Fundamentos de Aplicações Android

11. novembro 2011 Android, Mobile

Continuando com meus posts acerca da arquitetura da plataforma Android, hoje escrevo sobre o básico das aplicações Android, seu funcionamento, componentes, recursos, etc, para que você entenda um pouco mai antes de sair codificando para sua plataforma mobile favorita. Este material é fortemente baseado na documentação oficial do Android, disponível em http://developer.android.com (em inglês).

Fundamentos Iniciais

Aplicações Android são escritas na linguagem de programação Java. As ferramentas do Android SDK compilam o código - juntamente com quaisquer outros dados e arquivos de recurso - em um pacote Androic um arquivo com a extensão .apk. Todo o código em um único arquivo .apk é considerado uma aplicação e é o arquivo que dispositivos usando Android usam para instalar a aplicação.

Uma vez instalado no dispositivo, cada aplicação Android reside em sua própria sandbox segura:

- O sistema operacional Android é um sistema Linux multi-usuário onde cada aplicação roda com um usuário diferente.
- Por padrão, o sistema ssocia cada aplicação a um ID de usuário Linux único (o ID usado somente pe sistema e que é desconhecido pela aplicação). O sistema define permissões por todos os arquivos d uma aplicação para somente aquele ID de usuário poder acesá-los.
- Cada processo tem sua própria maquina virtual (VM), portanto o código de uma aplicação é executac isolado de outras aplicações.
- Por padrão, cada aplicação executa seu próprio processo Linux. Android inicia o processo quando qualquer dos componentes de uma aplicação precisa ser executado, desligando o processo quando mesmo não é mais necessário ou quando o sistema deve recuperar memória para outras aplicações

Desta forma, o sistema Android implementa o princípio do menos privilegiado. Isto é, cada aplicação, por padrão, tem acesso somente aos componentes que ele necessita para funcionar e nada mais. Isto cria um ambiente muito seguro onde cada aplicação não pode acessra partes do sistema onde não possui permissã

Entretanto, há algumas maneiras para uma aplicação compartilhar dados com outras aplicações e para um aplicação acessar serviços do sistema:

- É possível organizar duas aplicações para compartilhar o mesmo ID de usuário Linux, neste caso ela estarão aptas a acessar arquivos uma da outra. Para conservar recursos do sistema, aplicações com mesmo ID de usuário podem também organizar-se para executar no mesmo processo Linux e compartilhar a mesma VM (as aplicações devem ser assinadas com o mesmo certificado).

- Uma aplicação pode requisitar permissão para acessar dados do dispositivo como os contatos do usuário, mensagens SMS, o armazenamento removível (cartão SD), câmera, Bluetooth e muito mais Todas permissões de aplicação devem ser concedidas pelo usuário durante a instalação.

Isto cobre o básico a respeito de como uma aplicação Android existe dentro do sistema. O resto deste post dá uma introdução sobre:

- Os componentes centrais do framework que definem sua aplicação.
- O arquivo de manifesto onde você declara os componentes e as caracteristicas do dispositivo que sã obrigatórias para sua aplicação.
- Recursos que são separados do código da aplicação para otimizar enormemente seu comportament para uma variedade de configurações de dispositivo.

Componentes de Aplicação

Componentes de aplicação são os blocos de construção essenciais de uma aplicação Android. Cada componente é um ponto diferente através do qual o sistema pode entrar na sua aplicação. Nem todos componentes são na verdade pontos de entrada para o usuário e alguns dependem de outros, mas cada u existe como sua própria entidade e executa um papel específico - cada um é um bloco de construção específico que ajuda a definir o comportamento geral de sua aplicação.

Existem quatro tipos diferentes de componentes de aplicação. Cada tipo serve a um propósito distinto e ten ciclo de vida diferente que define como o componente é criado e destruído.

Aqui estão os quatro tipos de componentes de aplicação:

Activities

Uma activity (atividade) representa uma tela comum com uma interface de usuário. Por exemplo, uma aplicação de e-mail pode ter uma activity que mostra a lista de novos e-mails, outra activity para compór e-mails, e outra activity para ler e-mails. Emboras as activities trabalhem juntas para formar uma experiência usuário coesa na aplicação de e-mail, cada uma é independente das outras. Desta forma, uma aplicação diferente pode iniciar qualquer uma destas activities (se a aplicação de e-mail permitir). Por exemplo, uma aplicação de câmera pode iniciar a activity na aplicação de e-mail para compor um novo e-mail, visando ao usuário compartilhar a foto.

