



Sistemas Distribuídos

PyRO (Python *Remote Object*)


Prof.ª Ana Cristina Barreiras Kochem Vendramin
Universidade Tecnológica Federal do Paraná (UTFPR), Câmpus Curitiba
Engenharia de Computação, Sistemas de Informação
Programa de Pós-Graduação em Computação Aplicada (PPCA)
Departamento Acadêmico de Informática (DAINF)




Licença [Creative Commons Atribuição: NãoComercial-SemDerivações 4.0 Internacional](#)



MINISTÉRIO DA EDUCAÇÃO




1




Introdução ao Pyro

- Pyro de Irmen de Jong
- Permite a comunicação entre objetos Pyro
- Biblioteca Python
- Executa em diferentes plataformas e versões Python
- Utiliza a mesma sintaxe para invocações locais e remotas



MINISTÉRIO DA EDUCAÇÃO



2

Componentes

- **Objeto Pyro**

- Objeto Python que precisa estar registrado com o Pyro para que seus métodos possam ser chamados remotamente.
- Pyro utiliza URI (*Uniform Resource Identifier*) para identificar cada objeto.
- URI é uma string:
 - "PYRO:" + nome do objeto + "@" + nome do host + número da porta

3

Componentes

- **Proxy**

- Para acessar um objeto Pyro é preciso obter o proxy para ele
- Um proxy intercepta as chamadas de métodos a um objeto como se fosse o objeto atual
- Transfere a chamada para o computador que contém o objeto real

4

Componentes

- **Pyro Daemon (Servidor)**

- Escuta as chamadas de métodos remotos, encaminha as chamadas para o objeto correspondente e retorna o resultado
- Todo objeto que se deseja publicar como um objeto Pyro precisa estar registrado com um daemon
 - Esse registro retorna a URI do objeto Pyro → chave para acessar o objeto Pyro
`uri_objetoPyro = daemon.register(objeto)`



MINISTÉRIO DA
EDUCAÇÃO



5

Componentes

- **Servidor de nomes**

- Mapeia o nome lógico de um objeto remoto para a sua localização (URI)
- É um objeto Pyro de uso opcional que pode ser acessado através de um proxy Pyro
- O objeto exposto é:
`Pyro5.nameserver.NameServer`



MINISTÉRIO DA
EDUCAÇÃO



6

Servidor de Nomes

- Execução via linha de comando:

python -m Pyro5.nameserver [options] (ou pyro5-ns [options])

Saída:

NS running on localhost:9090

URI = PYRO:Pyro.NameServer@localhost:9090

pyro5-ns nome_host

Saída:

NS running on nome_host:9090

URI = PYRO:Pyro.NameServer@nome_host:9090

- Execução no código: **Pyro5.nameserver.start_ns()**



MINISTÉRIO DA
EDUCAÇÃO



7

Servidor de Nomes

- Para se obter um proxy para o servidor de nomes e executar seus métodos, utiliza-se a função

Pyro5.core.locate_ns() ou **Pyro5.api.locate_ns()**

- Realiza uma pesquisa em *broadcast* para localizar o servidor de nomes (a menos que seja especificado um *host* como parâmetro).

`locate_ns([host=None, port=None, broadcast=True])`



MINISTÉRIO DA
EDUCAÇÃO



8

Servidor de Nomes

- Interação com servidor de nomes através da ferramenta nsc (*name server control*).
`python -m Pyro5.nsc método` (ou `pyro5-nsc método`)

Exemplo: `pyro5-nsc list`

Saída:

```
-----START LIST
Pyro.NameServer --> PYRO:Pyro.NameServer@localhost:9090
metadata: {'class':Pyro5.nameserver.NameServer'}
-----END LIST
```



MINISTÉRIO DA
EDUCAÇÃO



9

Servidor de Nomes

- Como nsc sabe o endereço do serviço de nomes?
 - Emite um mensagem em **broadcast** na rede para saber se tem um servidor de nomes disponível.
 - Se existir, o servidor de nomes irá responder.
 - Se o servidor estiver vinculado com *localhost*, ele não responderá às mensagens *broadcast* e, portanto, não ficará visível na rede.
 - É necessário especificar um nome de host ou IP caso queira tornar o servidor de nomes acessível de outras máquinas.



