E1 DIGITAL HUB v1

ELGIN DEVELOPER COMMUNITY

Mais uma vez, através do conhecimento, pesquisa e desenvolvimento, a Elgin Developer Community inova e o maior beneficiado é você, nosso parceiro.

A Elgin Developer Community orgulhosamente disponibiliza ao mercado de Automação Comercial o E1_DigitalHub. Esperamos que com esta nova ferramenta, você, nosso parceiro, perceba que o nosso maior foco é Inovação. Nada menos do que isto. **Inovação do mercado de Automação Comercial**.

Este App é um Intent Server, que contêm todos os módulos presentes na E1_Android, a diferença é que instalando o App você não precisa configurar nada em sua Automação Comercial, o E1_DigitalHub resolve tudo. Através do nosso protocolo de comunicação você consegue enviar dados e obter as respostas de operações com a Impressora Térmica, com o SAT, com a Balança etc.

A seguir, vamos dar uma breve introdução sobre as tecnologias aplicadas e como as utilizar.

INTENT

Para quem ainda não teve a oportunidade de trabalhar com Intent, fique calmo, não é um bicho de sete cabeças. Vamos falar um pouco sobre.

O Intent é um objeto nativo Android, muito utilizado para realizar troca de mensagens entre Activities de uma mesma aplicação ou entre Apps distintos. Isto ocorre por conta de filtros definidos no Manifest.xml da aplicação, o sistema Android entende que determinada Activity pode receber mensagens quando algum Intent for instanciado com o caminho definido do filtro.

EXEMPLO

```
<manifest ...>
  <application ...>
    <activity
      android:name=".MainActivity"
      android:exported="true">
      // definição do filtro para troca de mensagens
      <intent-filter>
        // quando indicado este caminho ele está elegível a receber mensagens
        <action android:name="teste.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

Basicamente para se utilizar um Intent você precisa:

1. Instanciar um objeto atribuindo o nome do componente quer falar (caminho do filtro);
2. Adicionar os dados que deseja enviar (Payload);
3. Realizar o envio.

UTILIZANDO INTENTS

Nesta sessão veremos:

1. Envio de dados ao E1_DigitalHub;
2. Retorno dos processamentos feitos pelo E1_DigitalHub.

Agora que já tivemos uma breve definição de Intent, vamos como utilizar no nosso contexto seguindo os três passos.

ENVIO

Vamos fazer um exemplo de instancia de um Intent com caminho para o módulo Térmica. O conteúdo que será enviado está detalhado na sessão **Utilizando Payloads – Envio**, deve ser adicionada ao **putExtra** com a chave "***comando***".

EXEMPLO

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

        try {
            // 1) definição do filtro para falar com Activity de TERMICA em E1_DigitalHub
            String TERMICA = "com.elgin.e1.digitalhub.TERMICA";
            // 1.1) realiza a instancia do objeto
            Intent intent = new Intent(TERMICA);
            // 2) adiciona o putExtra com o Payload de Envio
            intent.putExtra("comando", "[{...}]");
            // 3) envio de dados para o E1_DigitalHub
            startActivityForResult(intent, 1);
        } catch (Exception e) {
             Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
}
```

RETORNO

Após realizar o envio, vamos obter o retorno. São dois passos:

1. Implementar a sobrecarga da função **onActivityResult()**;
2. Obter o valor de **getStringExtra** com a chave "***retorno***" com o valor de **Utilizando Payloads – Retorno** com Array (Json) de retornos dos métodos invocados.

Vamos realizar a implementação da sobrecarga do método **onActivityResult()**.

EXEMPLO

```
@Override
// 1) sobrecarga da função onActivityResult()
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        try {
            // 2) recebe o retorno do E1_DigitalHub (um Intent de retorno)
            String retorno = data.getStringExtra("retorno");
            Toast.makeText(getApplicationContext(), "Retorno = " + retorno, Toast.LENGTH_LONG).show();
        } catch (Exception e) {
          Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_LONG).show();
        }
}
```

Agora com conteúdo de retorno, basta realizar os devidos tratamentos para obter o conteúdo e utilizar em sua Automação.

PAYLOAD

De forma rasa, os Payloads são as informações em um bloco de dados que você envia ou recebe de um servidor ao fazer solicitações. Estes blocos podem estar em vários formatos e um deles é o Json, o que usamos para criar nosso protocolo de comunicação com a E1_DigitalHub.

O Json é um formato de texto simples para armazenar e transmitir dados. Este formato é muito utilizado para enviar e receber dados de um servidor, creio que seja o maior caso de uso. Uma de suas características que, no meu ponto de vista, é bem relevante, são os formatos suportados:

- String
- Number
- Boolean
- Null/Empty
- Array (Json)
- Object (Json)

Vamos ver como é composto nosso protocolo proprietário e quais são os atributos que são utilizados para o E1_DigitalHub.

