট: একটি ডবনঠরবি এর কনটেন্ট যা একটি লেআউটে অন্তর্ভুক্ত, এটা প্রিন্টেডও হতে পারে যখনই এটা একটি ডকুমেন্ট লোড করে।

আপনি যদি আরও পরিমিত প্রিন্ট আউটপুট তৈরী করতে চান এবং প্রিন্ট হওয়া পেজের উপরে ড্র

করা কনটেন্টের উপর সম্পূর্ণ নিয়ন্ত্রণ রাখতে চান: পরবর্তী অনুশীলনীতে (Printing a Custom Document) চলে যান।

প্রিন্ট জব তৈরী করুন

একটি WebView তৈরী এবং আপনার HTML কনটেন্ট লোড করার পর, আপনার অ্যাপলিকেশন এর প্রিন্টিং প্রক্রিয়ার অংশটুকু প্রায় সমাপ্ত করে ফেলেছেন। পরবর্তী ধাপ হচ্ছে PrintManager প্রবেশ, একটি প্রিন্ট অ্যাডাপটার তৈরী, এবং সর্বশেষে একটি প্রিন্ট জব তৈরী করা। নীচের উদাহরণ ব্যাখ্যা করে কীভাবে এই ধাপগুলো সম্পাদন করতে হয়:

```
private void createWebPrintJob(WebView webView) {  
  
    // Get a PrintManager instance  
    PrintManager printManager = (PrintManager) getActivity()  
        .getSystemService(Context.PRINT_SERVICE);  
  
    // Get a print adapter instance  
    PrintDocumentAdapter printAdapter = webView.createPrintDocumentAdapter();  
  
    // Create a print job with name and adapter instance  
    String jobName = getString(R.string.app_name) + " Document";  
    PrintJob printJob = printManager.print(jobName, printAdapter,  
        new PrintAttributes.Builder().build());  
  
    // Save the job object for later status checking  
    mPrintJobs.add(printJob);  
}
```

এই উদাহরণ অ্যাপলিকেশন দ্বারা ব্যবহার করার জন্য PrintJob অবজেক্টের একটি ইনস্ট্যান্সকে সেভ করে, যার কোন প্রয়োজন নেই। আপনার অ্যাপলিকেশন প্রিন্ট জবের অগ্রগতি অনুসরণ করতে এই অবজেক্ট ব্যবহার করতে পারে যে এটা প্রক্রিয়াজাত হচ্ছে। এই অ্যাপ্রোচ উপকারী তখনই যখন আপনি আপনার অ্যাপলিকেশনের প্রিন্ট জবের অবস্থা মনিটরিং করতে চান যেমন, কাজটা শেষ হয়েছে কিনা, অকার্যকর হয়েছে কিনা বা ইউজার ক্যানসেল করেছে কিনা। ইন-অ্যাপ নোটিফিকেশন তৈরী করার কোন প্রয়োজন নেই, কারণ প্রিন্ট ফ্রেমওয়ার্ক স্বয়ংক্রিয়ভাবে প্রিন্ট জবের জন্য একটি সিস্টেম নোটিফিকেশন তৈরী করে দেয়।

কাস্টম ডকুমেন্টস প্রিন্ট

(<http://developer.android.com/training/printing/custom-docs.html>)

কিছু অ্যাপলিকেশনের জন্য, যেমন ড্রয়িং অ্যাপস, পেজ লেআউট অ্যাপস এবং অন্যান্য অ্যাপস যা গ্রাফিক আউটপুটকে ফোকাস করে, সুন্দর প্রিন্ট করা পেজ তৈরী করা হচ্ছে প্রধান বৈশিষ্ট্য। এই ক্ষেত্রে একটি ইমেজ বা একটি HTML ডকুমেন্ট প্রিন্ট করাটাই যথেষ্ট নয়। এই ধরনের অ্যাপলিকেশনের প্রিন্ট আউটপুটের জন্য একটি পেজের মধ্যে যা কিছু যায় তার সব কিছুর উপর নির্দিষ্ট নিয়ন্ত্রন প্রয়োজন, যার মধ্যে রয়েছে ফন্ট, টেক্সট ফ্লো, পেজ ব্রেক, হেডার, ফুটার এবং গ্রাফিক এলিমেন্ট।

একটি প্রিন্ট আউটপুট তৈরী করা যা সম্পূর্ণভাবে আপনার অ্যাপলিকেশনের জন্য প্রযোজ্য করার জন্য প্রয়োজন পূর্ববর্তী আলোচনার অ্যাপ্রোচের চেয়ে অনেক বেশী প্রোগামিং বিনিয়োগ করে। আপনাকে অবশ্যই কম্পোনেন্ট তৈরী করতে হবে যা প্রিন্ট ফ্রেমওয়ার্কের সাথে যোগাযোগ করতে পারে, প্রিন্টার সেটিং এর সাথে খাপ খায়, পেজ এলিমেন্ট ড্র করে, বহুমুখি পেজে প্রিন্টিং পরিচালনা করা।

এই অনুশীলনী আপনাকে দেখায় কীভাবে আপনি প্রিন্ট ম্যানেজারের সাথে যুক্ত হতে পারবেন, একটি প্রিন্ট এডাপটর তৈরী করতে পারবেন এবং প্রিন্টারের জন্য কনটেন্ট বানাতে পারবেন।

প্রিন্ট ম্যানেজারের সাথে যুক্ত করুন

যখন আপনার অ্যাপলিকেশন সরাসরি প্রিন্টিং প্রক্রিয়া চালিত করে, আপনার ইউজারের কাছ থেকে একটি প্রিন্ট রিকোয়েস্ট রিসিভ করার পর প্রথম ধাপ হচ্ছে অ্যান্ড্রয়েড প্রিন্ট ফ্রেমওয়ার্কে যুক্ত করা এবং `PrintManager` ক্লাসের একটি ইনসটেন্স লাভ করা। এই ক্লাস আপনাকে একটি প্রিন্ট জব আরম্ভ করতে এবং প্রিন্টিং লাইফসাইকেল শুরু করতে অনুমোদন দেয়। নিম্নোক্ত কোড উদাহরণ আপনাকে দেখায় কীভাবে প্রিন্ট ম্যানেজার পেতে হয় এবং প্রিন্ট প্রক্রিয়া শুরু করতে হয়।

```
private void doPrint() {  
    // Get a PrintManager instance  
    PrintManager printManager = (PrintManager) getActivity()  
        .getSystemService(Context.PRINT_SERVICE);  
  
    // Set job name, which will be displayed in the print queue  
    String jobName = getActivity().getString(R.string.app_name) + " Document";  
  
    // Start a print job, passing in a PrintDocumentAdapter implementation  
    // to handle the generation of a print document  
    printManager.print(jobName, new MyPrintDocumentAdapter(getActivity()),  
        null);  
}
```

উপরের কোড উদাহরণ দেখায় কীভাবে একটি প্রিন্ট জবের নাম দিতে হয় এবং `PrintDocumentAdapter` ক্লাসের একটি ইনসটেন্স সেট করতে হয় যা প্রিন্টিং লাইফসাইকেলের ধাপগুলো পরিচালনা করে। প্রিন্ট এডাপটর ক্লাসের বাস্তবায়ন পরবর্তী সেকশনে আলোচনা করা হয়েছে।

নোট: `print()` মেথডের শেষ প্যারামিটার একটি `PrintAttributes` অবজেক্ট নেয়। পূর্ববর্তী প্রিন্টিং সাইকেল এর ভিত্তি করে প্রিন্টিং ফ্রেমওয়ার্কে এবং প্রি-সেট অপশনে হিন্টস প্রদান করতে আপনি এই প্যারামিটার ব্যবহার করতে পারেন, এই উপায়ে ইউজার এক্সপেরিয়েন্স উন্নত করে। আপনি অপশন সেট করতেও প্যারামিটার ব্যবহার করতে পারেন যেটা যে কনটেন্ট প্রিন্ট হবে তার জন্য বেশী উপযুক্ত, যেমন ল্যান্ডস্কেপে ওরিয়েন্টেশন সেট করুন যখন একটি ফটো প্রিন্ট করবেন যা ঐ ওরিয়েন্টেশনে আছে।

একটি প্রিন্ট অ্যাডাপটার তৈরী করুন

একটি প্রিন্ট অ্যাডাপটার অ্যান্ড্রয়েড প্রিন্ট ফ্রেমওয়ার্কের সাথে যোগাযোগ করে এবং প্রিন্টিং প্রক্রিয়ার ধাপগুলো পরিচালনা করে। এই প্রক্রিয়ার জন্য প্রয়োজন প্রিন্ট করার জন্য একটি ডকুমেন্ট তৈরী করার পূর্বে ইউজার কর্তৃক প্রিন্টার এবং প্রিন্ট অপশন নির্বাচন করা। এই নির্বাচন চূড়ান্ত আউটপুটকে প্রভাবিত করতে পারে যেহেতু ইউজার ভিন্ন আউটপুট সামর্থ্য, ভিন্ন পেজ সাইজ, বা ভিন্ন পেজ ওরিয়েন্টেশন দিয়ে প্রিন্টার পছন্দ করে। যেহেতু এই নির্বাচন তৈরী করা হয়েছে, প্রিন্ট ফ্রেমওয়ার্ক আপনার প্রিন্ট অ্যাডাপটরকে একটি প্রিন্ট ডকুমেন্ট পরিকল্পনা করতে বলে এবং এটা তৈরী করতে বলে। যখনই একজন ইউজার প্রিন্ট বাটন চাপে, ফ্রেমওয়ার্ক চূড়ান্ত প্রিন্ট ডকুমেন্ট নেয় এবং এটাকে প্রিন্ট প্রভাইডারে আউটপুটের জন্য পাঠায়। প্রিন্টিং প্রক্রিয়া সময়কালে ইউজার প্রিন্ট একশন ক্যানসেল করতে পারে, সুতরাং আপনার প্রিন্ট অ্যাডাপটরকে অবশ্যই একটি ক্যানসেল করা রিকোয়েস্ট শুনতে হবে এবং এর প্রতি প্রতিক্রিয়া করতে হবে।

`PrintDocumentAdapter` অ্যাবস্ট্রাক্ট ক্লাস প্রিন্টার লাইফসাইকেল পরিচালনা করার জন্য ডিজাইন করা হয়েছে, যার চারটি প্রধান কলব্যাক মেথড আছে। আপনাকে অবশ্যই প্রিন্ট ফ্রেমওয়ার্কের সাথে যথাযথভাবে যোগাযোগ করার জন্য আপনার প্রিন্ট অ্যাডাপটরের সাথে এই মেথডগুলো বাস্তবায়ন করতে হবে:

- `onStart()` – প্রিন্ট প্রক্রিয়ার শুরুতে শুধু একবার কল করা হয়। যদি আপনার অ্যাপলিকেশনের সম্পাদন করার জন্য যে কোন টাইম-প্রিপারেশন কাজ থাকে, যেমন যে ডাটা প্রিন্ট করতে হবে তার একটি -ক্যাশপাশট পাওয়া, সেগুলোকে এখানে সম্পাদন করুন। আপনার অ্যাডাপটরে এই মেথড বাস্তবায়ন করার কোন প্রয়োজন নেই।
- `onLayout()` – একজন ইউজার যতবার একটি প্রিন্ট সেটিং পরিবর্তন করে, প্রতিবারই এটা কল করা হয়, যেমন একটি ভিন্ন পেজ সাইজ, বা পেজ ওরিয়েন্টেশন, আপনার অ্যাপলিকেশনকে যে পেজ প্রিন্ট হবে তার একটি লেআউট গণনা করার একটি সুযোগ দেয়। ন্যূনতমভাবে, এই মেথড অবশ্যই প্রিন্ট করা ডকুমেন্ট এর মধ্যে কতগুলো পেজ আশা করে তা রিটার্ন করে।
- `onWrite()` – প্রিন্ট হবে এমন ফাইলের মধ্যে রেন্ডার প্রিন্ট পেজে কল করা হয়। এই মেথডকে প্রতিটা `onLayout()` কলের পর এক বা একাধিক বার কল হতে পারে।
- `onFinish()` – প্রিন্ট প্রক্রিয়ার শেষে একবার করা হয়। যদি আপনার অ্যাপলিকেশনের কোন ওয়ান টাইম টিয়ার-ডাউন কাজ থাকে, এখানে সম্পাদন করুন। আপনার অ্যাডাপটরে এই মেথড বাস্তবায়ন করার কোন প্রয়োজন নেই।

নিচের অধ্যায় আলোচনা করে কীভাবে লেআউট এবং রাইট মেথড বাস্তবায়ন করতে হয়, যা একটি প্রিন্ট অ্যাডাপটরে কাজ করা খুব জটিল।

নোট: এই অ্যাডাপটর মেথডকে আপনার অ্যাপলিকেশনের মেইন থ্রেডে কল করা হয়। যদি আপনি আপনার অ্যাপলিকেশনে এই মেথডগুলো কাজ করাতে উল্লেখযোগ্য পরিমাণ সময় নিতে

আশা করেন, তাদের কাজ করাতে একটি পৃথক থ্রেডে বাস্তবায়ন করুন। উদাহরনস্বরূপ, আপান পৃথক `AsyncTask` শঅবজেক্টে লেআউট বা প্রিন্ট ডকুমেণ্ট লেখার কাজ সংযুক্ত করতে পারেন।

প্রিন্ট ডকুমেন্ট ইনফো ক্যাপচার করুন

PrintDocumentAapter ক্লাসের একটি বাস্তবায়নের মধ্যে, আপনার অ্যাপলিকেশন অবশ্যই ডকুমেন্টের ধরন নির্দিষ্ট করতে সমর্থ হবে যা এটা তৈরী করছে এবং প্রিন্ট জবের জন্য সর্বমোট পেজের সংখ্যা গণনা করে, প্রিন্ট করা পেজের সাইজের তথ্য প্রদান করে। অ্যাডাপ্টরের onLayout()মেথডের বাস্তবায়ন একটি PrintDocumentInfo ক্লাস এর মধ্যে প্রিন্ট জবের কাঙ্ক্ষিত আউটপুট সম্পর্কিত এই গণনাগুলো তৈরী এবং তথ্য প্রদান করে, পেজের সংখ্যা এবং কনটেন্ট টাইপ সহ। নিম্নোক্ত কোড উদাহরণ একটি PrintDocumentAdapter এর জন্য onLayout()মেথডের মৌলিক বাস্তবায়ন দেখায়:

```
@Override
public void onLayout(PrintAttributes oldAttributes,
                    PrintAttributes newAttributes,
                    CancellationSignal cancellationSignal,
                    LayoutResultCallback callback,
                    Bundle metadata) {
    // Create a new PdfDocument with the requested page attributes
    mPdfDocument = new PrintedPdfDocument(getActivity(), newAttributes);

    // Respond to cancellation request
    if (cancellationSignal.isCancelled() ) {
        callback.onLayoutCancelled();
        return;
    }

    // Compute the expected number of printed pages
    int pages = computePageCount(newAttributes);

    if (pages > 0) {
        // Return print information to print framework
        PrintDocumentInfo info = new PrintDocumentInfo
            .Builder("print_output.pdf")
            .setContentType(PrintDocumentInfo.CONTENT_TYPE_DOCUMENT)
            .setPageCount(pages);
        .build();
        // Content layout reflow is complete
        callback.onLayoutFinished(info, true);
    } else {
        // Otherwise report an error to the print framework
        callback.onLayoutFailed("Page count calculation failed.");
    }
}
```

onLayout() মেথড সম্পাদনের তিনটা ফলাফল থাকে: কমপ্লিশন (শেষ করা), ক্যানসেলেশন (বাতিল করা) বা ফেইলিওর (অসফল) এর ঘটনা যেখানে লেআউটের ক্যালকুলেশন (গণনা) সমাপ্ত হয় না। PrintDocumentAdapter.LayoutResultCallback অবজেক্টের যথাযথ মেথডটি কল করার মাধ্যমে আপনাকে অবশ্যই এই ফলাফলগুলোর কোন একটি নির্দেশ করতে হবে।

নোট: onLayoutFinished()মেথডের বুলিয়ান প্যারামিটার লেআউট কনটেন্ট আসলেই পরিবর্তিত হয়েছে কিনা তা নির্দেশ করে শেষ রিকোয়েস্ট থেকে। এই প্যারামিটার যথাযথভাবে সেট করতে প্রিন্ট ফ্রেমওয়ার্ককে onWrite()মেথড অপ্রয়োজনীয়ভাবে কল করা, পূর্বে লেখা প্রিন্ট ডকুমেন্ট

অপারহাৰ্ষভাবে জামিয়ে রাখা এবং পারফরমেন্স উন্নয়ন করা থেকে বিরত রাখতে দিন।

onLayout()এর প্রধান কাজ হচ্ছে পেজের সংখ্যার ক্যালকুলেশন করা যা আউটপুট হিসাবে আকাঙ্ক্ষা করা হয় প্রিন্টের এট্রিবিউট দেয়। কীভাবে আপনি এই নাম্বার ক্যালকুলেট করবেন তা কীভাবে আপনার অ্যাপলিকেশন প্রিন্টিং এর জন্য পেজ বিন্যাস করা হয় তার উপর সম্পূর্ণভাবে নির্ভরশীল। নীচের কোড উদাহরন একটি বাস্তবায়ন দেখায় যেখানে পেজের সংখ্যা প্রিন্ট ওরিয়েন্টেশন কর্তৃক নির্ধারিত হয়ে থাকে:

```
private int computePageCount(PrintAttributes printAttributes) {
    int itemsPerPage = 4; // default item count for portrait mode

    MediaSize pageSize = printAttributes.getMediaSize();
    if (!pageSize.isPortrait()) {
        // Six items per page in landscape orientation
        itemsPerPage = 6;
    }

    // Determine number of print items
    int printItemCount = getPrintItemCount();

    return (int) Math.ceil(printItemCount / itemsPerPage);
}
```


একটি প্রিন্ট ডকুমেন্ট ফাইল লেখা (রাইট)

একটি ফাইলে প্রিন্ট আউটপুট লেখার সময়, অ্যান্ড্রয়েড প্রিন্ট ফ্রেমওয়ার্ক আপনার অ্যাপলিকেশনের `PrintDocumentAdapter` ক্লাসের `onWrite()` মেথড কল করে। মেথডের প্যারামিটার কোন পেজ লেখা উচিত এবং আউটপুট ফাইল ব্যবহৃত হবে তা নির্দিষ্ট করে। আপনার এই মেথডের বাস্তবায়ন অবশ্যই তারপর কনটেন্টের রিকোয়েস্ট করা প্রতিটা পেজ মাল্টি-পেজ PDF ডকুমেন্ট ফাইলে পেশ করতে হবে। যখন এই প্রক্রিয়া সম্পন্ন হবে আপনি কলব্যাক অবজেক্টের `onWriteFinished()` মেথড কল করেন।

নোট: অ্যান্ড্রয়েড প্রিন্ট ফ্রেমওয়ার্ক `onLayout()` এ প্রতিটা কলের জন্য এক বা একাধিক বার `onWrite()` মেথড কল করতে পারে। এই কারণে, `false` এ `onLayoutFinished()` মেথডের বুলিয়ান (boolean) প্যারামিটার সেট করাটা গুরুত্বপূর্ণ যখন প্রিন্ট কনটেন্ট লেআউট পরিবর্তন হয় না, প্রিন্ট কনটেন্টের অপ্রয়োজনীয় রি-রাইট পরিহার করার জন্য।

নোট: `onLayoutFinished()` মেথডের বুলিয়ান প্যারামিটার লেআউট কনটেন্ট আসলেই পরিবর্তিত হয়েছে কিনা তা নির্দেশ করে শেষ রিকোয়েস্ট থেকে অদ্যবধি। এই প্যারামিটার যথাযথভাবে সেট করতে প্রিন্ট ফ্রেমওয়ার্ককে `onLayout()` মেথড অপ্রয়োজনীয়ভাবে কল করা, পূর্বে লেখা প্রিন্ট ডকুমেন্ট অপরিহার্যভাবে জমিয়ে রাখা এবং পারফরমেন্স উন্নয়ন করা থেকে বিরত রাখতে দিন।

নিম্নোক্ত নমুনা একটি PDF ফাইল তৈরী করতে `PrintedPdfDocument` ক্লাস ব্যবহার করে এই প্রক্রিয়ার মৌলিক মেকানিকস দেখায়:

```
@Override
public void onWrite(final PageRange[] pageRanges,
                    final ParcelFileDescriptor destination,
                    final CancellationSignal cancellationSignal,
                    final WriteResultCallback callback) {
    // Iterate over each page of the document,
    // check if it's in the output range.
    for (int i = 0; i < totalPages; i++) {
        // Check to see if this page is in the output range.
        if (containsPage(pageRanges, i)) {
            // If so, add it to writtenPagesArray. writtenPagesArray.size()
            // is used to compute the next output page index.
            writtenPagesArray.append(writtenPagesArray.size(), i);
            PdfDocument.Page page = mPdfDocument.startPage(i);

            // check for cancellation
            if (cancellationSignal.isCancelled()) {
                callback.onWriteCancelled();
                mPdfDocument.close();
                mPdfDocument = null;
                return;
            }

            // Draw page content for printing
            drawPage(page);

            // Rendering is complete, so page can be finalized.
            mPdfDocument.finishPage(page);
        }
    }
}
```

```

    }

    // Write PDF document to file
    try {
        mPdfDocument.writeTo(new FileOutputStream(
            destination.getFileDescriptor()));
    } catch (IOException e) {
        callback.onWriteFailed(e.toString());
        return;
    } finally {
        mPdfDocument.close();
        mPdfDocument = null;
    }
    PageRange[] writtenPages = computeWrittenPages();
    // Signal the print framework the document is complete
    callback.onWriteFinished(writtenPages);

    ...
}

```

এই নমুনা drawPage() মেথডে PDF পেজ কনটেন্টের রেন্ডারিং হস্তান্তর করে, যা পরবর্তী অধ্যায়ে আলোচনা করা হবে।

যেহেতু লেআউটের সাথে, onWrite() মেথড কার্যসম্পাদনের তিনটা ফলাফল থাকতে পারে: কমপ্লিশন (শেষ করা), ক্যানসেলেশন (বাতিল করা) বা ফেইলিওর (অসফল) এর ঘটনা যেখানে কনটেন্ট লেখা হয় না। PrintDocumentAdapter.WriteResultCallback অবজেক্টের যথাযথ মেথডটি কল করার মাধ্যমে আপনাকে অবশ্যই এই ফলাফলগুলোর কোন একটি নির্দেশ করতে হবে।

নোট: প্রিন্টিং এর জন্য একটি ডকুমেন্ট রেন্ডারিং একটি রিসোর্স-ইনটেনসিভ অপারেশন হতে পারে। আপনার অ্যাপলিকেশনের প্রধান ইউজার ইন্টারফেস থ্রেড ব্লক করাকে পরিহার করার জন্য, আপনার উচিত একটি পৃথক থ্রেডে পেজ রেন্ডারিং বা রাইটিং অপারেশন সম্পান করার বিষয়টা বিবেচনা করা উচিত, উদাহরণস্বরূপ, একটি AsyncTask এর মধ্যে। আসিঙ্ক্রোনাস (asynchronous) কাজের মতো এক্সিকিউশন থ্রেডের সাথে কাজ করা সম্পর্কিত আরও তথ্য জানতে, Processes and Threads (<http://developer.android.com/guide/components/processes-and-threads.html>) দেখুন।

PDF পেজ কনটেন্ট ড্র করা

যখন আপনার অ্যাপলিকেশন প্রিন্ট করে, আপনার অ্যাপলিকেশন অবশ্যই একটি PDF ডকুমেন্ট তৈরী করে এবং প্রিন্ট করার জন্য এটা অ্যান্ড্রয়েড প্রিন্ট ফ্রেমওয়ার্কে পাঠিয়ে দেয়। এই উদ্দেশ্যে আপনি যে কোন PDF জেনারেশন রাইব্রেরী ব্যবহার করতে পারেন। এই অনুশীলনী দেখায় আপনার কনটেন্ট থেকে PDF পেজ তৈরী করতে কীভাবে `PrintedPdfDocument` ক্লাস ব্যবহার করতে হয়।

`PrintedPdfDocument` ক্লাস একটি PDF পেজে এলিমেন্ট ড্র করতে একটি `Canvas` অবজেক্ট ব্যবহার করে, একটা একটিভিটি লেআউটে ড্র করার মতো করে। ড্র `Canvas` মেথড ব্যবহার করে প্রিন্ট করা পেজে আপনি এলিমেন্ট ড্র করতে পারেন। নিম্নোক্ত উদাহরণ কোড দেখায় এই মেথড গুলো ব্যবহার করে একটি PDF ডকুমেন্টে কীভাবে কিছু সহজ এলিমেন্ট ড্র করতে হয়:

```
private void drawPage(PdfDocument.Page page) {
    Canvas canvas = page.getCanvas();

    // units are in points (1/72 of an inch)
    int titleBaseLine = 72;
    int leftMargin = 54;

    Paint paint = new Paint();
    paint.setColor(Color.BLACK);
    paint.setTextSize(36);
    canvas.drawText("Test Title", leftMargin, titleBaseLine, paint);

    paint.setTextSize(11);
    canvas.drawText("Test paragraph", leftMargin, titleBaseLine + 25, paint);

    paint.setColor(Color.BLUE);
    canvas.drawRect(100, 100, 172, 172, paint);
}
```

যখন PDF পেজে ড্র করতে `Canvas` ব্যবহার করা হয়, এলিমেন্ট পয়েন্টে নির্দিষ্ট হয়, যা এক ইঞ্চির ১/৭২। নিশ্চিত করুন পেজে এলিমেন্টের সাইজ নির্দিষ্ট করার জন্য আপনি এই পরিমাপের একক ব্যবহার করছেন। ড্র করা এলিমেন্টের অবস্থানের ঠিক করার জন্য, সমন্বয়কারী সিস্টেম ০ থেকে শুরু করে, পেজের উপরের বাম কোণের জন্য ০।

টিপ: যখন ঈধহাধং অবজেক্ট একটি PDF ডকুমেন্টের সীমানায় আপনাকে প্রিন্ট এলিমেন্ট রাখতে দেয়, অনেক প্রিন্টার পেপারের কোণের অংশটি প্রিন্ট দিতে সমর্থ হয় না। নিশ্চিত করুন যে আপনি পেজের আনপ্রিন্টেবল কোনা (এজ) বন্ধ করেছেন, যখন আপনি এই ক্লাস দিয়ে একটি প্রিন্ট ডকুমেন্ট তৈরী করেন।

গ্রাফিক্স এবং অ্যানিমেশন দিয়ে অ্যাপস তৈরী

(<http://developer.android.com/training/building-graphics.html>)

এই ক্লাস আপনাকে শেখাবে কীভাবে গ্রাফিক্স দিয়ে কার্য সম্পাদন করতে হয় যা আপনার অ্যাপকে প্রতিযোগিতায় কিছু সুবিধা প্রদান করতে পারে। আপনি যদি মৌলিক ইউজার ইন্টারফেসের সীমা ছাড়িয়ে একটি সুন্দর ভিজ্যুয়াল এক্সপেরিয়েন্স তৈরী করতে চান, এই ক্লাসগুলো আপনাকে সেখানে পৌঁছে দিতে সহায়তা করবে।

১. দক্ষতার সাথে বিটম্যাপ প্রদর্শন

কীভাবে বিটম্যাপ লোড এবং প্রক্রিয়াজাত করতে হয় যখন আপনার ইন্টারফেসকে সংবেদনশীল (রেসপনসিভ) রাখা হয় এবং মেমরীর মাত্রাকে ছাড়িয়ে যাওয়াকে পরিহার করে

1. বড় আকারের বিটম্যাপ লোড করা
2. ইউআই থ্রেড অফ করতে বিটম্যাপ প্রসেস করা
3. বিটম্যাপ জমিয়ে (Caching) রাখা
4. বিটম্যাপ মেমরী পরিচালনা করা
5. আপনার ইউজার ইন্টারফেসে বিটম্যাপ প্রদর্শন

২. OpenGL ES সহকারে গ্রাফিক্স প্রদর্শন

অ্যান্ড্রয়েড অ্যাপ ফ্রেমওয়ার্কের মধ্যে কীভাবে গুটবহএথ গ্রাফিক্স তৈরী করতে হয় এবং টাচ ইনপুটে রেসপন্স করতে হয়।

1. OpenGL ES পরিবেশ তৈরী করা
2. আকার (শেপ) নির্ণয়
3. আকার (শেপ) ড্র করা
4. প্রজেক্ট এবং ক্যামেরা ভিউ আবেদন
5. মোশন যুক্ত করা
6. টাচ ইভেন্টে রেসপন্স করা

৩. অ্যানিমেশন যুক্ত করা

কীভাবে আপনার ইউজার ইন্টারফেসে পরিবর্তন সূচক অ্যানিমেশন যুক্ত করতে হয়।

1. দুইটা ভিউ ক্রসফেড করুন (একটার পর আরেকটি আসা)
2. স্ক্রিন স্লাইডের জন্য ভিউ পেজার ব্যবহার করা
3. কার্ড ফ্লিপ অ্যানিমেশন প্রদর্শন
4. ভিউ জুম করা
5. লেআউট পরিবর্তন অ্যানিমেশন করা

দক্ষতার সাথে বিটম্যাপ প্রদর্শন

(<http://developer.android.com/training/displaying-bitmaps/index.html>)

শিখুন কীভাবে সাধারণ কৌশল ব্যবহার করে Bitmap অবজেক্ট লোড এবং প্রক্রিয়াজাত করতে হয় যা আপনার ইন্টারফেসকে সংবেদনশীল (রেসপনসিভ) রাখা হয় এবং মেমরীর মাত্রাকে ছাড়িয়ে যাওয়াকে পরিহার করে। আপনি যদি সতর্ক না হোন, বিটম্যাপ আপনার বিদ্যমান মেমরী বাজেট খুব দ্রুত খরচ করে ফেলতে পারে যা ড্রেড এক্সপেশনের কারণে একটি অ্যাপলিকেশনকে ধ্বংস হওয়ার পথে নিয়ে যায়:

```
java.lang.OutOfMemoryError: bitmap size exceeds VM budget.
```

এখানে কিছু কারন আছে যেটা দেখায় আপনার অ্যান্ড্রয়েডে অ্যাপলিকেশনে বিটম্যাপ লোড করা কেন জটিল:

- মোবাইল ডিভাইসের সাধারণভাবে দমিত সিস্টেম রিসোর্স আছে। অ্যান্ড্রয়েড ডিভাইসে একটি একক অ্যাপলিকেশনে ১৬ গই এর মতো ছোট মেমরী থাকতে পারে। Android Compatibility Definition Document (CDD), সেকশন ৩.৭. *Virtual Machine Compatibility* বিভিন্ন স্ক্রিন সাইজ এবং ঘনত্ব এর জন্য প্রয়োজনীয় ন্যূনতম অ্যাপলিকেশন মেমরী দেয়। অ্যাপলিকেশন এই ন্যূনতম মেমরী সীমার মধ্যে থেকে সম্পাদন করার মতো করে অপটিমাইজ করা উচিত। কিন্তু মনে রাখবেন অনেক ডিভাইস উচ্চ মাত্রায় কনফিগার করা হয়ে থাকে।
- বিটম্যাপ অনেক মেমরী নিয়ে নেয়, বিশেষ করে ফটোগ্রাফের মতো উচ্চ মাত্রার ইমেজের জন্য। উদাহরণস্বরূপ, Galaxy Nexus এ ক্যামেরা 2592x1936 pixels (5 megapixels) পর্যন্ত ফটো নিয়ে থাকে। যদি বিটম্যাপ কনফিগারেশন ARGB 8888 ব্যবহার করে থাকে (অ্যান্ড্রয়েড ২.৩ ও আরও অগ্রসর থেকে ডিফল্ট) তখন মেমরীতে এই ইমেজ লোড করতে মেমরীর প্রায় ১৯ MB নেয় (2592*1936*4 bytes), কিছু ডিভাইসে প্রি-অ্যাপ লিমিট তাৎক্ষনিকভাবে বের করে দেয়।
- অ্যান্ড্রয়েড অ্যাপ ইউজার ইন্টারফেস নিয়মিতভাবে একই সাথে কতিপয় বিটম্যাপের লোড হওয়ার আকাঙ্ক্ষা করে। কম্পোনেন্ট যেমন, ListView, GridView এবং ViewPager সাধারণভাবে একই সাথে বহুখুঁচি বিটম্যাপ অন-স্ক্রিনে অন্তর্ভুক্ত কওে, শুধু একটি আঙ্গুলের টোকায় সম্ভাবনাময় অনেক অফ-স্ক্রিন প্রদর্শন হওয়ার জন্য প্রস্তুত আছে।

অনুশীলনীসমূহ

বড় আকারের বিটম্যাপ লোড করা

এই অনুশীলনী প্রতি অ্যাপলিকেশনের মেমরী লিমিট অতিক্রম না করেই বড় আকারের বিটম্যাপ ডিকোড করতে হয় কীভাবে তা আপনাকে দেখাবে

বিটম্যাপ প্রক্রিয়া ইউআই থ্রেড বন্ধ করে

বিটম্যাপ প্রক্রিয়া (রিসাইজ করা, একটি দূরবর্তী সোর্স থেকে ডাউনলোড করা, ইত্যাদী) করা কখনই ইউজার ইন্টারফেস থ্রেডের জায়গা নেওয়া উচিত না। এই অনুশীলনী আপনাকে দেখাবে AsyncTask ব্যবহার করে একটি ব্যাকগ্রাউন্ড থ্রেডের মধ্যে বিটম্যাপ প্রসেস করতে হয় এবং ব্যাখ্যা করে কীভাবে কনকারেন্সি ইস্যু পরিচালনা করতে হয়।

বিটম্যাপ জমিয়ে রাখা

এই অনুশীলনী আপনাকে দেখাবে যখন মাল্টিপল বিটম্যাপ লোড করা হয় তখন আপনার ইউজার ইন্টারফেসের রেসপন্সিভনেস এবং ফুলয়িডিটি উন্নত করতে কীভাবে মেমরী এবং ডিস্ক বিটম্যাপ ক্যাশে ব্যবহার করতে হয়

বিটম্যাপ মেমরী পরিচালনা করা

এই অনুশীলনী ব্যাখ্যা করে আপনার অ্যাপ পারফরমেন্স বাড়াতে কীভাবে বিটম্যাপ মেমরী পরিচালনা করতে হয়।

আপনার ইউজার ইন্টারফেসে বিটম্যাপ প্রদর্শন

এই অনুশীলনী সব কিছু একসাথে নিয়ে আসে, আপনাকে দেখাবে একটি ব্যাকগ্রাউন্ড থ্রেড এবং বিটম্যাপ ক্যাশে ব্যবহার করে ViewPager এবং GridView মতো কম্পোনেন্টের মধ্যে কীভাবে মাল্টিপল বিটম্যাপ লোড করতে হয়

বড় আকারের বিটম্যাপ দক্ষতার সাথে লোড করা

(<http://developer.android.com/training/displaying-bitmaps/load-bitmap.html>)

ইমেজ সকল আকার ও গঠন নিয়ে আসতে পারে। অনেক ক্ষেত্রে একটি সাধারণ অ্যাপলিকেশন ইউজার ইন্টারফেসের জন্য প্রয়োজনের চেয়ে বড় হতে পারে। উদাহরণস্বরূপ, সিস্টেম গ্যালারী অ্যাপলিকেশন আপনার অ্যান্ড্রয়েড ডিভাইসের ক্যামেরা ব্যবহার করে নেয়া ফটো প্রদর্শন করে যা সাধারণত আপনার স্ক্রিনের ঘনত্বের চেয়েও উচ্চ রেজ্যুলেশন যুক্ত।

দেয়া আছে যে আপনি সীমিত মেমরী নিয়ে কাজ করছেন, এটাই আদর্শ যে আপনি শুধু মেমরীতে কম রেজ্যুলেশনের সংস্করন লোড করবেন। কম রেজ্যুলেশনের সংস্করন আপনার ইউজার ইন্টারফেসের কম্পোনেন্ট যা এটা প্রদর্শন করে তার সাথে সাথে ম্যাচ করা উচিত। একটি উচ্চ রেজ্যুলেশন যুক্ত ইমেজ কোন দৃশ্যমান সুবিধা প্রদান করে না, কিনুত এখনও মূল্যবান মেমরী নিয়ে নেয় এবং ফ্লাই স্কেলিং এ যুক্ততার কারনে অতিরিক্ত পারফরমেন্স ওভারহেড ডেকে আনে।

এই অনুশীলনী মেমরীতে একটি ছোট সাবস্যাম্পলড সংস্করন লোড করার মাধ্যমে প্রতি অ্যাপলিকেশনের মেমরীর সীমা অতিক্রম না করে বড় আকারের বিটম্যাপ ডিকোডিং করার কাজটি দেখাবে।

বিটম্যাপের মাত্রা (ডাইমেনশন) এবং টাইপ পড়ুন

বিভিন্ন উৎস থেকে একটি Bitmap তৈরী করার জন্য BitmapFactory ক্লাস কিছু ডিকোডিং মেথড (decodeByteArray(), decodeFile(), decodeResource() ইত্যাদী) সরবরাহ করে। আপনার ইমেজ ডাটা সোর্সের উপর ভিত্তি করে সবচেয়ে উপযুক্ত মেথড পছন্দ করুন। এই মেথডগুলো গঠন হওয়া বিটম্যাপের জন্য মেমরী বরাদ্দের পদক্ষেপ নেয় এবং সহজেই একটি OutOfMemory এক্সেপশনের মধ্যে ফলাফল নিয়ে আসে। ডিকোডের প্রতিটা ধরনের অতিরিক্ত সিগনেচার থাকে যা আপনাকে BitmapFactory.Options ক্লাসের মধ্য দিয়ে ডিকোডিং অপশন নির্দিষ্ট করতে দেয়। true এ inJustDecodeBounds প্রপার্টী সেট করা যখন ডিকোডিং মেমরী বরাদ্দ (এলোকেশন) পরিহার করে, বিটম্যাপ অবজেক্টের জন্য ইঁষষ ফেরত আনে কিনুত outWidth, outHeight এবং outMimeType সেট করে। এই কৌশল বিটম্যাপের নির্মাণের (এবং মেমরী এলোকেশন) পূর্বে আপনাকে ইমেজ ডাটার ডাইমেনশন এবং টাইপ পড়তে দেয়

```
BitmapFactory.Options options = new BitmapFactory.Options();
options.inJustDecodeBounds = true;
BitmapFactory.decodeResource(getResources(), R.id.myimage, options);
int imageHeight = options.outHeight;
int imageWidth = options.outWidth;
String imageType = options.outMimeType;
```

java.lang.OutOfMemory এক্সেপশন পরিহার করতে, ডিকোড করার পূর্বে একটি বিটম্যাপের ডাইমেনশন চেক করুন, যদি না আপনি সম্পূর্ণভাবে সোর্সকে বিশ্বাস করেন যা আপনাকে অনুমিত সাইজের ইমেজ ডাটা যা সহজেই বিদ্যমান মেমরীতে ফিট করবে তা প্রদান করতে।

মেমরীতে একটি স্কেলডাউন-সংস্করণ লোড করুন

এখন ইমেজ ডাইমেনশন জানা হয়েছে, মেমরীতে পূর্ণ ইমেজ লোড করা উচিত নাকি পরিবের্ত একটি সাবস্যাম্পলড সংস্করণ লোড করা উচিত তা ঠিক করতে তাদের এখন ব্যবহার করা যেতে পারে। এখানে কিছু বিষয় বিবেচনার জন্য তুলে ধরা হলো:

- মেমরীতে পূর্ণ ইমেজ লোড করার পূর্বাভাস করা মেমরী ব্যবহার
- মেমরীর যে পরিমান এই ইমেজ লোড করার জন্য আপনি ঠিক করে রেখেছেন তা যদি আপনার অ্যাপলিকেশনের অন্য কোন মেমরী দাবী করে।
- টার্গেট `ImageView` এর ডাইমেনশন বা ইউজার ইন্টারফেস কম্পোনেন্ট যার মধ্যে ইমেজ লোড হবে
- চলতি ডিভাইসের স্ক্রিন সাইজ এবং ঘনত্ব

উদাহরণস্বরূপ, এটা মেমরীতে একটি `1024x768 pixel` ইমেজ লোড করার যোগ্য নয় যদি এটা একটি `ImageView` এর মধ্যে আগে বা পরে একটি `128x96 pixel` থাম্বনেইলে প্রদর্শিত হয়।

ইমেজকে সাবস্যাম্পল করতে ডিকোডার কে বলতে, মেমরীতে একটি ছোট সংস্করণ লোড করুন, আপনার `BitmapFactory.Options` অবজেক্টের `true` এ `inSampleSize` সেট করুন। উদাহরণস্বরূপ, `2048x1536` রেজ্যুলেশন সহকারে একটি ইমেজ যা `8` এর `inSampleSize` দিয়ে ডিকোড করা একটি আনুমানিক `512x384` এর একটি বিটম্যাপ উৎপন্ন করে। এটা মেমরীতে লোড করতে পূর্ণ ইমেজের (`ARGB_8888` এর একটি বিটম্যাপ কনফিগারেশন ধারন করে) জন্য `১২ MB` না ব্যবহার করে `০.৭৫ MB` ব্যবহার করে। স্যাম্পল সাইজ ভ্যালু ক্যালকুলেট করতে এখানে একটি মেথড দেয়া আছে যা একটি টার্গেট দৈর্ঘ্য এবং উচ্চতার উপর ভিত্তি করে দুইয়ের ক্ষমতা।

```
public static int calculateInSampleSize(
    BitmapFactory.Options options, int reqWidth, int reqHeight) {
    // Raw height and width of image
    final int height = options.outHeight;
    final int width = options.outWidth;
    int inSampleSize = 1;

    if (height > reqHeight || width > reqWidth) {

        final int halfHeight = height / 2;
        final int halfWidth = width / 2;

        // Calculate the largest inSampleSize value that is a power of 2 and keeps both
        // height and width larger than the requested height and width.
        while ((halfHeight / inSampleSize) > reqHeight
            && (halfWidth / inSampleSize) > reqWidth) {
            inSampleSize *= 2;
        }
    }
}
```

```

    }

    return inSampleSize;
}

```

নোট: দুইটা ভ্যালুর পাওয়ার ক্যালকুলেট করা হয় কারন দুইয়ের নিকটস্থ পাওয়ারে রাউন্ড ডাউন করার মাধ্যমে ডিকোডার একটি ফাইনাল ভ্যালু ব্যবহার করে, inSampleSize ডকুমেন্টেশন অনুসারে।

এই মেথড ব্যবহার করতে, inJustDecodeBounds সহকারে প্রথম ডিকোড true এ সেট করা, এর মাধ্যমে অপশন পাস করা, তারপর ডিকোড করুন আবার নতুন inSampleSize ভ্যালু এবং inJustDecodeBounds ব্যবহার করে false এ সেট করুন:

```

public static Bitmap decodeSampledBitmapFromResource(Resources res, int resId,
    int reqWidth, int reqHeight) {

    // First decode with inJustDecodeBounds=true to check dimensions
    final BitmapFactory.Options options = new BitmapFactory.Options();
    options.inJustDecodeBounds = true;
    BitmapFactory.decodeResource(res, resId, options);

    // Calculate inSampleSize
    options.inSampleSize = calculateInSampleSize(options, reqWidth, reqHeight);

    // Decode bitmap with inSampleSize set
    options.inJustDecodeBounds = false;
    return BitmapFactory.decodeResource(res, resId, options);
}

```

এই মেথড একটা ImageView এর মধ্যে ইচ্ছামত বড় সাইজের একটি বিটম্যাপ লোড করাকে এটা সহজ করে দেয় যা একটি 100x100 pixel থাম্বনেইল প্রদর্শন করে, যেভাবে নীচের উদাহরণ কোডে দেওয়া হয়েছে:

```

mImageView.setImageBitmap(
    decodeSampledBitmapFromResource(getResources(), R.id.myimage, 100, 100));

```

আপনি অন্য রিসোর্স থেকে বিটম্যাপ ডিকোড করতে একই মেথড অনুসরণ করতে পারেন, যথাযথ BitmapFactory.decode* মেথড প্রয়োজন অনুসারে প্রতিস্থাপন করার মাধ্যমে।

বিটম্যাপ প্রক্রিয়া ইউআই থ্রেড ব্লক করে

(<http://developer.android.com/training/displaying-bitmaps/process-bitmap.html>)

BitmapFactory.decode* মেথড, Load Large Bitmaps Efficiently (বড় আকারের বিটম্যাপ দক্ষতার সাথে লোড করা) অনুশীলনীতে আলোচনা করা হয়েছে, প্রধান ইউজার ইন্টারফেস থ্রেডে কার্যকর করা উচিত নয় যদি সোর্স ডাটা ডিস্ক বা একটি নেটওয়ার্ক লোকেশন থেকে (বা সত্যিকার অর্থে অন্য সোর্স মেমরী নয়) পড়া হয়ে থাকে। লোড করতে ডাটা যে সময় নেয় তা অনুমান করা যায় না এবং বিভিন্ন ঘটনার উপর নির্ভর করে (ডিস্ক বা নেটওয়ার্ক থেকে পড়ার স্পিড, ইমেজ সাইজ, সিপিইউ এর পাওয়ার ইত্যাদি)। যদি এই কাজগুলোর একটা ইউআই থ্রেড ব্লক করে দেয়, সিস্টেমটি আপনার অ্যাপলিকেশন নন-রেসপনসিভ হিসাবে থামিয়ে দেয় এবং ইউজারের এটা ব্লক করাটা বেছে নিতে হয় (আরও তথ্যের জন্য Designing for Responsiveness: <http://developer.android.com/training/articles/perf-anr.html> দেখুন)।

এই অনুশীলনী আপনাকে দেখায় কীভাবে AsyncTask ব্যবহার করে একটি ব্যাকগ্রাউন্ড থ্রেডে বিটম্যাপ প্রসেসিং করতে হয় এবং আরও দেখায় কীভাবে কনকারেন্সি ইস্যু চালনা করতে হয়।

Async Task ব্যবহার করুন

AsyncTask ক্লাস একটি ব্যাকগ্রাউন্ড থ্রেডে কিছু কাজ সম্পাদনের জন্য সহজ রাস্তা প্রদান করে এবং ইউআই থ্রেডে ফলাফল প্রকাশ করে। এটা ব্যবহার করতে, একটি সাবক্লাস তৈরী করুন এবং প্রদত্ত মেথড ওভাররাইড করুন। এখানে AsyncTask এবং decodeSampledBitmapFromResource() ব্যবহার করে একটি ImageView এর মধ্যে একটি বড় আকারের ইমেজ লোড করার উদাহরণ দেয়া হলো:

```
class BitmapWorkerTask extends AsyncTask<Integer, Void, Bitmap> {
    private final WeakReference<ImageView> imageViewReference;
    private int data = 0;

    public BitmapWorkerTask(ImageView imageView) {
        // Use a WeakReference to ensure the ImageView can be garbage collected
        imageViewReference = new WeakReference<ImageView>(imageView);
    }

    // Decode image in background.
    @Override
    protected Bitmap doInBackground(Integer... params) {
        data = params[0];
        return decodeSampledBitmapFromResource(getResources(), data, 100, 100));
    }

    // Once complete, see if ImageView is still around and set bitmap.
    @Override
    protected void onPostExecute(Bitmap bitmap) {
        if (imageViewReference != null && bitmap != null) {
            final ImageView imageView = imageViewReference.get();
            if (imageView != null) {
                imageView.setImageBitmap(bitmap);
            }
        }
    }
}
```

ImageView এর প্রতি WeakReference নিশ্চিত করে যে AsyncTask ImageView কে এবং গারবেজ হওয়া কোন কিছুকে রেফারেন্স করে তা বন্ধ করে না। এটার কোন নিশ্চয়তা নেই যখন কাজ শেষ হবে তখনও আশেপাশে ImageView থাকবে, তাই আপনাকে অবশ্যই onPostExecute()এর মধ্যে রেফারেন্স চেক করতে হবে। ImageView এর অস্তিত্ব আর নাও থাকতে পারে, যদি উদাহরণস্বরূপ, একটিভিটি থেকে নেভিগেট করে চলে যায় বা যদি কাজ শেষ হওয়ার পূর্বেই কনফিগারেশনের পরিবর্তন ঘটে থাকে

বিটম্যাপ আসিঙ্ক্রোনাসলি (asynchronously) লোড শুরু করতে চাইলে, শুধু নতুন কাজ তৈরী করুন এবং সম্পাদন করুন:

```
public void loadBitmap(int resId, ImageView imageView) {
    BitmapWorkerTask task = new BitmapWorkerTask(imageView);
    task.execute(resId);
}
```

কনকারেন্সি চালনা করা

কমন ভিউ কম্পোনেন্ট যেমন ListView এবং GridView অন্য ইস্যু কে শুরু করে যখন AsyncTask দিয়ে কনজাকশনে ব্যবহার করা হয় যেভাবে পূর্ববর্তী সেকশনে দেখানো হয়েছে। মেমরীর সাথে সুসংগঠিত হওয়ার জন্য, এই কম্পোনেন্ট গুলো চাইল্ড ভিউকে ইউজার স্ক্রলের মতো রিসাইকেল করে। যদি প্রতিটা চাইল্ড ভিউ একটি AsyncTask সক্রিয় করে, এখানে কোন নিশ্চয়তা নেই যে যখন এটা শেষ হবে, সহায়তাকারী ভিউ অন্য চাইল্ড ভিউ এ ব্যবহারের জন্য ইতিমধ্যে রিসাইকেল হয় নাই। এছাড়াও, এখানে কোন নিশ্চয়তা নেই যে বিন্যাস যার মধ্যে সিঙক্রোনাস এর কাজ শুরু হয় সেটা সেই বিন্যাস যা তারা সম্পূর্ণ করে।

ব্লগ স্পট Multithreading for Performance (<http://android-developers.blogspot.com/2010/07/multithreading-for-performance.html>) কনকারেন্সি নিয়ে কাজ করা বিষয়ে বিস্তারিত আলোচনা করেছে, এবং একটি সমাধান প্রস্তাব করে যেখানে ImageView একটি রেফারেন্স সদ্য AsyncTask এ স্টোর করে যাকে পরবর্তীতে চেক করা হয় যখন কাজ শেষ হয়। একটি সমপর্যায়ের মেথড ব্যবহার করা, AsyncTask পূর্ববর্তী সেকশন থেকে একটি সমরূপ বৈশিষ্ট অনুসরণ করে AsyncTask সম্প্রসারিত হতে পারে।

ওয়ার্কার টাস্কে ফিরে একটি রেফারেন্স স্টোর করতে একটি নিবেদিত Drawable সাবক্লাস তৈরী করুন। এক্ষেত্রে একটি BitmapDrawable ব্যবহার করা হয় যাতে একটি প্লেস হোল্ডার ইমেজ ImageView এর মধ্যে প্রদর্শিত হতে পারে যখন কাজটি সম্পন্ন হবে:

```
static class AsyncDrawable extends BitmapDrawable {
    private final WeakReference<BitmapWorkerTask> bitmapWorkerTaskReference;

    public AsyncDrawable(Resources res, Bitmap bitmap,
        BitmapWorkerTask bitmapWorkerTask) {
        super(res, bitmap);
        bitmapWorkerTaskReference =
            new WeakReference<BitmapWorkerTask>(bitmapWorkerTask);
    }

    public BitmapWorkerTask getBitmapWorkerTask() {
        return bitmapWorkerTaskReference.get();
    }
}
```

BitmapWorkerTask সম্পাদন করার পূর্বে, আপনি একটি AsyncDrawable তৈরী করেন এবং এটাকে টার্গেট ImageView এ সংযুক্ত করেন:

```
public void loadBitmap(int resId, ImageView imageView) {
    if (cancelPotentialWork(resId, imageView)) {
        final BitmapWorkerTask task = new BitmapWorkerTask(imageView);
        final AsyncDrawable asyncDrawable =
            new AsyncDrawable(getResources(), mPlaceholderBitmap, task);
        imageView.setImageDrawable(asyncDrawable);
        task.execute(resId);
    }
}
```

কোড স্যাম্পল এর মধ্যে রেফারেন্স করা `cancelPotentialWork` মেথড উপরে চেক করে যদি অন্য রানিং কাজ `ImageView` এর সাথে ইতমধ্যে যুক্ত হয়। যদি তাই হয়, `cancel()` কল করার মাধ্যমে পূর্ববর্তী কাজ ক্যানসেল করার প্রচেষ্টা চালানো হয়। খুব কম ক্ষেত্রে নতুন টাস্ক ডাটা বিদ্যমান টাস্ক চেক করে এবং আর কিছুই ঘটার প্রয়োজন হয় না। এখানে `cancelPotentialWork`: এর বাস্তবায়ন আছে:

```
public static boolean cancelPotentialWork(int data, ImageView imageView) {
    final BitmapWorkerTask bitmapWorkerTask = getBitmapWorkerTask(imageView);

    if (bitmapWorkerTask != null) {
        final int bitmapData = bitmapWorkerTask.data;
        if (bitmapData != data) {
            // Cancel previous task
            bitmapWorkerTask.cancel(true);
        } else {
            // The same work is already in progress
            return false;
        }
    }
    // No task associated with the ImageView, or an existing task was cancelled
    return true;
}
```

একটি সাহাজ্যকারী মেথড, `getBitmapWorkerTask()`, উপরে ব্যবহার করা হয়েছে একটি বিশেষ ওসধমবঠরবত্রির সাথে যুক্ত কাজকে পূণরুদ্ধার করতে:

```
private static BitmapWorkerTask getBitmapWorkerTask(ImageView imageView) {
    if (imageView != null) {
        final Drawable drawable = imageView.getDrawable();
        if (drawable instanceof AsyncDrawable) {
            final AsyncDrawable asyncDrawable = (AsyncDrawable) drawable;
            return asyncDrawable.getBitmapWorkerTask();
        }
    }
    return null;
}
```

শেষ ধাপ হচ্ছে `BitmapWorkerTask` এ `onPostExecute()` আপডেট করা যাতে এটা চেক করতে পারে কাজটি ক্যানসেল হয়েছে কিনা এবং চলতি কাজটি `ImageView` এর সাথে সম্পর্কিত এককটি ম্যাচ করেছে কিনা:

```
class BitmapWorkerTask extends AsyncTask<Integer, Void, Bitmap> {
    ...

    @Override
    protected void onPostExecute(Bitmap bitmap) {
        if (isCancelled()) {
            bitmap = null;
        }

        if (imageViewReference != null && bitmap != null) {
            final ImageView imageView = imageViewReference.get();
            final BitmapWorkerTask bitmapWorkerTask =
                getBitmapWorkerTask(imageView);
            if (this == bitmapWorkerTask && imageView != null) {
```

```
        imageView.setImageBitmap(bitmap);  
    }  
}  
}
```

এই বাস্তবায়ন এখন ListView এবং GridView কম্পোনেন্ট একই সাথে অন্য কম্পোনেন্টে ব্যবহার করার জন্য উপযুক্ত যা তাদের চাইল্ড ভিউ রিসাইকেল করে। সাধারনভাবে loadBitmap কল করুন যেখানে আপনি স্বাভাবিকভাবে আপনার ImageView এ একটি ইমেজ সেট করেছেন। উদাহরণস্বরূপ, একটি getView বাস্তবায়নের মধ্যে এটা ব্যাকিং অ্যাডাপ্টরের getView() মেথডের মধ্যে হতে পারে।

বিটম্যাপ জমিয়ে রাখা (ক্যাশিং)

(<http://developer.android.com/training/displaying-bitmaps/cache-bitmap.html>)

একটি সিঙ্গেল বিটম্যাপ আপনার ইউজার ইন্টারফেসে লোড করা খুব সহজ, কিন্ত বিষয়টা জটিল হয়ে যায় যদি আপনাকে একটি বড় আকারের ইমেজ সেট একই সাথে লোড করতে হয়। অনেক ক্ষেত্রে (যেমন, ListView, GridView বা ViewPager এর মতো কম্পোনেন্ট দিয়ে), স্ক্রিনে মোট ইমেজের পরিমাণ কিছু ইমেজ দ্বারা একত্রিত যা শিঘ্রই স্ক্রিনের উপরে স্ক্রল হবে তা মূলত সীমাহীন।

মেমরী ব্যবহার এটার মতো কম্পোনেন্টের সাথে নীচে রাখা হয় চাইল্ড ভিউ রিসাইকেল করার মাধ্যমে যেভাবে তারা অফ-স্ক্রিনে মুভ করে। গারবেজ কালেক্টর (অপ্রয়োজনীয় তথ্য) আপনার লোড করা বিটম্যাপ ছেড়ে দেয়, আপনাকে জানিয়ে দিয়ে যায় কোন দীর্ঘায়ুর রেফারেন্স রাখবেন না। এগুলো ভালো, কিন্ত একটি ফুলইড এবং ফাস্ট-লোডিং ইউজার ইন্টারফেস রাখার জন্য আপনি এই ইমেজেগুলোর প্রতিনিয়ত প্রক্রিয়া করণ কে পরিহার করতে চাইতে পারেন প্রতিবার তারা যখন অন-স্ক্রিনে ফিরে আসে। একটি মেমরী এবং ডিস্ক ক্যাশে এখানে নিয়মিত সাহায্য করতে পারে, প্রসেস করা ইমেজ রিলোড করতে কম্পোনেন্টস কে অনুমোদন দিয়ে।

এই অনুশীলনী আপনাকে দেখাবে যখন মাল্টিপল বিটম্যাপ লোড করা হয় তখন কীভাবে আপনার ইউজার ইন্টারফেসে ফুলইডিটি এবং রেসপনসিভনেস উন্নত করতে একটি মেমরী এবং ডিস্ক বিটম্যাপ ব্যবহার করতে হয়।

একটি মেমরী ক্যাশে ব্যবহার

একটি মেমরী ক্যাশে মূল্যবান অ্যাপলিকেশন মেমরীর খরচের বিনিময়ে বিটম্যাপে দ্রুত প্রবেশগম্যতা প্রস্তাব করে। LruCache ক্লাস (এটা Support Library তেও পাওয়া যায়, এপিআই লেভেল ৪ এ ফিরে ব্যবহারের জন্য) ক্যাশিং বিটম্যাপের টাস্কের জন্য বিশেষ করে উপযোগী, একটি শক্তিশালী রেফারেন্স করা LinkedHashMap এর মধ্যে সম্প্রতি রেফারেন্স করা বসুত রাখে এবং ক্যাশের নির্ধারিত সাইজ ছাড়িয়ে যাওয়ার পূর্বে কম সাম্প্রতিক ব্যবহৃত মেম্বার উচ্ছেদ করে।

নোট: অতীতে, একটি জনপ্রিয় মেমরী ক্যাশে বাস্তবায়ন ছিল SoftReference বা WeakReference বিটম্যাপ ক্যাশে, কিন্তু এটা সুপারিশ করা হয় না। অ্যান্ড্রয়েড ২.৩ (এপিআই লেভেল ৯) থেকে শুরু করে সফট/উইক রেফারেন্স কালেকশন করা সহ গারবেজ কালেক্টর আরও আক্রমনাত্মক যা তাদেরকে কিছুটা অকার্যকর করে। উপরনুত, অ্যান্ড্রয়েড ৩.০ (এপিআই লেভেল ১১) এর পূর্বে, একটি বিটম্যাপের ব্যাকিং ডাটা একটি স্থানীয় (নেটিভ) মেমরীতে স্টোর হতো যা অনুমেয় উপায়ে রিলিজ হয় না, সম্ভাব্য একটি অ্যাপলিকেশনকে একটি সংক্ষিপ্ত সময়ের জন্য এর মেমরী সীমাকে অতিক্রম করার এবং ধ্বংস করার কারন হয়।

একটি LruCache এর জন্য উপযুক্ত সাইজ পছন্দ করার জন্য, কয়েকটি ফ্যাক্টর বিবেচনায় রাখতে হবে:

- কতটুকু মেমরী ইনটেনসিভ হচ্ছে আপনার একটিভিটি এবং/বা অ্যাপলিকেশনের বাকী অংশ?
- একসাথে কতগুলো ইমেজ অন-স্ক্রিন হতে পারে? কতগুলোকে অন-স্ক্রিনে প্রসূতত হয়ে আসতে সহজপ্রাপ্য হওয়ার প্রয়োজন?
- ডিভাইসের স্ক্রিনের সাইজ এবং ঘনত্ব কি? Galaxy Nexus এর মতো একটি এক্সট্রা হাই ডেনজিটি স্ক্রিনের (xhdpi) একটি মেমরী Nexus S (hdpi) এর মতো ডিভাইসে সাথে তুলনীয় তার মধ্যে একই সংখ্যক ইমেজ ধারণ করতে একটি বড় আকারের ক্যাশে দরকার হবে।
- বিটম্যাপের ডাইমেনশন এবং কনফিগারেশন কি এবং তারপর প্রতিটা কি পরিমান মেমরী নিবে?
- কত ঘণ ঘন এই ইমেজে প্রবেশ করা হতে পারে? কোনটাতে কি অন্যটার চেয়ে বেশী বার প্রবেশ করা হতে পারে? যদি হয়, সম্ভবত আপনি সবসময় কিছু আইটেম মেমরীতে রাখতে চাইতে পারেন বা বিটম্যাপের বিভিন্ন গ্রুপের জন্য এমনকি মাল্টিপল LruCache অবজেক্ট আছে।
- আপনি কি পরিমানের বিপরীতে গুণ এর ভারসাম্য রক্ষা করতে পারবেন? কিছু ক্ষেত্রে এটা কম গুণসম্পন্ন বিটম্যাপের একটি বড় সংখ্যাকে স্টোর করার জন্য আরও উপকারী হতে পারে, অন্য ব্যাকগ্রাউন্ড টাস্কের মধ্যে একটি উচ্চ গুণসম্পন্ন সংস্করণ সম্ভাব্য লোড করার জন্য।

এখানে কোন নির্দিষ্ট সাইজ এবং ফর্মুলা নেই যা সকল অ্যাপলিকেশনের জন্য উপযোগী, এটা নির্ভর করে আপনার ব্যবহারের বিশ্লেষণ এবং একটি সুন্দর সমাধানে আসার উপর। একটি ক্যাশে যা খুবই ছোট তা কোন সুবিধা ছাড়াই অতিরিক্ত ওভারহেড ঘটায়, একটি ক্যাশে যা বেশী বড় java.lang.OutOfMemory এক্সসেপশন ঘটায় এবং আপনার অ্যাপের বাকী ক্ষুদ্র মেমরী এর সাথে কাজ করার জন্য ছেড়ে দেয়।

এখানে বিটম্যাপের জন্য একটি LruCache সেটআপের একটি উদাহরন আছে :

```
private LruCache<String, Bitmap> mMemoryCache;

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Get max available VM memory, exceeding this amount will throw an
    // OutOfMemory exception. Stored in kilobytes as LruCache takes an
    // int in its constructor.
    final int maxMemory = (int) (Runtime.getRuntime().maxMemory() / 1024);

    // Use 1/8th of the available memory for this memory cache.
    final int cacheSize = maxMemory / 8;

    mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
        @Override
        protected int sizeOf(String key, Bitmap bitmap) {
            // The cache size will be measured in kilobytes rather than
            // number of items.
            return bitmap.getByteCount() / 1024;
        }
    };
    ...
}

public void addBitmapToMemoryCache(String key, Bitmap bitmap) {
    if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }
}

public Bitmap getBitmapFromMemCache(String key) {
    return mMemoryCache.get(key);
}
```

নোট: এই উদাহরনে, আটভাগের এক ভাগ অ্যাপলিকেশন মেমরী আপনার ক্যাশের জন্য বরাদ্দ। একটি নরমাল/ hdpi ডিভাইসে এটা ন্যূনতম ৪ MB এর কাছাকাছি (৩২/৮)। একটি ডিভাইসে ৪০০x৪৮০ রেজ্যুলেশন সহকারে একটি ফুল স্ক্রিন GridView এ পূর্ণ করে থাকা ইমেজ প্রায় ১.৫ MB (৪০০x৪৮০x৪ bytes) ব্যবহার করতে পারে, সুতরাং এটা নিদেনপক্ষে ইমেজের প্রায় ২.৫ পেজ মেমরীতে জমিয়ে রাখে।

যখন একটি ImageView এর মধ্যে একটি বিটম্যাপ লোড করা হয়, LruCache প্রথমে চেক করা হয়। যদি একটি এন্ট্রি পাওয়া যায়, এটা তাৎক্ষনিকভাবে ImageView আপডেট করতে ব্যবহৃত হয়, অন্যথায় ইমেজ প্রসেস করতে একটি ব্যাকগ্রাউন্ড থ্রেড উৎপন্ন হয়:

```
public void loadBitmap(int resId, ImageView imageView) {
    final String imageKey = String.valueOf(resId);

    final Bitmap bitmap = getBitmapFromMemCache(imageKey);
```

```

if (bitmap != null) {
    mImageView.setImageBitmap(bitmap);
} else {
    mImageView.setImageResource(R.drawable.image_placeholder);
    BitmapWorkerTask task = new BitmapWorkerTask(mImageView);
    task.execute(resId);
}
}

```

BitmapWorkerTask এর মেমরী ক্যাশেতে যুক্ত হতে আপডেট হওয়া প্রয়োজন:

```

class BitmapWorkerTask extends AsyncTask<Integer, Void, Bitmap> {
    ...
    // Decode image in background.
    @Override
    protected Bitmap doInBackground(Integer... params) {
        final Bitmap bitmap = decodeSampledBitmapFromResource(
            getResources(), params[0], 100, 100));
        addBitmapToMemoryCache(String.valueOf(params[0]), bitmap);
        return bitmap;
    }
    ...
}

```

ডিস্ক ক্যাশে ব্যবহার

একটি মেমরী ক্যাশে সাম্প্রতিক ভিউ হওয়া বিটম্যাপে একটি গতিশীল প্রবেশের জন্য উপকারী, কিন্ত আপনি এই ক্যাশেতে ইমেজের সহজপ্রাপ্য হওয়ার উপর নির্ভর করতে পারেন না। বড় আকারের ডাটাসেট সহকারে GridView এর মতো কম্পোনেন্ট সহজেই একটি মেমরী ক্যাশে পূর্ণ করতে পারে। আপনার অ্যাপলিকেশন অন্য কোন টাস্ক যেমন একটি ফোন কল দ্বারা বাধাগ্রস্ত হতে পারে, এবং যখন ব্যাকগ্রাউন্ডের মধ্যে এটা হতে পারে এবং মেমরী ক্যাশে ধ্বংস হবে। যখনই ইউজার আবার শুরু করে, আপনার অ্যাপলিকেশনকে প্রতিটা ইমেজ আবার শুরু করতে হয়।

একটি ডিস্ক ক্যাশে এই ক্ষেত্রে প্রসেস করা বিটম্যাপ অটলভাবে চালিয়ে যাওয়ার কাজে ব্যবহার করা যেতে পারে এবং লোড করার সময় কমাতে পারে যেখানে ইমেজ মেমরী ক্যাশের মধ্যে আর পাওয়া যাবে না। অবশ্যই, ডিস্ক থেকে ইমেজ আনয়ন করা মেমরী থেকে লোড করার চেয়ে ধীরগতির এবং অবশ্যই ব্যাকগ্রাউন্ড থ্রেডে সম্পন্ন করতে হবে, যেহেতু ডিস্ক রিড সব সময় অঅনুমেয়।

নোট: ক্যাশে ইমেজ স্টোর করতে ContentProvider একটা ভালো জায়গা হতে পারে, যদি তাতে বারবার প্রবেশ করা হয়, উদাহরণস্বরূপ, একটি ইমেজ গ্যালারী অ্যাপলিকেশনের মধ্যে।

এই ক্লাসের নমুনা কোড একটি DiskLruCache বাস্তবায়ন ব্যবহার করে যাকে Android source (<https://android.googlesource.com/platform/libcore/+jb-mr2-release/luni/src/main/java/libcore/io/DiskLruCache.java>) থেকে টানা হয়। এখানে একটি আপডেট উদাহরণ কোড আছে যা বিদ্যমান মেমরী ক্যাশে অতিরিক্ত হিসাবে একটি ডিস্ক ক্যাশে যুক্ত করে:

```
private DiskLruCache mDiskLruCache;
private final Object mDiskCacheLock = new Object();
private boolean mDiskCacheStarting = true;
private static final int DISK_CACHE_SIZE = 1024 * 1024 * 10; // 10MB
private static final String DISK_CACHE_SUBDIR = "thumbnails";

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    // Initialize memory cache
    ...
    // Initialize disk cache on background thread
    File cacheDir = getDiskCacheDir(this, DISK_CACHE_SUBDIR);
    new InitDiskCacheTask().execute(cacheDir);
    ...
}

class InitDiskCacheTask extends AsyncTask<File, Void, Void> {
    @Override
    protected Void doInBackground(File... params) {
        synchronized (mDiskCacheLock) {
            File cacheDir = params[0];
            mDiskLruCache = DiskLruCache.open(cacheDir, DISK_CACHE_SIZE);
            mDiskCacheStarting = false; // Finished initialization
            mDiskCacheLock.notifyAll(); // Wake any waiting threads
        }
        return null;
    }
}
```

```

class BitmapWorkerTask extends AsyncTask<Integer, Void, Bitmap> {
    ...
    // Decode image in background.
    @Override
    protected Bitmap doInBackground(Integer... params) {
        final String imageKey = String.valueOf(params[0]);

        // Check disk cache in background thread
        Bitmap bitmap = getBitmapFromDiskCache(imageKey);

        if (bitmap == null) { // Not found in disk cache
            // Process as normal
            final Bitmap bitmap = decodeSampledBitmapFromResource(
                getResources(), params[0], 100, 100);
        }

        // Add final bitmap to caches
        addBitmapToCache(imageKey, bitmap);

        return bitmap;
    }
    ...
}

public void addBitmapToCache(String key, Bitmap bitmap) {
    // Add to memory cache as before
    if (getBitmapFromMemCache(key) == null) {
        mMemoryCache.put(key, bitmap);
    }

    // Also add to disk cache
    synchronized (mDiskCacheLock) {
        if (mDiskLruCache != null && mDiskLruCache.get(key) == null) {
            mDiskLruCache.put(key, bitmap);
        }
    }
}

public Bitmap getBitmapFromDiskCache(String key) {
    synchronized (mDiskCacheLock) {
        // Wait while disk cache is started from background thread
        while (mDiskCacheStarting) {
            try {
                mDiskCacheLock.wait();
            } catch (InterruptedException e) {}
        }
        if (mDiskLruCache != null) {
            return mDiskLruCache.get(key);
        }
    }
    return null;
}

// Creates a unique subdirectory of the designated app cache directory. Tries to use external
// but if not mounted, falls back on internal storage.
public static File getDiskCacheDir(Context context, String uniqueName) {
    // Check if media is mounted or storage is built-in, if so, try and use external cache dir
    // otherwise use internal cache dir
    final String cachePath =
        Environment.MEDIA_MOUNTED.equals(Environment.getExternalStorageState()) ||
            !isExternalStorageRemovable() ? getExternalCacheDir(context).getPath() :
            context.getCacheDir().getPath();

    return new File(cachePath + File.separator + uniqueName);
}

