

## Практическая работа №5

### Задание:

Создадим тесты для этого бэкенда в JMeter.

#### 1. Создание тестового плана

- Откройте JMeter.
- Создайте новый тестовый план (Test Plan).

#### 2. Добавление Thread Group

- Правый клик на Test Plan → Add → Threads (Users) → Thread Group.
- Установите "Number of Threads (users)" на 5, "Ramp-Up Period (seconds)" на 10 и "Loop Count" на 1.

#### 3. Добавление HTTP Request Sampler для POST запроса

- Правый клик на Thread Group → Add → Sampler → HTTP Request.
- В настройках HTTP Request:
  - Name: Create Item
  - Server Name or IP: 127.0.0.1
  - Port Number: 8000
  - Method: POST
  - Path: /items/
  - Body Data: Вставьте пример данных:

json

Копировать код

```
{  
  "name": "Item 1",  
  "description": "A description of Item 1",  
  "price": 100.0  
}
```

#### 4. Добавление HTTP Request Sampler для GET запроса

- Правый клик на Thread Group → Add → Sampler → HTTP Request.
- В настройках HTTP Request:
  - Name: Get Items

- Server Name or IP: 127.0.0.1
- Port Number: 8000
- Method: GET
- Path: /items/

## 5. Добавление Listener

- Правый клик на Thread Group → Add → Listener → View Results Tree.

## 6. Запуск тестов

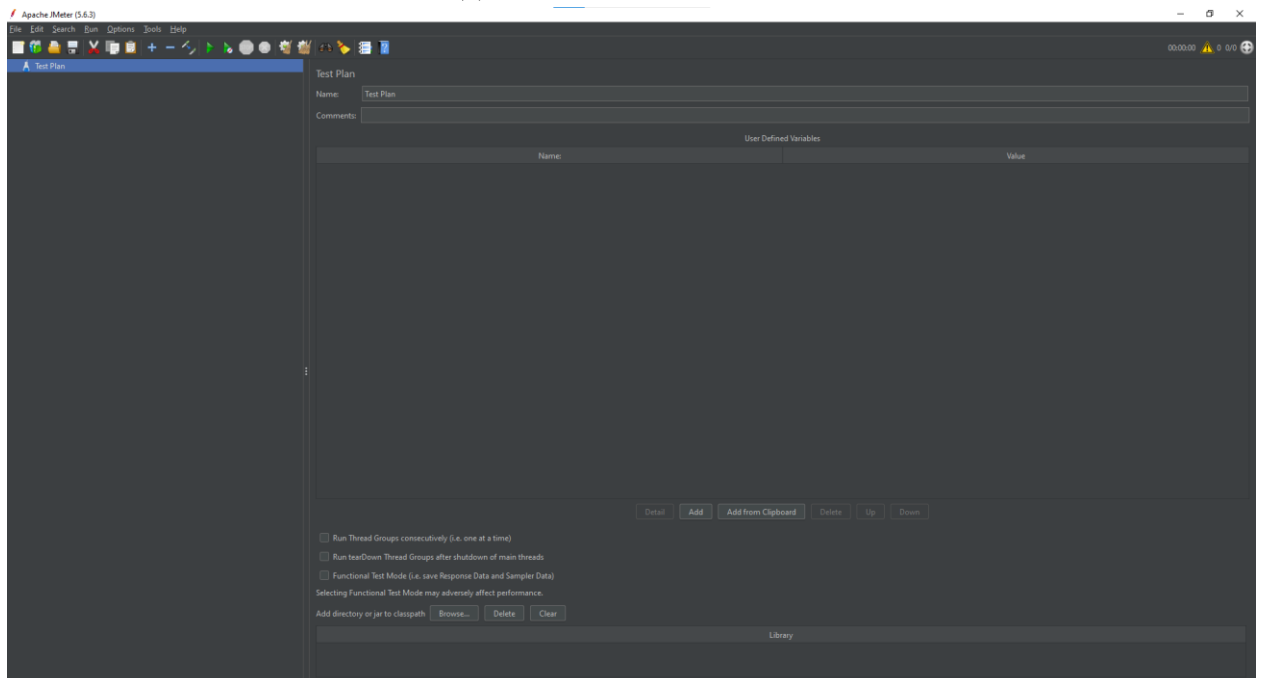
- Нажмите кнопку "Start" на панели инструментов JMeter.
- После завершения теста, просмотрите результаты в "View Results Tree".

