Ares e webOS

Desenvolvimento para webOS

A aquisição da Palm pela HP e o anúncio do webOS 2.0 trouxeram interesse renovado pelo desenvolvimento nesse sistema. Felizmente, é muito fácil começar a criar aplicativos para webOS com o ambiente de desenvolvimento Ares, da Palm.

por Roy Sutton

O Ares, da Palm, é um ambiente de desenvolvimento integrado (IDE, na sigla em inglês) que é executado diretamente dentro do navegador de Internet para a criação de aplicativos para webOS. O Ares, que suporta a criação de interfaces com o simples arrastar e soltar do mouse, possui depuração integrada e até controle de versões de código, e é baseado no Bespin, um editor de código aberto projetado para a web.

Dar os primeiros passos no Ares é muito fácil; requer apenas uma conta no *Palm Developer Center* [1] e um navegador web (que podem ser o Firefox 3.5 ou mais recente, o Safari 4.0 ou mais recente, ou até mesmo o Google Chrome). Entretanto, para testar seus aplicativos ou tirar proveito do depurador integrado, é preciso baixar e instalar o SDK do Palm webOS, disponível para Windows, Mac OS X e Linux, no site do Ares [2].

O Ares fornece o esqueleto pronto do aplicativo a ser criado e permite acessar componentes e recursos da plataforma Palm Mojo. O Mojo contém serviços e componentes pré-fabricados para uso com o webOS. O Mojo e o Ares facilitam a criação de aplicativos sofisticados com o visual e comportamento padrões do webOS (**quadro 1**).

Este artigo apresentará os principais passos para criar um aplicativo de exemplo com o Ares. Neste caso, o aplicativo usa a API de busca local do Yahoo! [3] para buscar cinemas próximos a um local específico para você usar na próxima vez que for à Disney – já que por enquanto só funciona nos EUA. O usuário pode digitar tanto um CEP quanto utilizar um dispositivo GPS para descobrir sua localização atual. O aplicativo em seguida exibe os resultados em um componente instalável do Google Maps através de sua API [4]. Também veremos vários conceitos importantes do webOS, tais como eventos, serviços, Ajax e CSS.

Primeiros passos

O IDE Ares possui três componentes básicos (**figura 1**): a paleta de combinações e o painel com o navegador do projeto ficam do lado esquerdo da tela. O painel de configurações ocupa o lado direito da tela. No meio da tela fica o painel de design e código do projeto. A maior parte do projeto da interface é feito arrastando componentes da paleta e soltando-os na tela de projeto.

Para começar a criar o projeto, clique em *New* no canto superior esquerdo da tela e selecione a opção *Application*. Dê ao aplicativo o nome *Movie*. Isto criará a estrutura do novo projeto e apresentará ao desenvolvedor uma tela em branco. O framework Mojo usa um padrão MVC (*Model*, *View*, *Controller* – Modelo, Visualização e Controle) para desenvolvimento de aplicativos. Cada interface do sistema terá um controlador (chamado de *assistente*) e uma visualização, que contém o código HTML usado para renderizá-la. Vamos ver também como usar o Ares para criar visualmente as interfaces ao invés de escrever manualmente o código HTML.

Começaremos diagramando os elementos de entrada e tentando manter a interface bastante simples. Primeiro, adicione um componente `PageHeader` à interface vazia. No painel de configurações, altere o título para *Encontrar Cinemas*. Em seguida, adicione um componente para agrupar logicamente a caixa de entrada de CEP que será incluída em breve e um botão de busca. O botão de grupo se encontra na seção *Layout* da paleta.

Quadro 1: Mojo e JavaScript

O Mojo inclui o framework JavaScript Prototype para manipulação de códigos DOM. Também é possível adicionar o jQuery ou usar outro framework multiplataforma como o Jo.

Clique e arraste-o para mostrar mais da paleta, caso você não o veja.

Depois de adicionar o componente de grupo, altere seu rótulo para *CEP*. Em seguida, arraste um campo de texto (`TextField`) para dentro da caixa de grupo e defina seu `hintText` como *Digite o CEP*, seu tamanho máximo (`maxLength`) como `5` e seu `modifierState` como `num-lock`. Por último, dê a ele o nome de *ZIPCode*. A mensagem definida em `hintText` é exibida sempre que o `TextField` estiver vazio. E ao definir o `modifierState` para `num-lock`, o usuário pode digitar números com facilidade.