```

নোট: ডিস্ক ক্যাশে আরম্ভ করা ডিস্ক অপারেশন আশা করতে পারে এবং সেজন্য মেইন থ্রেডে ঘটা

ডাচত না। কিনুত, এটা বোঝায় সেখানে একাট সুযোগ আছে ক্যাশে আরম্ভ হওয়ার পূর্বেই সেখানে প্রবেশ করা হয়। এটা সনাক্ত করতে, উপরোক্ত বাস্তবায়নে,একটি লক করা অবজেক্ট নিশ্চিত করে যে অ্যাপ ডিস্ক ক্যাশে থেকে পড়ে না যতক্ষন না ক্যাশে আরম্ভ করা হয়।

যখন মেমরী ক্যাশে ইউআই থ্রেডের মধ্যে চেক করা হয়, ডিস্ক ক্যাশে ব্যাকগ্রাউন্ড থ্রেডে চেক করা হয়। ডিস্ক অপারেশন ইউআই থ্রেডে সংঘটিত হওয়া উচিন নয়। যখন ইমেজ প্রসেস করা সম্পূর্ণ হয়, তখন চূড়ান্ত বিটম্যাপ মেমরী এবং ডিস্ক ক্যাশে দুইটাতেই ভবিষ্যত ব্যবহারের জন্য যুক্ত হয়।

কনফিগারেশন পরিবর্তন ব্যবস্থাপনা

রানটাইম কনফিগারেশন পরিবর্তন, যেমন একটি স্ক্রিন ওরিয়েন্টেশন পরিবর্তন, অ্যান্ড্রয়েডকে নতুন কনফিগারেশন দিয়ে একটি চলমান একটিভিটিকে ধ্বংস বা রিস্টার্ট করায় (এই আচরন সম্পর্কে আরও জানতে Handling Runtime Changes: <http://developer.android.com/guide/topics/resources/runtime-changes.html> দেখুন)। আপনি পুনরায় আপনার সকল ইমেজ প্রসেস হওয়াকে পরিহার করতে চাইতে পারেন যাতে ইউজারের একটি স্বচ্ছন্দ এবং গতিময় এক্সপেরিয়েন্স থাকে যখন একটি কনফিগারেশন ঘটে থাকে।

সৌভাগ্যবশত, আপনার বিটম্যাপের একটি সুন্দর মেমরী ক্যাশে আছে যা আপনি Use a Memory Cache সেকশনে তৈরী করেছেন। এই ক্যাশে একটি Fragment যা `setRetainInstance(true)` কল করার মাধ্যমে সংরক্ষিত তা ব্যবহার করে একটি নতুন একটিভিটিতে পাস হতে পারে। একটিভিটিটি পুনর্নুর্মত হওয়ার পরে, এই রক্ষিত Fragment পুনরায় অ্যাটাচ করা হয় এবং আপনি বিদ্যমান ক্যাশে অবজেক্টে প্রবেশগম্যতা পাবেন, ইমেজকে ImageView অবজেক্টের মধ্যে দ্রুত সংগ্রহিত এবং রিপুলেট হতে অনুমোদন করে।

এখানে একটি Fragment ব্যবহার করে কনফিগারেশন পরিবর্তন জুরে একটি LruCache অবজেক্ট ধরে রাখার উদাহরণ আছে:

```
private LruCache<String, Bitmap> mMemoryCache;

@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    RetainFragment retainFragment =
        RetainFragment.findOrCreateRetainFragment(getFragmentManager());
    mMemoryCache = retainFragment.mRetainedCache;
    if (mMemoryCache == null) {
        mMemoryCache = new LruCache<String, Bitmap>(cacheSize) {
            ... // Initialize cache here as usual
        }
        retainFragment.mRetainedCache = mMemoryCache;
    }
    ...
}

class RetainFragment extends Fragment {
    private static final String TAG = "RetainFragment";
    public LruCache<String, Bitmap> mRetainedCache;

    public RetainFragment() {}

    public static RetainFragment findOrCreateRetainFragment(FragmentManager fm) {
        RetainFragment fragment = (RetainFragment) fm.findFragmentByTag(TAG);
        if (fragment == null) {
            fragment = new RetainFragment();
            fm.beginTransaction().add(fragment, TAG).commit();
        }
        return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
```



```
super.onCreate(savedInstanceState);  
setRetainInstance(true);  
}  
}
```

এটা পরীক্ষা করতে, Fragment ধরে রেখে এবং ধরে না রেখে একটি ডিভাইন রোট্ট করার চেষ্টা করতে পারেন। আপনার একটি লক্ষ্য করা উচিত, দেবী না করা যেহেতু ইমেজ মেমরী থেকে প্রায় তাৎক্ষনিকভাবে একটিভিটিকে অধ্যুষিত করে যখন আপনি ক্যাশে ধারণ করবেন। কোন ইমেজ মেমরীতে খুজে পাওয়া না গেলে আশা করা যাবে যে এটা ডিস্ক ক্যাশেতে আছে, যদি না থাকে তারা যথারীতি প্রসেস হয়েছে।

বিটম্যাপ মেমরী ব্যবস্থাপনা

(<http://developer.android.com/training/displaying-bitmaps/manage-memory.html>)

Caching Bitmaps এ যে আলোচনা করা হয়েছে তা ছাড়াও, গারবেজ কালেকশন এবং বিটম্যাপ রিইউজ সঞ্চালন করতে আপনি নির্দিষ্টভাবে কিছু করতে পারবেন। সুপারিশকৃত কৌশল নির্ভর করে আপনি অ্যান্ড্রয়েডের কি ধরনের সংস্করণ কে লক্ষ্য হিসাবে রেখেছেন। ইরঃসধঢ়ঋঁহ স্যাম্পল অ্যাপ এই ক্লাসের অন্তর্ভুক্ত, আপনাকে দেখায় অ্যান্ড্রয়েডের বিভিন্ন সংস্করণ জুরে দক্ষতার সাথে কাজ করতে কীভাবে আপনার অ্যাপ কে ডিজাইন করতে হয়।

এই অনুশীলনীর জন্য ধাপ গুলো সেট করতে, এখানে আছে কীভাবে অ্যান্ড্রয়েডের বিটম্যাপ মেমরীর ব্যবস্থাপনা গড়ে ওঠে:

- অ্যান্ড্রয়েড ২.২ (এপিআই লেভেল ৮) এবং এর চেয়ে নীচের সংস্করণে, যখন গারবেজ কালেকশন ঘটে, আপনার অ্যাপের থ্রেড থেমে যায়। এটা একটি বিলম্ব ঘটায় যা পারফরমেন্স এর অবনতি ঘটায়। অ্যান্ড্রয়েড ২.৩ সমবর্তী গারবেজ কালেকশন যুক্ত করে, যেটা বোঝায় যে, একটি বিটম্যাপ আর রেফারেন্স না হওয়ার পর মেমরী খুব শিঘ্রই পুনরুদ্ধার হবে।
- অ্যান্ড্রয়েড ২.২.৩ (এপিআই লেভেল ১০) এবং এর চেয়ে নীচের সংস্করণে, একটি বিটম্যাপের জন্য ব্যাকিং পিক্সেল ডাটা স্থানীয় মেমরীতে স্টোর হয়। এটা নিজেই বিটম্যাপ থেকে পৃথক, যেটা ডালভিক হিপে স্টোর হয়। নেটিভ মেমরীর পিক্সেল ডাটা একটি অনুমেয় উপায়ে রিলিজ হয় না, একটি অ্যাপরিকেশনকে ক্ষুদ্রভাবে এর মেমরী সীমা অতিক্রমের এবং ধ্বংস হওয়ার সম্ভাব্য কারন ঘটাবো। অ্যান্ড্রয়েড ৩.০ (এপিআই লেভেল ১১) এর মতো, পিক্সেল ডাটা সংযুক্ত বিটম্যাপের সাথে ডালভিক হিপে স্টোর হয়।

নিচের সেকশন আলোচনা করে বিভিন্ন অ্যান্ড্রয়েড সংস্করণে জন্য কীভাবে বিটম্যাপ মেমরী ব্যবস্থাপনা অপটিমাইজ করতে হয়।

অ্যান্ড্রয়েড ২.২.৩ এবং এর চেয়ে নীচের সংস্করণে মেমরী ব্যবস্থাপনা

অ্যান্ড্রয়েড ২.২.৩ (এপিআই লেভেল ১০) এবং এর চেয়ে নীচের সংস্করণে, `recycle()` ব্যবহার সুপারিশ করা হয়। আপনি যদি আপনার অ্যাপে বড় আকারের বিটম্যাপ ডাটা প্রদর্শন করতে চান, আপনার `OutOfMemoryError` এরর এর মধ্যে রান করার সম্ভাবনা। `recycle()` মেথড একটি অ্যাপকে যত দ্রুত সম্ভব মেমরী পুনরুদ্ধার করতে দেয়।

সতর্কতা: আপনার উচিত `recycle()` ব্যবহার করা শুধুমাত্র তখন যখন আপনি নিশ্চিত হবেন যে বিটম্যাপ আর ব্যবহার হচ্ছে না। আপনি যদি `recycle()` কল করেন এবং পরবর্তীতে বিটম্যাপ ড্র করতে চেষ্টা করে, আপনি এরর পাবেন:

```
"Canvas: trying to use a recycled bitmap".
```

নিম্নোক্ত কোডের খন্ডচিত্রটি `recycle()` কলের একটি উদাহরণ দেয়। এটা একটি বিটম্যাপ বর্তমানে প্রদর্শিত হচ্ছে নাকি ক্যাশেতে আছে তা খুঁজে বের করতে রেফারেন্স কাউন্টিং(ভেরিয়েবল `mDisplayRefCount` এবং `mCacheRefCount` এর মধ্যে) ব্যবহার করে। কোড বিটম্যাপকে রিসাইকেল করে যখন এই অবস্থার সম্মুখীন হয়:

- রেফারেন্স কাউন্ট `mDisplayRefCount` এবং `mCacheRefCount` উভয়ের জন্য ০ হয়
- বিটম্যাপ `null` নয় এবং এটা এখনও রিসাইকেল হয় নাই

```
private int mCacheRefCount = 0; private int mDisplayRefCount = 0; ... // Notify the
drawable that the displayed state has changed. // Keep a count to determine when the
drawable is no longer displayed. public void setIsDisplayed(boolean isDisplayed) {
```

```
    synchronized (this) {
        if (isDisplayed) {
            mDisplayRefCount++;
            mHasBeenDisplayed = true;
        } else {
            mDisplayRefCount--;
        }
    }
    // Check to see if recycle() can be called.
    checkState();
}
```

```
// Notify the drawable that the cache state has changed. // Keep a count to determine
```

when the drawable is no longer being cached. public void setIsCached(boolean isCached) {

```
synchronized (this) {  
    if (isCached) {  
        mCacheRefCount++;  
    } else {  
        mCacheRefCount--;  
    }  
}  
// Check to see if recycle() can be called.  
checkState();
```

}

private synchronized void checkState() {

```
// If the drawable cache and display ref counts = 0, and this drawable  
// has been displayed, then recycle.  
if (mCacheRefCount <= 0 && mDisplayRefCount <= 0 && mHasBeenDisplayed  
    && isValidBitmap()) {  
    getBitmap().recycle();  
}
```

}

private synchronized boolean isValidBitmap() {

```
Bitmap bitmap = getBitmap();  
return bitmap != null && !bitmap.isRecycled();
```

}

অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উপরের সংস্করণে মেমরী ব্যবস্থাপনা

অ্যান্ড্রয়েড ৩.০ (এপিআই লেভেল ১১) BitmapFactory.Options.inBitmap ফিল্ড চালু করে। যদি এই অপশন সেট করা হয়, ডিকোড মেথড যা Options অবজেক্ট নেয় তা একটি বিদ্যমান বিটম্যাপ রিইউজ করার উদ্যোগ নেয় যখন কনটেন্ট লোড করা হয়। এটার মানে হচ্ছে বিটম্যাপের মেমরী পুনর্ব্যবহার হচ্ছে, উন্নত পারফরমেন্সে পরিনত হতে এবং মেমরী এলোকেশন এবং ডি-এলোকেশন উভয় বাদ দিতে। কিন্তু সেখানে কিছু বাধ্যবাধকতা আছে কীভাবে inBitmap ব্যবহার হতে পারে। বিশেষত, অ্যান্ড্রয়েড ৪.৪ (এপিআই লেভেল ১৯) এর পূর্বে শুধুমাত্র সমান সাইজের বিটম্যাপকে সাপোর্ট করা হতো। বিস্তৃত বিবরণের জন্য inBitmap ডকুমেন্টেশন দেখুন।

পরবর্তী ব্যবহারের জন্য একটি বিটম্যাপ সেভ করা

নিম্নোক্ত খন্ডচিহ্নটি দেখায় যে স্যাম্পল অ্যাপে পরবর্তীতে সম্ভাব্য ব্যবহারের জন্য কীভাবে একটি বিটম্যাপ স্টোর হয়। যখন একটি বিটম্যাপ অ্যান্ড্রয়েড ৩.০ বা এর চেয়ে উপরের লেভেলে রান করে এবং একটি বিটম্যাপ LruCache থেকে উচ্ছেদ হয়, বিটম্যাপের প্রতি একটি সফট রেফারেন্স একটি HashSet এর মধ্যে স্থাপিত হয়, inBitmap সহকারে পরবর্তীতে সম্ভাব্য ব্যবহারের জন্য:

```
Set<SoftReference<Bitmap>> mReusableBitmaps;
private LruCache<String, BitmapDrawable> mMemoryCache;

// If you're running on Honeycomb or newer, create a
// synchronized HashSet of references to reusable bitmaps.
if (Utils.hasHoneycomb()) {
    mReusableBitmaps =
        Collections.synchronizedSet(new HashSet<SoftReference<Bitmap>>());
}

mMemoryCache = new LruCache<String, BitmapDrawable>(mCacheParams.memCacheSize) {

    // Notify the removed entry that is no longer being cached.
    @Override
    protected void entryRemoved(boolean evicted, String key,
        BitmapDrawable oldValue, BitmapDrawable newValue) {
        if (RecyclingBitmapDrawable.class.isInstance(oldValue)) {
            // The removed entry is a recycling drawable, so notify it
            // that it has been removed from the memory cache.
            ((RecyclingBitmapDrawable) oldValue).setIsCached(false);
        } else {
            // The removed entry is a standard BitmapDrawable.
            if (Utils.hasHoneycomb()) {
                // We're running on Honeycomb or later, so add the bitmap
                // to a SoftReference set for possible use with inBitmap later.
                mReusableBitmaps.add
                    (new SoftReference<Bitmap>(oldValue.getBitmap()));
            }
        }
    }
}
....
}
```

একটি বিদ্যমান বিটম্যাপ ব্যবহার

রান করা অ্যাপের মধ্যে, ডিকোডার মেথড দেখার জন্য চেক করে যে যদি সেখানে কোন বিদ্যমান বিটম্যাপ থাকে যা তারা ব্যবহার করতে পারে। উদাহরণস্বরূপ:

```
public static Bitmap decodeSampledBitmapFromFile(String filename,
    int reqWidth, int reqHeight, ImageCache cache) {

    final BitmapFactory.Options options = new BitmapFactory.Options();
    ...
    BitmapFactory.decodeFile(filename, options);
    ...

    // If we're running on Honeycomb or newer, try to use inBitmap.
    if (Utils.hasHoneycomb()) {
        addInBitmapOptions(options, cache);
    }
    ...
    return BitmapFactory.decodeFile(filename, options);
}
```

পরবর্তী খন্ডাংশটি দেখায় `addInBitmapOptions()` মেথড যাকে উপরের খন্ডাংশতে ব্যবহার করা হয়েছে। এটা `inBitmap` এর ভ্যালু হিসাবে সেট করতে একটি বিদ্যমান বিটম্যাপ অনুসন্ধান করে। উল্লেখ্য এই মেথড শুধুমাত্র `inBitmap` এর জন্য ভ্যালু সেট করে যদি এটা একটি উপযুক্ত ম্যাচ খুঁজে পায় (আপনার কোডের কখনই অনুমান করা উচিত নয় যে একটি ম্যাচ খুঁজে পাওয়া যাবে): `private static void addInBitmapOptions(BitmapFactory.Options options, ImageCache cache) { // inBitmap only works with mutable bitmaps, so force the decoder to // return mutable bitmaps. options.inMutable = true;`

```
if (cache != null) {
    // Try to find a bitmap to use for inBitmap.
    Bitmap inBitmap = cache.getBitmapFromReusableSet(options);

    if (inBitmap != null) {
        // If a suitable bitmap has been found, set it as the value of
        // inBitmap.
        options.inBitmap = inBitmap;
    }
}

// This method iterates through the reusable bitmaps, looking for one
// to use for inBitmap:
protected Bitmap getBitmapFromReusableSet(BitmapFactory.Options options) {
    Bitmap bitmap = null;

    if (mReusableBitmaps != null && !mReusableBitmaps.isEmpty()) {
        synchronized (mReusableBitmaps) {
            final Iterator<SoftReference<Bitmap>> iterator
                = mReusableBitmaps.iterator();
            Bitmap item;

            while (iterator.hasNext()) {
```

```

        item = iterator.next().get();

        if (null != item && item.isMutable()) {
            // Check to see if the item can be used for inBitmap.
            if (canUseForInBitmap(item, options)) {
                bitmap = item;

                // Remove from reusable set so it can't be used again.
                iterator.remove();
                break;
            }
        } else {
            // Remove from the set if the reference has been cleared.
            iterator.remove();
        }
    }
}

return bitmap;
}

```

চূড়ান্তভাবে, এই মেথড নির্ধারণ করে একটি প্রার্থী বিটম্যাপ inBitmap এর জন্য ব্যবহার করা হবে এমন সাইজ ক্রাইটেরিয়াকে সনুতষ্ট করে:

```

static boolean canUseForInBitmap(
    Bitmap candidate, BitmapFactory.Options targetOptions) {

    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.KITKAT) {
        // From Android 4.4 (KitKat) onward we can re-use if the byte size of
        // the new bitmap is smaller than the reusable bitmap candidate
        // allocation byte count.
        int width = targetOptions.outWidth / targetOptions.inSampleSize;
        int height = targetOptions.outHeight / targetOptions.inSampleSize;
        int byteCount = width * height * getBytesPerPixel(candidate.getConfig());
        return byteCount <= candidate.getAllocationByteCount();
    }

    // On earlier versions, the dimensions must match exactly and the inSampleSize must be 1
    return candidate.getWidth() == targetOptions.outWidth
        && candidate.getHeight() == targetOptions.outHeight
        && targetOptions.inSampleSize == 1;
}

/**
 * A helper function to return the byte usage per pixel of a bitmap based on its configuration.
 */
static int getBytesPerPixel(Config config) {
    if (config == Config.ARGB_8888) {
        return 4;
    } else if (config == Config.RGB_565) {
        return 2;
    } else if (config == Config.ARGB_4444) {
        return 2;
    } else if (config == Config.ALPHA_8) {
        return 1;
    }
    return 1;
}

```


আপনার ইউআই একটি বিটম্যাপ প্রদর্শন

(<http://developer.android.com/training/displaying-bitmaps/display-bitmap.html>)

এই অনুশীলনী পূর্ববর্তী অনুশীলনীর সব কিছু এক সাথে নিয়ে আসে, আপনাকে দেখায় একটি ব্যাকগ্রাউন্ড থ্রেড এবং বিটম্যাপ ক্যাশে ব্যবহার করে ViewPager এবং GridView কম্পোনেন্টের এর মধ্যে কীভাবে মাল্টিপল বিটম্যাপ লোড করতে হয়, যখন কনকারেন্সি এবং কনফিগারেশন পরিবর্তন নিয়ে কাজ করতে হয়।

ভিউপেজ বাস্তবায়নে বিটম্যাপ লোড করা

একটি ইমেজ গ্যালারির বিস্তারিত ভিউ নেভিগেট করতে swipe view pattern একটি চমৎকার উপায়। একটি PagerAdapter দ্বারা সমর্থিত একটি ViewPager ব্যবহার আপনি এই প্যাটার্ন বাস্তবায়ন করতে পারেন। কিনুত এর চেয়ে বেশী উপযুক্ত ব্যাকিং দেয়া অ্যাডাপটর হচ্ছে সাবক্লাস FragmentStatePagerAdapter যা ViewPager এর মধ্যে Fragments এর স্টেট স্বয়ংক্রিয়ভাবে ধ্বংস এবং সেভ করে যেভাবে তারা অফ-স্ক্রিন দৃশ্যমান করে, মেমরী ব্যবহার নীচে রাখে।

নোট: আপনার যদি অল্প সংখ্যক ইমেজ থাকে এবং আপনি যদি আত্মবিশ্বাসি হোন যে সবগুলোই অ্যাপলিকেশন মেমরী সীমার মধ্যে ফিট করবে, তাহলে একটি নিয়মিত PagerAdaptore বা FragmentPagerAdapter ব্যবহার করাটা অধিক যুক্তিযুক্ত হবে।

এখানে ImageView চিলড্রেন সহকারে একটি ViewPager এর বাস্তবায়ন আছে। মেইন একটিভিটি ViewPager এবং অ্যাডাপটর ধারন করে:

```
public class ImageDetailActivity extends FragmentActivity {
    public static final String EXTRA_IMAGE = "extra_image";

    private ImagePagerAdapter mAdapter;
    private ViewPager mPager;

    // A static dataset to back the ViewPager adapter
    public final static Integer[] imageResIds = new Integer[] {
        R.drawable.sample_image_1, R.drawable.sample_image_2, R.drawable.sample_image_3,
        R.drawable.sample_image_4, R.drawable.sample_image_5, R.drawable.sample_image_6,
        R.drawable.sample_image_7, R.drawable.sample_image_8, R.drawable.sample_image_9};

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.image_detail_pager); // Contains just a ViewPager

        mAdapter = new ImagePagerAdapter(getSupportFragmentManager(), imageResIds.length);
        mPager = (ViewPager) findViewById(R.id.pager);
        mPager.setAdapter(mAdapter);
    }

    public static class ImagePagerAdapter extends FragmentStatePagerAdapter {
        private final int mSize;

        public ImagePagerAdapter(FragmentManager fm, int size) {
            super(fm);
            mSize = size;
        }

        @Override
        public int getCount() {
            return mSize;
        }

        @Override
        public Fragment getItem(int position) {
            return ImageDetailFragment.newInstance(position);
        }
    }
}
```

```
}
```

এখানে বিস্তারিত Fragment এর একটি বাস্তবায়ন আছে যা imageView চিলড্রেন ধারণ করে। এটা মনে হতে পারে একটি পুরাপুরি যুক্তিসংগত অ্যাপ্রোচ, কিন্ত আপনি কি এই বাস্তবায়নের ড্রব্যাক দেখতে পাচ্ছে? এটা কীভাবে উন্নত হবে?

```
public class ImageDetailFragment extends Fragment {
    private static final String IMAGE_DATA_EXTRA = "resId";
    private int mImageNum;
    private ImageView mImageView;

    static ImageDetailFragment newInstance(int imageNum) {
        final ImageDetailFragment f = new ImageDetailFragment();
        final Bundle args = new Bundle();
        args.putInt(IMAGE_DATA_EXTRA, imageNum);
        f.setArguments(args);
        return f;
    }

    // Empty constructor, required as per Fragment docs
    public ImageDetailFragment() {}

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mImageNum = getArguments() != null ? getArguments().getInt(IMAGE_DATA_EXTRA) : -1;
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
        Bundle savedInstanceState) {
        // image_detail_fragment.xml contains just an ImageView
        final View v = inflater.inflate(R.layout.image_detail_fragment, container, false);
        mImageView = (ImageView) v.findViewById(R.id.imageView);
        return v;
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        final int resId = ImageDetailActivity.imageResIds[mImageNum];
        mImageView.setImageResource(resId); // Load image into ImageView
    }
}
```

আশা করা যায় যে আপনি এই ইস্যুটি লক্ষ্য করেছেন: ইমেজ ইউআই থ্রেডে রিসোর্স থেকে পড়া হচ্ছে যা একটি অ্যাপলিকেশনকে বুলে থাকা বা জোর পূর্বক বন্ধ হয়ে যাওয়ার দিকে নিয়ে যায়। Processing Bitmaps Off the UI Thread অনুশীলনীতে যেভাবে আলোচনা করা হয়েছে সেভাবে একটি AsyncTask ব্যবহার, একটি ব্যাকগ্রাউন্ড থ্রেডে ইমেজ লোড এবং প্রসেস করতে করানোটা সোজাসাপ্টা:

```
public class ImageDetailActivity extends FragmentActivity {
    ...

    public void loadBitmap(int resId, ImageView imageView) {
        mImageView.setImageResource(R.drawable.image_placeholder);
        BitmapWorkerTask task = new BitmapWorkerTask(mImageView);
        task.execute(resId);
    }
}
```

```

    }

    ... // include BitmapWorkerTask class
}

public class ImageDetailFragment extends Fragment {
    ...

    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        if (ImageDetailActivity.class.isInstance(getActivity())) {
            final int resId = ImageDetailActivity.imageResIds[mImageNum];
            // Call out to ImageDetailActivity to load the bitmap in a background thread
            ((ImageDetailActivity) getActivity()).loadBitmap(resId, mImageView);
        }
    }
}

```

যে কোন অতিরিক্ত প্রসেস (যেমন, নেটওয়ার্ক থেকে ইমেজ রিসাইজ করা বা নিয়ে আসা) মেইন ইউআই এর রেসপনসিভনেস কে কোনভাবে প্রভাবিত না করাই BitmapWorkerTask এর মধ্যে সংঘটিত করতে পারে। যদি ব্যাকগ্রাউন্ড থ্রেড সরাসরি ডিস্ক থেকে একটি ইমেজ লোড করার চাইতেও বেশী কিছু করে, এটা একটি মেমরী এবং /বা ডিস্ক ক্যাশে যুক্ত করে লাভবান হতে পারে, যেভাবে Caching Bitmaps অনুশীলনীতে আলোচনা করা হয়েছে। এখানে একটি মেমরী ক্যাশের অতিরিক্ত পরিবর্তন দেয়া আছে:

```

public class ImageDetailActivity extends FragmentActivity {
    ...
    private LruCache<String, Bitmap> mMemoryCache;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        // initialize LruCache as per Use a Memory Cache section
    }

    public void loadBitmap(int resId, ImageView imageView) {
        final String imageKey = String.valueOf(resId);

        final Bitmap bitmap = mMemoryCache.get(imageKey);
        if (bitmap != null) {
            mImageView.setImageBitmap(bitmap);
        } else {
            mImageView.setImageResource(R.drawable.image_placeholder);
            BitmapWorkerTask task = new BitmapWorkerTask(mImageView);
            task.execute(resId);
        }
    }

    ... // include updated BitmapWorkerTask from Use a Memory Cache section
}

```

সকল অংশ একসাথে করা ন্যূনতম লোডিং ল্যাটেন্সি (কোন কিছু প্রসেস করার সময়কাল) এবং আপনার ইমেজে যতটুকু দরকার সেই মতে কম বা বেশী ব্যাকগ্রাউন্ড প্রসেসিং করার সামর্থ্য সহ আপনাকে একটি রেসপনসিভ ViewPager বাস্তবায়ন প্রদান করবে।

একটি গ্রিডভিউ বাস্তবায়নে বিটম্যাপ লোড করা

grid list building block ইমেজ ডাটা সেট দেখানোর জন্য উপকারী এবং একটি GridView কম্পোনেন্ট ব্যবহার করে বাস্তবায়িত হতে পারে যার মধ্যে অনেক ইমেজ যে কোন সময় এক সাথে অন-স্ক্রিন হতে পারে এবং আরও অনেকগুলো দৃশ্যমান হতে প্রস্তুত হওয়া দরকার যদি ইউজার আপ বা ডাউনে স্ক্রল করে। যখন এই ধরনের নিয়ন্ত্রণ বাস্তবায়ন করা হয়, আপনাকে নিশ্চিত করতে হবে যে ইউআই ফুলইড ধরে রাখে, মেমরী ব্যবহার নিয়ন্ত্রণে রাখে এবং কনকারেন্সি সঠিকভাবে ব্যবস্থাপন করা হয় (GridView চিলড্রেন ভিউ রিসাইকেল করার কারনে)।

শুরুতে, ImageView চিলড্রেন সহকারে একটি আদর্শ GridView বাস্তবায়ন একটি Fragment এর মধ্যে স্থাপন করে। পূরণায়, এটাকে মনে হতে পারে একটি সম্পূর্ণ যৌক্তিক অ্যাপ্রোচ, কিন্ত এটাকে কি ভালো করতে পারে?

```
public class ImageGridFragment extends Fragment implements AdapterView.OnItemClickListener {
    private ImageAdapter mAdapter;

    // A static dataset to back the GridView adapter
    public final static Integer[] imageResIds = new Integer[] {
        R.drawable.sample_image_1, R.drawable.sample_image_2, R.drawable.sample_image_3,
        R.drawable.sample_image_4, R.drawable.sample_image_5, R.drawable.sample_image_6,
        R.drawable.sample_image_7, R.drawable.sample_image_8, R.drawable.sample_image_9};

    // Empty constructor as per Fragment docs
    public ImageGridFragment() {}

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        mAdapter = new ImageAdapter(getActivity());
    }

    @Override
    public View onCreateView(
        LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        final View v = inflater.inflate(R.layout.image_grid_fragment, container, false);
        final GridView mGridView = (GridView) v.findViewById(R.id.gridview);
        mGridView.setAdapter(mAdapter);
        mGridView.setOnItemClickListener(this);
        return v;
    }

    @Override
    public void onItemClick(AdapterView<?> parent, View v, int position, long id) {
        final Intent i = new Intent(getActivity(), ImageDetailActivity.class);
        i.putExtra(ImageDetailActivity.EXTRA_IMAGE, position);
        startActivity(i);
    }

    private class ImageAdapter extends BaseAdapter {
        private final Context mContext;

        public ImageAdapter(Context context) {
            super();
            mContext = context;
        }
    }
}
```

```

@Override
public int getCount() {
    return imageResIds.length;
}

@Override
public Object getItem(int position) {
    return imageResIds[position];
}

@Override
public long getItemId(int position) {
    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup container) {
    ImageView imageView;
    if (convertView == null) { // if it's not recycled, initialize some attributes
        imageView = new ImageView(mContext);
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
        imageView.setLayoutParams(new GridView.LayoutParams(
            LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));
    } else {
        imageView = (ImageView) convertView;
    }
    imageView.setImageResource(imageResIds[position]); // Load image into ImageView
    return imageView;
}
}

```

আর একবার, এই বাস্তবায়নের সমস্যা হচ্ছে যে ইমেজ ইউআই থ্রেডে সেট হচ্ছে। যখন এটা ছোট, সহজ ইমেজের জন্য কাজ করে (সিস্টেম রিসোর্স লোড এবং ক্যাশে করার কারনে), যদি কোন অতিরিক্ত প্রসেস করার দরকার হয়, আপনার ইউআই একটি হাল্টে ধ্বংস হয়।

পূর্ববর্তী সেকশন থেকে একই আসিঙ্ক্রোনাস প্রসেস এবং ক্যাশে করার মেথড এখানে বাস্তবায়ন করা যেতে পারে। কিন্ত, আপনার কনকারেন্সি ইস্যুতে সতর্ক হওয়া প্রয়োজন কারন GridView চাইল্ড ভিউ রিসাইকেল করে। এটা পরিচালনা করতে, Processing Bitmaps Off the UI Thread অনুশীলনীতে আলোচিত কৌশল ব্যবহার করুন। এখানে আপডেট সমাধান দেয়া হলো:

```

public class ImageGridFragment extends Fragment implements AdapterView.OnItemClickListener {
    ...

    private class ImageAdapter extends BaseAdapter {
        ...

        @Override
        public View getView(int position, View convertView, ViewGroup container) {
            ...
            loadBitmap(imageResIds[position], imageView)
            return imageView;
        }
    }

    public void loadBitmap(int resId, ImageView imageView) {
        if (cancelPotentialWork(resId, imageView)) {
            final BitmapWorkerTask task = new BitmapWorkerTask(imageView);
            final AsyncDrawable asyncDrawable =
                new AsyncDrawable(getResources(), mPlaceholderBitmap, task);
            imageView.setImageDrawable(asyncDrawable);
            task.execute(resId);
        }
    }
}

```

```

    }
}

static class AsyncDrawable extends BitmapDrawable {
    private final WeakReference<BitmapWorkerTask> bitmapWorkerTaskReference;

    public AsyncDrawable(Resources res, Bitmap bitmap,
        BitmapWorkerTask bitmapWorkerTask) {
        super(res, bitmap);
        bitmapWorkerTaskReference =
            new WeakReference<BitmapWorkerTask>(bitmapWorkerTask);
    }

    public BitmapWorkerTask getBitmapWorkerTask() {
        return bitmapWorkerTaskReference.get();
    }
}

public static boolean cancelPotentialWork(int data, ImageView imageView) {
    final BitmapWorkerTask bitmapWorkerTask = getBitmapWorkerTask(imageView);

    if (bitmapWorkerTask != null) {
        final int bitmapData = bitmapWorkerTask.data;
        if (bitmapData != data) {
            // Cancel previous task
            bitmapWorkerTask.cancel(true);
        } else {
            // The same work is already in progress
            return false;
        }
    }
    // No task associated with the ImageView, or an existing task was cancelled
    return true;
}

private static BitmapWorkerTask getBitmapWorkerTask(ImageView imageView) {
    if (imageView != null) {
        final Drawable drawable = imageView.getDrawable();
        if (drawable instanceof AsyncDrawable) {
            final AsyncDrawable asyncDrawable = (AsyncDrawable) drawable;
            return asyncDrawable.getBitmapWorkerTask();
        }
    }
    return null;
}

... // include updated BitmapWorkerTask class

```

নোট: একই কোড ListView সাথে কাজ করতেও মানিয়ে নিতে পারে।

এই বাস্তবায়ন ফ্লেক্সিবিলিটির জন্য অনুমোদন করে যেভাবে ইমেজ প্রসেস এবং লোড হয় ইউআই এর মসৃনতাকে বাধাগ্রস্ত না করেই। ব্যাকগ্রাউন্ড টাস্কে আপনি নেটওয়ার্ক থেকে ইমেজ লোড করতে পারেন বা বড় ডিজিটাল ক্যামেরা ফটো এবং ইমেজ দৃশ্যমান তা রিসাইজ করতে পারেন যেহেতু টাস্ক প্রসেস করা শেষ করে।

এটার এবং অন্য কনসেপ্টের যা এই অনুশীলনীতে আলোচনা করা হয়েছে তার পূর্ণ উদাহরণের জন্য, দয়া করে অন্তর্ভুক্ত স্যাম্পল অ্যাপলিকেশন দেখুন।

OpenGL ES দিয়ে গ্রাফিক্স প্রদর্শন

(<http://developer.android.com/training/graphics/opengl/index.html>)

অ্যান্ড্রয়েড ফ্রেমওয়ার্ক আকর্ষণীয়, ফাংশনাল গ্রাফিক্যাল ইউজার ইন্টারফেস তৈরী করার জন্য বেশ কিছু আদর্শ টুলস সরবরাহ করে থাকে। কিন্ত স্ক্রিনে যা অঙ্কন করা হচ্ছে তার উপর আপনি যদি আরও নিয়ন্ত্রণ চান, বা ত্রিমাত্রিক গ্রাফিক্স এর মধ্যে কাজ করতে উদ্যোগী হোন, আপনার বিভিন্ন টুলস ব্যবহার করা প্রয়োজন। অ্যান্ড্রয়েড ফ্রেমওয়ার্ক দ্বারা প্রদত্ত OpenGL ES APIs মূল্যবান অ্যানিমেটেড গ্রাফিক্স যা শুধুমাত্র আপনার চিন্তাশীলতা দ্বারা সীমাবদ্ধ তা প্রদর্শন করার জন্য এক সেট টুল প্রস্তাব করে এবং এছাড়াও অনেক অ্যান্ড্রয়েড ডিভাইসে প্রদত্ত একসেলারেশন অব গ্রাফিক্স প্রসেসিং ইউনিট (GPUs) থেকে লাভবান হতে পারেন।

এই ক্লাস OpenGL ব্যবহার করা অ্যাপলিকেশন ডেভেলপ করার মৌলিক বিষয়গুলো আপনাকে দেখাবে, যার মধ্যে রয়েছে সেটআপ, ড্রয়িং অবজেক্ট, মুভিং ড্রঅন এলিমেন্ট এবং টাচ ইনপুটে রেসপন্স।

এই ক্লাসের মধ্যে উদাহরণ কোড OpenGL ES 2.0 APIs ব্যবহার করে, যা চলতি অ্যান্ড্রয়েড ডিভাইসের সাথে ব্যবহার করতে সুপারিশকৃত এপিআই সংস্করণ। OpenGL ES এর সংস্করণ সম্পর্কে আরও তথ্য জানতে OpenGL ডেভেলপার গাইড দেখুন।

নোট: সতর্ক থাকবেন, OpenGL ES 1.x API কে OpenGL ES 2.0 methods বলে ভুল করবেন না! এই দুইটা অদল-বদল করার মতো নয় এবং দুইটাকে একসাথে ব্যবহার করার ফলে আপনি শুধু হতাশই হবেন এবং কষ্টও বাড়বে।

অনুশীলনীসমূহ

একটি OpenGL ES পরিবেশ তৈরী করুন

শিখুন OpenGL গ্রাফিক্স অঙ্কন করতে সমর্থ হতে কীভাবে একটি অ্যান্ড্রয়েড অ্যাপলিকেশন সেটআপ করতে হয়

গঠন নির্ধারণ করুন

শিখুন কীভাবে গঠন (শেপ) নির্ধারণ করতে হয় এবং কেন আপনাকে ফেস এবং উইনডিং সম্পর্কে জানতে হবে

গঠন ড্র/অঙ্কন করুন

শিখুন আপনার অ্যাপলিকেশনে কীভাবে গঠন (শেপ) অঙ্কন করতে হয়

প্রজেক্ট এবং ক্যামেরা প্রয়োগ করুন

শিখুন আপনার ড্রঅন অবজেক্টে একটি নতুন প্রেক্ষাপট পেতে কীভাবে প্রজেকশন এবং ক্যামেরা ভিউ ব্যবহার করতে হয়

মোশন যুক্ত করুন

শিখুন OpenGL দিয়ে কীভাবে ড্রঅন অবজেক্ট এর মৌলিক গতিবিধি এবং অ্যানিমেশন করতে হয়

টাচ ইভেন্টে রেসপন্স করা

শিখুন OpenGL গ্রাফিক্স দিয়ে কীভাবে মৌলিক যোগাযোগ করতে হয়

একটি OpenGL ES পরিবেশ তৈরী করা

(<http://developer.android.com/training/graphics/opengl/environment.html>)

আপনার অ্যান্ড্রয়েড অ্যাপলিকেশনে OpenGL ES দিয়ে গ্রাফিক্স অঙ্কন করার জন্য, আপনাকে অবশ্যই তাদের জন্য একটি ভিউ কনটেইনার তৈরী করতে হবে। এটা করার সোজা-সাপ্টা অনেকগুলো উপায়ের একটি হচ্ছে একটি GLSurfaceView এবং একটি GLSurfaceView.Renderer উভয়ই বাস্তবায়ন করা। একটি GLSurfaceView হচ্ছে OpenGL দিয়ে গ্রাফিক্স অঙ্কন করার জন্য একটি ভিউ কনটেইনার, এবং GLSurfaceView.Renderer হচ্ছে ঐ ভিউয়ের মধ্যে যা অঙ্কন করা হচ্ছে তা নিয়ন্ত্রণ করে। এই ক্লাস সম্পর্কে আরও তথ্যের জন্য OpenGL ES (<http://developer.android.com/guide/topics/graphics/opengl.html>) ডেভেলপার গাইড দেখুন।

GLSurfaceView হচ্ছে আপনার অ্যাপলিকেশনের মধ্যে OpenGL ES গ্রাফিক্স সংযুক্ত করার একটি উপায়। একটি ফুল স্ক্রিন বা ফুল স্ক্রিনের কাছাকাছি গ্রাফিক্স ভিউ এর জন্য এটা একটা যৌক্তিক পছন্দ। ডেভেলপার যারা তাদের লেআউটের একটি ছোট অংশে OpenGL ES গ্রাফিক্স সংযুক্ত করতে চায় তাদের TextureView টি এক নজর দেখা উচিত। সত্যিকার অর্থে “ডু-ইট-ইয়োরসেল্ফ” ডেভেলপার, এটাও সম্ভব যে SurfaceView ব্যবহার করে একটি OpenGL ES ভিউ তৈরী করা কিনুত এটার একটু বেশী অতিরিক্ত কোড লেখার প্রয়োজন হয়।

এই অনুশীলনী ব্যাখ্যা করে একটি সরল অ্যাপলিকেশন কীভাবে GLSurfaceView এবং GLSurfaceView.Renderer এর একটি ন্যূনতম বাস্তবায়ন সম্পূর্ণ করতে হয়।

মেনিফেস্টে OpenGL ES ব্যবহার ডিক্লেয়ার করুন

আপনার অ্যাপলিকেশনের জন্য OpenGL ES 2.0 API ব্যবহার করতে, আপনাকে অবশ্যই আপনার মেনিফেস্টে নিম্নোক্ত ডিক্লেয়ারেশন যুক্ত করতে হবে:

```
<uses  
feature android:glEsVersion="0x00020000" android:required="true" />
```

যদি আপনার অ্যাপলিকেশন টেক্সচার কম্প্রেশন ব্যবহার করে, আপনাকে অবশ্যই কোন কম্প্রেশন ফরম্যাট আপনি সাপোর্ট করবেন সেটাও ডিক্লেয়ার করতে হবে যাতে যে ডিভাইস এই সকল ফরম্যাট সাপোর্ট করে না তারা আপনার অ্যাপলিকেশন রান করার চেষ্টা করতে না পারে:

```
<supports gl texture android:name="GL_OES_compressed_ETC1_RGB8_texture" />  
<supports gl texture android:name="GL_OES_compressed_paletted_texture" />
```

টেক্সচার কম্প্রেশন ফরম্যাট সম্পর্কে আরও তথ্য জানতে, OpenGL (<http://developer.android.com/guide/topics/graphics/opengl.html>) ডেভেলপার গাইড দেখুন।

OpenGL ES গ্রাফিক্স এর জন্য একটি একটিভিটি তৈরী করুন

অ্যান্ড্রয়েড অ্যাপলিকেশন যা OpenGL ES ব্যবহার করে তার একটিভিটি আছে, অন্যান্য অ্যাপলিকেশনের মতোই যার একটি ইউজার একটিভিটি আছে। অন্য অ্যাপলিকেশন থেকে এর প্রধান পার্থক্য হচ্ছে আপনার একটিভিটির জন্য লেআউটে আপনি কি রেখেছেন। যেখানে অনেক অ্যাপলিকেশনে আপনি TextView, Button এবং ListView ব্যবহার করতে পারেন, OpenGL ES ব্যবহার করা অ্যাপে, আপনি আরও একটি GLSurfaceView যুক্ত করতে পারেন।

নিম্নোক্ত কোড উদাহরণ একটি একটিভিটির ন্যূনতম বাস্তবায়ন যা এর প্রাথমিক ভিউ হিসাবে একটি GLSurfaceView ব্যবহার করা দেখায়:

```
public class OpenGL20Activity extends Activity {  
  
    private GLSurfaceView mGLView;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        // Create a GLSurfaceView instance and set it  
        // as the ContentView for this Activity.  
        mGLView = new MyGLSurfaceView(this);  
        setContentView(mGLView);  
    }  
}
```

নোট: OpenGL ES 2.0 চায় অ্যান্ড্রয়েড ২.২ (এপিআই লেভেল ৮) বা এর চেয়ে উপরের সংস্করণ, তাই নিশ্চিত করুন যাতে আপনার অ্যান্ড্রয়েড প্রজেক্ট ঐ API বা এর চেয়ে উপরের সংস্করণকে টার্গেট করে।

একটি GLSurfaceView অবজেক্ট তৈরী করুন

একটি GLSurfaceView হচ্ছে বিশেষায়িত ভিউ যেখানে আপনি OpenGL ES গ্রাফিক্স অঙ্কন করতে পারেন। এটা নিজ থেকে তেমন কিছু করে না। অবজেক্টের যথার্থ অঙ্কন GLSurfaceView.Renderer এর মধ্যে নিয়ন্ত্রণ হয়ে থাকে যা আপনি এই ভিউ এ সেট করেছেন। মূলত, এই অবজেক্টের জন্য কোডটি খুব পাতলা, আপনি একে প্রসারিত করা এড়িয়ে যেতে পারেন এবং শুধু একটি অপরিমিত GLSurfaceView ইনসটেন্স তৈরী করতে পারেন কিনুত এটা করবেন না। টাচ ইভেন্ট ক্যাপচার করার জন্য আপনাকে অবশ্যই এই ক্লাস প্রসারিত করার প্রয়োজন, যা Responding to Touch Events ক্লাসে আলোচনায় এসেছে।

একটি GLSurfaceView এর জন্য প্রয়োজনীয় কোড হচ্ছে ন্যূনতম (মিনিমাল), সুতরাং একটি দ্রুত বাস্তবায়নের জন্য, একটি একটিভিটি যা এটা ব্যবহার করে তার মধ্যে শুধু একটি ইনার ক্লাস তৈরী করাটা একটা সাধারণ বিষয়:

```
class MyGLSurfaceView extends GLSurfaceView {  
  
    public MyGLSurfaceView(Context context){  
        super(context);  
  
        // Set the Renderer for drawing on the GLSurfaceView  
        setRenderer(new MyRenderer());  
    }  
}
```

যখন OpenGL ES 2.0 ব্যবহার করা হয়, আপনাকে অবশ্যই আপনার GLSurfaceView কনস্ট্রাকটরে আরেকটি কল যুক্ত করতে হবে, এটা উল্লেখ করে যে আপনি ২.০ এপিআই ব্যবহার করতে চান:

```
// Create an OpenGL ES 2.0 context  
setEGLContextClientVersion(2);
```

নোট: যদি আপনি OpenGL ES 2.0 API ব্যবহার করতে থাকেন, নিশ্চিত করুন আপনি এটা আপনার অ্যাপলিকেশন মেনিফেস্টে ডিক্লেয়ার করেছেন। আরও তথ্যের জন্য Declare OpenGL ES Use in the Manifest দেখুন।

আপনার GLSurfaceView ইমপ্লিমেন্টেশনে অন্য আরেকটি ঐচ্ছিক সংযোজন হচ্ছে GLSurfaceView.RENDERMODE_WHEN_DIRTY সেটিং ব্যবহার করে শুধু ভিউ ড্র করার জন্য রেন্ডার মোড সেট করা যখন আপনার ড্রয়িং ডাটাতে পরিবর্তন থাকে:

```
// Render the view only when there is a change in the drawing data  
setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
```

আপনি যতক্ষণ না requestRender() কল করবেন এই সেটিং GLSurfaceView ফ্রেমকে রিড্র হওয়ার

থেকে বিরত রাখবে, যেটা এই স্যাম্পল অ্যাপের জন্য আধিক কার্যকর।

রেন্ডার ক্লাস তৈরী করা

GLSurfaceView.Renderer ক্লাসের বাস্তবায়ন, বা রেন্ডারার, একটি অ্যাপলিকেশনের মধ্যে যা OpenGL ES ব্যবহার করে যেখানে বিষয়বস্তু চমৎকার হওয়া শুরু হয়। এই ক্লাস GLSurfaceView এ ড্র করা হয় তা নিয়ন্ত্রণ করে যার সাথে এটা সংযুক্ত। একটি রেন্ডারের মধ্যে তিনটা পদ্ধতি আছে যাকে অ্যান্ড্রয়েড সিস্টেম কল করে, একটি GLSurfaceView এ কী এবং কীভাবে ড্র করতে হয় তা খুঁজে বের করার জন্য:

- onSurfaceCreated() – ভিউয়ের ওড়বহএখ উঝ বহারংড়হসবহঃ সেটআপ করতে একবার কল করা হয়
- onDrawFrame() - ভিউয়ের প্রতিটা রিড্র এর জন্য কল করা হয়
- onSurfaceChanged() – কল করা হয়, যদি ভিউয়ের জ্যামিতি পরিবর্তন করে, উদাহরণস্বরূপ যখন ডিভাইসের স্ক্রিন ওরিয়েন্টেশন পরিবর্তন করে

এখানে একটি OpenGL ES রেন্ডারের একটি মৌলিক বাস্তবায়ন আছে, যা GLSurfaceView এর মধ্যে একটি গ্রে ব্যাকগ্রাউন্ড ড্র করা ছাড়া আর বেশী কিছু করে না:

```
public class MyGLRenderer implements GLSurfaceView.Renderer {  
  
    public void onSurfaceCreated(GL10 unused, EGLConfig config) {  
        // Set the background frame color  
        GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);  
    }  
  
    public void onDrawFrame(GL10 unused) {  
        // Redraw background color  
        GLES20.glClear(GLES20.GL_COLOR_BUFFER_BIT);  
    }  
  
    public void onSurfaceChanged(GL10 unused, int width, int height) {  
        GLES20.glViewport(0, 0, width, height);  
    }  
}
```

এতে যা আছে তাই সব! উপরের কোড উদাহরণ একটি সহজ অ্যান্ড্রয়েড অ্যাপলিকেশন তৈরীর, যা OpenGL ব্যবহার করে একটি গ্রে স্ক্রিন প্রদর্শন করে। যখন এই কোড খুব চমৎকার কোন কিছু না করে, এই ক্লাসগুলো তৈরী করে, OpenGL দিয়ে ড্রয়িং গ্রাফিক্স এলিমেন্ট শুরু করার জন্য প্রয়োজনীয় ফাউন্ডেশন রাখে।

নোট: আপনি হয়তো অবাক হবেন, এই পদ্ধতিগুলোর কেন একটি GL10 প্যারামিটার থাকে, যখন আপনি OpenGL ES 2.0 APIs ব্যবহার করেন। এই পদ্ধতি সিগনেচার অ্যান্ড্রয়েড ফ্রেমওয়ার্ক কোড সিম্পলার ধরে রাখতে 2.0 APIs এর জন্য পুনব্যবহার হয়।

আপান যদি OpenGL ES APIs এর সাথে পারাচিত থাকেন, আপনার এখন আপনার অ্যাপে OpenGL ES পরিবেশ সেটআপ এবং গ্রাফিক্স ড্র করা শুরু করতে পারার সামর্থ থাকা উচিত। কিন্ত আপনার যদি OpenGL শুরু করার জন্য আর একটু সাহাজ্য প্রয়োজন হয়, আরও কিছু ইঙ্গিত পেতে পরবর্তী অনুশীলনীতে চলে যান।

গঠন/আকৃতি নির্ধারণ করা

(<http://developer.android.com/training/graphics/opengl/shapes.html>)

একটি OpenGL ES ভিউ এর প্রেক্ষাপটের মধ্যে অঙ্কিত হওয়া শেপ/গঠন নির্ধারণ করতে পারা হচ্ছে আপনার হাই-এন্ড গ্রাফিক্স মাস্টারপিস তৈরী করার প্রথম ধাপ। আপনার গ্রাফিক্স অবজেক্ট নির্ধারণ করাটা কীভাবে OpenGL ES আশা করে সে সম্পর্কে না জেনে OpenGL ES দিয়ে ড্র করাটা একটু জটিল হতে পারে।

এই অনুশীলনী একটি অ্যান্ড্রয়েড ডিভাইস স্ক্রিনের সাথে সম্পর্কিত OpenGL ES সমন্বয় সিস্টেম, একটি শেপ/গঠন, নির্ধারণের মৌলিক সূত্র, শেপ ফেস, একই সাথে একটি ত্রিভুজ এবং একটি বর্গক্ষেত্র বিষয়ে ব্যাখ্যা করে।

একটি ত্রিভুজ (ট্রায়াঙ্গেল) নির্ধারণ

OpenGL ES ত্রিমাত্রিক স্পেসে কোঅর্ডিনেটস ব্যবহার করে ড্র হওয়া অবজেক্ট নির্ধারণ করতে দেয়। সুতরাং একটি ত্রিভুজ অঙ্কন করতে পারার আগে, আপনাকে অবশ্যই এর কোঅর্ডিনেটস ঠিক করে নিতে হবে। OpenGL ES এর মধ্যে সাধারণভাবে এটা করা হচ্ছে কোঅর্ডিনেটস এর জন্য ফ্লোটিং পয়েন্ট নাম্বার এর একটি ভার্টিফাইড অ্যারে ঠিক করা। সর্বোচ্চ কার্যকারিতার জন্য, আপনি এই সকল কোঅর্ডিনেটস একটি ByteBuffer এর মধ্যে লিখেন, যা প্রসেস হওয়ার জন্য OpenGL ES গ্রাফিক্স পাইপলাইনে পাস হয়ে যায়।

```
public class Triangle {

    private FloatBuffer vertexBuffer;

    // number of coordinates per vertex in this array
    static final int COORDS_PER_VERTEX = 3;
    static float triangleCoords[] = { // in counterclockwise order:
        0.0f,  0.622008459f, 0.0f, // top
        -0.5f, -0.311004243f, 0.0f, // bottom left
        0.5f, -0.311004243f, 0.0f  // bottom right
    };

    // Set color with red, green, blue and alpha (opacity) values
    float color[] = { 0.63671875f, 0.76953125f, 0.22265625f, 1.0f };

    public Triangle() {
        // initialize vertex byte buffer for shape coordinates
        ByteBuffer bb = ByteBuffer.allocateDirect(
            // (number of coordinate values * 4 bytes per float)
            triangleCoords.length * 4);
        // use the device hardware's native byte order
        bb.order(ByteOrder.nativeOrder());

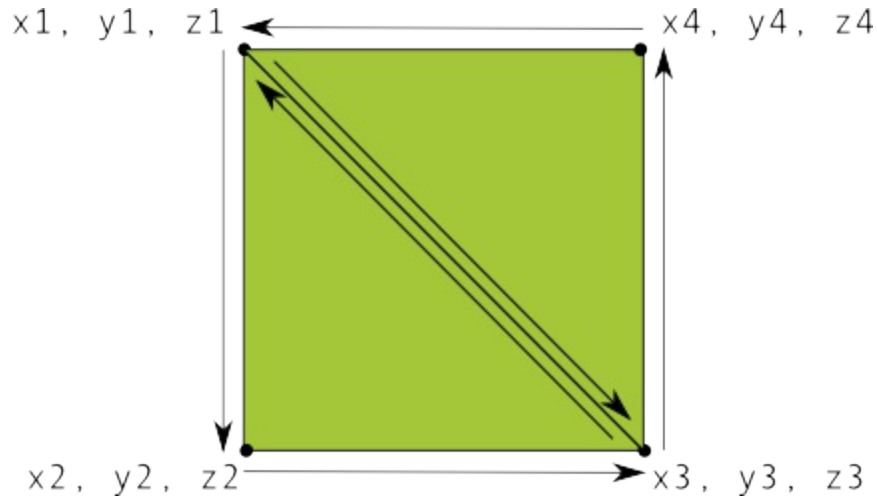
        // create a floating point buffer from the ByteBuffer
        vertexBuffer = bb.asFloatBuffer();
        // add the coordinates to the FloatBuffer
        vertexBuffer.put(triangleCoords);
        // set the buffer to read the first coordinate
        vertexBuffer.position(0);
    }
}
```

বাই ডিফল্ট, OpenGL ES মনে করে একটি কোঅর্ডিনেট সিস্টেম যেখানে [0,0,0] (X,Y,Z) GLSurfaceView ফ্রেমের কেন্দ্র নির্দিষ্ট করে দেয়, [1,1,0] ফ্রেমের উপরের ডান কোণ, [-1,-1,0] ফ্রেমের নীচের বাম কোণ নির্দিষ্ট করে। এই কোঅর্ডিনেট সিস্টেম সম্পর্কে আরও তথ্য পেতে OpenGL ES ডেভেলপার গাইড দেখুন।

উল্লেখ্য যে, এই গঠনের/শেপ কোঅর্ডিনেটস কাউন্টার-ক্লক-ওয়াইজ ক্রমে নির্ধারিত হয়। ড্র করার ক্রম থাকা গুরুত্বপূর্ণ কারণ এটা ঠিক করে শেপের কোন পাশটি ফ্রন্ট ফেস হবে, যেটা সাধারণভাবে আপনি ড্র হওয়াতে চান, এবং ব্যাক ফেস, যেটা ড্র না করার জন্য বেছে নেন, OpenGL ES কাল (cull) ফেস বৈশিষ্ট্য ব্যবহার করে। ফেস এবং কালিং সম্পর্কে আরও তথ্যের জন্য OpenGL ES ডেভেলপার গাইড দেখুন।

একটি বর্গক্ষেত্র (স্কয়ার) নির্ধারণ

OpenGL এ একটি ত্রিভুজ নির্ধারণ করা খুবই সহজ, কিন্ত আপনি যদি একটু বেশী জটিল কিছু চান তখন কী হবে? ধরুন বর্গক্ষেত্র? এটা করার অনেক উপায় আছে, কিন্ত OpenGL এর মধ্যে ঐ শেপ/গঠন অঙ্কন করার একটি সাধারণ পথ হচ্ছে দুইটা ত্রিভুজ এক সাথে অঙ্কন করা:



ফিগার ১. দুইটি ত্রিভুজ ব্যবহার করে একটি বর্গক্ষেত্র অঙ্কন করুন

পুনরায়, উভয় ত্রিভুজের জন্য একটি কাউন্টার-ক্লক-ওয়াইজ ক্রমের মধ্যে আপনার শীর্ষবিন্দুগুলো (vertices) নির্ধারণ করা যা এই শেপকে প্রতিনিধিত্ব করে, ভালো একটি ByteBuffer এর মধ্যে রাখে। প্রতিটা ত্রিভুজ কর্তৃক দুইবার শেয়ার হওয়া দুইটা কোঅর্ডিনেটস কে নির্ধারণ করা পরিহার করতে, OpenGL ES গ্রাফিক্স পাইপলাইন কে বলার জন্য যে এই শীর্ষবিন্দুগুলো (vertices) কীভাবে ড্র করতে হয় তার জন্য ড্রিং লিস্ট ব্যবহার করুন। এখানে এই শেপের জন্য কোড দেয়া হলো:

```
public class Square {

    private FloatBuffer vertexBuffer;
    private ShortBuffer drawListBuffer;

    // number of coordinates per vertex in this array
    static final int COORDS_PER_VERTEX = 3;
    static float squareCoords[] = {
        -0.5f,  0.5f,  0.0f,    // top left
        -0.5f, -0.5f,  0.0f,    // bottom left
        0.5f,  -0.5f,  0.0f,    // bottom right
        0.5f,   0.5f,  0.0f }; // top right

    private short drawOrder[] = { 0, 1, 2, 0, 2, 3 }; // order to draw vertices

    public Square() {
        // initialize vertex byte buffer for shape coordinates
        ByteBuffer bb = ByteBuffer.allocateDirect(
            // (# of coordinate values * 4 bytes per float)
            squareCoords.length * 4);
        bb.order(ByteOrder.nativeOrder());
        vertexBuffer = bb.asFloatBuffer();
        vertexBuffer.put(squareCoords);
        vertexBuffer.position(0);
```

```
        // initialize byte buffer for the draw list
        ByteBuffer dlb = ByteBuffer.allocateDirect(
            // (# of coordinate values * 2 bytes per short)
            drawOrder.length * 2);
        dlb.order(ByteOrder.nativeOrder());
        drawListBuffer = dlb.asShortBuffer();
        drawListBuffer.put(drawOrder);
        drawListBuffer.position(0);
    }
}
```

এই উদাহরণ আপনাকে OpenGL দিয়ে আরও জটিল শেপ তৈরী করা বিষয়গুলো দেখাবে। সাধারণভাবে, আপনি অবজেক্ট ড্র করতে ত্রিভুজের সংগ্রহ ব্যবহার করেন। পরবর্তী অনুশীলনীতে আপনি শিখবেন স্ক্রিনে কীভাবে এই শেপগুলো অঙ্কন করা হবে।

শেপ/আকৃতি অংকন করা

(<http://developer.android.com/training/graphics/opengl/draw.html>)

OpenGL দিয়ে অঙ্কন করার জন্য শেপ নির্ধারন করার পর, আপনি সম্ভবত তাদের অঙ্কন করতে চাইবেন। OpenGL 2.0 দিয়ে অঙ্কন করাটা আপনি যা চিন্তা করেছেন তার চেয়েও বেশী কোড নেয়, কারন এপিআই গ্রাফিক্স রেন্ডারিং পাইপলাইনের উপর আরও নিয়ন্ত্রন প্রদান করে।

এই অনুশীলনী ব্যাখ্যা করে পূর্ববর্তী অনুশীলনীতে নির্ধারন করা শেপ OpenGL ES 2.0 API ব্যবহার করে কীভাবে ড্র/অঙ্কন করা যায়।

শেপ/আকৃতি আরম্ভ করা

কোন ড্রয়িং/অঙ্কন করার পূর্বে, আপনি যে শেপ ড্র করার পরিকল্পনা করেছেন তা অবশ্যই আরম্ভ করা এবং লোড করতে হবে। যতক্ষণ না আপনার প্রোগ্রামে ব্যবহার করা শেপের কাঠামো (মূল কোঅরডিনেটস) কার্যনির্বাহের গতিপথ সময়কালীন পরিবর্তন না হয়, মেমরী এবং প্রসেসিং কার্যকারিতার জন্য আপনার উচিত তাদের আপনার রেন্ডারারের `onSurfaceCreated()` পদ্ধতির মধ্যে আরম্ভ করা

```
public void onSurfaceCreated(GL10 unused, EGLConfig config) {  
    ...  
  
    // initialize a triangle  
    mTriangle = new Triangle();  
    // initialize a square  
    mSquare = new Square();  
}
```

শেপ অংকন/ ড্র করা

OpenGL ES 2.0 ব্যবহার করে একটি নির্ধারিত শেপ অংকন করার জন্য একটি উল্লেখযোগ্য পরিমাণ কোড প্রয়োজন হয়, কারন আপনাকে অবশ্যই গ্রাফিক্স রেন্ডারিং পাইপলাইনে প্রচুর তথ্য-উপাত্ত দিতে হবে, বিশেষ করে নীচের বিষয়গুলো নির্ধারণ করে দিতে হবে:

- Vertex Shader – একটি শেপের শীর্ষবিন্দুগুলো (vertices) রেন্ডারিং করার জন্য OpenGL ES গ্রাফিক্স কোড
- Fragment Shader – কালার এবং টেক্সচার দিয়ে একটি শেপের ফেস রেন্ডারিং করার জন্য OpenGL ES গ্রাফিক্স কোড
- Program – একটি OpenGL ES অবজেক্ট যা শেডার ধারণ করে যা আপনি এক বা একাধিক শেপ ড্র করার জন্য ব্যবহার করতে চান

আপনার একটি শেপ ড্র করার জন্য কমপক্ষে একটি ভারটেক্স শেডার লাগবে এবং ঐ শেপ রঙ করতে একটি ফ্রাগমেন্ট শেডার লাগবে। এই শেডগুলো অবশ্যই সংকলিত হতে হবে এবং তারপর একটি OpenGL ES program এ যুক্ত হতে হবে, তারপর এটাকে শেপ ড্র করার কাজে ব্যবহার করা হবে। এখানে একটি উদাহরণ আছে, কীভাবে একটি মৌলিক শেডার নির্ধারণ করতে হয় যা আপনি একটি শেপ ড্র করার কাজে ব্যবহার করতে পারেন:

```
private final String vertexShaderCode =
    "attribute vec4 vPosition;" +
    "void main() {" +
    "    gl_Position = vPosition;" +
    "}";

private final String fragmentShaderCode =
    "precision mediump float;" +
    "uniform vec4 vColor;" +
    "void main() {" +
    "    gl_FragColor = vColor;" +
    "}";
```

শেডার OpenGL Shading Language (GLSL) কোড ধারণ করে যা OpenGL ES পরিবেশে ব্যবহারের পূর্বে অবশ্যই সংকলিত হতে হবে। এই কোড সংকলন করতে আপনার রেন্ডারার ক্লাসে একটি ইউটিলিটি পদ্ধতি তৈরি করুন:

```
public static int loadShader(int type, String shaderCode){

    // create a vertex shader type (GL_VERTEX_SHADER)
    // or a fragment shader type (GL_FRAGMENT_SHADER)
    int shader = GLES20.glCreateShader(type);

    // add the source code to the shader and compile it
    GLES20.glShaderSource(shader, shaderCode);
```

```

        GLES20.glCompileShader(shader);

        return shader;
    }

```

আপনার শেপ ড্র করার জন্য, আপনাকে অবশ্যই শেডার কোড সঙ্কলন করতে হবে, তাদের একটি OpenGL ES প্রোগ্রামে যুক্ত করুন এবং তারপর প্রোগ্রামটি লিংক করুন। এটা আপনার ড্র করা অবজেক্টের কনস্ট্রাকটরে করুন, ফলে এটা একবার ঘটবে।

নোট: সংকলিত OpenGL ES শেডার এবং লিংক করা প্রোগ্রাম সিপিইউ সাইকেল এবং প্রসেসকরা সময়ের শর্তে ব্যয়বহুল, তাই আপনাকে এটা একবারের বেশী করাটা পরিহার করতে হবে। আপনি যদি রান টাইমে আপনার শেডারের কনটেন্ট সম্পর্কে না জানেন, আপনার উচিত আপনার কোড তৈরী করা যেমনভাবে তারা শুধু তৈরী করা জিনিসটা পেয়ে থাকে এবং পরবর্তী ব্যবহারের জন্য জমিয়ে রাখে।

```

int vertexShader = loadShader(GLES20.GL_VERTEX_SHADER, vertexShaderCode);
int fragmentShader = loadShader(GLES20.GL_FRAGMENT_SHADER, fragmentShaderCode);

mProgram = GLES20.glCreateProgram();           // create empty OpenGL ES Program
GLES20.glAttachShader(mProgram, vertexShader); // add the vertex shader to program
GLES20.glAttachShader(mProgram, fragmentShader); // add the fragment shader to program
GLES20.glLinkProgram(mProgram);               // creates OpenGL ES program executables
}

```

এই মুহুর্তে, আপনি যথার্থ কল যুক্ত করতে প্রসূতত হয়েছেন যা আপনার শেপ ড্র করে। OpenGL ES দিয়ে শেপ অংকন চায় রেন্ডারিং পাইপলাইনকে বলতে পান যে আপনি কি ড্র করতে চান এবং কীভাবে এটা ড্র করতে চান সেই জন্য আপনি কিছু প্যারামিটার নির্দিষ্ট করুন। যেহেতু ড্রিং অপশন শেপ দ্বারা ভিন্ন ভিন্ন হতে পারে, তাই এটা একটা ভালো চিন্তা যে আপনার শেপ ক্লাস তাদেও নিজস্ব ড্রিং লজিক (যুক্তি) ধারন করে।

শেপ ড্র করার জন্য একটি draw() পদ্ধতি তৈরী করুন। এই কোড শেপের ভারটেক্স শেডার এবং ফ্রাগমেন্ট শেডারে অবস্থান (পজিশন) এবং কালার ভ্যালু সেট করে, এবং তারপর ড্র করার কাজ সম্পাদন করে।

```

public void draw() {
    // Add program to OpenGL ES environment
    GLES20.glUseProgram(mProgram);

    // get handle to vertex shader's vPosition member
    mPositionHandle = GLES20.glGetAttribLocation(mProgram, "vPosition");

    // Enable a handle to the triangle vertices
    GLES20.glEnableVertexAttribArray(mPositionHandle);

    // Prepare the triangle coordinate data
    GLES20.glVertexAttribPointer(mPositionHandle, COORDS_PER_VERTEX,
                                GLES20.GL_FLOAT, false,
                                vertexStride, vertexBuffer);

    // get handle to fragment shader's vColor member
    mColorHandle = GLES20.glGetUniformLocation(mProgram, "vColor");

    // Set color for drawing the triangle
}

```



```

GLS20.glUniform4fv(mColorHandle, 1, color, 0);

// Draw the triangle
GLS20.glDrawArrays(GLS20.GL_TRIANGLES, 0, vertexCount);

// Disable vertex array
GLS20.glDisableVertexAttribArray(mPositionHandle);
}

```

যখনই আপনার এই সকল কোড যথাস্থানে থাকবে, এই অবজেক্ট অংকন করতে শুধু আপনার রেন্ডারারের `onDrawFrame()` পদ্ধতির মধ্যে থেকে পদ্ধতিতে একটি কল করা প্রয়োজন। যখন আপনি অ্যাপলিকেশনটি রান করবেন, এটা দেখতে অনেকটা এরকম দেখাবে:



ফিগার ১. একটি প্রজেকশন বা ক্যামেরা ভিউ ছাড়া অংকিত ত্রিভুজ

এই কোড উদাহরনে কিছু সমস্যা আছে। প্রথমটি হচ্ছে, এটা আপনার বনুধদেরকে কাছে এটা কোন প্রভাব ফেলবে না। দ্বিতীয়ত, ত্রিভুজটি একটু চাপাচাপি অবস্থায় থাকে এবং আপনি যখন ডিভাইসের স্ক্রিন ওরিয়েন্টেশন পরিবর্তন করবেন তখন এটা শেপ পরিবর্তন করে। ত্রিভুজটি অসমান (স্কিউড) হওয়ার কারন কিছু ঘটনা যা হচ্ছে অবজেক্টের শীর্ষবিন্দুগুলোর (vertices) স্ক্রিন এলাকা যেখানে `GLSurfaceView` প্রদর্শিত হয় সেখানকার অনুপাত সঠিকভাবে হয় নাই। পরবর্তী অনুশীলনীতে একটি প্রজেকশন এবং ক্যামেরা ভিউ ব্যবহার করে আপনি এই সমস্যা ঠিক করতে পারেন।

সর্বশেষটা হচ্ছে ত্রিভুজটি, অনড়, ডেটা একটু বিরক্তিকর। Adding Motion অনুশীলনীতে আপনি এই শেপকে রোটেট করতে পারবেন এবং OpenGL ES গ্রাফিক্স পাইপলাইন এর ব্যবহার আরও আকর্ষণীয় করবে।

প্রজেকশন এবং ক্যামেরা ভিউ প্রয়োগ

(<http://developer.android.com/training/graphics/opengl/projection.html>)

