Software Engineering For Data Science (SEDS)

Class: 2nd Year 2nd Cycle
Branch: AIDS

Dr. Belkacem KHALDI| ESI-SBA

Lecture 09:

Web Development for Data Science: Building Restful API with FastAPI

Web Development for Data Science: Building Restful API with FastAPI

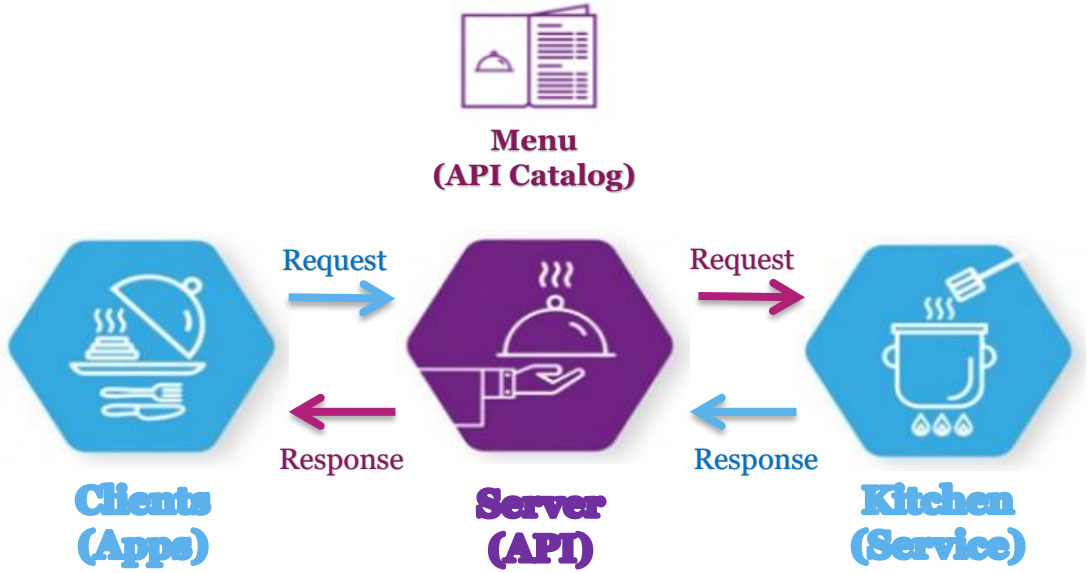
An Introduction to FastAPI

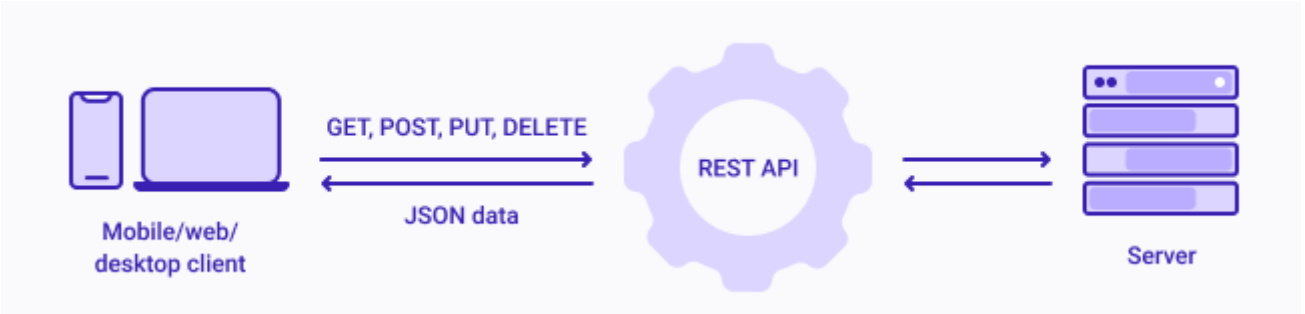
Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Building Restful API with FastAPI

An Introduction to FastAPI

- **What is an API?**
  - Stands for **A**pplication **P**rogramming **I**nterface.
    - Allows communications between application components
    - It is like a **waiter** (**Server**) in a restaurant.
- **What is A REST API?**
  - Stands for **RE**presentational **S**tate **T**ransfer API:
    - A Web API that follows the HTTP method constraints - get, post, put, delete
    - A way of accessing web services in a simple and flexible way using **json** data format without having any processing.





Building Restful API with FastAPI

An Introduction to FastAPI

What is FastAPI?

- FastAPI is a high-performing web framework for building APIs with **Python 3.7+**.
- Helps to build applications quickly and efficiently.
- Built on top of the **Starlette** web server.
- Includes many features such as:
  - **Automatic data validation**,
  - **Error handling**,
  - **Interactive API docs**.
  - **OpenAPI based**: Fully compatible with OpenAPI and JSON Schema
  - **Robust**: Production-ready code with automatic interactive documentation.