Uma activity é implementada como uma subclasse de Activity e falareos mais delas nos posts futuros.

Services

Um service (serviço) é um componente que roda em background para executar operações demoradas ou p executar processos remotos. Um service não possui interface do usuário. Por exemplo, um service pode to música no background, enquanto o usuário está em uma aplicação diferente, ou ele pode carregar dados

através da rede sem bloquear a interação do usuário com uma activity. Outro componente, como uma activ pode iniciar o service e deixá-lo rodando ou conectar-se nele para interagir com o mesmo.

Um service é implementado como uma subclasse de Service e falaremos mais deles em posts futuros.

Content Providers

Um content provider (provedor de conteúdo) gerencia um grupo de dados de aplicação compartilhados. Voc pode armazenar dados no sistema de arquivos, uma base SQLite, na web, ou em qualquer outro local de armazenamento persistente que sua aplicação possa acessar. Através do content provider, outras aplicaçõ podem consultar ou mesmo modificar os dados (se o content provider permiti-lo). Por exemplo, o sistema Android provê um content provider que gerencia as informações dos contatos do usuário. Desta forma, qualquer aplicação com as permissões apropriadas pode consultar parte do content provider (como um ContactsContract.Data) para ler e escrever informações sobre uma pessoa em particular.

Content providers também são úteis para ler e escrever dados quee stão privados na sua aplicação e não s compartilhados. Por exemplo, salvar notas pessoais.

Um content provider é implementado como uma subclasse de ContentProvider e deve implementar um conjunto de APIs que permite a outras aplicações executarem transações. Mais informações em posts futur

Broadcast Receivers

Um broadcast receiver (receptor de notificações) é um componente que responde a notificações a nível de sistema. Muitas notificações originam-se do sistema - por exemplo, uma notificação anunciando que a tela f desligada, a bateria está fraca, ou uma foto foi tirada. Aplicações também podem iniciar broadcasts—por exemplo, para deixar outras aplicações sabendo que alguns dados foram baixados para o dispositivo e está disponíveis para uso. Embora broadcast receivers não exibam interface ao usuário, eles podem criar notificações na barra de status para alertar o usuário quando um evento broadcast ocorre. Mais comumente embora, um broadcast receiver é somente um "gateway" para outros componentes e é intencionalmente cri para fazer a menor quantidade de trabalho. Por exemplo, ele pode iniciar um service para executar alguma tarefa baseada no evento.

Um broadcast receiver é implementado como uma subclasse de BroadcastReceiver e cada broadcast é entregue como um objeto Intent. Mais informações futuramente, neste mesmo blog.

Um aspecto único do design do sistema Android é que qualquer aplicação pode iniciar um componente de outra aplicação. Por exemplo, se você quer que o usuário tire uma foto com a câmera do dispositivo, provavelmente existe uma outra aplicação que faz isso e que sua aplicação pode usar, ao invés de desenvo uma activity sua para tirar a foto. Você não precisa incorporar ou mesmo linkar código da aplicação da câm Ao invés disso, você simplesmente inicia a activity na aplicação da câmera que tira a foto. Quando complet

a foto é retornado para sua aplicação e então você pode usá-la. Para o usuário, parece que a câmera é na verdade parte de sua aplicação.

Quando o sistema inicia um componente, ele inicia o processo para aquela aplicação (se ele ainda não est rodando) e instancia as classes necessárias para o componente. Por exemplo, se sua aplicação inicia a aci na aplicação da câmera que captura a foto, a activity roda no processo que pertence à aplicação da câmera não no processo de sua aplicação. Entretanto, diferente da maioria dos outros sistemas, aplicações Androic não possuem um único ponto de entrada (não existe uma função main(), por exemplo).

Devido ao sistema rodar cada aplicação em um processo separado com permissões de arquivo que restrige acesso de outras aplicaç~ies, sua aplicação não pode ativar diretamente um componente de outra aplicaçã sistema Android, entretanto, pode. Então, para ativar um componente de outra aplicação, você deve entreg uma mensagem para o sistema que especifica sua intenção (intent) de iniciar um componente em particula sistema então ativa o componente para você.