OpenGL ES পরিবেশে, প্রজেকশন এবং ক্যামেরা ভিউ অংকনকৃত অবজেক্ট একটা উপায়ে প্রদর্শন করতে দেয় যা আপনি নিজের চোখে বাস্তবে যা দেখেন তার খুব কাচাকাছি স্বদৃশ্যপূর্ণ। বাস্তবের মতো করে দেখার বিষয়টি ঘটে অংকিত অবজেক্ট কোঅরডিনেটস এর গাণিতিক রূপান্তরের মাধ্যমে:

- **প্রজেকশন-** এই ট্রান্সফর্মেশন GLSurfaceView যেখানে তারা প্রদর্শিত হয় তার দৈর্ঘ্য এবং উচ্চতার উপর ভিত্তি করে অংকিত অবজেক্টের কোঅরডিনেটস সমন্বয় করে। এই ক্যালকুলেশন ব্যতীত, OpenGL ES দ্বারা অংকিত অবজেক্ট ভিউ উইন্ডোর অসম অনুপাতের দ্বারা অসমান হয়। একটি প্রজেকশন ট্রান্সফর্মেশন সাধারণভাবে শুধুমাত্র তখন ক্যালকুলেট হওয়া উচিত যখন OpenGL ভিউ এর অনুপাত আপনার রেন্ডারার এর onSurfaceChanged() পদ্ধতির মধ্যে প্রতিষ্ঠিত হয় বা পরিবর্তিত হয়। OpenGL ES প্রজেকশন এবং কোঅরডিনেট ম্যাপিং সম্পর্কে আরও তথ্যের জন্য Mapping Coordinates for Drawn Objects দেখুন।
- **ক্যামেরা ভিউ-** এই ট্রান্সফর্মেশন একটি ভার্চুয়াল ক্যামেরা অবস্থানের উপর ভিত্তি করে অংকিত অবজেক্টের কোঅরডিনেটস সমন্বয় করে। এটা খেয়াল করা জরুরী, OpenGL ES একটি আসল ক্যামেরা অবজেক্ট নির্ধারণ করে না, কিন্তু পরিবর্তে ইউটিলিটি পদ্ধতি সরবরাহ করে যা অংকিত অবজেক্টের প্রদর্শনের ট্রান্সফর্মেশন ঘটানোর দ্বারা একটি ক্যামেরা কে অনুকরণ করে। একটি ক্যামেরা ভিউ ট্রান্সফর্মেশন শুধু একবার ক্যালকুলেট হয় যখন আপনি আপনার GLSurfaceView প্রতিষ্ঠিত করেন, বা ইউজারের একশন এবং আপনার অ্যাপলিকেশনের ফাংশন এর উপর ভিত্তি করে সক্রিয় পরিবর্তন করে।

এই অনুশীলনী আলোচনা করে কীভাবে একটি প্রজেকশন এবং ক্যামেরা ভিউ তৈরী করতে হয় এবং আপনার GLSurfaceView এর মধ্যে শেপ অংকন করতে কীভাবে এটা প্রয়োগ করতে হয়।

একটি প্রজেকশন নির্ধারণ

একটি প্রজেকশন রূপান্তরের জন্য ডাটা আপনার GLSurfaceView.Renderer ক্লাসের onSurfaceChanged() পদ্ধতির মধ্যে ক্যালকুলেট হয়। নিম্নোক্ত উদাহরণ কোড GLSurfaceView এর উচ্চতা এবং দৈর্ঘ্য নেয় এবং Matrix.frustumM() পদ্ধতি ব্যবহার করে একটি প্রজেকশন ট্রান্সফর্মেশন Matrix পূর্ণ করতে এটা ব্যবহার করে:

```
@Override
public void onSurfaceChanged(GL10 unused, int width, int height) {
    GLES20.glViewport(0, 0, width, height);

    float ratio = (float) width / height;

    // this projection matrix is applied to object coordinates
    // in the onDrawFrame() method
    Matrix.frustumM(mProjectionMatrix, 0, -ratio, ratio, -1, 1, 3, 7);
}
```

এই কোড একটি প্রজেকশন মেট্রিক্স mProjectionMatrix পূর্ণ করে যা আপনি পরবর্তীতে onDrawFrame() পদ্ধতির মধ্যে একটি ক্যামেরা ভিউ রূপান্তরের সাথে মিলিত করতে পারবেন, যা পরবর্তী অধ্যায়ে আলোচনা করা হবে।

নোট: আপনার অংকিত অবজেক্টে শুধু একটি প্রজেকশন ট্রান্সফর্মেশন প্রয়োগ করার সাধারণ ফলাফল হচ্ছে একটি খালি ডিসপ্লে। সাধারণভাবে, স্ক্রিনে কোন কিছু প্রদর্শনের জন্য আপনাকে অবশ্যই একটি ক্যামেরা ভিউ ট্রান্সফর্মেশন প্রয়োগ করতে হবে।

একটি ক্যামেরা ভিউ নির্ধারণ

অংকন করার প্রক্রিয়ার অংশ হিসাবে ক্যামেরা ভিউ ট্রান্সফর্মেশন যুক্ত করার মাধ্যমে আপনার অংকিত অবজেক্টের রূপান্তরিত করার প্রক্রিয়া সম্পূর্ণ করুন। নিচের উদাহরণ কোডের মধ্যে `Matrix.setLookAtM()` পদ্ধতি ব্যবহার করে ক্যামেরা ভিউ ট্রান্সফর্মেশন ক্যালকুলেট হয় এবং পূর্বে ন্যায় ক্যালকুলেট হওয়া প্রজেকশন মেট্রিক্স এর সাথে সম্মিলিত হয়। সম্মিলিত ট্রান্সফর্মেশন মেট্রিক্স তখন প্রেস করা হয় শেপ অংকন করার জন্য।

```
@Override
public void onDrawFrame(GL10 unused) {
    ...
    // Set the camera position (View matrix)
    Matrix.setLookAtM(mViewMatrix, 0, 0, 0, -3, 0f, 0f, 0f, 0f, 1.0f, 0.0f);

    // Calculate the projection and view transformation
    Matrix.multiplyMM(mMVPMatrix, 0, mProjectionMatrix, 0, mViewMatrix, 0);

    // Draw shape
    mTriangle.draw(mMVPMatrix);
}
```

প্রজেকশন এবং ক্যামেরা ট্রান্সফর্মেশন প্রয়োগ

পূর্ববর্তী অধ্যায়ে দেখানো সম্মিলিত প্রজেকশন এবং ক্যামেরা ভিউ ট্রান্সফর্মেশন মেট্রিক্স ব্যবহার করার জন্য, সম্মিলিত ট্রান্সফর্মেশন মেট্রিক্স গ্রহন করতে এবং শেপে এটা প্রয়োগ করতে আপনার গ্রাফিক অবজেক্টের draw() পদ্ধতি পরিমিত করুন:

```
public void draw(float[] mvpMatrix) { // pass in the calculated transformation matrix
    ...

    // get handle to shape's transformation matrix
    mMVPMatrixHandle = GLES20.glGetUniformLocation(mProgram, "uMVPMatrix");

    // Pass the projection and view transformation to the shader
    GLES20.glUniformMatrix4fv(mMVPMatrixHandle, 1, false, mvpMatrix, 0);

    // Draw the triangle
    GLES20.glDrawArrays(GLES20.GL_TRIANGLES, 0, vertexCount);
    ...
}
```

যখনই আপনি সঠিকভাবে প্রজেকশন এবং ক্যামেরা ভিউ ট্রান্সফর্মেশন ক্যালকুলেট এবং প্রয়োগ করতে পারবেন, আপনার গ্রাফিক অবজেক্ট সঠিক অনুপাতে অংকিত হবে এবং দেখতে এরকম হবে:



ফিগার ১. একটি প্রজেকশন এবং ক্যামেরা ভিউ প্রয়োগ করে ত্রিভুজ অংকন।

এখন আপনার একটি অ্যাপলিকেশন আছে যা আপনার শেপ সঠিক অনুপাতে প্রদর্শন হবে, এখন আপনার শেপে গতিশীলতা (মোশন) যুক্ত করার সময়।

গতিশীলতা (মোশন) যুক্ত করা

(<http://developer.android.com/training/graphics/opengl/motion.html>)

স্ক্রিনে অবজেক্ট অংকন করা OpenGL এর একটি মৌলিক কাজ, কিন্ত আপনি অন্য অ্যান্ড্রয়েড গ্রাফিক্স ফ্রেমওয়ার্ক ক্লাস দিয়েও এটা করতে পারবেন যার মধ্যে রয়েছে Canvas এবং Drawable অবজেক্ট। ত্রিমাত্রিক বা অন্য স্বতন্ত্র উপায়ে বাধ্যগত ইউজার এক্সপেরিয়েন্স তৈরী করতে অংকিত অবজেক্টকে চলন্ত এবং রূপান্তরিত করার জন্য OpenGL ES অতিরিক্ত সামর্থ প্রদান করে।

এই অনুশীলনীতে, রোটেশন দিয়ে কীভাবে একটি শেপে গতিশীলতা (মোশন) যুক্ত করা যায় এটা শেখার মাধ্যমে OpenGL ES ব্যবহারের ক্ষেত্রে আরেক পা সামনে আগাবেন।

শেপ রোটেট করা

OpenGL ES 2.0 দিয়ে একটি অংকিত অবজেক্ট রোটেট করা তুলনামূলকভাবে সহজ। আপনি অন্য আরেকটি ট্রান্সফরমেশন মেট্রিক্স (একটি রোটেশন মেট্রিক্স) তৈরী করুন এবং তারপর আপনার প্রজেকশন এবং ক্যামেরা ভিউ ট্রান্সফরমেশন মেট্রিক্স এর সাথে এটাকে সম্মিলিত করুন:

```
private float[] mRotationMatrix = new float[16];
public void onDrawFrame(GL10 gl) {
    ...
    float[] scratch = new float[16];

    // Create a rotation transformation for the triangle
    long time = SystemClock.uptimeMillis() % 4000L;
    float angle = 0.090f * ((int) time);
    Matrix.setRotateM(mRotationMatrix, 0, angle, 0, 0, -1.0f);

    // Combine the rotation matrix with the projection and camera view
    // Note that the mMVPMatrix factor *must* be first* in order
    // for the matrix multiplication product to be correct.
    Matrix.multiplyMM(scratch, 0, mMVPMatrix, 0, mRotationMatrix, 0);

    // Draw triangle
    mTriangle.draw(scratch);
}
```

যদি আপনার ত্রিভুজ এই পরিবর্তন করার পর রোটেট না করে, নিশ্চিত করুন আপনি GLSurfaceView.RENDERMODE_WHEN_DIRTY সেটিং কमेंট আউট করেছেন, যেভাবে পরবর্তী অধ্যায়ে আলোচনা করা হয়েছে।

অবিরাম রেন্ডারিং সক্রিয় করা

যদি আপনি এই পয়েন্টে এই ক্লাসের মধ্যে উদাহরণ কোড সযতেœ অনুসরন করেন, নিশ্চিত করুন আপনি লাইন কমেন্ট আউট করেন যা রেন্ডার মোড সেট করে শুধুমাত্র যখন অংকন অপরিষ্কার হয়, অন্যথায় শুধুমাত্র একবার শেপ রোট্টেট বৃদ্ধি করা হয় এবং তারপর GLSurfaceView কনটেইনার থেকে requestRender() প্রতি একটি কলের জন্য অপেক্ষা করা হয়:

```
public MyGLSurfaceView(Context context) {  
    ...  
    // Render the view only when there is a change in the drawing data.  
    // To allow the triangle to rotate automatically, this line is commented out:  
    //setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);  
}
```

যদি না কোন ইউজারের সাথে যোগাযোগ ছাড়াই আপনার অবজেক্ট পরিবর্তন থাকে, এটা সাধারনত ভালো চিন্তা মানদন্ড চালু রাখা। এই কোড আনকমেন্ট করার জন্য প্রসূতত হোন, কারন পরবর্তী অনুশীলনী এই কল কে আর একবার প্রাসঙ্গিক করবে।

টচ ইভেন্টে রেসপন্স করা

(<http://developer.android.com/training/graphics/opengl/touch.html>)

ত্রিভুজ রোট্ট করার মতো একটি পূর্বপরিকল্পিত প্রোগ্রাম অনুসারে অবজেক্ট ম্যুভ করা, কিছু মনোযোগ পাওয়ার জন্য উপকারী, কিন্ত আপনি যদি চান আপনার OpenGL ES graphics সাথে ইউজারের পারস্পরিক যোগাযোগ থাকুক তখন কী হবে? আপনার OpenGL ES অ্যাপলিকেশন টাচ যোগাযোগ তৈরী করার উপায়, টাচ ইভেন্ট এর জন্য লিসেন করতে `onTouchEvent()` ওভাররাইড করতে `GLSurfaceView` বাস্তবায়ন প্রসারিত করছে।

এই অনুশীলনী আপনাকে দেখায় ইউজারকে একটি OpenGL ES অবজেক্টকে রোট্ট করতে দিয়ে টাচ ইভেন্টের জন্য লিসেন করতে হয়।

টাচ লিসেনার সেটআপ

আপনার OpenGL ES অ্যাপলিকেশনকে টাচ ইভেন্টে রেসপন্স করার জন্য তৈরী করতে , আপনাকে অবশ্যই আপনার GLSurfaceView ক্লাসে onTouchEvent() পদ্ধতি বাস্তবায়ন করতে হবে। নীচের বাস্তবায়ন উদাহরন দেখায় কীভাবে MotionEvent.ACTION_MOVE ইভেন্ট লিসেন করতে হয় এবং একটি শেপের জন্য রোটেশনের একটি এ্যাঙ্গেলে তাদের অনুবাদ করতে হয়।

```
@Override
public boolean onTouchEvent(MotionEvent e) {
    // MotionEvent reports input details from the touch screen
    // and other input controls. In this case, you are only
    // interested in events where the touch position changed.

    float x = e.getX();
    float y = e.getY();

    switch (e.getAction()) {
        case MotionEvent.ACTION_MOVE:

            float dx = x - mPreviousX;
            float dy = y - mPreviousY;

            // reverse direction of rotation above the mid-line
            if (y > getHeight() / 2) {
                dx = dx * -1 ;
            }

            // reverse direction of rotation to left of the mid-line
            if (x < getWidth() / 2) {
                dy = dy * -1 ;
            }

            mRenderer.setAngle(
                mRenderer.getAngle() +
                ((dx + dy) * TOUCH_SCALE_FACTOR); // = 180.0f / 320
            requestRender();

        }

        mPreviousX = x;
        mPreviousY = y;
        return true;
    }
}
```

উল্লেখ্য যে রোটেশন এ্যাঙ্গেল ক্যালকুলেশন করার পর, এই পদ্ধতি requestRender() কে কল করে রেন্ডারকে বলতে চায় যে এটা ফ্রেমকে রেন্ডার করার সময়। এই উদাহরনে এই অ্যাপ্রোচ সবচেয়ে কার্যকরী কারন ফ্রেমটিকে নতুন করে অংকন করার কোন প্রয়োজন নেই যদি না সেখানে রোটেশনে একটি পরিবর্তন হয়। কিন্তু, এটা কার্যকারিতার উপর কোন প্রভাব ফেলে না যদি না আপনি আরও রিকোয়েস্ট করেন যে রেন্ডারার তখনই রিড্র (পুনরায় অংকন) করতে হবে যখন setRenderMode() পদ্ধতি ব্যবহার করে পরিবর্তন করে, সুতরাং নিশ্চিত করুন যে এই লাইন রেন্ডারের মধ্যে আনকমেন্টেড হয়:

```
public MyGLSurfaceView(Context context) {
```

```
...  
// Render the view only when there is a change in the drawing data  
setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);  
}
```

রোটেশন অ্যাংগেল উন্মোচন করুন

উপরের উদাহরন কোড চায় যে একটি পাবলিক মেম্বার যুক্ত করার মাধ্যমে আপনার রেন্ডারারের মধ্যে দিয়ে আপনি রোটেশন অ্যাংগেল উন্মোচিত করুন। যেহেতু রেন্ডারার কোড আপনার অ্যাপলিকেশনের প্রধান ইউজার ইন্টারফেস থ্রেড থেকে একটি পৃথক থ্রেডে রান করছে, আপনাকে অবশ্যই এই পাবলিক ভেরিয়েবলকে volatile হিসাবে ডিক্লেয়ার করতে হবে। এখানে এটা করার একটি কোড দেওয়া হলো:

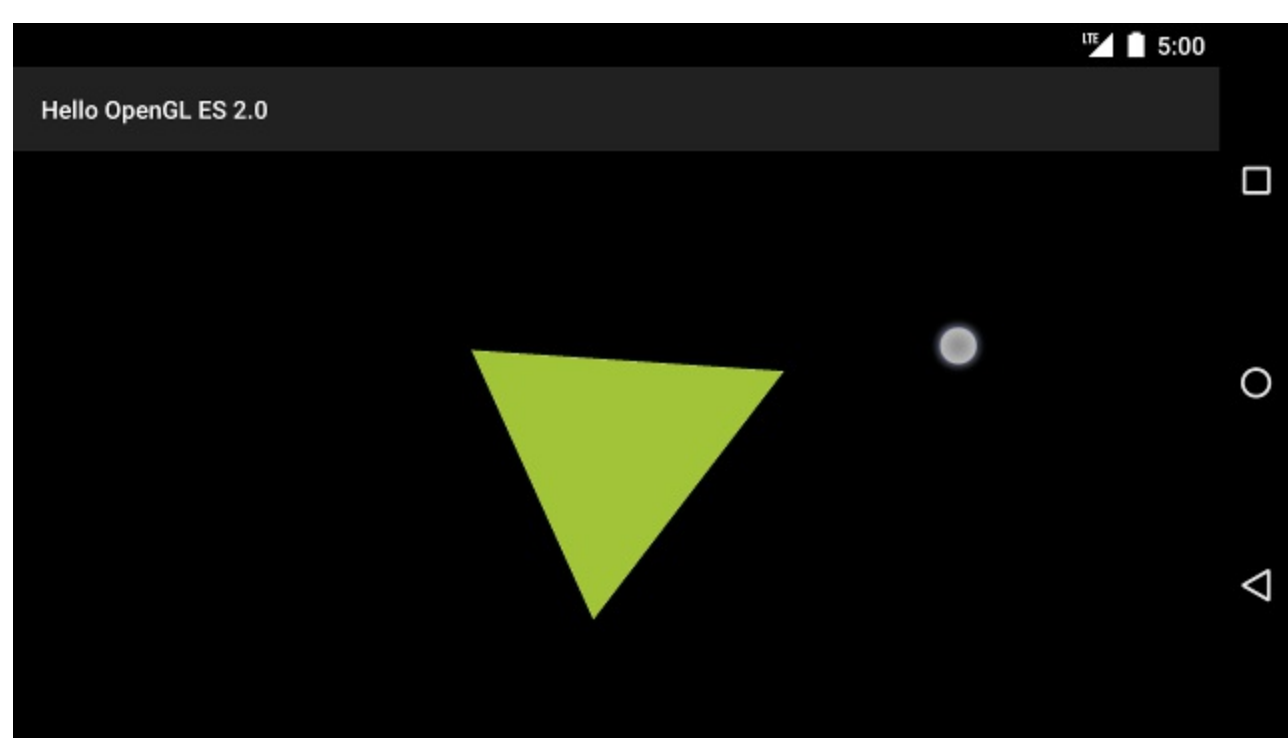
```
public class MyGLRenderer implements GLSurfaceView.Renderer {  
    ...  
    public volatile float mAngle;
```

রোটেশন প্রয়োগ

টাচ পয়েন্ট দিয়ে তৈরী রোটেশন প্রয়োগ করতে, কোডটি কমেন্ট আউট করুন যা একটি অ্যাংগেল তৈরী করে এবং `mAngle` যুক্ত করে যা টাচ পয়েন্ট দিয়ে তৈরী অ্যাংগেল ধারণ করে:

```
public void onDrawFrame(GL10 gl) {  
    ...  
    float[] scratch = new float[16];  
  
    // Create a rotation for the triangle  
    // long time = SystemClock.uptimeMillis() % 4000L;  
    // float angle = 0.090f * ((int) time);  
    Matrix.setRotateM(mRotationMatrix, 0, mAngle, 0, 0, -1.0f);  
  
    // Combine the rotation matrix with the projection and camera view  
    // Note that the mMVPMatrix factor *must be first* in order  
    // for the matrix multiplication product to be correct.  
    Matrix.multiplyMM(scratch, 0, mMVPMatrix, 0, mRotationMatrix, 0);  
  
    // Draw triangle  
    mTriangle.draw(scratch);  
}
```

যখন আপনি উপরে বর্ণিত ধাপগুলো সম্পূর্ণ করবেন, প্রোগ্রাম রান করবেন এবং ত্রিভুজ (ট্রায়াঙ্গল) রোটেশন করতে স্ক্রিন জুড়ে আপনার আঙ্গুল টেনে আনুন:



ফিগার ১. টাচ পয়েন্ট দিয়ে ত্রিভুজ রোটেশন হচ্ছে (বৃত্তটি টাচ লোকেশন দেখাচ্ছে)

অ্যানিমেশন সংযোজন করা

(<http://developer.android.com/training/animation/index.html>)

অ্যানিমেশন সুক্ষ্ণে ভিজ্যুয়াল কিউ যোগ করতে পারে যা ইউজারকে জানায় যে আপনার অ্যাপে কি হচ্ছে এবং আপনার অ্যাপের ইন্টারফেসের মেন্টাল মডেলকে উত্তম করে। অ্যানিমেশন বিশেষভাবে উপকারী যখন স্ক্রিন অবস্থা (স্টেট) পরিবর্তন করে, যেমন যখন কন্টেন্ট লোড করে বা নতুন একশন সহজলভ্য হয়। অ্যানিমেশন আপনার অ্যাপের একটি সুশোভিত দৃশ্যমানতা সংযোজনও করতে পারে, যা আপনার অ্যাপকে একটি উচ্চগুণ সম্পন্ন হওয়ার অনুভূতি দিবে।

যদিও মনে রাখবেন, যে অ্যানিমেশনের অতিব্যবহার বা তাদের অসময়ে ব্যবহার করাটা ক্ষতিকারক হতে পারে, যেমন তারা যখন বিলম্ব করে। এই প্রশিক্ষণ ক্লাস আপনাকে দেখাবে কীভাবে কিছু সাধারণ ধরনের অ্যানিমেশন বাস্তবায়ন করা যায় যা ব্যবহারযোগ্যতা বৃদ্ধি করে এবং আপনার ইউজারকে বিরক্ত না করেই ফ্লোর সংযোজন করে।

অনুশীলনীসমূহ

দুইটা ভিউকে ক্রসফেড (একটার পরে আরেকটি আসা) করা

শিখুন কীভাবে দুইটা ওভারল্যাপ করা ভিউকে ক্রসফেড করা হয়। এই অনুশীলনী দেখায় একটি ভিউয়ে একটি উন্নয়ন মানদন্ডকে ক্রসফেড করতে হয় যা টেক্সট কনটেন্ট ধারণ করে।

স্ক্রিন স্লাইডের জন্য ভিউ পেজ ব্যবহার করুন

শিখুন কীভাবে স্লাইডিং ট্রানজিশন (স্লাইড পরিবর্তন) অনুভূমিকভাবে সন্নিহিত স্ক্রিনেগুলোর মধ্যে অ্যানিমেট করা যায়

কার্ড ফ্লিপ অ্যানিমেশন প্রদর্শন

শিখুন কীভাবে দুইটা ভিউয়ের মধ্যে একটি ফ্লিপিং মোশন সহকারে অ্যানিমেট করা যায়

ভিউকে জুম করা

শিখুন কীভাবে একটি ভিউকে একটি টাচ-টু-জুম-অ্যানিমেশন দিয়ে বড় করা যায়।

লেআউট পরিবর্তন অ্যানিমেশন করা

শিখুন কীভাবে বিল্ট-ইন-অ্যানিমেশন সক্রিয় করতে হয় যখন লেআউটে চাইল্ড ভিউ কে সংযোজন, বিয়োজন বা আপডেট করা হয়।

দুইটা ভিউকে ক্রসফেড করা

(<http://developer.android.com/training/animation/crossfade.html>)

ক্রসফেড অ্যানিমেশন (ডিজলভ হিসাবেও পরিচিত) ধীরে ধীরে একটি ইউজার ইন্টারফেস কম্পোনেন্টকে অদৃশ্য করতে থাকে যখন একই সাথে অন্য আরেকটিকে দৃশ্যমান করা হয়, এই অ্যানিমেশন পরিস্থিতির জন্য উপকারী যেখানে আপনি কনটেন্ট বা ভিউকে আপনার অ্যাপে বিনিময় করতে চান। ক্রসফেড খুবই সুক্ষ্ম এবং ছোট কিনুত একটি স্ক্রিন থেকে পরবর্তীটিতে সহজভাবে পরিবর্তন হওয়ার সুযোগ প্রস্তাব করে। যখন আপনি এগুলো ব্যবহার করছেন না, তথাপি পরিবর্তন মাঝে মাঝে হঠাৎ বা ব্যস্ততা অনুভব করে।

এখানে কিছু টেক্সট কনটেন্ট একটি উন্নয়ন মানদণ্ড থেকে একটি ক্রসফেডের উদাহরণ দেয়া আছে।

Crossfade animation

আপনি যদি সামনে এগিয়ে যেতে চান এবং একটি পূর্ণ কার্যরত উদাহরণ দেখতে চান, স্যাম্পল/নমুনা অ্যাপটি ডাউনলোড (<http://developer.android.com/shareables/training/Animations.zip>) এবং রান করুন এবং ক্রসফেড উদাহরণ নির্বাচন করুন, কোড বাস্তবায়নের জন্য নিম্নোক্ত ফাইলগুলো দেখুন:

- src/CrossfadeActivity.java
- layout/activity_crossfade.xml
- menu/activity_crossfade.xml

ভিউগুলো তৈরী করুন

ভিউ দুইটি তৈরী করুন যা ক্রসফেড করতে চায়। নিচের উদাহরণ একটি উন্নয়ন মানদণ্ড তৈরী করে এবং একটি স্ক্রল করা যায় এমন টেক্সট ভিউ তৈরী করে:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView style="?android:textAppearanceMedium"
            android:lineSpacingMultiplier="1.2"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="@string/lorem_ipsum"
            android:padding="16dp" />

    </ScrollView>

    <ProgressBar android:id="@+id/loading_spinner"
        style="?android:progressBarStyleLarge"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center" />

</FrameLayout>
```

অ্যানিমেশন সেটআপ করা

অ্যানিমেশন সেট আপ করতে:

1. ভিউয়ের জন্য মেম্বার ভেরিয়েবল তৈরী করুন যা আপনি ক্রসফেড করতে চান। আপনার পরবর্তীতে এইসকল রেফারেন্স প্রয়োজন হবে যখন অ্যানিমেশনের সময়কালে ভিউ পরিমিত করা হবে।
2. ভিউয়ের জন্য যা দৃশ্যমান হচ্ছে, এর দৃশ্যমানতাকে GONE এ সেট করুন। এটা ভিউকে লেআউট স্পেস নেয়া থেকে বিরত রাখবে এবং এটা লেআউট ক্যালকুলেশন থেকে বাদ দিয়ে দিবে, প্রসেসকে আরও দ্রুততর করবে।
3. একটি মেম্বার ভেরিয়েবলের মধ্যে config_shortAnimTime সিস্টেম প্রপারটি জমিয়ে রাখুন (ক্যাশে)। এই প্রপারটি অ্যানিমেশনের জন্য একটি মানসম্পন্ন “ছোট” ("short") সময়কাল নির্ধারণ করে দেয়। এই সময়কাল সূক্ষ্ম অ্যানিমেশন বা যে অ্যানিমেশন বারবার ঘটে তার জন্য আদর্শ। config_longAnimTime এবং config_mediumAnimTime টাও সহজপ্রাপ্য যদি আপনি তাদের ব্যবহার করার ইচ্ছা প্রকাশ করেন।

একটিভিটি কনটেন্ট ভিউ হিসাবে পূর্ববর্তী কোড খন্ডচিত্র থেকে লেআউট ব্যবহার করার একটি উদাহরণ এখানে দেয়া আছে:

```
public class CrossfadeActivity extends Activity {  
  
    private View mContentView;  
    private View mLoadingView;  
    private int mShortAnimationDuration;  
  
    ...  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_crossfade);  
  
        mContentView = findViewById(R.id.content);  
        mLoadingView = findViewById(R.id.loading_spinner);  
  
        // Initially hide the content view.  
        mContentView.setVisibility(View.GONE);  
  
        // Retrieve and cache the system's default "short" animation time.  
        mShortAnimationDuration = getResources().getInteger(  
            android.R.integer.config_shortAnimTime);  
    }  
}
```

ভিউ ক্রসফেড

এখন ভিউগুলো সম্পূর্ণভাবে সেটআপ করা হয়েছে, নীচের কাজগুলো করে তাদের ক্রসফেড করুন:

1. যে ভিউ দৃশ্যমান হতে যাচ্ছে (fading in) তার জন্য, ০ এ আলফা ভ্যালু এবং দৃশ্যমানতা VISIBLE এ সেট করুন। (মনে রাখবেন যে, এটা প্রাথমিকভাবে GONE এ সেট করা হয়েছিল) এটা ভিউকে দৃশ্যমান করবে কিনুত সম্পূর্ণভাবে স্বচ্ছ থাকবে।
2. যে ভিউ দৃশ্যমান হতে যাচ্ছে তার জন্য, ০ থেকে ১ এ এর আলফা ভ্যালু অ্যানিমেট করুন। একই সময়ে যে ভিউ বিবর্ন হয়ে গেছে (fading out) তার জন্য আলফা ভ্যালু ১ থেকে ০ তে সেট করুন।
3. একটি Animator.AnimatorListener এর মধ্যে onAnimationEnd() ব্যবহার করে, ভিউয়ের দৃশ্যমানতাকে সেট করুন যা GONE এ অদৃশ্য হয়েছিল। এমনকি যদিও আলফা ভ্যালু হচ্ছে ০, GONE এ ভিউয়ের দৃশ্যমানতাকে সেট করাটা লেআউট স্পেস নেয়া থেকে বিরত রাখবে এবং এটা লেআউট ক্যালকুলেশন থেকে বাদ দিয়ে দিবে, প্রসেসকে আরও দ্রুততর করবে।

নীচের পদ্ধতিটি এটা কীভাবে করতে হয় তার একটি উদাহরণ দেখায়: private View mContentView; private View mLoadingView; private int mShortAnimationDuration;

...

```
private void crossfade() {  
  
    // Set the content view to 0% opacity but visible, so that it is visible  
    // (but fully transparent) during the animation.  
    mContentView.setAlpha(0f);  
    mContentView.setVisibility(View.VISIBLE);  
  
    // Animate the content view to 100% opacity, and clear any animation  
    // listener set on the view.  
    mContentView.animate()  
        .alpha(1f)  
        .setDuration(mShortAnimationDuration)  
        .setListener(null);  
  
    // Animate the loading view to 0% opacity. After the animation ends,  
    // set its visibility to GONE as an optimization step (it won't  
    // participate in layout passes, etc.)  
    mLoadingView.animate()  
        .alpha(0f)  
        .setDuration(mShortAnimationDuration)  
        .setListener(new AnimatorListenerAdapter() {  
            @Override  
            public void onAnimationEnd(Animator animation) {  
                mLoadingView.setVisibility(View.GONE);  
            }  
        })  
        .setListener(null);  
}
```

স্ক্রিন স্লাইড এর জন্য ভিউ পেজ ব্যবহার

(<http://developer.android.com/training/animation/screen-slide.html>)

স্ক্রিন স্লাইড একটি পূর্ণ স্ক্রিন থেকে আরেকটি স্ক্রিনের মধ্যে পরিবর্তন হয় এবং ইউআই এ এটা খুব সাধারণ যেমন সেটআপ উইজার্ড বা স্লাইড শো। এই অনুশীলনী দ্বারা প্রদত্ত একটি ViewPager দিয়ে কীভাবে এটা করতে হয় তা আপনাকে দেখায়। ViewPagers স্বয়ংক্রিয়ভাবে স্ক্রিন স্লাইড অ্যানিমেন্ট করতে পারে। এখানে আছে একটি স্ক্রিন স্লাইড দেখতে কেমন হয় যা একটি কনটেন্টের স্ক্রিন থেকে পরবর্তীটাতে পরিবর্তন করে:

Screen slide animation / স্ক্রিন স্লাইড অ্যানিমেশন

আপনি যদি সামনে এগিয়ে যেতে চান এবং একটি সম্পূর্ণ কার্যরত উদাহরণ দেখেন, নমুনা অ্যাপটি <http://developer.android.com/shareables/training/Animations.zip> করুন এবং রান করুন এবং স্ক্রিন স্লাইড উদাহরণ নির্বাচন করুন। কোড বাস্তবায়নের জন্য নীচের ফাইলগুলো দেখুন:

- src/ScreenSlidePageFragment.java
- src/ScreenSlideActivity.java
- layout/activity_screen_slide.xml
- layout/fragment_screen_slide_page.xml

ভিউ তৈরী করুন

একটি লেআউট ফাইল তৈরী করুন যা আপনি পরবর্তীতে একটি ফ্রাগমেন্টের কনটেন্টের জন্য ব্যবহার করবেন। নিম্নোক্ত উদাহরণ কিছু টেক্সট প্রদর্শন করতে একটি টেক্সট ভিউ ধারণ করে:

```
<com.example.android.animationsdemo.ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/content"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView style="?android:textAppearanceMedium"
        android:padding="16dp"
        android:lineSpacingMultiplier="1.2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/lorem_ipsum" />

</com.example.android.animationsdemo.ScrollView>
```

ফ্রাগমেন্ট তৈরী করুন

একটি ফ্রাগমেন্ট ক্লাস তৈরী করুন যা onCreateView() পদ্ধতির মধ্যে মাত্র তৈরী করা লেআউটটি ফেরত দেয়। তারপর আপনি প্যারেন্ট একটিভিটির মধ্যে এই ফ্রাগমেন্টের ইনসটান্স তৈরী করতে পারেন যখনই ইউজারের কাছে প্রদর্শন করতে আপনার একটি নতুন পেজ দরকার হবে:

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

ভিউ পেজ যুক্ত করা

ViewPager এর পেজ জুরে পরিবর্তন করতে বিল্ট-ইন সুইপ জেশচার আছে, এবং তারা বাই ডিফল্ট স্ক্রিন স্লাইড অ্যানিমেশন প্রদর্শন করে, তাই আপনাকে কোন টা তৈরী করতে হবে না। ViewPager গুলো নতুন পেজ প্রদর্শন করার জন্য একটি যোগান হিসাবে PagerAdapter ব্যবহার করে, সুতরাং PagerAdapter ফ্রাগমেন্ট ক্লাস ব্যবহার করবে যা পূর্বে তৈরী করা হয়েছিল।

শুরু করার জন্য, একটি লেআউট তৈরী করুন একটি ViewPager ধারন করে:

```
<android.support.v4.view.ViewPager
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

একটি একটিভিটি তৈরী করুন যা নিম্নোক্ত বিষয়গুলো করে থাকে:

- ViewPager সহকারে লেআউট হতে কনটেন্ট ভিউ সেট করুন
- একটি ক্লাস তৈরী করুন যা FragmentStatePagerAdapter এবস্ট্রাক্ট ক্লাসকে প্রসারিত করে এবং নতুন পেজ হিসাবে ScreenSlidePageFragment এর ইনসটেন্স যোগান দিতে getItem() পদ্ধতি বাস্তবায়ন করুন। পেজ অ্যাডাপটর আরও চায় যে আপনি getCount() পদ্ধতি বাস্তবায়ন করুন যা অ্যাডাপটর যে পেজ তৈরী করবে তার সংখ্যা কে ফেরত আনে (উদাহরনে ৫ টি)।
- ViewPager এ চধমবৎঅফধঢ়ঃবৎযুক্ত করুন।
- ফ্রাগমেন্টের ভার্চুয়াল স্ট্যাকের মধ্যে ডিভাইসের ব্যাক বাটন পেছনদিকে দেয়ার মাধ্যমে চালিত করুন।

```
public class ScreenSlidePagerActivity extends FragmentActivity {
```

```
    /**
     * The number of pages (wizard steps) to show in this demo.
     */
    private static final int NUM_PAGES = 5;

    /**
     * The pager widget, which handles animation and allows swiping horizontally to access pre
     * and next wizard steps.
     */
    private ViewPager mPager;

    /**
     * The pager adapter, which provides the pages to the view pager widget.
     */
```

```

private PagerAdapter mPagerAdapter;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_screen_slide_pager);

    // Instantiate a ViewPager and a PagerAdapter.
    mPager = (ViewPager) findViewById(R.id.pager);
    mPagerAdapter = new ScreenSlidePagerAdapter(getFragmentManager());
    mPager.setAdapter(mPagerAdapter);
}

@Override
public void onBackPressed() {
    if (mPager.getCurrentItem() == 0) {
        // If the user is currently looking at the first step, allow the system to handle
        // Back button. This calls finish() on this activity and pops the back stack.
        super.onBackPressed();
    } else {
        // Otherwise, select the previous step.
        mPager.setCurrentItem(mPager.getCurrentItem() - 1);
    }
}

/**
 * A simple pager adapter that represents 5 ScreenSlidePageFragment objects, in
 * sequence.
 */
private class ScreenSlidePagerAdapter extends FragmentStatePagerAdapter {
    public ScreenSlidePagerAdapter(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
        return new ScreenSlidePageFragment();
    }

    @Override
    public int getCount() {
        return NUM_PAGES;
    }
}

}

```


পেজ ট্রান্সফরমার দিয়ে অ্যানিমেশন কাস্টমাইজ করা

ডিফল্ট স্ক্রিন স্লাইড অ্যানিমেশন থেকে একটি ভিন্ন অ্যানিমেশন প্রদর্শন করতে, `ViewPager.PageTransformer` ইন্টারফেস বাস্তবায়ন করুন এবং ভিউ পেজারে সাপ্লাই করুন। ইন্টারফেস একটি একক পদ্ধতি `transformPage()` উন্মোচন করে। স্ক্রিনের পরিবর্তনের প্রতিটা পয়েন্টে, এই পদ্ধতি প্রতিটা দৃশ্যমান পেজের জন্য একবারই কল হয় (সাধারণভাবে এখানে একটিই দৃশ্যমান পেজ থাকে) পার্শ্ববর্তী পেজের জন্য শুধু স্ক্রিনটি বন্ধ করে দিন। উদাহরণস্বরূপ, যদি তিন নম্বর পেজ দৃশ্যমান হয় এবং ইউজার চার নম্বর পেজের যায়, ২, ৩ এবং চার নাম্বার পেজের জন্য অঙ্গভঙ্গির/ইংগিত প্রতিটা ধাপে `transformPage()` কল করা হয়।

আপনার `transformPage()` বাস্তবায়নে, স্ক্রিনে পেজের পজিশনের উপর ভিত্তি করে কোন পেজ পরিবর্তন হওয়া প্রয়োজন তা নির্ধারণ করার জন্য আপনি তারপরে কাস্টম স্লাইড এনিমেশন করতে পারেন, যা `transformPage()` পদ্ধতির `position` প্যারামিটার থেকে ধারণ করা হয়।

`Position` প্যারামিটার নির্দেশ করে কোথায় একটি প্রদত্ত পেজ আছে যা স্ক্রিনের কেন্দ্রের সম্পর্কিত। এটা একটা ডাইনামিক প্রপারটি যা ইউজারের স্ক্রিন জুরে স্কল করার মতো করে পরিবর্তন করে। যখন একটি পেজ স্ক্রিন পূর্ণ করে, এটার অবস্থান (পজিশন) ভ্যালু ০। যখন পেজ শুধু স্ক্রিনের ডান পাশ বন্ধ করে অংকন করা হয়, এটার অবস্থান ভ্যালু ১। যদি ইউজার এবং ২ পেজের মধ্যে অর্ধেক পথ স্কল করে, প্রথম পেজের অবস্থান -০.৫ এবং দুই নম্বর পেজের ০.৫ অবস্থান থাকে। স্ক্রিনে পেজের পজিশনের ভিত্তি করে, `setAlpha()`, `setTranslationX()`, বা `setScaleY()` এর মতো পদ্ধতি দিয়ে পেজ প্রাপ্যারটি সেট করার মাধ্যমে আপনি কাস্টম স্লাইড অ্যানিমেশন তৈরী করতে পারেন।

যখন আপনার `PageTransformer` এর একটি বাস্তবায়ন থাকে, আপনার কাস্টম অ্যানিমেশন প্রয়োগ করতে আপনার বাস্তবায়ন দিয়ে `setPageTransformer()` কল করুন। উদাহরণস্বরূপ, আপনার যদি `ZoomOutPageTransformer` নামে একটি `PageTransformer` থাকে, আপনি এটার মতো করে আপনার কাস্টম অ্যানিমেশন করতে পারেন:

```
ViewPager pager = (ViewPager) findViewById(R.id.pager);
...
pager.setPageTransformer(true, new ZoomOutPageTransformer());
```

উদাহরনের জন্য `Zoom-out page transformer` এবং `Depth page transformer` সেকশন দেখুন এবং `PageTransformer` এর ভিডিও দেখুন।

পেজ ট্রান্সফর্মার জুম আউট করা

এই পেজ ট্রান্সফর্মার পেজকে স্ক্রিং এবং স্লান করে যখন পার্শ্ববর্তী পেজের মধ্যে স্ক্রল কার হয়।
যেহেতু একটি পেজ কেন্দ্রের নিকটস্থ হয়, এটা এর স্বাভাবিক সাইজে ফিরে আসে এবং দৃশ্যমান হয়।

ZoomOutPageTransformer উদাহরণ

```
public class ZoomOutPageTransformer implements ViewPager.PageTransformer {
    private static float MIN_SCALE = 0.85f;
    private static float MIN_ALPHA = 0.5f;

    public void transformPage(View view, float position) {
        int pageWidth = view.getWidth();
        int pageHeight = view.getHeight();

        if (position < -1) { // [-Infinity,-1)
            // This page is way off-screen to the left.
            view.setAlpha(0);

        } else if (position <= 1) { // [-1,1]
            // Modify the default slide transition to shrink the page as well
            float scaleFactor = Math.max(MIN_SCALE, 1 - Math.abs(position));
            float vertMargin = pageHeight * (1 - scaleFactor) / 2;
            float horzMargin = pageWidth * (1 - scaleFactor) / 2;
            if (position < 0) {
                view.setTranslationX(horzMargin - vertMargin / 2);
            } else {
                view.setTranslationX(-horzMargin + vertMargin / 2);
            }

            // Scale the page down (between MIN_SCALE and 1)
            view.setScaleX(scaleFactor);
            view.setScaleY(scaleFactor);

            // Fade the page relative to its size.
            view.setAlpha(MIN_ALPHA +
                           (scaleFactor - MIN_SCALE) /
                           (1 - MIN_SCALE) * (1 - MIN_ALPHA));

        } else { // (1,+Infinity]
            // This page is way off-screen to the right.
            view.setAlpha(0);
        }
    }
}
```

ডেপথ পেজ ট্রান্সফরমার

এই পেজ ট্রান্সফরমার বাম পার্শে স্লাইডিং পেজের জন্য ডিফল্ট স্লাইড অ্যানিমেশন ব্যবহার করে, যখন ডান পার্শে স্লাইডিং পেজের জন্য একটি “ডেপথ” অ্যানিমেশন ব্যবহার করা হয়। এই ডেপথ অ্যানিমেশন পেজ আউটকে ফেড করে, এবং সুসঙ্গতভাবে আনুপাতিক হারে কমায়।

DepthPageTransformer উদাহরণ

নোট: ডেপথ অ্যানিমেশনের সময়, ডিফল্ট অ্যানিমেশন (একটি স্ক্রিন স্লাইড) তখনও জায়গা নেয়, সুতরাং আপনাকে অবশ্যই একটি নেগেটিভ X অনুবাদ দিয়ে স্ক্রিন স্লাইড নিবারণ করতে হবে।
উদাহরণ:

```
view.setTranslationX(  
1 * view.getWidth() * position);
```

নিম্নোক্ত উদাহরণ দেখায় একটি কার্যরত পেজ ট্রান্সফরমারে কীভাবে ডিফল্ট স্ক্রিন স্লাইড অ্যানিমেশন নিবারণ করা হয়:

```
public class DepthPageTransformer implements ViewPager.PageTransformer {  
    private static float MIN_SCALE = 0.75f;  
  
    public void transformPage(View view, float position) {  
        int pageWidth = view.getWidth();  
  
        if (position < -1) { // [-Infinity,-1)  
            // This page is way off-screen to the left.  
            view.setAlpha(0);  
  
        } else if (position <= 0) { // [-1,0]  
            // Use the default slide transition when moving to the left page  
            view.setAlpha(1);  
            view.setTranslationX(0);  
            view.setScaleX(1);  
            view.setScaleY(1);  
  
        } else if (position <= 1) { // (0,1]  
            // Fade the page out.  
            view.setAlpha(1 - position);  
  
            // Counteract the default slide transition  
            view.setTranslationX(pageWidth * -position);  
  
            // Scale the page down (between MIN_SCALE and 1)  
            float scaleFactor = MIN_SCALE  
                + (1 - MIN_SCALE) * (1 - Math.abs(position));  
            view.setScaleX(scaleFactor);  
            view.setScaleY(scaleFactor);  
  
        } else { // (1,+Infinity]  
            // This page is way off-screen to the right.  
            view.setAlpha(0);  
        }  
    }  
}
```


কার্ড ফ্লিপ অ্যানিমেশন প্রদর্শন করা

(<http://developer.android.com/training/animation/cardflip.html>)

এই অনুশীলনী আপনাকে শেখাবে কাস্টম ফ্রাগমেন্ট অ্যানিমেশনের সাথে একটি কার্ড ফ্লিপ অ্যানিমেশন কীভাবে করতে হয়। কার্ড ফ্লিপ একটি অ্যানিমেশন দেখানোর মাধ্যমে কনটেন্টের ভিউয়ের মধ্যে অ্যানিমেট করে যা একটি কার্ড ফ্লিপ করার অনুকরণ করে।

এখানে দেয়া আছে একটি ফ্লিপ দেখতে কেমন হবে:

Card flip animation

আপনি যদি সামনে এগিয়ে যেতে চান এবং একটি পূর্ণ কার্যরত উদাহরণ দেখতে চান, নমুনা অ্যাপ [Download](#) এবং রান করুন এবং কার্ড ফ্লিপ উদাহরণ নির্বাচন করুন। কোড বাস্তবায়নের জন্য নিম্নোক্ত ফাইল দেখুন:

- src/CardFlipActivity.java
- animator/card_flip_right_in.xml
- animator/card_flip_right_out.xml
- animator/card_flip_left_in.xml
- animator/card_flip_left_out.xml
- layout/fragment_card_back.xml
- layout/fragment_card_front.xml

অ্যানিমেটর তৈরী করুন

কার্ড ফ্লিপের জন্য অ্যানিমেশন তৈরী করুন। আপনার জন্য দুইটা অ্যানিমেটর লাগবে যখন কার্ডের সম্মুখভাগ অ্যানিমেট আউট করে এবং বাম পাশের দিকে এবং অ্যানিমেট ইন করে এবং বাম পাশ থেকে। আপনার জন্য আরও দুইটা অ্যানিমেটর লাগবে যখন কার্ডের পশ্চাদভাগ অ্যানিমেট ইন করে এবং ডান পাশ থেকে এবং অ্যানিমেট আউট করে ডান পাশের দিকে।

card_flip_left_in.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Before rotating, immediately set the alpha to 0. -->
  <objectAnimator
    android:valueFrom="1.0"
    android:valueTo="0.0"
    android:propertyName="alpha"
    android:duration="0" />

  <!-- Rotate. -->
  <objectAnimator
    android:valueFrom="-180"
    android:valueTo="0"
    android:propertyName="rotationY"
    android:interpolator="@android:interpolator/accelerate_decelerate"
    android:duration="@integer/card_flip_time_full" />

  <!-- Half-way through the rotation (see startOffset), set the alpha to 1. -->
  <objectAnimator
    android:valueFrom="0.0"
    android:valueTo="1.0"
    android:propertyName="alpha"
    android:startOffset="@integer/card_flip_time_half"
    android:duration="1" />
</set>
```

card_flip_left_out.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Rotate. -->
  <objectAnimator
    android:valueFrom="0"
    android:valueTo="180"
    android:propertyName="rotationY"
    android:interpolator="@android:interpolator/accelerate_decelerate"
    android:duration="@integer/card_flip_time_full" />

  <!-- Half-way through the rotation (see startOffset), set the alpha to 0. -->
  <objectAnimator
    android:valueFrom="1.0"
    android:valueTo="0.0"
    android:propertyName="alpha"
    android:startOffset="@integer/card_flip_time_half"
    android:duration="1" />
</set>
```


card_flip_right_in.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Before rotating, immediately set the alpha to 0. -->
  <objectAnimator
    android:valueFrom="1.0"
    android:valueTo="0.0"
    android:propertyName="alpha"
    android:duration="0" />

  <!-- Rotate. -->
  <objectAnimator
    android:valueFrom="180"
    android:valueTo="0"
    android:propertyName="rotationY"
    android:interpolator="@android:interpolator/accelerate_decelerate"
    android:duration="@integer/card_flip_time_full" />

  <!-- Half-way through the rotation (see startOffset), set the alpha to 1. -->
  <objectAnimator
    android:valueFrom="0.0"
    android:valueTo="1.0"
    android:propertyName="alpha"
    android:startOffset="@integer/card_flip_time_half"
    android:duration="1" />
```

card_flip_right_out.xml

```
<set xmlns:android="http://schemas.android.com/apk/res/android">
  <!-- Rotate. -->
  <objectAnimator
    android:valueFrom="0"
    android:valueTo="-180"
    android:propertyName="rotationY"
    android:interpolator="@android:interpolator/accelerate_decelerate"
    android:duration="@integer/card_flip_time_full" />

  <!-- Half-way through the rotation (see startOffset), set the alpha to 0. -->
  <objectAnimator
    android:valueFrom="1.0"
    android:valueTo="0.0"
    android:propertyName="alpha"
    android:startOffset="@integer/card_flip_time_half"
    android:duration="1" />
</set>
```

ভিউ তৈরী করুন

কার্ডেও ট্রিটিটা পার্শ্ব পৃথক লেআউট যা আপনি যা চান সেই কনটেন্ট ধারণ করে, যেমন টেক্সটের দুইটা স্ক্রিন, দুইটা ইমেজ, বা ফ্লিপের মধ্যে ভিউয়ের যে কোন সমাহার। আপনি তখন ফ্রাগমেন্টে দুইটা লেআউট ব্যবহার করতে পারেন যা আপনি পরে অ্যানিমেট করতে পারবেন। নীচের লেআউট একটি কার্ডেও একটি পার্শ্ব তৈরী করে যা টেক্সট দেখায়:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#a6c"
    android:padding="16dp"
    android:gravity="bottom">

    <TextView android:id="@android:id/text1"
        style="?android:textAppearanceLarge"
        android:textStyle="bold"
        android:textColor="#fff"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/card_back_title" />

    <TextView style="?android:textAppearanceSmall"
        android:textAllCaps="true"
        android:textColor="#80ffffff"
        android:textStyle="bold"
        android:lineSpacingMultiplier="1.2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/card_back_description" />

</LinearLayout>
```

এবং কার্ডের অপর পার্শ্ব যা একটি ImageView প্রদর্শন করে:

```
<ImageView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:src="@drawable/image1"
    android:scaleType="centerCrop"
    android:contentDescription="@string/description_image_1" />
```

ফ্রাগমেন্ট তৈরী করা

কার্ডের সম্মুখ এবং পেছন ভাগের জন্য ফ্রাগমেন্ট ক্লাস তৈরী করুন। এই ক্লাসগুলো লেআউট ফেরত দেয় যা আপনি পূর্বে প্রতিটা ফ্রাগমেন্টের onCreateView() পদ্ধতিতে তৈরী করেছিলেন। এরপর আপনি প্যারেন্ট একটিভিটিতে এই ফ্রাগমেন্টের ইনসটেন্স তৈরী করতে পারেন যেখানে আপনি কার্ড প্রদর্শন করতে চান। নীচের উদাহরন প্যারেন্ট একটিভিটির মধ্যে থাকা ফ্রাগমেন্ট ক্লাস দেখায় যা তাদের ব্যবহার করে:

```
public class CardFlipActivity extends Activity {  
    ...  
    /**  
     * A fragment representing the front of the card.  
     */  
    public class CardFrontFragment extends Fragment {  
        @Override  
        public View onCreateView(LayoutInflater inflater, ViewGroup container,  
            Bundle savedInstanceState) {  
            return inflater.inflate(R.layout.fragment_card_front, container, false);  
        }  
    }  
  
    /**  
     * A fragment representing the back of the card.  
     */  
    public class CardBackFragment extends Fragment {  
        @Override  
        public View onCreateView(LayoutInflater inflater, ViewGroup container,  
            Bundle savedInstanceState) {  
            return inflater.inflate(R.layout.fragment_card_back, container, false);  
        }  
    }  
}
```

কার্ড ফ্লিপ অ্যানিমেট করা

এখন আপনার একটি প্যারেন্ট একটিভিটির মধ্যে ফ্রাগমেন্ট প্রদর্শন করার প্রয়োজনে এটা করতে, প্রথমে আপনার একটিভিটির জন্য লেআউট তৈরী করুন। নীচের উদাহরন একটি `FrameLayout` করেছে যাতে আপনি ফ্রাগমেন্ট রানটাইমের সময় এতে এটা সেট করতে পারবেন:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

একটিভিটি কোডে, লেআউট হতে কনটেন্ট ভিউ সেট করুন যা আপনি মাত্র তৈরী করেছেন। একটি ডিফল্ট ফ্রাগমেন্ট দেখানো একটি ভালো চিন্তা যখন একটিভিটি তৈরী হয়, সুতরাং নিম্নোক্ত উদাহরন একটিভিটি আপনাকে দেখায় বাই ডিফল্ট কীভাবে কার্ডের সম্মুখভাগ প্রদর্শন করতে হয়:

```
public class CardFlipActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_activity_card_flip);

        if (savedInstanceState == null) {
            getSupportFragmentManager()
                .beginTransaction()
                .add(R.id.container, new CardFrontFragment())
                .commit();
        }
        ...
    }
}
```

এখন আপনার সম্মুখভাগ প্রদর্শন করার ব্যবস্থা আছে, আপনি একটি যথাযথ সময়ে ফ্লিপ অ্যানিমেশন দিয়ে কার্ডেও পেছনভাগ প্রদর্শন করতে পারেন। কার্ডের অন্য পার্শ্ব প্রদর্শন করতে একটি পদ্ধতি তৈরী করুন যা নিম্নোক্ত বিষয়গুলো সম্পাদন করে:

- কাস্টম অ্যানিমেশন সেট করুন যা ফ্রাগমেন্ট পরিবর্তন করার জন্য আপনি ইতিপূর্বে তৈরী করেছেন।
- একটি নতুন ফ্রাগমেন্টের সাথে বর্তমানে প্রদর্শিত ফ্রাগমেন্টকে প্রতিস্থাপন করুন এবং কাস্টম অ্যানিমেশন যা আপনি তৈরী করেছেন তা দিয়ে এই ইভেন্ট অ্যানিমেট করুন।
- ফ্রাগমেন্ট ব্যাক স্টেজে পূর্বে প্রদর্শিত ফ্রাগমেন্ট যুক্ত করুন তাই যখন ইউজার বাক বাটন প্রেস করবে, কার্ড ফ্লিপ করে টেছনে চলে যাবে।

```
private void flipCard() {
```

```
    if (mShowingBack) {
        getFragmentManager().popBackStack();
        return;
    }

    // Flip to the back.

    mShowingBack = true;

    // Create and commit a new fragment transaction that adds the fragment for the back of
    // the card, uses custom animations, and is part of the fragment manager's back stack.

    getFragmentManager()
        .beginTransaction()

        // Replace the default fragment animations with animator resources representing
        // rotations when switching to the back of the card, as well as animator
        // resources representing rotations when flipping back to the front (e.g. when
        // the system Back button is pressed).
        .setCustomAnimations(
            R.animator.card_flip_right_in, R.animator.card_flip_right_out,
            R.animator.card_flip_left_in, R.animator.card_flip_left_out)

        // Replace any fragments currently in the container view with a fragment
        // representing the next page (indicated by the just-incremented currentPage
        // variable).
        .replace(R.id.container, new CardBackFragment())

        // Add this transaction to the back stack, allowing users to press Back
        // to get to the front of the card.
        .addToBackStack(null)

        // Commit the transaction.
        .commit();

}
```

ভিউ জুম করা

(<http://developer.android.com/training/animation/zoom.html>)

এই অনুশীলনী আপনাকে করে দেখাবে কীভাবে একটি টাচ-টু-জুম অ্যানিমেশন করা যায়, যা অ্যাপসের জন্য প্রয়োজনীয় যেমন একটি পূর্ণ সাইজের ইমেজে একটি থাম্বনেইল থেকে একটি ভিউ অ্যানিমেট করার জন্য ফটো গ্যালারী যা স্ক্রিন পূর্ণ করে।

এখানে একটি টাচ-টু-জুম অ্যানিমেশন দেখতে কেমন হবে তা দেয়া হয়েছে যা স্ক্রিনকে পূর্ণ করতে একটি ইমেজ থাম্বনেইল প্রসারিত করে:

Zoom animation/জুম অ্যানিমেশন

আপনি যদি সামনে এগিয়ে যেতে চান এবং একটি পূর্ণ কার্যরত উদাহরণ দেখতে চান, নমুনা অ্যাপ [Download](#) এবং রান করুন এবং জুম উদাহরণ নির্বাচন করুন। কোড বাস্তবায়নের জন্য নিম্নোক্ত ফাইল দেখুন:

- `src/TouchHighlightImageButton.java` (a simple helper class that shows a blue touch highlight when the image button is pressed)
- `src/ZoomActivity.java`
- `layout/activity_zoom.xml`

ভিউ তৈরী করা

একটি লেআউট ফাইল তৈরী করুন যা কনটেন্টের বড় এবং ছোট সংস্করণ ধারণ করে যা আপনি জুম করতে চান। নীচের উদাহরণটি ক্লিকযোগ্য ইমেজ থাম্বনেইল এর জন্য একটি ImageButton তৈরী করে এবং একটি ImageView যা ইমেজের বড় ভিউ প্রদর্শন করে:

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:padding="16dp">

        <ImageButton
            android:id="@+id/thumb_button_1"
            android:layout_width="100dp"
            android:layout_height="75dp"
            android:layout_marginRight="1dp"
            android:src="@drawable/thumb1"
            android:scaleType="centerCrop"
            android:contentDescription="@string/description_image_1" />

    </LinearLayout>

    <!-- This initially-hidden ImageView will hold the expanded/zoomed version of
        the images above. Without transformations applied, it takes up the entire
        screen. To achieve the "zoom" animation, this view's bounds are animated
        from the bounds of the thumbnail button above, to its final laid-out
        bounds.
    -->

    <ImageView
        android:id="@+id/expanded_image"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="invisible"
        android:contentDescription="@string/description_zoom_touch_close" />

</FrameLayout>
```


জুম অ্যানিমেশন সেটআপ করা

ইতিপূর্বে আপনি যদি আপনার লেআউট প্রয়োগ করে থাকেন, ইভেন্ট হ্যান্ডলারস সেটআপ করুন যা জুম অ্যানিমেশনকে সক্রিয় করে। নীচের উদাহরণটি জুম অ্যানিমেশন সম্পাদন করতে ImageButton এ একটি View.OnClickListener যুক্ত করে যখন ইউজার ইমেজ বাটনে ক্লিক করে:

```
public class ZoomActivity extends FragmentActivity {
    // Hold a reference to the current animator,
    // so that it can be canceled mid-way.
    private Animator mCurrentAnimator;

    // The system "short" animation time duration, in milliseconds. This
    // duration is ideal for subtle animations or animations that occur
    // very frequently.
    private int mShortAnimationDuration;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_zoom);

        // Hook up clicks on the thumbnail views.

        final View thumb1View = findViewById(R.id.thumb_button_1);
        thumb1View.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                zoomImageFromThumb(thumb1View, R.drawable.image1);
            }
        });

        // Retrieve and cache the system's default "short" animation time.
        mShortAnimationDuration = getResources().getInteger(
            android.R.integer.config_shortAnimTime);
    }
    ...
}
```

ভিউ জুম করুন

যখন সুযোগ হবে আপনি তখন সাধারণ সাইজের ভিউ থেকে জুম ভিউয়ে অ্যানিমেট করতে পারেন। সাধারণভাবে আপনাকে সাধারণ সাইজের ভিউয়ের সীমা থেকে বড়-আকারের ভিউ এর দিকে অ্যানিমেট করতে হবে। নীচের পদ্ধতি দেখায় কীভাবে একটি জুম অ্যানিমেশন প্রয়োগ করা যায় যা নীচে কাজগুলো করার মাধ্যমে একটি বড় করা ভিউয়ে একটি ইমেজ থাম্বনেইল জুম করে:

1. হিডেন “জুম-ইন” (বড় করা) ImageView এ অধিক সংখ্যক ইমেজ বরাদ্দ করুন। নীচের উদাহরণ সরলতার জন্য ইউআই থ্রেডে একটি বড় আকারের ইমেজ রিসোর্স লোড করে। ইউআই থ্রেডে ব্লকিং বন্ধ করতে একটি পৃথক থ্রেডে লোড করার মাধ্যমে আপনি এটা করতে চাইতে পারেন এবং তারপর ইউআই থ্রেডে বিটম্যাপ সেট করতে পারেন।
2. ImageView এর জন্য বাউন্ড শুরু এবং শেষকে গণনা করুন।
3. X, Y, (SCALE X, এবং SCALE Y) এ ৪ টি অবস্থানের এবং সাইজের প্রতিটা প্রপারটির অ্যানিমেশন করুন, একইভাবে শুরুর সীমা থেকে শেষের সীমার মধ্যে। এই চার ধরনের অ্যানিমেশন একটি AnimatorSet এ যুক্ত করা হয় যাতে তারা একই সাথে শুরু করতে পারে।
4. একই ধরনের অ্যানিমেশন রান করলে জুম পেছনে চলে যায় কিনুত বিপরীতভাবে ইউজার স্ক্রিন টাচ করে যখন ইমেজ জুম ইন করা হয়। ImageView এ একটি View.OnClickListener যুক্ত করে আপনি এটা করতে পারেন। যখন ক্লিক করা হয়, ImageView ইমেজ থাম্বনেইল সাইজে পিছিয়ে আসাকে মিনিমাইজ কওে এবং এটা তার দৃশ্যমানতাকে GONE এ সেট করে এটাকে হাইড করতে।

```
private void zoomImageFromThumb(final View thumbView, int imageResId) {  
    // If there's an animation in progress, cancel it  
    // immediately and proceed with this one.  
    if (mCurrentAnimator != null) {  
        mCurrentAnimator.cancel();  
    }  
  
    // Load the high-resolution "zoomed-in" image.  
    final ImageView expandedImageView = (ImageView) findViewById(  
        R.id.expanded_image);  
    expandedImageView.setImageResource(imageResId);  
  
    // Calculate the starting and ending bounds for the zoomed-in image.  
    // This step involves lots of math. Yay, math.  
    final Rect startBounds = new Rect();  
    final Rect finalBounds = new Rect();  
    final Point globalOffset = new Point();  
  
    // The start bounds are the global visible rectangle of the thumbnail,  
    // and the final bounds are the global visible rectangle of the container  
    // view. Also set the container view's offset as the origin for the  
    // bounds, since that's the origin for the positioning animation
```

```

// properties (X, Y).
thumbView.getGlobalVisibleRect(startBounds);
findViewById(R.id.container)
    .getGlobalVisibleRect(finalBounds, globalOffset);
startBounds.offset(-globalOffset.x, -globalOffset.y);
finalBounds.offset(-globalOffset.x, -globalOffset.y);

// Adjust the start bounds to be the same aspect ratio as the final
// bounds using the "center crop" technique. This prevents undesirable
// stretching during the animation. Also calculate the start scaling
// factor (the end scaling factor is always 1.0).
float startScale;
if ((float) finalBounds.width() / finalBounds.height()
    > (float) startBounds.width() / startBounds.height()) {
    // Extend start bounds horizontally
    startScale = (float) startBounds.height() / finalBounds.height();
    float startWidth = startScale * finalBounds.width();
    float deltaWidth = (startWidth - startBounds.width()) / 2;
    startBounds.left -= deltaWidth;
    startBounds.right += deltaWidth;
} else {
    // Extend start bounds vertically
    startScale = (float) startBounds.width() / finalBounds.width();
    float startHeight = startScale * finalBounds.height();
    float deltaHeight = (startHeight - startBounds.height()) / 2;
    startBounds.top -= deltaHeight;
    startBounds.bottom += deltaHeight;
}

// Hide the thumbnail and show the zoomed-in view. When the animation
// begins, it will position the zoomed-in view in the place of the
// thumbnail.
thumbView.setAlpha(0f);
expandedImageView.setVisibility(View.VISIBLE);

// Set the pivot point for SCALE_X and SCALE_Y transformations
// to the top-left corner of the zoomed-in view (the default
// is the center of the view).
expandedImageView.setPivotX(0f);
expandedImageView.setPivotY(0f);

// Construct and run the parallel animation of the four translation and
// scale properties (X, Y, SCALE_X, and SCALE_Y).
AnimatorSet set = new AnimatorSet();
set
    .play(ObjectAnimator.ofFloat(expandedImageView, View.X,
        startBounds.left, finalBounds.left))
    .with(ObjectAnimator.ofFloat(expandedImageView, View.Y,
        startBounds.top, finalBounds.top))
    .with(ObjectAnimator.ofFloat(expandedImageView, View.SCALE_X,
        startScale, 1f)).with(ObjectAnimator.ofFloat(expandedImageView,
        View.SCALE_Y, startScale, 1f));
set.setDuration(mShortAnimationDuration);
set.setInterpolator(new DecelerateInterpolator());
set.addListener(new AnimatorListenerAdapter() {
    @Override
    public void onAnimationEnd(Animator animation) {
        mCurrentAnimator = null;
    }

    @Override
    public void onAnimationCancel(Animator animation) {
        mCurrentAnimator = null;
    }
});
set.start();
mCurrentAnimator = set;

// Upon clicking the zoomed-in image, it should zoom back down

```

```

// to the original bounds and show the thumbnail instead of
// the expanded image.
final float startScaleFinal = startScale;
expandedImageView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (mCurrentAnimator != null) {
            mCurrentAnimator.cancel();
        }

        // Animate the four positioning/sizing properties in parallel,
        // back to their original values.
        AnimatorSet set = new AnimatorSet();
        set.play(ObjectAnimator
            .ofFloat(expandedImageView, View.X, startBounds.left))
            .with(ObjectAnimator
                .ofFloat(expandedImageView,
                    View.Y, startBounds.top))
            .with(ObjectAnimator
                .ofFloat(expandedImageView,
                    View.SCALE_X, startScaleFinal))
            .with(ObjectAnimator
                .ofFloat(expandedImageView,
                    View.SCALE_Y, startScaleFinal));
        set.setDuration(mShortAnimationDuration);
        set.setInterpolator(new DecelerateInterpolator());
        set.addListener(new AnimatorListenerAdapter() {
            @Override
            public void onAnimationEnd(Animator animation) {
                thumbView.setAlpha(1f);
                expandedImageView.setVisibility(View.GONE);
                mCurrentAnimator = null;
            }

            @Override
            public void onAnimationCancel(Animator animation) {
                thumbView.setAlpha(1f);
                expandedImageView.setVisibility(View.GONE);
                mCurrentAnimator = null;
            }
        });
        set.start();
        mCurrentAnimator = set;
    }
});
}

```

লেআউট পরিবর্তন অ্যানিমিটে করা

(<http://developer.android.com/training/animation/layout.html>)

একটি লেআউট অ্যানিমেশন হচ্ছে একটি পূর্বে লোড করা অ্যানিমেশন যা লেআউট কনফিগারেশনে কোন পরিবর্তন করার সময় প্রতিবারই সিস্টেম রান করে থাকে। এইসব করতে আপনার যা দরকার হবে তা হচ্ছে লেআউটে একটি এট্রিবিউট সেট করুন অ্যান্ড্রয়েড সিস্টেম কে এই সকল লেআউট পরিবর্তন অ্যানিমিটে করার কথা বলতে এবং সিস্টেম ডিফল্ট অ্যানিমেশন আপনার জন্য সম্পন্ন হয়।

টিপ: আপনি যদি কাস্টম লেআউট অ্যানিমেশন সরবরাহ করতে চান, একটি `LayoutTransition` অবজেক্ট তৈরী করুন এবং `setLayoutTransition()` পদ্ধতি দিয়ে লেআউটে সরবরাহ করুন।

এখানে দেয়া আছে যখন একটি লিস্ট আইটেম যুক্ত হবে তখন ডিফল্ট লেআউট দেকতে কেমন হবে:

Layout animation/ লেআউট অ্যানিমেশন

আপনি যদি সামনে এগিয়ে যেতে চান এবং একটি পূর্ণ কার্যরত উদাহরণ দেখতে চান, নমুনা অ্যাপ [Download](#) এবং রান করুন এবং ক্রসফেড উদাহরণ নির্বাচন করুন। কোড বাস্তবায়নের জন্য নিম্নোক্ত ফাইল দেখুন:

1. `src/LayoutChangesActivity.java`
2. `layout/activity_layout_changes.xml`
3. `menu/activity_layout_changes.xml`

একটি লেআউট তৈরী করা

আপনার একটিভিটির লেআউট XML ফাইলে, লেআউটের জন্য `android:animateLayoutChanges` এট্রিবিউট `true` এ সেট করুন যা আপনি অ্যানিমেশন সক্রিয় করার জন্য চান। উদাহরণস্বরূপ:

```
<LinearLayout android:id="@+id/container"
    android:animateLayoutChanges="true"
    ...
/>
```

লেআউট থেকে আইটেম এ্যাড (যুক্ত), আপডেট বা রিম্যুভ (বিয়োজন) করা

এখন আপনার লেআউটে আইটেম এ্যাড (যুক্ত), আপডেট বা রিম্যুভ (বিয়োজন) করা প্রয়োজন এবং আইটেমগুলো স্বয়ংক্রিয়ভাবে অ্যানিমেট হবে:

