libGDX

https://github.com/libgdx/libgdx/wiki/

Rafael Vieira Coelho

rafael.coelho@farroupilha.ifrs.edu.br

Tópicos

- The Application Framework
- A Simple Game
- **File Handling**
- Networking
- Preferences
- Input Handling
- Memory Management
- Audio
- Graphics

https://github.com/libgdx/libgdx/wiki/

Uso de Arquivos

- Aplicações LibGDX rodam em **4 diferentes plataformas**:

  1. Sistemas Desktop (Linux, Windows e MAC OS X)
  2. Android
  3. iOS
  4. Navegador com capacidade de rodar JavaScript/WebGL

- Cada sistema trata a E/S de arquivos de forma diferente.

- Principais Tipos:
  - **Internal**: todos os assets do jogo (imagens, audio, etc.)
  - **Local**: arquivos pequenos (estado do jogo, etc)
  - **External**: arquivos grandes (arquivos baixados da internet, etc.)

- O módulo **Files** permite:
  - Ler de um arquivo;
  - Escrever em um arquivo;
  - Copiar um arquivo;
  - Mover um arquivo;
  - Apagar um arquivo;
  - Listar arquivos e diretórios;
  - Verificar se um arquivo/diretório existe.

Type	*Description, file path and features*	*Desktop*	*Android*	*HTML5*	*iOS*
Classpath	Classpath files are directly stored in your source folders. These get packaged with your jars and are always *read-only*. They have their purpose, but should be avoided if possible.	Yes	Yes	No	Yes
Internal	Internal files are relative to the application's *root* or *working* directory on desktops, relative to the *assets* directory on Android, and relative to the `core/assets/` directory of your GWT project. These files are *read-only*. If a file can't be found on the internal storage, the file module falls back to searching the file on the classpath. This is necessary if one uses the asset folder linking mechanism of Eclipse, see Project Setup	Yes	Yes	Yes	Yes
Local	Local files are stored relative to the application's *root* or *working* directory on desktops and relative to the internal (private) storage of the application on Android. Note that Local and internal are mostly the same on the desktop.	Yes	Yes	No	Yes
External	External files paths are relative to the SD card root on Android and to the home directory of the current user on desktop systems.	Yes	Yes	No	Yes
Absolute	Absolute files need to have their fully qualified paths specified. *Note*: For the sake of portability, this option must be used only when absolutely necessary	Yes	Yes	No	Yes

Verificando Disponibilidade e Caminhos

- Nem todos os tipos de armazenamento podem estar disponíveis.

```
boolean isExtAvailable = Gdx.files.isExternalStorageAvailable();
boolean isLocAvailable = Gdx.files.isLocalStorageAvailable();
```

- E podemos obter o caminho da pasta de armazenamento para arquivos locais e externos.

```
String extRoot = Gdx.files.getExternalStoragePath();
String locRoot = Gdx.files.getLocalStoragePath();
```

- **Como obter gerenciadores de arquivos:**
  - No projeto:

```
FileHandle handle = Gdx.files.internal("data/myfile.txt");
```

  - No projeto do Android:

```
FileHandle handle = Gdx.files.classpath("myfile.txt");
```

  - Na pasta do arquivo jar criado:

```
FileHandle handle = Gdx.files.external("myfile.txt");
```

  - No caminho absoluto do computador a partir da pasta home:

```
FileHandle handle = Gdx.files.absolute("/some_dir/subdir/myfile.txt");
```

Propriedades de Arquivos

- Podemos verificar se um arquivo existe:

```
boolean exists = Gdx.files.external("doitexist.txt").exists();
```

- Podemos verificar se é um diretório:

```
boolean isDirectory = Gdx.files.external("test/").isDirectory();
```

- Podemos listar os arquivos de um diretório:

```
FileHandle[] files = Gdx.files.local("mylocaldir/").list();
for(FileHandle file: files) {
   // do something interesting here
}
```

- Podemos obter a pasta pai de um arquivo e um gerenciador de um arquivo de um diretório:

```
FileHandle parent = Gdx.files.internal("data/graphics/myimage.png").parent();
FileHandle child = Gdx.files.internal("data/sounds/").child("myaudiofile.mp3");
```

Lendo um Arquivo

- Podemos repassar o gerenciador (FileHandle) para quem possa carregá-lo, como em imagens.
- Podemos ler dados binários:

```
FileHandle file = Gdx.files.internal("myblob.bin");
byte[] bytes = file.readBytes();
```

- Podemos ler dados de texto:

```
FileHandle file = Gdx.files.internal("myfile.txt");
String text = file.readString();
```

Escrevendo em um Arquivo

- Dados Textuais:

```
FileHandle file = Gdx.files.local("myfile.txt");
file.writeString("My god, it's full of stars", false);
```

- Dados Binários:

```
FileHandle file = Gdx.files.local("myblob.bin");
file.writeBytes(new byte[] { 20, 3, -2, 10 }, false);
```

