

Original: [How to access S60 resources in WRT or FlashLite, using PyS60](#)

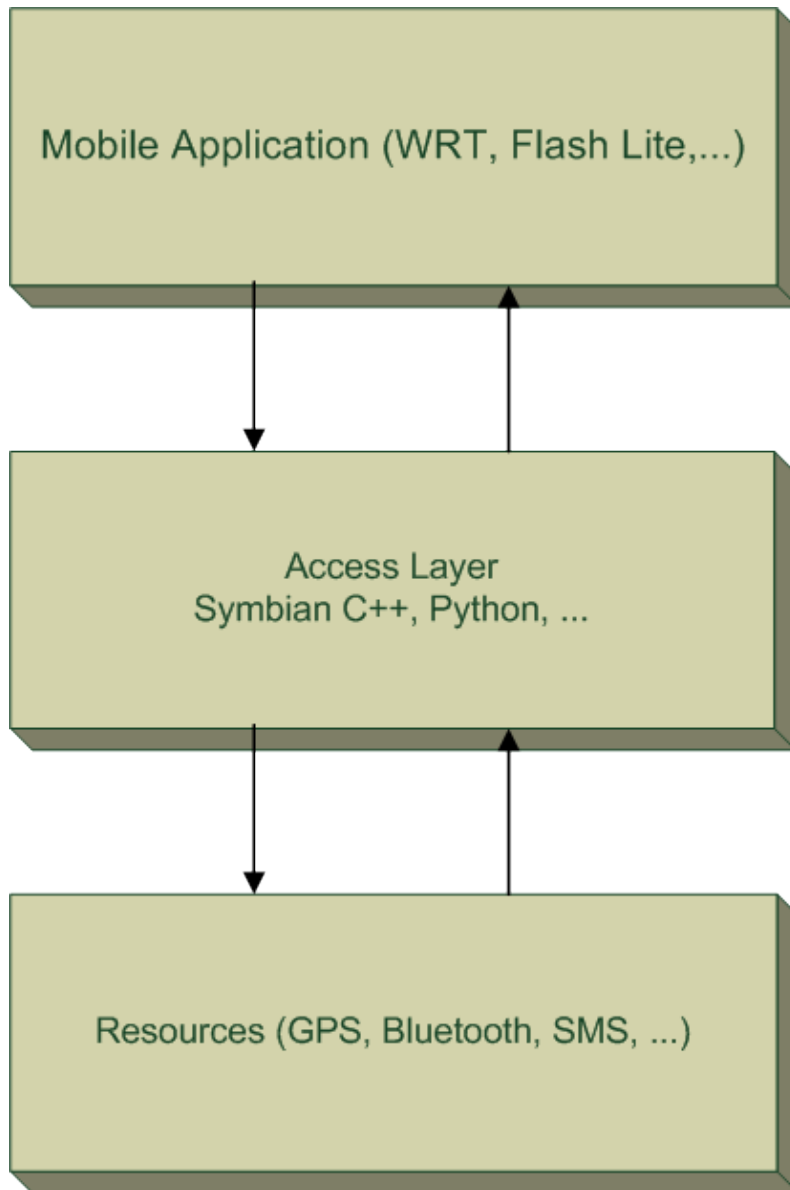
Contents

- [1 Introdução](#)
- [2 Solução](#)
- [3 O Código](#)
- [4 Passo-a-passo para instalar e usar o ServerPython](#)
- [5 Download](#)
- [6 Um exemplo de uma aplicação em Flash Lite](#)
- [7 Criado por](#)

Introdução

Algumas linguagens de desenvolvimento recém introduzidas na plataforma S60, como Flash Lite ou WRT, possuem algumas limitações de acesso a alguns recursos nativos do dispositivo como GPS, Acelerometro etc. Este problema é ocasionado devido ao fato que tais aplicações são executadas em uma *sandbox* como forma de evitar que códigos maliciosos ou mal projetados possam vir a danificar ou causar instabilidade ao sistema. Uma abordagem que pode ser utilizada com o intuito de eliminar essa limitação de acesso é o desenvolvimento de uma camada de acesso aos recursos. Tal camada deve ser localizada entre o dispositivo e a aplicação móvel (Flash Lite ou WRT). Esta camada de acesso pode ser desenvolvida em qualquer linguagem de programação que tenha acesso aos recursos nativos do dispositivo.

A figura abaixo mostra como esta camada de abstração funciona.



Solução

Para o desenvolvimento da camada de acesso supracitada, é proposto a implementação de um servidor *web* escrito em Python para ser executado no dispositivo, cujo principal objetivo é possibilitar um a recuperação dos serviços nativos utilizando uma abordagem baseada em *Web service REST*. O servidor *web* foi desenvolvido levando-se em consideração a extensibilidade de forma que novos serviços possam ser adicionados, separando o processamento da requisição e a invocação do serviço da lógica do serviço em si. Sendo assim, no desenvolvimento do serviço é possível adicionar apenas o código da lógica de negócio do serviço, facilitando a inclusão de novos serviços.

A figura 2 apresenta a arquitetura do servidor *web* desenvolvido. Existem três entidades principais associadas com a arquitetura: cliente, servidor HTTP e os serviços.

