

Лабораторная работа №4:

Цель задания

Разработать приложение для получения курсов валют с сайта Центробанка России, используя объектно-ориентированный подход и архитектурный паттерн MVC (Model-View-Controller).

Требования к функционалу

1. Компонент Model:

- Представляет собой класс CurrencyRates, который отправляет запросы к сайту ЦБ РФ.
- На вход принимает список отслеживаемых валют (строки длины три, например, ["USD", "EUR", "GBP"]).
- Реализует паттерн "Одиночка" (Singleton), чтобы в приложении существовал только один объект данного класса.
- Использует геттер для получения текущих курсов валют.
- Использует сеттер для обновления списка отслеживаемых валют.

2. Компонент View:

- Использует шаблонизатор Jinja2 или какую-либо другую библиотеку ([пример](#)) для отображения данных.
- Позволяет пользователю просматривать курсы валют в удобном формате.

3. Компонент Controller:

- Обрабатывает запросы и управляет логикой приложения.
- Сохраняет полученные данные в базу данных SQLite3 (реализован именованный стиль для параметризации запросов в БД).
- Передаёт данные в представление для отображения.

Дополнительные требования

- Данные должны загружаться с официального API или XML-файла Центробанка России.
- Для взаимодействия с API можно использовать библиотеку requests.
- Использование библиотек sqlite3 для работы с базой данных и jinja2 для шаблонов.

Этапы выполнения

1. Реализовать класс CurrencyRates (паттерн "Одиночка").
2. Реализовать контроллер для обработки запросов и сохранения данных в SQLite3.
3. Разработать шаблоны на Jinja2 для отображения информации.
4. Написать код для интеграции всех компонентов в MVC-структуре.
5. Протестировать приложение, проверив работу с разными валютами.

main.py:

```
ЛР-4 > main.py > CurrencyRates > __init__
1  import requests
2  from xml.etree import ElementTree
3  from datetime import datetime
4
5
6  class CurrencyRates:
7      _instance = None
8      URL = "https://www.cbr.ru/scripts/XML_daily.asp"
9      SUPPORTED_CURRENCIES = ['USD', 'EUR', 'GBP'] # Только эти валюты поддерживаются
10
11     def __new__(cls, char_codes=None):
12         if cls._instance is None:
13             cls._instance = super(CurrencyRates, cls).__new__(cls)
14             cls._instance.initialized = False
15         return cls._instance
16
17     def __init__(self, char_codes=None):
18         if not self.initialized:
19             # Фильтруем коды валют, оставляем только поддерживаемые
20             filtered_codes = [code for code in (char_codes or self.SUPPORTED_CURRENCIES)
21                               if code in self.SUPPORTED_CURRENCIES]
22             self._char_codes = filtered_codes or self.SUPPORTED_CURRENCIES
23             self._rates = {}
24             self.update_rates()
25             self.initialized = True
26
27     def update_rates(self):
28         try:
29             response = requests.get(self.URL, timeout=10)
30             response.raise_for_status()
31
32             tree = ElementTree.fromstring(response.content)
33             self._rates = {}
34             update_time = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
```

```

36         for valute in tree.findall('.//Valute'):
37             char_code = valute.find('CharCode').text
38             if char_code in self._char_codes:
39                 value = float(valute.find('Value').text.replace(',', '.'))
40                 nominal = int(valute.find('Nominal').text)
41                 self._rates[char_code] = {
42                     'value': round(value / nominal, 4),
43                     'name': valute.find('Name').text,
44                     'date': update_time
45                 }
46             return True
47         except Exception as e:
48             print(f"Ошибка при загрузке курсов: {e}")
49             return False
50
51     def get_all_rates(self):
52         return [
53             (code, info['name'], info['value'], info['date'])
54             for code, info in self._rates.items()
55         ]
56
57     @property
58     def char_codes(self):
59         return self._char_codes
60
61     @char_codes.setter
62     def char_codes(self, new_codes):
63         # Фильтруем новые коды валют
64         filtered_codes = [code for code in new_codes if code in self.SUPPORTED_CURRENCIES]
65         if filtered_codes:
66             self._char_codes = filtered_codes
67             self.update_rates()