UTILIZANDO PAYLOADS

Nesta sessão veremos:

1. Payload de envio;
2. Payload de retorno.

ENVIO

Os componentes Json escolhidos para realizar o Payload de envio serão estes três:

- String
- Array (Json)
- Object (Json)

EXEMPLO

```
[                                   // Array (Json)
        {                           // Object (Json)
                "key": "value"      // String
        }
]
```

O componente raiz é o Array (Json), ele terá um ou mais Objects (Json), representará uma lista de "***comandos***" que serão invocados. Estes comandos serão compostos por Objects (Json). Cada um destes Objects (Json) deve ter dois componentes:

1. String – será o nome do método que chamaremos;
2. Object (Json) – serão os parâmetros deste método.

Para melhor contextualização, vamos fazer a chamada de um método AbreConexaoImpressora do módulo Térmica (E1_Android): AbreConexaoImpressora(int, String, String, int)

EXEMPLO

```
[ // início do Array (Json)
        {  // início do Object (Json)
                "funcao": "AbreConexaoImpressora", // String com nome da função
                "parametros": {              // início do Object com os parâmetro
                        "tipo": 5,           // chave: tipo e valor: 5 (int)
                        "modelo": "",        // chave: modelo e valor: vazio (String)
                        "conexao": "",       // chave: conexao e valor: vazio (String)
                        "parametro": 0       // chave: parametro e valor: 0 (int)
                }                            // final do Object com os parâmetro
        }  // final do Object (Json)
]  // final do Array (Json)
```

OBS: Veja, estamos fazendo a chamada de somente um método.

Agora vamos ver como ficaria um caso real para realizar uma impressão de texto. Vamos chamar três funções:

1. AbreConexaoImpressora;
2. ImpressaoTexto;
3. FecharConexaoImpressora.

EXEMPLO

```
[
	{
		"funcao": "AbreConexaoImpressora",
		"parametros": {
			"tipo": 5,
			"modelo": "",
			"conexao": "",
			"parametro": 0
		}
	},
	{
		"funcao": "ImpressaoTexto",
		"parametros": {
			"dados": "teste de impressão",
			"posicao": 0,
			"stilo": 0,
			"tamanho": 0
		}
	},
	{
		"funcao": "FechaConexaoImpressora",
		"parametros": {}
	}
]
```

OBS: Basta seguir estes modelos para realização as chamadas dos métodos!

RETORNO

Assim como o Envio, o Retorno também é composto por um componente raiz Array (Json). Este contém um ou mais Object (Json), representará uma lista de "***retornos***" dos métodos que foram solicitados. Estes retornos serão compostos por Objects (Json).

Os Objects (Json) são compostos por três componentes:

1. String – com o nome da função que foi invocada;
2. String – com uma mensagem de execução ou exception gerada;
3. Object – com o resultado da função (se ela retornar um int, será int, se ela retorna String, será String etc).

EXEMPLO

```
[ // início do Array (Json)
        { // início do Object (Json)
                "funcao": "Método",                    // String com nome da método
                "mensagem": "Executado”,               // String com mensagem de retorno
                "resultado": 0                         // Object com resultado
         } // final do Object (Json)
] // final do Array (Json)
```

Agora vamos ver como ficaria o retorno dos três métodos do exemplo de Envio.

EXEMPLO

```
[
        {
                "funcao": "AbreConexaoImpressora",
                "mensagem": "Método executado",
                "resultado": 0
        },
        {
                "funcao": "ImpressaoTexto",
                "mensagem": "Método executado",
                "resultado": 0
        },
        {
                "funcao": "FechaConexaoImpressora",
                "mensagem": "Método executado",
                "resultado": 0
        }
]
```

OBS: Note que o resultado é int, no caso de funções que retornam String, o campo fica elegível para String.

INFORMAÇÕES GERAIS

FILTER ACTIONS PARA OS MÓDULOS

Caminhos dos filtros para criação dos Intents para cada módulo:

- **SAT** - "com.elgin.e1.digitalhub.SAT"
- **BRIDGE** - "com.elgin.e1.digitalhub.BRIDGE"
- **TERMICA** - "com.elgin.e1.digitalhub.TERMICA"

REFERENCIAS

INTENT

- https://developer.android.com/guide/components/intents-filters
- https://www.youtube.com/watch?v=aaSepKIO96E
- https://www.youtube.com/watch?v=4eso_7RyZ58

JSON

- https://www.digitalocean.com/community/tutorials/an-introduction-to-json
- https://www.youtube.com/watch?v=MkUY6ebsaXs
- https://www.youtube.com/watch?v=BWPUSXzSWA8

PAYLOAD

- https://reqbin.com/req/2xhbguy8/json-payload-example#:~:text=What%20is%20JSON%20Payload%3F,variety%20of%20formats%2C%20including%20JSON