```
private ViewGroup mContainerView;  
...  
private void addItem() {  
    View newView;  
    ...  
    mContainerView.addView(newView, 0);  
}
```

কানেকটিভিটি এবং ক্লাউড দিয়ে অ্যাপস তৈরী

(<http://developer.android.com/training/building-connectivity.html>)

এই ক্লাস আপনাকে শেখাবে আপনার অ্যাপকে ইউজারের ডিভাইসের বাইরেও কীভাবে বিশ্বের সাথে সংযুক্ত করা যাবে। আপনি জানবেন কীভাবে এলাকার মধ্যে অন্য ডিভাইসের যুক্ত করতে হয়, ইন্টারনেটের সাথে যুক্ত করতে হয় আপনার ডাটার ব্যাক আপ এবং সিঙ্ক (sync) এবং অন্যান্য কাজ গুলো করতে হয়।

১. ডিভাইসে তারবিহীন সংযোগ

নেটওয়ার্ক সার্ভিস ডিসকোভারি ব্যবহার করে কীভাবে রোকাল ডিভাইস খুজে বের করতে হয় এবং সংযোগ করতে হয় এবং ওয়াই-ফাই দিয়ে পিয়ার টু পিয়ার সংযোগ তৈরী করতে হয়।

1. নেটওয়ার্ক সার্ভিস ডিসকোভারি ব্যবহার
2. ওয়াই-ফাই দিয়ে পিয়ার টু পিয়ার (P2P) সংযোগ তৈরী
3. সার্ভিস ডিসকোভারি জন্য ওয়াই-ফাই পিয়ার টু পিয়ার ব্যবহার

২. নেটওয়ার্ক অপারেশন সম্পাদন

কীভাবে একটি নেটওয়ার্ক সংযোগ তৈরী করতে হয়, কানেকটিভিটিতে পরিবর্তনের জন্য সংযোগ মনিটর করতে হয় এবং XML ডাটার সাথে লেনদেন সম্পন্ন করতে হয়।

1. নেটওয়ার্কে সংযুক্ত করা
2. নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা
3. XML ডাটা পারসিং করা

৩. ব্যাটারী খরচ না করে ডাটা স্থানান্তর করা

ব্যাটারির উপর আপনার অ্যাপের প্রভাব কীভাবে কমিয়ে আনবেন যখন ডাউনলোড বা অন্যান্য নেটওয়ার্ক লেনদেন সংঘটিত হয়ে থাকে।

1. কার্যকরী নেটওয়ার্ক প্রবেশযোগ্যতার জন্য ডাইনলোড অপটিমাইজ করা
2. নিয়মিত আপডেটের প্রভাব কমিয়ে আনা
3. অপ্রয়োজনীয় ডাউনলোড প্রয়োজন নেই
4. কানেকটিভিটির উপর ভিত্তি করে প্যাটার্ন পরিবর্তন করা

৪. ক্লাউডে সিঙ্ক করা

কীভাবে অ্যাপ এবং ইউজারের ডাটা ক্লাউডের দুরবর্তী ওয়েব সার্ভিসে সিঙ্ক এবং ব্যাকআপ করতে

হয় এবং ডাটা কীভাবে বহুবাধ ডিভাইসে রিস্টোর করতে হয়।

1. ব্যাকআপ API ব্যবহার
2. গুগল ক্লাউড মেসেজিং এর অধিকাংশ তৈরী করুন

৫. ক্লাউড সেভ সংঘাত সমাধান করা

অ্যাপের জন্য কীভাবে একটি শক্তিশালী সংঘাত সমাধান কৌশল ডিজাইন করা যায় যা ক্লাউডে ডাটা সেভ করবে।

৬. সিন্ধ অ্যাডাপ্টর ব্যবহার করে ডাটা স্থানান্তর

অ্যান্ড্রয়েড সিন্ধ অ্যাডাপ্টর ফ্রেমওয়ার্ক ব্যবহার করে কীভাবে ক্লাউড এবং ডিভাইসের মধ্যে ডাটা স্থানান্তর করা যায়।

1. স্টাব অথিন্টিকেটর তৈরী করা
2. স্টাব কনটেন্ট প্রদানকরী তৈরী করা
3. সিন্ধ অ্যাডাপ্টর তৈরী করা
4. সিন্ধ অ্যাডাপ্টর রান করা

ডিভাইসে ওয়ারলেস সংযোগ

(<http://developer.android.com/training/connect-devices-wirelessly/index.html>)

ক্লাউড এর সাথে কমিউনিকেশন সক্রিয় করা ছাড়াও, অ্যান্ড্রয়েডের তারবিহীন APIs একই লোকাল নেটওয়ার্কে অন্য ডিভাইসের সাথেও কমিউনিকেশন সক্রিয় করে এবং এমনকি যেটা কোন নেটওয়ার্কে নেই কিনুত আশেপাশে উপস্থিত আছে তার সাথেও। নেটওয়ার্ক সার্ভিস ডিস্কোভারি (এনএসডি) এর সংযোজন একটি অ্যাপলিকেশনকে নিকটবর্তী চলমান ডিভাইস যার সাথে এটা সংযোগ স্থাপন করতে পারে তাকে খুঁজে বের করতে দিয়ে এটাকে সামনে নিয়ে যায়। আপনার অ্যাপলিকেশনের মধ্যে এই কার্যকারিতাকে একিভূত করা আপনাকে ব্যাপক আকারের বৈশিষ্ট্য প্রদান করা, যেমন একই কক্ষে ইউজারের সাথে গেম খেলা, একটি নেটওয়ার্ক করা এনএসডি সক্রিয়- ওয়েবক্যাম থেকে ইমেজ আনা, বা একই নেটওয়ার্কে অন্য মেশিনের মধ্যে লগ ইন করাতে সহায়তা করে।

এই ক্লাস আপনার অ্যাপলিকেশন থেকে অন্য ডিভাইস খুঁজে বের করা বা এর সাথে যুক্ত হওয়ার জন্য যে কি APIs সে বিষয়ে আরোচনা করে। বিশেষ করে, এটা সহজপ্রাপ্য সার্ভিস খুঁজে ডাওয়ার জন্য এনএসডি APIs আলোচনা করে এবং পিয়ার টু পিয়ার (P2P) ওয়াররেস সংযোগ করার জন্য ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) APIs নিয়েও আলোচনা করে। এই ক্লাস আরও আলোচনা করে একটি ডিভাইস কর্তৃক প্রস্তাবিত সার্ভিস চিহ্নিত করতে কীভাবে এনএসডি এবং ওয়াই-ফাই (P2P) একসাথে সন্নিবিষ্টভাবে ব্যবহার করা হয় এবং ডিভাইসের সাথে সংযোগ করতে হয় যখন কোন ডিভাইস নেটওয়ার্কের সাথে যুক্ত থাকে না।

অনুশীলনীসমূহ

নেটওয়ার্ক সার্ভিস ডিসকোভারি ব্যবহার

শিখুন কীভাবে আপনার নিজস্ব অ্যাপ্লিকেশন দ্বারা প্রস্তাব করা সার্ভিস সম্প্রচার করতে হয়, লোকাল নেটওয়ার্কে প্রস্তাবিত সার্ভিস খুঁজে বের করতে হয় এবং আপনি যে সার্ভিসের সাথে যুক্ত হতে চান তার জন্য সংযুক্তির বিস্তারিত বিবরণ নির্ধারণ করতে এনএসডি ব্যবহার করা হয়।

ওয়াই-ফাই দিয়ে পিয়ার টু পিয়ার (P2P) সংযোগ তৈরী

শিখুন কীভাবে একটি নিকটস্থ পিয়ার ডিভাইস থেকে একটি লিস্ট পাওয়া যায়, লিগ্যাসি ডিভাইসের জন্য একটি একসেস পয়েন্ট তৈরি করা হয় এবং ওয়াই-ফাই (P2P) সংযোগে সামর্থ অন্য ডিভাইসে সংযোগ স্থাপন করা হয়।

সার্ভিস ডিসকোভারি জন্য ওয়াই-ফাই পিয়ার টু পিয়ার ব্যবহার

শিখুন কীভাবে ওয়াই-ফাই পিয়ার টু পিয়ার ব্যবহার করে একই নেটওয়ার্ক না হয়েই নিকটস্থ ডিভাইস দ্বারা প্রকাশিত সার্ভিস খুঁজে বের করতে হয়।

নেটওয়ার্ক সার্ভিস ডেলিভারি ব্যবহার

(<http://developer.android.com/training/connect-devices-wirelessly/nsd.html>)

নেটওয়ার্ক সার্ভিস ডেলিভারি (এনএসডি) আপনার অ্যাপে যুক্ত করা আপনার ইউজার কে লোকাল নেটওয়ার্কে অন্য ডিভাইস চিহ্নিত করতে দেয় যা আপনার অ্যাপের করা রিকোয়েস্টকে সাপোর্ট করে। এটা বিভিন্ন পিয়ার টু পিয়ার অ্যাপলিকেশনের জন্য উপকারী যেমন, ফাইল শেয়ার করা বা মাল্টি-প্লেয়ার গেম খেলা। অ্যান্ড্রয়েডের এনএসডি এপিআই আপনাকে এই বৈশিষ্ট্য বাস্তবায়ন করার জন্য প্রয়োজনীয় প্রচেষ্টাকে সহজ করে দেয়।

এই অনুশীলনী আপনাকে দেখায় কীভাবে একটি অ্যাপলিকেশন তৈরী করা হয় যা এর নাম এবং সংযোগ সম্পর্কিত তথ্য লোকাল নেটওয়ার্কে সম্প্রচারিত করতে পারে এবং একই ভাবে অন্য অ্যাপলিকেশন থেকে তথ্যের জন্য অনুসন্ধান করে থাকে। চূড়ান্তভাবে, এই অনুশীলনী দেখায় কীভাবে অন্য ডিভাইসে রান করা একই অ্যাপলিকেশনের সাথে যোগাযোগ করতে হয়।

নেটওয়ার্কে আপনার সার্ভিস রেজিস্ট্রেশন করা

নোট: এই ধাপ ঐচ্ছিক। লোকাল নেটওয়ার্ক জুরে আপনার অ্যাপের সার্ভিস সম্প্রচারিত হওয়া বিষয়ে আপনি তেমন গুরুত্ব না দেন, আপনি পরবর্তী সেকশনে চরে যেতে পারেন, Discover Services on the Network।

লোকাল নেটওয়ার্কে আপনার সার্ভিস রেজিস্টার করতে, প্রথমে একটি NsdServiceInfo অবহেজেক্ট তৈরী করুন। এই অবহেজেক্ট তথ্য প্রদান করে যাতে নেটওয়ার্কের অন্য ডিভাইস ব্যবহার করে যখন তারা ঠিক করে কখন আপনার সার্ভিসের সাথে সংযোগ স্থাপন করতে পারে।

```
public void registerService(int port) {  
    // Create the NsdServiceInfo object, and populate it.  
    NsdServiceInfo serviceInfo = new NsdServiceInfo();  
  
    // The name is subject to change based on conflicts  
    // with other services advertised on the same network.  
    serviceInfo.setServiceName("NsdChat");  
    serviceInfo.setServiceType("_http._tcp.");  
    serviceInfo.setPort(port);  
    ....  
}
```

এই কোড খন্ডচিত্রটি "NsdChat"এ সার্ভিস নাম সেট করে। নামটি নেটওয়ার্কে যে কোন ডিভাইসে দৃশ্যমান হয় যা লোকাল সার্ভিস খুজতে এনএসডি ব্যবহার করে। মনে রাখবেন যে, নেটওয়ার্কে যে কোন সার্ভিসের জন্য নামটি অবশ্যই স্বতন্ত্র হতে হবে এবং অ্যান্ড্রয়েড স্বয়ংক্রিয়ভাবে সংঘাত সমাধান পরিচালনা করে। যদি নেটওয়ার্কে দুইটা ডিভাইসই "NsdChat" অ্যাপলিকেশন ইনস্টল করে থাকে, এদেও মধ্যে একটি স্বয়ংক্রিয়ভাবে তার সার্ভিস নাম পরিবর্তন করে, যেমন হতে পারে "NsdChat (1)"।

দ্বিতীয় প্যারামিটার সার্ভিস টাইপ সেট করে, কোন প্রটোকল এবং ট্রান্সপোর্ট লেয়ার অ্যাপলিকেশন ব্যবহার করে তা নির্ধারণ করে। সিনট্যাক্স হচ্ছে "."। কোড খন্ডাংশটিতে, সার্ভিসটি TCP ধরে চলমান HTTP প্রটোকল ব্যবহার করে। একটি অ্যাপলিকেশন একটি প্রিন্টার সার্ভিস প্রস্তাব করে (উদাহরণস্বরূপ, একটি নেটওয়ার্ক প্রিন্টার) "_ipp._tcp"এ সার্ভিস টাইপ সেট করতে পারে।

নোট: দ্য ইন্টারন্যাশনাল অ্যাসাইনড নাম্বার অথরিটি (IANA) সার্ভিস ডিস্কোভারি প্রটোকল দ্বারা ব্যবহৃত সার্ভিস টাইপের একটি কেন্দ্রীয়, নির্ভরযোগ্য তালিকা ব্যবস্থাপনা করে, যেমন এনএসডি এবং ইডহলডুং। আপনি the IANA list of service names and port numbers (<http://www.iana.org/assignments/service-names-port-numbers/> service-names-port-numbers.xml) থেকে তালিকাটি ডাউনলোড করে নিতে পারেন। আপনি যদি একটি নতুন সার্ভিস টাইপ ব্যবহার করার পরিকল্পনা করে থাকেন, IANA Ports and Service registration form পূর্ণ করার মাধ্যমে আপনার এর মজুদ রাখা উচিত।

যখন আপনার সার্ভিসের জন্য পোর্ট সেটিং করা হয়, এটাকে হার্ডকোডিং করা পরিহার করুন, যেহেতু এটা অন্য অ্যাপলিকেশনের সাথে সংঘাত করে। উদাহরণস্বরূপ, ধরুন যে আপনার

অ্যাপলিকেশন সবসময় পোর্ট ১৩৩৭ ব্যবহার করে যা একই পোর্ট ব্যবহার করা অন্য ইনস্টল করা অ্যাপলিকেশনকে সম্ভাব্য সংঘাতের দিকে ঠেলে দেয়। পরিবর্তে ডিভাইসের অন্যান্য বিদ্যমান পোর্ট ব্যবহার করুন। কারন এই ইনফর্মেশন একটি সার্ভিস সম্প্রচারের মাধ্যমে অন্য অ্যাপে সরবরাহ করা হয়, এখানে কম্পাইল সময়ে অন্য অ্যাপলিকেশন কর্তৃক পরিচিত হওয়ার জন্য আপনার অ্যাপলিকেশনের কোন পোর্ট ব্যবহার করার দরকার নেই। পরিবর্তে অ্যাপলিকেশনটি আপনার সার্ভিস ব্রডকাস্ট থেকে এই তথ্য পেতে পারে, আপনার সার্ভিসে সংযুক্ত হওয়ার ঠিক পূর্বে।

আপনি যদি কোন সকেটের সাথে কাজ করতে থাকেন, এখানে আছে কীভাবে আপনি একটি সকেট যে কোন বিদ্যমান পোর্টে শুরু করতে পারেন শুধুমাত্র এটাকে ০ তে সেটিং করার মাধ্যমে।

```
public void initializeServerSocket() {  
    // Initialize a server socket on the next available port.  
    mServerSocket = new ServerSocket(0);  
  
    // Store the chosen port.  
    mLocalPort = mServerSocket.getLocalPort();  
    ...  
}
```

এখন আপনি NsdServiceInfo অবজেক্ট নির্ধারণ করেছেন, আপনার RegistrationListener ইন্টারফেস বাস্তবায়ন করা প্রয়োজন। এই ইন্টারফেস অ্যান্ড্রয়েড দ্বারা ব্যবহৃত কলব্যাক ধারণ করে, আপনার অ্যাপলিকেশনকে সার্ভিসের রেজিস্ট্রেশন এবং আনরেজিস্ট্রেশনের সাফল্য বা ব্যর্থতার বিষয়ে সতর্ক করতে। public void initializeRegistrationListener() { mRegistrationListener = new NsdManager.RegistrationListener() {

```
@Override  
public void onServiceRegistered(NsdServiceInfo NsdServiceInfo) {  
    // Save the service name. Android may have changed it in order to  
    // resolve a conflict, so update the name you initially requested  
    // with the name Android actually used.  
    mServiceName = NsdServiceInfo.getServiceName();  
}  
  
@Override  
public void onRegistrationFailed(NsdServiceInfo serviceInfo, int errorCode) {  
    // Registration failed! Put debugging code here to determine why.  
}  
  
@Override  
public void onServiceUnregistered(NsdServiceInfo arg0) {  
    // Service has been unregistered. This only happens when you call  
    // NsdManager.unregisterService() and pass in this listener.  
}  
  
@Override  
public void onUnregistrationFailed(NsdServiceInfo serviceInfo, int errorCode) {  
    // Unregistration failed. Put debugging code here to determine why.  
}  
};  
}
```

এখন আপনার সার্ভিস রেজিস্টার করতে সকল অংশ আপনার আছে। registerService() পদ্ধতি কল করুন। উল্লেখ্য যে এই পদ্ধতিটি আসিঙ্ক্রনাস, সুতরাং যে কোন কোড যা সার্ভিস রেজিস্টার হওয়ার পর রান করা প্রয়োজন তা অবশ্যই onServiceRegistered() পদ্ধতির মধ্যে যাওয়া উচিত।

```
public void registerService(int port) {
    NsdServiceInfo serviceInfo = new NsdServiceInfo();
    serviceInfo.setServiceName("NsdChat");
    serviceInfo.setServiceType("_http._tcp.");
    serviceInfo.setPort(port);

    mNsdManager = Context.getSystemService(Context.NSD_SERVICE);

    mNsdManager.registerService(
        serviceInfo, NsdManager.PROTOCOL_DNS_SD, mRegistrationListener);
}
```

নেটওয়ার্কে সার্ভিস উন্মোচন করা

নেটওয়ার্ক জীবনে পরিপূর্ণ, নৃশংস নেটওয়ার্ক প্রিন্টার থেকে শুরু করে নিরীহ নেটওয়ার্ক ওয়েবক্যাম পর্যন্ত, নিষুঠর, নিকটস্থ টিক-ট্যাক-টো প্লেয়ার এর ভয়ংকর যুদ্ধ পর্যন্ত। আপনার অ্যাপলিকেশনকে এই জীবন্ত ইকোসিস্টেমটি দেখতে দেওয়ার উপায় হচ্ছে সার্ভিস ডিসকোভারি (খুঁজে বের করা)। কোন সার্ভিস সহজপ্রাপ্য আছে তা দেখতে আপনার অ্যাপলিকেশনের নেটওয়ার্কে সার্ভিস সম্প্রচার (ব্রডকাস্ট) শোনা প্রয়োজন এবং অ্যাপলিকেশন যেটার সাথে কাজ করতে পারবেসেটা বেছে নেয়া।

সার্ভিস ডিসকোভারি (খুঁজে বের করা), সার্ভিস রেজিস্ট্রেশনের মতো কোন ধাপ নেই: প্রাসঙ্গিক কলব্যাকের সাথে একটি ডিসকোভারি লিসেনার সেট করা, এবং `discoverServices()` এ কর করা একটি একক আসিঙ্ক্রোনাস এপিআই তৈরী করা।

প্রথমে, একটি নামহীন ক্লাস শুরু করুন যা `NsdManager.DiscoveryListener` বাস্তবায়ন করে। নিম্নোক্ত খন্ডাংশটি একটি সহজ উদাহরন দেখাচ্ছে:

```
public void initializeDiscoveryListener() {

    // Instantiate a new DiscoveryListener
    mDiscoveryListener = new NsdManager.DiscoveryListener() {

        // Called as soon as service discovery begins.
        @Override
        public void onDiscoveryStarted(String regType) {
            Log.d(TAG, "Service discovery started");
        }

        @Override
        public void onServiceFound(NsdServiceInfo service) {
            // A service was found! Do something with it.
            Log.d(TAG, "Service discovery success" + service);
            if (!service.getServiceType().equals(SERVICE_TYPE)) {
                // Service type is the string containing the protocol and
                // transport layer for this service.
                Log.d(TAG, "Unknown Service Type: " + service.getServiceType());
            } else if (service.getServiceName().equals(mServiceName)) {
                // The name of the service tells the user what they'd be
                // connecting to. It could be "Bob's Chat App".
                Log.d(TAG, "Same machine: " + mServiceName);
            } else if (service.getServiceName().contains("NsdChat")){
                mNsdManager.resolveService(service, mResolveListener);
            }
        }

        @Override
        public void onServiceLost(NsdServiceInfo service) {
            // When the network service is no longer available.
            // Internal bookkeeping code goes here.
            Log.e(TAG, "service lost" + service);
        }

        @Override
        public void onDiscoveryStopped(String serviceType) {
            Log.i(TAG, "Discovery stopped: " + serviceType);
        }
    }
}
```



```

@Override
public void onStartDiscoveryFailed(String serviceType, int errorCode) {
    Log.e(TAG, "Discovery failed: Error code:" + errorCode);
    mNsdManager.stopServiceDiscovery(this);
}

@Override
public void onStopDiscoveryFailed(String serviceType, int errorCode) {
    Log.e(TAG, "Discovery failed: Error code:" + errorCode);
    mNsdManager.stopServiceDiscovery(this);
}
};
}

```

NSD API আপনার অ্যাপলিকেশনের জ্ঞাতার্থে এই ইন্টারফেসে পদ্ধতিটি ব্যবহার করে যখন ডিসকোভারি শুরু হয়, যখন এটা ব্যর্থ হয় এবং সার্ভিস পাওয়া যায় এবং হারিয়ে যায় (হারিয়ে যাওয়া মানে "এখন আর সহজপ্রাপ্য নয়")। মনে রাখবেন এই খন্ডাংশটি কিছু চেক করে না যখন একটি সার্ভিস পাওয়া যায়।

1. খুজে পাওয়া সার্ভিসের সার্ভিস নাম লোকাল সার্ভিসের সার্ভিস নামের সাথে তুলনা করা হয়, এটা নির্ধারণ করতে যে যদি ডিভাইস শুধু তার নিজস্ব সম্প্রচার (ব্রডকাস্ট) তুলে নেয়।
2. এটা এমন ধরনে সার্ভিস যার সাথে আপনার অ্যাপলিকেশন সংযোগ স্থাপন করতে পারে, তাই সার্ভিস টাইপ চেক করা হয়।
3. সঠিক অ্যাপলিকেশনে সংযোগ যাচাই করতে সার্ভিস নাম চেক করা হয়

সার্ভিস নাম চেক করা সবসময় জরুরী নয় এবং আপনি যদি কোন নির্দিষ্ট অ্যাপলিকেশনের সাথে যুক্ত হতে চান তখনই শুধু এটা প্রসঙ্গিক। উদাহরণস্বরূপ, অ্যাপলিকেশন অন্য ডিভাইসে রান করার সময় এর নিজস্ব ইনসটেন্স এর সাথে যুক্ত হতে চাইবে। কিন্ত, যদি অ্যাপলিকেশন একটি নেটওয়ার্ক কানেকশনের সাথে যুক্ত হতে চায় এটা দেখাই যথেষ্ট যে সার্ভিস টাইপ হচ্ছে "_ipp._tcp"।

লিসেনার সেটআপ করার পর, `discoverServices()` কল করুন, আপনার অ্যাপলিকেশনের যা খোজা উচিত সেই সার্ভিস টাইপের মধ্যে পাস করতে, ডিসকোভারি প্রটোকল ব্যবহার করতে এবং লিসেনার আপনি যা মাত্র তৈরী করেছেন।

```

mNsdManager.discoverServices(
    SERVICE_TYPE, NsdManager.PROTOCOL_DNS_SD, mDiscoveryListener);

```

নেটওয়ার্কের সার্ভিসে যুক্ত করুন

যখন আপনার অ্যাপলিকেশন নেটওয়ার্কে একটি সার্ভিস সংযুক্ত করার জন্য খুঁজে পায়, `resolveService()` পদ্ধতি ব্যবহার করে ঐ সার্ভিসের জন্য কানেকশন তথ্য অবশ্যই প্রথমে নির্ধারণ করা উচিত। এই পদ্ধতির মধ্যে পাস করতে একটি `NsdManager.ResolveListener` বাস্তবায়ন করুন এবং কানেকশন তথ্য ধারণ করা একটি `NsdServiceInfo` পাওয়ার জন্য এটা ব্যবহার করুন।

```
public void initializeResolveListener() {
    mResolveListener = new NsdManager.ResolveListener() {

        @Override
        public void onResolveFailed(NsdServiceInfo serviceInfo, int errorCode) {
            // Called when the resolve fails. Use the error code to debug.
            Log.e(TAG, "Resolve failed" + errorCode);
        }

        @Override
        public void onServiceResolved(NsdServiceInfo serviceInfo) {
            Log.e(TAG, "Resolve Succeeded. " + serviceInfo);

            if (serviceInfo.getServiceName().equals(mServiceName)) {
                Log.d(TAG, "Same IP.");
                return;
            }
            mService = serviceInfo;
            int port = mService.getPort();
            InetAddress host = mService.getHost();
        }
    };
}
```

একসময় সার্ভিস স্থির হবে, আপনার অ্যাপলিকেশন সার্ভিস তথ্য সম্পর্কে বিস্তারিত গ্রহণ করবে যার মধ্যে রয়েছে একটি আইপি এড্রেস এবং পোর্ট নাম্বার। সার্ভিসে আপনার নিজের নেটওয়ার্ক কানেকশন তৈরী করতে আপনাকে এই সব করতে হবে।

অ্যাপলিকেশন বন্ধ করার সময় আপনার সার্ভিস আনরেজিস্টার করুন

NSD এর কার্যকারিতা সক্রিয় এবং নিষ্ক্রিয় করা গুরুত্বপূর্ণ অ্যাপলিকেশনের লাইফসাইকেলের সময়কালে যেভাবে উপযুক্ত হয়। আপনার অ্যাপলিকেশন আনরেজিস্টার করুন যখন এটা বন্ধ হবে, এটা অন্য অ্যাপলিকেশনের চিন্তা করা থেকে বিরত রাখতে সাহায্য করবে যে এখনও এটা সক্রিয় আছে এবং এটার সাথে সংযোগ স্থাপনের চেষ্টা করবে। এছাড়াও সার্ভিস ডিসকভারি একটি ব্যয়বহুল অপারেশন হতে পারে এবং এটা থামানো উচিত হবে যখন প্যারেন্ট একটিভিটিতে পজ দেওয়া হবে এবং সক্রিয় করা হবে যখন একটিভিটি পূণরায় শুরু হবে। আপনার প্রধান একটিভিটির লাইফসাইকের পদ্ধতি ওভাররাইড করুন এবং সার্ভিস ব্রডকাস্ট এবং ডিসকভারি যেভাবে উপযুক্ত হয় স্টপ এবং স্টার্ট করতে কোড প্রবেশ করান।

```
//In your application's Activity

@Override
protected void onPause() {
    if (mNsdHelper != null) {
        mNsdHelper.tearDown();
    }
    super.onPause();
}

@Override
protected void onResume() {
    super.onResume();
    if (mNsdHelper != null) {
        mNsdHelper.registerService(mConnection.getLocalPort());
        mNsdHelper.discoverServices();
    }
}

@Override
protected void onDestroy() {
    mNsdHelper.tearDown();
    mConnection.tearDown();
    super.onDestroy();
}

// NsdHelper's tearDown method
public void tearDown() {
    mNsdManager.unregisterService(mRegistrationListener);
    mNsdManager.stopServiceDiscovery(mDiscoveryListener);
}
```

ওয়াই-ফাই সহকারে P2P কানেকশন তৈরী করা

(<http://developer.android.com/training/connect-devices-wirelessly/wifi-direct.html>)

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) এপিআই একটি নেটওয়ার্ক বা হটস্পটে যুক্ত না হয়েই অ্যাপলিকেশনকে নিকটস্থ ডিভাইসে যুক্ত হতে দেয় (অ্যান্ড্রয়েডের ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ফ্রেমওয়ার্ক Wi-Fi Direct™ সার্টিফিকেশনের অনুবর্তী হয়)। ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) আপনার অ্যাপলিকেশন কে বুলটুথ এর সামর্থের সীমার বাইরের নিকটবর্তী ডিভাইসকে দ্রুত খুঁজে বের করা বা এর সাথে যোগাযোগ করতে অনুমোদন করে।

এই অনুশীলনী আপনাকে দেখায় ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ব্যবহার করে কীভাবে নিকটবর্তী ডিভাইসকে খুঁজে বের করা বা এর সাথে যোগাযোগ করতে হয়।

অ্যাপলিকেশন পারমিশন সেটআপ করা

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ব্যবহার করার জন্য, আপনার মেনিফেস্টে CHANGE_WIFI_STATE, ACCESS_WIFI_STATE এবং INTERNET পারমিশন যুক্ত করুন। ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) এর ইন্টারনেট সংযোগের প্রয়োজন নেই, কিন্ত এটা স্ট্যান্ডার্ড জাভা সকেট ব্যবহার করতে পারে, যা INTERNET পারমিশন চায়। সুতরাং ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ব্যবহার করতে আপনার নিম্নোক্ত পারমিশন প্রয়োজন হবে।

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.nsdchat"
    ...

    <uses-permission
        android:required="true"
        android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission
        android:required="true"
        android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission
        android:required="true"
        android:name="android.permission.INTERNET"/>
    ...
```

একটি ব্রডকাস্ট রিসিভার এবং ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ম্যানেজার সেটআপ করুন

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ব্যবহার করতে, আপনার ব্রডকাস্ট ইনটেন্ট কে শোনার চেষ্টা করা প্রয়োজন যা আপনার অ্যাপলিকেশনকে বলবে যখন নির্দিষ্ট কিছু ইভেন্ট ঘটে থাকবে। আপনার অ্যাপলিকেশনকে, একটি `IntentFilter` শুরু করুন এবং নিচের বিষয়গুলো শুনতে এটা সেট করুন:

`WIFI_P2P_STATE_CHANGED_ACTION`

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) কিনা তা নির্দেশ করে

`WIFI_P2P_PEERS_CHANGED_ACTION`

নির্দেশ করে যে বিদ্যমান পিয়ার লিস্ট পরিবর্তন করেছে

`WIFI_P2P_CONNECTION_CHANGED_ACTION`

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) কানেকটিভিটি পরিবর্তন করেছে তার অবস্থান নির্দেশ করে

`WIFI_P2P_THIS_DEVICE_CHANGED_ACTION`

এই ডিভাইসের কনফিগারেশন ডিটেইলস পরিবর্তন করেছে নির্দেশ করে

```
private final IntentFilter intentFilter = new IntentFilter();
...
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // Indicates a change in the Wi-Fi P2P status.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);

    // Indicates a change in the list of available peers.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);

    // Indicates the state of Wi-Fi P2P connectivity has changed.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);

    // Indicates this device's details have changed.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

    ...
}
```

`onCreate()` পদ্ধতির শেষে, একটি `WifiP2pManager` এর ইনস্টেন্স পাবেন, এবং এর `initialize()` পদ্ধতি কল করুন। এই পদ্ধতি একটি `WifiP2pManager.Channel` অবজেক্ট ফেরত দেয়, যা আপনি

পরবর্তীতে ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ফ্রেমওয়ার্কে আপনার অ্যাপ যুক্ত করার সময় ব্যবহার করতে পারেন।

```
@Override

Channel mChannel;

public void onCreate(Bundle savedInstanceState) {
    ....
    mManager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
    mChannel = mManager.initialize(this, getMainLooper(), null);
}
```

এখন একটি নতুন BroadcastReceiver ক্লাস তৈরী করুন যা সিস্টেমের ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) অবস্থার পরিবর্তন শুনতে ব্যবহার করতে পারবেন। onReceive() পদ্ধতির মধ্যে, উপরে লিস্ট করা প্রতিটা (P2P) অবস্থা পরিবর্তন কে চালনা করতে একটি শর্ত যুক্ত করুন।

```
@Override
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {
        // Determine if Wifi P2P mode is enabled or not, alert
        // the Activity.
        int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, -1);
        if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
            activity.setIsWifiP2pEnabled(true);
        } else {
            activity.setIsWifiP2pEnabled(false);
        }
    } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {

        // The peer list has changed! We should probably do something about
        // that.

    } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {

        // Connection state changed! We should probably do something about
        // that.

    } else if (WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)) {
        DeviceListFragment fragment = (DeviceListFragment) activity.getFragmentManager()
            .findFragmentById(R.id.frag_list);
        fragment.updateThisDevice((WifiP2pDevice) intent.getParcelableExtra(
            WifiP2pManager.EXTRA_WIFI_P2P_DEVICE));
    }
}
```

চূড়ান্তভাবে, ইনটেন্ট ফিল্টার এবং ব্রডকাস্ট রিসিভার রেজিস্টার করতে কোড যুক্ত করুন যখন আপনার প্রধান একটিভিটি সক্রিয় থাকে, এবং তাদের আনরেজিস্টার করুন যখন একটিভিটি পজ অবস্থায় থাকে। এটা করার সবচেয়ে ভালো স্থান হচ্ছে onResume() এবং onPause() পদ্ধতি।

```
/** register the BroadcastReceiver with the intent values to be matched */
@Override
public void onResume() {
    super.onResume();
    receiver = new WifiDirectBroadcastReceiver(mManager, mChannel, this);
}
```

```
        registerReceiver(receiver, intentFilter);  
    }  
  
    @Override  
    public void onPause() {  
        super.onPause();  
        unregisterReceiver(receiver);  
    }
```


পিয়ার ডিসকভারি শুরু করুন

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) সহকারে নিকটস্থ ডিভাইস অনুসন্ধান শুরু করতে, `discoverPeers()` কল করুন। এই পদ্ধতি নিম্নোক্ত আলোচনা গ্রহণ করে:

- যখন আপনি পিয়ার টু পিয়ার ম্যানেজার শুরু করবেন আপনি `WifiP2pManager.Channel` ফিরে পাবেন
- পদ্ধতিগুলো সহকারে `WifiP2pManager.ActionListener` এর একটি বাস্তবায়ন, সিস্টেমটি সফল এবং ব্যর্থ ডিসকভারী আহ্বান করে।

```
mManager.discoverPeers(mChannel, new WifiP2pManager.ActionListener() {
```

```
    @Override
    public void onSuccess() {
        // Code for when the discovery initiation is successful goes here.
        // No services have actually been discovered yet, so this method
        // can often be left blank. Code for peer discovery goes in the
        // onReceive method, detailed below.
    }
```

```
    @Override
    public void onFailure(int reasonCode) {
        // Code for when the discovery initiation fails goes here.
        // Alert the user that something went wrong.
    }
```

```
});
```

মনে রাখবেন যে এটা শুধু পিয়ার ডিসকভারী শুরু করে। `discoverPeers()` পদ্ধতি ডিসকভারী প্রক্রিয়া শুরু করে এবং তারপর তাৎক্ষণিকভাবে ফেরত দেয়। সিস্টেমটি আপনাকে জানায় যদি পিয়ার ডিসকভারী প্রক্রিয়া প্রদত্ত একশন লিসেনারের মধ্যে পদ্ধতি কল করার মাধ্যমে সফলভাবে শুরু হয়। এছাড়াও, ডিসকভারী সক্রিয় থাকে যতক্ষণ না একটি কানেকশন শুরু হয় বা একটি পিয়ার টু পিয়ার গ্রুপ শুরু হয়।

পিয়ার লিস্ট নিয়ে আসা

এখন কোড লিখুন যা পিয়ারের লিস্ট নিয়ে আসে এবং প্রক্রিয়া করে। প্রথমে WifiP2pManager.PeerListListener ইন্টারফেস বাস্তবায়ন করুন, যা পিয়ার সম্পর্কে তথ্য প্রদান করবে যাতে ওয়াই-ফাই পিয়ার টু পিয়ার চিহ্নিত হয়। নিম্নোক্ত খন্ডাংশটি এটা সম্পর্কে বিস্তারিত আলোচনা করে:

```
private List peers = new ArrayList();
...

private PeerListListener peerListListener = new PeerListListener() {
    @Override
    public void onPeersAvailable(WifiP2pDeviceList peerList) {

        // Out with the old, in with the new.
        peers.clear();
        peers.addAll(peerList.getDeviceList());

        // If an AdapterView is backed by this data, notify it
        // of the change. For instance, if you have a ListView of available
        // peers, trigger an update.
        ((WifiPeerListAdapter) getListAdapter()).notifyDataSetChanged();
        if (peers.size() == 0) {
            Log.d(WifiDirectActivity.TAG, "No devices found");
            return;
        }
    }
}
```

এখন requestPeers() কল করতে আপনার ব্রডকাস্ট রিসিভারের onReceive() পদ্ধতি পরিবর্তন করুন যখন একশন WIFI_P2P_PEERS_CHANGED_ACTION সহকারে একটি ইনটেন্ট রিসিভ করা হয়। আপনাকে যেভাবেই হোক এই লিসেনারকে রিসিভারের মধ্যে পাস করে দিন। ব্রডকাস্ট রিসিভারের কনস্ট্রাকটরে একটি আরগুমেন্ট হিসাবে এটা সেভ করার একটি উপায়।

```
public void onReceive(Context context, Intent intent) {
    ...
    else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {

        // Request available peers from the wifi p2p manager. This is an
        // asynchronous call and the calling activity is notified with a
        // callback on PeerListListener.onPeersAvailable()
        if (mManager != null) {
            mManager.requestPeers(mChannel, peerListListener);
        }
        Log.d(WifiDirectActivity.TAG, "P2P peers changed");
    }...
}
```

এখন, একশন WIFI_P2P_PEERS_CHANGED_ACTION ইনটেন্ট সহকারে একটি ইনটেন্ট আপডেট পিয়ার লিস্টের জন্য রিকোয়েস্ট কে চালু করবে।

একটি পিয়ারের সাথে সংযোগ

একটি পিয়ারে সংযোগ স্থাপনের জন্য, একটি নতুন WifiP2pConfig অবজেক্ট তৈরী করুন, এবং যে ডিভাইসে আপনি যুক্ত হতে চান তার প্রতিনিধিত্ব করা WifiP2pDevice এর থেকে এর মধ্যে ডাটা কপি করুন।

```
@Override
public void connect() {
    // Picking the first device found on the network.
    WifiP2pDevice device = peers.get(0);

    WifiP2pConfig config = new WifiP2pConfig();
    config.deviceAddress = device.deviceAddress;
    config.wps.setup = WpsInfo.PBC;

    mManager.connect(mChannel, config, new ActionListener() {

        @Override
        public void onSuccess() {
            // WiFiDirectBroadcastReceiver will notify us. Ignore for now.
        }

        @Override
        public void onFailure(int reason) {
            Toast.makeText(WiFiDirectActivity.this, "Connect failed. Retry.",
                Toast.LENGTH_SHORT).show();
        }

    });
}
```

WifiP2pManager.ActionListener এই খন্ডাংশে বাস্তবায়িত হয় শুধুমাত্র আপনাকে জানায় কখন শুরুটা সফল বা ব্যর্থ। কানেকশন অবস্থার মধ্যে পরিবর্তন শোনার জন্য, WifiP2pManager.ConnectionInfoListener ইন্টারফেস বাস্তবায়ন করুন। এর onConnectionInfoAvailable() কলব্যাক আপনাকে জানাবে কখন কানেকশনের অবস্থা পরিবর্তন করে। একটি একক ডিভাইসে সংযুক্ত হতে যাওয়া মাল্টিপল ডিভাইসের ক্ষেত্রে যেমন, একটি গেম কিনুত তিনজন প্লেয়ার বা একটি চ্যাট অ্যাপ) একটি ডিভাইস "মৎড়ুড় ড়হিবৎ"হিসাবে বিবেচিত হবে।

```
@Override
public void onConnectionInfoAvailable(final WifiP2pInfo info) {

    // InetAddress from WifiP2pInfo struct.
    InetAddress groupOwnerAddress = info.groupOwnerAddress.getHostAddress();

    // After the group negotiation, we can determine the group owner.
    if (info.groupFormed && info.isGroupOwner) {
        // Do whatever tasks are specific to the group owner.
        // One common case is creating a server thread and accepting
        // incoming connections.
    } else if (info.groupFormed) {
        // The other device acts as the client. In this case,
        // you'll want to create a client thread that connects to the group
        // owner.
    }
}
```

```
}  
}
```

এখন ব্রডকাস্ট রিসিভারের `onReceive()` পদ্ধতিতে ফিরে যান এবং সেকশন পরিবর্তন করুন যা একটি `WIFI_P2P_CONNECTION_CHANGED_ACTION` ইনটেন্ট শুনে থাকে। যখন এই ইনটেন্ট রিসিভ করা হয়, `requestConnectionInfo()` কল করুন। এটা একটি সিগন্যাল কল, সুতরাং আপনি যাকে একটি প্যারামিটার হিসাবে প্রদান করেছেন সেই কানেকশন ইনফো লিসেনার দ্বারা ফলাফল রিসিভ করা হয়।

```
...  
    } else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {  
  
        if (mManager == null) {  
            return;  
        }  
  
        NetworkInfo networkInfo = (NetworkInfo) intent  
            .getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);  
  
        if (networkInfo.isConnected()) {  
  
            // We are connected with the other device, request connection  
            // info to find group owner IP  
  
            mManager.requestConnectionInfo(mChannel, connectionListener);  
        }  
        ...  
    }  
}
```

সার্ভিস ডিসকভারির জন্য ওয়াই-ফাই P2P ব্যবহার করা

(<http://developer.android.com/training/connect-devices-wirelessly/nsd-wifi-direct.html>)

এই ক্লাসের পূর্ব অনুশীলনীতে, Using Network Service Discovery, আপনাকে দেখায় কীভাবে সার্ভিস খুঁজে বের করা যায় যা একটি লোকাল নেটওয়ার্কে যুক্ত। যাহোক, ওয়াই-ফাই পিয়ার টু পিয়ার সার্ভিস ডিসকভারী ব্যবহার করা আপনাকে নিকটস্থ ডিসকভারীর সার্ভিস সরাসরি খুঁজে বের করতে দেয়, কোন নেটওয়ার্কে যুক্ত না হয়েই। আপনার ডিভাইসে রান করা সার্ভিসের প্রচারণাও আপনি করতে পারেন। এই সামর্থ্যগুলি আপনাকে অ্যাপসের মধ্যে যোগাযোগ সহায়তা করে, এমনকি যখন কোন লোকাল নেটওয়ার্ক বা হটস্পট না থাকে।

যখন APIs এর এই সেট নেটওয়ার্ক সার্ভিস ডিসকভারী APIs যার রূপরেখা পূর্ববর্তী অনুশীলনীতে দেয়া হয়েছে তারা যদি উদ্দেশ্যগত একই হয়, তাদের কোডে বাস্তবায়ন করা খুব ভিন্ন। এই অনুশীলনী আপনাকে দেখায় ওয়াই-ফাই পিয়ার টু পিয়ার ব্যবহার করে অন্য ডিভাইস থেকে বিদ্যমান সার্ভিস কীভাবে খুঁজতে হয়। এই অনুশীলনী ধারণা করে আপনি পূর্বে থেকেই সম্পর্কে Wi-Fi-P2P API জানেন।

মেনিফেস্ট সেটআপ করুন

ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ব্যবহার করার জন্য আপনার মেনিফেস্টে CHANGE_WIFI_STATE, ACCESS_WIFI_STATE এবং INTERNET যুক্ত করুন। যদিও ওয়াই-ফাই পিয়ার টু পিয়ার (P2P) ইন্টারনেট কানেকশন চায় না, এটা স্ট্যান্ডার্ড জাভা সকেট ব্যবহার করে, এবং অ্যান্ড্রয়েডে এই সব ব্যবহার করা রিকোয়েস্ট করা পারমিশন চায়।

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.nsdchat"
    ...

    <uses-permission
        android:required="true"
        android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission
        android:required="true"
        android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission
        android:required="true"
        android:name="android.permission.INTERNET"/>
    ...
```

লোকাল সার্ভিস যুক্ত করুন

যদি আপনি একটি লোকাল সার্ভিস সরবরাহ করতে থাকেন, আপনার এটা সার্ভিস ডিসকভারির জন্য রেজিস্টার করতে হবে। একসময় আপনার লোকাল সার্ভিস রেজিস্টার হয়ে থাকে, পিয়ার থেকে সার্ভিস ডিসকভারী রিকোয়েস্ট এ ফ্রেমওয়ার্ক স্বয়ংক্রিয়ভাবে রেসপন্স করবে।

একটি লোকাল সার্ভিস তৈরী করতে:

1. একটি WifiP2pServiceInfo অবজেক্ট তৈরী করুন
2. আপনার সার্ভিস সম্পর্কিত তথ্য এটার সাথে প্রচারিত করুন
3. সার্ভিস ডিসকভারির জন্য লোকাল সার্ভিস রেজিস্টার করতে addLocalService() কল করুন

```
private void startRegistration() {
    // Create a string map containing information about your service.
    Map record = new HashMap();
    record.put("listenport", String.valueOf(SERVER_PORT));
    record.put("buddyname", "John Doe" + (int) (Math.random() * 1000));
    record.put("available", "visible");

    // Service information. Pass it an instance name, service type
    // _protocol._transportlayer , and the map containing
    // information other devices will want once they connect to this one.
    WifiP2pDnsSdServiceInfo serviceInfo =
        WifiP2pDnsSdServiceInfo.newInstance("_test", "_presence._tcp", record);

    // Add the local service, sending the service info, network channel,
    // and listener that will be used to indicate success or failure of
    // the request.
    mManager.addLocalService(channel, serviceInfo, new ActionListener() {
        @Override
        public void onSuccess() {
            // Command successful! Code isn't necessarily needed here,
            // Unless you want to update the UI or add logging statements.
        }

        @Override
        public void onFailure(int arg0) {
            // Command failed. Check for P2P_UNSUPPORTED, ERROR, or BUSY
        }
    });
}
```

নিকটস্থ সার্ভিস উদঘাটন (ডিসকভার)

অ্যান্ড্রয়েড সহজপ্রাপ্য সার্ভিস সম্পর্কে আপনার অ্যাপলিকেশনকে অবহিত করতে কলব্যাক পদ্ধতি ব্যবহার করে, সুতরাং এটা করতে প্রথমে আপনাকে ঐগুলো সেটআপ করা। ইনকামিং রেকর্ড শুনতে একটি `WifiP2pManager.DnsSdTxtRecordListener` তৈরী করুন।

অ্যান্ড্রয়েড আপনার অ্যাপলিকেশনে বিদ্যমান সার্ভিসের বিষয়ে জানাতে কলব্যাক মেথড ব্যবহার করে, সুতরাং প্রথম কাজ হচ্ছে ঐ গুলো সেটআপ করা। ইনকামিং রেকর্ড শুনতে একটি `WifiP2pManager.DnsSdTxtRecordListener` তৈরী করুন। এই রেকর্ড ঐচ্ছিকভাবে অন্য ডিভাইস কর্তৃক ব্রডকাস্ট হতে পারে। যখন একজন ভিতরে আসে, ডিভাইস এড্রেস এবং চলতি পদ্ধতির বাইরের একটি ডাটা কাঠামোতে আপনি যে কোন প্রসঙ্গিক তথ্য চান তা কপি করে, তাই আপনি এটাতে পরে প্রবেশ করতে পারবেন। নিম্নোক্ত উদাহরণ মনে করে যে রেকর্ড একটি "buddyname" ফিল্ড ধারণ করে, ইউজারের আইডেনটিটির সাথে থাকে।

```
final HashMap<String, String> buddies = new HashMap<String, String>();
...
private void discoverService() {
    DnsSdTxtRecordListener txtListener = new DnsSdTxtRecordListener() {
        @Override
        /* Callback includes:
         * fullDomain: full domain name: e.g "printer._ipp._tcp.local."
         * record: TXT record data as a map of key/value pairs.
         * device: The device running the advertised service.
         */

        public void onDnsSdTxtRecordAvailable(
            String fullDomain, Map record, WifiP2pDevice device) {
            Log.d(TAG, "DnsSdTxtRecord available -" + record.toString());
            buddies.put(device.deviceAddress, record.get("buddyname"));
        }
    };
    ...
}
```

সার্ভিস ইনফর্মেশন পেতে, একটি `WifiP2pManager.DnsSdServiceResponseListener` তৈরী করুন। এটা সঠিক বিবরণ এবং কানেকশন তথ্য গ্রহণ করে। পূর্ববর্তী কোড অংশটি একটি ডিভাইস এড্রেস এবং বাদি নেম এক করতে একটি Map অবজেক্ট বাস্তবায়ন করে। সার্ভিস রেসপন্স লিসেনার সমগোত্রিয় সার্ভিস তথ্যের সাথে DNS রেকর্ড লিংক করতে এটা ব্যবহার করে। যখনই উভয় লিসেনার বাস্তবায়িত হবে, `setDnsSdResponseListeners()` পদ্ধতি ব্যবহার করে তাদের `WifiP2pManager` এ যুক্ত করুন।

```
private void discoverService() {
    ...

    DnsSdServiceResponseListener servListener = new DnsSdServiceResponseListener() {
        @Override
        public void onDnsSdServiceAvailable(String instanceName, String registrationType,
            WifiP2pDevice resourceType) {
```



```

        // Update the device name with the human-friendly version from
        // the DnsTxtRecord, assuming one arrived.
        resourceType.deviceName = buddies
            .containsKey(resourceType.deviceAddress) ? buddies
            .get(resourceType.deviceAddress) : resourceType.deviceName;

        // Add to the custom adapter defined specifically for showing
        // wifi devices.
        WifiDirectServicesList fragment = (WifiDirectServicesList) getFragmentManager()
            .findFragmentById(R.id.frag_peerlist);
        WifiDevicesAdapter adapter = ((WifiDevicesAdapter) fragment
            .getListAdapter());

        adapter.add(resourceType);
        adapter.notifyDataSetChanged();
        Log.d(TAG, "onBonjourServiceAvailable " + instanceName);
    }
};

mManager.setDnsSdResponseListeners(channel, servListener, txtListener);
...
}

```

এখন একটি সার্ভিস রিকোয়েস্ট তৈরী করুন এবং `addServiceRequest()` কল করুন। এই পদ্ধতিও সফলতা বা ব্যর্থতা রিপোর্ট করতে একটি লিসেনার নিয়ে থাকে।

```

serviceRequest = WifiP2pDnsSdServiceRequest.newInstance();
mManager.addServiceRequest(channel,
    serviceRequest,
    new ActionListener() {
        @Override
        public void onSuccess() {
            // Success!
        }

        @Override
        public void onFailure(int code) {
            // Command failed. Check for P2P_UNSUPPORTED, ERROR, or BUSY
        }
    });

```

চূড়ান্তভাবে, `discoverServices()` এ কল করুন।

```

mManager.discoverServices(channel, new ActionListener() {

    @Override
    public void onSuccess() {
        // Success!
    }

    @Override
    public void onFailure(int code) {
        // Command failed. Check for P2P_UNSUPPORTED, ERROR, or BUSY
        if (code == WifiP2pManager.P2P_UNSUPPORTED) {
            Log.d(TAG, "P2P isn't supported on this device.");
        } else if (...)
            ...
    }
});

```

যদি সব কিছু ভালোভাবে হয়, আপনাকে অভিনন্দন আপান এটা করতে পেরেছেন। আর আপান যদি কোন সমস্যার মুখোমুখি হন, মনে রাখবেন যে আপনি যে আসিঙ্ক্রোনাস কল তৈরী করেছেন তা একটি আরগুমেন্ট হিসাবে একটি WifiP2pManager.ActionListener নেয় এবং এটা আপনাকে কলব্যাক দিয়ে নির্দেশিত সফলতা বা ব্যর্থতা প্রদান করবে। পদ্ধতি দ্বারা প্রদত্ত এরর কোড প্রব্লেম সম্পর্কে আভাস দেয়। এখানে কিছু সম্ভাব্য এরর দেয়া হলো এবং তারা কি বোঝায় তা দেয়া হলো

P2P_UNSUPPORTED

ওয়াই-ফাই পিয়ার টু পিয়ার অ্যাপ রান করা ডিভাইস সাপোর্ট করে না। সিস্টেমটি রিকোয়েস্ট প্রসেস

BUSY

সিস্টেমটি ব্যাস্ত আছে রিকোয়েস্ট প্রসেস করার মতো যথেষ্ট সময় নেই

ERROR

একটি অভ্যন্তরিন এররের কারনে অপারেশন ব্যর্থ হয়েছে।

নেটওয়ার্ক অপারেশন সম্পাদন করা

(<http://developer.android.com/training/basics/network-ops/index.html>)

এই ক্লাস নেটওয়ার্কে সংযুক্ত হওয়ার জন্য যে মৌলিক কাজগুলো রয়েছে সে সম্পর্কে ব্যাখ্যা করা, নেটওয়ার্ক কানেকশন মনিটরিং করে (এর মধ্যে কানেকশন পরিবর্তন রয়েছে) এবং ইউজারকে একটি অ্যাপের নেটওয়ার্ক ব্যবহারের উপর নিয়ন্ত্রণ প্রদান বিষয়ে আলোচনা করে। এটা এক্সএমএল ডাটা কীভাবে পার্স (parse) করা এবং ব্যবহার করা হয়।

এই ক্লাস একটি নমুনা অ্যাপলিকেশন অন্তর্ভুক্ত করেছে যা সাধারণ নেটওয়ার্ক অপারেশন সম্পাদন করার বিষয়টা আলোচনা করে। আপনি নমুনাটি ডাউনলোড করতে পারেন (ডান পাশে দেয়া আছে) এবং এটা আপনার নিজের অ্যাপলিকেশনের জন্য একটি পুনর্ব্যবহারযোগ্য কোডের সোর্স হিসাবে ব্যবহার করতে পারেন।

এই লেসনগুলোর মধ্যে গিয়ে আপনি অ্যাপলিকেশন তৈরী করার জন্য মৌলিক বিল্ডিং ব্লক পাবেন যা কার্যকরীভাবে কন্টেন্ট এবং পার্স ডাটা ডাউনলোড করে, যখন নেটওয়ার্ক ট্রাফিক মিনিমাইজ করা হয়।

অনুশীলনী সমূহ

নেটওয়ার্কে যুক্ত হওয়া

শিখুন কীভাবে নেটওয়ার্কে যুক্ত হতে হয়, একটি ঐ HTTP ক্লায়েন্ট পছন্দ করতে হয়, এবং নেটওয়ার্ক অপারেশন ইউজার ইন্টারফেসের বাইরে কার্য সম্পাদন করতে হয়।

নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা

শিখুন কীভাবে একটি ডিভাইসের নেটওয়ার্ক সংযোগ চেক করতে হয়, নেটওয়ার্ক ব্যবহার নিয়ন্ত্রণ করার জন্য একটি প্রিফারেন্স ইউজার ইন্টারফেস তৈরী করতে হয় এবং সংযোগ পরিবর্তনে রেসপন্স করতে হয়।

এক্সএমএল ডাটা পার্স (parse) করা

শিখুন কীভাবে এক্সএমএল ডাটা কীভাবে পার্স (parse) করা এবং ব্যবহার করা হয়

নেটওয়ার্কে যুক্ত হওয়া

(<http://developer.android.com/training/basics/network-ops/connecting.html>)

এই অনুশীলনী আপনাকে দেখায় একটি সহজ অ্যাপলিকেশন কীভাবে বাস্তবায়ন করতে হয় যা নেটওয়ার্কে যুক্ত করে। এটা কিছু উৎকৃষ্ট চর্চা বিশ্লেষণ করে যা আপনার এমনকি একটি সহজ নেটওয়ার্কে যুক্ত অ্যাপ তৈরী করার সময়ও অনুসরণ করা উচিত।

মনে রাখবেন যে এই অনুশীলনীতে আলোচনা করা নেটওয়ার্ক অপারেশন সম্পাদন করতে, আপনার অ্যাপলিকেশন মেনিফেস্টের অবশ্যই নিম্নোক্ত পারমিশনগুলো অন্তর্ভুক্ত করতে হবে:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

একটি HTTP ক্লায়েন্ট নির্বাচন

অধিকাংশ নেটওয়ার্কে-যুক্ত অ্যান্ড্রয়েড অ্যাপস ডাটা সেন্ড এবং রিসিভ করতে HTTP ব্যবহার করে। অ্যান্ড্রয়েড দুইটা HTTP ক্লায়েন্ট অন্তর্ভুক্ত করে: `HttpURLConnection` এবং `HttpClient` অ্যাপাশে (Apache)। উভয়ই HTTPS, স্ট্রিমিং আপলোড এবং ডাউনলোড, কনফিগারেবল টাইমআউট, IPv6, এবং কানেকশন পুলিং সাপোর্ট করে। আমরা জিঞ্জারব্রেড বা এর উপরের সংস্করণের প্রতি লক্ষ্য করে তৈরী করা অ্যাপলিকেশনের জন্য `HttpURLConnection` সুপারিশ করি। এই এরও আলোচনার জন্য, এই ব্লগপোস্টটি [Android's HTTP Clients](#) দেখুন।

নেটওয়ার্ক কানেকশন চেক করুন

আপনার অ্যাপ নেটওয়ার্কে যুক্ত হওয়ার জন্য চেষ্টা করার পূর্বে, `getActiveNetworkInfo()` এবং `isConnected()` ব্যবহার করে এটার চেক করে দেখা উচিত কোথায় একটি নেটওয়ার্ক কানেকশন সহজপ্রাপ্য আছে। মনে রাখবেন ডিভাইসটা একটি নেটওয়ার্কের বাইরে থাকতে পারে, অথবা ইউজার ওয়াই-ফাই এবং মোবাইল ডাটা এক্সেস দুইটাই নিষ্ক্রিয় করে রাখতে পারে। এই বিষয়ে আরও আলোচনার জন্য Managing Network Usage অনুশীলনীটা দেখুন।

```
public void myClickHandler(View view) {  
    ...  
    ConnectivityManager connMgr = (ConnectivityManager)  
        getSystemService(Context.CONNECTIVITY_SERVICE);  
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();  
    if (networkInfo != null && networkInfo.isConnected()) {  
        // fetch data  
    } else {  
        // display error  
    }  
    ...  
}
```

একটি পৃথক থ্রেডে নেটওয়ার্ক অপারেশন সম্পাদন করুন

নেটওয়ার্ক অপারেশন অঅনুমেয় বিলম্ব সম্পৃক্ত করতে পারে। একটি দুর্বল ইউজার এক্সপেরিয়েন্স হওয়া থেকে বিরত রাখতে, ইউজার ইন্টারফেস থেকে একটি পৃথক থ্রেডে সব সময় নেটওয়ার্ক অপারেশন পারফর্ম করতে হয়। AsyncTask ক্লাস ইউজার ইন্টারফেস থ্রেড থেকে একটি নতুন কাজ বন্দ করার একটি অন্যতম সহজ উপায় প্রদান করে। এ বিষয়ে আরও আলোচনার জন্য এই Multithreading For Performance ব্লগ পোস্টটি দেখুন। (<http://android-developers.blogspot.com/2010/07/multithreading-for-performance.html>)

নিচের চিত্রটিতে, myClickHandler() পদ্ধতি new DownloadWebpageTask().execute(stringUrl) কে আহ্বান করে। DownloadWebpageTask ক্লাস AsyncTask এর একটি সাব-ক্লাস। DownloadWebpageTask নিচের AsyncTask পদ্ধতিগুলো বাস্তবায়ন করে:

- doInBackground() পদ্ধতি downloadUrl() টি সম্পাদন করে। এটা ওয়েব পেজ URL কে একটি প্যারামিটার হিসাবে পাস করে। পদ্ধতি downloadUrl() টি ওয়েব পেজ কন্টেন্ট ধারণ করে এবং প্রক্রিয়াজাত করে। যখন এটা শেষ করে, এটা একটি রেজাল্ট স্ট্রিং ফেরত পাঠায়।
- onPostExecute() ফিরতি স্ট্রিংটি গ্রহণ করে এবং ইউজার ইন্টারফেসে এটা প্রদর্শন করে।

```
public class HttpExampleActivity extends Activity {
    private static final String DEBUG_TAG = "HttpExample";
    private EditText urlText;
    private TextView textView;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        urlText = (EditText) findViewById(R.id.myUrl);
        textView = (TextView) findViewById(R.id.myText);
    }

    // When user clicks button, calls AsyncTask.
    // Before attempting to fetch the URL, makes sure that there is a network connection.
    public void myClickHandler(View view) {
        // Gets the URL from the UI's text field.
        String stringUrl = urlText.getText().toString();
        ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
        if (networkInfo != null && networkInfo.isConnected()) {
            new DownloadWebpageTask().execute(stringUrl);
        } else {
            textView.setText("No network connection available.");
        }
    }
}
```



```

// Uses AsyncTask to create a task away from the main UI thread. This task takes a
// URL string and uses it to create an HttpURLConnection. Once the connection
// has been established, the AsyncTask downloads the contents of the webpage as
// an InputStream. Finally, the InputStream is converted into a string, which is
// displayed in the UI by the AsyncTask's onPostExecute method.
private class DownloadWebpageTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {

        // params comes from the execute() call: params[0] is the url.
        try {
            return downloadUrl(urls[0]);
        } catch (IOException e) {
            return "Unable to retrieve web page. URL may be invalid.";
        }
    }
    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
        textView.setText(result);
    }
    ...
}

```

এই চিত্রটির ইভেন্টের ক্রমটি নিচের বিষয়গুলোর মতো:

1. যখন ইউজার বাটনে ক্লিক করে যা myClickHandler()কে আহ্বান করে, অ্যাপটি AsyncTask এর সাব ক্লাস DownloadWebpageTask তে নির্দিষ্ট URL টি পাস করে।
2. AsyncTask পদ্ধতি doInBackground(),downloadUrl() পদ্ধতিকে কল করে।
3. The downloadUrl() পদ্ধতি একটি প্যারামিটার হিসাবে একটি URL স্ট্রিং গ্রহণ করে এবং এটা একটি URL অবজেক্ট তৈরী করতে ব্যবহার করে।
4. URL অবজেক্ট একটি HttpURLConnection প্রতিষ্ঠিত করতে ব্যবহৃত হয়।
5. যখনই কানেকশন প্রতিষ্ঠিত হবে, HttpURLConnection অবজেক্ট একটি InputStream হিসাবে ওয়েব পেজ কনটেন্ট ধারণ করে।
6. InputStream readIt() পদ্ধতিটিতে পাস হয়, যা স্ট্রিমটিকে একটি স্ট্রিং এ পরিবর্তন করে।
7. AsyncTask এর onPostExecute() পদ্ধতি প্রধান একটিভিটির ইউজার ইন্টারফেসে স্ট্রিংটি প্রদর্শন করে।

ডাটা সংযোগ এবং ডাউনলোড

আপনার থ্রেডে আপনার নেটওয়ার্ক লেনদেন করতে, আপনার ডাটা ডাউনলোড বা একটি এউএস সম্পাদন করতে আপনি `URLConnection` ব্যবহার করতে পারেন। `Connect()` কল করার পর, `getInputStream()` কল করার মাধ্যমে আপনি ডাটাটির একটি `InputStream` পেতে পারেন।

নিম্নোক্ত চিত্রটিতে, `doInBackground()` পদ্ধতিটি পদ্ধতি `downloadUrl()` কল করে। `downloadUrl()` পদ্ধতি প্রদত্ত URL গ্রহণ করে এবং `URLConnection` হয়ে নেটওয়ার্কের সঙ্গে যুক্ত হতে এটা ব্যবহার করে থাকে। যখনই একটি কানেকশন প্রতিষ্ঠিত হয় অ্যাপটি একটি `InputStream` হিসাবে ডাটা পূরণরুদ্ধার করতে পদ্ধতি `getInputStream()` ব্যবহার করে।

```
// Given a URL, establishes an HttpURLConnection and retrieves
// the web page content as a InputStream, which it returns as
// a string.
private String downloadUrl(String myurl) throws IOException {
    InputStream is = null;
    // Only display the first 500 characters of the retrieved
    // web page content.
    int len = 500;

    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        int response = conn.getResponseCode();
        Log.d(DEBUG_TAG, "The response is: " + response);
        is = conn.getInputStream();

        // Convert the InputStream into a string
        String contentAsString = readIt(is, len);
        return contentAsString;

        // Makes sure that the InputStream is closed after the app is
        // finished using it.
    } finally {
        if (is != null) {
            is.close();
        }
    }
}
```

উল্লেখ্য যে, পদ্ধতি `getResponseCode()` কানেকশনের status code ফেরত দেয়। এটা কানেকশন সম্পর্কে অতিরিক্ত তথ্য পেতে এটা একটি কার্যকরী পথ। ২০০ এর স্ট্যাটাস কোড সফলতা নির্দেশ করে।

ইনপুটস্ট্রিম কে একটি স্ট্রিং এ পরিবর্তন (কনভার্ট)

একটি `InputStream` বাইটের (bytes) একটি পঠনযোগ্য সোর্স। যখনই আপনি একটি `InputStream` পান, এটা একটি স্বাভাবিক ঘটনা যে এটাকে একটি বৃহৎ আকারের ডাটা টাইপে ডিকোড বা পরিবর্তন (কনভার্ট) করা হয়। উদাহরণস্বরূপ, আপনি যদি ইমেজ ডাটা ডাউনলোড করতে থাকেন, আপনি সম্ভবত এটার মতো করে ডিকোড বা প্রদর্শন করবেন:

```
InputStream is = null;
...
Bitmap bitmap = BitmapFactory.decodeStream(is);
ImageView imageView = (ImageView) findViewById(R.id.image_view);
imageView.setImageBitmap(bitmap);
```

উপরোক্ত উদাহরণ দেখায়, `InputStream` একটি ওয়েব পেজের টেক্সটকে প্রতিনিধিত্ব করে। এভাবেই উদাহরণটি `InputStream` কে একটি স্ট্রিং এ পরিবর্তন করে যাতে একটিভিটিটি ইউজার ইন্টারফেসের মধ্যে এটা প্রদর্শন করতে পারে।

```
// Reads an InputStream and converts it to a String.
public String readIt(InputStream stream, int len) throws IOException, UnsupportedEncodingException {
    Reader reader = null;
    reader = new InputStreamReader(stream, "UTF-8");
    char[] buffer = new char[len];
    reader.read(buffer);
    return new String(buffer);
}
```

নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা

(<http://developer.android.com/training/basics/network-ops/managing.html>)

এই অনুশীলনী আলোচনা করে কীভাবে অ্যাপলিকেশন লিখতে (রাইট করতে হয়) হয় যার নেটওয়ার্ক রিসোর্স ব্যবহারের একটি চমৎকার নিয়ন্ত্রণ আছে। যদি আপনার অ্যাপলিকেশন প্রচুর পরিমাণে নেটওয়ার্ক অপারেশন সম্পাদন করে থাকে, আপনার ইউজার সেটিং প্রদান করা উচিত যা ইউজারকে আপনার অ্যাপের ডাটা অভ্যাস (হ্যাবিট) কে নিয়ন্ত্রণ করতে দেয়, যেমন, কীভাবে আপনার অ্যাপ ডাটা সিল্ক করে, যখন ওয়াই-ফাইয়ে থাকে শুধুমাত্র তখন আপলোড/ডাউনলোড করতে, রোয়ামিং এর সময় ডাটা ব্যবহার করতে এবং অন্যান্য কাজ করতে। তাদের কাছে এই নিয়ন্ত্রণগুলো সহজপ্রাপ্য করার মাধ্যমে ইউজার ব্যাকগ্রাউন্ড ডাটাতে আপনার অ্যাপের প্রবেশগম্যতা অনেক কম নিষ্ক্রিয় করে, কারন কতটুকু ডাটা আপনার অ্যাপ ব্যবহার করে তারা পরিবর্তে সঠিকভাবে তা নিয়ন্ত্রণ করে।

কীভাবে অ্যাপ রাইট করা যায় যা ডাউনলোড এবং নেটওয়ার্ক কানেকশনের উপরে ব্যাটারির উপরে প্রভাব কমিয়ে আনে সে বিষয়ে সাধারণ গাইডলাইনের জন্য, Optimizing Battery Life এবং Transferring Data Without Draining the Battery দেখুন।

একটি ডিভাইসের নেটওয়ার্ক কানেকশন চেক করা

একটি ডিভাইসের বিভিন্ন ধরনের নেটওয়ার্ক কানেকশন টাইপ থাকে। এই অনুশীলনী হয় ওয়াই-ফাই বা মোবাইল কানেকশনের উপর ফোকাস করে। সম্ভাব্য নেটওয়ার্কের ধরনের পূর্ণ তালিকার জন্য, `ConnectivityManager` দেখুন।

ওয়াই-ফাই সাধারণভাবে গতিশীল। মোবাইল ডাটাও মাঝে মাঝে পরিমাপ হয় যা ব্যয়বহুল হতে পারে। অ্যাপের জন্য একটি সাধারণ কৌশল হচ্ছে শুধু মাত্র বড় ডাটা ধারণ করা যদি একটি ওয়াই-ফাই থেকে থাকে।

নেটওয়ার্ক অপারেশন সম্পাদন করার পূর্বে, নেটওয়ার্ক কানেক্টিভিটির অবস্থা চেক করাটা একটি ভালো চর্চা। অন্যান্য বিষয়ের মধ্যে এটা আপনার অ্যাপকে অসচেতনভাবে ভুল রেডিও ব্যবহার করা থেকে বিরত রাখে। যদি একটি নেটওয়ার্ক কানেকশন না থাকে, আপনার অ্যাপলিকেশনের উচিত সাবলিলভাবে রেসপন্স করা। নেটওয়ার্ক কানেকশন চেক করতে, আপনি সাধারণভাবে নীচের ক্লাসগুলো ব্যবহার করতে পারেন।

- `ConnectivityManager`: নেটওয়ার্ক কানেক্টিভিটির অবস্থান সম্পর্কে অনুসন্ধানের উত্তর দেয়। এটা অ্যাপলিকেশনকে জানায় যখন নেটওয়ার্ক কানেকশন পরিবর্তন করা হয়।
- `NetworkInfo`: একটি প্রদত্ত টাইপের (বর্তমানে হয় মোবাইল বা ওয়াই-ফাই) একটি নেটওয়ার্ক ইন্টারফেসের অবস্থা আলোচনা করে।

এই চিত্রটি ওয়াই-ফাই এবং মোবাইলের জন্য নেটওয়ার্ক কানেক্টিভিটি পরীক্ষা করে। এটা মনস্থির করে এই নেটওয়ার্ক ইন্টারফেস আছে কিনা (এটা হচ্ছে, নেটওয়ার্ক কানেক্টিভিটি সম্ভব কিনা) এবং/বা এটা সংযুক্ত আছে কিনা (এটা হচ্ছে নেটওয়ার্ক কানেক্টিভিটি বিদ্যমান আছে কিনা এবং যদি সকেট প্রতিষ্ঠা করা সম্ভব হয় এবং ডাটা পাস করা):

```
private static final String DEBUG_TAG = "NetworkStatusExample";
...
ConnectivityManager connMgr = (ConnectivityManager)
    getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_WIFI);
boolean isWifiConn = networkInfo.isConnected();
networkInfo = connMgr.getNetworkInfo(ConnectivityManager.TYPE_MOBILE);
boolean isMobileConn = networkInfo.isConnected();
Log.d(DEBUG_TAG, "Wifi connected: " + isWifiConn);
Log.d(DEBUG_TAG, "Mobile connected: " + isMobileConn);
```

উল্লেখ্য যে একটি নেটওয়ার্ক আছে কিনা তার উপর ভিত্তি করে আপনার সিদ্ধান্ত তৈরি করা উচিত নয়। নেটওয়ার্ক অপারেশন সম্পাদন করার পূর্বে আপনার উচিত সবসময় `isConnected()` চেক করা,

যেহেতু isConnected() ফ্ল্যাগ মোবাইল নেটওয়ার্ক, এরারপ্লেন মোড এবং সামিত ব্যাকগ্রাউন্ড ডাটা চালিত করে।

একটি নেটওয়ার্ক ইন্টারফেস আছে কিনা তা চেক করতে নিম্নোক্ত চিত্রটির মতো আরও সংক্ষিপ্ত পথ। পদ্ধতি getActiveNetworkInfo() একটি NetworkInfo ইনসটেন্স প্রথম খুজে পাওয়া সংযুক্ত প্রতিনিধিত্ব করা নেটওয়ার্ক ইন্টারফেস ফেরত দেয়, বা আর যদি কোন ইন্টারফেস খুজে পাওয়া না যায় তাহলে ইঁষষ ফেরত দেয় (বোঝায় যে ইন্টারনেট সংযোগ পাওয়া যাচ্ছে না):

```
public boolean isOnline() {  
    ConnectivityManager connMgr = (ConnectivityManager)  
        getSystemService(Context.CONNECTIVITY_SERVICE);  
    NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();  
    return (networkInfo != null && networkInfo.isConnected());  
}
```

আরও সুন্দর অবস্থা অনুসন্ধান করতে আপনি NetworkInfo.DetailedState ব্যবহার করতে পারেন, কিনুত তার খুব কমই প্রয়োজন ঘটে।

নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা

আপনি একটি প্রিফারেন্স একটিভিটি বাস্তবায়ন করতে পারেন যা আপনার অ্যাপের নেটওয়ার্ক রিসোর্সের উপরে ইউজারকে পরিষ্কার নিয়ন্ত্রণ দিয়ে থাকে। উদাহরণস্বরূপ:

- আপনি ইউজারকে তখনই একটি ভিডিও আপলোড করতে দিবেন যখন ডিভাইস একটি ওয়াই-ফাইয়ের সাথে যুক্ত থাকে।
- কিছু নির্দিষ্ট বৈশিষ্ট যেমন নেটওয়ার্কেও উপস্থিতি/বিদ্যমানতা, সময়ের বিরতী (ইন্টারভাল) এবং অন্য সব আপনি সিলেক্ট করতে পারেন।

একটি অ্যাপ লিখতে যা নেটওয়ার্ক প্রবেশগম্যতা এবং নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা সাপোর্ট করতে আপনার মেনিফেস্টে অবশ্যই সঠিক পারমিশন এবং ইনটেন্ট ফিল্টার থাকতে হবে।

- মেনিফেস্ট নিম্নোক্ত পারমিশনগুলোর আকাঙ্ক্ষা করে:
 - `android.permission.INTERNET`—অ্যাপলিকেশনকে নেটওয়ার্ক সকেট ওপেন করতে দেয়
 - `android.permission.ACCESS_NETWORK_STATE`— অ্যাপলিকেশনকে নেটওয়ার্ক সম্পর্কে তথ্যের মধ্যে প্রবেশ করতে দেয়।
- `ACTION_MANAGE_NETWORK_USAGE` একশনের জন্য আপনি ইনটেন্ট ফিল্টার ডিক্লেয়ার করতে পারেন নির্দেশ করতে যে আপনার অ্যাপলিকেশন একটি একটিভিটি নির্ধারণ করেছে যা ডাটা ব্যবহার নিয়ন্ত্রণ করার প্রস্তাব দেয়। `ACTION_MANAGE_NETWORK_USAGE` একটি নির্দিষ্ট অ্যাপলিকেশনের নেটওয়ার্ক ডাটা ব্যবস্থাপনা করার জন্য সেটিং দেখায়। যখন আপনার অ্যাপের একটি সেটিং একটিভিটি থাকে যা ইউজারকে নেটওয়ার্ক ব্যবহার নিয়ন্ত্রণ করতে দেয়, আপনার উচিত ঐ একটিভিটির জন্য এই ইনটেন্ট ফিল্টার ডিক্লেয়ার করা। নমুনা অ্যাপলিকেশনের মধ্যে এই একশন ক্লাস `SettingsActivity` কর্তৃক চালিত হয়ে থাকে, যা কখন একটি ফিড ডাউনলোড করতে হবে তা ইউজারকে দিয়ে ঠিক করতে একটি প্রিফারেন্স ইউজার ইন্টারফেস প্রদর্শন করে।

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.networkusage"
    ...>

    <uses-sdk android:minSdkVersion="4"
        android:targetSdkVersion="14" />

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

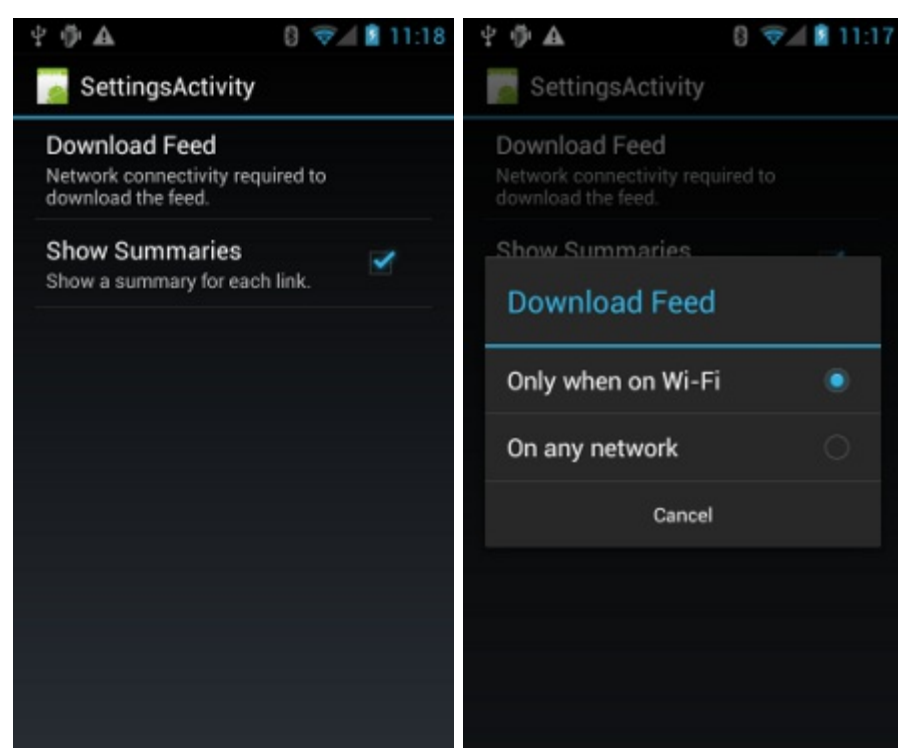
    <application
        ...>
        ...
```

```
        <activity android:label="SettingsActivity" android:name=".SettingsActivity">
            <intent-filter>
                <action android:name="android.intent.action.MANAGE_NETWORK_USAGE" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```


একটি প্রিফারেন্স একটিভিটি বাস্তবায়ন

যেহেতু আপনি উপরে একটি মেনিফেস্টের বর্ণনা দেখতে পারেন, নমুনা অ্যাপের একটিভিটি `SettingsActivity` এর `ACTION_MANAGE_NETWORK_USAGE` একশনের জন্য একটি ইনটেন্ট আছে। `SettingsActivity` হচ্ছে `PreferenceActivity` এর একটি সাব-ক্লাস। এটা একটি প্রিফারেন্স স্ক্রিন প্রদর্শন করে (ফিগার ১ এ দেখানো হয়েছে) যা ইউজারকে নীচের বিষয়গুলো নির্দিষ্ট করতে দেয়:

- প্রতিটা XML ফিড এন্ট্রির জন্য সার সংক্ষেপ প্রদর্শন করতে হবে কিনা অথবা প্রতিটা এন্ট্রির জন্য শুধু একটি লিংক।
- যদি কোন নেটওয়ার্ক কানেকশন থেকে থাকে XML ফিড ডাউনলোড করতে হবে কিনা অথবা যদি শুধুমাত্র ওয়াই-ফাই বিদ্যমান থেকে থাকে।



ফিগার ১. প্রিফারেন্স একটিভিটি এটা হচ্ছে `SettingsActivity` । উল্লেখ্য যে এটা `OnSharedPreferenceChangeListener` বাস্তবায়ন করে। যখন একজন ইউজার একটি প্রিফারেন্স পরিবর্তন করেন, এটা `onSharedPreferenceChanged()` বন্ধ করে দেয়, যেটা `refreshDisplay` সেট করে। এটা রিফ্রেশ করার জন্য প্রদর্শন করার কারন ঘটায় যখন ইউজার প্রধান একটিভিটিতে ফিরে আসবেন:

```
public class SettingsActivity extends PreferenceActivity implements OnSharedPreferenceChangeListener {
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    // Loads the XML preferences file
```

```

        addPreferencesFromResource(R.xml.preferences);
    }

    @Override
    protected void onResume() {
        super.onResume();

        // Registers a listener whenever a key changes
        getPreferenceScreen().getSharedPreferences().registerOnSharedPreferenceChangeListener(this);
    }

    @Override
    protected void onPause() {
        super.onPause();

        // Unregisters the listener set in onResume().
        // It's best practice to unregister listeners when your app isn't using them to cut down on
        // unnecessary system overhead. You do this in onPause().
        getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChangeListener(this);
    }

    // When the user changes the preferences selection,
    // onSharedPreferenceChanged() restarts the main activity as a new
    // task. Sets the refreshDisplay flag to "true" to indicate that
    // the main activity should update its display.
    // The main activity queries the PreferenceManager to get the latest settings.