Ativando Componentes

Três dos quatro tipos de componentes - activities, services, e broadcast receivers - são ativados por mensagens assíncronas chamadas de intent (intenção). Intents conectam componentes individuais com ou componentes em tempo de execução (você pode pensar neles como mensageiros que requisitam uma açã outros componentes, quer o componente pertença à sua aplicação ou outra.

Um intent é criado com um objeto Intent, que define uma mensagem para ativar um componente específico tipo específico de componente - um intent pode ser explícito ou implícito, respectivamente.

Para activities e servicesm um intent define a ação a ser executada (por exemplo, para ver ou enviar algo) e pode especificar a URI do dado para agir sobre (ou outras coisas que o componente a ser iniciado precisa saber). Por exemplo, um intent pode ser uma requisição para uma activity mostrar uma imagem a ser abert em uma página web. Em alguns casos, você pode iniciar uma activity para receber um resultado, neste cas activity também retorna o resultado como um Intent (por exemplo, você pode criar um intent para pegar contatos pessoais e ter eles retornados para você - o intent de retorno inclui a URI apontando para o contat escolhido).

Para broadcast receivers, o intent simplesmente define a notificação sendo divulgada (por exemplo, uma notificação que indica que a bateria do dispositivo está baixa inclui somente uma string que indica "bateria baixa").

O outro tipo de componente, content provider, não é ativado por intents. Ao invés disso, ele é ativado quanc alvejado por uma requisição de um ContentResolver. O content resolver manipula todas transações diretas com o content provider para que o componente que está executando as transações com o provider não pre fazê-lo e ao invés disso chame métodos no objeto ContentResolver. Isto deixa uma camada de abstração e

o content provider e o componente requisitando a informação (por segurança).

Existem métodos separados para ativar cada tipo de componente:

- Você pode iniciar uma activity (ou dar algo novo pra ela fazer) passando um Intent para startActivity() startActivityForResult() (quando você quer que a activity retorne um resultado).
- Você pode iniciar um service (ou dar novas instruções a um service em execução) passando um Inte para startService(). Ou você pode ligar ao service passando um Intent para bindService().
- Você pode iniciar um broadcast passando um Intent para métodos como sendBroadcast(), sendOrderedBroadcast(), ou sendStickyBroadcast().
- Você pode executar uma consulta a um content provider chamando query() em um ContentResolver.

Mais informações sobre o uso de intents serão vistos em posts futuros, bem como a ativação de componen específicos.

O Arquivo Manifest

Antes do sistema Android poder iniciar um componente de aplicação, o sistema deve saber que o compone existe lendo o arquivo AndroidManifest.xml da aplicação (o arquivo "manifest"). Sua aplicação deve declara todos seus componentes neste arquivo, que deve estar na raix do diretório do projeto da aplicação.

O manifest faz um número de coisas em adição à declaração de componentes da aplicação, como:

- Identificar quaisquer permissões de usuário necessárias à aplicação, como acesso à Internet ou aces de leitura nos contatos do usuário.
- Declarar o API Level mínimo necessário pela aplicação, baseado nas APIs que a aplicação usa.
- Declarar as características de hardware e software usadas ou requeridas pela aplicação, como câme serviço de bluetooth, ou tela multitouch.
- Bibliotecas de API necessárias para a execução (além das nativas das APIs do framework Android), como a biblioteca do Google Maps.
- E muito mais.

Declarando Componentes

A tarefa primária do manifesto é informar ao sistema sobre os componentes da aplicação. Por exemplo, um arquivo de manifesto pode declarar uma activity como segue:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
  <application android:icon="@drawable/app_icon.png" ... >
    <activity android:name="com.example.project.ExampleActivity"
              android:label="@string/example_label" ... >
```