## Пример тестов

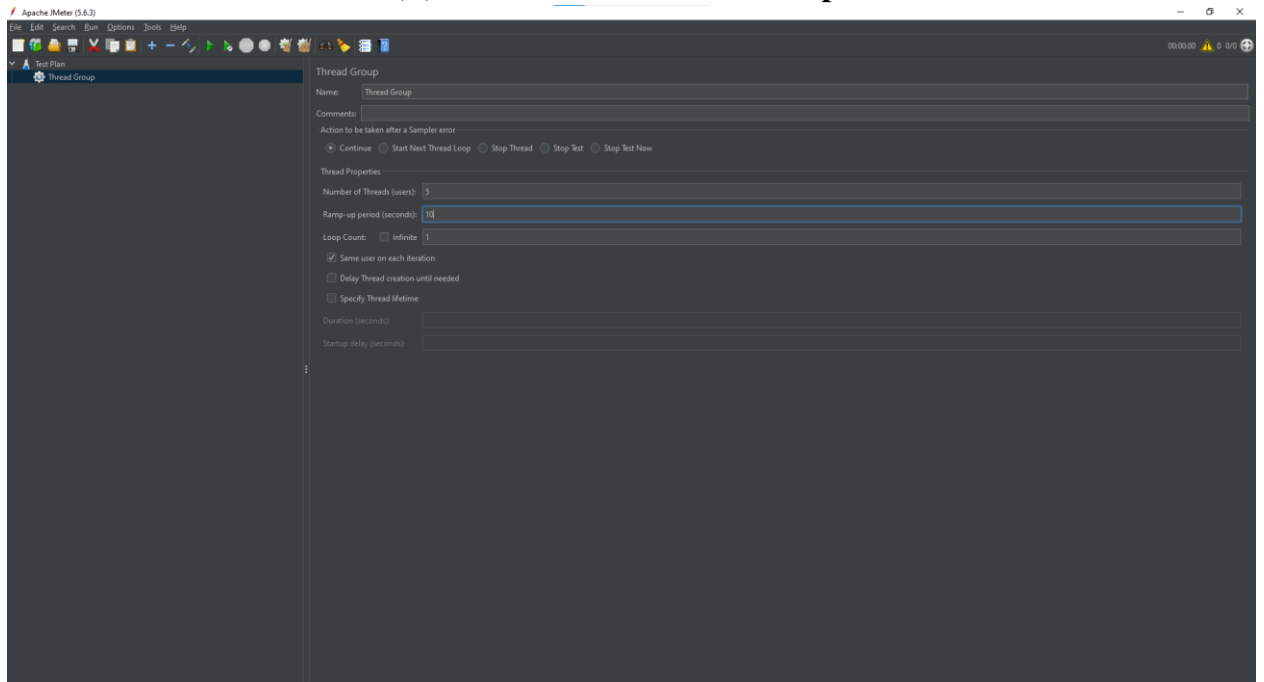
- POST запрос добавляет новый элемент на аукцион и возвращает его данные.
- GET запрос получает список всех элементов на аукционе.