    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        // Sets refreshDisplay to true so that when the user returns to the main
        // activity, the display refreshes to reflect the new settings.
        NetworkActivity.refreshDisplay = true;
    }
}

```

প্রিফারেন্স পরিবর্তনে রেসপন্স করা

যখন ইউজার সেটিং স্ক্রিনে প্রিফারেন্স পরিবর্তন করে, এটাতে সাধারণভাবে অ্যাপের আচরনের জন্য ফলাফল থাকে। এই চিত্রটিতে অ্যাপ `onStart()` এর মধ্যে প্রিফারেন্স সেটিং চেক করে থাকে। যদি সেখানে সেটিং এবং ডিভাইসের নেটওয়ার্ক কানেকশন এর মধ্যে মিলে যায় (উদাহরণস্বরূপ, যদি সেটিং "Wi-Fi হয় এবং ডিভাইসের একটি Wi-Fi কানেকশন থেকে থাকে), অ্যাপটি ফিড ডাউনলোড করে এবং ডিসপ্লে প্রদর্শন রিফ্রেশ করে।

```
public class NetworkActivity extends Activity {
    public static final String WIFI = "Wi-Fi";
    public static final String ANY = "Any";
    private static final String URL = "http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest";

    // Whether there is a Wi-Fi connection.
    private static boolean wifiConnected = false;
    // Whether there is a mobile connection.
    private static boolean mobileConnected = false;
    // Whether the display should be refreshed.
    public static boolean refreshDisplay = true;

    // The user's current network preference setting.
    public static String sPref = null;

    // The BroadcastReceiver that tracks network connectivity changes.
    private NetworkReceiver receiver = new NetworkReceiver();

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Registers BroadcastReceiver to track network connection changes.
        IntentFilter filter = new IntentFilter(ConnectivityManager.CONNECTIVITY_ACTION);
        receiver = new NetworkReceiver();
        this.registerReceiver(receiver, filter);
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        // Unregisters BroadcastReceiver when app is destroyed.
        if (receiver != null) {
            this.unregisterReceiver(receiver);
        }
    }

    // Refreshes the display if the network connection and the
    // pref settings allow it.

    @Override
    public void onStart () {
        super.onStart();

        // Gets the user's network preference settings
        SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);

        // Retrieves a string value for the preferences. The second parameter
        // is the default value to use if a preference value is not found.
        sPref= sharedPrefs.getString("listPref", "Wi-Fi");
    }
}
```

```

        updateConnectedFlags();

        if(refreshDisplay){
            loadPage();
        }
    }

    // Checks the network connection and sets the wifiConnected and mobileConnected
    // variables accordingly.
    public void updateConnectedFlags() {
        ConnectivityManager connMgr = (ConnectivityManager)
            getSystemService(Context.CONNECTIVITY_SERVICE);

        NetworkInfo activeInfo = connMgr.getActiveNetworkInfo();
        if (activeInfo != null && activeInfo.isConnected()) {
            wifiConnected = activeInfo.getType() == ConnectivityManager.TYPE_WIFI;
            mobileConnected = activeInfo.getType() == ConnectivityManager.TYPE_MOBILE;
        } else {
            wifiConnected = false;
            mobileConnected = false;
        }
    }

    // Uses AsyncTask subclass to download the XML feed from stackoverflow.com.
    public void loadPage() {
        if (((sPref.equals(ANY)) && (wifiConnected || mobileConnected))
            || ((sPref.equals(WIFI)) && (wifiConnected))) {
            // AsyncTask subclass
            new DownloadXmlTask().execute(URL);
        } else {
            showErrorPage();
        }
    }
}

...
}

```

কানেকশন পরিবর্তন চিহ্নিত করা

পায়েজেলর চূড়ান্ত অংশ হচ্ছে BroadcastReceiver সাবক্লাস NetworkReceiver। যখন ডিভাইসের নেটওয়ার্ক কানেকশন পরিবর্তন করে, NetworkReceiver একশন CONNECTIVITY_ACTION কে থামিয়ে দেয়, নির্ধারণ করে দেয় যে নেটওয়ার্ক কানেকশনের স্ট্যাটাসের অবস্থান কি, এবং ফ্ল্যাগস wifiConnected এবং mobileConnected তদানুসারে ট্রু/ফলস এ সেট করে দেয়। পরিনতি হচ্ছে যে পরবর্তী সময়ে ইউজার অ্যাপে ফিরে আসবে, অ্যাপ শুধুমাত্র সর্বশেষ ফিড ডাউনলোড করবে এবং ডিসপ্লে আপডেট করে দেয় যদি NetworkActivity.refreshDisplay ট্রু তে সেট হয়।

একটি ব্রডকাস্ট রিসিভার (BroadcastReceiver) সেটআপ করা যা অপ্রয়োজনীয়ভাবে সিস্টেম রিসোর্সে একটি ড্রেইন কল করা হতে পারে। নমুনা অ্যাপলিকেশন onCreate()এর মধ্যে BroadcastReceiver NetworkReceiver রেজিস্টার করে, এবং onDestroy()এর মধ্যে এটা আনরেজিস্টার করতে পারেন। মেনিফেস্টের মধ্যে একটি <receiver> ডিক্লেয়ার করার চেয়েও এটা আরও অধিক লাইটওয়েট। যখন আপনি মেনিফেস্টে একটি <receiver> ডিক্লেয়ার করেন, এটা যে কোন সময় আপনার অ্যাপকে জাগিয়ে তুরতে পারে, এটাও হতে পারে আপনি এটাকে গত এক সপ্তাহ চালু করেননি। প্রধান একটিভিটির মধ্যে NetworkReceiver রেজিস্টার এবং আন রেজিস্টার করতে, আপনি নিশ্চিত করুন যে ইউজার অ্যাপ ত্যাগ করার পর অ্যাপ টি জেগে উঠবে না। আপনি যদি মেনিফেস্টে একটি <receiver> ডিক্লেয়ার করেন এবং আপনি সঠিকভাবে জানেন যে এটা কোথায় প্রয়োজন, আপনি যথাযথ কাজ হিসাবে এটাকে সক্রিয় এবং নিষ্ক্রিয় করতে setComponentEnabledSetting()ব্যবহার করতে পারেন।

এটা হচ্ছে NetworkReceiver:

```
public class NetworkReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        ConnectivityManager conn = (ConnectivityManager)
            context.getSystemService(Context.CONNECTIVITY_SERVICE);
        NetworkInfo networkInfo = conn.getActiveNetworkInfo();

        // Checks the user prefs and the network connection. Based on the result, decides whether
        // to refresh the display or keep the current display.
        // If the userpref is Wi-Fi only, checks to see if the device has a Wi-Fi connection.
        if (WIFI.equals(sPref) && networkInfo != null && networkInfo.getType() == ConnectivityManager.TYPE_WIFI)
            // If device has its Wi-Fi connection, sets refreshDisplay
            // to true. This causes the display to be refreshed when the user
            // returns to the app.
            refreshDisplay = true;
            Toast.makeText(context, R.string.wifi_connected, Toast.LENGTH_SHORT).show();

        // If the setting is ANY network and there is a network connection
        // (which by process of elimination would be mobile), sets refreshDisplay to true.
    } else if (ANY.equals(sPref) && networkInfo != null) {
        refreshDisplay = true;

        // Otherwise, the app can't download content--either because there is no network
        // connection (mobile or Wi-Fi), or because the pref setting is WIFI, and there
        // is no Wi-Fi connection.
        // Sets refreshDisplay to false.
    } else {
```

```
        refreshDisplay = false;
        Toast.makeText(context, R.string.lost_connection, Toast.LENGTH_SHORT).show();
    }
}
```

XML ডাটা পারসিং করা

(<http://developer.android.com/training/basics/network-ops/xml.html>)

এক্সটেনসিবল মার্কআপ ল্যাঙ্গুয়েজ (XML) হচ্ছে মেশিন-রিডেবল ফর্মে ডকুমেন্ট এনকোডিং করার জন্য এক সেট নিয়মাবলী। XML হচ্ছে ইন্টারনেটে ডাটা শেয়ার করার একটি জনপ্রিয় ফরমেট। ওয়েবসাইট যা প্রতিনিয়ত তাদের কনটেন্ট আপডেট করছে, যেমন কোন নিউজ সাইট বা ব্লগ, মাঝে মাঝে একটি XML ফিড সরবরাহ করে যাতে বাইরের প্রোগ্রামগুলো পাশাপাশি কনটেন্ট পরিবর্তন ধারণ করতে পারে। ডাটা আপলোড করা বা পার্স করা নেটওয়ার্কে যুক্ত হওয়া অ্যাপসের একটি সাধারণ কাজ। এই অনুশীলনী ব্যাখ্যা করে কীভাবে XML ডকুমেন্ট পার্স করতে হয় এবং তাতেও ডাটা ব্যবহার করতে হয়।

একটি পার্সার নির্বাচন করুন

আমরা `XmlPullParser` সুপারিশ করি, যা অ্যান্ড্রয়েডে XML পার্স করার একটি কার্যকরী এবং তত্ত্বাবধানের উপযোগী পথ। ঐতিহাসিকভাবে অ্যান্ড্রয়েডের এই ইন্টারফেসের দুইটা বাস্তবায়ন আছে:

- `KXmlParser` via `XmlPullParserFactory.newPullParser()`.
- `ExpatPullParser`, via `Xml.newPullParser()`.

উভয় নির্বাচনই ঠিক আছে। এই অধ্যায়ের উদাহরণ `Xml.newPullParser()` হয়ে `ExpatPullParser` ব্যবহার করে।

ফিড বিশ্লেষণ

একটি ফিড পার্স করার প্রথম ধাপ হচ্ছে আপনি কোন ফিল্ডে কাজ করতে আগ্রহী তা ঠিক করা। পার্সার ঐ ফিল্ড থেকে ডাটা নিয়ে থাকে এবং বাকীগুলোকে এড়িয়ে যায়।

এখানে ফিড থেকে একটি সংগ্রহ যা একটি নমুনা অ্যাপ থেকে পার্স করা হয়েছে। StackOverflow.com তে প্রতিটা পোস্ট একটি entry হিসাবে ফিডে দৃশ্যমান হয় যা বেশ কিছু বিদ্যমান ট্যাগসকে ধারণ করে:

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom" xmlns:creativeCommons="http://backend.userland.com/creativeCommonsR
<title type="text">newest questions tagged android - Stack Overflow</title>
...
  <entry>
    ...
  </entry>
  <entry>
    <id>http://stackoverflow.com/q/9439999</id>
    <re:rank scheme="http://stackoverflow.com">0</re:rank>
    <title type="text">Where is my data file?</title>
                                <category          scheme="http://stackoverflow.com/feeds/tag?
tagnames=android&sort=newest/tags" term="android"/>
    <category scheme="http://stackoverflow.com/feeds/tag?tagnames=android&sort=newest/tags" term="file"/>
    <author>
      <name>cliff2310</name>
      <uri>http://stackoverflow.com/users/1128925</uri>
    </author>
    <link rel="alternate" href="http://stackoverflow.com/questions/9439999/where-is-my-data-file" />
    <published>2012-02-25T00:30:54Z</published>
    <updated>2012-02-25T00:30:54Z</updated>
    <summary type="html">
      <p>I have an Application that requires a data file...</p>

    </summary>
  </entry>
  <entry>
    ...
  </entry>
...
</feed>
```

নমুনা অ্যাপ entry ট্যাগ থেকে ডাটা সংগ্রহ করে এবং এটা title, link এবং summary ট্যাগে রেখে দেয়

পার্সার ইনসটেনসিয়েট করা

পরবর্তী ধাপ হচ্ছে একটি পার্সার ইনসটেনসিয়েট করা এবং পারসিং প্রক্রিয়াকে ছেড়ে দেয়া। এই চিত্রটিতে, নেমস্পেস প্রক্রিয়া না করার জন্য এবং এর ইনপুট হিসাবে সরবরাহকৃত `InputStream` ব্যবহার করতে একটি পার্সার চালু করা হয়েছে। এটা `nextTag()` এ একটি কল করার মধ্য দিয়ে পারসিং প্রক্রিয়া শুরু করে, এবং `readFeed()` পদ্ধতিকে আহবান করে যা অ্যাপ যে ডাটার প্রতি বাগ্রহী সেই ডাটা নিয়ে আসে এবং প্রক্রিয়া করে:

```
public class StackOverflowXmlParser {
    // We don't use namespaces
    private static final String ns = null;

    public List parse(InputStream in) throws XmlPullParserException, IOException {
        try {
            XmlPullParser parser = Xml.newPullParser();
            parser.setFeature(XmlPullParser.FEATURE_PROCESS_NAMESPACES, false);
            parser.setInput(in, null);
            parser.nextTag();
            return readFeed(parser);
        } finally {
            in.close();
        }
    }
    ...
}
```

ফিড পাঠ করা

`readFeed()` পদ্ধতি ফিড প্রসেস করার সঠিক কাজটি করে। এটা রিকার্সিভভাবে ফিড প্রক্রিয়া করণ করার প্রারম্ভিক পয়েন্ট হিসাবে উপাদান ট্যাগ করা "entry" এর অনুসন্ধান করে। একটি ট্যাগ যদি entry ট্যাগ না হয়, এটা সেটাকে ছেড়ে চলে যায়। যখনই সম্পূর্ণ ফিড রিকার্সিভভাবে প্রক্রিয়া করণ করা হবে, `readFeed()` এন্ট্রি (নেস্টেড ডাটা মেম্বার সহ) ধারণ করা একটি List ফেরত দেয় যেগুলো সে ফিড থেকে গ্রহণ করেছিল। এই তালিকা তারপর ডারসার দ্বারা ফেরত আসে।

```
private List readFeed(XmlPullParser parser) throws XmlPullParserException, IOException {
    List entries = new ArrayList();

    parser.require(XmlPullParser.START_TAG, ns, "feed");
    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        // Starts by looking for the entry tag
        if (name.equals("entry")) {
            entries.add(readEntry(parser));
        } else {
            skip(parser);
        }
    }
    return entries;
}
```

XML পার্স করা

XML পার্স করার ধাপগুলো নিম্নরূপ:

১. যেভাবে Analyze the Feed এ আলোচনা করা হয়েছে, আপনি আপনার অ্যাপে যে ট্যাগ অন্তর্ভুক্ত করতে চান তা চিহ্নিত করুন। এই উদাহরণ `entry` ট্যাগের জন্য ডাটা তুলে আনতে এবং `title`, `link`, এবং `summary` ট্যাগে এগুলো রেখে দেয়।

২. নিম্নোক্ত পদ্ধতিগুলো তৈরী করুন:

- যে ট্যাগগুলোর প্রতি আপনি আগ্রহী তার প্রতিটার জন্য একটি "read" পদ্ধতি উদাহরণস্বরূপ, `readEntry()`, `readTitle()` এবং অন্যান্য। পার্সার ইনপুট স্ট্রিম থেকে ট্যাগ রিড করে। যখন এটা `entry`, `title`, `link` বা `summary` নামে একটি ট্যাগের মুখোমুখি হয়, এটা ঐ ট্যাগের জন্য উপযোগী পদ্ধতি কল করে। অন্যথায় এটা ট্যাগটিকে ছেড়ে দিয়ে চলে যায়।
- প্রতিটা ভিন্ন ধরনের ট্যাগের জন্য ডাটা তুলে আনার জন্য এবং পার্সার কে পরবর্তী ট্যাগে এগিয়ে নিয়ে যাওয়ার জন্য পদ্ধতি তৈরী করা:
 - `Title` এবং `Summary` ট্যাগগুলোর জন্য পার্সার `readText()` কল করে। এই পদ্ধতি `parser.getText()` কল করার মাধ্যমে এই ট্যাগগুলোর জন্য ডাটা তুলে আনে।
 - `link` ট্যাগের জন্য, পার্সার লিংকের জন্য ডাটা তুলে আনে, প্রথমেই এটা নির্ধারণ করে নেয় যে লিংকটা সেই ধরনের যার প্রতি এটার আগ্রহ আছে। তারপর এটা লিংকের ভ্যালু নিয়ে আসতে `parser.getAttributeValue()` ব্যবহার করে।
 - `entry tag` ট্যাগের জন্য, পার্সার `readEntry()` কল করে। এই পদ্ধতি এন্ট্রির নেস্টেড ট্যাগ পার্স করে এবং ডাটা মেম্বার `title`, `link`, এবং `summary` সহ একটি `Entry` অবজেক্ট ফেরত পাঠায়।
- একটি হেল্পার (সহায়তাকারী) `skip()` পদ্ধতি যা রিকার্সিভ। এই বিষয়ে আলোচনার জন্য `Skip Tags You Don't Care About` দেখুন।

এই চিত্রটিতে দেখানো হচ্ছে কীভাবে পার্সার `entries`, `titles`, `links`, এবং `summaries` কে পার্স করে।

```
public static class Entry {  
    public final String title;  
    public final String link;  
    public final String summary;  
  
    private Entry(String title, String summary, String link) {
```

```

        this.title = title;
        this.summary = summary;
        this.link = link;
    }
}

// Parses the contents of an entry. If it encounters a title, summary, or link tag, hands them off
// to their respective "read" methods for processing. Otherwise, skips the tag.
private Entry readEntry(XmlPullParser parser) throws XmlPullParserException, IOException {
    parser.require(XmlPullParser.START_TAG, ns, "entry");
    String title = null;
    String summary = null;
    String link = null;
    while (parser.next() != XmlPullParser.END_TAG) {
        if (parser.getEventType() != XmlPullParser.START_TAG) {
            continue;
        }
        String name = parser.getName();
        if (name.equals("title")) {
            title = readTitle(parser);
        } else if (name.equals("summary")) {
            summary = readSummary(parser);
        } else if (name.equals("link")) {
            link = readLink(parser);
        } else {
            skip(parser);
        }
    }
    return new Entry(title, summary, link);
}

// Processes title tags in the feed.
private String readTitle(XmlPullParser parser) throws IOException, XmlPullParserException {
    parser.require(XmlPullParser.START_TAG, ns, "title");
    String title = readText(parser);
    parser.require(XmlPullParser.END_TAG, ns, "title");
    return title;
}

// Processes link tags in the feed.
private String readLink(XmlPullParser parser) throws IOException, XmlPullParserException {
    String link = "";
    parser.require(XmlPullParser.START_TAG, ns, "link");
    String tag = parser.getName();
    String relType = parser.getAttributeValue(null, "rel");
    if (tag.equals("link")) {
        if (relType.equals("alternate")) {
            link = parser.getAttributeValue(null, "href");
            parser.nextTag();
        }
    }
    parser.require(XmlPullParser.END_TAG, ns, "link");
    return link;
}

// Processes summary tags in the feed.
private String readSummary(XmlPullParser parser) throws IOException, XmlPullParserException {
    parser.require(XmlPullParser.START_TAG, ns, "summary");
    String summary = readText(parser);
    parser.require(XmlPullParser.END_TAG, ns, "summary");
    return summary;
}

// For the tags title and summary, extracts their text values.
private String readText(XmlPullParser parser) throws IOException, XmlPullParserException {
    String result = "";
    if (parser.next() == XmlPullParser.TEXT) {
        result = parser.getText();
        parser.nextTag();
    }
}

```

```
    }  
    return result;  
}  
...  
}
```

আপনার কাছে গুরুত্বপূর্ণ নয় এমন ট্যাগগুলো এড়িয়ে (স্কিপ) যান

XML পার্স করার ধাপ যা উপরে আলোচনা করা হয়েছে তার মধ্যে একটি হচ্ছে পার্সারের যে ট্যাগে আগ্রহ নেই সেগুলো এড়িয়ে যাওয়া। এখানে পার্সারের skip() পদ্ধতি দেয়া আছে:

```
private void skip(XmlPullParser parser) throws XmlPullParserException, IOException {
    if (parser.getEventType() != XmlPullParser.START_TAG) {
        throw new IllegalStateException();
    }
    int depth = 1;
    while (depth != 0) {
        switch (parser.next()) {
            case XmlPullParser.END_TAG:
                depth--;
                break;
            case XmlPullParser.START_TAG:
                depth++;
                break;
        }
    }
}
```

এভাবে এটা কাজ করে:

- এটা একটি ব্যতিক্রমকে ছেড়ে দেয় যদি চলতি ইভেন্ট একটি START_TAG না হয়।
- এটা START_TAG এবং END_TAG এর সাথে ম্যাচ করা সকল ইভেন্ট ব্যবহার করে।
- এটা নিশ্চিত করতে যে এটা সঠিক END_TAG এ বন্ধ করে এবং আসল START_TAG এর পরে প্রথম যে ট্যাগের মুখোমুখি হয় সেটাতে নয়, এটা নেস্টিং ডেপথের ট্র্যাক ধরে রাখে।

তাই যদি চলতি এলিমেন্টের নেস্টেড এলিমেন্ট থাকে, depth এর ভ্যালু ০ হবে না যতক্ষণ না পার্সার আসল START_TAG এবং এর ম্যাচ করা END_TAG এর মধ্যে সকল ইভেন্ট ব্যবহার করে। উদাহরণস্বরূপ, কীভাবে পার্সার < author> এলিমেন্ট ছেড়ে দেয় তা বিবেচনা করা যার ২ নেস্টেড এলিমেন্ট আছে, < name>এবং < uri>:

- while লুপের মধ্য দিয়ে প্রথমবার, < author>এর পর পার্সার যে ট্যাগের মুখোমুখি হয় তা হচ্ছে < name>এর জন্য START_TAG। depth এর জন্য ভ্যালু বেড়ে ২ হবে।
- while লুপের মধ্য দিয়ে দ্বিতীয়বার, পরবর্তী ট্যাগ পার্সার যার মুখোমুখি হয় তা হচ্ছে END_TAG < /name>। depth এর জন্য ভ্যালু কমে ১ হবে।

- while লুপের মধ্য দিয়ে তৃতীয়বার, পরবর্তী ট্যাগ পার্সার যার মুখোমুখি হয় তা হচ্ছে START_TAG < /uri>। depth এর জন্য ভ্যালু বেড়ে ২ হবে।
- while লুপের মধ্য দিয়ে চতুর্থবার, , পরবর্তী ট্যাগ পার্সার যার মুখোমুখি হয় তা হচ্ছে END_TAG < /uri>। depth এর জন্য ভ্যালু কমে ১ হবে।
- while লুপের মধ্য দিয়ে পঞ্চমবার এবং শেষবার, পরবর্তী ট্যাগ পার্সার যার মুখোমুখি হয় তা হচ্ছে END_TAG < /author>। depth এর জন্য ভ্যালু কমে ০ হবে, এটা নির্দেশ করতে যে < author> এলিমেন্টকে সফলার সাথে স্কিপ করা হয়েছে।

XML ডাটা ব্যবহার

নমুনা উদাহরণটি অ্যাপলিকেশন একটি AsyncTaskএর মধ্যে থেকে XML ফিড আনতে যায় এবং পার্স করে। এটা প্রধান ইউজার ইন্টারফেস থ্রেড বন্ধ করার প্রক্রিয়াকে গ্রহণ করে। যখন প্রক্রিয়া শেষ হয় অ্যাপ প্রধান একটিভিটির (NetworkActivity) ইউআইকে আপডেট করে।

নীচের অংশটি দেখায়, loadPage() পদ্ধতি নীচের কাজগুলো করে:

- XML ফিডের জন্য URL সহকারে একটি স্ট্রিং ভেরিয়েবল আরম্ভ করা।
- যদি ইউজারের সেটিং এবং নেটওয়ার্ক কানেকশন এটাকে সমর্থ করে, new DownloadXmlTask().execute(url) আহবান করে। এটা একটি নতুন DownloadXmlTask অবজেক্ট (AsyncTask সাবক্লাস) ইনস্টেনসিয়েট কওে এবং এর execute() পদ্ধতি রান করে, যা ফিড ডাউনলোড এবং পার্স করে এবং ইউআই তে প্রদর্শিত হতে একটি স্ট্রিং রেজাল্ট ফেরত দেয়।

```
public class NetworkActivity extends Activity {
```

```
    public static final String WIFI = "Wi-Fi";
    public static final String ANY = "Any";
    private static final String URL = "http://stackoverflow.com/feeds/tag?
tagNames=android&sort=newest";

    // Whether there is a Wi-Fi connection.
    private static boolean wifiConnected = false;
    // Whether there is a mobile connection.
    private static boolean mobileConnected = false;
    // Whether the display should be refreshed.
    public static boolean refreshDisplay = true;
    public static String sPref = null;

    ...

    // Uses AsyncTask to download the XML feed from stackoverflow.com.
    public void loadPage() {

        if((sPref.equals(ANY)) && (wifiConnected || mobileConnected)) {
            new DownloadXmlTask().execute(URL);
        }
        else if ((sPref.equals(WIFI)) && (wifiConnected)) {
            new DownloadXmlTask().execute(URL);
        }
        else {
            // show error
        }
    }
}
```

AsyncTask সাবক্লাস নীচে দেখানো হলো, DownloadXmlTask নীচের AsyncTask পদ্ধতিগুলো বাস্তবায়ন করে:

- doInBackground() পদ্ধতি loadXmlFromNetwork() বাস্তবায়ন করে। এটা ফিড কে একটি প্যারামিটার হিসাবে পাস করে। পদ্ধতি loadXmlFromNetwork() ফিড নিয়ে আসে এবং প্রক্রিয়া করে। যখন এটা শেষ করে, একটি রেজাল্ট স্ট্রিং ফেরত দেয়।
- onPostExecute() ফেরত আসা স্ট্রিং গ্রহণ করে এবং ইউজার ইন্টারফেসে প্রদর্শন করে।

```
// Implementation of AsyncTask used to download XML feed from stackoverflow.com.
private class DownloadXmlTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        try {
            return loadXmlFromNetwork(urls[0]);
        } catch (IOException e) {
            return getResources().getString(R.string.connection_error);
        } catch (XmlPullParserException e) {
            return getResources().getString(R.string.xml_error);
        }
    }

    @Override
    protected void onPostExecute(String result) {
        setContentView(R.layout.main);
        // Displays the HTML string in the UI via a WebView
        WebView myWebView = (WebView) findViewById(R.id.webview);
        myWebView.loadData(result, "text/html", null);
    }
}
```

নিচেরটা হচ্ছে পদ্ধতি loadXmlFromNetwork() যা DownloadXmlTask থেকে আহবায়িত হয়। এটা নীচের বিষয়গুলো করে থাকে:

1. একটি StackOverflowXmlParser ইনস্টেনসিয়েট করে। এটা এন্ট্রি অবজেক্টের (এন্ট্রিস), এবং title, url এবং summary এর একটি তালিকার (List) জন্য ভেরিয়েবলও তৈরী করে, ঐ ফিল্ডগুলোর জন্য এক্সএমএল ফিড থেকে বের করে নিয়ে আসা ভ্যালু ধারণ করতে।
2. downloadUrl কল করে, যা ফিড নিয়ে আসে এবং একটি InputStream হিসাবে এটা ফেরত দেয়।
3. InputStream কে পার্স করতে StackOverflowXmlParser ব্যবহার করে। StackOverflowXmlParser ফিড থেকে পাওয়া ডাটা সহকারে entries এর একটি List অধ্যুষিত করে।
4. entriesList প্রসেস করে, HTML মার্কআপ দিয়ে ফিড ডাটা একত্রিত করে।
5. একটি HTML স্ট্রিং ফেরত আনে যা AsyncTask পদ্ধতি onPostExecute() দ্বারা প্রধান একটিভিটির মধ্যে প্রদর্শন করে।

```

// Uploads XML from stackoverflow.com, parses it, and combines it with
// HTML markup. Returns HTML string.
private String loadXmlFromNetwork(String urlString) throws XmlPullParserException, IOException {
    InputStream stream = null;
    // Instantiate the parser
    StackOverflowXmlParser stackOverflowXmlParser = new StackOverflowXmlParser();
    List<Entry> entries = null;
    String title = null;
    String url = null;
    String summary = null;
    Calendar rightNow = Calendar.getInstance();
    DateFormat formatter = new SimpleDateFormat("MMM dd h:mmaa");

    // Checks whether the user set the preference to include summary text
    SharedPreferences sharedPrefs = PreferenceManager.getDefaultSharedPreferences(this);
    boolean pref = sharedPrefs.getBoolean("summaryPref", false);

    StringBuilder htmlString = new StringBuilder();
    htmlString.append("<h3>" + getResources().getString(R.string.page_title) + "</h3>");
    htmlString.append("<em>" + getResources().getString(R.string.updated) + " " +
        formatter.format(rightNow.getTime()) + "</em>");

    try {
        stream = downloadUrl(urlString);
        entries = stackOverflowXmlParser.parse(stream);
        // Makes sure that the InputStream is closed after the app is
        // finished using it.
    } finally {
        if (stream != null) {
            stream.close();
        }
    }

    // StackOverflowXmlParser returns a List (called "entries") of Entry objects.
    // Each Entry object represents a single post in the XML feed.
    // This section processes the entries list to combine each entry with HTML markup.
    // Each entry is displayed in the UI as a link that optionally includes
    // a text summary.
    for (Entry entry : entries) {
        htmlString.append("<p><a href='");
        htmlString.append(entry.link);
        htmlString.append("'>" + entry.title + "</a><p>");
        // If the user set the preference to include summary text,
        // adds it to the display.
        if (pref) {
            htmlString.append(entry.summary);
        }
    }
    return htmlString.toString();
}

// Given a string representation of a URL, sets up a connection and gets
// an input stream.
private InputStream downloadUrl(String urlString) throws IOException {
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setReadTimeout(10000 /* milliseconds */);
    conn.setConnectTimeout(15000 /* milliseconds */);
    conn.setRequestMethod("GET");
    conn.setDoInput(true);
    // Starts the query
    conn.connect();
    return conn.getInputStream();
}

```

ব্যাটারি শেষ না করেই ডাটা ট্রান্সফার

(<http://developer.android.com/training/efficient-downloads/index.html>)

এই ক্লাসে আপনি শিখবেন কীভাবে ডাইনলোড এবং নেটওয়ার্ক কানেকশনের ব্যাটারি লাইফ প্রভাব কমিয়ে ফেলা যায়, বিশেষত ওয়ারলেস রেডিও এর সাথে সম্পর্কিত। এই ক্লাস ক্যাশিং, পলিং এবং প্রিফেচিং এর মতো কৌশল ব্যবহার করে ডাউনলোড এর পরিকল্পনা এবং সম্পাদন করার সর্বোত্তম চর্চাও বিষয়টি দেখায়। আপনি শিখবেন ব্যাটারি লাইফের উপর প্রভাব কমিয়ে আনার জন্য কখন, কি এবং কীভাবে ডাটা ট্রান্সফার করতে হবে তাকে ওয়ারলেস রেডিওর পাওয়ার-ব্যবহার প্রোফাইল কীভাবে প্রভাবিত করে।

অনুশীলনীসমূহ

কার্যকরী নেটওয়ার্কে প্রবেশ করার জন্য ডাউনলোড অপটিমাইজ করা

এই অনুশীলনী ওয়ারলেস রেডিও স্টেট মেশিনকে পরিচিত করে, ব্যাখ্যা করে কীভাবে আপনার অ্যাপের কানেকটিভিটি মডেল এটার সাথে পারস্পরিক ক্রিয়া করে থাকে, এবং কীভাবে আপনি আপনার ডাটা কানেকশন কমিয়ে আনতে পারেন এবং আপনার ডাটা ট্রান্সফার করার সাথে সম্পর্কিত ব্যাটারি নিঃশেষ করা কমিয়ে আনতে প্রিফেটিং এবং বান্ডলিং ব্যবহার করা।

নিয়মিত আপডেটের প্রভাব কমিয়ে আনা

প্রাথমিক ওয়ারলেস রেডিও স্টেট মেশিনে ব্যাকগ্রাউন্ড আপডেট সর্বোত্তমভাবে উপশম করতে কীভাবে আপনার রিফ্রেশ ফ্রিকোয়েন্সি ভিন্ন হতে পাওে তা এই অনুশীলনী পরীক্ষা করবে।

অনাবশ্যক (রেডুন্ট) ডাউনলোড হচ্ছে অপ্ৰয়োজনীয়

আপনার ডাউনলোড কমানোর সবচেয়ে মৌলিক উপায় হচ্ছে আপনার যেটা প্রয়োজন শুধুমাত্র সেটাই ডাউনলোড করা। এই অনুশীলনী অনাবশ্যক ডাউনলোড পরিহার করার কিছু উৎকৃষ্ট চর্চার সাথে পরিচিত করিয়ে দেয়।

কানেকটিভিটি টাইপের উপর ভিত্তি করে আপনার ডাউনলোড ধরণ পরিবর্তন করুন

যখন এটা ব্যাটারির লাইফের উপরে প্রভাব হিসাবে আসে, কানেকশনের সকল ধরন সমানভাবে তৈরী হয় না। ওয়াইফাই রেডিও শুধুমাত্র সুনির্দিষ্টভাবে এর অনুরূপ ওয়ারলেস রেডিও এর চেয়ে কম ব্যাটারি ব্যবহারই করে না বরং বিভিন্ন ওয়ারলেস রেডিও প্রযুক্তির মধ্যে ব্যবহার করা রেডিওর বিভিন্ন ব্যাটারি ইমিপ্লিকেশন আছে।

কার্যকরী নেটওয়ার্কে প্রবেশ করার জন্য ডাউনলোড অপটিমাইজ করা

(<http://developer.android.com/training/efficient-downloads/efficient-network-access.html>)

ডাটা ট্রান্সফার করতে ওয়ারলেস ব্যবহার করা হচ্ছে আপনার অ্যাপের ব্যাটারী নিঃশেষ করার সবচেয়ে সুনির্দিষ্ট উৎসের মধ্যে সম্ভাবনাময় কোন একটি। নেটওয়ার্ক কানেকটিভিটির সাথে সম্পর্কিত ব্যাটারি ব্যবহার মিনিমাইজ করতে, আপনার পক্ষে এটা বোঝা মুশকিল যে কীভাবে আপনার কানেকটিভিটি মডেল মৌলিক রেডিও হার্ডওয়ারকে প্রভাবিত করে।

এই অনুশীলনী ওয়ারলেস রেডিও স্টেট মেশিনকে পরিচিত করে এবং ব্যাখ্যা করে কীভাবে আপনার অ্যাপের কানেকটিভিটি মডেল এটার সাথে পারস্পরিক ক্রিয়া করে থাকে। এই অধ্যায় দেখায় কীভাবে আপনি আপনার ডাটা কানেকশন কমিয়ে আনতে পারেন এবং আপনার ডাটা ট্রান্সফার করার সাথে সম্পর্কিত ব্যাটারি নিঃশেষ করা কমিয়ে আনতে প্রিফেচিং এবং বাল্ডলিং ব্যবহার করা।

রেডিও স্টেট মেশিন

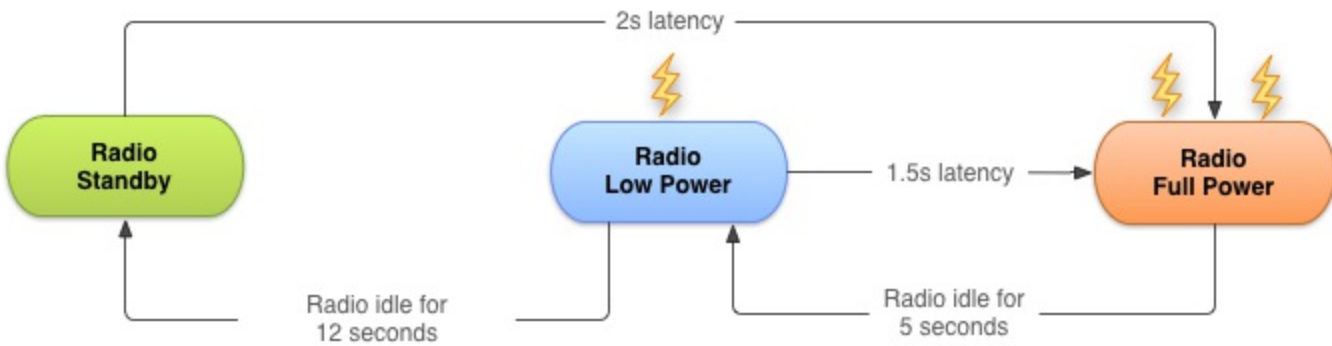
একটি পূর্ণাঙ্গভাবে সক্রিয় ওয়ারলেস রেডিও নির্দিষ্ট পাওয়ার ব্যবহার করে, সুতরাং এটা যখন কোন ব্যবহারের সাথে সম্পৃক্ত না থাকে তখন পাওয়ার রক্ষা (সেভ) করতে বিভিন্ন এনার্জি অবস্থার মধ্যে পরিণত হতে থাকে, প্রয়োজনের সময় রেডিও এর "পাওয়া আপের" (শুরু করা) এর সাথে সম্পর্কিত ল্যাটেন্সিকে কমিয়ে আনতে চেষ্টা করে।

একটি 3G নেটওয়ার্ক রেডিওর জন্য স্টেট মেশিনে তিনটা এনার্জি অবস্থা থাকে:

1. সম্পূর্ণ পাওয়ার (Full power): এককানেকশন যখন সক্রিয় থাকে তখন ব্যবহৃত হয়, ডিভাইসকে সর্বোচ্চ সম্ভাব্য হারে এর ডাটা ট্রান্সফার করতে দেয়।
2. কম পাওয়ার (Low power): একটি মাঝামাঝি অবস্থা যা পূর্ণ স্টেটে প্রায় ৫০% ভাগ ব্যাটারি ব্যবহার করে।
3. স্ট্যান্ডবাই : সর্বনিম্ন এনার্জি স্টেট (অবস্থা) যে সময়ে কোন নেটওয়ার্ক কানেকশন সক্রিয় থাকে না বা প্রয়োজন হয় না।

যখন কম এবং নিষ্ক্রিয় স্টেট (অবস্থা) উল্লেখযোগ্যভাবে কম ব্যাটারি নিঃশেষ করে, তারা নেটওয়ার্ক রিকোয়েস্টে উল্লেখযোগ্য লেটেন্সিও নিয়ে আসে। অল্প স্টেট থেকে পূর্ণ পাওয়ারে ফিরে আসতে এটা ১.৫ সেকেন্ড সময় নেয়, নিষ্ক্রিয় অবস্থা থেকে পূর্ণ পাওয়ারে ফিরে আসতে এটা ২ সেকেন্ড সময় নেয়।

লেটেন্সি কমিয়ে আনতে, স্টেট মেশিন লোয়ার (কম) এনার্জি অবস্থায় পরিবর্তন স্থগিত করতে একটি ডিলে (বিলম্ব) ব্যবহার করে। ফিগার ১ একটি প্রচলিত 3G রেডিওর জন্য AT&T's এর টাইমিং ব্যবহার করে।



ফিগার ১: একটি প্রচলিত ৩এ ওয়ারলেস রেডিও স্টেট মেশিন

প্রতিটা ডিভাইসে রেডিও স্টেট মেশিন, নির্দিষ্টভাবে যুক্ত পরিবর্তন লেটেন্সি বিলম্ব ("tail time") এবং শুরু করে, ব্যবহৃত ওয়ারলেস রেডিও প্রযুক্তির (2G, 3G, LTE ইত্যাদি) উপর ভিত্তি করে ভিন্ন হতে পারে এবং কেরিয়ার নেটওয়ার্ক যার উপরে ডিভাইস পরিচালিত হয় তার দ্বারা নির্ধারিত এবং

কনফিগারেশন করা হয়।

এই অনুশীলনী একটি প্রচলিত 3G ওয়ারলেস রেডিওর জন্য প্রতিনিধি স্টেট মেশিনকে নিয়ে আলোচনা করে, data provided by AT&T (http://www.research.att.com/articles/featured_stories/2011_03/201102_Energy_efficient?fbid=z12VzO4Xy_R) এর ভিত্তি করে। কিনুত সাধারণ নিয়মকানুন এবং ফলে যে সর্বোত্তম চর্চা হয় তা সকল ওয়ারলেস রেডিও বাস্তবায়নের জন্য প্রযোজ্য।

এই পদ্ধতি বিশেষভাবে সাধারণ ওয়েব ব্রাউজিং এর জন্য কার্যকরী যেহেতু এটা ইউজার কর্তৃক ওয়েব ব্রাউজ করার সময় অনাহুত লেটেন্সি পরিহার করে। তুলনামূলকভাবে কম টেইল-টাইমও নিশ্চিত করে যাতে যখনই একটি ব্রাউজিং সময় শেষ হয় রেডিও একটি কম এনার্জি অবস্থায় চলে যেতে পারে।

দুর্ভাগ্যজনকভাবে, এই পদ্ধতি অ্যাপ্লয়েডের মতো আধুনিক স্মার্টফোনে অকার্যকরী অ্যাপ নিয়ে আসতে পারে, যখন অ্যাপ ফোরগ্রাউন্ড (যেখানে লেটেন্সি গুরুত্বপূর্ণ) এবং ব্যাকগ্রাউন্ড (যেখানে ব্যাটারি লাইফ গুরুত্ব পায়) উভয়ই রান করে।

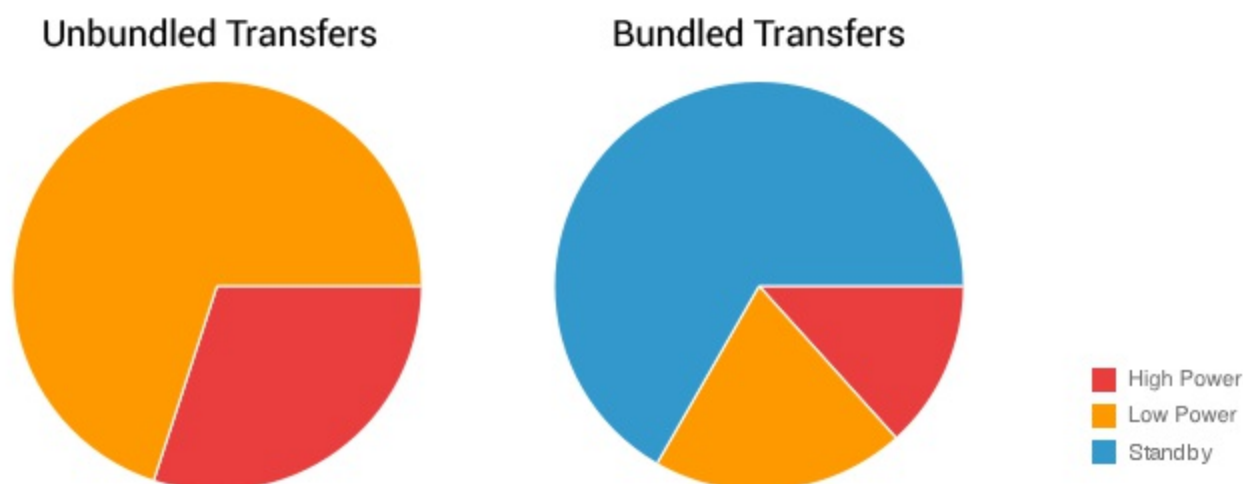
অ্যাপ কীভাবে রেডিও স্টেট মেশিন প্রভাবিত করে

একটি নতুন নেটওয়ার্ক কানেকশন তৈরী করার প্রতিটি সময়, রেডিও পূর্ণ পাওয়ার স্টেটে পরিবর্তন করে। উপরে আলোচিত সাধারণ 3G রেডিও স্টেট মেশিনের ক্ষেত্রে, এটা আপনার ট্রান্সফার সময়কালের জন্য পূর্ণ পাওয়ারে থাকবে-এর সাথে টেইল-টাইমের অতিরিক্ত আরও ৫ সেকেন্ড- কম এনার্জি স্টেটে ১২ সেকেন্ডকে অনুসরণ করার মাধ্যমে। সুতরাং একটি সাধারণ 3G ডিভাইসের জন্য, প্রতিটা ডাটা ট্রান্সফার সেশন রেডিওকে প্রায় ২০ সেকেন্ডের জন্য এনার্জি ড্র করার কারন ঘটায়।

বাস্তবে, এটা বোঝায় একটি অ্যাপ যা ১ সেকেন্ডের জন্য আনবান্ডল ডাটা ট্রান্সফার করে প্রতি ১৮ সেকেন্ড ওয়ারলেস রেডিও টিকে অবিরাম সক্রিয় রাখে, এটাকে উচ্চ পাওয়ারে ফিরিয়ে নিয়ে আসে যেহেতু এটা নিষ্ক্রিয় হয়েছিল। ফলশ্রুতিতে প্রতি মিনিটে উচ্চ পাওয়ার অবস্থায় এটা ১৮ সেকেন্ড ব্যাটারি খরচ করে, এবং বাকী ৪২ সেকেন্ড কম পাওয়ার অবস্থায় খরচ করে।

তুলনামূলকভাবে, একই অ্যাপ যা প্রতি মিনিটের ৩ সেকেন্ডের বান্ডল ট্রান্সফার রেডিওকে ৮ সেকেন্ডের জন্য উচ্চ পাওয়ার স্টেটে রাখে এবং কম পাওয়ারে মাত্র অতিরিক্ত ১২ সেকেন্ডের জন্য রাখে।

দ্বিতীয় উদাহরণ প্রতি মিনিটের অতিরিক্ত ৪০ সেকেন্ডের জন্য রেডিওকে নিষ্ক্রিয় রাখতে দেয়, যা ব্যাটারি খরচের একটি বিরাট ফল এনে দেয়।



ফিগার ২. বান্ডল এবং আনবান্ডল ট্রান্সফার এর মধ্যকার আপেক্ষিক ওয়ারলেস রেডিও পাওয়ার ব্যবহার

ডাটা প্রিফেচ (প্রধান মেমরী থেকে অস্থায়ী স্টোরেজে পরবর্তী ব্যবহারের জন্য) করা

ডাটা প্রিফেচিং স্বাধীন ডাটা ট্রান্সফার সেশন সংখ্যা কমিয়ে আনার একটি কার্যকর উপায়। প্রিফেচিং আপনাকে পূর্ণ ক্ষমতায়, একটি একক কানেকশনের জুরে, একটি একক বাস্ট এর মধ্যে একটি প্রদত্ত সময়ের জন্য আপনার জন্য প্রয়োজনীয় ডাটা ডাউনলোড করতে দেয়।

আপনার ট্রান্সফার সম্মুখে লোড করার দ্বারা, আপনি ডাটা ডাউনলোড করতে প্রয়োজনীয় রেডিও একটিভেশনের সংখ্যা কমিয়ে দেন। ফলশ্রুতিতে আপনি শুধুমাত্র ব্যাটারি লাইফ সেভই করবেন না বরং আপনি লেটেন্সি উন্নতও করতে পারেন, প্রয়োজনীয় ব্যান্ডউইথ কমিয়ে দিতে পারেন এবং ডাউনলোড সময় কমিয়ে দিতে পারেন।

প্রিফেচিং একটি একশন বা ডাটা ভিউ সম্পাদন করার পূর্বে ডাউনলোড সম্পূর্ণ করার জন্য অপেক্ষার কারণে ইন-অ্যাপ লেটেন্সির দ্বারা একটি উন্নত ইউজার এক্সপেরিয়েন্স সরবরাহও করে।

কিন্তু, আক্রমণাত্মকভাবে ব্যবহৃত হওয়া, প্রিফেচিং ব্যাটারি নিঃশেষ হওয়া এবং ব্যান্ডউইথ ব্যবহারের বৃদ্ধির ঝুঁকিকে প্রবর্তন করে- একইভাবে ডাউনলোড কোটা-ডাটা যেটা ব্যবহার হচ্ছে না সেটা ডাউনলোড করার মাধ্যমে। এটা নিশ্চিত করার জন্যও গুরুত্বপূর্ণ যে প্রিফেচিং অ্যাপলিকেশনের শুরু করাটাকে বিলম্ব করায় না যখন অ্যাপ সম্পূর্ণ করার জন্য প্রিফেচের জন্য অপেক্ষা করে। ব্যবহারিক ভাষায় যা প্রগতিশীলভাবে ডাটা প্রক্রিয়াকরনকে বোঝাতে পারে, বা ধারাবাহিক ট্রান্সফার আরম্ভ করে অ্যাপলিকেশনের শুরু করার জন্য প্রয়োজনীয় ডাটার কোনটা ডাউনলোড হবে এবং কোন প্রসেস করা হবে সেটার গুরুত্ব অনুসারে।

আপনি কতটা আক্রমণাত্মকভাবে প্রিফেচ করছেন সেটা নির্ভর করে ডাউনলোড হওয়া ডাটার আকারের উপর এবং এটা কীভাবে ব্যবহৃত হচ্ছে তার উপর। একটি রাফ গাইড হিসাবে, উপরের বর্ণিত স্টেট মেশিনের উপর ভিত্তি করে, ডাটার জন্য যার চলতি ইউজার সেশনের মধ্যে থেকে ৫০% ব্যবহৃত হওয়ার সুযোগ আছে, আপনি প্রচলিতভাবে প্রায় ৬ সেকেন্ডের জন্য প্রিফেচ করতে পারেন (প্রায় ১-২ MB) সম্ভাব্য ডাউনলোড করা অব্যবহৃত ডাটার খরচ, ডাউনলোড না হওয়া সেভ করা ওই ডাটা ম্যাচ করে শুরু করার পূর্বে।

সাধারণভাবে বলতে, ঐ ধরনের ডাটা প্রিফেচ করা একটি ভালো চর্চা এমন যে প্রতি ২ থেকে ৫ মিনিটে আপনার অন্য ডাটা ডাউনলোড শুরু করার প্রয়োজন হবে এবং ১ থেকে ৫ মেগাবাইটস অনুযায়ী।

এই নিয়ম অনুসারে, বড় ডাউনলোড- যেমন ভিডিও ফাইল- নিয়মিত বিরতিতে একসাথে ডাউনলোড হওয়া উচিত (প্রতি ২ থেকে ৫ মিনিট), কার্যকরীভাবে কয়েক মিনিটের মধ্যেই দেখা হবে শুধু এমন ভিডিও ডাটা প্রিফেচ করা।

উল্লেখ্য পরবর্তী ডাউনলোড বান্ডলড হওয়া উচিত, যা পরবর্তী অধ্যায়ে আলোচনা করা হয়েছে, Batch Transfers and Connections, এবং এই অনুমান কানেকশনের ধরণ এবং গতির উপর নির্ভর করে ভিন্ন হয়, যা Modify your Download Patterns Based on the Connectivity Type আলোচনা করা হয়েছে।

আসুন কিছু বাস্তব উদাহরণ দেখা যাক:

একটি মিউজিক প্লেয়ার

আপনি একটি সম্পূর্ণ এ্যালবাম প্রিফেচ করার জন্য পছন্দ করতে পারেন, কিনুত উজার প্রথম গান শোনার পর শোনা বন্ধ করে দিতে পারে, সেক্ষেত্রে আপনি একটি বেশ কিছু ব্যান্ডউইথড এর পরিমান এবং ব্যাটারির লাইফ অপচয় করবেন।

এর চেয়ে ভালো পদ্ধতি হতে পারে একটি গান হচ্ছে তার সাথে একটি গানের বাফার রাখা। মিউজিক স্ট্রিমিং এর জন্য, একটি অবিরত স্ট্রিম যা সব সময় রেডিও একটিভ ধারণ করে তার বদলে, বার্স্ট (bursts) অডিও স্ট্রিমটি প্রেরণ করতে HTTP লাইভ স্ট্রিমিং ব্যবহার করাটা বিবেচনা করুন, উপরে আলোচনা করা প্রিফেচিং পদ্ধতি অনুসরণ করুন।

একটি নিউজ রিডার

একটি ক্যাটাগরি নির্বাচিত (সিলেক্ট) করার পর শুধু হেডলাইন ডাউনলোড করার মাধ্যমে অনেক নিউজ অ্যাপ ব্যান্ডউইথড কমিয়ে আনার চেষ্টা করে, পূর্ণ আর্টিকেল ডাউনলোড করে যখন ইউজার এটা পড়তে চায় এবং থাম্বনেইল (ছবি) দেখানো হবে শুধু মাত্র যখন তারা ভিউয়ে স্ক্রল করে ঢুকবে।

এই অ্যাপ্রোচ ব্যবহার করে, ইউজারের নিউজ রিডিং সেশনের জন্য রেডিও কে সক্রিয় থাকতে বাধ্য করে, যেহেতু তারা হেডলাইন স্ক্রল করে, ক্যাটাগরি পরিবর্তন করে, আর্টিকেল পড়ে। শুধু তাই নয়, প্রতিনিয়ত এনার্জি স্টেটের মধ্যে বদল হওয়া উল্লেখযোগ্য লেট্যান্সি তৈরী করবে যখন ক্যাটাগরি বা রিডিং আর্টিকেল পরিবর্তন হয়।

শুরুর সময়ে একটি যৌক্তিক পরিমান ডাটা প্রিফেচ করা একটি ভালো পদ্ধতি হতে পারে, নিউজ হেডলাইন এবং ছবির (থাম্বনেইল) প্রথম সেট দিয়ে শুরু করুন-একটি কম ল্যাটেন্সি শুরুর সময় নিশ্চিত করুন-এবং বাকী হেডলাইন এবং থাম্বনেইল জারি রাখুন একইভাবে নিদেন পক্ষে প্রাথমিক হেডলাইন তালিকা থেকে বিদ্যমান প্রতিটা আর্টিকেলের জন্য আর্টিকেল টেক্সট।

প্রতিটা শিরোনাম, থাম্বনেইল, আর্টিকেল টেক্সট এবং এমনকি হয়তো পূর্ণ আর্টিকেল ছবি প্রিফেঞ্চ করতে অন্য বিকল্প হচ্ছে- সাধারণত একটি পূর্ব নির্ধারিত সুচিতে ব্যাকগ্রাউন্ডের মধ্যে। এই পদ্ধতি যে কনটেন্ট কখনই ব্যবহৃত হয় নাই তা ডাউনলোড করে উল্লেখযোগ্য ব্যান্ডউইথড এবং ব্যাটারি লাইফ ব্যয় করার ঝুঁকি নেয়, তাই এটা সতর্কতার সাথে ব্যবহার করা উচিত।

ডাউনলোড করার পূর্ণ শিডিউল তৈরী করার একটি উপায় হচ্ছে শুধুমাত্র তখন যখন ওয়াই-ফাইয়ে যুক্ত হয়, এবং সম্ভবত ডিভাইস পরিবর্তন হচ্ছে শুধুমাত্র তখন। এটা Modify your Download Patterns Based on the Connectivity Type অধ্যায়ে আরো বিস্তারিত আলোচনা করা হয়েছে।

ব্যাচ ট্রান্সফার এবং সংযোগ

একটি সংযোগ শুরু করার প্রতিটা সময়- নির্বিশেষে সম্পৃক্ত ডাটা ট্রান্সফার এর সাইজ- আপনি রেডিওকে ২০ সেকেন্ডের কাচাকাছি পাওয়ার ড্র করার কারন ঘটাতে পারেন যখন সাধারণ 3G ওয়াররেস রেডিও ব্যবহার করেন।

একটি অ্যাপ যা প্রতি ২০ সেকেন্ড পরপর অনুসন্ধান করে শুধুমাত্র জানার জন্য যে অ্যাপটি রান করছে এবং ইউজারের কাছে দৃশ্যমান আছে, অনির্দিষ্ট কাল ধরে রেডিও পাওয়ার চালু করে রাখে, ফলশ্রুতিতে একটি উল্লেখযোগ্য ব্যাটারি খরচ করে প্রায় কোন ডাটা স্থানান্তরিত না করেই।

এটার সাথে মনে রাখবেন আপনার ডাটা স্থানান্তর বাবদল করা এবং একটি অমিমাংসিত ট্রান্সফার কিউ তৈরী করাটা গুরুত্বপূর্ণ। সঠিকভাবে করতে, আপনি কার্যকরীভাবে ফেজ-শিফট ট্রান্সফার করতে পারেন যা একটি অনুরূপ সময় উইন্ডোর মধ্যে ঘটার কারনে, তাদের একই সময়ে সংঘটিত হতে- নিশ্চিত করতে যে যত কম সময়ে সম্ভব রেডিও পাওয়ার ড্র করতে পারে।

এই পদ্ধতির মৌলিক দর্শন হচ্ছে প্রতিটা ট্রান্সফার সময়কালে যতটুকু সম্ভব ডাটা ট্রান্সফার করা যা আপনার জন্য প্রয়োজনীয় সেশনের সংখ্যা কমিয়ে আনতে পারে।

তার মানে আপনার উচিত বিলম্ব সহনশীল স্থানান্তর কিউ করার মাধ্যমে আপনার ডাটা ব্যাচ করা এবং শিডিউল আপডেট এবং প্রিফেচিংকে বিরত রাখা, যাতে এটার সবগুলোই সম্পাদতি হয় যখন সময় স্পর্ষকাতর ট্রান্সফার প্রয়োজন হয়। একইভাবে, আপনার শিডিউল করা আপডেট এবং নিয়মিত প্রিফেচিং এর উচিত আপনার অসমাপ্ত ট্রান্সফার কিউয়ের সম্পাদন শুরু করা।

একটি ব্যবহারিক উদাহরনের জন্য Prefetch Data থেকে পূর্ববর্তী উদাহরণে ফিরে যান।

একটি নিউজ অ্যাপলিকেশন নিন যা উপরে বর্ণিত প্রিফেচিং রুটিন ব্যবহার করে। নিউজ রিডার এর ইউজারের রিডিংয়ের ধরণ বুঝতে এবং সবচেয়ে জনপ্রিয় গল্প র্যাংকিং করতে বিশ্লেষণাত্মক তথ্য সংগ্রহ করে। খবরকে ফ্রেশ রাখতে এটা প্রতি ঘন্টায় আপডেট চেক করে। ব্যান্ডউইথড সংরক্ষণ করতে, প্রতিটা আর্টিকেলের জন্য পূর্ণ ফটো ডাউনলোড না করে, এটা শুধুমাত্র থাম্বনেইল প্রিফেচ করে এবং ফটো ডাউনলোড করে তখনই যখন এটা সিলেক্ট করা হয়।

এই উদাহরনে, অ্যাপের মধ্যে থেকে সংগ্রহিত সকল বিশ্লেষণধর্মী তথ্যর ডাউনলোড হওয়ার জন্য একসাথে বাবদল হওয়া এবং কিউ হওয়া উচিত, স্থানান্তরিত হওয়ার বদলে যেহেতু এটা সংগ্রহিত হয়েছে। ফলশ্রুতিতে যে বাবদল ডাওয়া যায় তার স্থানান্তরিত হওয়া উচিত যখন হয় একটি পূর্ণ সাইজের ফটো ডাউনলোড হতে থাকে অথবা যখন প্রতিঘন্টার আপডেট তার কার্যক্রম করতে থাকে।

কোন সময়-স্পর্ষকাতর বা চাহিদা ভিত্তিক ট্রান্সফার- যেমন একটি পূর্ণ সাইজের ইমেজ ডাউনলোড করা- নিয়মিত শিডিউল করা আপডেট করা থেকে বিরত রাখা উচিত। একটি পরিকল্পিত আপডেট শিডিউল একই সময়ে নির্বাহ হওয়া উচিত যেহেতু চাহিদা ভিত্তিক ট্রান্সফার, ইন্টারভেল (অন্তর্বর্তী কাল) সেট করার পর পরবর্তী আপডেট এর সাথে ঘটতে নির্ধারিত হয়ে থাকে। এই পদ্ধতি একটি নিয়মিত আপডেট সম্পাদনের ব্যয় কমিয়ে আনে, প্রয়োজনীয় সময়-স্পর্ষকাতর ফটো ডাউনলোডে পিগি-ব্যাংকিং করার মাধ্যমে।

কানেকশন (সংযোগ) কমিয়ে আনা

এটা সাধারণভাবে নতুন একটা শুরু করার চেয়ে বিদ্যমান সংযোগ পূর্ণব্যবহার করতে আরও কার্যকরী। সংযোগের পূর্ণব্যবহার নেটওয়ার্ককে জনাধিক্যেও কার জ্যাম হওয়ার এবং এই ধরনের নেটওয়ার্ক ডাটা ইস্যুও ক্ষেত্রে আরও বুদ্ধিমত্তার সাথে প্রতিক্রিয়া করতে অনুমোদনও করে।

ডাটা ডাউনলোড করতে মাল্টিপল যুগপোযোগী সংযোগ, বা বহুমুখি উপর্যুপরি GET রিকোয়েস্ট তৈরী করার বদলে, যেখানে আপনার উচিত ঐ রিকোয়েস্ট একটি একক GET এর মধ্যে বান্ডল করা।