	Django	Flask	FastAPI
Community	Big. 66K GitHub ★	Big. 61K GitHub ★	Big. 50K GitHub ★
Performance	Not the best in terms of performance	Performs better than Django	More efficient
Async support	Yes, with limited latency.	No. Needs Asyncio	Provides native async support
Ease of use	A bit complicated to learn.	Easy to learn and pretty straightforward	The simplest of all three.
Interactive Documentation	Not interactive	No	Yes (OpenAI, Redoc)
Data Verification	No	No	Yes

Building Restful API with FastAPI

An Introduction to FastAPI

❑ Creating your first FastAPI

1 **Packages installation**

```
conda install fastapi uvicorn
```

```
pip install fastapi uvicorn
```

2 **helloWordAPI.py**

```python
from fastapi import FastAPI

# Instantiating a FastAPI object handling all API routes
app = FastAPI()

# Creating a GET endpoint at the root path
@app.get("/")
async def hello_world():
        return {"hello": "world"}
# Async method returning a JSON response autmatically
handled by FastAPI
```

- FastAPI exposes one decorator (@) per HTTP method to add new routes to the application.
- The one that is shown here adds a **GET** endpoint with the root path as the first argument

3 **Running a localhost server**

```
$uvicorn <FILE>:<FastAPI_Object>
```

```
$uvicorn helloWordAPI:app
```

```
←[32mINFO←[om:     Started server process [←[36m7824←[om]
←[32mINFO←[om:     Waiting for application startup.
←[32mINFO←[om:     Application startup complete.
←[32mINFO←[om:     Uvicorn running on ←[1mhttp://127.0.0.1:8000←[om (Press CTRL+C to quit)
←[32mINFO←[om:     127.0.0.1:57678 - "←[1mGET / HTTP/1.1←[om" ←[32m200 OK←[om
←[32mINFO←[om:     127.0.0.1:57678 - "←[1mGET /favicon.ico HTTP/1.1←[om" ←[31m404 Not Found←[om
```

Building Restful API with FastAPI

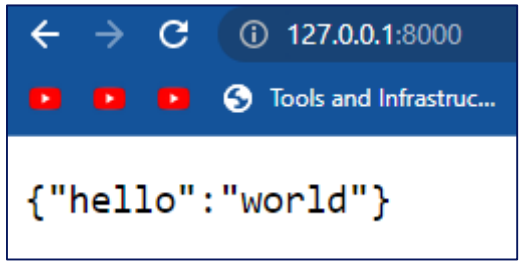
An Introduction to FastAPI

- ❑ **Creating your first FastAPI**

4 **Testing the API**

Browser

http://127.0.0.1:8000



Alternatively, we may try our endpoint with **HTTPie**

HTTP request made to localhost server

```
(base) C:\Users\user>http http://localhost:8000
HTTP/1.1 200 OK
content-length: 17
content-type: application/json
date: Tue, 10 Jan 2023 12:19:43 GMT
server: uvicorn

{
    "hello": "world"
}
```

"/" was requested so the results of the "/" will be sent back to browser. In this case is a **JSON RESPONSE MESSAGE**

Menu (FastAPI Catalog)

Local uvicorn Server

FastAPI is running on port 8000 serving the root path: "/"

```
@app.get("/")
async def hello_world():
        return {"hello": "world"}
```

