

# JSON-WSP

17 listopada 2017

## 1 Zadania

### 1.1 Wstęp

1. Konfiguracja projektu. Utwórz nowy projekt w środowisku PyCharm. Pobierz archiwum z <https://pypi.python.org/pypi/ladon>. Z folderu scripts przekopiuj plik ladon-3.5-ctl. W konsoli wykonaj polecenie **pip install -user ladon**.

2. W projekcie utwórz plik helloservice.py.

3. Zaimportuj:

```
from ladon.ladonizer import ladonize
```

4. Utwórz klasę HelloService i dodaj w niej funkcję helloWorld, która jako parametr przyjmuje name i zwraca string "Hello [name]".

```
class HelloService(object):  
    def helloWorld(self , name):  
        return "Hello {}".format(name)
```

5. Nad funkcją helloWorld dodaj dekorator:

```
@ladonize(str , rtype=str)
```

6. Otwórz konsolę, przejdź do folderu głównego projektu i wykonaj polecenie `python ladon-3.5-ctl testserve helloservice.py`.

7. Otwórz przeglądarkę i wpisz adres `http://localhost:8888/HelloService/`.

8. Dodaj dokumentację usługi. (Wskazówka: [http://ladonize.org/index.php/Python\\_Example](http://ladonize.org/index.php/Python_Example))

9. Po udokumentowaniu usługi, przerwij działanie serwera i uruchom go ponownie.

10. Przeanalizuj opis usługi (description).

11. Z wykorzystaniem programu Postman wyślij żądanie do usługi (zaimportuj plik hello.postman.json, który znajdziesz w załącznikach do laboratoriów).
12. Utwórz nowy plik helloclient.py.
13. Zaimportuj:

```
from ladon.clients.jsonwsp import JSONWSPClient
```

14. Utwórz klienta (wykorzystaj url opisu usługi) i wywołaj metodę helloWorld z parametrem zawierającym Twoje imię.

```
url = "http://localhost:8888/HelloService/jsonwsp/description"
client = JSONWSPClient(url)
response = client.helloWorld(name="John_Doe")
print(response.response_dict)
```

## 1.2 Typy zagnieżdżone i wyjątki

1. Utwórz nowy plik pokemonservice.py. Zaimportuj:

```
from ladon.ladonizer import ladonize
from ladon.types.ladontype import LadonType
from ladon.exceptions.service import ClientFault
```

2. Dodaj dwa typy (LadonType): Pokemon oraz zagnieżdżony w nim Attack.

```
class Attack(LadonType):
    name = str
    damage = int
```

```
class Pokemon(LadonType):
    name = str
    hp = int
    attack = Attack
```

3. Dodaj metodę rzucającą wyjątek ClientFault w przypadku gdy pokemon ma za dużo punktów życia.

```
def checkPokemon(pokemon):
    if pokemon.hp > 100:
        raise ClientFault('Too_much_HP!')
```

4. Dodaj PokemonService i dodaj w nim metody odpowiednio umożliwiające: dodanie nowego pokemona i zwrócenie listy pokemonów.

```

class PokemonService(object):
    pokemons = []
    @ladonize(Pokemon, rtype=str)
    def addPokemon(self, pokemon):
        # TODO
    @ladonize(rtype=[Pokemon])
    def getPokemons(self):
        # TODO

```

5. Utwórz plik pokemonclient.py i zaimplementuj klienta usługi PokemonService. Dodaj kilka pokemonów a następnie wyświetl ich listę.

### 1.3 Załączniki (zadanie dodatkowe)

1. W pliku pokemonservice.py zaimportuj:

```

from ladon.types.attachment import attachment

```

2. Dodaj typ:

```

class File(LadonType):
    name = str
    data = attachment

```

3. Dodaj metodę:

```

@ladonize(rtype=File)
def getPokeballImg(self):
    file = File()
    file.name = "MyPokeball.png"
    file.data = attachment(open("pokeball.png", 'rb'))
    return file

```

4. W pliku pokemonclient.py dodaj:

```

result = client.getPokeballImg()
imgName = result.response_dict['result']['name']
imgData = result.response_dict['result']['data']
file = open(imgName, 'wb')
file.write(imgData.read())
file.close()

```

5. Pora na wykazanie swoich talentów malarskich. W dowolnym programie do rysowania (np. Paint) narysuj pokeballa i zapisz go jako plik 'pokeball.png', następnie dodaj go do projektu.
6. Wywołaj metodę getPokeballImg w kliencie.