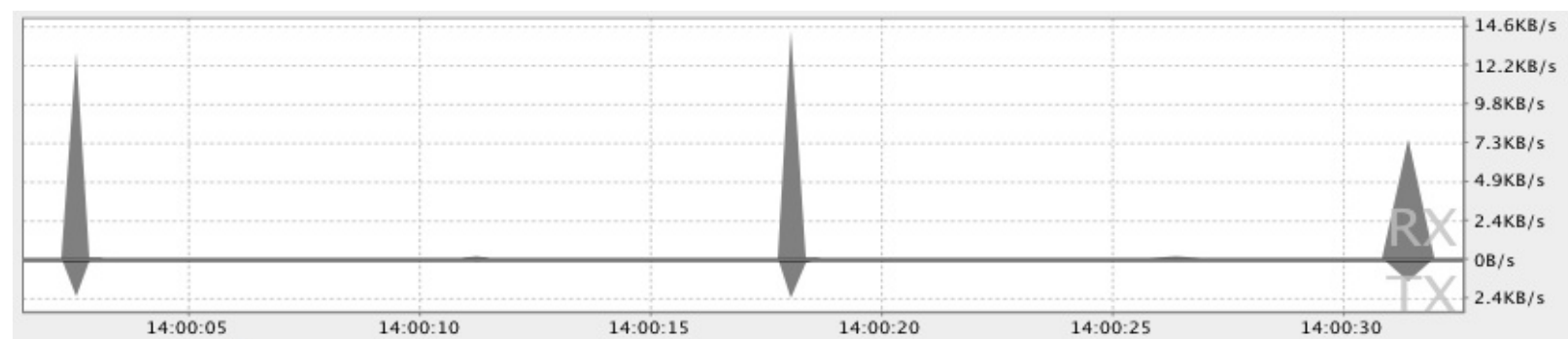
উদাহরনস্বরূপ, কিছু নিউজ ধরনের জন্য বহুমুখি অনুসন্ধান করার চেয়ে একটি একক রিকোয়েস্ট/রেসপন্স এর মধ্যে ফেরত হওয়া প্রতিটা নিউজের জন্য একটি একক রিকোয়েস্ট তৈরী করতে এটা আরও কার্যকরী হতে পারে। সার্ভার এবং ক্লায়েন্ট টাইমআউটের সাথে সম্পৃক্ত টারমিনেশন/টারমিনেশন স্বীকারোক্তি প্যাকেট স্থানান্তর করার জন্য ওয়ারলেস রেডিওর সক্রিয় হওয়ার প্রয়োজন, সুতরাং ঐ টাইমআউটের জন্য অপেক্ষা করার চেয়ে এটা আপনার সংযোগ যখন তা কোন ব্যবহারে না থকে তা বন্ধ করার আরও একটি ভালো চর্চা।

যেটা বলে, একটি সংযোগ তারাতারি বন্ধ করা এটাকে পূর্ণব্যবহৃত হওয়ার হাত থেকে রক্ষা করে, যা পরবর্তীতে একটি নতুন সংযোগ প্রতিষ্ঠা করার জন্য অতিরিক্ত ওভারহেড আকাঙ্ক্ষা করে। একটি উপকারী বোঝাপড়া হচ্ছে তাৎক্ষণিকভাবে সংযোগ বন্ধ না করা, কিনুত অন্তর্নিহিত টাইমআউট মেয়াদশেষের পূর্বে এর খুব কাছাকাছি থাকে।

চিহ্নিত জায়গা চিহ্নিত করতে DDMS নেটওয়ার্ক ট্রাফিক টুল ব্যবহার করে

অ্যান্ড্রয়েড ডিভিএমএস (Dalvik Debug Monitor Server) একটি বিস্তারিত করা নেটওয়ার্ক ব্যবহারের ট্যাব অন্তর্ভুক্ত করে যা এটাকে অনুসরণ করতে সম্ভব করে যখন আপনার অ্যাপ্লিকেশন নেটওয়ার্ক রিকোয়েস্ট তৈরী করছে। এই টুল ব্যবহার করে, আপনি মনিটর করতে পারেন কীভাবে এবং কখন আপনার অ্যাপ ডাটা ট্রান্সফার করে এবং যথাযথভাবে মৌলিক কোড অপটিমাইজ করে।

ফিগার ৩ দেখায় প্রায় ১৫ সেকেন্ড পরপর অল্প পরিমান ডাটা ট্রান্সফার করার একটি প্যাটার্ন, যা পরামর্শ দেয় যে, প্রতিটা রিকোয়েস্ট বা আপলোড বান্ডল করা প্রিফেচ করতে কার্যকারিতা নাটকীয়ভাবে উন্নত হতে পারে।



ফিগার ৩. DDMS দিয়ে নেটওয়ার্ক ব্যবহার ট্র্যাক করা

আপনার ডাটা ট্রান্সফারের ফ্রিকোয়েন্সি (পুনরাবৃত্তি) এবং প্রতিটা সংযোগের সময় ট্রান্সফার হওয়া ডাটার পরিমান মনিটর করার মাধ্যমে, আপনি আপনার অ্যাপ্লিকেশনের এলাকা চিহ্নিত করতে পারেন যা আরও ব্যাটারি-কার্যকারিতা তৈরী করতে পারে। সাধারনভাবে আপনি সংক্ষিপ্ত স্পাইক খুজতে পারেন যা বিলম্ব হতে পারে, অথবা যা একটি পরবর্তী ট্রান্সফার বিরত হওয়ার কারন ঘটায়।

স্পাইক ট্রান্সফারের কারন আরও ভালোভাবে চিহ্নিত করতে, ট্রাফিক স্টেট এপিআই পদ্ধতি `TrafficStats.setThreadStatsTag()` ব্যবহার করে একটি থ্রেডের মধ্যে থেকে ঘঠিত ডাটা ট্রান্সফারে ট্যাগ করতে আপনাকে অনুমোদন করে, ম্যানুয়ালি `tagSocket()` এবং `untagSocket()` ব্যবহার করে একক সকেট ট্যাগ করা অনুসরণ করার মাধ্যমে। উদাহরণস্বরূপ:

```
TrafficStats.setThreadStatsTag(0xF00D);
TrafficStats.tagSocket(outputSocket);
// Transfer data using socket
TrafficStats.untagSocket(outputSocket);
```

চলতি `getThreadStatsTag()` ভ্যালুর উপর ভিত্তি করে অ্যাপাশে `HttpClient` এবং `URLConnection` লাইব্রেরী স্বয়ংক্রিয়ভাবে সকেট ট্যাগ করে। এই লাইব্রেরীগুলো সকেটকে ট্যাগ এবং আনট্যাগ করে

যখন জীবন্ত-রাখা পুল এর মাধ্যমে রিসাইকেল করা হয়।

```
TrafficStats.setThreadStatsTag(0xF00D);
try {
    // Make network request using HttpClient.execute()
} finally {
    TrafficStats.clearThreadStatsTag();
}
```

অ্যান্ড্রয়েড ৪.০ সকেট ট্যাগিংকে সাপোর্ট করে, কিন্ত রিয়েল-টাইম স্ট্যাটস শুধুমাত্র অ্যান্ড্রয়েড ৪.০.৩ বা এর চেয়ে উপরের সংস্করণ দ্বারা রান করা ডিভাইসে প্রদর্শিত হবে।

নিয়মিত আপডেটের প্রভাব কমানো

(http://developer.android.com/training/efficient-downloads/regular_updates.html)

নিয়মিত আপডেটের অপটিমাল ফ্রিকোয়েন্সি ডিভাইসের অবস্থান, নেটওয়ার্ক কানেকটিভিটি, ইউজারের আচরণ এবং ইউজারের পছন্দের উপর নির্ভর করে।

Optimizing Battery Life কীভাবে ব্যাটারি-কার্যকর অ্যাপ তৈরী করতে হয় তা আলোচনা করে যা ধারক ডিভাইসের অবস্থার উপর নির্ভর করে তাদের রিফ্রেশ ফ্রিকোয়েন্সি পরিবর্তন করে। ঐগুলির মধ্যে রয়েছে নিষ্ক্রিয় করা ব্যাকগ্রাউন্ড সার্ভিস আপডেট যখন আপনি আপনার কানেকটিভিটি হারিয়ে ফেলেন এবং আপডেটের হার কমিয়ে ফেলে যখন ব্যাটারি লেভেল কমে যায়।

এই অনুশীলনী কীভাবে আপনার রিফ্রেশ ফ্রিকোয়েন্সি মূল ওয়ারলেস রেডিও স্টেট মেশিনে ব্যাকগ্রাউন্ড আপডেটের প্রভাব সবচেয়ে ভালোভাবে কমিয়ে আনতে ভিন্ন হতে পারে তা পরীক্ষা করে।

পোলিং এ বিকল্প হিসাবে গুগল ক্লাউড মেসাজিং ব্যবহার

সবসময় আপনার অ্যাপ আপনার সার্ভার পোল করে যদি কোন আপডেটের প্রয়োজন হয় তা পরীক্ষা করে, যদি একটি সাধারণ ৩ জি কানেকশনে প্রায় ২০ সেকেন্ডের জন্য আপনি ওয়ারলেস রেডিও সক্রিয় করেন, অপ্রয়োজনীয়ভাবে পাওয়ার ড্র করেন।

Google Cloud Messaging for Android (GCM) (দেখুন <http://developer.android.com/google/gcm/index.html>) হচ্ছে একটি হালকা ধরনের কৌশল যা সার্ভার থেকে একটি নির্দিষ্ট অ্যাপ ইনস্ট্যান্সে ডাটা স্থানান্তরের কাজে ব্যবহৃত হয়। GCM ব্যবহার করে, আপনার সার্ভার একটি নির্দিষ্ট ডিভাইসে রান করা আপনার অ্যাপকে জানাতে পারে যে এর জন্য নতুন ডাটা রয়েছে।

পোলিং এর সাথে তুলনা করে, যেখানে আপনার অ্যাপকে অবশ্যই নতুন ডাটার জন্য অনুসন্ধান করতে নিয়মিত সার্ভারকে পিং করতে হবে, এই ইভেন্ট চালিত মডেল আপনার অ্যাপকে একটি নতুন কানেকশন তৈরী করতে দেয় শুধু তখন যখন এটা জানে ডাউনলোড করা জন্য ডাটা আছে।

ফলাফল হচ্ছে একটি অপ্রয়োজনীয় কানেকশন কমানো এবং আপনার অ্যাপলিকেশনের মধ্যে আপডেট ডাটার জন্য একটি কমিয়ে আনা লেটেন্সি।

GCM একটি স্থায়ী TCP/IP কানেকশন ব্যবহার করে বাস্তবায়িত হয়। যখন আপনার নিজস্ব পুশ সার্ভিস বাস্তবায়ন করা সম্ভব, এটাই সবচেয়ে ভালো যে এক্ষেত্রে GCM ব্যবহার করা। এটা স্থির কানেকশনের সংখ্যা কমিয়ে আনে এবং প্লাটফর্মকে ব্যান্ডউইথড অপটিমাইজ করতে এবং ব্যাটারি লাইফের সাথে সম্পৃক্ত প্রভাব কমিয়ে আনতে দেয়।

ইনএক্স্যাক্ট রিপিটিং এ্যালার্ম এবং এক্সপোনেন্ট ব্যাকঅফ দিয়ে পোলিং অপটিমাইজ করা

যেখানে পোলিং প্রয়োজন, ইউজার এক্সপেরিয়েন্স থেকে হ্রাস না করেই আপনার অ্যাপের ডিফল্ট ডাটা রিফ্রেশ ফ্রিকোয়েন্সি যতদূর সম্ভব কমিয়ে সেট করাটাই ভালো।

একটি সরল পদ্ধতি হচ্ছে ইউজারকে তাদের প্রয়োজনীয় আপডেট রেট নিজ পছন্দ মতো সুস্পষ্টভাবে সেট করতে দেয়া, তাদেরকে ডাটা রিফ্রেশনেস এবং ব্যাটারি লাইফের মধ্যকার নিজস্ব ভারসাম্য নির্ধারণ করতে দেয়া।

যখন আপডেট শিডিউল করা হয়, ইনএক্স্যাক্ট রিপিটিং এ্যালার্ম ব্যবহার করুন যা সিস্টেমকে প্রতিটা এ্যালার্ম ট্রিগারের সময় "ফেজ শিফট" ("phase shift") করতে দেয়।

```
int alarmType = AlarmManager.ELAPSED_REALTIME;
long interval = AlarmManager.INTERVAL_HOUR;
long start = System.currentTimeMillis() + interval;

alarmManager.setInexactRepeating(alarmType, start, interval, pi);
```

যদি কতিপয় এ্যালার্ম একই সময়ে ট্রিগার করার জন্য শিডিউল করা হয়ে থাকে, এই ফেজ শিডিউল একই সাথে ট্রিগার হতে দিতে পারে, প্রতিটা আপডেটকে একটি একক সক্রিয় রেডিও স্টেট পরিবর্তনের উপর piggyback করতে দেয়।

যেখানেই সম্ভব, সমগোত্রীয় WAKEUP এ না করে ELAPSED_REALTIME বা RTC এ আপনার এ্যালার্ম টাইপ সেট করুন। এটা অপেক্ষা করার মাধ্যমে ব্যাটারি প্রভাব অধিক হারে কমিয়ে আনে, ফোনটি এ্যালার্ম ট্রিগার হওয়ার পূর্ব পর্যন্তত যতক্ষণ না স্ট্যান্ডবাই মোডে না থাকে।

আপনি আপনার অ্যাপ কত সম্প্রতি ব্যবহার করেছেন তার উপর ভিত্তি করে তাদের ফ্রিকোয়েন্সি কমিয়ে দেয়ার মাধ্যমে এই শিডিউল করা এ্যালার্মগুলোর প্রভাব কমিয়ে আনতে পারেন।

একটি উপায় হচ্ছে আপনার আপডেটের ফ্রিকোয়েন্সি (এবং/বা প্রিফেচিং এর ডিগ্রি আপনি সম্পাদন করেন) কমাতে একটি এক্সপোনেনশিয়াল ব্যাক-অফ ধরন বাস্তবায়ন করা, যদি অ্যাপটি পূর্ববর্তী আপডেট থেকে ব্যবহৃত হয়ে না আসে। এটা কখনও কখনও একটি কমপক্ষে আপডেট ফ্রিকোয়েন্সি জানাতে এবং যখনই অ্যাপ ব্যবহৃত হয় তখন ফ্রিকোয়েন্সি পূরণায় সেট করতে উপকারী। উদাহরণস্বরূপ:

```
SharedPreferences sp =
    context.getSharedPreferences(PREFS, Context.MODE_WORLD_READABLE);

boolean appUsed = sp.getBoolean(PREFS_APPUSED, false);
long updateInterval = sp.getLong(PREFS_INTERVAL, DEFAULT_REFRESH_INTERVAL);

if (!appUsed)
    if ((updateInterval *= 2) > MAX_REFRESH_INTERVAL)
```

```
updateInterval = MAX_REFRESH_INTERVAL;
```

```
Editor spEdit = sp.edit();  
spEdit.putBoolean(PREFS_APPUSED, false);  
spEdit.putLong(PREFS_INTERVAL, updateInterval);  
spEdit.apply();
```

```
rescheduleUpdates(updateInterval);  
executeUpdateOrPrefetch();
```

আপনি ব্যর্থ কানেকশন এবং ডাউনলোড এররের প্রভাব কমিয়ে আনতে একই ধরনের এক্সপোনেনশিয়াল ব্যাক-অফ ধরন ব্যবহার করতে পারেন।

আপনি আপনার সার্ভারের সাথে যোগযোগ করতে সামর্থ্য হোন এবং ডাটা ডাউনলোড করুন বা না করুন একটি নেটওয়ার্ক কানেকশন আরম্ভ করার খরচ একই। সময়-সংবেদী ট্রান্সফার যেখানে সফল সমাপ্তি গুরুত্বপূর্ণ সম্পৃক্ত ব্যাটারি প্রভাব কমিয়ে আনার জন্য পূরণায় চেষ্টা করার ফ্রিকোয়েন্সি কমাতে একটি এক্সপোনেনশিয়াল ব্যাক-অফ এ্যালোগরিদম ব্যবহার করা যেতে পারে, উদাহরণস্বরূপ:

```
private void retryIn(long interval) {  
    boolean success = attemptTransfer();  
  
    if (!success) {  
        retryIn(interval*2 < MAX_RETRY_INTERVAL ?  
            interval*2 : MAX_RETRY_INTERVAL);  
    }  
}
```

বিকল্পভাবে, ব্যর্থতা সহিষ্ণু ট্রান্সফারের জন্য (যেমন, নিয়মিত আপডেট), আপনি শুধু অসফল কানেকশন এবং ট্রান্সফার চেষ্টা এড়িয়ে যেতে পারেন।

রিডানডেন্ট ডাউনলোড অনাবশ্যক

(http://developer.android.com/training/efficient-downloads/redundant_redundant.html)

আপনার ডাউনলোড কমানোর সবচেয়ে মৌলিক উপায় হচ্ছে যা আপনার প্রয়োজন শুধু সেটাই ডাউনলোড করুন। ডাটার নিরিখে, যার মানে REST APIs বাস্তবায়ন করা যা আপনাকে অনুসন্ধান নির্ণয়কে সুনির্দিষ্ট করতে দেয় যা প্যারামিটার ব্যবহার করে ডাটা ফেরত আসাকে সীমিত করে,, যেমন আপনার শেষ আপডেটের সময়টি।

একইভাবে, যখন ইমেজ ডাউলোড করা হয়, ক্লায়েন্ট কমানো পূর্ণ সাইজের ইমেজ ডাউনলোড না করে সার্ভার-সাইডের ইমেজ সাইজ কমানোটাই ভালো।

স্থানীয়ভাবে ফাইল ক্যাশে (Cache) করা

আরেকটি গুরুত্বপূর্ণ কৌশল হচ্ছে একই/অনুরূপ ডাটা ডাউনলোড করা পরিহার করা। দৃঢ়ভাবে ক্যাশে (caching) করার মাধ্যমে আপনি এটা করতে পারেন। সবসময় স্ট্যাটিক রিসোর্স ক্যাশে করুন, যার মধ্যে রয়েছে চাহিদাকৃত (অন-ডিমান্ড) ডাউনলোড যেমন পূর্ণ সাইজের ইমেজ, যৌক্তিকভাবে যতটুকু দীর্ঘক্ষণ সম্ভব ততক্ষণ পর্যন্ত। আপনার অন-ডিমান্ড ক্যাশের সাইজ ব্যবস্থা করতে আপনাকে এটা নিয়মিত ফ্ল্যাশ করতে সক্রিয় করতে চাহিদাকৃত (অন-ডিমান্ড) রিসোর্স পৃথকভাবে স্টোর করা উচিত।

এটা নিশ্চিত করতে যে আপনার ক্যাশিং আপনার অ্যাপের মধ্যে পুরাতন ডাটা প্রদর্শন করার কারন না হয়, সময় বের করার ব্যাপারে নিশ্চিত হোন, যার মধ্যে রিকোয়েস্ট করা কনটেন্ট শেষবার আপডেট হয়েছিল এবং যখন মেয়াদ শেষ হয়েছিল, যার মধ্যে থেকে HTML কে রেসপন্স করা হেডারস। এটা আপনাকে স্থির করতে দেয় যখন সম্পূর্ণ কনটেন্ট রিফ্রেশড হওয়া উচিত।

```
long currentTime = System.currentTimeMillis();

URLConnection conn = (URLConnection) url.openConnection();

long expires = conn.getHeaderFieldDate("Expires", currentTime);
long lastModified = conn.getHeaderFieldDate("Last-Modified", currentTime);

setDataExpirationDate(expires);

if (lastModified < lastUpdateTime) {
    // Skip update
} else {
    // Parse update
}
```

এই পদ্ধতি ব্যবহার করে, আপনি কার্যকরীভাবে ডায়নামিক কনটেন্ট ক্যাশে করতে পারেন যখন নিশ্চিত করে এটা আপনার অ্যাপের মধ্যে পুরাতন ডাটা প্রদর্শন করবে না।

আপনি কোন ব্যবস্থাপনা ছাড়া বহিরাগত ক্যাশে ডিরেক্টরীতে অসংবেদনশীল ডাটা ক্যাশে করতে পারেন:

```
Context.getExternalCacheDir();
```

ব্যতিক্রমভাবে, আপনি সুযুগ্ম ব্যবস্থাপিত/ নিরাপদ অ্যাপলিকেশন ব্যবহার করতে পারেন। উল্লেখ্য এই অন্তর্গত ক্যাশে ফ্ল্যাশ হতে পারে যখন সিস্টেম সহজপ্রাপ্য স্টোরেজে কমভাবে (Low) রান করছে।

```
Context.getCache();
```

ক্যাশে লোকেশনে স্টোর হওয়া ফাইল মুছে যেতে পারে যখন অ্যাপলিকেশন আন-ইনস্টল করা হয়।

HttpURL কানেকশন রেসপন্স ক্যাশে ব্যবহার করুন

অ্যান্ড্রয়েড 8.0 HttpURLConnection এ একটি রেসপন্স ক্যাশে যুক্ত করে। আপনি নিচের মতো প্রতিফলন ব্যবহার করে সাপোর্ট করা ডিভাইসে HTTP কে রেসপন্স করা ক্যাশিং সক্রিয় করতে পারে:

```
private void enableHttpResponseBodyCache() {  
    try {  
        long httpCacheSize = 10 * 1024 * 1024; // 10 MiB  
        File httpCacheDir = new File(getCacheDir(), "http");  
        Class.forName("android.net.http.HttpResponseBodyCache")  
            .getMethod("install", File.class, long.class)  
            .invoke(null, httpCacheDir, httpCacheSize);  
    } catch (Exception httpResponseBodyCacheNotAvailable) {  
        Log.d(TAG, "HTTP response cache is unavailable.");  
    }  
}
```

এই উদাহরণ কোড প্রথম দিককার রিলিজকে প্রভাবিত না করেই অ্যান্ড্রয়েড 8.0+ ডিভাইসে রেসপন্স ক্যাশে চালু করতে পারে।

একটি নেটওয়ার্ক কানেকশন ওপেন করার প্রয়োজনীয়তাকে বাদ দিয়ে ক্যাশে ইনস্টলড দিয়ে, সম্পূর্ণ ক্যাশেড HTTP রিকোয়েস্ট স্থানিক স্টোরেজ থেকে সরাসরি পরিবেশিত হতে পারে,। ডাউনলোডের সাথে সম্পৃক্ত ব্যান্ডউইথ বাদ দিয়ে শর্তসাপেক্ষভাবে ক্যাশেড রেসপন্স সারভার থেকে তাদের পরিচ্ছন্নতাকে বৈধ করতে পারে।

আনক্যাশেড রেসপন্স ভবিষ্যত রিকোয়েস্টের জন্য রেসপন্স ক্যাশের মধ্যে স্টোর হয়।

কানেকটিভিটি ধরনের উপর ভিত্তি করে ডাউনলোডের ধরন পরিবর্তন করুন

(http://developer.android.com/training/efficient-downloads/connectivity_patterns.html)

যখন এটা ব্যাটারি লাইফে প্রভাব ফেলতে আসে, সকল কানেকশন ধরন সমানভাবে তৈরী হয় না। ওয়াই-ফাই রেডিও এর সমরূপ ওয়ারলেস রেডিওর চাইতে উল্লেখযোগ্য কম ব্যাটারি ব্যবহারই করে না, বিভিন্ন ওয়ারলেস রেডিও প্রযুক্তিতে ব্যবহৃত হওয়া রেডিওতে বিভিন্ন ব্যাটারি ইমপ্লিকেশনও থাকে।

ওয়াই-ফাই ব্যবহার

অধিকাংশ ক্ষেত্রে একটি ওয়াই-ফাই রেডিও উল্লেখযোগ্য পরিমাণ কম ব্যাটারি খরচে অধিকতর ভালো ব্যান্ডউইথ দেয়ার প্রস্তাব করে থাকে। ফলশ্রুতিতে, আপনার উচিত যখনই সম্ভব ডাটা ট্রান্সফার কার্য সম্পাদনের চেষ্টা করা যখন ওয়াই-ফাইয়ে সংযুক্ত হোন।

কানেকটিভিটি পরিবর্তন শোনার জন্য আপনি একটি ব্রডকাস্ট রিসিভার ব্যবহার করতে পারেন যখন একটি ওয়াই-ফাই কানেকশন নির্দিষ্ট পরিমাণ ডাউনলোড, শিডিউল করা আপডেট অগ্রাধিকার, এবং Optimizing Battery Life অনুশীলনের Determining and Monitoring the Connectivity Status তে আলোচিত নিয়মিত আপডেটের ফ্রিকোয়েন্সি অস্থায়ীভাবে বৃদ্ধি করতে সংস্থাপিত হয়।

আরও বেশী বেশী ডাউনলোড করতে ব্যান্ডউইথ ব্যবহার করুন

যখন একটি ওয়ারলেস রেডিওতে সংযুক্ত হয়, উচ্চ পর্যায়ের ব্যান্ডউইথ সাধারণত উচ্চ ব্যাটারি খরচে হয়ে থাকে। এর মানে LTE সাধারণভাবে 3G এর চাইতে বেশী শক্তি খরচ করে, যেটা যথাক্রমে 2G এর চাইতেও ব্যয়বহুল।

এটা বোঝায় যে যখন রেডিও প্রযুক্তির উপর ভিত্তি করে মূল রেডিও স্টেট মেশিন ভিন্ন হয়ে থাকে, সাধারণভাবে বলতে গেলে স্টেট চেঞ্জ টেইল-টাইম এর সম্পর্কিত ব্যাটারি প্রভাব উচ্চমাত্রার ব্যান্ডউইথ রেডিওর জন্য বেশী।

একই সময়ে, উচ্চমাত্রার ব্যান্ডউইথ বোঝায় আপনি আরও দৃঢ়ভাবে প্রিফেচ করতে পারেন, ওই সময় জুড়ে আরও ডাটা ডাউনলোড করতে পারেন। সম্ভবত কম সম্ভ্রানে, কারন টেইল-টাইম ব্যাটারি খরচ অপেক্ষাকৃত বেশী, এটা আপডেটের ফ্রিকোয়েন্সি কমাতে প্রতিটা ট্রান্সফার সেশনের দীর্ঘ সময়ের জন্য রেডিওকে সক্রিয় হিসাবে রাখতে আরও কার্যকরীও।

উদাহরণস্বরূপ, যদি একটি LTE রেডিওর ব্যান্ডউইথ দ্বিগুণ এবং 3G এর শক্তির ব্যয় দ্বিগুণ করে, আপনার উচিত প্রতিটা সেশনে ৪ বার যতটুকু সম্ভব ডাটা ডাউনলোড করা-বা ১০mb এর কাছাকাছি যতটুকু সম্ভব। যখন এত ডাটা ডাউনলোড করবেন, বিদ্যমান স্থানীয় স্টোরেজে আপনার প্রিফেচের প্রভাব বিবেচনা করা এবং নিয়মিতভাবে আপনার ক্যাশে ফ্লাশ করাটা গুরুত্বপূর্ণ।

সক্রিয় ওয়ারলেস রেডিও নির্ধারণ করতে আপনি কানেকটিভিটি ম্যানেজার ব্যবহার করতে পারেন এবং আপনার প্রিফেচিং রেডিও সূচী পরিবর্তন করতেও পারেন:

```
ConnectivityManager cm =  
    (ConnectivityManager) getSystemService (Context.CONNECTIVITY_SERVICE);  
  
TelephonyManager tm =  
    (TelephonyManager) getSystemService (Context.TELEPHONY_SERVICE);  
  
NetworkInfo activeNetwork = cm.getActiveNetworkInfo();  
  
int PrefetchCacheSize = DEFAULT_PREFETCH_CACHE;  
  
switch (activeNetwork.getType()) {  
    case (ConnectivityManager.TYPE_WIFI):  
        PrefetchCacheSize = MAX_PREFETCH_CACHE; break;  
    case (ConnectivityManager.TYPE_MOBILE): {  
        switch (tm.getNetworkType()) {  
            case (TelephonyManager.NETWORK_TYPE_LTE |  
                TelephonyManager.NETWORK_TYPE_HSPAP):  
                PrefetchCacheSize *= 4;  
                break;  
            case (TelephonyManager.NETWORK_TYPE_EDGE |  
                TelephonyManager.NETWORK_TYPE_GPRS):  
                PrefetchCacheSize /= 2;  
                break;  
            default: break;  
        }  
    }
```

```
    break;
}
default: break;
}
```

ক্লাউডে সিঙ্ক (Syncing) করা

(<http://developer.android.com/training/cloudsync/index.html>)

ইন্টারনেট কানেকটিভিটির জন্য শক্তিশালী এপিআই সরবরাহ করে, অ্যান্ড্রয়েড ফ্রেমওয়ার্ক সমৃদ্ধ ক্লাউড-সক্রিয় অ্যাপ তৈরী করতে সহায়তা করে যা তাদের ডাটাকে একটি রিমোট ওয়েব সার্ভিসে সমন্বিত করে, নিশ্চিত করুন আপনার সকল ডিভাইস সবসময় সিঙ্ক থাকে এবং আপনার মূল্যবান ডাটা সবসময় ক্লাউডে একটি ব্যাকআপ হিসাবে থাকে।

এই ক্লাস ক্লাউড সক্রিয় অ্যাপলিকেশনের জন্য বিভিন্ন কৌশল আলোচনা করে। আপনার নিজস্ব ব্যাক-এন্ড ওয়েব অ্যাপলিকেশন ব্যবহার করে এটা ক্লাউড দিয়ে ডাটা সিঙ্ক করার বিষয়ে আলোচনা করে যাতে ইউজার তাদের ডাটা রিস্টোর করতে পারে যখন একটি নতুন ডিভাইসে আপনার অ্যাপলিকেশন ইনস্টল করা হয়।

অনুশীলনী

ব্যাকআপ এপিআই ব্যবহার

শিখুন কীভাবে আপনার অ্যান্ড্রয়েড অ্যাপলিকেশনের মধ্যে ব্যাকআপ এপিআই একীভূত করা হয়, যাতে ইউজার ডাটা প্রিফারেন্স, নোট এবং গেম হাই স্কোর আপডেট নিখুতভাবে ইউজারের ডিভাইস জুড়ে রাখতে পারে।

অধিকাংশ গুগল ক্লাউড মেসেজিং তৈরী করা

শিখুন কীভাবে কার্যকরীভাবে মাল্টিকাস্ট মেসাজ সেন্ড করা যায়, বুদ্ধিমত্তার সাথে গুগল ক্লাউড মেসেজিং (GCM) মেসেজে প্রতিক্রিয়া জানানো হয়, এবং সার্ভার দিয়ে কার্যকরীভাবে সিস্ক করতে এন্টগ মেসেজ ব্যবহার করা হয়।

ব্যাকআপ এপিআই ব্যবহার

(<http://developer.android.com/training/cloudsync/backupapi.html>)

যখন একজন ইউজার একটি নতুন ডিভাইস ক্রয় করে বা বিদ্যমান কোন একটিকে পুনরায় সেট করে, তারা আশা করতে পারে যে যখন গুগল প্লে প্রারম্ভিক সেটআপের সময় আপনার অ্যাপকে তাদের ডিভাইসে রিস্টোর করে, একই সাথে অ্যাপের সমস্ত পূর্ববর্তী ডাটাও রিস্টোর করে। বাই ডিফল্ট এটা হয় না, এবং আপনার অ্যাপে ইউজারের সকল অর্জন ও সেটিং হারিয়ে যাবে।

যেখানে ডাটার আয়তন অপেক্ষাকৃত কম (মেগাবাইটের কম) সে অবস্থায়, ইউজারের প্রিফারেন্স, নোটস, গেম হাই স্কোর অথবা অন্য পরিসংখ্যান, ব্যাকআপ এপিআই একটি হালকা সমাধান দিয়ে থাকে। এই অনুশীলনী আপনার অ্যাপের মধ্যে ব্যাকআপ এপিআই একীভূত করার এবং ব্যাকআপ এপিআই ব্যবহার করে নতুন ডিভাইসে ডাটা রিস্টোর করার বিষয়ে আলোচনা করবে।

অ্যান্ড্রয়েড ব্যাকআপ সার্ভিসের জন্য রেজিস্টার

এই অনুশীলনী Android Backup Service (<https://developer.android.com/google/backup/index.html?csw=1>) এর ব্যবহার চায় যার জন্য প্রয়োজন রেজিট্রেশন। সামনে এগিয়ে যান এবং এখানে (<https://developer.android.com/google/backup/signup.html?csw=1>) রেজিস্টার করুন। যখনই এটা করা হবে, সার্ভিসটি একটি এক্সএমএল ট্যাগ আগে থেকেই রেখে দিবে আপনার অ্যান্ড্রয়েড মেনিফেস্টের মধ্যে যুক্ত করতে, যেটা দেখতে অনেকটা এরকম হবে:

```
<meta-data android:name="com.google.android.backup.api_key"
android:value="ABcDe1FGHij2KlmN3oPQRs4TUvW5xYZ" />
```

উল্লেখ্য প্রতিটা ব্যাকআপ কি একটি স্পেসিফিক প্যাকেজ নামের সাথে কাজ করে। যদি আপনার ভিন্ন ভিন্ন অ্যাপলিকেশন থাকে, প্রতিটার জন্য পৃথক কি রেজিস্টার করতে হবে।

আপনার মেনিফেস্ট কনফিগার করুন

অ্যান্ড্রয়েড ব্যাকআপ সার্ভিস ব্যবহার করার জন্য আপনার অ্যাপলিকেশন মেনিফেস্টে দুইটা এডিশন প্রয়োজন। প্রথমত, ক্লাসের নাম ডিক্লেয়ার করুন যা আপনার ব্যাকআপ এজেন্ট হিসাবে কাজ করে, এবং তারপর উপরের চিত্রটি অ্যাপলিকেশন ট্রাগের একটি চাইল্ড এলিমেন্ট হিসাবে যুক্ত করুন। আপনার ব্যাকআপ এজেন্টকে TheBackupAgent হিসাবে আখ্যায়িত করা হবে, এখানে একটি উদাহরণ আছে এই ট্যাগ অন্তর্ভুক্ত করে মেনিফেস্ট দেখতে কেমন হবে:

```
<application android:label="MyApp"
              android:backupAgent="TheBackupAgent">

    ...
    <meta-data android:name="com.google.android.backup.api_key"
              android:value="ABcDe1FGHiJ2KlMn3oPQRs4TUvW5xYZ" />
    ...
</application>
```


আপনার ব্যাকআপ এজেন্ট লিখুন (রাইট)

আপনার ব্যাকআপ এজেন্ট তৈরী করার সহজ উপায় হচ্ছে র‍্যাপার ক্লাস BackupAgentHelper প্রসারিত করা। এই সাহায্যকারী ক্লাস তৈরী করাটা একটি সহজ প্রক্রিয়া। পূর্ববর্তী ধাপের মেনিফেস্টে (এই উদাহরনে, TheBackupAgent) ব্যবহার করার মতো করে একটি নাম দিয়ে শুধু ক্লাস তৈরী করুন, এবং BackupAgentHelper প্রসারিত করুন। তারপর onCreate()ওভাররাইড করুন।

onCreate() পদ্ধতির মধ্যে একটি BackupHelper তৈরী করুন। নির্দিষ্ট কিছু ডাটার ব্যাকআপ করতে এই হেল্পারসগুলো বিশেষ ক্লাস। অ্যান্ড্রয়েড ফ্রেমওয়ার্ক বর্তমানে এই রকম দুইটা ক্লাস অন্তর্ভুক্ত করে: BackupAgentHelper এবং SharedPreferencesBackupHelper । আপনি হেল্পার তৈরী করার পর এবং আপনি যে ডাটা ব্যাকআপ করতে চান তার প্রতি নির্দেশ করার পর, addHelper() পদ্ধতি ব্যবহার করে শুধু এটাকে ব্যাকআপ এজেন্ট হেল্পারে যুক্ত করে যুক্ত করে দিন, একটি কি যুক্ত করে দিন যা পরবর্তীতে ডাটা পূনরুদ্ধার করতে ব্যবহৃত হবে। অধিকাংশ ক্ষেত্রে সম্পূর্ণ বাস্তবায়ন সম্ভবত কোডের ১০ লাইন।

এখানে একটি উদাহরণ আছে যা একটি হাই স্কোর ফাইলকে ব্যাকআপ করে:

```
import android.app.backup.BackupAgentHelper;
import android.app.backup.FileBackupHelper;

public class TheBackupAgent extends BackupAgentHelper {
    // The name of the SharedPreferences file
    static final String HIGH_SCORES_FILENAME = "scores";

    // A key to uniquely identify the set of backup data
    static final String FILES_BACKUP_KEY = "myfiles";

    // Allocate a helper and add it to the backup agent
    @Override
    void onCreate() {
        FileBackupHelper helper = new FileBackupHelper(this, HIGH_SCORES_FILENAME);
        addHelper(FILES_BACKUP_KEY, helper);
    }
}
```

নমনীয়তা যুক্ত করার জন্য, FileBackupHelper এর কনস্ট্রাক্টর ফাইলনেমের একটি ভেরিয়েবল নাম্বার নিতে পারে। আপনি শুধু একটি হাই স্কোর ফাইল এবং একটি গেম প্রগ্রেস ফাইল সহজে ব্যাকআপ রাখতে নীচের মতো করে শুধু একটি অতিরিক্ত প্যারামিটার যুক্ত করে দিতে পারেন:

```
@Override
void onCreate() {
    FileBackupHelper helper = new FileBackupHelper(this, HIGH_SCORES_FILENAME, PROGRESS_FILENAME);
    addHelper(FILES_BACKUP_KEY, helper);
}
```

প্রিফারেন্সের ব্যাকআপ করা একইভাবে সহজ। একটি FileBackupHelper আপনি যেভাবে করেছেন

ঠিক সেভাবেই একটি SharedPreferencesBackupHelper তৈরী করুন। এই ক্ষেত্রে, কনস্ট্রাকটরে ফাইলনেম যুক্ত করার বদলে আপনার অ্যাপলিকেশন দ্বারা ব্যবহৃত শেয়ার প্রিফারেন্স গ্রুপের নাম যুক্ত করুন। এখানে যদি একটি ফ্লাট ফাইলের বদলে হাই স্কোর প্রিফারেন্স বাস্তবায়ন করে তাহলে আপনার ব্যাকআপ এজেন্ট হেল্পার দেখতে কেমন হবে তার একটি উদাহরণ আছে:

```
import android.app.backup.BackupAgentHelper;
import android.app.backup.SharedPreferencesBackupHelper;

public class TheBackupAgent extends BackupAgentHelper {
    // The names of the SharedPreferences groups that the application maintains. These
    // are the same strings that are passed to getSharedPreferences(String, int).
    static final String PREFS_DISPLAY = "displayprefs";
    static final String PREFS_SCORES = "highscores";

    // An arbitrary string used within the BackupAgentHelper implementation to
    // identify the SharedPreferencesBackupHelper's data.
    static final String MY_PREFS_BACKUP_KEY = "myprefs";

    // Simply allocate a helper and install it
    void onCreate() {
        SharedPreferencesBackupHelper helper =
            new SharedPreferencesBackupHelper(this, PREFS_DISPLAY, PREFS_SCORES);
        addHelper(MY_PREFS_BACKUP_KEY, helper);
    }
}
```

আপনার যত ইচ্ছা ততগুলো ব্যাকআপ হেল্পার ইনস্টেন্স আপনার ব্যাকআপ এজেন্ট হেল্পারে যুক্ত করাতে পারবেন, কিনুত মনে রাখবেন যে প্রতিটার মধ্যে মাত্র একটি আপনার প্রয়োজন হবে। একটি FileBackupHelper সকল ফাইলকে চালিত করতে পারে যা আপনার ব্যাকআপের প্রয়োজন, এবং একটি SharedPreferencesBackupHelper সকল শেয়ার প্রিফারেন্স গ্রুপকে চালিত করে যা আপনার ব্যাকআপ করা প্রয়োজন।

ব্যাকআপ রিকোয়েস্ট

একটি ব্যাকআপ রিকোয়েস্ট করার জন্য, শুধু BackupManager এর একটি ইনসটেন্স তৈরী করুন, এবং এর dataChanged() পদ্ধতি কল করুন।

```
import android.app.backup.BackupManager;
...

public void requestBackup() {
    BackupManager bm = new BackupManager(this);
    bm.dataChanged();
}
```

এই কল ব্যাকআপ ম্যানেজারকে জানায় যে ক্লাউডে ব্যাকআপ হওয়ার জন্য সেখানে একটি ডাটা আছে। ভবিষ্যতে কিছু বিষয়ে কলব্যাক ম্যানেজার তখন আপনার ব্যাকআপ এজেন্টের onBackup() পদ্ধতিকে কল করে। অতিরিক্ত নেটওয়ার্ক একটিভিটির ঘটার কারনে চিন্তিত না হয়ে যখনই আপনার ডাটা পরিবর্তন হবে আপনি কলটি করতে পারেন। আপনি যদি একটি ব্যাকআপ ঘটার পূর্বে একটি ব্যাকআপকে দুইবার রিকোয়েস্ট করেন, তাহলে ব্যাকআপটি একবারই ঘটবে।

একটি ব্যাকআপ থেকে রিস্টোর করুন

সাধারণভাবে আপনাকে কখনই ম্যানুয়ালি একটি রিস্টোরকে রিকোয়েস্ট করতে হবে না, যেহেতু এটা স্বয়ংক্রিয়ভাবে হবে যখন আপনার অ্যাপলিকেশন একটি ডিভাইসে ইনসটল করা হবে। কিন্ত যদি একটি ম্যানুয়েল রিস্টোর সক্রিয় করার প্রয়োজন হয়, শুধু `requestRestore()` পদ্ধতিকে কল করুন।

অধিকাংশ গুগল ক্লাউড মেসেজিং তৈরী করা

(<http://developer.android.com/training/cloudsync/gcm.html>)

অধিকাংশ গুগল ক্লাউড মেসেজিং (GCM) অ্যান্ড্রয়েড ডিভাইসে বিনা মূল্যে মেসেজ সেন্ড করার একটি সার্ভিস। GCM মেসেজিং ভালোভাবে ইউজার এক্সপেরিয়েন্সকে বৃদ্ধি করতে পারে। রেডিওকে জাগাতে কোন ব্যাটারি পাওয়ার খরচ না করেই এবং যখন সেখানে কোন আপডেট না থাকে তখন সার্ভাও পোল করে আপনি আপনার অ্যাপলিকেশন সময়পোযোগী করতে পারেন। এছাড়াও GCM আপনাকে একটি একক মেসেজে ১০০০ পর্যন্ত রিসিপিয়েন্ট যুক্ত করতে দেয়, আপনাকে বৃহৎ বেজে দ্রুত যোগাযোগ করতে দেয় যখন প্রয়োজন হয়, যখন আপনার সার্ভারে কাজের চাপ কমিয়ে আনে।

এই অনুশীলনী আপনার অ্যাপলিকেশনের মধ্যে GCM একীভূত করার কিছু ভালো চর্চা নিয়ে আলোচনা করে এবং মনে করে আপনি ইতিমধ্যে এই সার্ভিসের মৌলিক বাস্তবায়নের সাথে পরিচিত। যদি এক্ষেত্রে তা না হয় আপনি GCM demo app tutorial (<http://developer.android.com/google/gcm/demo.html>) পড়তে পারেন।

কার্যকরীভাবে মাল্টিকাস্ট মেসেজ সেন্ড (পাঠানো) করুন

GCM এর একটি কার্যকরী বৈশিষ্ট্য হচ্ছে একটি একক মেসেজের জন্য ১০০০ রিসিপিয়েন্ট (গ্রাহক) পর্যন্ত সাপোর্ট করে। এই সামর্থ্য আপনার পূর্ণাঙ্গ ইউজার বেজে গুরুত্বপূর্ণ মেসেজ পাঠানোর কাজটি সহজ করে দিয়েছে। উদাহরণস্বরূপ, মনে করুন আপনার ১,০০০,০০০ ইউজারকে একটি মেসেজ পাঠাতে হবে এবং আপনার সার্ভাও প্রতি সেকেন্ডে ৫০০ পাঠাতে পারে। আপনি যদি প্রতি মেসেজ মাত্র একটি একক রিসিপিয়েন্ট দিয়ে পাঠান, সেক্ষেত্রে সময় লাগবে $1,000,000/500 = 2,000$ সেকেন্ড বা প্রায় আধা ঘন্টা। কিন্তু প্রতি মেসেজে যদি ১০০০ রিসিপিয়েন্ট যুক্ত (অ্যাটাচ) করা যায় তাহলে $1,000,000$ জনকে একটি মেসেজ পাঠাতে মোট সময় কমে গিয়ে হবে $(1,000,000/1000)/500 = 2$ সেকেন্ড। এটা শুধু কার্যকরই নয়, যথাসময়ে ডাটা পাঠানোর জন্য গুরুত্বপূর্ণ, যেমন প্রাকৃতিক বিপর্যয়ের সতর্কতা বা খেলাধুরার স্কোর যেখানে ৩০ মিনিটের বিরতি তথ্যকে অপ্রয়োজনীয় করে ফেলতে পারে।

এই কর্মপদ্ধতির সুবিধা নিয়া সহজ। আপনি যদি Java'র জন্য GCM helper library (<http://developer.android.com/google/gcm/gs.html#libs>) ব্যবহার করে থাকেন, তাহলে একটি একক রেজিস্ট্রেশন আইডি এর পরিবর্তে send বা sendNoRetry পদ্ধতিতে শুধু রেজিস্ট্রেশন আইডির একটি খরঃ সংগ্রহ সরবরাহ করুন।

```
// This method name is completely fabricated, but you get the idea.
List regIds = whoShouldISendThisTo(message);

// If you want the SDK to automatically retry a certain number of times, use the
// standard send method.
MulticastResult result = sender.send(message, regIds, 5);

// Otherwise, use sendNoRetry.
MulticastResult result = sender.sendNoRetry(message, regIds);
```

Java'র বদলে একটি ল্যাম্বুয়েজের মধ্যে ওই GCM সাপোর্ট বাস্তবায়ন করার জন্য, নিম্নোক্ত হেডারগুলো দিয়ে একটি HTTP POST রিকোয়েস্ট তৈরী করুন:

- Authorization: key=YOUR_API_KEY
- Content-type: application/json

তারপর যে প্যারামিটার আপনি একটি অবজেক্টের মধ্যে চান তা এনকোড করুন, সকল রেজিস্ট্রেশন আইডি কি'র registration ids অধীনে তালিকা করা। নিম্নোক্ত কোড চিত্রটি উদাহরণ হিসাবে দেয়া আছে। registration ids ব্যতিত আর সকল প্যারামিটার ঐচ্ছিক, এবং data এর মধ্যে থাকা আইটেমগুলো ইউজার নির্ধারিত পেলোড কে প্রতিনিধিত্ব করে, GCM নির্ধারিত প্যারামিটার নয়। এই HTTP POST মেসেজের জন্য শেষ বিন্দু হবে <https://android.googleapis.com/gcm/send>।

```
{ "collapse_key": "score_update",  
  "time_to_live": 108,  
  "delay_while_idle": true,  
  "data": {  
    "score": "4 x 8",  
    "time": "15:16.2342"  
  },  
  "registration_ids":["4", "8", "15", "16", "23", "42"]  
}
```

মাল্টিকাস্ট GCM মেসেজের ফরমেট সম্পর্কে আরও বিস্তারিত জানতে GCM গাইডের অধ্যায়
Sending Messages (<http://developer.android.com/google/gcm/gcm.html#send-msg>) দেখুন।

যে মেসেজটি প্রতিস্থাপিত হতে পারে সেটা বন্ধ করে দিন

GCM মেসেজ কখনও কখনও টিকল হতে পারে, ফ্রেশ ডাটার জন্য সার্ভারের সাথে যোগাযোগ করতে মোবাইল অ্যাপলিকেশনকে বলা। GCM এ এই পরিস্থিতির জন্য বন্ধ হতে সক্ষম মেসেজে তৈরী করা সম্ভব, যার মধ্যে পুরাতনের স্থানে নতুন মেসেজ প্রতিস্থাপিত হয়। এখানে একটি খেলার স্কোরের উদাহরণ দেয়া হলো। যদি আপনি আপডেট স্কোর সহকারে একটি নির্দিষ্ট খেলা অনুসারে সকল ইউজারকে একটি মেসেজ পাঠান, এবং ১৫ মিনিট পরে স্কোরের আপডেট মেসেজ যায়, প্রথম মেসেজটির আর গুরুত্ব থাকবে না। যে কোন ইউজারের জন্য যে এখনও প্রথম মেসেজটি গ্রহণ করে নাই, তাকে দুইটা মেসেজই না পাঠানোর কোন কারন নেই এবং ডিভাইসকে দুইবার প্রতিক্রিয়া করতে (এবং ইউজারকে সম্ভাব্য সতর্ক করা) বাধ্য করা যখন মেসেজের মাত্র একটিই তখনও গুরুত্বপূর্ণ।

যখন আপনি একটি বন্ধ কি নির্ধারণ করবেন, যখন একই ইউজারের জন্য GCM সার্ভারের মধ্যে মাল্টিপল মেসেজকে সারিবদ্ধ করে, শুধুমাত্র শেষেরটা সাথে যে কোন প্রদত্ত কলাপস কি ডেলিভারি করা হয়। খেলার স্কোরের মতো অবস্থার জন্য, এটা ডিভাইসকে অপ্রয়োজনীয় কাজ করা এবং সম্ভাব্য ওভার নোটিফাই করা থেকে রক্ষা করে। এই অবস্থায় এটা একটি সার্ভার সিস্টেমকে সম্পৃক্ত করে (মেইল চেক করার মতো), এটা ডিভাইসকে করতে হবে এমন সিস্টেমের সংখ্যা কমিয়ে আনতে পারে। উদাহরণ হিসাবে, যদি সার্ভারে ১০টি ইমেইল অপেক্ষা করে থাকে এবং দশটি "নতুন ইমেইল" GCM টিকলস পাঠানো হয়ে থাকে, এটার একটাই প্রয়োজন, যেহেতু এটা একবার সিস্টেম করে।

এই বৈশিষ্ট্য ব্যবহার করার জন্য, আপনার বহিরাগমন মেসেজে একটি কলাপস কি যুক্ত করুন। আপনি যদি GCM হেল্পার লাইব্রেরী ব্যবহার করতে থাকেন, মেসেজ ক্লাসের `collapseKey(String key)` পদ্ধতি ব্যবহার করুন।

```
Message message = new Message.Builder(regId)
    .collapseKey("game4_scores") // The key for game 4.
    .ttl(600) // Time in seconds to keep message queued if device offline.
    .delayWhileIdle(true) // Wait for device to become active before sending.
    .addPayload("key1", "value1")
    .addPayload("key2", "value2")
    .build();
```

যদি হেল্পার লাইব্রেরী ব্যবহার না করে থাকেন, শুধু আপনি তৈরী করছেন এমন POST হেডারে একটি ভেরিয়েবল যুক্ত করুন, সাথে ফিল্ডনেম হিসাবে `collapse_key`, এবং ভ্যালু হিসাবে ওই আপডেটের সেট জন্য ব্যবহার করা স্ট্রিং।

সরাসরি GCM মেসেজের মধ্যে ডাটা বসানো

কখনও, GCM মেসেজ একটি টিকল হওয়াকে বোঝায়, বা ডিভাইসে নির্দেশনা যে সার্ভারের কোথাও একটি ফ্রেশ ডাটা অপেক্ষা করছে। কিন্ত একটি GCM মেসেজ 4kb সাইজ পর্যন্ত হতে পারে তাই কখনও কখনও এটা শুধু GCM মেসেজের মধ্যে ডাটা সেল্ড করে নিজেই অর্থপূর্ণ হয়, যাতে ডিভাইসকে মোটেই সার্ভারের সাথে যোগাযোগ করার প্রয়োজন না হয়। যে অবস্থায় নীচের বিবতিগুলি সঠিক সেই অবস্থায় এই পদ্ধতি বিবেচনা করুন:

- 4kb সীমার মধ্যে সকল ডাটা ফিট হবে
- প্রতিটা মেসেজই গুরুত্বপূর্ণ এবং সংরক্ষণ করা উচিত
- একটি একক "সার্ভারে নতুন ডাটা"টিকল এর মধ্যে মাল্টিপল GCM মেসেজ কলাপস করানো কোন মানে তৈরী করে না।

উদাহরণস্বরূপ, একটি টার্ন-বেজড নেটওয়ার্ক গেমের মধ্যে শর্ট মেসেজ বা এনকোডেড প্লেয়ার হচ্ছে একটি GCM মেসেজের মধ্যে সরাসরি ডাটা বসানোর জন্য ভালো ইউজ-কেসের উদাহরণ। ইমেইল হচ্ছে খারাপ ইউজ-কেসের একটি উদাহরণ, সেহেতু মেসেজ কখনও কখনও 4kb এর চাইতে বড় হতে পারে, এবং সার্ভারে তাদের জন্য অপেক্ষা করা ইমেইলের প্রতিটার জন্য ইউজারের একটি GCM মেসেজের প্রয়োজজন নেই।

আপনি এই পদ্ধতিও বিবেচনা করতে পারেন যখন মাল্টিকাস্ট মেসেজ সেল্ড করা হয়, সুতরাং আপনাকে ইউজার বেজ জুড়ে প্রতিটা ডিভাইসকে তাৎক্ষনিকভাবে আপডেটের জন্য আপনার সার্ভারকে হিট করতে বলতে হবে না। এই কৌশল অধিক পরিমানে ডাটা সেল্ড করার জন্য কয়েকটি কারনে উপযুক্ত নয়:

- মেসেজ সহকারে একটি একক ডিভাইসকে স্প্যামিং করা থেকে বাজে বা দুর্বল কোড করা অ্যাপকে বিরত রাখতে রেট সীমা নির্দিষ্ট স্থানে আছে।
- মেসেজ নির্দিষ্ট ক্রমানুসারে পৌছাবে তার কোন নিশ্চয়তা নেই
- আপনি যত দ্রুত মেসেজ বাইরে পাঠিয়েছেন তা সেভাবেই পৌছাবে তার নিশ্চয়তা নেই। এমন কি যদি ডিভাইস একটি GCM মেসেজ এক সেকেন্ডে গ্রহণ করে, ম্যাক্স. 1k এ যা হচ্ছে 8kbps অথবা ১৯৯০'র দশকের প্রথম দিকে হোম ডায়াল-আপ ইন্টারনেটের গতি সম্পর্কিত। গুগল প্লেতে আপনার অ্যাপ রেটিং ইউজারের কাছে এইসব কাজগুলোকে প্রতিফলিত করবে। যখন যথাযথভাবে ব্যবহৃত হয়, সরাসরি GCM মেসেজে স্থাপিত ডাটা আপনার অ্যাপলিকেশনের অর্জিত গতিকে আরও বৃদ্ধি করতে পারে, এটাকে সার্ভারের দিকে রাউন্ড ট্রিপ কে পরিহার করতে দিয়ে।

GCM মেসেজের প্রতি Intelligently রিয়েক্ট করা

আপনার অ্যাপলিকেশনের শুধুমাত্র ইনকামিং GCM মেসেজের প্রতি রিয়েক্ট করা উচিত নয়, *intelligently* ও রিয়েক্ট করা উচিত। কীভাবে রিয়েক্ট করতে হয় তা নির্ভর করে এর প্রেক্ষাপটের উপর।

বিরক্তিকর করবেন না

যখন এটা আপনার ইউজারকে ফ্রেশ ডাটার সতর্কতা নিয়ে আসে, "উপকারী"("useful") অবস্থা থেকে "বিরক্তিকর" ("annoying") চলে আসাটা সহজ। যদি আপনার অ্যাপলিকেশন স্ট্যাটাসবার নোটিফিকেশন ব্যবহার করে থাকেন, দ্বিতীয়টা তৈরী না করে আপনার যেটা আছে সেটাকেই আপডেট করুন (<http://developer.android.com/guide/topics/ui/notifiers/notifications.html#Updating>)। আপনি যদি ইউজার কে সতর্ক করতে বিপ বা ভাইব্রেট করতে চান তাহলে একটি টাইমার সেট করার কথাটি বিবেচনা করতে পারেন। অ্যাপলিকেশনকে এক মিনিটে একবারের বেশী সতর্ক করতে দিবেন না, এটা ইউজারকে আপনার অ্যাপলিকেশনকে আন-ইনস্টল করতে, ডিভাইস বন্ধ করতে বা ডিভাইস ছুড়ে ফেলে দিতে প্ররোচিত করতে পারে। বিরক্তিকর হবেন না

স্মার্টভাবে সিঙ্ক করুন কঠিন ভাবে নয়

যখন ডিভাইসে একটি ইনডিকেটর (মানদণ্ড) হিসাবে GCM ব্যবহার করা যা সার্ভার থেকে ডাউনলোড করা প্রয়োজন, এ বিষয়ে আপনার অ্যাপলিকেশনকে স্মার্ট হতে সহায়তা করতে মনে রাখবেন আপনার 4kb মেটাডাটা আছে। উদাহরণস্বরূপ, যদি আপনার একটি ফিড রিডিং অ্যাপ থাকে এবং আপনার ইউজারের ১০০ ফিডস আছে যা তারা অনুসরণ করে, ডিভাইস সার্ভার থেকে যা ডাউনলোড করে সে সম্পর্কে এটাকে স্মার্ট হতে সহায়তা করে! নীচের উদাহরণটি দেখুন পেলোডের মধ্যে আপনার অ্যাপলিকেশনে কোন মেটাডেটা পাঠানো হয়েছে এবং কীভাবে অ্যাপলিকেশন প্রতিক্রিয়া করতে পারে: স্মার্টভাবে সিঙ্ক করুন কঠিন ভাবে নয়

- `refresh` — আপনার অ্যাপলিকেশনকে মূলত এটার অনুসরণকৃত প্রতিটা ফিডের একটি ডাম্প কে রিকোয়েস্ট করতে বলা হয়। আপনার অ্যাপের হয় ১০০ ভিন্ন সার্ভারে ফিড রিকোয়েস্ট পাঠানোর প্রয়োজন, নয়তো যদি আপনার সার্ভারে আপনার একটি সংযোগকারী থাকে, পুনরুদ্ধার করতে একটি রিকোয়েস্ট পাঠান, ১০০ ভিন্ন ফিড থেকে সাম্প্রতিক ডাটা বাবুদল এবং স্থানান্তরিত করা, প্রতিটা সময় একটি আপডেট।
- `refresh, feedID` — Better: আপনার অ্যাপ আপডেটের জন্য একটি নির্দিষ্ট ফিড কে চেক করতে জানে।
- `refresh, feedID, timestamp` — Best: যদি ইউজার GCM মেসেজ পৌছার পূর্বেই ম্যানুয়েলি রিফ্রেশ ঘটায়, অ্যাপলিকেশন অতি সাম্প্রতিক পোস্টের টাইমস্ট্যাম্প তুলনা করতে পারে এবং নির্দারন করতে পারে যে এটার কোন কিছু করা প্রয়োজন নেই।

ক্লাউডে সেভ করা সম্পর্কিত সাংঘর্ষিক বিষয়ের সমাধান করা

(<http://developer.android.com/training/cloudsave/conflict-res.html>)

এই আর্টিকেলটি বর্ণনা করবে কীভাবে অ্যাপের জন্য একটি রোবাস্ট কনফ্লিক্ট রেজোল্যুশন কৌশল ডিজাইন করতে হয় যা Cloud Save service (<https://developers.google.com/games/services/common/concepts/cloudsave>) ব্যবহার করে ক্লাউডে ডাটা সেভ করে। ক্লাউড সেভ সার্ভিস আপনাকে গুগলের সার্ভিসে একটি অ্যাপলিকেশনের প্রতিটা ইউজারের জন্য অ্যাপলিকেশন ডাটা সেভ করতে দেয়। আপনার অ্যাপলিকেশন ক্লাউড সেভ APIs ব্যবহার করে অ্যান্ড্রয়েড সার্ভিস, iOS ডিভাইস, বা ওয়েব অ্যাপলিকেশন থেকে এই ইউজার ডাটা পুনরুদ্ধার এবং আপডেট করতে পারে।

ক্লাউড সেভে সেভ এবং লোড করার প্রক্রিয়া খুবই সহজ: এটা বাইট বিন্যাস থেকে এবং বাইট বিন্যাসের প্রতি প্লেয়ারের ডাটা ধারাবাহিকভাবে ক্রমানুসারে সাজানোর বিষয় এবং ঐ বিন্যাসটি ক্লাউডে স্টোর করা। কিন্ত, যখন আপনার ইউজারের মাল্টিপল (বহুমুখি) ডিভাইস থাকে এবং এর মধ্যে দুই বা ততোধিক ক্লাউডে ডাটা সেভ করার প্রয়াস নেয়, সেভ করার করাটা সাংঘর্ষিক হতে পারে, এবং আপনাকে অবশ্যই ঠিক করতে হবে কীভাবে এর সমাধান করা যায়। আপনার ক্লাউড সেভ ডাটার গঠন ব্যাপকভাবে দেখায় আপনার কনফ্লিক্ট রেজোল্যুশন কতটা শক্তিশালী হতে পারে তাই আপনাকে অবশ্যই আপনার কনফ্লিক্ট রেজোল্যুশন লজিককে প্রতিটা কেস সঠিকভাবে চালিত করতে দেয়ার জন্য সতর্কতার সাথে আপনার ডাটা ডিজাইন করতে হবে।

আর্টিকেলটি শুরু করে কিছু ত্রুটিপূর্ণ পদ্ধতি আলোচনা দিয়ে এবং ব্যাখ্যা করে কোথায় তাদের ঘাটতি আছে। তারপর এটা সংঘর্ষ বন্ধ করতে একটি সমাধান উপস্থাপন করে। এই আলোচনায় গেমকে ফোকাস করা হয়েছে, কিন্ত আপনি একই নিয়ম অন্য অ্যাপে প্রয়োগ করতে পারেন যা ক্লাউডে ডাটা সেভ করে।

কনফ্লিক্ট সম্পর্কে অবগত হতে

OnStateLoadedListener পদ্ধতি গুগলের সার্ভার থেকে একটি অ্যাপলিকেশনের স্টেট ডাটা লোড করার জন্য দায়িত্বপ্রাপ্ত। কলব্যাক OnStateLoadedListener.onStateConflict ইউজারের ডিভাইসে লোকাল স্টেট এবং ক্লাউডের মধ্যে স্টোর করা স্টেটের মধ্যকার সাংঘর্ষিক বিষয়গুলো সমাধান করতে আপনার অ্যাপলিকেশনের জন্য একটি মেকানিজম সরবরাহ করে:

```
@Override
public void onStateConflict(int stateKey, String resolvedVersion,
    byte[] localData, byte[] serverData) {
    // resolve conflict, then call mAppStateClient.resolveConflict()
    ...
}
```

এই ক্ষেত্রে আপনার অ্যাপলিকেশনকে অবশ্যই ঠিক করতে হবে কোন ডাটা সেটটাকে রাখা উচিত, অথবা এটা একটা নতুন ডাটা সেট জমা দিতে পারে যা একত্রিত (মার্জ) করা ডাটাকে প্রতিনিধিত্ব করে। সাংঘর্ষিক রেজুলেশন লজিক বাস্তবায়ন করা আপনার উপর নির্ভর করে।

এটা অনুধাবন করা জরুরী যে ক্লাউড সেভ সার্ভিস ব্যাকগ্রাউন্ডের মধ্যে ডাটা সিঙ্ক্রনাইজ করে। অতএব, আপনার নিশ্চিত করা উচিত যে আপনার অ্যাপ ঐ কলব্যাক কনটেক্সটের বাইরে রিসিভ করার জন্য প্রসূত আছে যেখানে আপনি শুরুতে ডাটা তৈরী করেছিলেন। বিশেষ করে যদি গুগল প্লে সার্ভিস অ্যাপলিকেশন ব্যাকগ্রাউন্ডে একটি সাংঘর্ষিক বিষয় সনাক্ত করে, কলব্যাক পরবর্তী সময়ে ডাটা লোড করার চেষ্টা করার সময় কল করে, যা ততক্ষণ পর্যন্ত হবে না যতক্ষণ পরবর্তী সময়ে ইউজার অ্যাপ শুরু না করে।

অতএব, ক্লাউড সেভ ডাটা ডিজাইন করুন এবং কনফ্লিক্ট রেজুলেশন কোড অবশ্যই হবে (*context-independent*): প্রদত্ত দুইটা কনফ্লিক্টিং সেভ স্টেট, ডাটা সেটের মধ্যে আছে শুধুমাত্র এমন ডাটা ব্যবহার করে আপনাকে অবশ্যই সংঘর্ষ সমাধান করতে পারতে হবে, বহিরাগত কোন কনটেক্সটের সাথে পরামর্শ না করেই।

সাধারণ কেসগুলো পরিচালিত করা

এখানে কনফ্লিক্ট (সংঘাত) রেজ্যুলেশনের কিছু সাধারণ কেস দেয়া হলো। অনেক অ্যাপের জন্য, এই সকল কৌশলের কোন একটির বিকল্প গ্রহণ করা যথেষ্ট:

- পুরাতনের চেয়ে নতুন ভালো। কিছু ক্ষেত্রে, পুরাতন ডাটার স্থলে নতুন ডাটা প্রতিস্থাপন হওয়া জরুরী। উদাহরণস্বরূপ, যদি ডাটা একটি চরিত্রের শার্টের রঙের প্লেয়ারের পছন্দের বিষয়টার প্রতিনিধিত্ব করে, তখন চলতি পছন্দটি পুরাতন পছন্দকে খারিজ করে দেওয়া উচিত। এক্ষেত্রে আপনি ক্লাউড সেভ ডাটাতে টাইমস্ট্যাম্প সেট করার বিষয়টা বেছে নিতে পারেন। যখন কনফ্লিক্ট (সংঘাত) নিরসন হবে তখন অতি সাম্প্রতিক টাইমস্ট্যাম্প দিয়ে ডাটা সেট বেছে নিতে হবে (একটি গ্রহণযোগ্য ক্লক ব্যবহার করার বিষয়টা স্বরণ রাখবেন এবং টাইম জোনের ভিন্নতার বিষয়ে সতর্ক থাকবেন)।
- ডাটার একটি সেট পরিষ্কারভাবে আরেকটির চেয়ে ভালো। অন্য ক্ষেত্রে, কোন ডাটা সর্বোৎকৃষ্ট হিসাবে নির্ধারণ হতে পারে সে বিষয়ে এটা সব সময় পরিষ্কার। উদাহরণস্বরূপ, যদি ডাটা একটি রেসিং গেমের মধ্যে প্লেয়ারের বেস্ট টাইমকে প্রতিনিধিত্ব করে, তখন এটা পরিষ্কার যে, কনফ্লিক্টের (সংঘাতের) ক্ষেত্রে, আপনার উচিত বেস্ট টাইম (ক্ষুদ্রতর) ধারণ করা।
- ইউনিয়ন দ্বারা একত্রিত করা (মার্জ করা)। দুইটা সাংঘর্ষিক সেটের একটি সমন্বয় (ইউনিয়ন) গণনা করার মাধ্যমে সাংঘর্ষিক অবস্থার সমাধান কার সম্ভব হতে পারে। উদাহরণস্বরূপ, যদি আপনার ডাটা লেভেলের সেট প্রতিনিধিত্ব করে যা প্লেয়ার আনলক করে, তখন সমাধান করা ডাটা সাধারণভাবে দুইটা কনফ্লিক্ট সেটের ইউনিয়ন হয়। এই উপায়ে, প্লেয়াররা যে লেভেলগুলো আনলক করেছে তা হারিয়ে ফেলবে না। CollectAllTheStars নমুনা গেমটি এই কৌশলের একটি ভেরিয়েন্ট (বিকল্প) ব্যবহার করে।

আরও জটিল কেসের জন্য একটি কৌশল ডিজাইন করা

আরও জটিল কেস ঘটে যখন আপনার গেম প্লেয়ারকে পরিবর্তনশীল (ফাঞ্জিবল) আইটেম বা ইউনিট সংগ্রহ করতে দেয় যেমন সোনার কয়েন (মুদ্রা) বা এক্সপেরিয়েন্স পয়েন্ট। একটি অনুমেয় (হাইপোথেটিক্যাল) গেম বিবেচনা করা যাক, যার নাম কয়েন রান, একটি অসিম রানার যেখানে লক্ষ হচ্ছে কয়েন সংগ্রহ করা এবং খুব ধনি হওয়া। প্রতিটা হয়েন সংগ্রহ প্লেয়ারের পিগি ব্যাল্কে জমা হবে।

নিচের অধ্যায় মাল্টিপল ডিভাইসের মধ্যে সিক্স কনফ্লিক্ট সমাধানের জন্য তিনটা কৌশল আলোচনা করে: তার মধ্যে দুইটা ভালোই কিনুত চূড়ান্তভাবে সফলতার সাথে সকল পরিস্থিতি সমাধান করতে ব্যর্থ হয় এবং একটি চূড়ান্ত সমাধান যা যে কোন ডিভাইসের সংখ্যার মধ্যকার সংঘাত (কনফ্লিক্ট) ব্যবস্থাপনা করতে পারে।

প্রথম প্রচেষ্টা: শুধু সর্বমোটটি (টোটাল) স্টোর করুন

প্রথম চিন্তাতে, এটা মনে হতে পারে যে ক্লাউড সেভ ডাটাকে শুধুমাত্র ব্যাঙ্কে কয়েনের সংখ্যা হওয়া উচিত। কিন্ত যদি যাটা যেটুকু বিদ্যমান আছে সেটুকুই হয়, কনফ্লিক্ট রেজুলেশন জোড়ালোভাবে সীমিত হবে। সবচেয়ে ভালো হচ্ছে একটি কনফ্লিক্টের ক্ষেত্রে দুইটা সংখ্যার বড়টা বেছে নেয়া।

টেবিল ১. এ ব্যাখ্যা করা চিত্রটি বিবেচনা করুন। হতে পারে প্লেয়ারের প্রাথমিকভাবে ২০টি কয়েন আছে এবং তারপর ডিভাইস A থেকে ১০ টি কয়েন এবং ডিভাইস B থেকে ১৫ কয়েন সংগ্রহ করলো। তখন ডিভাইস B ক্লাউডে ডাটা সেট সেভ করে। যখন ডিভাইস A সেভ করার চেষ্টা করে, একটি কনফ্লিক্ট চিহ্নিত হয়। "Store Only the Total"/ “শুধু সর্বমোটটি স্টোর করুন” কনফ্লিক্ট রেজুলেশন এলগরিদম ৩৫(দুইটা সংখ্যার মধ্যে বড়টা) লেখার মাধ্যমে কনফ্লিক্ট সমাধান করতে পারে।

টেবিল ১. শুধু কয়েনের সর্বমোটটি (টোটাল) স্টোর করা (অসফল কৌশল):