Em seguida, será preciso expandir um pouco a caixa de grupo, o que pode ser feito selecionando-a e arrastando o manipulador de controle inferior. Solte um `ActivityButton` na segunda linha da caixa de grupo. Defina o rótulo como *Procurar* e dê a ele o nome de *ZIPSearchButton*. Ajuste a caixa de grupo caso o `ActivityButton` não fique totalmente visível.

O último componente visual será outro `ActivityButton`, que deve ser posicionado abaixo da caixa de grupo. Este botão será usado para buscar por coordenadas de um dispositivo GPS. Dê o nome de `GPSSearchButton` ao botão, e o rótulo de *Buscar nas proximidades*. Sua tela de design deve ser parecida com a da **figura 2**.

O Ares inclui diversos componentes não visuais que incluem diversos serviços, incluindo dois que serão necessários: um componente de webservices (localizado na seção *Web Services* da paleta) e um componente de GPS (localizado em *Sensors*). Arraste os dois para a área de design. Eles aparecerão na área de componentes não visuais, na parte de baixo da área de design (**figura 3**).

Como queremos apenas leituras simples do GPS, selecione o componente GPS, clique em *Settings* e desmarque a caixa *Subscribe*. Para ajudar a economizar bateria, defina o parâmetro `maximumAge` como `300` (segundos), para que você use os valores em cache do GPS caso eles não sejam muito antigos.

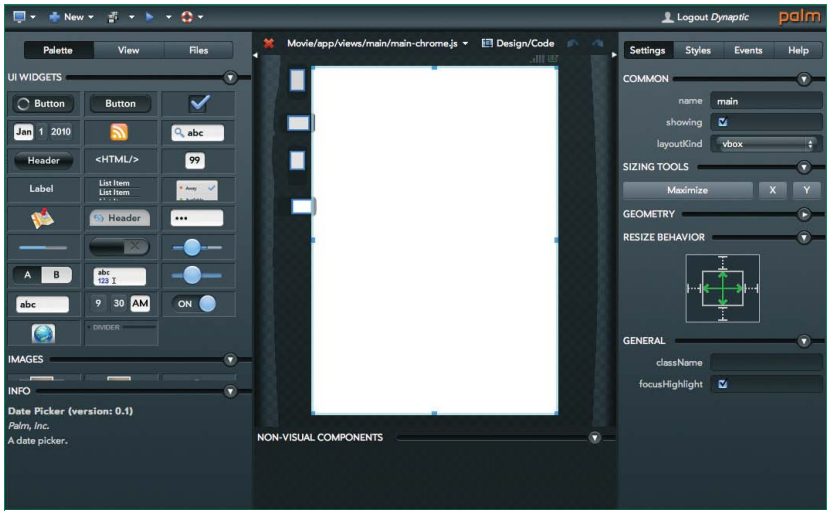

Figura 1 O IDE Ares – uma tela em branco aguarda o início do projeto.

Para uma resposta relativamente rápida, defina a precisão do GPS e seu tempo de resposta (`responseTime`) como `2`. Se você tiver curiosidade a respeito dessas configurações, pode clicar na aba *Help* no canto superior direito do Ares para obter mais informações sobre o componente de GPS.

Em seguida, selecione o componente WebServices e defina a URL como `http://local.yahooapis.com/LocalSearchService/V3/localSearch`. Depois, defina o `handleAs` como JSON. Por último, clique no ícone de parâmetros (logo abaixo de handleAS) e adicione o código da **listagem 1** entre as chaves. Substitua o termo *YahooDemo* pela sua chave de desenvolvedor do Yahoo!, caso tenha uma.

Escrever código

A interface de usuário já está definida, mas isso nem parece programação de verdade. Agora, vamos começar a adicionar algum código para reagir aos eventos do usuário e conectar todos os componentes. O Ares permite adicionar rapidamente *listeners* de eventos a partir do IDE.

Clique no botão de busca de CEP e, no painel mais à direita no Ares, clique na aba *Events* (uma dica: é possível obter uma visão em árvore dos componantes da interface atual clicando na opção *View* no alto do painel esquerdo. A partir dali, é possível clicar em qualquer componente da sua interface). Em *Events* (localizado no painel direito), clique no pequeno ícone de código à direita de `onTap`. O Ares criará uma nova função chama-

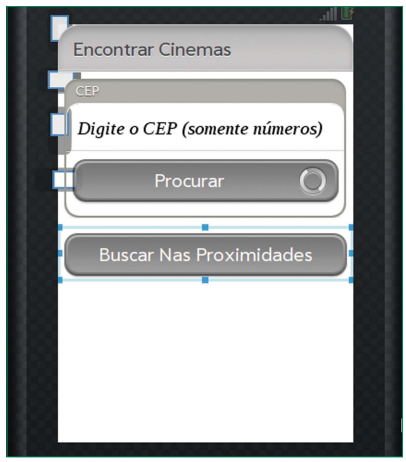

Figura 2 A interface principal com componentes arranjados corretamente.