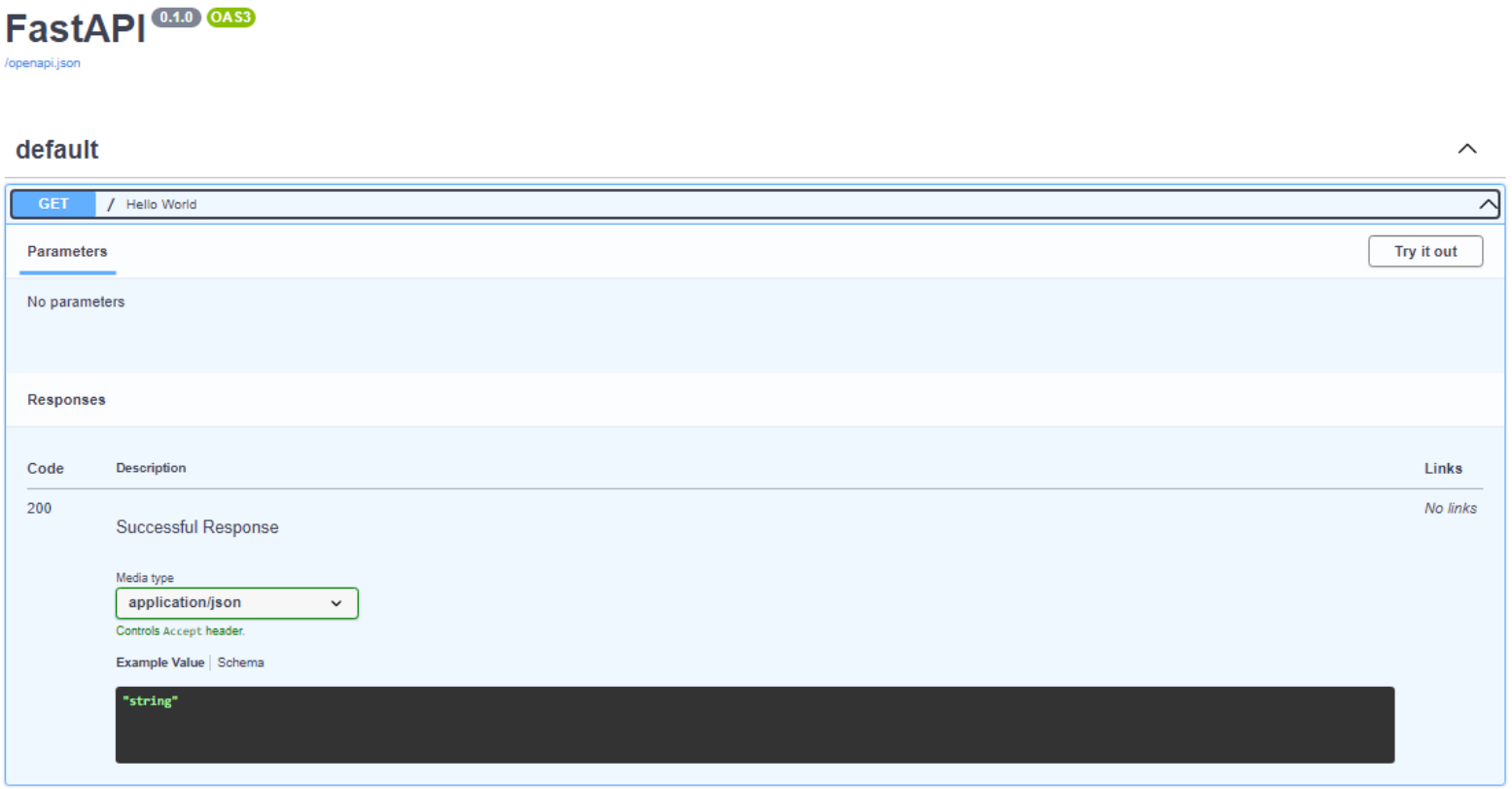
Building Restful API with FastAPI

An Introduction to FastAPI

- ❑ **Creating your first FastAPI**

5 **Testing the API:**
Automatic Interactive Documentation

http://localhost:8000/docs



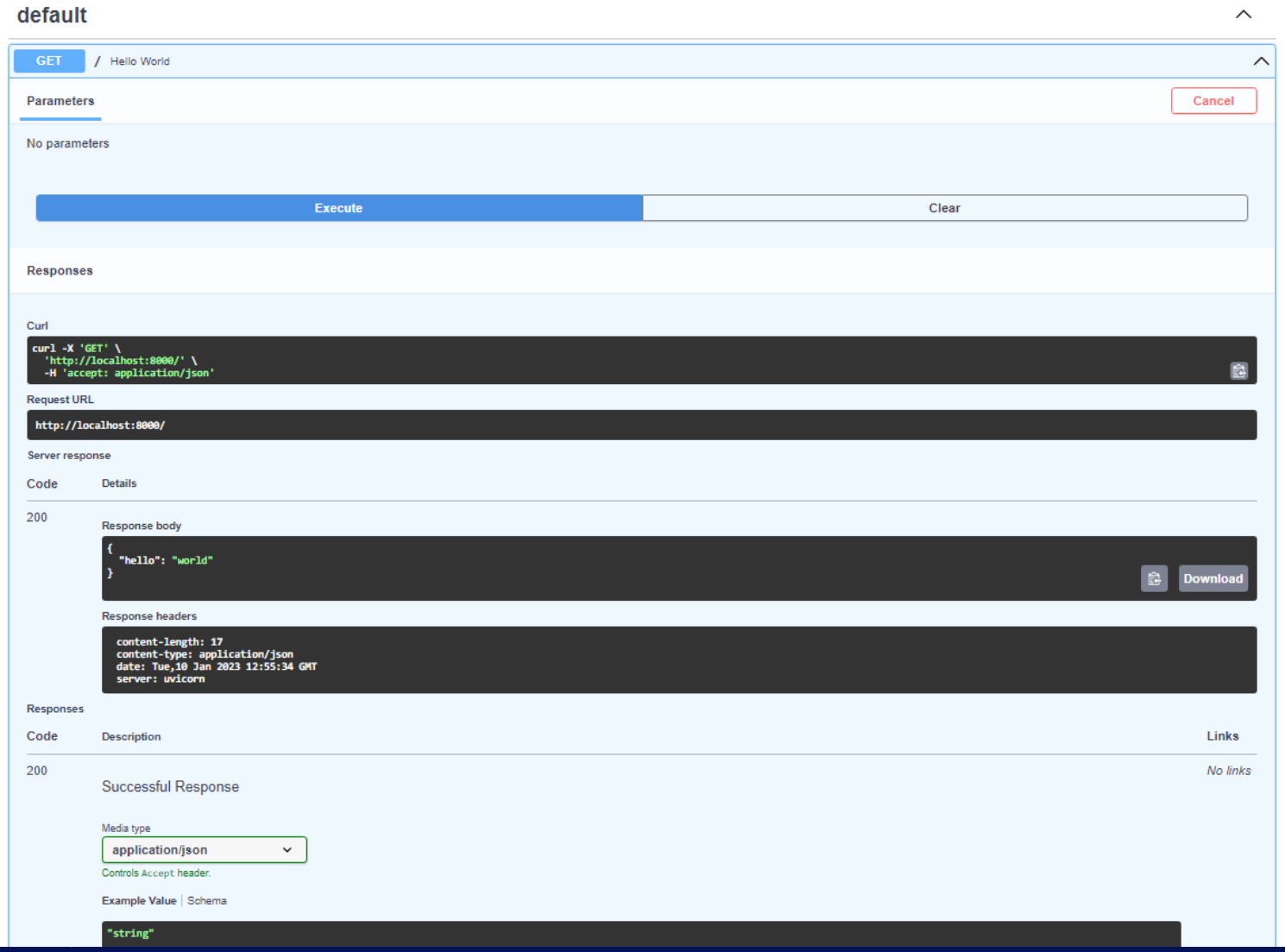
Building Restful API with FastAPI

An Introduction to FastAPI

- ❑ **Creating your first FastAPI**

5 **Testing the API:**
Automatic Interactive Documentation

http://localhost:8000/docs



Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

```python
from fastapi import FastAPI

app = FastAPI()

# API that expects an integer in the last part of its path
@app.get("/users/{id}")
async def get_user(id: int):
    return {"id": id}
```

http http://localhost:8000/users

```
HTTP/1.1 404 Not Found
content-length: 22
content-type: application/json
date: Tue, 10 Jan 2023 13:48:37 GMT
server: uvicorn

{
    "detail": "Not Found"
}
```

http http://localhost:8000/users/100

```
HTTP/1.1 200 OK
content-length: 10
content-type: application/json
date: Tue, 10 Jan 2023 13:51:17 GMT
server: uvicorn

{
    "id": 100
}
```