Event	Data on Device A	Data on Device B	Data on Cloud	Actual Total
Starting conditions	20	20	20	20
Player collects 10 coins on device A	30	20	20	30
Player collects 15 coins on device B	30	35	20	45
Device B saves state to cloud	30	35	35	45
Device A tries to save state to cloud. Conflict detected.	30	35	35	45
Device A resolves conflict by picking largest of the two numbers.	30	35	35	45

এই কৌশল অসফল হতে পারে- প্লেয়ারের ব্যাঙ্ক ২০ থেকে ৩৫ চলে যেতে পারে, যখন ইউজার আসলেই সর্বমোট ২৫ কয়েন সংগ্রহ করে (ডিভাইস A তে ১০ টি কয়েন এবং ডিভাইস B তে ১৫ কয়েন)। সুতরাং ১০ টি কয়েন হারিয়ে যায়। ক্লাউড সেভের মধ্যে শুধু কয়েনের সর্বমোটটি (টোটাল) স্টোর করা একটি রোবাস্ট কনফ্লিক্ট রেজুলেশন এলগরিদম বাস্তবায়নের জন্য যথেষ্ট নয়।

দ্বিতীয় প্রচেষ্টা: সর্বমোট (টোটাল) এবং ডেল্টা স্টোর করুন

একটি ভিন্ন পদ্ধতি হচ্ছে সেভ ডাটাতে একটি অতিরিক্ত ফিল্ড যুক্ত করুন: কয়েনের সংখ্যা যোগ হবে (ডেল্টা) সর্বশেষ বার থেকে। এই পদ্ধতিতে সেভ ডাটা একটি tuple (T, d) কর্তৃক প্রতিনিধিত্ব হতে পারে যেখানে T হচ্ছে কয়েনের মোট সংখ্যা এবং d হচ্ছে মাত্রই যুক্ত হওয়া কয়েনের সংখ্যা।

এই কার্ঠামো দিয়ে আপনার কনফ্লিক্ট রেজ্যুলেশন এলগরিদমের আরও বলিষ্ঠ (রোবাস্ট) জায়গা থাকে, যেভাবে নিচে ব্যাখ্যা করা হয়েছে। কিনুত এই পদ্ধতি এখনও আপনার অ্যাপকে প্লেয়ারের সার্বিক অবস্থার গ্রহণযোগ্য চিত্র দিতে পারে না।

এখানে ডাটা অন্তর্ভুক্তকরনের জন্য কনফ্লিক্ট রেজ্যুলেশন এলগরিদম আছে:

- Local data: (T, d)
- Cloud data: (T', d')
- Resolved data: $(T' + d, d)$

উদাহরণস্বরূপ, যখন আপনি লোকাল স্টেট (T, d) এবং ক্লাউড স্টেটের (T', d') মধ্যে একটি কনফ্লিক্ট পান, আপনি এটাকে $(T' + d, d)$ হিসাবে সমাধান করতে পারেন। এটা যা বোঝায় তা হচ্ছে আপনি আপনার লোকাল ডাটা ডেল্টা নিয়েছেন এবং এটাকে ক্লাউড ডাটার মধ্যে একত্রিত করে, আশা করে যে, এটা সঠিকভাবে যে কোন স্বর্ণ কয়েনের জন্য ব্যাখ্যা করবে যা অন্য ডিভাইসে সংগৃহীত হয়েছে।

এই পদ্ধতি আশাপ্রদ লাগতে পারে, কিনুত এটা একটি ডাইনামিক মোবাইল এনভায়রনমেন্টের মধ্যে ভেঙ্গে পড়ে:

- ইউজার স্টেট (অবস্থা) সেভ করতে পারে যখন ডিভাইস অফলাইনে থাকে। এই পরিবর্তন জমা দেয়ার জন্য শ্রেণীবদ্ধ হয় যখন ডিভাইস অনলাইনে ফিরে আসে।
- সিস্টেম যেভাবে কাজ করে সে উপায় হচ্ছে যে সবচেয়ে সমসাময়িক পরিবর্তন যে কোন পূর্ববর্তী পরিবর্তনকে নতুন ভাবে লেখে(ওভাররাইট)। অর্থাৎ দ্বিতীয় রাইট একমাত্র যা ক্লাউডে পাঠানো হয়ে থাকে (এটা ঘটে থাকে যখন ডিভাইস চূড়ান্তভাবে অনলাইনে আসে), এবং প্রথম রাইটের মধ্যকার ডেল্টা অবহেলিত হয়।

ব্যাখ্যা করার জন্য, টেবিল ২. তে ব্যাখ্যা করা চিত্রটি বিবেচনা করুন। ধারাবাহিক অপারেশনগুলোর পরবর্তীতে টেবিলটাতে দেখানো হয়েছে, ক্লাউড স্টেট $(১৩০, +৫)$ হতে পারে। এটার মানে নির্ধারিত স্টেট $(১৪০, +১০)$ হতে পারে। এটা সঠিক নয় কারণ সর্বমোটের মধ্যে ইউজার ডিভাইস A তে ১১০ টি কয়েন এবং ডিভাইস B তে ১২০টি কয়েন সংগ্রহ করেছে। মোট হওয়া উচিত ২৫০ কয়েন।

টেবিল ২, মোট+ডেল্টা কৌশলের জন্য অসফল কেস।

Event	Data on Device A	Data on Device B	Data on Cloud	Actual Total
Starting conditions	(20, x)	(20, x)	(20, x)	20
Player collects 100 coins on device A	(120, +100)	(20, x)	(20, x)	120
Device A saves state to cloud	(120, +100)	(20, x)	(20, x)	120
Player collects 10 more coins on device A	(130, +110)	(20, x)	(120, +100)	130
Player collects 1 coin on device B	(130, +110)	(21, +1)	(120, +100)	131
Device B attempts to save state to cloud. Conflict detected.	(130, +110)	(21, +1)	(120, +100)	131
Device B solves conflict by applying local delta to cloud total.	(130, +110)	(121, +1)	(121, +1)	131
Device A tries to upload its data to the cloud. Conflict detected.	(130, +110)	(121, +1)	(121, +1)	131
Device A resolves the conflict by applying the local delta to the cloud total.	(131, +110)	(121, +1)	(231, +110)	131

(*): x ডাটা পরিবেশন করে যা আমাদের এই প্রেক্ষাপটে অপ্রাসঙ্গিক।

এখন আপনার বিপরীত সমস্যা থাকবে: আপনি প্লেয়ারকে অনেক বেশী কয়েন দিয়ে দিচ্ছেন। এই প্লেয়ার ২১১ কয়েন অর্জন করেছেন, যখন বসুতত সে মাত্র ১১১ কয়েন সংগ্রহ করেছেন।

সমাধান: ডিভাইসে প্রতি সাব-টোটাল স্টোর করুন

পূর্ববর্তী প্রচেষ্টা বিশ্লেষণ করে, মনে হয় যে ওই কৌশল কোন কয়েনটি ইতমধ্যে গণনা করা হয়েছে এবং কোন কয়েনটি এখনও গণনা করা হয় নাই তা জানার যে সামর্থ্য তা মিস করে, বিশেষ করে বিভিন্ন ডিভাইস থেকে আসা বহুমুখি ঘটমানতার উপস্থিতিতে।

সমস্যার সমাধান হচ্ছে একটি ডিকশনারী হওয়া আপনার ক্লাউড সেভের কাঠামো পরিবর্তন করা যা ইনটিজারে স্ট্রিং ম্যাপ করে। এই ডিকশনারীর মধ্যে প্রতিটা কি-ভ্যালু পেয়ার একটি "drawer" পরিবেশন করে যা কয়েন ধারণ করে, এবং সেভের মধ্যে কয়েনের মোট সংখ্যা হচ্ছে সকল এন্ট্রির ভ্যালুর যোগফল। এই ডিজাইনের মূল নীতি হচ্ছে যে প্রতিটা ডিভাইসের তার নিজস্ব ড্রয়ার থাকা, এবং শুধুমাত্র ডিভাইস নিজে ড্রয়ারের মধ্যে কয়েন রাখতে পারবে।

ডিকশনারির কাঠামো হচ্ছে $(A:a, B:b, C:c, \dots)$ যেখানে a হচ্ছে ড্রয়ার A এর মধ্যে রাখা কয়েনের মোট সংখ্যা, b হচ্ছে ড্রয়ার B এর মধ্যে রাখা কয়েনের মোট সংখ্যা এবং এরকম আরও অন্যান্যগুলো।

"drawer" সমাধানের জন্য নতুন কনফ্লিক্ট রেজ্যুলেশন এলগরিদম নিচের মতো:

- Local data: $(A:a, B:b, C:c, \dots)$
- Cloud data: $(A:a', B:b', C:c', \dots)$
- Resolved data: $(A:\max(a,a'), B:\max(b,b'), C:\max(c,c'), \dots)$

উদাহরণস্বরূপ, যদি লোকাল ডাটা $(A:20, B:4, C:7)$ হয় এবং ক্লাউড ডাটা $(B:10, C:2, D:14)$ হয় তখন রিজলভড ডাটা $(A:20, B:10, C:7, D:14)$ হবে। উল্লেখ্য যে আপনি কীভাবে এই ডিকশনারিতে কনফ্লিক্ট রেজ্যুলেশন প্রয়োগ করবেন সেটা আপনার অ্যাপের উপর নির্ভর করে। উদাহরণের জন্য বলা যায় কিছু অ্যাপের জন্য আপনি লোয়ার ভ্যালু নিতে চাইতে পারেন। এই নতুন এলগরিদম পরীক্ষা করতে, উপরে বর্ণিত পরীক্ষামূলক চিত্রগুলোর কোন একটিতে এটা প্রয়োগ করুন। আপনি দেখতে পারবেন যে এটা একটি সঠিক ফলাফলে পৌঁছেছে। টেবিল ৩ এর উপর ভিত্তি করে টেবিল ৪ নিম্নে লিখিত দৃষ্টব্য ব্যাখ্যা করে:

- শুরুতে প্লেয়ারের ২০ কয়েন ছিল। এটা প্রতিটা ডিভাইস এবং ক্লাউডে সঠিকভাবে প্রতিফলিত হয়। এই ভ্যালু একটি ডিকশনারি $(X:20)$ হিসাবে পরিবেশিত হয় যেখানে X এর ভ্যালু গুরুত্বপূর্ণ নয়- এই প্ররঞ্চিক ডাটা কোথা থেকে আসলো সে বিষয়ে আমরা পরোয়া করি না।
- যখন প্লেয়ার ডিভাইস A তে ১০০ কয়েন সংগ্রহ করে, এই পরিবর্তন একটি ডিকশনারি হিসাবে প্যাকেজ হয় এবং ক্লাউডে সেভ হয়। ডিকশনারির ভ্যালু হচ্ছে ১০০ কারন ওটা কয়েনের সংখ্যা যা ইউজার ডিভাইস A তে সংগ্রহ করেছেন। এই পয়েন্টে ডাটাতে কোন গণনা সংঘটিত হয় নাই- ডিভাইস A সাদাসিধেভাবে এটাতে

ইউজার কর্তৃক সংগৃহীত কয়েনের সংখ্যা রিপোর্ট করে।

- কয়েনের প্রতিটা নতুন জমা ডিভাইসের সাথে সম্পৃক্ত ডিকশনারিতে প্যাকেজ হয় যা এটাকে ক্লাউডে সেভ করে। যখন প্লেয়ার ডিভাইস A তে আরও ১০ কয়েন সংগ্রহ করে, উদাহরণস্বরূপ, ডিভাইস A এর ভ্যালু ১১০ এ উন্নিত হয়।
- নেট ফলঅফল হচ্ছে যে প্রতিটা ডিভাইসে ইউজার কর্তৃক সংগৃহীত কয়েনের সংখ্যা অ্যাপস জানে। এইভাবে এটা সহজ ভাবে সর্বমোট গণনা করে।

টেবিল ৪. সফল কি-ভ্যালু প্লেয়ার কৌশল প্রয়োগ

Event	Data on Device A	Data on Device B	Data on Cloud	Actual Total
Starting conditions	(X:20, x)	(X:20, x)	(X:20, x)	20
Player collects 100 coins on device A	(X:20, A:100)	(X:20)	(X:20)	120
Device A saves state to cloud	(X:20, A:100)	(X:20)	(X:20, A:100)	120
Player collects 10 more coins on device A	(X:20, A:110)	(X:20)	(X:20, A:100)	130
Player collects 1 coin on device B	(X:20, A:110)	(X:20, B:1)	(X:20, A:100)	131
Device B attempts to save state to cloud. Conflict detected.	(X:20, A:110)	(X:20, B:1)	(X:20, A:100)	131
Device B solves conflict	(X:20, A:110)	(X:20, A:100, B:1)	(X:20, A:100, B:1)	131
Device A tries to upload its data to the cloud. Conflict detected.	(X:20, A:110)	(X:20, A:100, B:1)	(X:20, A:100, B:1)	131
Device A resolves the conflict	(X:20, A:110, B:1)	(X:20, A:100, B:1)	(X:20, A:110, B:1) total 131	131

আপনার ডাটা পরিষ্কার করুন

ক্লাউড সেভ ডাটার সাইজে সীমাবদ্ধতা আছে, তাই এই আর্টিকেলে যে কৌশল অনুসরণ করে রূপরেখা করা হয়েছে, ইচ্ছেমত বড় ডিকশনারি তৈরী না করারটার প্রতি খেয়াল রাখা হয়েছে। প্রথম দর্শনে এটা মনে হতে পারে যে ডিকশনারিটির ডিভাইস প্রতি শুধু একটি এন্ট্রি থাকবে, এবং এমনকি অতি উৎসাহি ইউজারের এগুলোর হাজারটা থাকবে না। অন্যদিকে একটি ডিভাইস আইডি অর্জন করা কঠিন এবং একটি খারাপ চর্চা হিসাবে বিবেচনা করা হয়, সুতরাং পরিবর্তে আপনার উচিত একটি ইনসটলেশন আইডি ব্যবহার করা উচিত যা অর্জন করা সহজ এবং এটা খুবই নির্ভরযোগ্য। এর মানে হচ্ছে যে প্রতিটা ডিভাইসে ইউজার কর্তৃক অ্যাপলিকেশন ডাউনলোড করার প্রতিটা সময় ডিকশনারির একটি এন্ট্রি থাকবে। ধারণ করা প্রতিটা কি-ভ্যালু ১২ বাইটস নেয়, এবং যেহেতু একটি একক ক্লাউড সেভ বাফার 128 K পর্যন্ত হতে পারে, আপনি নিরাপদ যদি আপনার 4,096 পর্যন্ত এন্ট্রি থেকে থাকে।

বাস্তবিক জীবনে আপনার ডাটা কয়েনের একটি সংখ্যার চেয়ে সম্ভবত আরও জটিল হবে। এই ক্ষেত্রে, এই ডিকশনারির এন্ট্রির সংখ্যা আরও সীমিত হবে। আপনার বাস্তবায়নের ভিত্তি করে, টাইমস্ট্যাম্প সেটার করার একটি মানে থাকবে যখন ডিকশনারির প্রতিটা এন্ট্রি পরিবর্তন হবে। যখন আপনি চিহ্নিত করবেন যে একটি প্রদত্ত এন্ট্রি গত সপ্তাহগুলোতে বা মাসে পরিবর্তন হয় নাই, কয়েনটিকে আরেকটি এন্ট্রিতে বদলি করাটা পুরাতন এন্ট্রিকে ডিলিট করাটা সম্ভবত নিরাপদ।

সিঙ্ক অ্যাডাপটর ব্যবহার করে ডাটা ট্রান্সফার

(<http://developer.android.com/training/sync-adapters/index.html>)

একটি অ্যান্ড্রয়েড ডিভাইস এবং ওয়েব সার্ভারের মধ্যে ডাটা সিঙ্ক্রোনাইজ করাটা আপনার অ্যাপলিকেশনকে ইউজারের জন্য আরও নির্দিষ্টভাবে উপকারী এবং প্রভাববিস্তারকারী করে তুলবে। উদাহরণ হিসাবে বলা যায় ওয়েব সার্ভারে ডাটা ট্রান্সফার করা একটি কার্যকরী ব্যাকআপ তৈরী করে, এবং সার্ভার থেকে ডাটা ট্রান্সফার করাটা এটাকে ইউজারের কাছে সহজপ্রাপ্য করে তোলে এমনকি যখন এটা অফ লাইনে থাকে তখনও। কিছু কিছু ক্ষেত্রে ইউজার দেখবে ওয়েব ইন্টারফেসে প্রবেশ করা এবং ডাটা এডিট করা খুব সহজ এবং তখন ওই ডাটা তাদের ডিভাইসে সহজপ্রাপ্য হবে, অথবা তারা সময় জুড়ে ডাটা সংগ্রহ করতে পারে এবং এগুলো একটি কেন্দ্রীয় স্টোর এলাকায় আপলোড করতে পারে।

যদিও আপনি আপনার নিজস্ব অ্যাপের ডাটা স্থানান্তরের জন্য আপনার নিজস্ব ব্যবস্থা ডিজাইন করতে পারেন, কিনুত আপনার উচিত অ্যান্ড্রয়েডের সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক ব্যবহার করার বিষয়টা বিবেচনা করা। এই ফ্রেমওয়ার্ক ডাটা স্থানান্তরকে পরিচালিত এবং স্বয়ংক্রিয় করতে সহায়তা করে এবং বিভিন্ন অ্যাপ জুরে যে সিঙ্ক্রোনাইজ কার্যকলাপ হয়ে থাকে তা সমন্বয় করে। যখন আপনি এই ফ্রেমওয়ার্ক ব্যবহার করবেন আপনি কিছু বিশেষত্বের সুবিধা নিতে পারবেন যা আপনার নিজের তৈরী করা ডাটা ট্রান্সফার স্কিমে পাওয়া যাবে না:

Plug-in architecture / প্লাগ-ইন আর্কিটেকচার

সিস্টেমে ক্যালাবল উপাদান রূপে ডাটা ট্রান্সফার কোড যুক্ত করতে দেয়।

Automated execution/ স্বয়ংক্রিয় কার্যসম্পাদন

কিছু মানদন্ডের যেমন ডাটার পরিবর্তন, অতিবাহিত সময় বা অতিবাহিত দিনের উপর ভিত্তি করে আপনাকে ডাটা ট্রান্সফারকে স্বয়ংক্রিয় করতে দেয়। উপরনুত সিস্টেম কিছু স্থানান্তরকে যুক্ত করে যা একটি কিউয়ে রান করতে অসামর্থ এবং তাতেও যখন সম্ভব তখন রান করানো হয়।

Automated network checking / স্বয়ংক্রিয় নেটওয়ার্ক পরিষ্করণ

এই সিস্টেম শুধুমাত্র তখনই আপনার ডাটা স্থানান্তর করে যখন ডিভাইসের নেটওয়ার্ক কানেকটিবিটি থাকে।

Improved battery performance/ ব্যাটারির কার্যসম্পাদনের উন্নয়ন

আপনাকে আপনার অ্যাপের সকল ডাটা স্থানান্তরের কাজগুলো একটি স্থানে কেন্দ্রীভূত করতে দেয়, যাতে সেগুলো একসাথে রান করতে পারে। আপনার ডাটা ট্রান্সফার অন্য অ্যাপের ডাটা স্থানান্তরের সাথে একত্রে নির্ধারিত। এই বিষয়গুলো নেটওয়ার্কে বারবার সুইচ করাটাকে কমিয়ে আনে যা ব্যাটারির ব্যবহার কমিয়ে আনে।

Account management and authentication / একাউন্ট ব্যবস্থাপনা এবং প্রমানীকরণ (অথেনটিকেশন)

একীভূত করা

যদি আপনার অ্যাপ ইউজারের পরিচয় বা সার্ভাও লগ-ইন চায়, আপনি ঐচ্ছিকভাবে আপনার ডাটা স্থানান্তরের মধ্যে একাউন্ট ব্যবস্থাপনা এবং প্রমিতীকরণ (অথেনটিকেশন) একীভূত করতে পারেন।

এই ক্লাস আপনাকে দেখাবে কীভাবে একটি সিন্ক অ্যাডাপটর তৈরী করতে হয় এবং এটাকে র্যাপ (wraps) করা Service কে বাউন্ড করে, অন্য উপাদান যা ফ্রেমওয়ার্কের মধ্যে সিন্ক অ্যাডাপটর প্লাগ করতে সহায়তা করে তা কীভাবে সরবরাহ করা যায় এবং কীভাবে ভিন্ন উপায়ে সিন্ক অ্যাডাপটরকে রান করানো যায়।

নোট: সিন্ক অ্যাডাপটর আসিঙ্কনাস ভাবে রান করে, সুতরা আপনাকে সেই আকাঙ্ক্ষা নিয়ে তাদেরকে রান করানো উচিত যে তারা নিয়মিতভাবে এবং দক্ষতার সাথে ডাটা ট্রান্সফার করতে পারে, কিন্ত তাৎক্ষনিকভাবে নয়। যদি আপনার রিয়েল-টাইম ডাটা ট্রান্সফার করার প্রয়োজন হয়, আপনি এটা একটি AsyncTask বা একটি IntentService তে করতে পারেন।

অনুশীলনীসমূহ

একটি স্টাব-অথেনটিকেটর (প্রমাণকারী) তৈরী করা

এখানে শিখুন কীভাবে একটি একাউন্ট-চালিত করা উপাদান যুক্ত করতে হয় যে সিক্স অ্যাডাপটর ফ্রেমওয়ার্ক আপনার অ্যাপের অংশ হওয়ার আকাঙ্ক্ষা করে। এই অনুশীলনী আপনাকে দেখাবে সহজ করার জন্য কীভাবে একটি একটি স্টাব-অথেনটিকেটর (প্রমাণকারী) উপাদান তৈরী করতে হয়।

একটি স্টাব-কনটেন্ট প্রভাইডার তৈরী করা

এখানে শিখুন কীভাবে একটি কনটেন্ট প্রভাইডার উপাদান যুক্ত করতে হয় যে সিক্স অ্যাডাপটর ফ্রেমওয়ার্ক আপনার অ্যাপের অংশ হওয়ার আকাঙ্ক্ষা করে। এই অনুশীলনী মনে করে যে আপনার অ্যাপ একটি কনটেন্ট প্রভাইডার ব্যবহার করে না, তাই এটা আপনাকে দেখায় কীভাবে একটি স্টাব উপাদান যুক্ত করতে হয়। যদি আপনার অ্যাপের মধ্যে ইতিমধ্যে একটি কনটেন্ট প্রভাইডার থেকে থাকে, তাহলে এই অনুশীলনী বাদ দিয়ে পরের টাতে চলে যান।

একটি সিক্স অ্যাডাপটর তৈরী করুন

এখানে শিখুন একটি উপাদানের মধ্যে কীভাবে আপনার ডাটা ট্রান্সফার কোড প্রবেশ করাবেন যাতে সিক্স অ্যাডাপটর ফ্রেমওয়ার্ক স্বয়ংক্রিয়ভাবে রান করতে পারে।

একটি সিক্স অ্যাডাপটর রান করানো

এখানে শিখুন সিক্স অ্যাডাপটর ফ্রেমওয়ার্ক ব্যবহার করে কীভাবে ডাটা ট্রান্সফার সক্রিয় এবং পরিকল্পনা করা যায়।

একটি স্টাব-অথেনটিকেটর (প্রমাণকারী) তৈরি করা

(<http://developer.android.com/training/sync-adapters/creating-authenticator.html>)

সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক মনে করে যে আপনার সিঙ্ক অ্যাডাপটর একটি একাউন্ট এবং সার্ভার স্টোরেজ যা লগ-ইন এক্সেস চায় তাদের সাথে সম্পৃক্ত ডিভাইস স্টোরেজের মধ্যে ডাটা ট্রান্সফার করে। এই জন্যে, ফ্রেমওয়ার্ক আপনার কাছে আপনার সিঙ্ক অ্যাডাপটরের অংশ হিসাবে অথেনটিকেটর নামে একটি উপাদান আকাঙ্ক্ষা করে। এই উপাদান অ্যান্ড্রয়েড একাউন্টের এবং অথেনটিকেশন ফ্রেমওয়ার্কের মধ্যে প্লাগ করে এবং ইউজারের পরিচয় যেমন লগ-ইন তথ্য পরিচালনার জন্য একটি আদর্শ ইন্টারফেস প্রদান করে।

এমনকি যদি আপনার অ্যাপ একাউন্ট ব্যবহার না করে, আপনাকে তখনও একটি অথেনটিকেটর উপাদান প্রদান করতে হবে। যদি আপনি একাউন্ট বা সার্ভার লগ-ইন ব্যবহার করে না থাকেন, অথেনটিকেটর দ্বারা চালিত তথ্য অবহেলিত হবে, তাই আপনি একটি অথেনটিকেটর উপাদান সরবরাহ করতে পারেন যা স্টাব পদ্ধতি বাস্তবায়ন ধারণ করে। আপনার একটি বাউন্ড Service ও সরবরাহ করা উচিত যা সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ককে অথেনটিকেটরের পদ্ধতি কল করতে দেয়।

এই অনুশীলনী আপনাকে দেখায় কীভাবে একটি স্টাব অথেনটিকেটরের সকল অংশ নির্ধারণ করতে হয় যে আপনার সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্কের প্রয়োজনীয়তাকে সনুতষ্ট করা প্রয়োজন। যদি আপনার একটি সত্যিকারের অথেনটিকেটর প্রয়োজন হয় যা ইউজার একাউন্ট চালিত করে, AbstractAccountAuthenticator জন্য রেফারেন্স ডকুমেন্টেশনটি পড়ুন।

একটি স্টাব অথেনটিকেটর উপাদান যুক্ত করুন

আপনার অ্যাপে একটি স্টাব অথেনটিকেটর উপাদান যুক্ত করতে, একটি ক্লাস তৈরী করুন যা `AbstractAccountAuthenticator` কে প্রসারিত করে, এবং তখন প্রয়োজনীয় পদ্ধতি লোপ করে, হয় `null` ফেরত দিয়ে বা একটি ব্যতিক্রম দেয়ার মাধ্যমে।

নিচের চিত্রটি একটি স্টাব অথেনটিকেটর ক্লাসের উদাহরণ দেখাচ্ছে:

```
/*
 * Implement AbstractAccountAuthenticator and stub out all
 * of its methods
 */
public class Authenticator extends AbstractAccountAuthenticator {
    // Simple constructor
    public Authenticator(Context context) {
        super(context);
    }
    // Editing properties is not supported
    @Override
    public Bundle editProperties(
        AccountAuthenticatorResponse r, String s) {
        throw new UnsupportedOperationException();
    }
    // Don't add additional accounts
    @Override
    public Bundle addAccount(
        AccountAuthenticatorResponse r,
        String s,
        String s2,
        String[] strings,
        Bundle bundle) throws NetworkErrorException {
        return null;
    }
    // Ignore attempts to confirm credentials
    @Override
    public Bundle confirmCredentials(
        AccountAuthenticatorResponse r,
        Account account,
        Bundle bundle) throws NetworkErrorException {
        return null;
    }
    // Getting an authentication token is not supported
    @Override
    public Bundle getAuthToken(
        AccountAuthenticatorResponse r,
        Account account,
        String s,
        Bundle bundle) throws NetworkErrorException {
        throw new UnsupportedOperationException();
    }
    // Getting a label for the auth token is not supported
    @Override
    public String getAuthTokenLabel(String s) {
        throw new UnsupportedOperationException();
    }
    // Updating user credentials is not supported
    @Override
    public Bundle updateCredentials(
        AccountAuthenticatorResponse r,
```

```
        Account account,
        String s, Bundle bundle) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}
// Checking features for the account is not supported
@Override
public Bundle hasFeatures(
    AccountAuthenticatorResponse r,
    Account account, String[] strings) throws NetworkErrorException {
    throw new UnsupportedOperationException();
}
}
```

ফ্রেমওয়ার্কে অথেনটিকেটর সংযুক্ত করা

আপনার অথেনটিকেটরে সিঙ্ক অ্যাডাপ্টর ফ্রেমওয়ার্ক প্রবেশ করানোর জন্য, আপনাকে অবশ্যই এটার জন্য একটি বাউন্ড সার্ভিস তৈরী করতে হবে। এই সার্ভিস একটি অ্যান্ড্রয়েড বাইন্ডার অবজেক্ট প্রদান করবে যা ফ্রেমওয়ার্ককে আপনার অথেনটিকেটরকে কল করতে দেয় এবং অথেনটিকেটর ও ফ্রেমওয়ার্ক এই দুইয়ের মধ্যে ডাটা পাস করে দেয়।

যেহেতু ফ্রেমওয়ার্ক প্রথমবার এই সার্ভিস শুরু করে এটার অথেনটিকেটরে প্রবেশ করা প্রয়োজন হয়, সার্ভিসের `Service.onCreate()` পদ্ধতিতে অথেনটিকেশনের কনস্ট্রাক্টরকে কল করা মাধ্যমে আপনি অথেনটিকেটর আরম্ভ করতে সার্ভিসটি ব্যবহারও করতে পেরেন।

নিচের চিত্রটি দেখায় কীভাবে বাইন্ড `Service` নির্ধারণ করতে হয়:

```
/**
 * A bound Service that instantiates the authenticator
 * when started.
 */
public class AuthenticatorService extends Service {
    ...
    // Instance field that stores the authenticator object
    private Authenticator mAuthenticator;
    @Override
    public void onCreate() {
        // Create a new authenticator object
        mAuthenticator = new Authenticator(this);
    }
    /*
     * When the system binds to this Service to make the RPC call
     * return the authenticator's IBinder.
     */
    @Override
    public IBinder onBind(Intent intent) {
        return mAuthenticator.getIBinder();
    }
}
```

অথেনটিকেটর মেটাডেটা ফাইল যুক্ত করা

সিঙ্ক অ্যাডাপ্টর এবং একাউন্ট ফ্রেমওয়ার্কের মধ্যে আপনার অথেনটিকেটর উপাদান যুক্ত করতে, আপনার প্রয়োজন মেটাডেটা সহকারে এই ফ্রেমওয়ার্কগুলো সরবরাহ করা যা উপাদান বর্ণনা করে। এই মেটাডেটা একাউন্ট টাইপ ঘোষণা করে যা আপনি আপনার সিঙ্ক একাউন্টের জন্য তৈরী করেছেন এবং ইউজার ইন্টারফেস ডিক্লেয়ার করে যে যদি আপনি আপনার একাউন্ট টাইপ ইউজারের কাছে দৃশ্যমান করতে চান সিস্টেম এটা প্রদর্শন করে। এই মেটাডেটা আপনার অ্যাপ প্রজেক্টের `/res/xml/` ডিরেক্টরির মধ্যে স্টোর হওয়া একটি XML ফাইলে ডিক্লেয়ার করুন। আপনি ফাইলটির যে কোন নাম দিতে পারেন, যদিও এটা সাধারনত `authenticator.xml` নামে পরিচিত।

এই XML ফাইল একটি একক এলিমেন্ট `< account-authenticator>` ধারণ করে যার নিচের গুণগুলো রয়েছে:

`android:accountType`

সিঙ্ক অ্যাডাপ্টর ফ্রেমওয়ার্ক প্রতিটা সিঙ্ক অ্যাডাপ্টরের একটি একাউন্ট টাইপ থাকুক, একটি ডোমেইন নেম আকারে। ফ্রেমওয়ার্ক সিঙ্ক অ্যাডাপ্টরের অভ্যন্তরিন পরিচিতির অংশ হিসাবে একাউন্ট টাইপ ব্যবহার করে। সার্ভারের জন্য লগ-ইন প্রয়োজন, একটি ইউজার একাউন্টের সাথে একাউন্ট টাইপ লগ-ইন তথ্যের অংশ হিসাবে সার্ভারে পাঠানো হয়ে থাকে।

যদি আপনার সার্ভার লগ-ইন না চায়, আপনাকে তখনও একটি একাউন্ট টাইপ সরবরাহ করতে হবে। ভ্যালুর জন্য, একটি ডোমেইন নেম ব্যবহার করুন যা আপনি নিয়ন্ত্রণ করেন। যখন ফ্রেমওয়ার্ক আপনার সিঙ্ক অ্যাডাপ্টর পরিচালনা করতে এটা ব্যবহার করে, ভ্যালু আপনার সার্ভারে পাঠানো হয় না।

`android:icon`

একটি আইকন ধারণ করা `Drawable` রিসোর্স এর প্রতি নির্দেশক। আপনি যদি `res/xml/syncadapter.xml` এর মধ্যে বিশেষত্ব `android:userVisible="true"` নির্দিষ্ট করার মাধ্যমে সিঙ্ক অ্যাডাপ্টর দৃশ্যমান করেন, তখন আপনাকে অবশ্যই এই আইকন রিসোর্স সরবরাহ করতে হবে। এটা সিস্টেমের সেটিংস অ্যাপের একাউন্টস (Accounts) সেকশনে দেখা যায়।

`android:smallIcon`

একটি ছোট আকারের আইকন ধারণ করা `Drawable` রিসোর্স এর প্রতি নির্দেশক। স্ক্রিনের আকারের উপর ভিত্তি করে এই রিসোর্স সিস্টেমের সেটিংস অ্যাপের একাউন্টস (Accounts) সেকশনের মধ্যে থাকা `android:icon` এর পরিবর্তে ব্যবহৃত হয়।

`android:label`

স্থানীয়করণযোগ্য স্ট্রিং যা ইউজারের একাউন্ট টাইপ পরীক্ষা করে। আপনি যদি `res/xml/syncadapter.xml` এর মধ্যে বিশেষত্ব `android:userVisible="true"` নির্দিষ্ট করার মাধ্যমে সিঙ্ক অ্যাডাপ্টর দৃশ্যমান করেন, তখন আপনাকে এই স্ট্রিং সরবরাহ করতে হবে। এটা সিস্টেমের

সেটিংস অ্যাপের একাউন্টস (Accounts) সেকশনে দেখা যায়, অথেনটিকেটরের জন্য যে আইকন নির্ধারণ করেছেন তার পাশে দেখা যায়।

নিচের চিত্রটি অথেনটিকেটরের জন্য XML ফাইল দেখায় যা আপনি পূর্বে তৈরী করেছেন:

```
<?xml version="1.0" encoding="utf-8"?>
<account-authenticator
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:accountType="example.com"
    android:icon="@drawable/ic_launcher"
    android:smallIcon="@drawable/ic_launcher"
    android:label="@string/app_name"/>
```

মেনিফেস্টের মধ্যে অথেনটিকেটর ডিক্লেয়ার করা

পূর্ববর্তী ধাপে, আপনি একটি বাউন্ড Service তৈরী করেছেন যা অথেনটিকেটরকে সিন্ধ অ্যাডাপটর ফ্রেমওয়ার্কের সাথে যুক্ত করে। সিস্টেমে এই সার্ভিস চিহ্নিত করতে, নিচের `<service>` এলিমেন্টটি `<application>` এর একটি চাইল্ড এলিমেন্ট হিসাবে যুক্ত করার মাধ্যমে আপনার অ্যাপ মেনিফেস্টের মধ্যে এটা ডিক্লেয়ার করতে হবে:

```
<service
    android:name="com.example.android.syncadapter.AuthenticatorService">
    <intent-filter>
        <action android:name="android.accounts.AccountAuthenticator"/>
    </intent-filter>
    <meta-data
        android:name="android.accounts.AccountAuthenticator"
        android:resource="@xml/authenticator" />
</service>
```

`< intent-filter>` এলিমেন্টটি একটি ফিল্টার সেট আপ করে যা ইনটেন্ট একশন `android.accounts.AccountAuthenticator` দ্বারা সক্রিয় হয়, যা অথেনটিকেটর রান করতে সিস্টেম দ্বারা পাঠানো হয়ে থাকে। যখন ফিল্টার সক্রিয় হয়ে থাকে, সিস্টেমটি `AuthenticatorService` শুরু করে, বাউন্ড Service সার্ভিস যা আপনি অথেনটিকেটর র‍্যাপ করার জন্য প্রদান করেছেন।

`< meta-data>` এলিমেন্ট অথেনটিকেটরের জন্য মেটাডাটা ডিক্লেয়ার করে। `android:name` এট্রিবিউট অথেনটিকেশন ফ্রেমওয়ার্কে মেটাডেটা সংযোগ করে। `android:resource` এলিমেন্ট আপনি পূর্বে তৈরী করেছেন এমন অথেনটিকেটর মেটাডেটা ফাইলের নাম নির্দিষ্ট করে।

একটি অথেনটিকেটর ছাড়াও, একটি সিন্ধ অ্যাডাপটরও কনটেন্ট প্রভাইডার চায়। যদি আপনার অ্যাপ ইতমধ্যে কনটেন্ট প্রভাইডার ব্যবহার না করে থাকে, তাহলে পরবর্তী অনুশীলনীতে চলে যান যেখানে একটি স্টাব কনটেন্ট প্রভাইডার তৈরী করা শেখানো হয়েছে; অন্যথায় `Creating a Sync Adapter` অনুশীলনীতে চলে যান।

একটি স্টাব কনটেন্ট প্রভাইডার তৈরী করা

(<http://developer.android.com/training/sync-adapters/creating-stub-provider.html>)

সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক নমনীয় এবং অতি নিরাপদ কনটেন্ট প্রভাইডার ফ্রেমওয়ার্ক দ্বারা চালিত ডিভাইস ডাটার সাথে কাজ করার মতো করে ডিজাইন করা। এই কারনে, সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক আকাঙ্ক্ষা করে যে একটি অ্যাপ যা ফ্রেমওয়ার্ক ব্যবহার করে সেটা ইতমধ্যে এর লোকাল ডাটার জন্য একটি কনটেন্ট প্রভাইডার নির্ধারণ করা হয়েছে। যদি সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক আপনার সিঙ্ক অ্যাডাপটর রান করার চেষ্টা করে এবং আপনার অ্যাপের কনটেন্ট প্রভাইডার না থাকে, আপনার সিঙ্ক অ্যাডাপটর ক্র্যাশ করবে।

যদি আপনি একটি নতুন অ্যাপ ডেভেলপ করতে থাকেন যা একটি সার্ভার থেকে ডিভাইসে ডাটা স্থানান্তরিত করে, আপনার উচিত একটি কনটেন্ট প্রভাইডারের মধ্যে লোকাল ডাটা স্টোর করার বিষয়টি জোড়ালোভাবে বিবেচনা করা। সিঙ্ক অ্যাডাপটরের জন্য তাদের গুরুত্ব ছাড়াও কনটেন্ট প্রভাইডার নানা ধরনের নিরাপত্তার (সিকিউরিটির) সুবিধা দেয়ার প্রস্তাব করে এবং বিশেষভাবে অ্যান্ড্রয়েড সিস্টেমে ডাটা স্টোরেজ পরিচালনা করার জন্য ডিজাইন করা হয়। একটি কনটেন্ট প্রভাইডার তৈরী করার বিষয়ে বিস্তারিত জানতে Creating a Content Provider দেখুন।

কিন্তু যদি আপনি ইতমধ্যে লোকাল ডাটা অন্য কোন আকারে স্টোর করে থাকেন, আপনি এখনও ডাটা ট্রান্সফার পরিচালিত করতে একটি সিঙ্ক অ্যাডাপটর ব্যবহার করতে পারেন। একটি কনটেন্ট প্রভাইডারের জন্য সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক প্রয়োজনীয়তা তৃপ্ত করতে, আপনার অ্যাপে একটি স্টাব কনটেন্ট প্রভাইডার যুক্ত করুন। একটি স্টাব প্রভাইডার কনটেন্ট প্রভাইডার ক্লাস বাস্তবায়ন করে, কিন্তু এর সকল প্রয়োজনীয় পদ্ধতি null বা ০ ফেরত দেয়। আপনি যদি একটি স্টাব প্রভাইডার যুক্ত করেন, আপনি যেটা পছন্দ করেন তার যে কোন স্টোরেজ পদ্ধতি থেকে ডাটা স্থানান্তরিত করতে আপনি তখন একটি সিঙ্ক অ্যাডাপটর ব্যবহার করতে পারেন।

আপনার অ্যাপে যদি ইতমধ্যে একটি কনটেন্ট প্রভাইডার থেকে থাকে, আপনার স্টাব কনটেন্ট প্রভাইডারের কোন প্রয়োজন নেই। সেক্ষেত্রে আপনি এই অনুশীলনী ছেড়ে Creating a Sync Adapter অনুশীলনীতে চলে যান। যদি আপনার অ্যাপের এখনও কনটেন্ট প্রভাইডার না থেকে থাকে, এই অনুশীলনী আপনাকে শেখাবে একটি স্টাব কনটেন্ট প্রভাইডার যুক্ত করতে হয় যা আপনাকে ফ্রেমওয়ার্কের মধ্যে সিঙ্ক অ্যাডাপটর প্লাগ করতে দেয়।

একটি স্টাব কনটেন্ট প্রভাইডার যুক্ত করা

আপনার অ্যাপের জন্য একটি স্টাব কনটেন্ট প্রভাইডার তৈরী করতে, ক্লাস `ContentProvider` প্রসারিত করুন এবং এর প্রয়োজনীয় পদ্ধতির রৌপ করো। নিচের চিত্রটি আপনাকে দেখায় কীভাবে স্টাব প্রভাইডার তৈরী করতে হয়:

```
/*
 * Define an implementation of ContentProvider that stubs out
 * all methods
 */
public class StubProvider extends ContentProvider {
    /*
     * Always return true, indicating that the
     * provider loaded correctly.
     */
    @Override
    public boolean onCreate() {
        return true;
    }
    /*
     * Return an empty String for MIME type
     */
    @Override
    public String getType() {
        return new String();
    }
    /*
     * query() always returns no results
     */
    @Override
    public Cursor query(
        Uri uri,
        String[] projection,
        String selection,
        String[] selectionArgs,
        String sortOrder) {
        return null;
    }
    /*
     * insert() always returns null (no URI)
     */
    @Override
    public Uri insert(Uri uri, ContentValues values) {
        return null;
    }
    /*
     * delete() always returns "no rows affected" (0)
     */
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        return 0;
    }
    /*
     * update() always returns "no rows affected" (0)
     */
    public int update(
        Uri uri,
        ContentValues values,
        String selection,
```

```
        String[] selectionArgs) {  
    return 0;  
}  
}
```

মেনিফেস্টের মধ্যে প্রভাইডার ডিক্লেয়ার করা

আপনার অ্যাপ তার মেনিফেস্টের মধ্যে একটি প্রভাইডার ডিক্লেয়ার করেছে তা চেক করার মাধ্যমে সিন্ধ অ্যাডাপটর ফ্রেমওয়ার্ক যাচাই করে যে আপনার অ্যাপের একটি কনটেন্ট প্রভাইডার আছে। মেনিফেস্টের মধ্যে স্টাব প্রভাইডার ডিক্লেয়ার করতে, একটি `<provider>` এলিমেন্ট যুক্ত করুন যার মধ্যে নিম্নোক্ত গুণাগুণগুলো রয়েছে:

```
android:name="com.example.android.datasync.provider.StubProvider"
```

ক্লাসের পূর্ণ-যোগ্যতাসম্পন্ন নাম নির্দিষ্ট করে যা স্টাব কনটেন্ট প্রভাইডার বাস্তবায়ন করে।

```
android:authorities="com.example.android.datasync.provider"
```

একটি URI অথরিটি যা স্টাব কনটেন্ট প্রভাইডার চিহ্নিত করে। এই ভ্যালু কে আপনার অ্যাপের প্যাকেজ নাম করুন স্ট্রিং এর সাথে সংযোজন করা `".provider"` দিয়ে। যদিও আপনি সিস্টেমে আপনার স্টাব প্রভাইডার ডিক্লেয়ার করছেন, প্রভাইডার নিজে ছাড়া আর কোন কিছুই প্রবেশ করতে চেষ্টা করে না।

```
android:exported="false"
```

অন্য অ্যাপসও কনটেন্ট প্রভাইডারে প্রবেশ করতে পারে কিনা তা নির্ধারণ করে। আপনার স্টাব কনটেন্ট প্রভাইডারের জন্য, `false` এ ভ্যালু সেট করুন, যেহেতু প্রভাইডার দেখার জন্য অন্য অ্যাপকে অনুমোদন করার কোন প্রয়োজন নেই। এই ভ্যালু সিন্ধ অ্যাডাপটর এবং কনটেন্ট প্রভাইডারের মধ্যকার মিথস্ক্রিয়ায় কোন প্রভাব ফেলে না।

```
android:syncable="true"
```

একটি ফ্ল্যাগ সেট করে যা নির্দেশ করে যে প্রভাইডার হচ্ছে সিন্ধ করার যোগ্য। আপনি যদি এই ফ্ল্যাগ `true` এ সেট করেন, আপনার কোডে `setIsSyncable()` কল করার কোন প্রয়োজন নেই। ফ্ল্যাগটি সিন্ধ অ্যাডাপটর ফ্রেমওয়ার্ক কে কনটেন্ট প্রভাইডার দিয়ে ডাটা স্থানান্তরিত করতে দেয়, কিন্ত ট্রান্সফার তখনই ঘটবে আপনি যদি সোজাসুজিভাবে তা করে থাকেন।

নিচের চিত্রটি আপনাকে দেখায় কীভাবে অ্যাপ মেনিফেস্ট `<provider>` এলিমেন্ট যুক্ত করা যায়:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.network.sync.BasicSyncAdapter"
    android:versionCode="1"
    android:versionName="1.0" >
```

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    ...
    <provider
        android:name="com.example.android.datasync.provider.StubProvider"
        android:authorities="com.example.android.datasync.provider"
        android:export="false"
        android:syncable="true"/>
    ...
</application>
</manifest>
```

এখন আপনি যে সিন্ক অ্যাডাপটার ফ্রেমওয়ার্ক দ্বারা কাঙ্ক্ষিত নির্ভরতা (dependencies) তৈরী করেছেন, আপনি উপাদান তৈরী করতে পারেন যা আপনার ডাটা ট্রান্সফার কোড পরিবেষ্টিত করে। এই উপাদানকে সিন্ক অ্যাডাপটার বলা হয়। পরবর্তী অনুশীলনী দেখাবে কীভাবে এই উপাদান আপনার অ্যাপে যুক্ত করা যায়।

একটি সিন্ক অ্যাডাপটর তৈরী করা

(<http://developer.android.com/training/sync-adapters/creating-sync-adapter.html>)

আপনার অ্যাপের সিন্ক অ্যাডাপটর উপাদান কাজের জন্য কোড পরিবেষ্টিত করে রাখে যা ডিভাইস এবং একটি সার্ভারের মধ্যে ডাটা ট্রান্সফার করে। আপনার অ্যাপে প্রদান করা সুচি এবং ট্রিগার অনুযায়ী সিন্ক অ্যাডাপটর ফ্রেমওয়ার্ক সিন্ক অ্যাডাপটর উপাদানের মধ্যে কোড রান করে। আপনার অ্যাপে সিন্ক অ্যাডাপটর উপাদান যুক্ত করতে আপনার নিম্নোক্ত অংশগুলো প্রয়োজন হবে:

সিন্ক অ্যাডাপটর ক্লাস

একটি ক্লাস যা সিন্ক অ্যাডাপটর ফ্রেমওয়ার্ক সাথে যথোপযুক্ত একটি ইন্টারফেসের মধ্যে আপনার ডাটা ট্রান্সফার কোড র‍্যাপ করে(আচ্ছাদিত) করে।

বাইন্ড সার্ভিস

একটি উপাদান যা সিন্ক অ্যাডাপটর ফ্রেমওয়ার্ককে আপনার সিন্ক অ্যাডাপটর ক্লাসে কোড রান করতে দেয়।

অ্যাডাপটর XML মেটাডেটা ফাইল সিন্ক করুন

আপনার সিন্ক অ্যাডাপটর সম্পর্কিত তথ্য ধারণ করা একটি ফাইল। কীভাবে আপনার ডাটা ট্রান্সফার লোড হবে এবং পরিকল্পিত হবে খুঁজে বের করতে ফ্রেমওয়ার্ক এই ফাইল পড়ে।

অ্যাপ মেনিফেস্টের মধ্যে ডিক্লেয়ারেশন

XML যা বাইন্ড সার্ভিস ডিক্লেয়ার করে এবং সিন্ক অ্যাডাপটর নির্দিষ্ট মেটাডেটা নির্দেশ করে।

এই অনুশীলনী আপনাকে দেখাবে কীভাবে এই উপাদানগুলো নির্ধারণ করতে হয়।

সিন্ক অ্যাডাপটর ক্লাস তৈরী করুন

অনুশীলনীর এই অংশে আপনি শিখবেন কীভাবে সিন্ক অ্যাডাপটর ক্লাস তৈরী করতে হয় যা ডাটা ট্রান্সফার কোডকে পরিবেষ্টিত করে। ক্লাস তৈরী করা সিন্ক অ্যাডাপটর ভিত্তিক ক্লাস প্রসারিত করা, ক্লাসের জন্য কনস্ট্রাক্টর নির্ধারণ করা, এবং পদ্ধতি বাস্তবায়ন করা কে অন্তর্ভুক্ত করে যেখানে আপনি ডাটা ট্রান্সফার টাস্ক নির্ধারণ করেছেন।

বেজ সিঙ্ক অ্যাডাপটর ক্লাস AbstractThreadedSyncAdapter প্রসারিত করুন

সিঙ্ক অ্যাডাপটর উপাদান তৈরী করতে, AbstractThreadedSyncAdapter প্রসারিত করার এবং এর কনস্ট্রাক্টর লেখার মাধ্যমে শুরু করুন। স্ক্র্যাচ থেকে প্রতিবার আপনার সিঙ্ক অ্যাডাপটর উপাদান তৈরী হওয়ার সময় টাস্ক (কর্মকান্ড) সেটআপ করার সময় কনস্ট্রাক্টর ব্যবহার করুন, যেভাবে একটি একটিভিটি সেটআপ করতে আপনি Activity.onCreate() ব্যবহার করেছেন ঠিক সেভাবে। উদাহরণস্বরূপ, যদি আপনার অ্যাপ ডাটা স্টোর করতে একটি কনটেন্ট প্রভাইডার ব্যবহার করে থাকে, একটি ContentResolver ইনস্টেন্স পাওয়ার জন্য কনস্ট্রাক্টর ব্যবহার করুন। যেহেতু ডুপ্লিকেশনবহুল পূর্ণ আর্গুমেন্ট সাপোর্ট করার জন্য অ্যান্ড্রয়েড প্ল্যাটফর্ম সংস্করণ ৩.০ এর মধ্যে কনস্ট্রাক্টরের দ্বিতীয় ধরন (ফর্ম) যুক্ত করা হয়েছে, সর্ব ক্ষেত্রে উপযোগীতা জারি রাখতে আপনাকে কনস্ট্রাক্টরের দুইটা ধরন (ফর্ম) তৈরী করতে হবে।

নোট: সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক সিঙ্ক অ্যাডাপটর উপাদানের সাথে কাজ করার জন্য ডিজাইন করা হয়েছে যা হচ্ছে সিঙ্গেলটোন ইনসটেন্স। ইনসটেনসিয়েট করা সিঙ্ক অ্যাডাপটর উপাদান Bind the Sync Adapter to the Framework অধ্যায়ে আরও বিস্তারিতভাবে আলোচনায় এসেছে।

নীম্নোক্ত উদাহরণ আপনাকে দেখাবে কীভাবে AbstractThreadedSyncAdapter এবং এর কনস্ট্রাক্টর বাস্তবায়ন করতে হয়:

```
/**
 * Handle the transfer of data between a server and an
 * app, using the Android sync adapter framework.
 */
public class SyncAdapter extends AbstractThreadedSyncAdapter {
    ...
    // Global variables
    // Define a variable to contain a content resolver instance
    ContentResolver mContentResolver;
    /**
     * Set up the sync adapter
     */
    public SyncAdapter(Context context, boolean autoInitialize) {
        super(context, autoInitialize);
        /**
         * If your app uses a content resolver, get an instance of it
         * from the incoming Context
         */
        mContentResolver = context.getContentResolver();
    }
    ...
    /**
     * Set up the sync adapter. This form of the
     * constructor maintains compatibility with Android 3.0
     * and later platform versions
     */
    public SyncAdapter(
        Context context,
        boolean autoInitialize,
        boolean allowParallelSyncs) {
        super(context, autoInitialize, allowParallelSyncs);
    }
}
```

```
/*
 * If your app uses a content resolver, get an instance of it
 * from the incoming Context
 */
mContentResolver = context.getContentResolver();
...
}
```


onPerformSync() এ ডাটা ট্রান্সফার কোড যুক্ত করুন

সিঙ্ক অ্যাডাপ্টর উপাদান স্বয়ংক্রিয়ভাবে ডাটা ট্রান্সফার করে না। পরিবর্তে এটা আপনার ডাটা ট্রান্সফার কোড আবর্তিত করে, যাতে আপনার অ্যাপ থেকে কোন ধরনের সম্পৃক্ততা ছাড়াই সিঙ্ক অ্যাডাপ্টর ফ্রেমওয়ার্ক ব্যাকগ্রাউন্ডে ডাটা ট্রান্সফার রান করাতে পারে। যখন ফ্রেমওয়ার্ক আপনার অ্যাপলিকেশনের ডাটা সিঙ্ক করতে প্রস্তুত হয়, এটা পদ্ধতি `onPerformSync()` এর বাস্তবায়ন আহবান করে।

সিঙ্ক অ্যাডাপ্টর উপাদানে আপনার মূল অ্যাপ কোড থেকে ডাটার ট্রান্সফার সঞ্চালন করতে, সিঙ্ক অ্যাডাপ্টর ফ্রেমওয়ার্ক নিচের আর্গুমেন্টগুলো সহকারে `onPerformSync()` কল করে:

Account

ইভেন্টের সাথে সম্পৃক্ত একটি `Account` অবজেক্ট যা সিঙ্ক অ্যাডাপ্টরকে সক্রিয় করে। যতি আপনার সার্ভার অ্যাকাউন্ট ব্যবহার না করে, আপনার এই অবজেক্টের মধ্যে তথ্য ব্যবহার করার কোন প্রয়োজন নেই।

Extras

ইভেন্ট কর্তৃক পাঠানো একটি `Bundle` ধারণ করা ফ্ল্যাগস।

Authority

সিস্টেমের মধ্যে একটি কনটেন্ট প্রভাইডারের অথরিটি। আপনার অ্যাপের এই প্রভাইডারে প্রবেশগম্যতা থাকতে হবে। সাধারনত, অথরিটি আপনার নিজের অ্যাপের মধ্যে একটি কনটেন্টে যোগাযোগ করে।

Content provider client

কনটেন্ট প্রভাইডারের জন্য একটি `ContentProviderClient` অথরিটি আর্গুমেন্ট দ্বারা নির্দেশ করে। একটি `ContentProviderClient` হচ্ছে একটি কনটেন্ট প্রভাইডারে লাইটওয়েট পাবলিক ইন্টারফেস। একটি `ContentResolver` হিসাবে এটার একই রকম মৌলিক কার্যপ্রণালী রয়েছে। আপনি যদি আপনার অ্যাপের জন্য ডাটা স্টোর করতে একটি কনটেন্ট প্রভাইডার ব্যবহার করতে থাকেন, আপনি এই অবজেক্ট দিয়ে কনটেন্ট প্রভাইডারের সাথে সংযোগ স্থাপন করতে পারেন, অন্যথায় আপনি এটাকে অবহেলা করতে পারেন।

Sync result

একটি `SyncResult` যা আপনি সিঙ্ক অ্যাডাপ্টর ফ্রেমওয়ার্কে তথ্য পঠাতে ব্যবহার করেন।

নিচের অংশটি `onPerformSync()` এর সার্বিক কাঠামো তুলে ধরেছে:

```

/*
 * Specify the code you want to run in the sync adapter. The entire
 * sync adapter runs in a background thread, so you don't have to set
 * up your own background processing.
 */
@Override
public void onPerformSync(
    Account account,
    Bundle extras,
    String authority,
    ContentProviderClient provider,
    SyncResult syncResult) {

/*
 * Put the data transfer code here.
 */
...
}

```

যখন `onPerformSync()` এর আসল বাস্তবায়ন আপনার অ্যাপের ডাটা সিঙ্ক্রোনাইজেশন আকাঙ্ক্ষায় এবং সার্ভার কানেকশন প্রটোকলে নির্দিষ্ট হয়, সেখানে কিছু সাধারণ কাজ থাকে যা আপনার বাস্তবায়নকে অবশ্যই সম্পাদন করতে হবে।

সার্ভারে যুক্ত করা

যদিও আপনি মনে করতে পারেন যে যখন আপনার ডাটা ট্রান্সফার শুরু হয় তখন নেটওয়ার্ক থাকে, সিস্টেম অ্যাডাপটর ফ্রেমওয়ার্ক স্বয়ংক্রিয়ভাবে একটি সার্ভারে যুক্ত হয় না।

ডাট ডাউনলোড এবং আপলোড করা

একটি সিস্টেম অ্যাডাপটর কোন ডাটা ট্রান্সফারের কাজ স্বয়ংক্রিয় পদ্ধতি প্রয়োগ করে না। আপনি যদি সার্ভার থেকে ডাউনলোড করতে চান এবং সেটা কনটেন্ট প্রভাইডারে স্টোর করতে চান, আপনাকে কোড প্রদান করতে হবে যা ডাটাকে রিকোয়েস্ট করে, এটাকে ডাউনলোড করে, এবং প্রভাইডারের মধ্যে এটাকে প্রবেশ করায়। একইভাবে, যদি আপনি সার্ভারে ডাটা সেভ করতে চান, আপনাকে এটা একটি ফাইল, ডাটাবেজ বা প্রভাইডার থেকে পড়তে হবে এবং প্রয়োজনীয় আপলোড রিকোয়েস্ট পাঠাতে হবে। আপনাকে নেটওয়ার্ক এররকেও চালিত করতে হবে যা আপনার ডাটা ট্রান্সফার হওয়ার সময় ঘটতে পারে।

ডাটা কনফ্লিক্ট (সংঘাত/সংঘর্ষ) পরিচালনা বা ডাটা কতটা সাম্প্রতিক তা নির্ধারণ

একটি সিস্টেম অ্যাডাপটর ডিভাইসের ডাটা এবং সার্ভারের ডাটার মধ্যকার সাংঘর্ষিক বিষয় স্বয়ংক্রিয়ভাবে পরিচালনা করে না। যদি সার্ভারের মধ্যকার ডাটা ডিভাইসের মধ্যকার ডাটার চেয়ে অধিকতর সাম্প্রতিক হয় বা এর বিপরীতভাবে ঘটলে এটা সে বিষয়টাও স্বয়ংক্রিয়ভাবে পরিচালনা করে না। পরিবর্তে, এই অবস্থা চালিত করতে আপনাকে আপনার নিজস্ব এলগরিদম সরবরাহ করতে হবে।

পরিচ্ছন্ন করা

আপনার ডাটা ট্রান্সফার শেষে সবসময় একটি সার্ভারের সাথে সংযোগ বন্ধ করতে হবে এবং টেম্প ফাইল এবং ক্যাশে ফাইল পরিচ্ছন্ন করতে হবে।

নোট: সিস্টেম অ্যাডাপটর ফ্রেমওয়ার্ক একটি ব্যাকগ্রাউন্ড থ্রেডে `onPerformSync()` রান করায়, তাই

আপনাকে আপনার নিজস্ব ব্যাকগ্রাউন্ড প্রসেসিং কে সেটআপ করতে হবে না।

concentrating আপনার সিঙ্ক-সম্পর্কিত কাজ ছাড়াও, আপনার নিয়মিত নেটওয়ার্ক-সম্পর্কিত কাজ একত্রিত করার চেষ্টা করা উচিত এবং তাদেরকে `onPerformSync()` এ যুক্ত করা উচিত। এই পদ্ধতিতে আপনার সকল নেটওয়ার্ক কাজ কেন্দ্রীভূত করার মাধ্যমে আপনি ব্যাটারি পাওয়ার সংরক্ষণ করতে পারেন যা নেটওয়ার্ক ইন্টারফেস শুরু করতে এবং বন্ধ করতে প্রয়োজন। নেটওয়ার্ক এক্সেসকে আরও কর্মদক্ষ করার বিষয়ে বিস্তারিত জানতে [Transferring Data Without Draining the Battery](#) প্রশিক্ষন ক্লাসটি দেখুন, যা বেশকিছু নেটওয়ার্ক এক্সেস টাস্ক বিষয়ে আলোচনা করে যেটা আপনি আপনার ডাটা ট্রান্সফার কোডের মধ্যে অন্তর্ভুক্ত করতে পারেন।

ফ্রেমওয়ার্কে সিঙ্ক অ্যাডাপটর একত্রিত করা

এখন আপনার সিঙ্ক অ্যাডাপটর উপাদানের মধ্যে আবদ্ধ ডাটা ট্রান্সফার কোড আছে, কিনুত আপনাকে আপনার কোডে প্রবেশগম্যতা সহ ফ্রেমওয়ার্ক সরবরাহ করতে হবে। এটা করতে আপনার একটি বাউন্ড Service তৈরী করা প্রয়োজন যা সিঙ্ক অ্যাডাপটর থেকে ফ্রেমওয়ার্কে একটি বিশেষ অ্যান্ড্রয়েড বাইন্ডার অবজেক্ট পাস করে। এই বাইন্ডার অবজেক্ট দিয়ে ফ্রেমওয়ার্ক `onPerformSync()` পদ্ধতি আহ্বান করতে পারে এবং ডাটা এটাতে পাস করে।

সার্ভিসের `onCreate()` পদ্ধতির মধ্যে একটি সিঙ্গেলটোন হিসাবে আপনার সিঙ্ক অ্যাডাপটর উপাদান ইনস্টেনসিয়েট করুন। `onCreate()` এর মধ্যে উপাদান ইনস্টেনসিয়েট করার মাধ্যমে, যতক্ষণ সার্ভিস শুরু না হয় আপনি এটা তৈরী করা মূলতবী করতে পারেন, যা ঘটবে যখন ফ্রেমওয়ার্ক আপনার ডাটা ট্রান্সফার প্রথমে রান করার চেষ্টা করে। সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্কের সক্রিয় করা বা পরিকল্পনা করার জবাবে আপনার সিঙ্ক অ্যাডাপটরের বহুমুখি কর্মসম্পাদন সারিবদ্ধ করার ক্ষেত্রে আপনার একটি থ্রেড-সেফ ম্যানারের মধ্যে উপাদান ইনস্টেনসিয়েট করা প্রয়োজন।

উদাহরণস্বরূপ, নীচের চিত্রটি আপনাকে দেখায় কীভাবে একটি ক্লাস তৈরী করতে হয় যা বাউন্ড সার্ভিস বাস্তবায়ন করে, আপনার সিঙ্ক অ্যাডাপটর উপাদান ইনস্টেনসিয়েট করে, এবং অ্যান্ড্রয়েড বাইন্ডার অবজেক্ট লাভ করে:

```
package com.example.android.syncadapter;
/**
 * Define a Service that returns an IBinder for the
 * sync adapter class, allowing the sync adapter framework to call
 * onPerformSync().
 */
public class SyncService extends Service {
    // Storage for an instance of the sync adapter
    private static SyncAdapter sSyncAdapter = null;
    // Object to use as a thread-safe lock
    private static final Object sSyncAdapterLock = new Object();
    /*
     * Instantiate the sync adapter object.
     */
    @Override
    public void onCreate() {
        /*
         * Create the sync adapter as a singleton.
         * Set the sync adapter as syncable
         * Disallow parallel syncs
         */
        synchronized (sSyncAdapterLock) {
            if (sSyncAdapter == null) {
                sSyncAdapter = new SyncAdapter(getApplicationContext(), true);
            }
        }
    }
    /**
     * Return an object that allows the system to invoke
     * the sync adapter.
     */
    @Override
    public IBinder onBind(Intent intent) {
```

```
/*
 * Get the object that allows external processes
 * to call onPerformSync(). The object is created
 * in the base class code when the SyncAdapter
 * constructors call super()
 */
return sSyncAdapter.getSyncAdapterBinder();
}
}
```

নোট: একটি সিন্ক অ্যাডাপটরের জন্য একটি বাউন্ডের বিস্তারিত উদাহরনের জন্য উদাহরণ অ্যাপ দেখুন।

ফ্রেমওয়ার্ক কর্তৃক চাওয়া একাউন্ট যুক্ত করুন

সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক চায় প্রতিটা সিঙ্ক অ্যাডাপটরের একটি একাউন্ট টাইপ থাক। আপনি Add the Authenticator Metadata File অধ্যায়ে আপনি একাউন্ট টাইপ ভ্যালু ডিক্লেয়ার করেছিলেন। এখন আপনাকে অ্যান্ড্রয়েড সিস্টেমের মধ্যে এই একাউন্ট টাইপ সেট আপ করতে হবে। একাউন্ট টাইপ সেটআপ করতে, একটি ডামি (প্রতিকৃতি) একাউন্ট যুক্ত করুন যা addAccountExplicitly() কল করার মাধ্যমে একাউন্ট টাইপ ব্যবহার করে।

পদ্ধতি কল করার সবচেয়ে ভালো জায়গা হচ্ছে আপনার অ্যাপের প্রারম্ভিক কার্যক্রমের onCreate() পদ্ধতির মধ্যে। নিচের কোড চিত্রটি এটা কীভাবে করতে হয় তা দেখায়:

```
public class MainActivity extends FragmentActivity {
    ...
    ...
    // Constants
    // The authority for the sync adapter's content provider
    public static final String AUTHORITY = "com.example.android.datasync.provider"
    // An account type, in the form of a domain name
    public static final String ACCOUNT_TYPE = "example.com";
    // The account name
    public static final String ACCOUNT = "dummyaccount";
    // Instance fields
    Account mAccount;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ...
        // Create the dummy account
        mAccount = CreateSyncAccount(this);
        ...
    }
    ...
    /**
     * Create a new dummy account for the sync adapter
     *
     * @param context The application context
     */
    public static Account CreateSyncAccount(Context context) {
        // Create the account type and default account
        Account newAccount = new Account(
            ACCOUNT, ACCOUNT_TYPE);
        // Get an instance of the Android account manager
        AccountManager accountManager =
            (AccountManager) context.getSystemService(
                ACCOUNT_SERVICE);

        /*
         * Add the account and account type, no password or user data
         * If successful, return the Account object, otherwise report an error.
         */
        if (accountManager.addAccountExplicitly(newAccount, null, null)) {
            /*
             * If you don't set android:syncable="true" in
             * in your <provider> element in the manifest,
             * then call context.setIsSyncable(account, AUTHORITY, 1)
             * here.
             */
        }
    }
}
```

```
    } else {
        /*
         * The account exists or some other error occurred. Log this, report it,
         * or handle it internally.
         */
    }
}
...
}
```

সিঙ্ক অ্যাডাপটর মেটাডেটা ফাইল যুক্ত করুন

ফ্রেমওয়ার্কের মধ্যে আপনার সিঙ্ক অ্যাডাপটর উপাদান প্লাগ করতে আপনার মেটাডেটা সহকারে ফ্রেমওয়ার্ক সরবরাহ করা যা উপাদান আলোচনা করে এবং অতিরিক্ত ফ্ল্যাগস প্রদান করে। আপনি আপনার সিঙ্ক অ্যাডাপটরের জন্য যে একাউন্ট টাইপ তৈরী করেছেন মেটাডেটা তা মেটাডেটা নির্দিষ্ট করে, আপনার অ্যাপের সাথে সম্পৃক্ত কনটেন্ট প্রভাইডার অথরিটি ডিক্লেয়ার করে, সিঙ্ক অ্যাডাপটরের সাথে জরিত সিস্টেম ইউজার ইন্টারফেসের একটি অংশকে নিয়ন্ত্রণ করে এবং অন্য সিঙ্ক সম্পর্কিত ফ্ল্যাগ ডিক্লেয়ার করে। আপনার অ্যাপ প্রজেক্টেও মধ্যকার `/res/xml/` ডিরেক্টরির মধ্যে স্টোর করা একটি বিশেষ XML ফাইলের মধ্যে এই মেটাডেটা ডিক্লেয়ার করুন। আপনি এই ফাইলের যে কোন নাম দিতে পারেন, যদি এটাকে `syncaapter` নামে ডাকা হয়।

এই XML ফাইল একটি একক XML এলিমেন্ট `< sync-adapter>` ধারণ করে যার নিম্নোক্ত বিশেষত্ব গুলো রয়েছে:

android:contentAuthority

আপনার কনটেন্ট প্রভাইডারের জন্য URI অথরিটি পূর্ববর্তী অনুশীলনীতে (Creating a Stub Content Provider) আপনি যদি আপনার অ্যাপের জন্য একটি স্টাব কনটেন্ট প্রভাইডার তৈরী করে থাকেন, আপনি অ্যাপ মেনিফেস্টে যে `< provider>` এলিমেন্ট যুক্ত করেছেন তার মধ্যে এট্রিবিউট `android:authorities` জন্যে যে ভ্যালু নির্দিষ্ট করেছেন তা ব্যবহার করুন। এই এট্রিবিউট `Declare the Provider in the Manifest` অধ্যায়ে আরও বিস্তারিতভাবে আলোচনা করা হয়েছে। আপনি যদি সিঙ্ক অ্যাডাপটর দিয়ে একটি কনটেন্ট প্রভাইডার থেকে সার্ভারে ডাটা ট্রান্সফার করতে থাকেন, এই ভ্যালু একই রকম হওয়া উচিত যেমন আপনি ডাটার জন্য কনটেন্ট টজও ব্যবহার করছেন সেরকম। এই ভ্যালু একটি অথরিটিও যা `< provider>` এলিমেন্টের `android:authorities` এট্রিবিউটের মধ্যে যা আপনি নির্দিষ্ট করেছেন যা আপনার অ্যাপ মেনিফেস্টের মধ্যে আপনার প্রভাইডারকে ডিক্লেয়ার করে।

android:accountType

সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক দ্বারা চাওয়া একাউন্ট টাইপ। যখন আপনি অথেনটিকেটর মেটাডেটা ফাইল তৈরী করেছিলেন তখন যে একাউন্ট টাইপ ভ্যালু প্রদান করেছিলেন ভ্যালুটিকে অবশ্যই সেই রকম হতে হবে, যেভাবে `Add the Authenticator Metadata File` অধ্যায়ে আলোচনা করা হয়েছে। এটা `Add the Account Required by the Framework` অধ্যায়ে মধ্যে কোড চিত্রের (কোড স্নিপেট) মধ্যে কনস্টেন্ট `ACCOUNT_TYPE` এর জন্য নির্দিষ্ট ভ্যালুও।