**Далее, придерживаясь инструкции создаем тесты:**

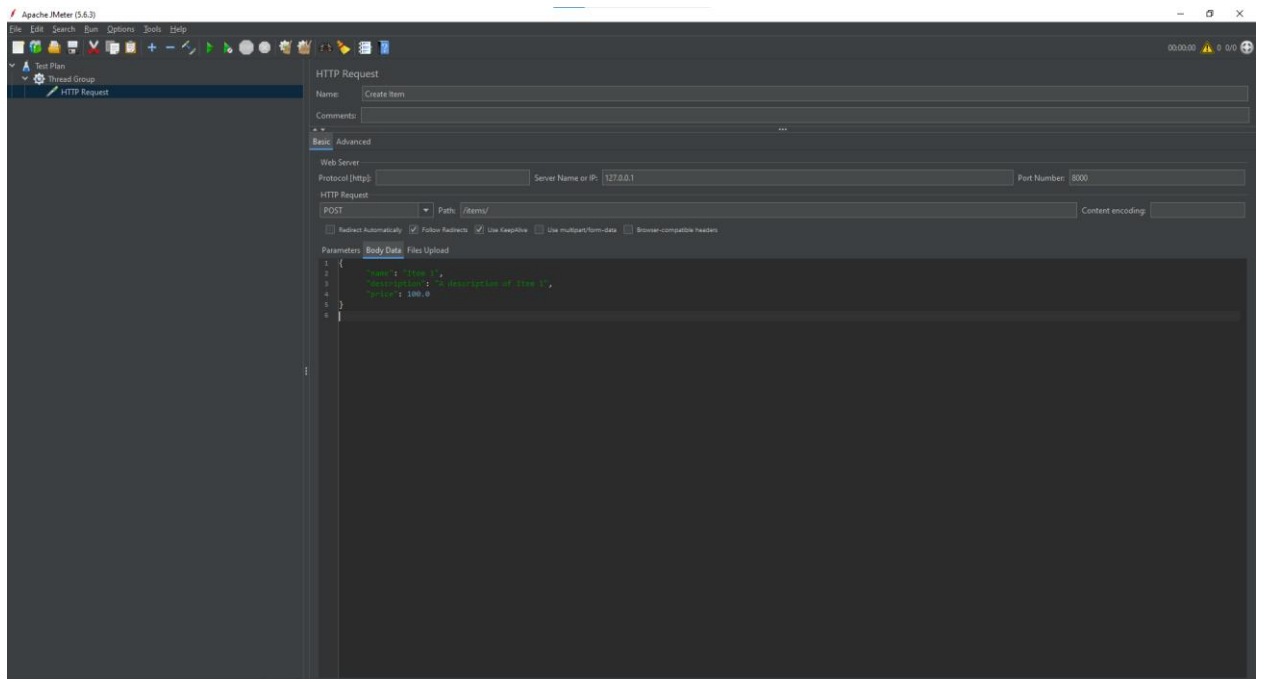
## 1.Создание тестового плана



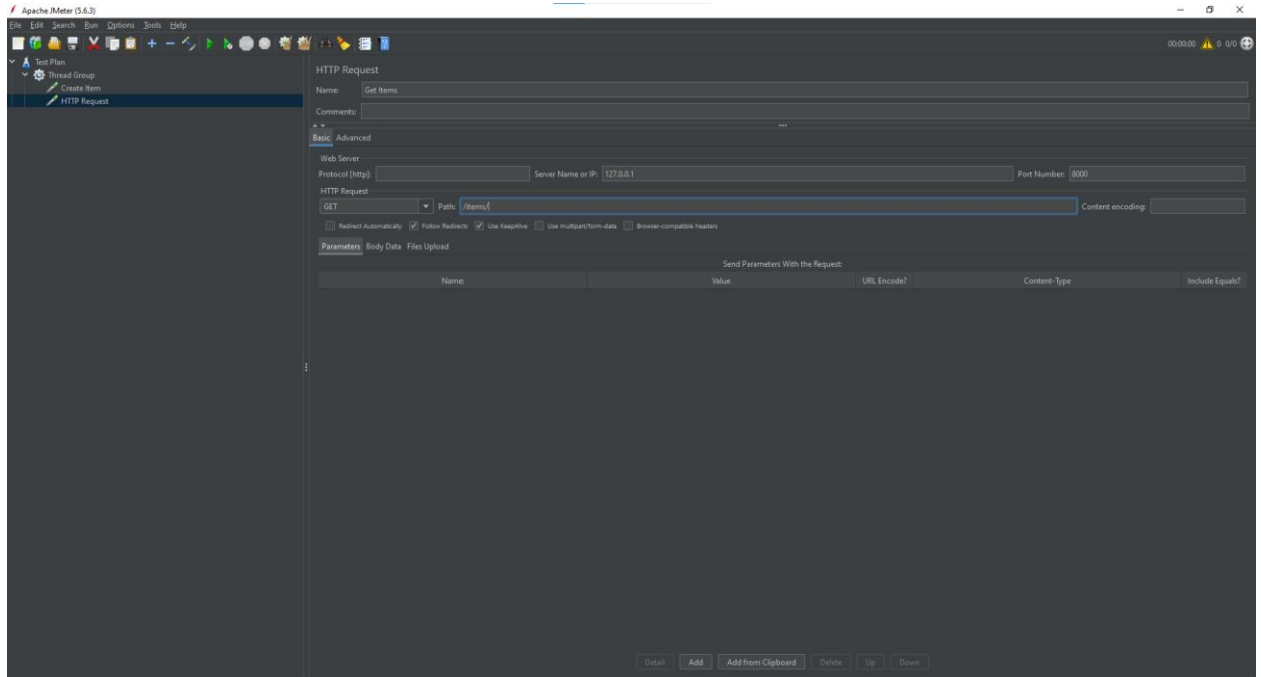
## 2.Добавление Thread Group



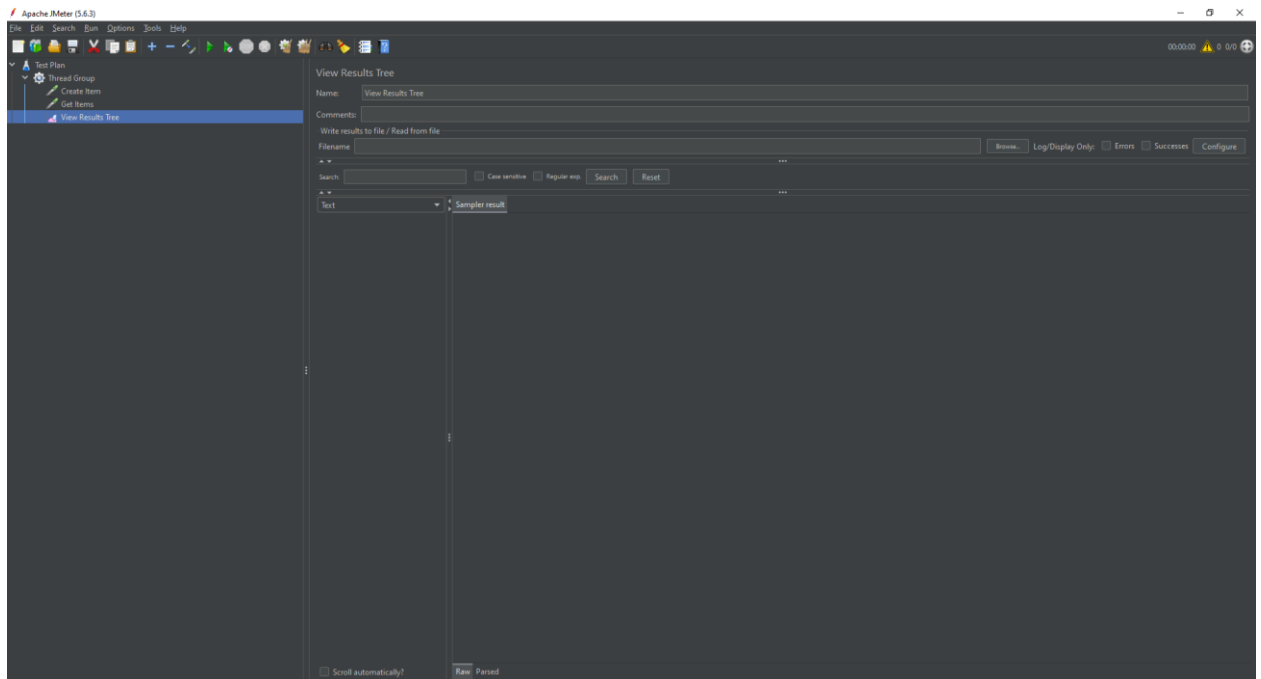
## 3. Добавление HTTP Request Sampler для POST запроса



## 4.Добавление HTTP Request Sampler для GET запроса



## 5. Добавление Listener



## 6. Запуск сервера

```
main.py x
32 def create_item(item: Item):
33     cursor = conn.cursor()
34     cursor.execute( sql: """
35     INSERT INTO items (name, description, price)
36     VALUES (?, ?, ?)
37     """, parameters: (item.name, item.description, item.price))
38     conn.commit()
39     item_id = cursor.lastrowid
40     conn.close()
41     return Item(id=item_id, name=item.name, description=item.description, price=item.price)
42
43
44 @app.get( path: "/items/", response_model=List[Item])
45 def get_items():
46     conn = sqlite3.connect('auction.db')
47     cursor = conn.cursor()
48     cursor.execute('SELECT * FROM items')
49     rows = cursor.fetchall()
50     conn.close()
51     return [Item(id=row[0], name=row[1], description=row[2], price=row[3]) for row in rows]
52
53 @app.get( path: "/items/{item_id}", response_model=Item)
54 def get_item(item_id: int):
55     conn = sqlite3.connect('auction.db')
56     cursor = conn.cursor()
57     cursor.execute( sql: 'SELECT * FROM items WHERE id = ?', parameters: (item_id,))