- Cliente: a aplicação escrita em qualquer tecnologia, como Flash Lite ou Wrt, que necessita de acesso

Como acessar os recursos de dispositivos S60 em WRT ou Flash Lite usando PyS60

aos recursos internos do dispositivo. esta aplicação é completamente independente do *ServerPython*, visto que ele faz seu acesso utilizando apenas requisições HTTP.

- Servidor HTTP: gerencia todas as requisições dos clientes e invoca os serviços solicitados.
- Serviços: um conjunto de serviços implementados pelo desenvolvedor. Estes são os *hotspots* do *framework ServerPython*.

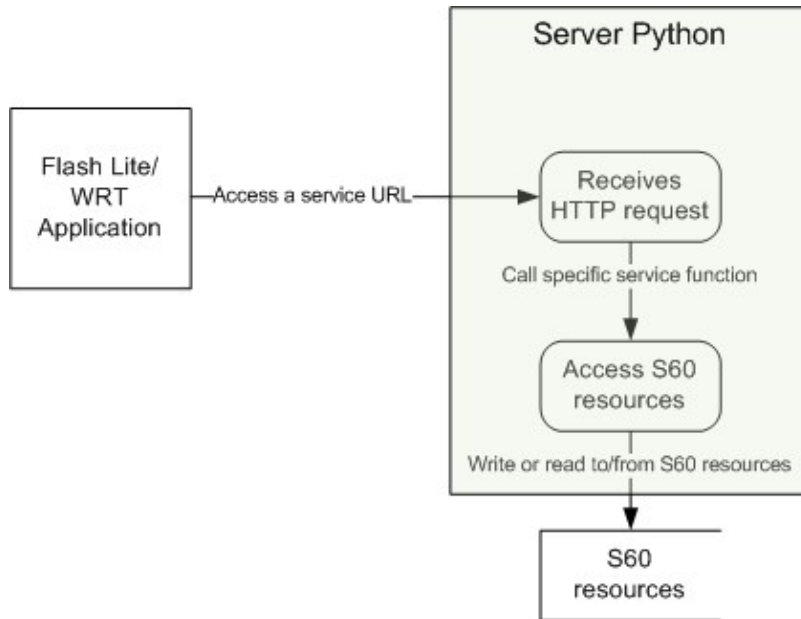


Figura 2. Arquitetura do servidor *Web*

A figura 3 apresenta o diagrama de classes do *ServerPython*. A classe *Services* é o *hotspot* do framework. O desenvolvedor tem de criar os métodos e chamar a função *addCallBack* da classe *MyServer* para registrar o serviço a ser invocado.

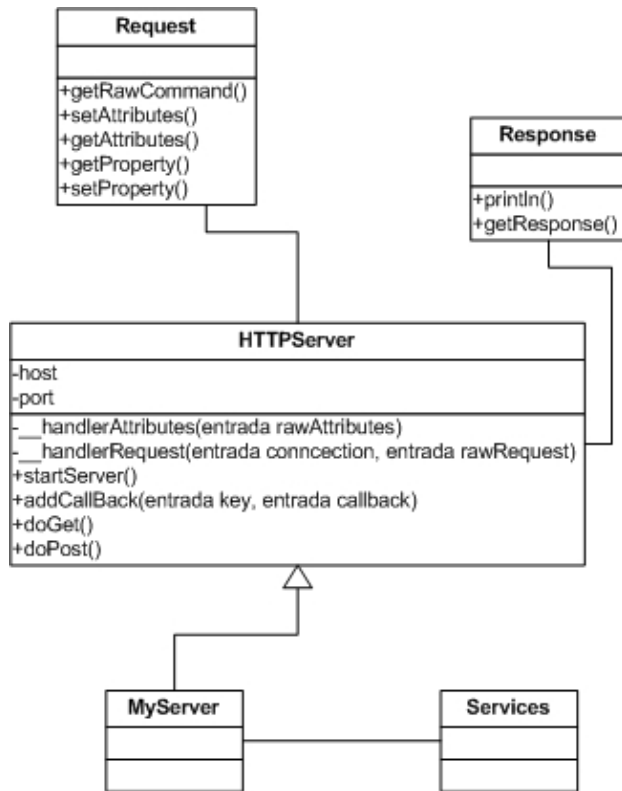


Figura 3. Diagrama de Classe

O Código

Nesta seção, são apresentadas algumas características do servidor web desenvolvido.