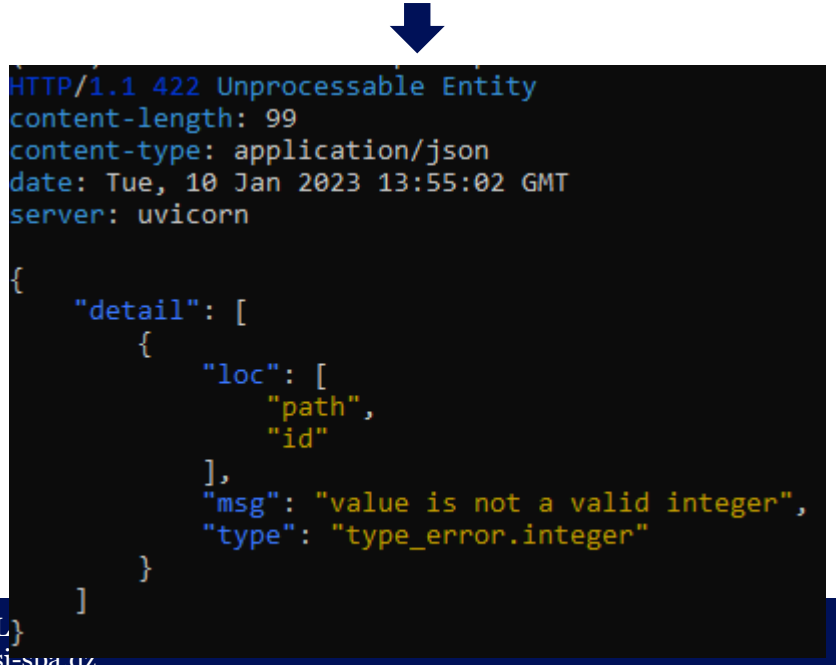
Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

```python
from fastapi import FastAPI

app = FastAPI()

# API that expects an integer in the last part of its path
@app.get("/users/{id}")
async def get_user(id: int):
    return {"id": id}
```

```
http http://localhost:8000/users/khaldi
```



So, what happens if we pass a value that's not a valid integer

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

```python
from fastapi import FastAPI
from enum import Enum

app = FastAPI()

class UserType(str, Enum):
    STANDARD = "standard"
    ADMIN = "admin"

@app.get("/users/{type}/{id}/")
async def get_user(type: UserType, id: int):
        return {"type": type, "id": id}
```

So, what happens if we want Limiting allowed values.

http http://localhost:8000/users/admin/123

```
HTTP/1.1 200 OK
content-length: 25
content-type: application/json
date: Tue, 10 Jan 2023 14:07:20 GMT
server: uvicorn

{
    "id": 123,
    "type": "admin"
}
```

http http://localhost:8000/users/hello/123

```
HTTP/1.1 422 Unprocessable Entity
content-length: 184
content-type: application/json
date: Tue, 10 Jan 2023 14:07:00 GMT
server: uvicorn

{
    "detail": [
        {
            "ctx": {
                "enum_values": [
                    "standard",
                    "admin"
                ]
            },
            "loc": [
                "path",
                "type"
            ],
            "msg": "value is not a valid enumeration member; permitted: 'standard', 'admin'",
            "type": "type_error.enum"
        }
    ]
}
```


Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

```python
from fastapi import FastAPI, Path

app = FastAPI()


@app.get("/users/{id}")
async def get_user(id: int = Path(..., ge=1)):
    return {"id": id}
```

Condition Ops	
gt	Greater than
ge	Greater than or equal to
lt	Less than
le	Less than or equal to

So, what happens if we want **Advanced Validation on Integers values**.

http http://localhost:8000/users/100

```
HTTP/1.1 200 OK
content-length: 10
content-type: application/json
date: Tue, 10 Jan 2023 14:28:19 GMT
server: uvicorn

{
    "id": 100
}
```

http http://localhost:8000/users/0

```
HTTP/1.1 422 Unprocessable Entity
content-length: 149
content-type: application/json
date: Tue, 10 Jan 2023 14:28:27 GMT
server: uvicorn

{
    "detail": [
        {
            "ctx": {
                "limit_value": 1
            },
            "loc": [
                "path",
                "id"
            ],
            "msg": "ensure this value is greater than or equal to 1",
            "type": "value_error.number.not_ge"
        }
    ]
}
```


Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Building Restful API with FastAPI

An Introduction to FastAPI

❑ Handling request parameters

- ❑ FastAPI allows providing REST APIs in a structured way in which to interact with data
- ❑ It allows end user to send some information to tailor the response they need, such as:
  - **Path parameters,**
  - **Query parameters,**
  - **Body payloads,**
  - **Headers**

```python
from fastapi import FastAPI, Path

app = FastAPI()

@app.get("/license-plates/{license}")
async def get_license_plate(license: str = Path(...,
                                regex=r"^\d{5}-\d{3}-\d{2}$")):
    return {"license": license}
```

So, what happens if we want **Advanced Validation on String values using regex.**

http http://localhost:8000/license-plates/dd-120-AAC

http http://localhost:8000/license-plates/00154-120-34

```
HTTP/1.1 200 OK
content-length: 26
content-type: application/json
date: Tue, 10 Jan 2023 14:55:12 GMT
server: uvicorn

{
    "license": "00154-120-34"
}
```

```
content-length: 176
content-type: application/json
date: Tue, 10 Jan 2023 14:54:51 GMT
server: uvicorn

{
    "detail": [
        {
            "ctx": {
                "pattern": "^\\d{5}-\\d{3}-\\d{2}$"
            },
            "loc": [
                "path",
                "license"
            ],
            "msg": "string does not match regex \"^\\d{5}-\\d{3}-\\d{2}$\"",
            "type": "value_error.str.regex"
        }
    ]
}
```


Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Building Restful API with FastAPI

An Introduction to FastAPI

❑ Handling request parameters

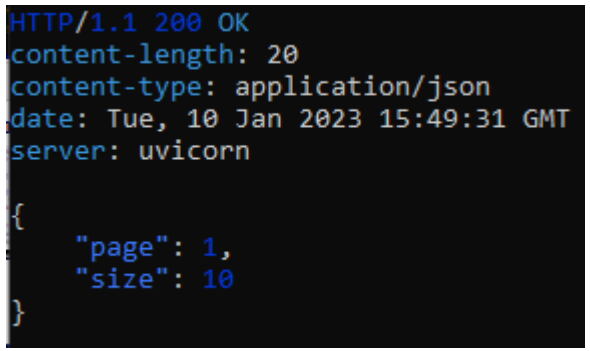
- ❑ FastAPI allows providing REST APIs in a structured way in which to interact with data
- ❑ It allows end user to send some information to tailor the response they need, such as:
  - **Path parameters,**
  - **Query parameters,**
  - **Body payloads,**
  - **Headers**

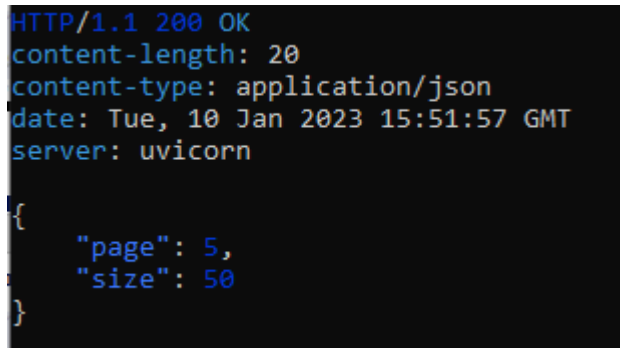
Query parameters ➔ A common way to add some dynamic parameters to a URL. Usually found at the end of the URL in the following form: ?param1=foo&param2=bar

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/users")
async def get_user(page: int = 1, size: int = 10):
          return {"page": page, "size": size}
```

http "http://localhost:8000/users?page=5&size=50"

http http://localhost:8000/users





Building Restful API with FastAPI

An Introduction to FastAPI

❑ Handling request parameters

- ❑ FastAPI allows providing REST APIs in a structured way in which to interact with data
- ❑ It allows end user to send some information to tailor the response they need, such as:
  - **Path parameters,**
  - **Query parameters,**
  - **Body payloads,**
  - **Headers**

Query parameters ➔ A common way to add some dynamic parameters to a URL. Usually found at the end of the URL in the following form: ?param1=foo&param2=bar

```python
from fastapi import FastAPI, Query

app = FastAPI()

@app.get("/users")
async def get_user(page: int = Query(1,  gt = 0),
                   size: int = Query(10, le = 100)):
    return {"page": page, "size": size}
```

Advanced validations through the **Query** function

http "http://localhost:8000/users?page=3&size=30"

```
HTTP/1.1 200 OK
content-length: 20
content-type: application/json
date: Tue, 10 Jan 2023 16:04:10 GMT
server: uvicorn

{
    "page": 3,
    "size": 30
}
```

http "http://localhost:8000/users?page=5&size=150"

```
HTTP/1.1 422 Unprocessable Entity
content-length: 153
content-type: application/json
date: Tue, 10 Jan 2023 16:01:47 GMT
server: uvicorn

{
    "detail": [
        {
            "ctx": {
                "limit_value": 100
            },
            "loc": [
                "query",
                "size"
            ],
            "msg": "ensure this value is less than or equal to 100",
            "type": "value_error.number.not_le"
        }
    ]
}
```


Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Body ➔ Part of the HTTP request that contains **raw data**, **representing documents**, **files**, or **form submissions**. In a **REST API**, it's usually encoded in JSON and used to create structured objects in a database.

```python
from fastapi import FastAPI, Body

app = FastAPI()

@app.post("/users")
async def create_user(name: str = Body(...),
                      age: int = Body(...)):
    return {"name": name, "age": age}
```

```
http -v POST http://localhost:8000/users name="John" age=30
```

```
POST /users HTTP/1.1
Accept: application/json, */*;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 29
Content-Type: application/json
Host: localhost:8000
User-Agent: HTTPie/3.2.1

{
    "age": "30",
    "name": "John"
}
```

```
HTTP/1.1 200 OK
content-length: 24
content-type: application/json
date: Tue, 10 Jan 2023 16:10:03 GMT
server: uvicorn

{
    "age": 30,
    "name": "John"
}
```

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Form data and file uploads Handling

```
pip install python-multipart
conda install python-multipart
```

Form Data Handling

```python
from fastapi import FastAPI, Form

app = FastAPI()

@app.post("/createUser")
async def create_user(name: str = Form(...),
                      age: int =  Form(...)):
    return {"name": name, "age": age}
```

```
http -v --form POST http://localhost:8000/createUser name=Ali age=23
```

--form option to force the data to be form-encoded

```
POST /createUser HTTP/1.1
Accept: */*
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Content-Length: 15
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Host: localhost:8000
User-Agent: HTTPie/3.2.1

name=Ali&age=23
```

```
HTTP/1.1 200 OK
content-length: 23
content-type: application/json
date: Tue, 10 Jan 2023 19:33:02 GMT
server: uvicorn

{
    "age": 23,
    "name": "Ali"
}
```

Pay attention to how the **Content-Type** header and the body data representation have changed in the request

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Form data and file uploads Handling