```
        </activity>
        ...
    </application>
</manifest>
```

No elemento <application>, o atributo android:icon aponta para o recurso de um único que identifica a aplicação.

No elemento <activity>, o atributo android:name especifica o nome completo da subclasse de Activity e o atributo android:label especifica a string a ser usada no rótulo visível ao usuário para a activity.

Você deve declarar todos componentes da aplicação desta maneira:

- elementos <activity> para activities
- elementos <service> para services
- elementos <receiver> para broadcast receivers
- elementos <provider> para content providers

Activities, services, e content providersque você incluir no seu fonte mas não declarar no manifesto não est visíveis para o sistema e, consequentemente, nunca podem rodar. Entretanto, broadcast receivers podem s declarados no manifesto ou criados dinamicamente no código (como objetos BroadcastReceiver) e registra no sistema chamando registerReceiver().

Veremos mais detalhes do arquivo de manifesto em posts futuros.

Declarando Capacidades dos Componentes

Como discutido acima, você pode usar um Intent para iniciar activities, services, e broadcast receivers. Voc pode fazê-lo explicitamente nomeando o componente alvo (usando o nome de classe do componente) no intent. Entretanto, o verdadeiro poder dos intents reside no conceito de ações intencionais (intent actions). C intent actions, você simplesmente descreve o tipo da ação que você quer executar (e opcionalmente, os da que você deseja que sejam utilizados) e permite ao sistema encontrar um componente no dispositivo que possa executar a ação e iniciá-la. Se existem múltiplos componentes que possam executar a ação descrita intent, então o usuário seleciona qual quer usar.

A maneira do sistema identificar os componentes que podem responder a um intent é comparando o intent recebido com os filtros de intenção (intent filters) providos no XML de manifesto de outras aplicações no dispositivo.

Quando você declara um componente no manifetso de sua aplicação, você pode opcionalmente incluir inte filters que declaram as capacidade do componente para que ele possa responder a intents de outras

aplicações. Você pode declarar um intent filter para seu componente adicionando um elemento <intent-filter como nó filho do elemento de declaração de um componente.

Por exemplo, uma aplicação de e-mail com um activity de compor um novo e-mail pode declarar um intent f em seu manifesto para responder a intents "send" (para enviar e-mails). uma activity na sua aplicação pode então criar um intent com a ação “send” (ACTION_SEND), que o sistema irá combinar com a activity "send" aplicação de e-mail e lançá-la quando você invocar o intent com startActivity().

Mais informações sobre criar intent filters serão discutidas futuramente.

Declarando Requisitos da Aplicação

Existem uma infinidade de dispositivos rodando Android e nem todos provêem as mesmas características e funcionalidades. Para prevenir que sua aplicação seja instalada em dispositivos que não possuem as características necessárias, é importante definir claramente um perfil de tipos de dispositivos que suportam aplicação declarando os requisitos de hardware e software no seu arquivo de manifesto. A maioria destas declarações são informacionais somenyte e o sistema não lê elas, mas serviços externos como o Android Market, visando fornecer serviço de filtragem aos usuários utilizam.

Por exemplo, se sua aplicação requer uma câmera e usa as APIs introduzidas no Android 2.1 (API Level 7) você deve declarar estes requisitos no seu manifesto. Desta forma, dispositivos que não possuem uma câm e tem uma versão de Android inferior à 2.1 não podem instalar sua aplicação pelo Android Market.

Entretanto, você também pode declarar que sua aplicação usa câmera, mas que ela não é obrigatória. Nes caso, sua aplicação deve executar uma checagem em tempo de execução para determinar se o dispositivo possui uma câmera e desabilitar quaisquer características que usam a câmera se ela não estiver disponíve

Aqui estão algumas características importantes dos dispositivos que você deve considerar quando projeta u aplicação:

Tamanho e Densidade de Tela

Para categorizar dispositivos pelo seu tipo d etela, Android define duas características para cada dispositivo tamanho de tela (as dimensões físicas da tela) e densidade de tela (a densidade física dos pixels na tela, o - pontos por polegada). Para simplificar todos os tipos diferentes de configurações de tela, o sistema Andro generaliza em grupos que tornam mais simples a declaração.

Os tamanhos de tela são: small, normal, large, e extra large.
As densidades de tela são: low density, medium density, high density, e extra high density.

Por padrão, sua aplicação é compatível com todos os tamanhos e densidades de telas, porque o sistema Android faz os ajustes apropriados para seu layout de UI e recursos de imagem. Entretanto, você deve cria layouts especializados para certos tamanhos de telas e prover imagens específicas para certas densidades

