Cuarta semana de trabajo [21 - 25 de abril]

Milton Inostroza Aguilera

27 Abril de 2008

Resumen

Se propone un borrador del protocolo de comunicación que se utilizará para comunicar pyTOD {Python} con la base de datos TOD {Java}.

Indice

Lista de Figuras

Lista de Tablas

1 Desarrollo

La semana comenzó por solucionar el problema que se tenía con settrace para los threads.

Los threads en python implementados por la clase threading, no eran detectados por el capturador de huella, ya que utilizan su propio *sys*. Investigando un poco este módulo se logra plantear la siguiente solución:

- Utilizar el método settrace del módulo threading y enviar nuestro capturador de huella.
- Importar desde nuestro capturador de huella el módulo threading, en especial la funcion settrace.
- Al hacer lo anterior, al momento que el programador importe la librería threading desde su script, el sistema no lo importará debido a que ya existe una referencia de importanción a ese módulo. Con lo anterior nos aseguramos que los threads implementados por el programador utilizarán nuestro capturador de huellas. Para mas información: [1]
  Proceso de Importación:

  1. Se comprueba si existe una clave en 'sys.modules' con el nombre del módulo. Si existe, se usa el valor de esa clave para evitar "reimportar" el módulo. Así terminaría el proceso de importación. (Es la razón por la que no debe dolernos reimportar un módulo tantas veces como queramos).

  2. Si no existe ninguna clave con el nombre del módulo, se crea en 'sys.modules' una entrada nueva con el nombre del módulo como clave y se inicializa como módulo vacío (types.ModuleType).

  3. Este módulo vacío se utiliza como "espacio de nombres" y se irá "poblando" de referencias a medida que se ejecute el módulo.

     ¿Qué pasa si durante la ejecución del módulo se importan otros módulos?

  4. Si durante la ejecución del código se importan nuevos módulos, éstos módulos van a ver al primogénito como ya importado, pero en realidad todavía estará sin inicializar del todo. Habrá referencias que todavía no habránn sido creadas ya que iban en orden por detrás de las importaciones.

2 Protocolo de comunicación - Draft

Para poder enviar los datos a la bases de datos de TOD se utilizarán socket's. Para codificar se utilizara la librería xdrlib que está basada en el estandar rfc1832 [2]

Se muestran las tablas bases para el registro de los objetos y para el envío de eventos:

2.1 Identificadores bases

La siguiente tabla muestra que cada suceso tiene un identificador en el sistema de capturación de huella.

Suceso	Identificador
Registro	0
Llamada	1
Retorno	2
Asignacion	3

Tabla 1: Identificadores de sucesos

La siguiente tabla muestra que cada objeto tiene un identificador en el sistema de capturación de huella.

Id Objeto	Identificador
Clase	0
método	1
Atributo	2
Funcion	3
Variable local	4
Probe	5
Thread	6

Tabla 2: Identificadores de objetos

La siguiente tabla muestra que cada tipo de datos un identificador en el sistema de capturación de huella.

Type	Identificador
int	0
str	1
float	2
long	3
bool	4
other	5

Tabla 3: Identificadores de tipo de datos

2.2 Registro de objetos

A continuación se muestra el formato que tienen el registro de los diferentes objetos dentro del capturador de huellas:

Se describe el registro del objeto función:

```
eventId | objectId | Id | name | argsN | {argName1 | argId1}^n
```

Se describe el registro del objeto variable local:

```
eventId | objectId | Id | parentId | name
```

Se describe el registro del objeto clases:

```
eventId | objectId | Id | name | classBases
```

Se describe el registro del objeto método:

```
eventId | objectId | Id | classId | name | argsN | {argName | argId}^n
```

Se describe el registro del objeto atributo:

```
eventId | objectId | Id | parentId | name
```

Se describe el registro del objeto thread:

```
eventId | objectId | Id | sysId
```

Se describe el registro del objeto probe:

```
eventId | objectId | Id | currentLasti | parentId
```

2.3 Llamada de objetos

A continuación se muestra el formato que tienen las llamadas de los objetos función y método dentro del capturador de huellas:

Se describe la llamada al objeto función:

```
eventId | objectId | Id | parentId | argsN | {argId | argValue}^n
```

Se describe la llamada al objeto método:

```
eventId | objectId | Id | parentId | classId | argsN | {argId | argValue}^n
```

Es importante señalar que todas estas llamadas estan acompañadas de los siguientes datos que se describen a continuación:

```
probeId | parentTimeStampFrame | depth | currentTimeStamp | threadId
```

2.4 Asignación - Modificación de objetos

A continuación se muestra el formato que tienen las asignaciones — modificaciones de los objetos variable local y atributo dentro del capturador de huellas:

Se describe la asignación - modificacion al objeto variable local:

```
eventId | objectId | Id | parentId | value
```

Se describe la asignación - modificación al objeto atributo:

```
eventId | objectId | Id | parentId | value
```

Es importante señalar que todas estas asignaciones - modificaciones estan acompañadas de los siguientes datos que se describen a continuación:

```
probeId | parentTimeStampFrame | depth | currentTimeStamp | threadId
```

2.5 Return

A continuación se muestra el formato que tiene el return dentro del capturador de huellas:

Se describe return:

```
eventId | probeId | hasThrown
```

3 Lecturas recomendadas

Se recomienda leer y revizar los siguientes proyectos:

- PyChecker, analisis estatico
- PyLint, analiza si el codigo satisface la codificicion estandar
- Py_Compile, compila archivos .py
- pdb, depurador break-point

Bibliografía

[1] `http://listas.aditel.org/archivos/python-es/2007-March/016605.html`

[2] `http://www.faqs.org/rfcs/rfc1832.html`