```

__init__.py:

```
ЛР-4 > controllers > __init__.py > CurrencyRatesCRUD
1  import sqlite3
2  from datetime import datetime
3
4
5  class CurrencyRatesCRUD:
6      def __init__(self, currency_rates_obj):
7          self.__connection = sqlite3.connect('currency_rates.db')
8          self.cursor = self.__connection.cursor()
9          self.currency_rates = currency_rates_obj
10         self._create_table()
11
12     def _create_table(self):
13         self.cursor.execute("""
14             CREATE TABLE IF NOT EXISTS currency_rates (
15                 id INTEGER PRIMARY KEY AUTOINCREMENT,
16                 char_code TEXT NOT NULL,
17                 name TEXT NOT NULL,
18                 value REAL NOT NULL,
19                 date TEXT NOT NULL
20             )
21         """)
22         self.__connection.commit()
23
24     def create(self, data=None):
25         try:
26             if data is None:
27                 raw_data = self.currency_rates.get_all_rates()
28                 data = [
29                     {
30                         'char_code': code,
31                         'name': name,
32                         'value': value,
33                         'date': date
34                     }
35                     for code, name, value, date in raw_data
36                 ]
37
38             if not data:
39                 print("Нет данных для записи")
40                 return False
```

```
42         sql = """
43             INSERT INTO currency_rates
44             (char_code, name, value, date)
45             VALUES (:char_code, :name, :value, :date)
46         """
47
48         self.cursor.executemany(sql, data)
49         self.__connection.commit()
50         print(f"Успешно записано {len(data)} записей")
51         return True
52
53     except Exception as e:
54         print(f"Ошибка при записи в БД: {e}")
55         self.__connection.rollback()
56         return False
57
58     def read(self, char_code=None):
59         try:
60             if char_code:
61                 self.cursor.execute("""
62                     SELECT char_code, name, value, date
63                     FROM currency_rates
64                     WHERE char_code = ?
65                     ORDER BY date DESC
66                 """, (char_code,))
67             else:
68                 self.cursor.execute("""
69                     SELECT char_code, name, value, date
70                     FROM currency_rates
71                     ORDER BY date DESC
72                 """)
73             return self.cursor.fetchall()
74         except sqlite3.Error as e:
75             print(f"Ошибка при чтении из БД: {e}")
76             return []
```

```
78     def update_rates(self):
79         try:
80             if not self.currency_rates.update_rates():
81                 return False
82             return self.create()
83         except Exception as e:
84             print(f"Ошибка при обновлении данных: {e}")
85             return False
86
87     def close(self):
88         try:
89             self.__connection.close()
90         except sqlite3.Error as e:
91             print(f"Ошибка при закрытии соединения: {e}")
92
93
94 class ViewController:
95     def __init__(self, currency_rates):
96         self.currency_rates = currency_rates
97
98     def __call__(self):
99         rates = self.currency_rates.get_all_rates()
100         if not rates:
101             return "Нет данных в курсах валют"
102
103         result = ["Текущие курсы валют:"]
104         for code, name, value, date in rates:
105             result.append(f"{code} ({name}): {value} руб. (на {date})")
106
107         return "\n".join(result)
```

app.py:

```
ЛР-4 > app.py > ...
1  from main import CurrencyRates
2  from controllers import CurrencyRatesCRUD, ViewController
3
4
5  def main():
6      # Инициализация с поддержкой только USD, EUR, GBP
7      c_r = CurrencyRates(['USD', 'EUR', 'GBP'])
8
9      # Обновляем курсы валют
10     if not c_r.update_rates():
11         print("Не удалось обновить курсы валют")
12         return
13
14     # Инициализация контроллера БД
15     crud = CurrencyRatesCRUD(c_r)
16
17     # Запись данных в БД
18     if not crud.create():
19         print("Не удалось записать данные в БД")
20         return
21
22     # Вывод информации
23     view = ViewController(c_r)
24     print(view())
25
26     # Закрытие соединения с БД
27     crud.close()
28
29 if __name__ == "__main__":
30     main()
```

Результат:

```
PS C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс  
313/python.exe "c:/Users/sambu/OneDrive/Рабочий стол/Учеба в Ге  
Успешно записано 3 записей  
Текущие курсы валют:  
GBP (Фунт стерлингов): 106.7723 руб. (на 2025-06-12 22:14:00)  
USD (Доллар США): 79.0028 руб. (на 2025-06-12 22:14:00)  
EUR (Евро): 90.0068 руб. (на 2025-06-12 22:14:00)
```