```
pip install python-multipart
conda install python-multipart
```

```python
from fastapi import FastAPI, File

app = FastAPI()

@app.post("/files")
async def upload_file(file: bytes = File(...)):
    return {"file_size": len(file)}
```

File uploads Handling

```
http --form POST http://localhost:8000/files file@./cat.jpg
```

```
HTTP/1.1 200 OK
content-length: 18
content-type: application/json
date: Tue, 10 Jan 2023 19:53:58 GMT
server: uvicorn

{
    "file_size": 5055
}
```

Drawback:

- The uploaded file is entirely stored in memory.
- Works for small files, but likely an issues for larger files.

Problem Fixing:

- Uses the FastAPI built-on **UploadFile** class.

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

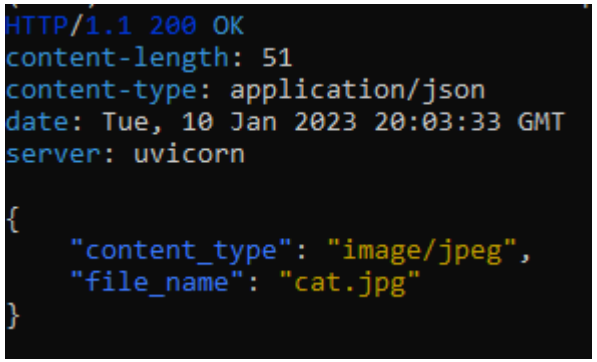
Form data and file uploads Handling

```
pip install python-multipart
conda install python-multipart
```

```python
from fastapi import FastAPI, File, UploadFile

app = FastAPI()

@app.post("/uploadFile")
async def upload_file(file: UploadFile = File(...)):
    return {"file_name": file.filename,
            "content_type": file.content_type}
```

File uploads Handling

```
http --form POST http://localhost:8000/uploadFile file@./cat.jpg
```



Remark:

- This class will store the data in memory up to a certain threshold and, after this, will automatically store it on disk in a temporary location.

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Form data and file uploads Handling

```
pip install python-multipart
conda install python-multipart
```

```python
from typing import List
from fastapi import FastAPI, File, UploadFile

app = FastAPI()

@app.post("/uploadMultipleFiles")
async def upload_multiple_files(files: List[UploadFile]=File(...)):
    return [
            {"file_name": file.filename,
             "content_type": file.content_type}
            for file in files
           ]
```

Multiple Files uploads Handling

```
http --form POST http://localhost:8000/uploadMultipleFiles files@./cat.jpg files@./dog.jpg
```

```
HTTP/1.1 200 OK
content-length: 105
content-type: application/json
date: Tue, 10 Jan 2023 20:21:00 GMT
server: uvicorn

[
    {
        "content_type": "image/jpeg",
        "file_name": "cat.jpg"
    },
    {
        "content_type": "image/jpeg",
        "file_name": "dog.jpg"
    }
]
```

Building Restful API with FastAPI

An Introduction to FastAPI

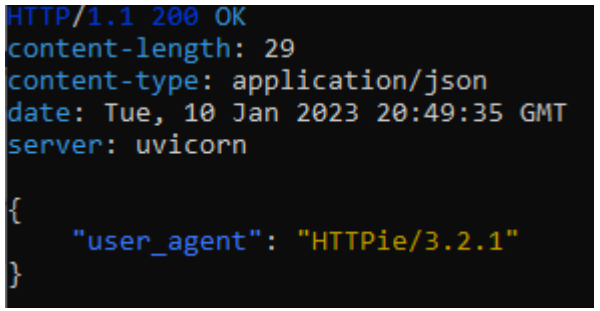
- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Headers ➔ major part of the HTTP request that contains **all sorts of metadata** that can be useful when handling requests. A common usage is to use them for **authentication**, for example, via the famous **cookies**.

```
from fastapi import FastAPI, Header

app = FastAPI()

@app.get("/getHeader")
async def get_header(user_agent: str = Header(...)):
    return {"user_agent": user_agent}
```

User agent

```
http GET http://localhost:8000/getHeader
```



Building Restful API with FastAPI

An Introduction to FastAPI

❑ Handling request parameters

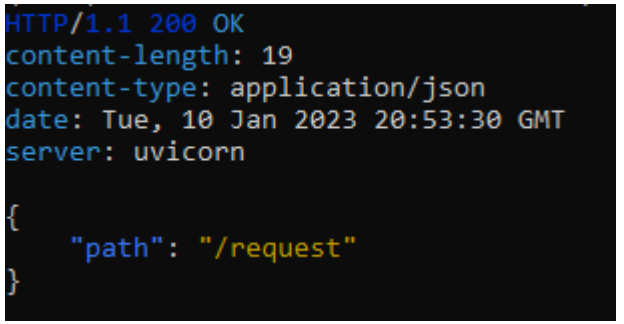
- ❑ FastAPI allows providing REST APIs in a structured way in which to interact with data
- ❑ It allows end user to send some information to tailor the response they need, such as:
  - **Path parameters,**
  - **Query parameters,**
  - **Body payloads,**
  - **Headers**

Headers ➔ major part of the HTTP request that contains **all sorts of metadata** that can be useful when handling requests. A common usage is to use them for **authentication**, for example, via the famous **cookies**.

Using Request Object

```python
from fastapi import FastAPI, Request

app = FastAPI()

@app.get("/request")
async def get_request_object(request: Request):
    return {"path": request.url.path}
```

```
http GET http://localhost:8000/request
```



Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Headers ➔ major part of the HTTP request that contains **all sorts of metadata** that can be useful when handling requests. A common usage is to use them for **authentication**, for example, via the famous **cookies**.

Setting cookies