MINISTÉRIO DA
EDUCAÇÃO



10

Servidor de Nomes

- Principais métodos:
 - **count()**: retorna o número de registros
 - **list()**: recupera os itens registrados como um dicionário nome-URI
 - **lookup(nome_objeto)**: procura a URI pelo nome do objeto fornecido
 - **register(nome, URI)**: registra o nome com a URI

Para mais detalhes sobre outros métodos e parâmetros:

<https://pyro5.readthedocs.io/en/latest/nameserver.html#name-server>



MINISTÉRIO DA
EDUCAÇÃO



11

Servidor – Permitindo acesso remoto aos objetos Pyro

- Decorador **@expose**
 - Expõe classes, métodos e propriedades → permite o acesso remoto
 - Expor uma classe expõe automaticamente todos os seus métodos não privados e propriedades

<https://pyro5.readthedocs.io/en/latest/servercode.html>



MINISTÉRIO DA
EDUCAÇÃO



12

Servidor – Publicando objetos Pyro

- Cria uma classe que se deseja publicar como objeto Pyro
- Cria um **Pyro daemon** para escutar os pedidos dos clientes e processá-los.
- Registra a classe com o daemon.
- Chama o método **requestloop()** do daemon para ficar esperando os pedidos

13

Servidor – Publicando um objeto Pyro

```
@expose
class MeuPyro(object):
    # ... métodos...
    daemon = Daemon()
    uri_objetoPyro = daemon.register(MeuPyro)
    daemon.requestLoop()
```

- Quando um cliente conectar, o Pyro criará uma instância da classe e utilizará o objeto para manipular as chamadas de métodos remotos durante uma sessão de proxy desse cliente
- O objeto será removido quando o cliente desconectar
- Quando um novo cliente conectar, uma outra instância será criada para sua sessão

14

Servidor – Registrando um objeto Pyro no Servidor de Nomes

- A URI retornada do método de registro do daemon pode ser armazenada no servidor de nomes Pyro

```
uri_objetoPyro = daemon.register(objeto)
servidor_nomes = locate_ns()
servidor_nomes.register("nome_objeto", uri_objetoPyro)
```

- Registrar um objeto é associar sua URI com um nome de modo que os clientes podem usar esse nome para consultar a URI no serviço de nomes

<https://pyro5.readthedocs.io/en/latest/nameserver.html#nameserver-registering>



MINISTÉRIO DA
EDUCAÇÃO



15

Cliente - Localizando objetos Pyro

- Utilizando diretamente o nome do objeto e sua localização
URI → PYRO:nome_objeto@nome_host:numero_porta
- Pesquisando pelo nome em um **Servidor de Nomes**

```
servidor_nomes = Pyro5.api.locate_ns()
uri_objetoPyro = servidor_nomes.lookup("nome_objeto")
```
- Ou simplesmente utilizando o tipo de protocolo PYRONAME:

```
uri_objetoPyro = "PYRONAME:nome_objeto"
```

→ Pyro irá localizar o servidor de nomes e consultará a URI nele



MINISTÉRIO DA
EDUCAÇÃO



16

Cliente – Chamando métodos

- Com a URI do objeto Pyro, cria-se um Proxy para ele
 - Lembrando que as chamadas de métodos remotos em objetos Pyro passam por um Proxy
`nome_objeto = Pyro5.api.Proxy(uri_objetoPyro)`
`nome_objeto.metodo_objeto`
- Por padrão, o Pyro espera infinitamente por um retorno de uma chamada de método a menos que seja configurado um temporizador
<https://pyro5.readthedocs.io/en/latest/clientcode.html#timeouts>
ou que se faça uma chamada oneway
<https://pyro5.readthedocs.io/en/latest/clientcode.html#oneway-calls>



MINISTÉRIO DA
EDUCAÇÃO



17

Chamadas One-way

- Métodos no servidor marcados com o decorador **@oneway**
 - Quando o proxy do cliente se conecta ao servidor, ele sabe automaticamente quais métodos são one-way. Nada precisa ser feito no cliente.
 - Qualquer chamada que o cliente fizer no proxy para métodos no servidor marcados com **@oneway** acontecerá como chamadas unidirecionais.
 - Pyro não esperará uma resposta do objeto remoto. Valor de retorno é sempre None, o qual é retornado imediatamente após submeter a chamada ao método remoto one-way
 - Servidor processará a chamada enquanto o cliente continua a execução
 - Cliente não saberá se a chamada do método foi bem sucedida



MINISTÉRIO DA
EDUCAÇÃO



18

Pyro Callbacks – chamadas “reversas”

- Os métodos no cliente podem ser marcados com o decorador especial **@callback** (<https://pyro5.readthedocs.io/en/latest/clientcode.html#pyro-call-backs>)
 - Cliente inicia um Pyro daemon e registra o **objeto Pyro callback** nele, como se estivesse escrevendo um programa servidor
 - O método callback também precisa ter o decorador **@expose**
 - Cliente terá que executar o *request loop* do daemon em uma *thread*
 - Caso contrário, o cliente não seria capaz de processar as chamadas *callback* porque ele estaria esperando resultado de chamada ao servidor (a não ser que fossem chamadas de métodos oneway).

19

Referências

- Irmem de Jong. “Pyro - Python Remote Objects - 5.14”.
<https://pyro5.readthedocs.io/en/latest/>

20