Figura 3 A seção de componentes não visuais exibe os componentes de GPS e de WebServices.

Listagem 1: Parâmetros de WebServices

```
01 query: “movie theater”,
02 output: “json”,
03 results: “10”,
04 appid: “YahooDemo”
```

Listagem 2: ZipSearchButtonTap

```
01 var zip = this.
02 $.ZIPCode.fetchModelProperty(“value”);
03 this.$.webService1.parameters.zip = zip;
04 this.$.webService1.execute();
```

Listagem 3: webService1Success

```
01 if(inResponse.ResultSet && (inResponse.ResultSet. ➦
   totalResultsReturned > 0))
02 {
03 this.controller.stageController.pushScene(“results”,inResponse. ➦
   ResultSet);
04 }
05 else
06 {
07 Mojo.Controller.errorDialog(“Nenhum resultado encontrado!”);
08 }
```

da `ZIPSearchButtonTap` e alternará a visualização para o editor de código.

Para o Yahoo! saber onde deve procurar as coordenadas, será necessário fornecer à aplicação o CEP que foi digitado pelo usuário. Adicione o código da **listagem 2** à sua nova função.

Talvez você tenha notado o termo `this.$.ZIPCode` no código. O Ares cria automaticamente os componentes para você, e fornece o atalho `this.$` para acessá-los pelo nome. Na primeira instância, é preciso usar o nome atribuído ao `TextField`; na segunda, `webService1` é o nome que o Ares atribuiu ao componante `WebService` quando este foi inserido.

O próximo aspecto interessante é a chamada a `fetchModelProperty`.

Listagem 4: Lista itemHtml

```
01 <div class=”palm-row”>
02 <div class=”theater-name”>#{Title}</div>
03 <div class=”theater-address”>#{Address}</div>
04 <div class=”theater-phone”>#{Phone}</div>
05 <div class=”theater-distance”>Distance: #{Distance} mi.</div>
06 </div>
```

Listagem 5: Função de configuração ResultsAssistant

```
01 this.$.list1.model.items = this.results.Result;
02 this.controller.modelChanged(this.$.list1.model);
```

Listagem 6: Inclusão do arquivo index.html

```
01 <script type=”text/javascript” src=”http://maps.google.com/ ➦
   maps/api/js?sensor=true”></script>
```

Muitos componentes possuem várias configurações e estados que podem ser acessados e ajustados em tempo de execução. Campos do tipo `TextField` têm uma propriedade de valor que contém o texto digitado pelo usuário.

As últimas duas linhas da função adicionam os demais parâmetros necessários à requisição Ajax e em seguida iniciam-na. A chamada para execução retorna imediatamente. Para recuperar os resultados da requisição web, vamos definir manipuladores de sucesso e falha para o objeto `WebRequest`.

Para definir os manipuladores de sucesso e falha, clique no botão *Design/Code* acima do código para retornar a tela de desenho visual da interface. Selecione o componente `webService1` localizado na parte de baixo da tela, na área de componentes não visuais. Na aba *Events*, clique no ícone de código para adicionar o código da **listagem 3** à chamada `onSuccess`.

A chamada `onSuccess` de `WebServices` passa três parâmetros para a aplicação. O mais importante é o `inResponse`, que é o resultado da requisição Ajax. O Yahoo! devolve a resposta em um objeto JavaScript com uma propriedade `ResultSet` legível. Dentro do `ResultSet` há uma propriedade chamada `totalResultsReturned`, que contém o número de respostas. Teste-a para assegurar-se de que você tenha recebido alguns resultados antes de prosseguir.

Você também pode notar no mesmo lugar, a chamada a `pushScene`. Este código está solicitando que o controlador de estágio adicione uma nova interface à pilha de interfaces (substituindo a cena atual como ativa). Você adicionará a nova interface daqui a pouco. Em caso de erro, uma breve mensagem será exibida para alertar o usuário sobre a situação infeliz ocorrida.

A aba *Events* ainda deve ser exibida para que se possa clicar no ícone de código para `onFailure` e adicionar:

```
Mojo.Controller. ➦
errorDialog("Incapaz de me ➦
comunicar com o servidor!");
```

para ter uma mensagem de erro simples.

Nova interface

Após obter uma resposta positiva de um serviço web, o próximo passo é exibir os resultados para o usuário. Para isto, crie uma nova interface com um componente de mapa e um de lista, contendo os resultados.

Clique no menu de seleção *New* no canto superior esquerdo do editor Ares e selecione a opção *Scene*. Nomeie a nova cena como *Resultados* e clique em *OK*. Você verá novamente uma tela em branco. Para começar, adicione um componante de mapa e defina sua altura como `200` (localizada em *Settings/Geometry*, que talvez seja necessário expandir para ser completamente visualizado). Além disso, defina o nível de zoom para `12` e desmarque a opção `showMarker` (ambos localizados em General).

Em seguida, adicione um componente de barra de rolagem (`scroll`) (localizado na seção *Layout* da paleta) abaixo do mapa e depois adicione a ele um componente de lista, defina a altura do componente `scroller` para `100%`. A barra de rolagem garante que a lista se movimente com suavidade, sem fazer o mapa sair do alto da interface.

A plataforma passos são definir o estilo da lista, depois configurar o template para fazê-lo funcionar da forma desejada. Para começar, selecione a lista e, como não queremos que o usuário a modifique, desmarque as opções `swipeToDelete` e `reorderable`. Em seguida, clique no ícone de código `itemHtml`. Apague o conteúdo da caixa de diálogo que será exibida e em seu lugar cole o script da **listagem 4**.

O framework Mojo inclui um sistema de templates. Nós o utilizaremos para preencher a lista com as informações retornadas da requisição web. Após atribuir a lista de cinemas ao modelo de lista, esta automaticamente será populada (após você informar a ela que há novos dados disponíveis).

Para configurar este comportamento, será preciso buscar as informações recebidas da requisição web. Clique no botão *Design/Code*. No topo da janela de código há um construtor para o assistente da cena: o `ResultadosAssistant`. Adicione o seguinte ao corpo da declaração:

```
this.results = argFromPusher;
```

Agora, na função de configuração, adicione o código para informar à lista sobre os dados que você tem. Logo abaixo da chamada a `setupSceneAssistant`, adicione o conteúdo da **listagem 5**.

A primeira linha desta listagem copia o vetor `Result` (onde o Yahoo! inclui a lista de cinemas) no modelo da lista. A segunda linha informa ao controlador da cena que o modelo indicado foi atualizado, para que ele possa atualizar os dados exibidos.

Teste da primeira fase

Apenas para verificar que tudo está funcionando da forma esperada, pode ser uma boa ideia testar o aplicativo neste ponto. Se você ainda não o fez, inicie o *Palm Emulator*. Ao clicar na seta azul no canto superior esquerdo da tela, o aplicativo iniciará o emulador. Primeiro, digite um CEP (experimente 20500 se você não souber um CEP americano) e depois clique no botão *Procurar* de acordo com o nome dado ao botão).

Você receberá um mapa centralizado em Sunnyvale, Califórnia, e uma lista de cinemas próximos à sua localização (**figura 4**). Para voltar à tela principal da aplicação, pressione a tecla [**Esc**] (que simula um deslize de dedo para a direita no dispositivo).

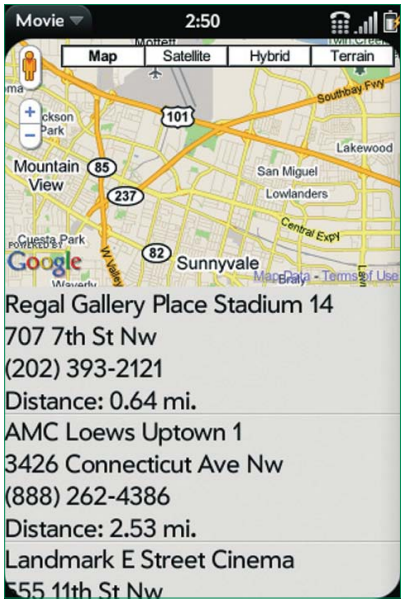

Figura 4 A interface de resultados.

Aqui está o que aconteceu: ao clicar no botão de busca, um evento foi disparado e ativou o manipulador `onTap`. Este, por sua vez, executou uma requisição Ajax no serviço web. Quando a requisição terminou, ela disparou a chamada ao evento `onSuccess`, que enviou os resultados obtidos para a tela. Se não foi o que aconteceu com você, ainda é possível verificar alguns itens para encontrar o problema. De volta ao Ares, você verá um visualiza-

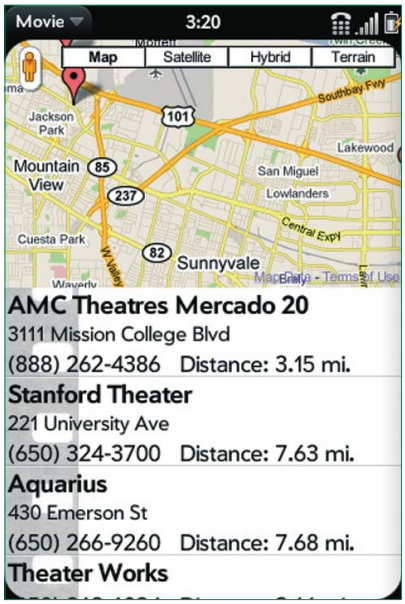

Figura 5 O aplicativo após a inclusão de estilos personalizados.

Listagem 7: gps1Success

```
01 if(inResponse.errorCode == 0)
02 {
03 this.latitude = inResponse.latitude;
04 this.longitude = inResponse.longitude;
05 delete this.$.webService1.parameters.zip;
06 this.$.webService1.parameters.latitude = this.latitude;
07 this.$.webService1.parameters.longitude = this.longitude;
08 this.$.webService1.execute();
09 }
10 else
01 {
11 Mojo.Controller.errorDialog("GPS Error!");
12 this.$.GPSSearchButton.active = false;
13 this.$.GPSSearchButton.activeChanged();
14 }
```

Listagem 8: Sucesso na requisição WebServices

```
01 this.controller.stageController.pushScene("results", inResponse.
   ResultSet,this.latitude,);
```

Listagem 9: Construtor de ResultsAssistant

```
01 this.results = argFromPusher[0];
02 this.latitude = argFromPusher[1];
03 this.longitude = argFromPusher[2];
```

Listagem 10: Folha de estilos movie.css

```
01 body.palm-default {
02 background: url('../images/
   film.jpg') top left repeat-y;
03 }
04
05 .theater-name {
06 padding-left: 3px;
07 font-weight: bold;
08 }
09
10 .theater-address {
11 padding-left: 3px;
12 font-size: .8em;
13 }
14
15 .theater-phone {
16 padding-left: 3px;
17 display: inline;
18 font-size: .9em;
19 }
20
21 .theater-distance {
22 display: inline;
23 padding-left: 10px;
24 font-size: .9em;
25 }
```

dor de logs que exibe as mensagens de erro obtidas, se houverem.

Além disso, é possível experimentar o recurso `JSLint` localizado no mesmo local para verificar se existem erros em seu código-fonte. Ou, é possível usar o depurador integrado para definir alguns pontos de parada no seu código e inspecionar o valor das suas variáveis.

Mapeamento

Agora que já temos uma lista, seria interessante se os pontos de interesse fossem exibidos no mapa (e também seria muito bom se o mapa fosse centralizado na sua localização). Precisaremos usar a API do Google Maps para acessar as classes usadas pelo componente de mapas. Embora este componente importe a API, ele é carregado dinamicamente e não ficará disponível na função de criação. Para contornar isso, importe uma cópia da API dos mapas em seu primeiro carregamento. Uma forma de fazer isto é adicionar esse código ao documento HTML que forma a base do aplicativo. Alterne para a visualização *Files* no painel esquerdo e dê um duplo clique no arquivo `index.html`. Adicione o código da **listagem 6** a ele logo após a instrução que inclui o arquivo `ares.js`.

Para centralizar o mapa e desenhar alguns pontos de interesse, também será preciso adicionar algumas funções. É possível copiar e colar `setMarkers` e `centerMap` a partir do código-fonte [5]. Por último, adicione chamadas a essas duas novas funções no fim da função de configuração. Alterne de volta para o arquivo `results-assistant.js` clicando no menu de seleção no alto do editor de código e selecionando-o. Por último, adicione estas duas linhas à função de configuração:

```
this.setMarkers();
this.centerMap();
```

para adicionar as chamadas às novas funções.

GPS

Agora podemos incluir o dispositivo GPS. No arquivo `main-chrome.js`, selecione o botão *Buscar nas proximidades*, clique em *Events* e adicione um manipulador `onTap` como anteriormente. A Palm possui um serviço para obter coordenadas GPS, e é bom configurá-lo para fornecer leituras contínuas ou uma leitura única.

