SPLN - TP2

Serviço remoto de filtro-linha-comando genérico

Tiago Silva a93277 and Gonçalo Afonso a93178

Universidade do Minho, Braga, Portugal

1 Introdução

O presente relatório segue o trabalho realizado para o segundo trabalho prático da unidade curricular de Scripting e Processamento de Linguagens Naturais. Neste segundo trabalho, foi proposto pelos docentes a concretização de uma solução para um dos temas num conjunto por eles eleito, tendo o nosso grupo, dentro destes, escolhido a realização do tema 4: Serviço remoto para um serviço filtro-linha-de-comando genérico.

Este problema revolve na criação de um serviço remoto, que disponibilize diferentes ferramentas para realizar pedidos de uso destas. Um dos principais requisitos neste trabalho é a necessidade de criar um serviço cuja acessibilidade seja garantida mesmo para pessoas não versadas com o uso 'puro' das ferramentas (ex.: em linha de comando), assim como a possibilidade de facilmente criar serviços com conjuntos de ferramentas diferentes de formar mais fácil automática.

O serviço que criamos é então um programa em python, instalável, que recebendo um ficheiro de configuração de um servidor, modifica um servidor modelo feito em Node.js com uso de *express-generator*[2], criando as suas rotas, instalando dependências e com opção para o iniciar automaticamente, assim como expor para um url público. A este programa demos o nome de *Tool Server Generator* [1]

De igual modo, para a criação do servidor, temos como dependência a instalação do *Node.js* numa versão igual ou superior a 10.5, para suportar o módulo de *worker_threads*

2 Solução

A solução desenvolvida envolveu dois pontos principais, o programa em Python para a criação automática do servidor em Node.js e o desenvolvimento do próprio modelo do servidor.

2.1 Programa

Começando pelo programa em python, este é instalável realizando:

```
pip install tool_server_generator
```

, sendo que o seu uso segue a chamada do comando com a indicação obrigatório de um ficheiro de texto ou json, que deve ter a configuração do servidor:

```
ts <ficheiro de configuração>
```

Outros modos de uso envolvem a utilização da opção *'-s'* que indica a intenção de iniciar automaticamente o servidor assim que ele estiver pronto e *'-ng'* que indica a intenção de, se a flag de iniciar o servidor estiver acionado, expor o servidor utilizando *ngrok*[3]. Esta última opção, assume no entanto que o *ngrok* já se encontra disponível no sistema, assim como a autenticação dele realizada.

2.2 Configuração do servidor

O ficheiro de configuração é o que irá definir desde a porta e rota do servidor, como as ferramentas que ele possui, até à palete de cores das páginas web.

Este ficheiro pode ser interpretado de dois modos, em formato de texto, ou em formato JSON, sendo que a identificação de um ficheiro JSON é feita automaticamente (sem uso de opções), através da extensão do ficheiro.

No caso do ficheiro JSON, naturalmente ele terá de seguir os campos específicos que explicaremos em seguida. Quanto ao ficheiro de texto, este seguirá uma sintaxe por nós construída com a utilização do módulo *lark* e do *Interpreter* deste, tendo portanto de seguir uma estrutura mais rigorosa no que toca à ordem dos campos.

Campos de Configuração
Servidor

- *Nome* : nome do servidor.
- *Diretoria*: diretoria onde o servidor vai ser criado.
- *Porta*: porta do servidor.
- *Trabalhadores*: número de trabalhadores disponíveis para processar pedidos. Opcional, 1 por defeito.

Ferramentas

- *Família* : família da ferramenta. Permite agrupar ferramentas da mesma família, por exemplo que tenham o mesmo comando base mas opções diferentes. Determina a rota da ferramenta.
- *Título* : nome da ferramenta. Determina a rota da ferramenta.
- *Descrição* : descrição da ferramenta.
- *Comando*: comando que correrá a ferramenta, com os inputs variáveis (pelo utilizador) delimitados utilizando a palavra exclusiva 'INPUT<number>'.
- *Inputs*: para cada ferramenta, os seus inputs variáveis têm de ser explicitamente configurados, caso contrário serão tratados como texto simples no comando. Opcional.

Para cada Input de uma ferramenta

- *Nome*: nome do input. Opcional.
- *Descrição*: descrição do input. Opcional.
- *Tipo*: tipo do input (STR|NUM|FILE).

Visuais (Opcional)

- *Favicon:* icon do servidor.
- *Colors:* paleta de 6 cores do servidor. 2 cores para o *background* (primária e secundária), 2 cores para o texto (primária e secundária), 1 cor para as *labels*, e uma para as bordas.

Exemplo de ficheiro de configuração em texto

```
* Servidor
- Nome: Servidor
- Diretoria: "teste"
- Porta: 15213
* Ferramentas
--
- Família:Grep
- Título: "Grep"
- Descrição: "grep  searches  for  PATTERNS in each FILE.
```

```
PATTERNS is one or more patterns separated by newline characters,and grep prints each line
that matches a pattern.
Typically PATTERNS should be quoted when grep is used in a shell command."
-Comando: cat INPUT1 | grep INPUT2
-Inputs:
    - INPUT1:
        - Nome : file
        - Descrição : "Ficheiro onde se vai aplicar a procura"
        - Tipo : FILE
    - INPUT2:
        - Nome : pattern
        - Descrição : "Padrao de procura"
        - Tipo : STR
* Visuais
- Favicon : "image.png"
- PrimaryBgColor : #000
- SecondaryBgColor : #fff
- PrimaryTextColor : #fff
- SecondaryTextColor : #3f51b5
- LabelColor : #3f51b5
- BorderColor : #9e9e9e
```

É importante realçar novamente, que no formato de ficheiro de texto simples, a ordem dos campos é importante, assim como os seus delimitadores (ex.: '* Servidor', '–' antes de cada ferramenta, etc.)