O código abaixo apresenta a classe *HTTPServer* que efetua o processamento das requisições HTTP.

```

class HTTPServer:
    def __init__(self, host, port):
        """This class is responsible for parsing the HTTP request and call the correct method"""
        self.host = host;
        self.port = port;
        self.callbacks={}

    def doGet(self, request, response):
        """This method is called when a GET request is sent to the web server. It MUST be overwri
        pass
    def doPost(self, request, response):
        """This method is called when a POST request is sent to the web server. It MUST be overwr
        pass

    def __handlerAttributes(self, rawAttributes):
        """This method is called to handler the attributes of the HTTP request"""
        rawAttributes=rawAttributes[:len(rawAttributes)-2]
        map={}
        for i in rawAttributes:
            map[i[:i.find(':')]] = i[i.find(':')+2:len(i)-1]
        return map
    
```

Como acessar os recursos de dispositivos S60 em WRT ou Flash Lite usando PyS60

```
def __handlerRequest(self, connection, rawRequest):
    """This method is called to handler the request. It split the tokens, call the '__handlerA
    self._rawRequest = rawRequest
    tokenizedLine=self._rawRequest.split('\n')
    requestLine=tokenizedLine[0]
    attributes = self.__handlerAttributes(tokenizedLine[1:])
    tokenizedLine = requestLine.split()
    attributes["request-method"]=tokenizedLine[0]
    attributes["requisition"]=tokenizedLine[1]
    attributes["http-version"]=tokenizedLine[2]
    request_object = attributes["requisition"]
    if request_object.startswith('/'):
        request_object=request_object[1:]
    objects=request_object.split('?')
    attributes["object-requested"]=objects[0]
    map={}
    if len(objects)>1:
        objects=objects[1].split('&')
        for i in objects:
            iObject = i.split('=')
            map[iObject[0]]=iObject[1]
    attributes["parameters"]=map
    request = Request(self._rawRequest, attributes)
    response = Response()
    if attributes["request-method"]=="GET":
        self.doGet(request, response)
    elif attributes["request-method"]=="POST":
        self.doPost(resquest, response)
    rsp = response.getResponse()
    connection.send(rsp)
```

```
class MyServer(HTTPServer):
    """This is a subclass of HTTPServer and here the implementation of the GET HTTP method is pro

    def __init__(self,host, port):
        HTTPServer.__init__(self, host, port)

    def doGet(self, request, response):
        """Implementation of the GET method"""
        functionName=request.getProperty("object-requested")
        attributes=request.getProperty("parameters")
        response.println(str(self.callbacks[functionName](attributes))) # Call the service request
```

Este é um exemplo de serviço. Nesta função nós recuperamos a posição do dispositivo utilizando GPS.

```
def get_position(attributes):
    positioning.select_module(positioning.default_module())
    pos=positioning.position()['position']
    ret='latitude=' + str(pos['latitude']) + '&longitudo=' + str(pos['longitudo'])
    return ret
```

O trecho de código abaixo mostra como iniciar o servidor e adicionar os serviços.

```
server = MyServer('127.0.0.1',5004)
server.addCallBack("get_position", get_position)
server.startServer()
```

Passo-a-passo para instalar e usar o *ServerPython*

- Faça o download e instale `spython.sis` para seu dispositivo.
- Inicie a aplicação `spython`.
- Inicie a aplicação que faz uso dos recursos disponibilizados (Flash Lite or WRT).

O desenvolvedor pode facilmente incluir novos métodos ao *ServerPython*, sendo necessário apenas um pouco de conhecimento da linguagem Python. Abaixo é apresentado as etapas necessárias para esta tarefa.

Implemente o método `method_x` na classe *Service*. É imposta apenas uma condição a estrutura do novo método. O método DEVE receber uma estrutura de mapa representando os parâmetros da URL.

Execute o método `server.addCallCack('url_to_x', method_x)`, onde *server* é um atributo do tipo *Myserver* incluso na classe *Services*.

Download

Abaixo, é possível efetuar o download do código fonte e o arquivo SIS do *ServerPython*. Clique [aqui](#) para pegar o arquivo zip com o código fonte e o arquivo SIS.

Um exemplo de uma aplicação em Flash Lite

Como mostrado anteriormente, uma aplicação cliente (Flash Lite) terá acesso aos recursos internos de um dispositivo S60 através de requisições HTTP. Em ActionScript 2.0 (utilizado em nas versões do Flash Lite 2.x e 3.x) uma requisição HTTP pode ser executada usando a classe *LoadVars*. A utilização desta classe é apresentada abaixo:

```
var lv:LoadVars = new LoadVars();
lv.onLoad = function(success:Boolean) {
//Ações quando o dado é carregado
}
lv.sendAndLoad("http://example.com", lv, "GET");
```

A URL <http://example.com> url tem de ser alterada para http://127.0.0.1:5004/get_position se voce quiser recuperar a posição utilizando do dispositivo via GPS usando o *ServerPython*.

Um exemplo completo de recuperar e acessar os recursos de mapas do Google map pode ser encontrado aqui: [File:Gps-serverpython.zip](#). Mais detalhes sobre este exemplo pode ser visto no seguinte artigo: [Acessando os recursos nativos do sistema usando PyS60 em dispositivos S60 3rd Edition](#).

Criado por

- Ivo Calado (ivocalado [at] embedded [dot] ufcg [dot] edu [dot] br)
- Marcos Fábio Pereira (marcos [at] embedded [dot] ufcg [dot] edu [dot] bt)

Como acessar os recursos de dispositivos S60 em WRT ou Flash Lite usando PyS60

Para ter mais detalhes sobre desenvolvimento de aplicações para dispositivos móveis, veja (em inglês) [Web Effort](#), [Flash Lite Effort](#), [WRT Effort](#) e [Python S60 Effort](#)