Settings attributes

`android:userVisible` সিঙ্ক অ্যাডাপটরের একাউন্ট টাইপের দৃশ্যমানতা সেট করুন। বাই ডিফল্ট একাউন্ট টাইপের সাথে সম্পৃক্ত একাউন্ট আইকন এবং লেভেল সিস্টেমের সেটিংস অ্যাপের `Accounts` অধ্যায়ের মধ্যে দৃশ্যমান, আপনার উচিত আপনার সিঙ্ক অ্যাডাপটর অদৃশ্য করা যতক্ষণ না আপনার একটি একাউন্ট টাইপ বা ডোমেইন থাকছে যা আপনার অ্যাপের সাথে সহজেই সম্পৃক্ত হয়। আপনি যদি আপনার একাউন্ট টাইপ অদৃশ্য করেন, আপনি তখনও ইউজারকে আপনার অ্যাপের একটি একটিভিটির মধ্যে একটি ইউজার ইন্টারফেস দিয়ে আপনার সিঙ্ক অ্যাডাপটর

নিয়ন্ত্রণ করতে দিতে পারবেন।

android:supportsUploading

আপনাকে ক্লাউডে ডাটা আপলোড করতে দেয়। আপনার অ্যাপ যদি শুধু ডাটা ডাউনলোড করে এটাকে false এ সেট করুন।

android:allowParallelSyncs

একই সময়ে রান করতে আপনার সিন্ক অ্যাডাপটর উপাদানের মাল্টিপল ইনসটেন্স অনুমোদন। এটা ব্যবহার করুন যদি আপনার অ্যাপ মাল্টিপল (বহুবিধ) ইউজার একাউন্ট সাপোর্ট করে এবং সমান্তরালভাবে আপনি মাল্টিপল ইউজারকে ডাটা ট্রান্সফার করতে দিতে চান। ফ্ল্যাগের কোন প্রভাব নেই যদি আপনি কখনই মাল্টিপল ডাটা ট্রান্সফার না করেন।

android:isAlwaysSyncable

সিন্ক অ্যাডাপটর ফ্রেমওয়ার্কে নির্দেশ করে যে এটা আপনার সিন্ক অ্যাডাপটরকে রান করবে আপনি যখনই নির্দিষ্ট করেন। আপনি যদি প্রোগ্রামেটিক্যালি নিয়ন্ত্রণ করতে চান, যখন আপনার সিন্ক অ্যাডাপটর রান করতে পারবে, এই ফ্ল্যাগস ভধষংব এ সেট করুন, এবং তারপর সিন্ক অ্যাডাপটর রান করতে requestSync()কল করুন। একটি সিন্ক অ্যাডাপটর রান করার বিষয়ে বিস্তারিত জানতে Running a Sync Adapter দেখুন।

নিচের উদাহরণ দেখায়, একটি সিন্ক অ্যাডাপটরের জন্য XML যা একটি একক প্রতিক্রাপ (ডামি) একাউন্ট ব্যবহার করে এবং শূধুমাত্র ডাউনলোড করে।

```
<?xml version="1.0" encoding="utf-8"?>
<sync-adapter
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:contentAuthority="com.example.android.datasync.provider"
    android:accountType="com.android.example.datasync"
    android:userVisible="false"
    android:supportsUploading="false"
    android:allowParallelSyncs="false"
    android:isAlwaysSyncable="true"/>
```

মেনিফেস্টের মধ্যে সিঙ্ক অ্যাডপটর ডিক্লেয়ার করা

যখনই আপনি আপনার অ্যাপে সিঙ্ক অ্যাডাপটর উপাদান যুক্ত করবেন, আপনাকে উপাদান ব্যবহার সম্পর্কিত পারমিশন রিকোয়েস্ট করতে হবে, আপনি যে বাউন্ড Service যুক্ত করেছেন তাকে ডিক্লেয়ার করতে হবে।

যেহেতু সিঙ্ক অ্যাডাপটর উপাদান কোড রান করে যা নেটওয়ার্ক এবং ডিভাইসের মধ্যে ডাটা ট্রান্সফার করে, আপনাকে ইন্টারনেটে প্রবেশ করার জন্য পারমিশন রিকোয়েস্ট করা প্রয়োজন। উপরনুত সিঙ্ক অ্যাডাপটর সেটিং পড়তে এবং লেখার জন্য আপনার অ্যাপের পারমিশন রিকোয়েস্ট করা প্রয়োজন, সুতরাং আপনি আপনার অ্যাপের মধ্যে অন্য উপাদান থেকে সিঙ্ক অ্যাডাপট কে প্রোগ্রামেটিক্যালী নিয়ন্ত্রণ করতে পারবেন। আপনার একটি বিশেষ পারমিশন রিকোয়েস্ট করতে হবে যা আপনি Creating a Stub Authenticator অনুশীলনীতে যে অথেনটিকেটর উপাদান তৈরী করেছিলেন তা আপনার অ্যাপকে ব্যবহার করতে দেয়।

এই পারমিশনগুলোকে রিকোয়েস্ট করার জন্য `<manifest>` এর চাইল্ড এলিমেন্ট হিসাবে নিচের বিষয়গুলো আপনার অ্যাপ মেনিফেস্টে যুক্ত করুন:

android.permission.INTERNET

ইন্টারনেটে প্রবেশ করার জন্য সিঙ্ক অ্যাডাপটর কোড অনুমোদন করা যাতে এটা ডিভাইস থেকে সার্ভারে ডাটা ডাউনলোড এবং আপলোড করতে পারে। আপনি যদি পূর্বে এই পারমিশন রিকোয়েস্ট করে থাকেন আপনাকে নতুন করে এই এটাকে যুক্ত করতে হবেনা।

android.permission.READ_SYNC_SETTINGS

চলতি সিঙ্ক অ্যাডাপটর সেটিং পড়ার জন্য আপনার অ্যাপকে অনুমোদন করে। উদাহরণস্বরূপ, `getIsSyncable()` কল করার জন্য আপনার এই পারমিশন প্রয়োজন।

android.permission.WRITE_SYNC_SETTINGS

সিঙ্ক অ্যাডাপটর সেটিং নিয়ন্ত্রণ করতে আপনার অ্যাপকে অনুমোদন করে। `addPeriodicSync()` ব্যবহার করে মেয়াদকালীন সিঙ্ক অ্যাডাপটর রানস সেটআপ করার জন্য আপনার এই পারমিশন প্রয়োজন। `requestSync()` কল করা এই পারমিশনের জন্য প্রয়োজনীয় নয়। সিঙ্ক অ্যাডপটর রান করা বিষয়ক আরও তথ্যের জন্য Running A Sync Adapter দেখুন।

android.permission.AUTHENTICATE_ACCOUNTS

আপনি Creating a Stub Authenticator অনুশীলনীতে যে অথেনটিকেটর উপাদান তৈরী করেছিলেন তা আপনার ব্যবহার করতে দেয়।

নিচের চিত্রটি দেখায় কীভাবে পারমিশন যুক্ত করতে হয়:

```

<manifest>
...
    <uses-permission
        android:name="android.permission.INTERNET"/>
    <uses-permission
        android:name="android.permission.READ_SYNC_SETTINGS"/>
    <uses-permission
        android:name="android.permission.WRITE_SYNC_SETTINGS"/>
    <uses-permission
        android:name="android.permission.AUTHENTICATE_ACCOUNTS"/>
...
</manifest>

```

চূড়ান্তভাবে, বাউন্ড Service ডিক্লেয়ার করতে, যা ফ্রেমওয়ার্ক আপনার সিন্ক অ্যাডাপটরের সাথে যোগাযোগ করতে ব্যবহার করে, < application> এর একটি চাইল্ড এলিমেন্ট হিসাবে আপনার অ্যাপ মেনিফেস্টে নিচের XML যুক্ত করুন:

```

<service
    android:name="com.example.android.datasync.SyncService"
    android:exported="true"
    android:process=":sync">
    <intent-filter>com.example.android.datasync.provider
        <action android:name="android.content.SyncAdapter"/>
    </intent-filter>
    <meta-data android:name="android.content.SyncAdapter"
        android:resource="@xml/syncadapter" />
</service>

```

< intent-filter> এলিমেন্ট একটি ফিল্টার সেটআপ করে যা ইনটেন্ট একশন android.content.SyncAdapter দ্বারা সক্রিয় হয়, সিন্ক অ্যাডাপটর রান করানোর জন্য সিস্টেম কর্তৃক পাঠানো হয়। যখন ফিল্টার সক্রিয় করনো হয়, আপনি যে বাউন্ড সার্ভিস তৈরী করেছেন সিস্টেম তা শুরু করে, এই উদাহরণে যেটা হচ্ছে SyncService। এট্রিবিউট android:exported="true" টি Service এ প্রবেশ করতে প্রসেসকে অনুমোদন করে আপনার অ্যাপকে (সিস্টেম সহ) নয়। এট্রিবিউট android:process=":sync" টি সিস্টেমকে একটি বৈশ্বিক শেয়ার করা প্রসেস যার নাম sync তার মধ্যে Service কে রান করতে বলে। যদি আপনার যদি আপনার অ্যাপের মধ্যে মাল্টিপল সিন্ক অ্যাডপটর থেকে থাকে যা এই প্রসেস শেয়ার করতে পারে, এটা ওভারহেড কমিয়ে দেয়।

< meta-data> এলিমেন্ট পূর্বে তৈরী করেছেন এমন সিন্ক অ্যাডাপটর মেটাডেটা XML ফাইলের নাম প্রদান করে। android:name এট্রিবিউট নির্দেশ করে যে এই মেটাডেটা সিন্ক অ্যাডাপটর ফ্রেমওয়ার্কের জন্য। android:resource এলিমেন্ট মেটাডেটা ফাইলের নাম নির্দিষ্ট করে।

এখন আপনার কাছে সিন্ক অ্যাডাপটরের জন্য সকল উপাদান আছে। পরবর্তী অনুশীলনী আপনাকে দেখাবে কীভাবে আপনার সিন্ক অ্যাডাপটরকে রান করতে সিন্ক অ্যাডাপটর ফ্রেমওয়ার্ককে বলতে হয়, একটি ইভেন্টের প্রতিক্রিয়ায় নাকি বা একটি নিয়মিত সময়সূচী অনুযায়ী।

একটি সিঙ্ক অ্যাডাপটর রান করা

(<http://developer.android.com/training/sync-adapters/running-sync-adapter.html>)

এই ক্লাসের পূর্ববর্তী অনুশীলনীতে, আপনি শিখেছেন কীভাবে একটি সিঙ্ক অ্যাডাপটর উপাদান তৈরী করতে হয় যা ডাটা ট্রান্সফার কোডের মধ্যে থাকে, এবং কীভাবে অতিরিক্ত উপাদান যোগ করতে হয় যা সিস্টেমের মধ্যে আপনাকে সিঙ্ক অ্যাডাপটর প্লাগ করতে দেয়। একটি অ্যাপ ইনস্টল করার জন্য যা প্রয়োজন এখন আপনার তা আছে যা একটি সিঙ্ক অ্যাডাপটর অন্তর্ভুক্ত করে, কিন্ত আপনি যে কোডগুলো দেখেছেন তার কোন টাই আসলে তা সিঙ্ক অ্যাডাপটর রান করাতে পারে না।

আপনার উচিত একটি সময়সূচির উপর ভিত্তি করে বা কিছু ইভেন্টের পরোক্ষ ফল হিসাবে আপনার সিঙ্ক অ্যাডাপটর রান করার চেষ্টা করা। উদাহরণস্বরূপ, আপনি হয়তো হয় একটি নির্দিষ্ট সময় পর বা দিনের একটি নির্দিষ্ট সময়ে একটি নিয়মিত সময়সূচির (শিডিউল) উপর আপনার সিঙ্ক অ্যাডাপটর রান করতে চাইতে পারেন। আপনি হয়তো আপনার সিঙ্ক অ্যাডাপটর রান করতে চাইতে পারেন যখন ডিভাইসে ডাটা স্টোর করার ক্ষেত্রে কোন পরিবর্তন থাকে তখনও। আপনার উচিত সিঙ্ক অ্যাডাপটর একটি ইউজার একশনের প্রত্যক্ষ ফল হিসাবে রান করাটা পরিহার করা, কারন এটা করার মাধ্যমে আপনি সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্কের শিডিউল করার সামর্থের পূর্ণ সুবিধা পাবেন না। উদাহরণস্বরূপ বলা যায়, আপনার উচিত আপনার ইউজার ইন্টারফেসের মধ্যে একটি রিফ্রেশ বাটন যুক্ত করার বিষয়টা পরিহার করা।

আপনার সিঙ্ক অ্যাডাপটর রান করার জন্য নিচের কোন একটি বেছে নিতে পারেন:

যখন সার্ভার ডাটা পরিবর্তন করে

একটি সার্ভার থেকে আসা মেসেজের প্রতিক্রিয়া হিসাবে সিঙ্ক অ্যাডাপটর রান করা, নির্দেশ করে যে সার্ভার ভিত্তিক ডাটা পরিবর্তন হয়েছে। এই পছন্দ আপনাকে কার্যসম্পাদনের কোন রূপ পরিবর্তন ছাড়াই বা সার্ভার পোল করার মাধ্যমে ব্যাটারির আয়ুষ্কাল অপচয় করা ছাড়াই সার্ভার থেকে ডিভাইসে ডাটা রিফ্রেশ করতে দেয়

যখন ডিভাইস ডাটা পরিবর্তন করে

একটি সিঙ্ক অ্যাডাপটর রান করা যখন ডিভাইসে ডাটা পরিবর্তন হয়। এই পছন্দ আপনাকে ডিভাইস থেকে পরিবর্তিত ডাটা সার্ভারে পাঠাতে দেয়, এবং বিশেষভাবে উপকারী হয় যদি আপনার নিশ্চিত করার প্রয়োজন হয় যে সার্ভারের সব সময় সর্বশেষ তথ্য রয়েছে। এই পছন্দ বাস্তবায়নের ক্ষেত্রে অকপট যদি আপনি আপনার কনটেন্ট প্রভাইডারে যথাযথই ডাটা স্টোর করেন। যদি আপনি একটি স্টাব কনটেন্ট প্রভাইডার করতে থাকেন, ডাটা পরিবর্তন খুজে বের করাটা কঠিন হয়ে যেতে পারে।

যখন সিস্টেম একটি নেটওয়ার্ক মেসেজ পাঠায়

একটি সিঙ্ক অ্যাডাপটর রান করা যখন অ্যান্ড্রয়েড সিস্টেম একটি নেটওয়ার্ক মেসেজ পাঠায় যা ঐচ্ছিক/ওচ কানেকশন ওপেন ধারণ করে: এই মেসেজ হচ্ছে নেটওয়ার্কিং ফ্রেমওয়ার্কের একটি মৌলিক অংশ। এই পছন্দ ব্যবহার করাটা সিঙ্ক অ্যাডাপটর স্বয়ংক্রিয়ভাবে রান করার একটা উপায়

যা বিরতী (ইন্টারভ্যাল)-ভিত্তিক সিস্টেম অ্যাডাপটর রানের সাথে এটার একাত্মত ব্যবহার বিবেচনা করুন।

নিয়মিত বিরতীতে (ইন্টারভ্যাল)

আপনি যে বিরতী (ইন্টারভ্যাল) পছন্দ করেছেন তার সমাপ্তির পরে একটি সিস্টেম অ্যাডাপটর রান করা বা এটাকে প্রতিদিন একটি নির্দিষ্ট সময়ে রান করুন।

প্রয়োজন অনুসারে

একজন ইউজারের একশনের জবাবে সিস্টেম অ্যাডাপটর রান করা। যাহোক, সবচেয়ে ভালো ইউজার এক্সপেরিয়েন্স সরবরাহ করতে প্রাথমিকভাবে আপনার আরও স্বয়ংক্রিয় (অটোমেটেড) পছন্দের উপর নির্ভর করা উচিত। স্বয়ংক্রিয় (অটোমেটেড) পছন্দ ব্যবহার করার মাধ্যমে আপনি ব্যাটারি এবং নেটওয়ার্ক রিসোর্স সংরক্ষণ করে।

এই অনুশীলনীর পরবর্তীতে প্রতিটা পছন্দের বিষয়ে বিস্তারিত ভাবে আলোচনা করা হবে।

একটি সিন্ক অ্যাডাপটর রান করা যখন সার্ভার ডাটা পরিবর্তন করে

যদি আপনার অ্যাপ একটি সার্ভার থেকে ডাটা ট্রান্সফার করে এবং সার্ভার ডাটা বারংবার পরিবর্তন করে, আপনি ডাটা পরিবর্তনের প্রতিক্রিয়ায় ডাউনলোড করতে একটি সিন্ক অ্যাডাপটর ব্যবহার করুন। সিন্ক অ্যাডাপটর রান করতে, সার্ভারকে আপনার অ্যাপের মধ্যে একটি BroadcastReceiver এ একটি বিশেষ মেসেজ পাঠাতে হবে। এই মেসেজের জবাবে, আপনার সিন্ক অ্যাডাপটর রান করতে সিন্ক অ্যাডাপটর ফ্রেমওয়ার্ককে সিগন্যাল দিতে ContentResolver.requestSync() কল করুন।

গুগল ক্লাউড মেসেজিং (GCM) (<http://developer.android.com/google/gcm/index.html>) এই মেসেজ সিস্টেম কাজ তৈরী করার জন্য প্রয়োজনীয় সার্ভার এবং ডিভাইস উপাদান সরবরাহ করে। ট্রান্সফার সক্রিয় করতে GCM ব্যবহার স্ট্যাটাসের জন্য সার্ভারকে পলিং করার চেয়ে আরও নির্ভরযোগ্য এবং কর্মদক্ষ। যখন পোলিং একটি Service আকাঙ্ক্ষা করে যা সবসময় সক্রিয় থাকে, GCM একটি ব্যবহার BroadcastReceiver করে যা সক্রিয় হয় যখন একটি মেসেজ পৌঁছায়। যখন পোলিং নিয়মিত বিরতিতে ব্যাটারি পাওয়ার ব্যবহার করে এমনকি যদিও আপডেট বিদ্যমান আছে, GCM শুধু মেসেজ পাঠায় যখন প্রয়োজন হয়।

নোট: যদি আপনার সিন্ক অ্যাডাপটর সক্রিয় করতে সকল ডিভাইসে ব্রডকাস্ট করার মাধ্যমে আপনি GCM ব্যবহার করেন যেখানে অ্যাপ ইনস্টল করা হয়েছে, মনে রাখবেন যে তারা আপনার মেসেজ মোটামুটিভাবে একই সময়ে গ্রহণ করে। এই অবস্থা একই সময়ে রান করতে আপনার সিন্ক অ্যাডাপটরের মাল্টিপল ইনস্ট্যান্স করতে পারে, সার্ভার এবং নেটওয়ার্ক ওভারলোড করে। সকল ডিভাইসে ব্রডকাস্টের জন্য এই পরিস্থিতি পরিহার করতে, আপনার উচিত একটি সময়কালের জন্য সিন্ক অ্যাডাপটরের শুরু বিবেচনা করা যা প্রতি টা ডিভাইসের জন্য ইউনিক (অদ্বিতীয়)।

নিম্নোক্ত কোড চিত্রটি আপনাকে দেখাবে কীভাবে একটি ইনকামিং GCM মেসেজের জবাবে requestSync() রান করা হয়:

```
public class GcmBroadcastReceiver extends BroadcastReceiver {
    ...
    // Constants
    // Content provider authority
    public static final String AUTHORITY = "com.example.android.datasync.provider"
    // Account type
    public static final String ACCOUNT_TYPE = "com.example.android.datasync";
    // Account
    public static final String ACCOUNT = "default_account";
    // Incoming Intent key for extended data
    public static final String KEY_SYNC_REQUEST =
        "com.example.android.datasync.KEY_SYNC_REQUEST";
    ...
    @Override
    public void onReceive(Context context, Intent intent) {
        // Get a GCM object instance
        GoogleCloudMessaging gcm =
            GoogleCloudMessaging.getInstance(context);
        // Get the type of GCM message
```

```

String messageType = gcm.getMessageType(intent);
/*
 * Test the message type and examine the message contents.
 * Since GCM is a general-purpose messaging system, you
 * may receive normal messages that don't require a sync
 * adapter run.
 * The following code tests for a a boolean flag indicating
 * that the message is requesting a transfer from the device.
 */
if (GoogleCloudMessaging.MESSAGE_TYPE_MESSAGE.equals(messageType)
    &&
    intent.getBooleanExtra(KEY_SYNC_REQUEST)) {
    /*
     * Signal the framework to run your sync adapter. Assume that
     * app initialization has already created the account.
     */
    ContentResolver.requestSync(ACCOUNT, AUTHORITY, null);
    ...
}
...
}
...
}

```

সিঙ্ক অ্যাডাপটর রান করা যখন কনটেন্ট প্রভাইডার ডাটা পরিবর্তন করে

যদি আপনার অ্যাপ একটি কনটেন্ট প্রভাইডারের মধ্যে ডাটা সংগ্রহ করে, এবং আপনি সার্ভার আপডেট করতে চান, যখনই আপনি প্রভাইডার আপডেট করেন, আপনি আপনার সিঙ্ক অ্যাডাপটর স্বয়ংক্রিয়ভাবে রান করতে আপনার অ্যাপ সেটআপ করতে পারেন। এটা করতে কনটেন্ট প্রভাইডারের জন্য আপনি একটি অবজার্ভার রেজিস্টার করেন। যখন আপনার কনটেন্ট প্রভাইডারের ডাটা পরিবর্তন করে, কনটেন্ট প্রভাইডার ফ্রেমওয়ার্ক অবজার্ভার কল করে। অবজার্ভারের মধ্যে ফ্রেমওয়ার্ক কে আপনার সিঙ্ক অ্যাডাপটর রান করতে বলতে `requestSync()` কল করুন।

নোট: যদি আপনি স্টার কনটেন্ট প্রভাইডার ব্যবহার করেন, কনটেন্ট প্রভাইডারে আপনার কোন ডাটা নাই এবং `onChange()` কে কখনই কল করা হয় না। এই ক্ষেত্রে, ডিভাইস ডাটাতে পরিবর্তন চিহ্নিত করতে আপনার নিজস্ব প্রক্রিয়া (মেকানিজম) প্রদান করতে হবে। এই প্রক্রিয়া `requestSync()` কল করার জন্য দায়িত্বপ্রাপ্ত যখন ডাটা পরিবর্তন করে।

আপনার কনটেন্ট প্রভাইডারের জন্য একটি অবজার্ভার তৈরী করতে, ক্লাস `ContentObserver` প্রসারিত করুন এবং এর `onChange()` পদ্ধতির উভয় রূপই বাস্তবায়ন করুন। `onChange()` এর মধ্যে, সিঙ্ক অ্যাডাপটর শুরু করতে `requestSync()` কল করুন।

অবজার্ভার রেজিস্টার করতে, `registerContentObserver()` এ একটি কলের মধ্যে একটি আর্গুমেন্ট হিসাবে এটাকে পাস করুন। এই কলে, আপনি যে ডাটা দেখতে চান তার জন্য একটি কনটেন্ট URI এ আপনাকে পাস করতে হবে। কনটেন্ট প্রভাইডার ফ্রেমওয়ার্কটি এই ওয়াচ URI কে কনটেন্ট URI এ তুলনা করে যা `ContentResolver` পদ্ধতিতে আর্গুমেন্ট হিসাবে পাস হয়েছে যা আপনার প্রভাইডার পরিমিত করে, যেমন, `ContentResolver.insert()`। যদি সেখানে কোন সমরূপ কিছু থাকে, আপনার `ContentObserver.onChange()` এর বাস্তবায়ন কে কল করা হয়।

নিচের কোড চিত্রটি দেখাচ্ছে যে কীভাবে আপনি একটি `ContentObserver` নির্ধারণ করবেন যা `requestSync()` কল করে যখন একটি টেবিল পরিবর্তন করে:

```
public class MainActivity extends FragmentActivity {  
    ...  
    // Constants  
    // Content provider scheme  
    public static final String SCHEME = "content://";  
    // Content provider authority  
    public static final String AUTHORITY = "com.example.android.datasync.provider";  
    // Path for the content provider table  
    public static final String TABLE_PATH = "data_table";  
    // Account  
    public static final String ACCOUNT = "default_account";  
    // Global variables  
    // A content URI for the content provider's data table  
    Uri mUri;  
    // A content resolver for accessing the provider  
    ContentResolver mResolver;
```



```

...
public class TableObserver extends ContentObserver {
    /*
     * Define a method that's called when data in the
     * observed content provider changes.
     * This method signature is provided for compatibility with
     * older platforms.
     */
    @Override
    public void onChange(boolean selfChange) {
        /*
         * Invoke the method signature available as of
         * Android platform version 4.1, with a null URI.
         */
        onChange(selfChange, null);
    }
    /*
     * Define a method that's called when data in the
     * observed content provider changes.
     */
    @Override
    public void onChange(boolean selfChange, Uri changeUri) {
        /*
         * Ask the framework to run your sync adapter.
         * To maintain backward compatibility, assume that
         * changeUri is null.
         */
        ContentResolver.requestSync(ACCOUNT, AUTHORITY, null);
    }
    ...
}
...
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
    // Get the content resolver object for your app
    mResolver = getContentResolver();
    // Construct a URI that points to the content provider data table
    mUri = new Uri.Builder()
        .scheme(SCHEME)
        .authority(AUTHORITY)
        .path(TABLE_PATH)
        .build();

    /*
     * Create a content observer object.
     * Its code does not mutate the provider, so set
     * selfChange to "false"
     */
    TableObserver observer = new TableObserver(false);
    /*
     * Register the observer for the data table. The table's path
     * and any of its subpaths trigger the observer.
     */
    mResolver.registerContentObserver(mUri, true, observer);
    ...
}
...
}

```

একটি নেটওয়ার্ক মেসেজের পর সিঙ্ক অ্যাডাপটর রান করা

যখন একটি নেটওয়ার্ক কানেকশন পাওয়া যায়, অ্যান্ড্রয়েড সিস্টেম ডিভাইসের TCP/IP কানেকশন ওপেন ধরে রাখতে কিছু সেকেন্ড পরপরই একটি মেসেজ পাঠায়। এই মেসেজ প্রতিটা অ্যাপের ContentResolver এ যায়। setSyncAutomatically() কল করার মাধ্যমে, আপনি সিঙ্ক অ্যাডাপটর রান করতে পারবেন যখনই ContentResolver মেসেজ গ্রহণ করে।

যখন নেটওয়ার্ক মেসেজ পাঠানো হয় তখন আপনার সিঙ্ক অ্যাডাপটর রান করতে শিডিউল করে, আপনি নিশ্চিত করেন যে যখনই নেটওয়ার্ক সহজপ্রাপ্য হয় আপনার সিঙ্ক অ্যাডাপটর সবসময় রান করার জন্য শিডিউল করা থাকে। ডাটা পরিবর্তনের জবাবে আপনাকে একটি ডাটা ট্রান্সফার বল পূর্বক করতে হয় না, কিনুত আপনি নিশ্চিত করতে চান যে আপনার ডাটা নিয়মিতভাবে আপডেট হয় তখন এই পছন্দ ব্যবহার করুন। একইভাবে, যদি আপনি আপনার সিঙ্ক অ্যাডাপটরের জন্য কোন নির্দিষ্ট শিডিউল না চান, কিনুত আপনি চান এটা নিয়মিতভাবে রান করুক সেক্ষেত্রে এই পছন্দ ব্যবহার করতে পারেন।

যেহেতু পদ্ধতি setSyncAutomatically() টি addPeriodicSync() কে নিষ্ক্রিয় করে না, আপনার সিঙ্ক অ্যাডাপটর অল্প সময়ের মধ্যে বারবার সক্রিয় হতে পারে। যদি আপনি আপনার সিঙ্ক অ্যাডাপটর নিয়মিত শিডিউলে পর্যায়ক্রমে রান করতে চান, আপনার উচিত setSyncAutomatically() নিষ্ক্রিয় করা।

নিম্নের কোড চিত্রটি আপনাকে দেখাচ্ছে যে একটি নেটওয়ার্ক মেসেজের জবাবে সিঙ্ক অ্যাডাপটর রান করতে কীভাবে আপনার ContentResolver কনফিগার করতে হয়:

```
public class MainActivity extends FragmentActivity {  
    ...  
    // Constants  
    // Content provider authority  
    public static final String AUTHORITY = "com.example.android.datasync.provider";  
    // Account  
    public static final String ACCOUNT = "default_account";  
    // Global variables  
    // A content resolver for accessing the provider  
    ContentResolver mResolver;  
    ...  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        ...  
        // Get the content resolver for your app  
        mResolver = getContentResolver();  
        // Turn on automatic syncing for the default account and authority  
        mResolver.setSyncAutomatically(ACCOUNT, AUTHORITY, true);  
        ...  
    }  
    ...  
}
```

পর্যায়ক্রমে সিঙ্ক অ্যাডাপটর রান করা যখন

রানের মধ্যে অপেক্ষা করতে বা দিনের একটি সময়ে এটাকে রান করা বা উভয় করার জন্য একটি সময়কাল সেটিং করার মাধ্যমে আপনি পর্যায়ক্রমে আপনার সিঙ্ক অ্যাডাপটর রান করতে পারেন। পর্যায়ক্রমে আপনার সিঙ্ক অ্যাডাপটর রান করাটা আপনাকে আপনার সার্ভারের হালনাগাদ বিরতি (ইন্টারভ্যাল) মোটামুটিভাবে ম্যাচ করতে দেয়।

একইভাবে, আপনি ডিভাইস থেকে ডাটা আপলোড করতে পারেন যখন আপনার সার্ভার অপেক্ষাকৃত নিষ্ক্রিয় হয়, রাত্রে রান করতে আপনার সিঙ্ক অ্যাডাপটর শিডিউল করার মাধ্যমে। অধিকাংশ ইউজার রাতের বেলায় পাওয়ার অন বা প্লাগ ইন করা থেকে বিরত থাকে, সুতরাং এই সময় টি সাধারণভাবে সহজলভ্য। অধিকনূত, আপনার সিঙ্ক অ্যাডাপটরের মতো ডিভাইস একই সময়ে অন্য কাজ রান করে না। যদি আপনি এই পদ্ধতি গ্রহণ করে থাকেন, আপনাকে নিশ্চিত করতে হবে যে প্রতিটা ডিভাইস একটু ভিন্ন সময়ে একটি ডাটা ট্রান্সফার করে থাকে। যদি সকল ডিভাইস আপনার সিঙ্ক অ্যাডাপটর একই সময়ে রান করে, আপনি সম্ভবত আপনার সার্ভার এবং সেল প্রভাইডার ডাটা নেটওয়ার্ক ওভারলোড এ আছেন।

সাধারণভাবে, পর্যায়ক্রমিক রানের অর্থ থাকে যদি আপনার ইউজারের তাৎক্ষণিক আপডেটের প্রয়োজন না হয় কিনুত নিয়মিত আপডেট থাকার আকাঙ্ক্ষা করে। পর্যায়ক্রমিক রানের সে অর্থও থাকে যদি আপনি হালনাগাদ ডাটা এবং ক্ষুদ্র সিঙ্ক অ্যাডাপটরের কর্মদক্ষতার মধ্যে ভারসাম্য করতে চান যা ডিভাইস রিসোর্সের অতি-ব্যবহার করে না।

নিয়মিত বিরতিতে আপনার সিঙ্ক অ্যাডাপটর রান করতে `addPeriodicSync()` কল করুন। এটা একটি নির্দিষ্ট সময় অতিক্রান্ত হওয়ার পর রান করতে আপনার সিঙ্ক অ্যাডাপটরকে শিডিউল করে। যেহেতু সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক অন্য সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্কের কর্মসম্পাদনের জন্য দায়ী এবং ব্যাটারির দক্ষতা বাড়াতে চেষ্টা করে, অতিক্রান্ত সময় কয়েক সেকেন্ডে ভিন্ন হতে পারে। এছাড়াও, ফ্রেমওয়ার্ক আপনার সিঙ্ক অ্যাডাপটর রান করবে না যদি নেটওয়ার্ক না থাকে।

লক্ষ করুন যে `addPeriodicSync()` দিনের নির্দিষ্ট সময়ে সিঙ্ক অ্যাডাপটর রান করায় না। মোটামুটিভাবে প্রতিদিনের একই সময়ে সিঙ্ক অ্যাডাপটর রান করতে একটি ট্রিগার হিসাবে একটি পুনরাবৃত্ত করা এলার্ম ব্যবহার করুন। `AlarmManager` এর রেফারেন্স ডকুমেন্টেশনে পুনরাবৃত্ত করা এলার্ম বিষয়ে আরও বিস্তারিতভাবে আলোচনা করা হয়েছে। আপনি যদি দিনের সময়কালীন ট্রিগার সেট করতে পদ্ধতি `setInexactRepeating()` ব্যবহার করেন যার কিছু বৈচিত্র রয়েছে, আপনার উচিত তখনও শুরুর সময় দৈবচয়ন করা এটা নিশ্চিত করতে যে বিভিন্ন ডিভাইস থেকে রান করা সিঙ্ক অ্যাডাপটর বিস্তৃত হয়।

পদ্ধতি `addPeriodicSync()`, `setSyncAutomatically()` কে নিষ্ক্রিয় করে না, তাই আপনি অপেক্ষাকৃত কম সময়ে রান করা মাল্টিপল সিঙ্ক পেতে পারেন। এছাড়াও, শুধু কিছু সিঙ্ক অ্যাডাপটর নিয়ন্ত্রণ ফ্ল্যাগস `addPeriodicSync()` এর প্রতি কলের মধ্যে অনুমোদিত; যে ফ্ল্যাগের অনুমোদন নেই তা `addPeriodicSync()` এর জন্য রেফারেন্স ডকুমেন্টেশনে আলোচনা করা হয়েছে।

নিচের কোড চিত্রটি দেখায় কীভাবে পর্যায়ক্রমিক সিঙ্ক অ্যাডাপটর রানস শিডিউল করতে হয়:

```

public class MainActivity extends FragmentActivity {
    ...
    // Constants
    // Content provider authority
    public static final String AUTHORITY = "com.example.android.datasync.provider";
    // Account
    public static final String ACCOUNT = "default_account";
    // Sync interval constants
    public static final long MILLISECONDS_PER_SECOND = 1000L;
    public static final long SECONDS_PER_MINUTE = 60L;
    public static final long SYNC_INTERVAL_IN_MINUTES = 60L;
    public static final long SYNC_INTERVAL =
        SYNC_INTERVAL_IN_MINUTES *
        SECONDS_PER_MINUTE *
        MILLISECONDS_PER_SECOND;
    // Global variables
    // A content resolver for accessing the provider
    ContentResolver mResolver;
    ...
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        ...
        // Get the content resolver for your app
        mResolver = getContentResolver();
        /*
         * Turn on periodic syncing
         */
        ContentResolver.addPeriodicSync(
            ACCOUNT,
            AUTHORITY,
            null,
            SYNC_INTERVAL);
        ...
    }
    ...
}

```

চাহিদা অনুযায়ী সিঙ্ক অ্যাডাপটর রান করা

ইউজারের অনুরোধের জবাবে আপনার সিঙ্ক অ্যাডাপটর রান করা হচ্ছে একটি সিঙ্ক অ্যাডাপটর রান করার সবচেয়ে কম কাঙ্ক্ষিত কৌশল। ফ্রেমওয়ার্ক ব্যাটারি পাওয়ার সংরক্ষণ করার জন্য বিশেষ ভাবে তৈরী করা হয়েছে যখন এটা একটি শিডিউল অনুযায়ী সিঙ্ক অ্যাডাপটর রান করে। পছন্দ (অপশন) যা ডাটা পরিবর্তনের জবাবে একটি সিঙ্ক রান করে তা ব্যাটারির পাওয়ার কার্যকরীভাবে ব্যবহার করে, যেহেতু পাওয়ার নতুন ডাটা প্রদান করার কাজে ব্যবহৃত হয়।

তুলনামূলকভাবে, ইউজারকে তার চাহিদার উপর একটি সিঙ্ক রান করতে দেয়ার মানে হচ্ছে যে সিঙ্ক নিজে নিজেই রান করছে, যা নেটওয়ার্ক এবং পাওয়ার রিসোর্স ব্যবহারে অদক্ষ। এছাড়াও, চাহিদার উপর একটি সিঙ্ক রান করতে দেয়া ইউজারকে একটি সিঙ্ক রিকোয়েস্টের দিকে নিয়ে যায় এমনকি সেখানে কোন প্রমান নেই যে ডাটা পরিবর্তন হয়েছে এবং একটি সিঙ্ক রান করা যা ডাটাকে রিফ্রেশ করে না তা ব্যাটারি পাওয়ারের অকার্যকর ব্যবহার। সাধারণভাবে, আপনার অ্যাপের উচিত হয় একটি সিঙ্ক সক্রিয় করতে অন্য সিগনাল ব্যবহার করা বা ইউজার ইনপুট ছাড়াই নিয়মিত বিরতীতে তাদের শিডিউল করা হয়।

কিন্তু, যদি আপনি এখনও সিঙ্ক অ্যাডাপটর চাহিদার উপর রান করতে চান, একটি ম্যানুয়েল সিঙ্ক অ্যাডাপটর রান এর জন্য সিঙ্ক অ্যাডাপটর ফ্ল্যাগ সেট করুন, তারপর `ContentResolver.requestSync()` কল করুন।

নিচের ফ্ল্যাগগুলো দিয়ে চাহিদাকৃত ট্রান্সফারগুলো রান করুন:

`SYNC_EXTRAS_MANUAL`/সিঙ্ক এক্সট্রাস ম্যানুয়েল

একটি ম্যানুয়েল সিঙ্ক ফোর্স করে (বল প্রয়োগ)। সিঙ্ক অ্যাডাপটর ফ্রেমওয়ার্ক বিদ্যমান সেটিংকে এড়িয়ে যায়, যেমন ফ্ল্যাগ `setSyncAutomatically()` কর্তৃক সেট হয়।

`SYNC_EXTRAS_EXPEDITED`/ সিঙ্ক এক্সট্রাস এক্সপেডেড

অবিলম্বে শুরু করতে সিঙ্ক কে ফোর্স করে (বল প্রয়োগ)। যদি আপনি এটা সেট করে না থাকেন, সিস্টেম সিঙ্ক রিকোয়েস্ট রান করার জন্য কয়েক সেকেন্ড অপেক্ষা করে, কারন এটা অতি অল্প সময়ে অনেক রিকোয়েস্ট শিডিউল করার মাধ্যমে ব্যাটারি ব্যবহার অপটিমাইজ করার চেষ্টা করে।

নিচের কোড চিত্রটি আপনাকে দেখায় একটি বাটন ক্লিক করার জবাবে কীভাবে `requestSync()` কল করতে হয়:

```
public class MainActivity extends FragmentActivity {  
    ...  
    // Constants  
    // Content provider authority  
    public static final String AUTHORITY =  
        "com.example.android.datasync.provider"  
    // Account type  
    public static final String ACCOUNT_TYPE = "com.example.android.datasync";  
    // Account
```

```

public static final String ACCOUNT = "default_account";
// Instance fields
Account mAccount;
...
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ...
    /*
     * Create the dummy account. The code for CreateSyncAccount
     * is listed in the lesson Creating a Sync Adapter
     */

    mAccount = CreateSyncAccount(this);
    ...
}
/**
 * Respond to a button click by calling requestSync(). This is an
 * asynchronous operation.
 *
 * This method is attached to the refresh button in the layout
 * XML file
 *
 * @param v The View associated with the method call,
 * in this case a Button
 */
public void onRefreshButtonClick(View v) {
    ...
    // Pass the settings flags by inserting them in a bundle
    Bundle settingsBundle = new Bundle();
    settingsBundle.putBoolean(
        ContentResolver.SYNC_EXTRAS_MANUAL, true);
    settingsBundle.putBoolean(
        ContentResolver.SYNC_EXTRAS_EXPEDITED, true);
    /*
     * Request the sync for the default account, authority, and
     * manual sync settings
     */
    ContentResolver.requestSync(mAccount, AUTHORITY, settingsBundle);
}

```

ইউজার ইনফো এবং লোকেশন (অবস্থান) সহকারে অ্যাপ তৈরী করা

(<http://developer.android.com/training/building-userinfo.html>)

এই ক্লাসগুলো আপনাকে শেখাবে কীভাবে আপনার অ্যাপে ইউজার পারসোনালাইজেশন () যুক্ত করতে হয়। এটা করার কিছু উপায়ের মধ্যে রয়েছে ইউজারকে চিহ্নিত করা, তাদের সম্পর্কে প্রাসঙ্গিক তথ্য প্রদান করার মাধ্যমে এবং তাদের পারিপার্শ্বিক অবস্থা সম্পর্কে তথ্য প্রদান করার মাধ্যমে।

১. কনট্যাক্টস ডাটায় প্রবেশকরা

কীভাবে অ্যান্ড্রয়েডের কেন্দ্রীয় এড্রেস বুক, কনট্যাক্টস প্রভাইডার ব্যবহার করা যায়, কনট্যাক্ট প্রদর্শন এবং তাদের বিস্তারিত এবং কনট্যাক্ট তথ্য পরিবর্তন।

1. কনট্যাক্ট লিস্ট পূণরুদ্ধার
2. কনট্যাক্ট এর জন্য বিবরণ পূণরুদ্ধার
3. ইনটেন্ট ব্যবহার করে কনট্যাক্ট পরিবর্তন
4. কুইক কনট্যাক্ট ব্যাজ প্রদর্শন

২. আপনার অ্যাপকে লোকেশন-অ্যাওয়ার করে তৈরী করা

কীভাবে ইউজারের বর্তমান অবস্থান পাওয়ার মাধ্যমে আপনার অ্যাপে অবস্থান-সচেতন বৈশিষ্ট্য(লোকেশন-অ্যাওয়ার ফিচার) যোগ করা যায়।

1. বর্তমান লোকেশন (অবস্থান) পূণরুদ্ধার
2. লোকেশন (অবস্থান) আপডেট গ্রহণ
3. লোকেশন (অবস্থান)
4. এবডভহপবং তৈরী করা এবং মনিটরিং করা
5. ইউজারের বর্তমান একটিভিটি সনাক্ত করা
6. মোক লোকেশন ব্যবহার করে পরীক্ষা

Table of Contents

বাংলায় অ্যান্ড্রয়েড সহায়িকা	
মন্তব্য/ সংশোধন/ অবদান	
অ্যাপলিকেশন তৈরীর কাজ শুরু করুন	
আপনার প্রথম অ্যাপ তৈরী করুন	
অ্যান্ড্রয়েড প্রজেক্ট তৈরী করা	
ইক্সিপস দিয়ে প্রজেক্ট তৈরী করা	
কমান্ড লাইন টুলস দিয়ে প্রজেক্ট তৈরী করুন	
অ্যাপ রান করা	
একটা সত্যিকার ডিভাইসে রান করুন	
ইমুলেটরে রান করুন	
একটি সরল ইউজার ইন্টারফেস তৈরী করা	
লিনিয়ার লে আউট তৈরী করুন	
টেক্সট ফিল্ড এ্যাড (সংযোজন) করুন	
স্ট্রিং রিসোর্স এ্যাড(সংযোজন) করুন	
বাটন এ্যাড (সংযোজন) করুন	
স্ক্রিন প্রস্থ ভরাট করার মতো করে ইনপুট বক্স তৈরী করুন	
অন্য একটিভিটি শুরু করা	
সেভ বাটনকে রেসপন্স করা	
ইনটেন্ট তৈরী করা	
দ্বিতীয় কার্যক্রম(একটিভিটি) শুরু করুন	
দ্বিতীয় একটিভিটি তৈরী করুন	
টাইটেল স্ট্রিং এ্যাড করুন	
এটাকে মেনিফেস্ট এ্যাড করুন	
ইনটেন্ট গ্রহণ করুন	
মেসেজ ডিসপ্লে করুন	
একশন বার এ্যাড (সংযোজন) করা	
এই অধ্যায়ের অনুশীলনী সমূহ	
একশন বার সেট আপ করা	
শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণকে সাপোর্ট করা	
অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণকে সাপোর্ট করে	
একশন বাটন এ্যাড (সংযোজন) করা	
ডগথ এ একশনকে সুনির্দিষ্ট করুন	
একশন বারে একশন এ্যাড করুন	
একশন বাটন রেসপন্স করা	
লো-লেভেল একটিভিটির জন্যে আপ বাটন এ্যাড (সংযোজন) করা	
একশন বারটিকে স্টাইল করা	
অ্যান্ড্রয়েড থীম ব্যবহার করুন	
ব্যাকগ্রাউন্ড কাস্টমাইজ করুন	
শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য	
অ্যান্ড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য	
টেক্সট কালার কাস্টমাইজ	
শুধুমাত্র অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য	

অ্যাড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য
ট্যাব ইন্ডিকেটর কাস্টমাইজ করা

শুধুমাত্র অ্যাড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য
অ্যাড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য

একশন বার ওভারলে করা

ওভারলে মোড সক্রিয় করুন

এ অ্যাড্রয়েড ৩.০ এবং এর চেয়ে উন্নত সংস্করণের জন্য
অ্যাড্রয়েড ২.১ এবং এর চেয়ে উন্নত সংস্করণের জন্য

লেআউট টপ মার্জিন নির্দিষ্ট করুন

বিভিন্ন ডিভাইসকে সাপোর্ট করানো

এই অধ্যায়ের অনুশীলনী সমূহ

ভিন্ন ভিন্ন ভাষাকে সাপোর্ট করা

লোকাল ডিরেক্টরী এবং স্ট্রিং ফাইল তৈরী করা

স্ট্রিং রিসোর্স ব্যবহার করুন

ভিন্ন ভিন্ন স্ক্রিনকে সাপোর্ট করা

ভিন্ন ভিন্ন লেআউট তৈরী করুন

বিভিন্ন ধরনের বিটম্যাপ তৈরী করুন

ভিন্ন প্লাটফর্ম সংস্করণকে সাপোর্ট করা

মিনিমাম (ন্যূনতম) এবং টার্গেট এপিআই লেভেল সুনির্দিষ্ট করা

রানটাইমে সিস্টেম সংস্করণকে চেক করুন

প্লাটফর্ম স্টাইল এবং থিম ব্যবহার

একটিভিটি লাইফসাইকেল ব্যবস্থাপনা

এই অধ্যায়ের অনুশীলনী সমূহ

একটিভিটি শুরু করা

লাইফসাইকেল কলব্যাক সম্পর্কে ধারণা

আপনার অ্যাপের লক্ষ্যের একটিভিটি নির্দিষ্ট করুন

একটি নতুন ইনসটেন্স তৈরী করুন

একটিভিটির বিলোপ

একটি একটিভিটির পজ এবং রিজিউম করা

একটিভিটিতে পজ দেওয়া

আপনার একটিভিটি রিজিউম (পুনরায় শুরু) করা

একটিভিটি স্টপ এবং রিস্টার্ট করা

আপনার একটিভিটি বন্ধ (স্টপ) করুন

আপনার একটিভিটি স্টার্ট/রিস্টার্ট করুন

একটিভিটি পুনরায় তৈরী করা

আপনার একটিভিটি স্টেট (অবস্থান) সেভ করা

আপনার একটিভিটি স্টেটকে (অবস্থান) রিস্টোর করুন

ফ্র্যাগমেন্ট সহ ডাইনামিক ইউজার ইন্টারফেস তৈরী করা

এই অধ্যায়ের অনুশীলনী সমূহ

একটি ফ্রাগমেন্ট তৈরী করা

একটি ফ্রাগমেন্ট ক্লাস তৈরী করুন

XML ব্যবহার করে একটিভিটিতে ফ্র্যাগমেন্ট সংযোজন করুন

একটি নমনীয় (ফ্লেক্সিবল) ইউজার ইন্টারফেস তৈরী করুন

রানটাইমে একটি একটিভিটিতে ফ্রাগমেন্ট সংযোজন (এ্যাড) করুন

একটা ফ্রাগমেন্টের সাথে আরেকটি ফ্রাগমেন্টের প্রতিস্থাপন (রিপ্লেস)

অন্য ফ্রাগমেন্টের সাথে যোগাযোগ
একটি ইন্টারফেস নির্ধারন করা
ইন্টারফেস বাস্তবায়ন করা
ফ্রাগমেন্টে বার্তা (মেসেজ) পৌছে দিন

ডাটা সেভ করা

এই অধ্যায়ের অনুশীলনীসমূহ

কি-ভ্যালু সেট সেভ করা

একটি Shared Preferences এ হ্যান্ডেল লাভ করা
শেয়ারড প্রিফারেন্সে লেখা (রাইট)
শেয়ারড প্রিফারেন্স থেকে পাঠ করা

ফাইল সেভ করা

ইন্টারনাল বা এক্সটার্নাল স্টোরেজ পছন্দ করা
এক্সটার্নাল স্টোরেজের জন্য অনুমতি গ্রহণ
ইন্টারনাল স্টোরেজে ফাইল সেভ করা
এক্সটার্নাল স্টোরেজে ফাইল সেভ করা
ফ্রি স্পেস অনুসন্ধান
ফাইল ডিলিট করা

SQL ডাটাবেজে ডাটা সেভ করা

স্কিমা (Schema) এবং কনট্রাক্ট নির্ধারণ
SQL হেলপার ব্যবহার করে ডাটাবেজ তৈরী
তথ্য একটা ডাটাবেজে রাখুন
ডাটাবেজ থেকে তথ্য পাঠ করা
ডাটাবেজ থেকে তথ্য ডিলিট করা
ডাটাবেজ আপডেট করা

অন্য অ্যাপের সাথে সম্পর্ক স্থাপন

এই অধ্যায়ের অনুশীলনী সমূহ

ইউজারকে অন্য অ্যাপে পাঠানো

একটি ইমপ্লিসিট ইনটেন্ট তৈরী করা
ইনটেন্টটি গ্রহণ করতে একটি অ্যাপ আছে কিনা যাচাই করুন
ইনটেন্ট দিয়ে একটি একটিভিটি শুরু করুন
একটি অ্যাপ চুজার প্রদর্শন করা

একটি একটিভিটি থেকে রেজাল্ট পাওয়া

একটিভিটি শুরু করা
রেজাল্টটি গ্রহণ (রিসিভ) করা

বোনাস: কনট্রাক্ট ডাটা পড়া

অন্য অ্যাপকে আপনার অ্যাপ শুরু করতে দেয়া

ইনটেন্ট ফিল্টার সংযোজন (এ্যাড) করা
আপনার একটিভিটিতে ইনটেন্টটি ব্যবস্থাপনা করা
একটি রেজাল্ট ফেরত নিয়ে আসা

কনটেন্ট শেয়ারিং সহ অ্যাপস তৈরী করুন

সাধারণ ডাটা শেয়ার করা

অনুশীলনীসমূহ

অন্য অ্যাপে সাধারণ ডাটা সেভ করা

টেক্সট কনটেন্ট সেভ করা
বাইনারি কনটেন্ট সেভ করুন

কনটেন্ট এর মাল্টিপল অংশ সেল্ড করুন
অন্য অ্যাপ থেকে সাধারণ ডাটা রিসিভ করা
আপনার মেনিফেস্ট আপডেট করা
ইনকামিং কনটেন্ট ব্যবস্থাপনা করা
একটি সহজ শেয়ার একশন যুক্ত করা
মেনু ডিক্লেয়ারেশন আপডেট করা
শেয়ার ইনটেন্ট সেট করা
ফাইল শেয়ার করা
অনুশীলনীসমূহ
ফাইল শেয়ার করতে অ্যাপ সেট আপ করা
ফাইল প্রভাইডার (প্রদানকারী) নির্দিষ্ট করুন
শেয়ারযোগ্য ডিরেক্টরী নির্দিষ্টকরণ
একটি ফাইল শেয়ার করা
ফাইল রিকোয়েস্ট রিসিভ করা
ফাইল সিলেকশন একটিভিটি তৈরী করা
ফাইল সিলেকশন একটিভিটি কোডে নির্ধারণ করুন
ফাইল সিলেকশনকে রেসপন্স করা
ফাইলের জন্য পারমিশন (অনুমতি) প্রদান
রিকোয়েস্ট করা অ্যাপের সাথে ফাইল শেয়ার
একটি শেয়ার করা ফাইল রিকোয়েস্ট করা
ফাইলের জন্য একটি রিকোয়েস্ট সেল্ড (পাঠানো) করা
রিকোয়েস্ট করা ফাইলে প্রবেশ
ফাইল তথ্য উদ্ধার
একটি ফাইলের MIME টাইপ উদ্ধার
একটি ফাইলের নাম ও সাইজ উদ্ধার
এনএফসি দিয়ে ফাইল শেয়ার করা
অনুশীলনীসমূহ
মেনিফেস্ট ফিচারগুলো (বৈশিষ্ট্যগুলো) ডিক্লেয়ার করা
পারমিশন (অনুমতি) রিকোয়েস্ট (অনুরোধ) করা
NFC বৈশিষ্ট্য গুলো (ফিচারগুলো) নির্দিষ্ট করুন
অ্যাম্বুয়েড বিম ফাইল ট্রান্সফার নির্দিষ্ট করুন
অ্যাম্বুয়েড বিম ফাইল ট্রান্সফার সাপোর্টের জন্য পরীক্ষা
একটি কলব্যাক মেথড তৈরী করুন যা ফাইল সরবরাহ করে
সেল্ড করতে ফাইল নির্দিষ্ট করুন
অন্য ডিভাইস থেকে ফাইল রিসিভ করা
ডাটা প্রদর্শন করতে একটি রিকোয়েস্টের প্রতি রেসপন্স করা
ফাইল পারমিশন রিকোয়েস্ট
কপিকৃত ফাইলের জন্য ডিরেক্টরী পাওয়া
একটি ফাইল ইউআরআই থেকে ডিরেক্টরী লাভ
একটি কনটেন্ট ইউআরআই থেকে ডিরেক্টরী লাভ করা
কনটেন্ট প্রভাইডার নির্ণয় করা
মাল্টিমিডিয়া সহ অ্যাপ তৈরী
অডিও প্লেব্যাক ব্যবস্থাপনা
অনুশীলনীসমূহ
আপনার অ্যাপের ভলিউম এবং প্লেব্যাক নিয়ন্ত্রণ

[কোন আউট স্ট্রিম ব্যবহার করতে হবে তা চিহ্নিত করা](#)
[আপনার অ্যাপের অডিও ভলিউম নিয়ন্ত্রণ করতে হার্ডওয়ার ভলিউম কি ব্যবহার করুন](#)
[আপনার অ্যাপের অডিও প্লেব্যাক নিয়ন্ত্রণ করতে হার্ডওয়ার প্লেব্যাক নিয়ন্ত্রণ কি ব্যবহার করুন](#)

[অডিও ফোকাস ব্যবস্থাপনা](#)

[অডিও ফোকাস রিকোয়েস্ট](#)

[অডিও ফোকাসের ক্ষতি/হারিয়ে ফেলা কে ব্যবস্থাপনা করা](#)

[Duck! / ডাক !](#)

[অডিও আউটপুট হার্ডওয়ার ব্যবস্থা করা](#)

[চেক করুন কোন হার্ডওয়ার ব্যবহৃত হচ্ছে](#)

[অডিও আউটপুট হার্ডওয়ার এর পরিবর্তন সামলানো](#)

[ফটো ক্যাপচার করা](#)

[অনুশীলনীসমূহ](#)

[ছবি নেয়া](#)

[ক্যামেরা পারমিশন রিকোয়েস্ট করা](#)

[ক্যামেরা অ্যাপ দিয়ে ফটো তুলুন](#)

[ফটো ভিউ করা](#)

[ফটো সেভ করা](#)

[ফাইল নেম সেট করুন](#)

[ইনটেন্টের উপর ফাইলনেম যুক্ত করুন](#)

[একটি গ্যালারিতে ফটো সংযোজন করুন](#)

[একটি স্কেলড ইমেজ ডিকোড করা](#)

[ভিডিও রেকর্ড করা](#)

[ক্যামেরা পারমিশন রিকোয়েস্ট](#)

[ক্যামেরা অ্যাপ দিয়ে ভিডিও রেকর্ড করা](#)

[ভিডিও ভিউ](#)

[ক্যামেরা নিয়ন্ত্রণ করা](#)

[ক্যামেরা অবজেক্ট ওপেন করা](#)

[ক্যামেরা প্রিভিউ তৈরী করা](#)

[প্রিভিউ ক্লাস](#)

[প্রিভিউ সেট করুন এবং শুরু করা](#)

[ক্যামেরা সেটিং পরিবর্তন করা](#)

[প্রিভিউ ওরিয়েন্টেশন সেট করুন](#)

[একটি ছবি নেয়া](#)

[রিভিউ পুণরায় শুরু করনি \(রিস্টার্ট\)](#)

[প্রিভিউ বন্ধ করুন এবং ক্যামেরা রিলিজ \(ছেড়ে দেয়া\) করুন](#)

[কনটেন্ট প্রিন্ট করা](#)

[অনুশীলনীসমূহ](#)

[ফটো প্রিন্ট](#)

[ইমেজ প্রিন্ট](#)

[HTML ডকুমেন্ট প্রিন্ট](#)

[HTML ডকুমেন্ট লোড করুন](#)

[প্রিন্ট জব তৈরী করুন](#)

[কাস্টম ডকুমেন্টস প্রিন্ট](#)

[প্রিন্ট ম্যানেজারের সাথে যুক্ত করুন](#)

একটি প্রিন্ট অ্যাডাপটার তৈরী করুন

প্রিন্ট ডকুমেন্ট ইনফো ক্যাপচার করুন

একটি প্রিন্ট ডকুমেন্ট ফাইল লেখা (রাইট)

PDF পেজ কনটেন্ট ড্র করা

গ্রাফিক্স এবং অ্যানিমেশন দিয়ে অ্যাপস তৈরী

দক্ষতার সাথে বিটম্যাপ প্রদর্শন

অনুশীলনীসমূহ

বড় আকারের বিটম্যাপ দক্ষতার সাথে লোড করা

বিটম্যাপের মাত্রা (ডাইমেনশন) এবং টাইপ পড়ন

মেমরীতে একটি স্কেলডাউন-সংস্করণ লোড করুন

বিটম্যাপ প্রক্রিয়া ইউআই থ্রেড বন্ধ করে

Async Task ব্যবহার করুন

কনকারেন্সি চালনা করা

বিটম্যাপ জমিয়ে রাখা (ক্যাশিং)

একটি মেমরী ক্যাশে ব্যবহার

ডিস্ক ক্যাশে ব্যবহার

কনফিগারেশন পরিবর্তন ব্যবস্থাপনা

বিটম্যাপ মেমরী ব্যবস্থাপনা

অ্যান্ড্রয়েড ২.২.৩ এবং এর চেয়ে নীচের সংস্করণে মেমরী ব্যবস্থাপনা

অ্যান্ড্রয়েড ৩.০ এবং এর চেয়ে উপরের সংস্করণে মেমরী ব্যবস্থাপনা

পরবর্তী ব্যবহারের জন্য একটি বিটম্যাপ সেভ করা

একটি বিদ্যমান বিটম্যাপ ব্যবহার

আপনার ইউআই একটি বিটম্যাপ প্রদর্শন

ভিউপেজ বাস্তবায়নে বিটম্যাপ লোড করা

একটি গ্রিডভিউ বাস্তবায়নে বিটম্যাপ লোড করা

OpenGL ES দিয়ে গ্রাফিক্স প্রদর্শন

অনুশীলনীসমূহ

একটি OpenGL ES পরিবেশ তৈরী করা

মেনিফেস্টে OpenGL ES ব্যবহার ডিক্লেয়ার করুন

OpenGL ES গ্রাফিক্স এর জন্য একটি একটিভিটি তৈরী করুন

একটি GLSurfaceView অবজেক্ট তৈরী করুন

রেন্ডার ক্লাস তৈরী করা

গঠন/আকৃতি নির্ধারণ করা

একটি ত্রিভুজ (ট্রায়াঙ্গেল) নির্ধারণ

একটি বর্গক্ষেত্র (ক্বয়ার) নির্ধারণ

শেপ/আকৃতি অংকন করা

শেপ/আকৃতি আরম্ভ করা

শেপ অংকন/ ড্র করা

প্রজেকশন এবং ক্যামেরা ভিউ প্রয়োগ

একটি প্রজেকশন নির্ধারণ

একটি ক্যামেরা ভিউ নির্ধারণ

প্রজেকশন এবং ক্যামেরা ট্রান্সফর্মেশন প্রয়োগ

গতিশীলতা (মোশন) যুক্ত করা

শেপ রোটেশন করা

অবিরাম রেন্ডারিং সক্রিয় করা

[ট্যাচ ইভেন্টে রেসপন্স করা](#)

[ট্যাচ লিসেনার সেটআপ](#)

[রোটেশন অ্যাংগেল উন্মোচন করুন](#)

[রোটেশন প্রয়োগ](#)

[অ্যানিমেশন সংযোজন করা](#)

[অনুশীলনীসমূহ](#)

[দুইটা ভিউকে ক্রসফেড করা](#)

[ভিউগুলো তৈরী করুন](#)

[অ্যানিমেশন সেটআপ করা](#)

[ভিউ ক্রসফেড](#)

[স্ক্রিন স্লাইড এর জন্য ভিউ পেজ ব্যবহার](#)

[ভিউ তৈরী করুন](#)

[ফ্রাগমেন্ট তৈরী করুন](#)

[ভিউ পেজ যুক্ত করা](#)

[পেজ ট্রান্সফরমার দিয়ে অ্যানিমেশন কাস্টমাইজ করা](#)

[পেজ ট্রান্সফরমার জুম আউট করা](#)

[ডেপথ পেজ ট্রান্সফরমার](#)

[কার্ড ফ্লিপ অ্যানিমেশন প্রদর্শন করা](#)

[অ্যানিমেটর তৈরী করুন](#)

[card_flip_left_in.xml](#)

[card_flip_left_out.xml](#)

[card_flip_right_in.xml](#)

[card_flip_right_out.xml](#)

[ভিউ তৈরী করুন](#)

[ফ্রাগমেন্ট তৈরী করা](#)

[কার্ড ফ্লিপ অ্যানিমেট করা](#)

[ভিউ জুম করা](#)

[ভিউ তৈরী করা](#)

[জুম অ্যানিমেশন সেটআপ করা](#)

[ভিউ জুম করুন](#)

[লেআউট পরিবর্তন অ্যানিমেট করা](#)

[একটি লেআউট তৈরী করা](#)

[লেআউট থেকে আইটেম গ্র্যাড \(যুক্ত\), আপডেট বা রিমুভ \(বিয়োজন\) করা](#)

[কানেকটিভিটি এবং ক্লাউড দিয়ে অ্যাপস তৈরী](#)

[ডিভাইসে ওয়ারলেস সংযোগ](#)

[অনুশীলনীসমূহ](#)

[নেটওয়ার্ক সার্ভিস ডেলিভারি ব্যবহার](#)

[নেটওয়ার্কে আপনার সার্ভিস রেজিট্রেশন করা](#)

[নেটওয়ার্কে সার্ভিস উন্মোচন করা](#)

[নেটওয়ার্কে সার্ভিসে যুক্ত করুন](#)

[অ্যাপলিকেশন বন্ধ করার সময় আপনার সার্ভিস আনরেজিস্টার করুন](#)

[ওয়াই-ফাই সহকারে P2P কানেকশন তৈরী করা](#)

[অ্যাপলিকেশন পারমিশন সেটআপ করা](#)

[একটি ব্রডকাস্ট রিসিভার এবং ওয়াই-ফাই পিয়ার টু পিয়ার \(P2P\) ম্যানেজার সেটআপ](#)

[করুন](#)

পিয়ার ডিসকভারি শুরু করুন

পিয়ার লিস্ট নিয়ে আসা

একটি পিয়ারের সাথে সংযোগ

সার্ভিস ডিসকভারির জন্য ওয়াই-ফাই P2P ব্যবহার করা

মেনিফেস্ট সেটআপ করুন

লোকাল সার্ভিস যুক্ত করুন

নিকটস্থ সার্ভিস উদঘাটন (ডিসকভারি)

নেটওয়ার্ক অপারেশন সম্পাদন করা

অনুশীলনী সমূহ

নেটওয়ার্কে যুক্ত হওয়া

একটি HTTP ক্লায়েন্ট নির্বাচন

নেটওয়ার্ক কানেকশন চেক করুন

একটি পৃথক থ্রেডে নেটওয়ার্ক অপারেশন সম্পাদন করুন

ডাটা সংযোগ এবং ডাউনলোড

ইনপুটস্ট্রিম কে একটি স্ট্রিং এ পরিবর্তন (কনভার্ট)

নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা

একটি ডিভাইসের নেটওয়ার্ক কানেকশন চেক করা

নেটওয়ার্ক ব্যবহার ব্যবস্থাপনা

একটি প্রিফারেন্স একটিভিটি বাস্তবায়ন

প্রিফারেন্স পরিবর্তনে রেসপন্স করা

কানেকশন পরিবর্তন চিহ্নিত করা

XML ডাটা পারসিং করা

একটি পার্সার নির্বাচন করুন

ফিড বিশ্লেষণ

পার্সার ইনসটেনসিয়েট করা

ফিড পাঠ করা

XML পার্স করা

আপনার কাছে গুরুত্বপূর্ণ নয় এমন ট্যাগগুলো এড়িয়ে (স্কিপ) যান

XML ডাটা ব্যবহার

ব্যাটারি শেষ না করেই ডাটা ট্রান্সফার

অনুশীলনীসমূহ

কার্যকরী নেটওয়ার্কে প্রবেশ করার জন্য ডাউনলোড অপটিমাইজ করা

রেডিও স্টেট মেশিন

অ্যাপ কীভাবে রেডিও স্টেট মেশিন প্রভাবিত করে

ডাটা প্রিফেচ (প্রধান মেমরী থেকে অস্থায়ী স্টোরেজে পরবর্তী ব্যবহারের জন্য) করা

ব্যাচ ট্রান্সফার এবং সংযোগ

কানেকশন (সংযোগ) কমিয়ে আনা

চিন্তার জায়গা চিহ্নিত করতে DDMS নেটওয়ার্ক ট্রাফিক টুল ব্যবহার করে

নিয়মিত আপডেটের প্রভাব কমানো

পোলিং এ বিকল্প হিসাবে গুগল ক্লাউড মেসেজিং ব্যবহার

ইনএক্স্যাক্ট রিপটিং এ্যালার্ম এবং এক্সপোনেন্ট ব্যাকঅফ দিয়ে পোলিং অপটিমাইজ করা

রিডানডেন্ট ডাউনলোড অনাবশ্যক

স্থানীয়ভাবে ফাইল ক্যাশে (Cache) করা

HttpURL কানেকশন রেসপন্স ক্যাশে ব্যবহার করুন

কানেকটিভিটি ধরনের উপর ভিত্তি করে ডাউনলোডের ধরন পরিবর্তন করুন

ওয়াই-ফাই ব্যবহার

আরও বেশী বেশী ডাউনলোড করতে ব্যান্ডউইথ ব্যবহার করুন

ক্লাউডে সিঙ্ক (Syncing) করা

অনুশীলনী

ব্যাকআপ এপিআই ব্যবহার

অ্যান্ড্রয়েড ব্যাকআপ সার্ভিসের জন্য রেজিস্টার

আপনার মেনিফেস্ট কনফিগার করুন

আপনার ব্যাকআপ এজেন্ট লিখুন (রাইট)

ব্যাকআপ রিকোয়েস্ট

একটি ব্যাকআপ থেকে রিস্টোর করুন

অধিকাংশ গুগল ক্লাউড মেসেজিং তৈরী করা

কার্যকরীভাবে মাল্টিকাস্ট মেসেজ সেভ (পাঠানো) করুন

যে মেসেজটি প্রতিস্থাপিত হতে পারে সেটা বন্ধ করে দিন

সরাসরি GCM মেসেজের মধ্যে ডাটা বসানো

GCM মেসেজের প্রতি Intelligently রিয়েক্ট করা

বিরক্তিকর করবেন না

স্মার্টভাবে সিঙ্ক করুন কঠিন ভাবে নয়

ক্লাউডে সেভ করা সম্পর্কিত সাংঘর্ষিক বিষয়ের সমাধান করা

কনফ্লিক্ট সম্পর্কে অবগত হতে

সাধারণ কেসগুলো পরিচালিত করা

আরও জটিল কেসের জন্য একটি কৌশল ডিজাইন করা

প্রথম প্রচেষ্টা: শুধু সর্বমোটটি (টোটাল) স্টোর করুন

দ্বিতীয় প্রচেষ্টা: সর্বমোট (টোটাল) এবং ডেল্টা স্টোর করুন

সমাধান: ডিভাইসে প্রতি সাব-টোটাল স্টোর করুন

আপনার ডাটা পরিষ্কার করুন

সিঙ্ক অ্যাডাপটর ব্যবহার করে ডাটা ট্রান্সফার

অনুশীলনীসমূহ

একটি স্টাব-অথেনটিকেটর (প্রমাণকারী) তৈরী করা

একটি স্টাব অথেনটিকেটর উপাদান যুক্ত করুন

ফ্রেমওয়ার্কে অথেনটিকেটর সংযুক্ত করা

অথেনটিকেটর মেটাডেটা ফাইল যুক্ত করা

মেনিফেস্টের মধ্যে অথেনটিকেটর ডিক্লেয়ার করা

একটি স্টাব কনটেন্ট প্রভাইডার তৈরী করা

একটি স্টাব কনটেন্ট প্রভাইডার যুক্ত করা

মেনিফেস্টের মধ্যে প্রভাইডার ডিক্লেয়ার করা

একটি সিঙ্ক অ্যাডাপটর তৈরী করা

বেজ সিঙ্ক অ্যাডাপটর ক্লাস AbstractThreadedSyncAdapter প্রসারিত করুন

onPerformSync() এ ডাটা ট্রান্সফার কোড যুক্ত করুন

ফ্রেমওয়ার্কে সিঙ্ক অ্যাডাপটর একত্রিত করা

ফ্রেমওয়ার্ক কর্তৃক চাওয়া একাউন্ট যুক্ত করুন

সিঙ্ক অ্যাডাপটর মেটাডেটা ফাইল যুক্ত করুন

মেনিফেস্টের মধ্যে সিঙ্ক অ্যাডাপটর ডিক্লেয়ার করা

একটি সিঙ্ক অ্যাডাপটর রান করা

একটি সিঙ্ক অ্যাডাপটর রান করা যখন সার্ভার ডাটা পরিবর্তন করে

সিঙ্ক অ্যাডাপটর রান করা যখন কনটেন্ট প্রভাইডার ডাটা পরিবর্তন করে

একটি নেটওয়ার্ক মেসেজের পর সিঙ্ক অ্যাডাপটর রান করা
পর্যায়ক্রমে সিঙ্ক অ্যাডাপটর রান করা যখন
চাহিদা অনুযায়ী সিঙ্ক অ্যাডাপটর রান করা

ইউজার ইনফো এবং লোকেশন (অবস্থান) সহকারে অ্যাপ তৈরী করা