Exemplo de ficheiro de configuração JSON

```
{
    "nome" : "Servidor",
    "diretoria" : "teste",
    "porta" : 20202,
    "ferramentas" : {
        "Grep" : {
            "Grep" : {
                "descricao" : "Descrição do comando",
                "comando" : "cat INPUT1 | grep INPUT2 & sleep 10",
                "inputs" : [
                    {
                        "id": "INPUT1",
                        "opcoes" : {
                            "nome" : "file",
                            "descricao" : "Ficheiro onde se vai aplicar a procura",
                            "tipo" : "FILE"
                        }
                    },
                    {
                        "id": "INPUT2",
                        "opcoes" : {
                            "nome" : "pattern",
                            "tipo" : "STR",
                            "descricao" : "Padrao de procura"
                        }
```

```
                    }
                ]
            }
        }
    }
}
```

2.3 Servidor modelo

O servidor modelo serve como base para ser modificada de acordo com as configurações de servidor dadas. Deste modo, não é necessário o processo de instalação e chamada do *express-generator* sempre que se quer criar um novo servidor. Este servidor é então uma forma reduzida de um servidor criado com este módulo, e que usa views em formato *pug*.

O servidor modelo depende apenas dos módulos *express* ,*http-errors* e *adm-zip*, sendo incluído o *multer* no caso de haverem inputs do tipo ficheiro, ou seja, que precisam ser enviados para o servidor.

Além dos ficheiros base que mantemos resultantes do *express-generator*, incluímos ainda a view principal modificada, e scripts de javascript para: thread principal que gere pedidos (request_listerner.js); script de um worker, i.e. quem irá realizar a tarefa do pedido (process_command.js); scripts de lado do cliente para estilização e pedido de resultado rápido (normalmente stdout).

O servidor tem como página principal o local onde se pode observar as requests feitas, onde se observa o estado destas e, no caso das requests completas, pode-se pedir para dar download dos resultados, ou apagá-las para libertar espaço.

2.4 Criação do servidor

A partir do servidor modelo, realizamos as modificações necessárias para atender à configuração dada pelo utilizador, i.e. rotas para as ferramentas, modificar portas, número de trabalhadores, visuais, etc.

Operações realizadas

Começando pelas configurações base do servidor, no ficheiro *www* é modificada a porta de acesso.

De igual modo, é modificado o layout para condizer com o nome de servidor dado.

Para cada uma das ferramentas são criadas rotas de get e post, tendo para cada uma, em conta que se tiverem inputs que envolvam upload de ficheiros, precisamos de criar uma função de middleware usando *multer* para tratar destes.

Por último, são aplicadas as configurações visuais.

Iniciação do servidor

A iniciação do servidor é realizada através do próprio programa em python, chamando *npm start* dentro da diretoria do servidor, após as instalações das dependências estarem terminadas.

Expor servidor

A exposição do servidor é realizada com uso do *ngrok*, também no programa em python, através da biblioteca *pyngrok*. A porta exposta é a mesma que a definida no ficheiro de configuração.

Como mencionado anteriormente, esta opção presume que o *ngrok* está funcional no sistema.

3 Conclusão

Em suma, fizemos uma aplicação em Python que apenas com uma configuração simples (ficheiro de texto simples ou em json) é capaz de criar um serviço remoto para uma ou várias ferramentas. Para tal, fizemos uso dos conhecimentos obtidos ao longo de todo o perfil de Engenharia de

Linguagens. Utilizamos boa parte do conhecimento obtido em Representação e Processamento de Conhecimento na Web (RPCW) na criação do modelo de servidor inteiramente em nodejs. Exploramos a ferramenta lark, como aprendemos em Engenharia Gramatical (EG), na criação de uma gramática para ler o ficheiro de texto com as configurações do servidor. E como seria de esperar, aplicamos os conhecimentos de scripting obtidos em Scripting no Processamento de Linguagem Natural (SPLN) com a criação da aplicação em si feita em Python, de forma a facilmente automatizar a tarefa de criação de servidor e com a sua disponibilização no Pypi com a ajuda do flit.

O produto final atingiu o resultado esperado e ainda mais, sendo que objetivo era fornecer ferramentas de linha de comando mais simples de forma remota, como filtros unix mas, se o utilizador entender, pode introduzir ferramentas mais complexas, como por exemplo o Whisper, exemplo por nós testado.

Entendemos ainda, que esta aplicação que criamos tem muita margem de progressão naquilo que pode ser adicionado como configurações para facilitar a vida do utilizador. Configurações estas como: autenticação no servidor, limite de usos das ferramentas por cliente do servidor, limite no tamanho dos ficheiros dos inputs, limite no tempo que os ficheiros outputs ficam no servidor, mais opções nas ferramentas (flags alternativas numa mesma ferramenta invés de criar duas semelhantes), maior facilidade para alterar as views do servidor, etc.

Referências

1. Tool Server Generator
2. Express-generator
3. Ngrok