```python
from fastapi import FastAPI, Response

app = FastAPI()

@app.get("/setCookie")
async def custom_cookie(response: Response):
    response.set_cookie("cookie-name",
                        "cookie-value",
                        max_age=86400)
    return {"hello": "world"}
```

```
http GET http://localhost:8000/setCookie
```

```
HTTP/1.1 200 OK
content-length: 17
content-type: application/json
date: Tue, 10 Jan 2023 21:01:51 GMT
server: uvicorn
set-cookie: cookie-name=cookie-value; Max-Age=86400; Path=/; SameSite=lax

{
    "hello": "world"
}
```

Building Restful API with FastAPI

An Introduction to FastAPI

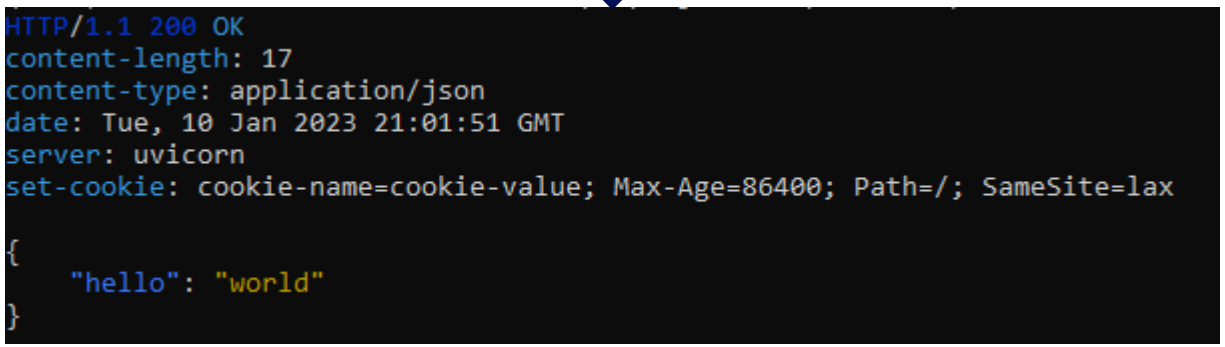
- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Headers ➔ major part of the HTTP request that contains **all sorts of metadata** that can be useful when handling requests. A common usage is to use them for **authentication**, for example, via the famous **cookies**.

Setting cookies

```python
from fastapi import FastAPI, Response

app = FastAPI()

@app.get("/setCookie")
async def custom_cookie(response: Response):
    response.set_cookie("cookie-name",
                        "cookie-value",
                        max_age=86400)
    return {"hello": "world"}
```

```
http GET http://localhost:8000/setCookie
```



Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Headers ➔ major part of the HTTP request that contains **all sorts of metadata** that can be useful when handling requests. A common usage is to use them for **authentication**, for example, via the famous **cookies**.

Raising HTTP errors

```
from fastapi import FastAPI, Body, status, HTTPException

app = FastAPI()

@app.post("/password")
async def check_password(password: str = Body(...),
                         password_confirm: str = Body(...)):
    if password != password_confirm:
        raise HTTPException(
                            status.HTTP_400_BAD_REQUEST,
                            detail="Passwords don't match.",
                           )
        return {"message": "Passwords match."}
```

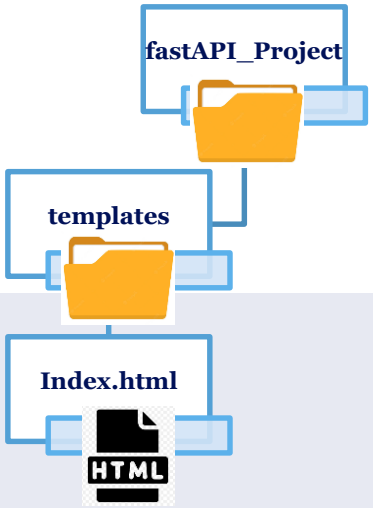
```
http POST http://localhost:8000/password password="aa" password_confirm="bb"
```

```
HTTP/1.1 400 Bad Request
content-length: 35
content-type: application/json
date: Tue, 10 Jan 2023 21:08:49 GMT
server: uvicorn

{
    "detail": "Passwords don't match."
}
```

Building Restful API with FastAPI

An Introduction to FastAPI

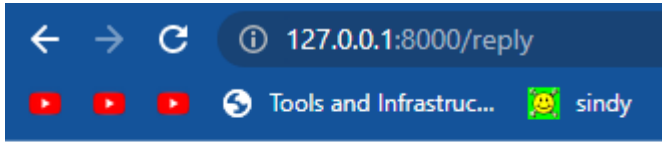
- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**



Building a custom HTML response

```
from fastapi import FastAPI
from fastapi.responses import HTMLResponse
from fastapi.templating import Jinja2Templates

app = FastAPI()
templates = Jinja2Templates(directory="templates")

@app.get("/reply")
async def home(request: Request):
    return
templates.TemplateResponse("/index.html",{"request":request})
```

http GET http://localhost:8000/reply



Hello world!

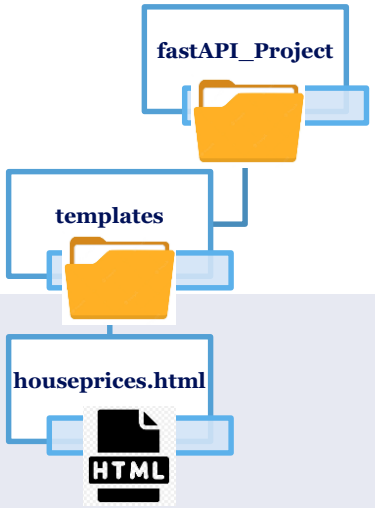
```
HTTP/1.1 200 OK
content-length: 164
content-type: text/html; charset=utf-8
date: Tue, 10 Jan 2023 21:34:35 GMT
server: uvicorn