Como basta uma única leitura para encontrar o local, basta chamar o método `getCurrentPosition()` na aplicação. No manipulador `onTap`, adicione:

```
this.$.gps1.getCurrentPosition();
```

O GPS retorna seus resultados de forma assíncrona. Para que o `gps1` obtenha uma boa leitura, configure a chamada ao evento `onSuccess`. Para

isto, adicione o código da **listagem 7** à chamada de `onSuccess`.

Este código verifica se foi recebida uma leitura válida do GPS, e em seguida inicia o serviço web. Adicionamos o código para retirar o parâmetro de CEP caso o usuário tenha feito uma busca por CEP anteriormente. Também adicionamos código para o manipulador `onTap` do botão da busca por CEP, para apagar a latitude e a longitude de buscas anteriores.

Em seguida, modifique a linha da função `onSuccess` do `WebService` para informar a latitude e a longitude na tela dos resultados. Altere a linha de `pushScene` substituindo-a pelo código da **listagem 8**.

Agora, adicione a seguinte linha à função `ZIPSearchButtonTap` para indicar que você não possui um par de latitude/longitude para buscas por CEP:

```
this.latitude = this.longitude ➥
= undefined;
```

Por último, modifique o construtor `ResultadosAssistant` para aceitar os novos parâmetros e substitua seu código pelo contido na **listagem 9**.

Teste da fase dois

Neste ponto, temos um aplicativo funcional, embora básico, para encontrar cinemas nos EUA. As funções de busca por CEP e por localização GPS devem funcionar. Experimente o aplicativo e certifique-se de que tudo funciona.

Próximos passos

Sem entrar nos detalhes mais finos do design, ainda é possível melhorar o aplicativo, principalmente no que se refere à página de resultados e ao mapa. No mínimo, é possível usar folhas de estilos CSS para melhorar o design do aplicativo. Para começar, será preciso adicionar um arquivo CSS ao projeto. Clique na aba *Files* no painel esquerdo, selecione a pasta do projeto *Movie*, clique no botão da nova pasta e adicione um novo diretório chamado `stylesheets`. Em seguida, clique no botão *New* no canto superior esquerdo e selecione *Document*. Selecione a pasta `stylesheets` e digite `movie.css` como nome do arquivo. Adicione o conteúdo da **listagem 10** ao arquivo `movie.css` recém-criado.

Listagem 11: Inclusão da folha de estilo no arquivo index.html

```
01 <link href="stylesheets/movie.css" media="screen" ➥
   rel="stylesheet" type="text/css" />
```

No arquivo `index.html`, adicione o conteúdo da **listagem 11** ao elemento de cabeçalho. Por último, faça o upload de uma imagem para ser usada como parte do papel de parede do aplicativo. Baixe o o arquivo de imagem `film.jpg` [5] para o seu computador. Em seguida, na visualização *File*, selecione a pasta `images`. Por último, arraste o arquivo do seu computador e solte-o na porção inferior esquerda da tela do Ares, onde diz *Drop local files below to upload to 'images'*. Seu aplicativo deve estar semelhante ao da **figura 5**.

Conclusão

Usando o Ares, vimos como criar rapidamente um aplicativo que combina dados de várias fontes e os exibe para o usuário. Também passamos por vários aspectos da programação em JavaScript para webOS. Se você tiver interesse em outras melhorias, experimente rotular cada um dos resultados e os marcadores no mapa, ou associar um manipulador de clique à lista, para, por exemplo, ligar diretamente para o telefone do cinema para obter informações. Ou então, você poderia adicionar manipuladores `onTap` à lista de cinemas para chamar o telefone do cinema escolhido.

Considere o webOS como plataforma para seu próximo aplicativo. Os usuários de webOS apreciam aplicativos bem escritos e recompensam os desenvolvedores que fornecem recursos inovadores em seus softwares. ■

Mais informações

[1] Palm Developer Center: http://developer.palm.com/

[2] Ares: https://ares.palm.com/Ares/about.html

[3] API de busca de locais do Yahoo!: http://developer.yahoo.com/search/local/V3/localSearch.html

[4] API do Google Maps: http://code.google.com/apis/maps/documentation/javascript/

[5] Código-fonte deste artigo: http://www.smart-developer.com/Resources/Article-Code

Gostou do artigo?

Queremos ouvir sua opinião. Fale conosco em cartas@linuxmagazine.com.br

Este artigo no nosso site: http://lnm.com.br/article/4762