Apagando, Copiando, Renomeando e Apagando Arquivos

```
FileHandle from = Gdx.files.internal("myresource.txt");
from.copyTo(Gdx.files.external("myexternalcopy.txt"));

Gdx.files.external("myexternalcopy.txt").rename("mycopy.txt");
Gdx.files.external("mycopy.txt").moveTo(Gdx.files.local("mylocalcopy.txt"));

Gdx.files.local("mylocalcopy.txt").delete();
```

http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/files/FileHandle.html

Tópicos

- The Application Framework
- A Simple Game
- File Handling
- **Networking**
- Preferences
- Input Handling
- Memory Management
- Audio
- Graphics

https://github.com/libgdx/libgdx/wiki/

Rede de Computadores

- LibGDX tem um módulo net com classes que proporcionam operações através da rede:
  - Requisições HTTP
  - Suporte a sockets TCP (cliente e servidor)
  - Acesso a navegadores (abrir link a partir do jogo).

    - Net.java is an interface used for the cross-platform networking. This is where you can get the objects needed to communicate with the network.
    - Socket.java is an interface that provides you with the remote socket address, connection state, and a java.io.InputStream and java.io.OutputStream to work with the socket.
    - SocketHints.java is a class used to configure TCP client sockets
    - ServerSocket.java is an interface used to create TCP server sockets. It provides the standard accept() method to get a TCP client that connected.
    - ServerSocketHints.java is a class used to configure TCP server sockets.
    - HttpStatus.java is a class used to give an easy way to see what the status code returned is.
    - HttpParameterUtils.java is a class used to provide utility methods for HTTP requests.
    - HttpRequestBuilder is a class to help with creating `HttpRequests` .

Criação de Sockets e Requisições HTTP

- Podemos criar um **cliente TCP**:

```
Socket socket = Gdx.net.newClientSocket(Protocol protocol, String host, int port, SocketHints hints);
```

- Podemos criar um **servidor TCP**:

```
ServerSocket server = Gdx.net.newServerSocket(Protocol protocol, int port, ServerSocketHints hints);
```

- Para enviar uma **requisição HTTP**:

```
HttpRequestBuilder requestBuilder = new HttpRequestBuilder();
HttpRequest httpRequest = requestBuilder.newRequest().method(HttpMethods.GET).url("http://www.google.de").build();
Gdx.net.sendHttpRequest(httpRequest, httpResponseListener);
```

- Para enviar uma **requisição HTTP com argumentos**:

```
HttpRequestBuilder requestBuilder = new HttpRequestBuilder();
HttpRequest httpRequest = requestBuilder.newRequest().method(HttpMethods.GET).url("http://www.google.de").content("q=libgdx&example=example").build();
Gdx.net.sendHttpRequest(httpRequest, httpResponseListener);
```

- Para abrir um **link no navegador**:

```
Gdx.net.openURI(String URI)
```

Recebendo Respostas: Opção 1

- Extendendo a classe HttpResponseListener:

```
class MyReceiverOfResult : HttpResponseListener {
   override fun cancelled() {
       // do something when request gets cancelled
   }

   override fun failed(t: Throwable?) {
       // do something when it fails
   }

   override fun handleHttpResponse(httpResponse: Net.HttpResponse) {
       // do something when gets result back
   }
}

...

// assume you hold instance of such class as variable namely `receiver`.
// then you just pass it to `sendHttpRequest()` method
Gdx.net.sendHttpRequest(req, receiver)
```

Recebendo Respostas: Opção 2

- Através de um objeto anônimo:

```
Gdx.net.sendHttpRequest(req, object: Net.HttpResponseListener {
      override fun cancelled() {
          // do something when request gets cancelled
      }

      override fun failed(t: Throwable?) {
          // do something when it fails
      }

      override fun handleHttpResponse(httpResponse: Net.HttpResponse) {
          // do something when gets result back
      }
   })
```

https://github.com/libgdx/libgdx/tree/master/gdx/src/com/badlogic/gdx/net

Tópicos

- The Application Framework
- A Simple Game
- File Handling
- Networking
- **Preferences**
- Input Handling
- Memory Management
- Audio
- Graphics

https://github.com/libgdx/libgdx/wiki/

Preferências

- Trata-se de uma forma simples de armazenar dados pequenos da aplicação: configurações do usuário, estado do jogo, etc.
- Trata-se de um **HashMap** para armazenar pequenas informações.
- Ele é armazenado na pasta home do usuário.
- Obtendo as Preferências:

```
Preferences prefs = Gdx.app.getPreferences("My Preferences");
```

- Alternado as Preferências:

```
prefs.putString("name", "Donald Duck");
String name = prefs.getString("name", "No name stored");

prefs.putBoolean("soundOn", true);
prefs.putInteger("highscore", 10);
```

- Persistindo as alterações:

```
prefs.flush();
```