Loiane Groner

App Ionic com HTML5 LocalStorage

Criação da app e commandos úteis

1 – Para criar a app, execute o commando abaixo de dentro do diretório onde estão seus projetos Ionic/Phonegap:

ionic start NomeApp blank

Esse commando irá criar um app sem nenhum componente.

2 – Para compilar o projeto e criar o projeto nativo do Android, usamos o seguinte commando:

ionic platform add android

3 – E para emular localmente no emulador nativo do Android, usamos o commando – lembrar que precisa ter um Android AVD já criado:

ionic emulate android

4 – E para emular no browser (quando não usamos nenhum plugin do phonegap/cordova):

ionic serve

5 – E para emular no browser e visualizar como o app irá ficar no iOS e Android:

ionic serve –l

Criando uma lista para CRUD

O ionic tem um componente chamado de Complex List que suporta as 4 operações CRUD: create (criar), read (listar), update (editar), destroy (deletar). Vamos usar esse componente para criar uma app para gerenciar uma lista de tarefas (TODO).

Dentro da pasta www/js vamos criar dois novos arquivos:

- controllers.js
- services.js

No ***index.html***, logo após o import do ***app.js***, vamos adicionar o import desses dois arquivos:

```
<!-- your app's js -->
<script src="js/app.js"></script>
<script src="js/services.js"></script>
<script src="js/controllers.js"></script>
```

Vamos também mudar o nome do módulo de ***starter*** para ***tasks***.

```
<body ng-app="tasks" ng-controller="TasksCtrl">
```

A diretiva ng-controller informa a app que deverá utilizar o controller TasksCtrl como controller da app.

Dentro do corpo do html, vamos adicionar o seguinte conteúdo, que é referente ao componente Complex List do ionic:

Para mais informações sobre esse componente:
http://ionicframework.com/docs/api/directive/ionList/

Segue a primeira parte do código:

```
<ion-pane>
  <ion-header-bar class="bar-positive">
    <div class="buttons">
      <button class="button button-icon icon ion-ios-minus-outline"
          ng-click="data.showDelete = !data.showDelete; data.showReorder =
false"></button>
    </div>
    <h1 class="title">TODO Tasks</h1>

    <div class="buttons">
      <button class="button" ng-click="data.showDelete = false; add()">
        Add
      </button>
      <button class="button" ng-click="data.showDelete = false;
data.showReorder = !data.showReorder">
        Reorder
      </button>
    </div>
  </ion-header-bar>
```

Essa primeira parte contém o cabeçalho da lista, com suporte aos botões delete, add e reorder que vamos lidar depois no controller.

A segunda parte do código:

```
      <ion-content>

    <ion-list show-delete="data.showDelete" show-
reorder="data.showReorder">

      <ion-item ng-repeat="task in tasks" item="task" class="item-remove-
animate">
        {{ task.name }}
        <ion-delete-button class="ion-minus-circled"
```

```
                ng-click="onItemDelete(task)">
            </ion-delete-button>
            <ion-option-button class="button-calm"
                ng-click="edit(task)">
              Edit
            </ion-option-button>
            <ion-reorder-button class="ion-navicon"
                on-reorder="moveItem(item, $fromIndex, $toIndex)"></ion-
reorder-button>
        </ion-item>

    </ion-list>

  </ion-content>
</ion-pane>
```

O código acima se refere à lista de tarefas.

Criando o Serviço

A camada de serviço no Angular.JS é responsável pela manipulação de dados. É nesse arquivo que vamos adicionar, editar e deletar tarefas do HTML5 LocalStorage.

Dentro do arquivo *services.js* vamos adicionar o seguinte código:

```
angular.module('tasks.services', [])

.factory("TasksService", function () {

        //código CRUD vai aqui

});
```

Lendo lista de tarefas do LocalStorage

As informações são guardadas no localStorage usando chave-valor. Para obter alguma informação, usamos window.localStorage[chave] e para ler usamos window.localStorage[chave] = {informação}.

Dentro do serviço vamos usar o seguinte código:

```
var key = 'tasks';

function getTasks(){
  var tasks = window.localStorage[key];
  if(tasks) {
```

```
    return angular.fromJson(tasks);
  }
  return [];
}
```

Vamos usar o valor "tasks" como chave e vamos retornar a lista de tarefas no formato JSON para o ionic.

Salvando tarefas

Para salvar tarefas no LocalStorage, basta setarmos o novo valor no localStorage:

```
function save(tasks){
  window.localStorage[key] = angular.toJson(tasks);
}
```

O parâmetro tasks é um array JavaScript com a lista das tarefas da App.

Gerenciando Ids (chave primária)

O LocalStorage não faz gerenciamento de chave primária. Se quisermos adicionar essa funcionalidade para faciliar a edição ou remoção de tarefas da lista, temos que faze-la manualmente.

```
function getNextId(){
  var taskId = window.localStorage['taskId'];
  if (taskId){
    taskId++;
  } else {
    taskId = 1;
  }
  window.localStorage['taskId'] = taskId;
  return taskId;
}
```

Com a função acima, podemos chamar toda a vez que criarmos uma nova tarefa. Iremos guardar o Id como valor numeric no localStorage usando a chave "taskId". Assim, toda vez que criamos uma nova tarefa basta incrementarmos esse valor e salvar novamente no localStorage.

Expondo as funções para o Controller

Depois de criar as funções para adicionar e salvar informações no localStorage, temos que deixar essas funções públicas para o Controller. Para isso basta adicionarmos o código abaixo:

```
return {
```

```
  all: function() {
    return getTasks();
  },
  save: function(tasks) {
    return save(tasks);
  },
  getNextId: function(){
    return getNextId();
  }
}
```

Criando o Controller

O Controller é a camada que conversa entre os components da parte visual da App e a camada de serviço. É nele que tratamos os eventos que são disparados pelos components visuais (que iteragem com o usuário da app).

Dentro do arquivo controllers.js vamos adicionar o seguinte código:

```
angular.module('tasks.controllers', ['ionic'])

.controller('TasksCtrl', function($scope, TasksService) {

  //código vai aqui
});
```

Note que na lista da declaração do TasksCtrl estamos passando o objeto **$scope**, que podemos utilizar para manipular valores que estão disponíveis nos componentes (data binding), e o serviço **TasksService** que criamos no tópico anterior.

Mostrando o botão delete ou não

Por padrão, vamos começar não mostrando o botão de deletar tarefas em cada tarefa da lista:

```
$scope.data = {
  showDelete: false
};
```

Assim, somente quando o usuário fizer o touch no botão é que o mesmo será mostrado para o usuário. Essa funcionalidade é controlada automaticamente pelo Ionic.

Lendo a lista de tarefas

Para ler as tarefas do TasksService é simples:

```
$scope.tasks = TasksService.all();
```

Esse código vai chamar o TasksService e carregar toda a lista de tarefas que está no localStorage.

No componente lista, através da diretiva do Angular.JS, a lista sera iterada e iremos mostrar o nome da cada tarefa, como declaramos no ***index.html*** anteriormente:

```
<ion-item ng-repeat="task in tasks" item="task" class="item-remove-animate">
  {{ task.name }}
```

Criando uma nova tarefa

Quando o usuário fizer o touch no botão Add que declaramos no index.html

```
<button class="button" ng-click="data.showDelete = false; add()">
  Add
</button>
```

A função add do Controller sera chamada:

```
var createTask = function(taskName) {
  var newTask = {
    id: TasksService.getNextId(),
    name: taskName
  };
  $scope.tasks.push(newTask);
  TasksService.save($scope.tasks);
};

$scope.add = function() {
  var taskName = prompt('Enter task name');
  if(taskName) {
    createTask(taskName);
  }
};
```

A função add do $scope irá mostrar um alert com prompt e caso o usuário digite alguma coisa e aperte OK, vamos criar uma tarefa nova com esse nome.

Na função createTask estamos criando um novo Objeto JavaScript (newTask) e estamos chamando a função getNextId do TasksService para obtermos qual é o próximo número disponível para o Id. Também estamos setando o nome conforme foi digitado pelo usuário.

No final, vamos adicionar esse objeto novo na lista de tarefas (método push da classe Array do JavaScript adiciona um novo elemento na lista). E vamos pegar todo esse array e passar para o TasksService salvar. E conforme desenvolvemos no TasksService, vamos pegar esse array, transformar em JSON e salvar no localStorage com a chave "tasks".

Editando uma tarefa

Quando o usuário faz um ***swipe*** para a esquerda, um botão de editar irá aparecer para o usuário. Esse comportamento faz parte do componente Complex List do ionic. Esse é o código que declaramos anteriormente no index.html:

```
<ion-option-button class="button-calm" ng-click="edit(task)">
  Edit
</ion-option-button>
```

Quando o usuário tocar nesse botão de Edit, a funcão edit do Controller será chamada passando o objeto task que será editado.

A função edit do Controller tem o seguinte código:

```
var editTask = function(taskName, task){
  for(var i=0; i<$scope.tasks.length; i++){
    if ($scope.tasks[i].id === task.id){
      $scope.tasks[i].name = taskName;
      break;
    }
  }
  TasksService.save($scope.tasks);
};

$scope.edit = function(task) {
  var taskName = prompt('Edit task name');
  if(taskName) {
    editTask(taskName, task);
  }
};
```

Assim como fizemos para criar uma tarefa, também estamos mostrando um alert pro usuário entrar com o novo nome da tarefa. Então chamamos a função editTask passando o novo nome e também o objeto task que será modificado.

Iteramos o array de tarefas que está disponível no $scope e fazemos a comparação através do Id. Caso entramos o id, modificamos o objeto task com o novo nome da tarefa.

No final, salvamos o array de tarefas no localStorage novamente.

Removendo uma tarefa

Para remover uma tarefa, basta tocar no botão de deletar que está no cabeçalho da lista. Isso faz com que cada tarefa da lista mostre seu próprio botão. Esse é o código responsável que declaramos anteriormente:

```
<ion-delete-button class="ion-minus-circled" ng-click="onItemDelete(task)">
</ion-delete-button>
```

Quando o usuário tocar esse botão, a função onItemDelete do controller sera chamada passando o objeto task.

No controller vamos usar o seguinte código:

```
$scope.onItemDelete = function(task) {
  $scope.tasks.splice($scope.tasks.indexOf(task), 1);
  TasksService.save($scope.tasks);
};
```

Vamos usar a função splice que remove um determinado element de um array JavaScript. Então primeiro temos que achar a posição do element (indexOf) e depois usamos a função splice que irá remover apenas 1 elemento contando a partir do índice informado.

Outra forma de fazer essa mesma lógica seria iterar o array para achar o índice como fizemos na função de editar.

No final, apenas salvamos o array de tarefas e atualizamos o valor no localStorage.

Modificando o app.js

Como ultimo passo, precisamos fazer algumas modificações no app.js

```
angular.module('tasks', ['ionic','tasks.controllers','tasks.services'])
```

Precisamos informar o nome da app/modulo e também injetar os módulos de controller e service que criamos.

Enviando para o Ionic View

Após terminar de desenvolver a app, é possível enviar para o Ionic View através do commando:

ionic upload

Informe o Id para que colegas possam testar a app!