58     row = cursor.fetchone()
59     conn.close()
60     if row is None:
61         raise HTTPException(status_code=404, detail="Item not found")
62     return Item(id=row[0], name=row[1], description=row[2], price=row[3])
63
64 if __name__ == "__main__":
65     import uvicorn
66     uvicorn.run(app, host="127.0.0.1", port=8000)

Run main x
INFO: Started server process [15492]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
```

## 7. Запуск тестов

View Results Tree (C:\WINDOWS\system32\cmd - Apache JMeter 3.6.3)

Test Plan

- Thread Group
  - Create Item
  - HTTP Header Manager
  - HTTP Header Manager
  - View Results Tree

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Browse Log/Display Only: Errors Successes Configure

Search: Case sensitive Regular exp Search Reset

Test

- Test
  - Sampler result Request Response data
  - Test
    - Thread Name: Thread Group 1-1
    - Sample Start: 2025-10-26 15:39:40 GMT+0400
    - Load Name: 50
    - Connect Time: 50
    - Latency: 62
    - Size in bytes: 405
    - Send bytes: 277
    - Header's size in bytes: 145
    - Body size in bytes: 260
    - Sample Count: 1
    - Error Count: 0
    - Data type ("text/plain") text
    - Response message: Unprocessable Content
    - HTTPSampleResult fields:
      - Content type: application/json
      - Data encoding: null

Score: automatically? Run: Passed