<html>
        <head>
            <title>Hello world!</title>
        </head>
        <body>
          <h1>Hello world!</h1>
        </body>
</html>
```

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Building a custom HTML response with a dataframe



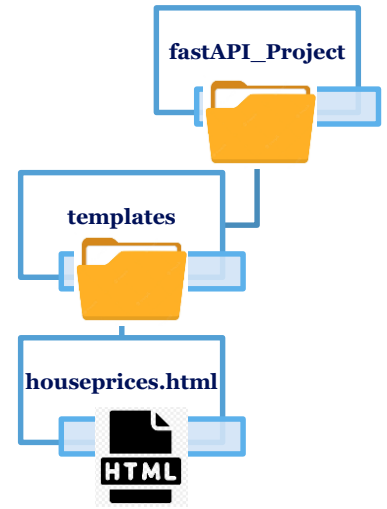

```
from fastapi import FastAPI
from fastapi.responses import HTMLResponse
from fastapi.templating import Jinja2Templates
import pandas as pd
import json

app = FastAPI()
templates = Jinja2Templates(directory="templates")

@app.get("/houseprices")
async def home(request: Request):
    df = pd.read_csv("data/house_pricing.csv", nrows=25)
    js = df.to_json(orient="records")
    data=json.loads(js)

    return templates.TemplateResponse("/houseprices.html",
                                      {"request":request,
                                      "house_prices":data})
```

Building Restful API with FastAPI

An Introduction to FastAPI

- **Handling request parameters**
  - FastAPI allows providing REST APIs in a structured way in which to interact with data
  - It allows end user to send some information to tailor the response they need, such as:
    - **Path parameters,**
    - **Query parameters,**
    - **Body payloads,**
    - **Headers**

Building a custom HTML response with a dataframe




```
<html>
<head>
    <title>House Prices Details</title>
</head>

<body>
    <center>
    <h1>house_prices </h1>
    <table>
        <tr>
        <th>date</th>
        <th>price</th>
        <th>bedrooms</th>
        <th>bathrooms</th>
        <th>sqft_living</th>
        <th>floors</th>
        <th>city</th>
        <th>country</th>
        </tr>
        {% for item in house_prices %}
        <tr>
            <td>{{ item.date }}</td>
            <td>{{ item.price }}</td>
            <td>{{ item.bedrooms }}</td>
            <td>{{ item.bathrooms }}</td>
            <td>{{ item.sqft_living }}</td>
            <td>{{ item.floors }}</td>
            <td>{{ item.city }}</td>
            <td>{{ item.country }}</td>
            </tr>
        {% endfor %}
    </table>
</center>
</body>
</html>
```

127.0.0.1:8000/houseprices

house_prices

date	price	bedrooms	bathrooms	sqft_living	floors	city	countr
2014-05-02 00:00:00	313000.0	3.0	1.5	1340	1.5	Shoreline	USA
2014-05-02 00:00:00	2384000.0	5.0	2.5	3650	2.0	Seattle	USA
2014-05-02 00:00:00	342000.0	3.0	2.0	1930	1.0	Kent	USA
2014-05-02 00:00:00	420000.0	3.0	2.25	2000	1.0	Bellevue	USA
2014-05-02 00:00:00	550000.0	4.0	2.5	1940	1.0	Redmond	USA
2014-05-02 00:00:00	490000.0	2.0	1.0	880	1.0	Seattle	USA
2014-05-02 00:00:00	335000.0	2.0	2.0	1350	1.0	Redmond	USA
2014-05-02 00:00:00	482000.0	4.0	2.5	2710	2.0	Maple Valley	USA
2014-05-02 00:00:00	452500.0	3.0	2.5	2430	1.0	North Bend	USA
2014-05-02 00:00:00	640000.0	4.0	2.0	1520	1.5	Seattle	USA
2014-05-02 00:00:00	463000.0	3.0	1.75	1710	1.0	Lake Forest Park	USA
2014-05-02 00:00:00	1400000.0	4.0	2.5	2920	1.5	Seattle	USA
2014-05-02 00:00:00	588500.0	3.0	1.75	2330	1.0	Sammamish	USA
2014-05-02 00:00:00	365000.0	3.0	1.0	1090	1.0	Seattle	USA
2014-05-02 00:00:00	1200000.0	5.0	2.75	2910	1.5	Seattle	USA
2014-05-02 00:00:00	242500.0	3.0	1.5	1200	1.0	Kent	USA
2014-05-02 00:00:00	419000.0	3.0	1.5	1570	1.0	Bellevue	USA
2014-05-02 00:00:00	367500.0	4.0	3.0	3110	2.0	Auburn	USA
2014-05-02 00:00:00	257950.0	3.0	1.75	1370	1.0	Des Moines	USA
2014-05-02 00:00:00	275000.0	3.0	1.5	1180	1.0	North Bend	USA
2014-05-02 00:00:00	750000.0	3.0	1.75	2240	2.0	Seattle	USA
2014-05-02 00:00:00	435000.0	4.0	1.0	1450	1.0	Bellevue	USA
2014-05-02 00:00:00	626000.0	3.0	2.25	1750	2.5	Seattle	USA
2014-05-02 00:00:00	612500.0	4.0	2.5	2730	2.0	Bothell	USA
2014-05-02 00:00:00	495000.0	4.0	1.75	1600	1.0	Seattle	USA

Thanks for your Listening

Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz