JPCERT CC®

JEB Plugin 開発チュートリアル 第3回

– バイトコードについての理解 –
JEB Pluginからバイトコードを扱う方法を修得する

一般社団法人JPCERTコーディネーションセンター

目次

- 第0回 JEBとは？
- 第1回 JEB Pluginとは
  - 1. JEB Pluginの使い方
  - 2. JEB Pluginの構造
  - 3. JEBのUIを利用するためのAPI
  - 4. ViewとSignature
- 第2回 DEXファイルの構造を理解する
  - 1. DEXファイルの構造
  - 2. jeb.api.dex
  - 3. クロスリファレンス
- **第3回 バイトコードについての理解**
  - **1. CodeItem**
- 第4回 JEB PluginからASTを扱う

1. CODEITEM

DexCodeItem

- DEXのcode_itemにはメソッド定義が入っている
  - JEBではDexCodeItemクラスが担当している

- DexCodeItemクラスで使用できる主なメソッド
  - getDebugInfo()
    - メソッドのデバッグ情報を取得
  - getExceptionItems()
    - メソッドのException Itemを取得
  - getInstructions()
    - パースされたバイトコード命令列を取得
      - DexDalvikInstructionを取得できる
  - Count系
    - getInputArgumentCount()
    - getInstructionCount()
    - getOutputArgumentCount()
    - getRegisterCount()

DexDalvikInstruction

- バイトコード命令が含まれている

- getMnemonic()
  - —オプコードを文字列化したもの
- getCode()
  - — バイナリデータ
- getOffset ()
  - —methodの先頭からのバイト数
- getParameters()
  - —オペランドの取得(配列)
- その他
  - —getSwitchData() → packed-switch, sparse-switch
  - —getArrayData() → fill-array-data
    - switch, array系の命令の時に使われる

例題1

- メソッド内のオプコードを表示しよう
  - —DexCodeItemクラスとDexDalvikInstructionクラスの使い方を理解する

[例題1] メソッド内のオプコードの表示

- 課題
  - Assembly Viewでフォーカスしているメソッドのオフセットとオプコードの一覧を表示するPluginを作成する
- 期待する出力結果

```
[Lcom/example/contentprovider/MainActivity;->showListView()V] assembly
0x000000: const/4
0x000002: new-instance
0x000006: invoke-direct
0x00000c: iput-object
0x000010: sget-object
0x000014: const-string
0x000018: move-object
0x00001a: move-object
0x00001c: move-object
0x00001e: invoke-virtual/range
0x000024: move-result-object
0x000026: invoke-interface
0x00002c: move-result
0x00002e: if-nez
```

- ヒント
  - フォーカス位置のSignatureの取得
    - View → CodePosition → getSignature()
  - 命令列の取得
    - CodeItem.getInstructions()
  - offsetとオプコードの取得
    - DexDalvikInstruction.getOffset()
    - DexDalvikInstruction.getMnemonic()

例題1の解答例

- Opcode.py

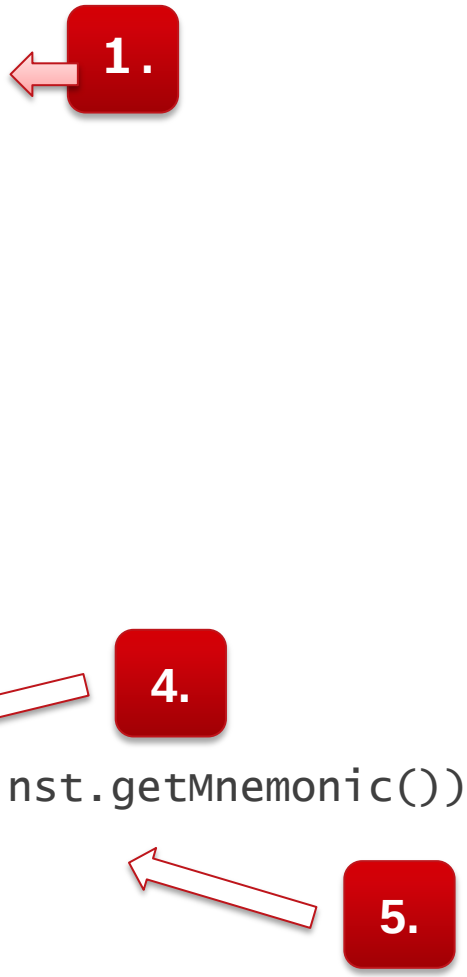
[解説] メソッド内のオプコードの表示

```
view = self.ui.getView(View.Type.ASSEMBLY)
sig = view.getCodePosition().getSignature()

md = self.dex.getMethodData(sig)
if md is None:
    print "method data is not found."
    return

code_item = md.getCodeItem()
if code_item is None:
    print "code item is not found."
    return

print "[%s] assembly" % sig
for inst in code_item.getInstructions():
    print "%#08x: %s" % (inst.getOffset(), inst.getMnemonic())
```



[解説] メソッド内のオプコードの表示

1. フォーカス位置のSignatureを取得する

```
sig = view.getCodePosition().getSignature()
```

1. Signatureを使用してmethod_dataを取得する

```
md = self.dex.getMethodData(sig)
```

2. method_dataからcode_itemを取得する

```
code_item = md.getCodeItem()
```

3. code_itemからバイトコード命令列を取得し

```
for inst in code_item.getInstructions():
```

4. offsetとオプコードを出力する

```
 print "%#08x: %s" % (inst.getOffset(), inst.getMnemonic())
```

DexDalvikInstruction.Parameter

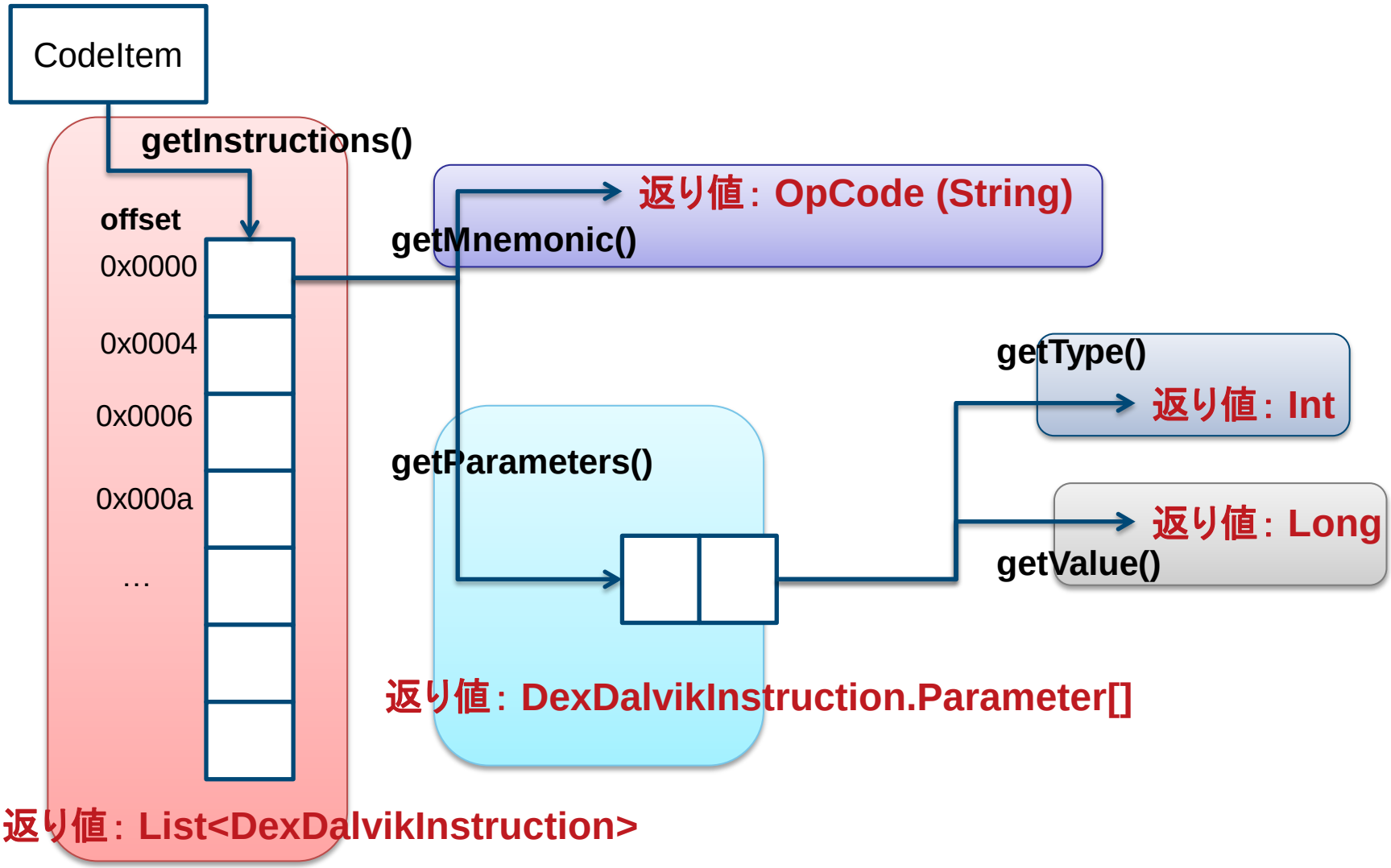
- Dalvikインストラクションのパラメータを扱うクラス

- getType() – パラメータ種別を取得する
  - TYPE_REG (0)
    - register レジスタ番号
  - TYPE_IMM (1)
    - immediate value 即値
  - TYPE_IDX (2)
    - Index (method id, field id, class id…)
  - TYPE_BRA (3)
    - branch target → goto 相対アドレス
  - TYPE_RGR (4)
    - register range レジスタの範囲
- getValue() – パラメータ値を取得する

- 型と命令に合わせて変換が必要 → そのままは使えない
  - Indexだったら、どのListを参照する命令かによって参照するものが違ってくる
- 特定の命令だけを参照する場合は、決め打ちで対応できる
  - invoke-xxx → 必ずメソッドIDの参照
- どの命令がどのような引数を取るかは仕様を参照
  - http://www.android-decompiler.com/help/dalvik-bytecode.html#op_nop

DexDalvikInstructions

- DexDalvikInstructionsクラスの各メソッドの関連図



例題2

- メソッド内のメソッド呼出一覧を表示しよう
  —DexDalvikInstructionクラスをもっと理解する

[例題2] メソッド内のメソッド呼出一覧の表示

- 課題
  - Assembly Viewでフォーカスしているメソッドの命令の中でメソッド呼び出ししているもの一覧をコンソールに表示する
- 期待する出力結果

```
[Lcom/example/contentprovider/MainActivity;->showListView()V]
invoked method
    Ljava/util/ArrayList;-><init>()V
    Lcom/example/contentprovider/MainActivity;->managedQuery(Landroid/net/Uri;[Ljava/lang/String;Ljava/lang
    Landroid/database/Cursor;->moveToNext()Z
    Lcom/example/contentprovider/MainActivity;->getApplicationContext()Landroid/content/Context;
    Lcom/example/contentprovider/MyListAdapter;-><init>(Landroid/content/Context;Ljava/util/ArrayList;)V
    Landroid/widget/ListView;->setAdapter(Landroid/widget/ListAdapter;)V
    Landroid/widget/ListView;->setOnItemClickListener(Landroid/widget/AdapterView$OnItemClickListener;)V
    Lcom/example/contentprovider/MyItem;-><init>()V
    Landroid/database/Cursor;->getInt(I)I
    Lcom/example/contentprovider/MyItem;->setLevel(I)V
    Landroid/database/Cursor;->getString(I)Ljava/lang/String;
    Lcom/example/contentprovider/MyItem;->setIdentifier(Ljava/lang/String;)V
    Lcom/example/contentprovider/MyItem;->setTitle(Ljava/lang/String;)V
```