usando recursos alternativos de layout, e declarando no seu manifesto exatamente quais tamanhos de tela aplicação suporta com o elemento <supports-screens>.

Mais informações em breve.

Configurações de Entrada

Muitos dispositivos provêem um tipo diferente de mecanismo de entrada, como teclado, trackball, ou botões navegação. Se sua aplicação requer um hardware de entrada específico, entãao você deve declarar no seu manifesto com o elemento <uses-configuration>. Entretanto, é raro uma aplicação que requer certo tipo de configuração de entrada.

Características do Dispositivo

Existem muitas características de hardware e software que podem ou não existir em um dado dispositivo Android, como uma câmera, sensor de luz, bluetooth, versão de OpenGL específica ou fidelidade do touchscreen. Você nunca deve assumir que uma certa característica está disponível em todos dispositivos Android (exceto as disponíveis na biblioteca padrão do Android), então você deve declarar quaisquer características usadas pela sua aplicação com o elemento <uses-feature>.

Versão da Plataforma

Dispositivos Android diferentes muitas vezes rodam versões diferentes da plataforma Android, como Androi Android 1.6 ou Android 2.3. Cada versão sucessiva inclui APIs adicionais não disponíveis em versões anteriores. Para indicar qual grupo de APIs está disponível, cada versão especifica um API Level (por exem Android 1.0 é API Level 1 e Android 2.3 é API Level 9). Se você usa quaisquer APIs que foram adicionadas versões superiores à 1.0, você deve declarar um API Level mínimo no qualestas APIs foram introduzidas usando o elemento <uses-sdk>.

É importante que você declare todos os requisitos para sua aplicação, porque, quando você distribui sua aplicação no Android market, ele usa estas declarações para filtrar quais aplicações estão disponíveis para cada dispositivo. Desta forma, sua aplicação deve estar disponível somente para os dispositivos que estão acordo com os requisitos da mesma.

Mais informações sobre os filtros de aplicação do Android Market serão dados futuramente.

Recursos da Aplicação

Uma aplicação Android é composta por mais do que somente código - ela requer recursos que estão separados de código fonte, como imagens, arquivos de áudio, e qualquer coisa relacionada à apresentaçã visual da aplicação. Por exemplo, você deve definir animações, menus, estilos, cores, e o layout das interfa de usuário da activity com arquivos XML. Usar recursos de aplicação torna mais fácil atualizar várias características de sua aplicação sem ter de modificar o código além de prover grupos de recursos alternativ

que servem para otimizar sua aplicação para uma variedade de configurações de dispositivos (como linguagens diferentes e tamanhos de tela).

Para cada recurso que você inclui no seu projeto Android, as ferramentas de compilação do SDK definem u ID inteiro único, que você pode usar para referenciar o recurso no código da sua aplicação ou de outro recu definido no XML. por exemplo, se sua aplicação contém um arquivo de imagem chamado logo.png (salvo n diretório res/drawable/), as ferramentas do SDK geram um ID de recurso chamado R.drawable.logo, que vo pode usar para referencuar a imagem e inseri-la na interface do usuário.

Um dos aspectos mais importantes de prover recursos separados de seu código fonte é a habilidade de voc fornecer recursos alternativos para diferentes configurações de dispositivos. Por exemplo, definindo strings UI em XML, você pode traduzir as strings para outro idioma e salvar estas strings em arquivos separados. Então, baseado no qualificados de idioma que você anexou ao nome do diretório do recurso (como um res/values-fr/ para strings em francês) e as configurações de idioma do usuário, o sistema Android aplica o idioma apropriado na sua interface.

Android suporta muitos qualificadores diferentes para seus recursos alternativos. O qualificador é uma strin curta que você inclui no nome dos seus diretórios de recursos visando definir as configurações do dispositiv para as quais estes recursos devem ser usados. Como outro exemplo, você deve muitas vezes criar diferer layouts para suas activities, dependendo da orientação e tamanho d atela do dispositivo. Por exemplo, quar a tela do dispositivo está em modo vertical (retrato), você pode querer um layout com botões verticais, mas quando a tela está na horizontal (paisagem), os botões devem ser alinhados horizontalmente. Para mudar o layout dependendo da orientação, você pode definir dois diferentes layouts e aplicar o qualificador apropria em cada nome de diretório de layout. Então, o sistema automaticamente aplica o layout apropriado dependendo da orientação atual do dispositivo.

Eu sei que tudo isso é muito superficial, mas espero que tenha gostado, e aos poucos vamos explorando es nova plataforma que cresce cada vez mais!