- ヒント
  - CodeItem → DexDalvikInstructionsの取得
  - DexDalvikInstruction. getMnemonic()で命令を取得
    - “invoke-xxx”だったらメソッド呼び出し
  - パラメータの取得
    - DexDalvikInstruction.getParameters()
    - メソッド呼び出しの場合はパラメータ値はメソッドIndex
      - 対応するメソッド名を表示する
        - Dex.getMethodSignatures()

例題2の解答例

- InvokedMethod.py

[解説] メソッド内のメソッド呼出一覧の表示

```
view = self.ui.getView(View.Type.ASSEMBLY)
sig = view.getCodePosition().getSignature()

md = self.dex.getMethodData(sig)
if md is None:
    print "method data is not found."
    return

code_item = md.getCodeItem()
if code_item is None:
    print "code item is not found."
    return

print "[%s]" % sig
print "invoked method"
method_ids = []
for inst in code_item.getInstructions():
    if inst.getMnemonic().startswith('invoke-'):

        mid = inst.getParameters()[0].getValue()
        if mid not in method_ids:
            method_ids.append(mid)

msigs = self.dex.getMethodSignatures(False)
for mname in map(lambda x: msigs[x], method_ids):
    print "¥t" + mname
```

1.

2.

3.

4.

5.

[解説] メソッド内のメソッド呼出一覧の表示

1. フォーカス位置のSignatureを取得し、method_dataを取得する
2. method_dataからcode_itemを取得する
3. code_itemからバイトコード命令列を取得する
4. DexDalvikInstruction.getMnemonic()でバイトコード命令を取得する
   - バイトコード命令が“invoke-”から始まるものであれば、**メソッド呼び出し**なのでその値を取得する。
     - **メソッド呼び出し**の時のパラメータ値はメソッドIndexになる
5. Dex.getMethodSignatures()でSignature一覧を取得し、メソッドIndexに対応するメソッド名(Signature)を表示する

例題3

- メソッド内のオプコードを表示しよう その2
  - —DexDalvikInstruction.getParameters()

[例題3] メソッド内のオプコードの表示 その2

- 課題
  - 演習2で作成したメソッドのオフセットとオプコードの一覧を表示するPluginを次のとおり拡張する
    - 全てのオペランドの種別と値の表示を追加する
    - 種別はLongから文字列に変換する
      - 文字列 → REG, IMM, IDX, BRA, RGR
- 期待する出力結果

```
[Lcom/example/contentprovider/MainActivity;->showListView()V] assembly
0x000000: const/4 params:2
    REG: 2(0x2)
    IMM: 0(0x0)
0x000002: new-instance params:2
    REG: 0(0x0)
    IDX: 601(0x259)
0x000006: invoke-direct params:2
    IDX: 3871(0xf1f)
    REG: 0(0x0)
0x00000c: iput-object params:3
    REG: 0(0x0)
    REG: 9(0x9)
    IDX: 829(0x33d)
```

- ヒント
  - オペランド(パラメータ)の取得
    - DexDalvikInstruction.getParameters()
  - 種別: DexDalvikInstruction.Parameter.getType()
  - 値: DexDalvikInstruction.Parameter.getValue()

例題3の解答例

- Opcode2.py

[解説] メソッド内のオプコードの表示 その2

```
param_type = [ 'REG', 'IMM', 'IDX', 'BRA', 'RGR', ]
```

種別を配列に入れておく

```
for inst in code_item.getInstructions():
    print "%#08x: %s params:%d" % (inst.getOffset(), inst.getMnemonic(),
            len(inst.getParameters()))
    for p in inst.getParameters():
        print "¥t%s: %d(%#x)" % (param_type[p.getType()],
                p.getValue(), p.getValue())
```

getType()の返り値を添字として指定すれば、目的の文字列を取得できる