

Объектно-ориентированное программирование

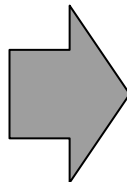
Object-oriented programming

XXII. Интерфейсы программирования приложений

Application Programming Interfaces

Application Programming Interface

Способ для двух или более компьютерных программ взаимодействовать друг с другом



Микроинтерфейс

```
void __cdecl qsort(  
    void *          _Base,  
    size_t         _NumOfElements,  
    size_t         _SizeOfElements,  
    int (__cdecl * _PtFuncCompare) (const void *, const void *));
```

Windows.h (WinAPI)

```
#include <Windows.h>
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam);
int WINAPI WinMain(
    HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR commandLine, INT commandShow)
{
    HWND hwnd = NULL; MSG msg = {}; WNDCLASS wc = {};
    wc.hInstance = hInstance; wc.lpfnWndProc = WndProc; wc.lpszClassName = "Window Class";
    wc.style = CS_HREDRAW | CS_VREDRAW;
    if (!RegisterClass(&wc)) return 1;
    hwnd = CreateWindow("Window Class", "App Title", WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
        CW_USEDEFAULT, 500, 500, NULL, NULL, hInstance, NULL);
    if (!hwnd) return 1;
    while (GetMessage(&msg, NULL, 0, 0)) {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
    return msg.wParam;
}
```

<https://learn.microsoft.com/en-us/windows/win32/apiindex/windows-api-list>

Windows.h (WinAPI)

```
LRESULT CALLBACK WndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam) {  
    switch(msg) {  
        case WM_LBUTTONDOWN:  
            MessageBox(  
                hwnd, "Left Button Clicked", "An Event Happened", MB_OK | MB_ICONINFORMATION);  
            return 0;  
        case WM_CLOSE:  
            PostQuitMessage(0);  
            return 0;  
        default:  
            return DefWindowProc(hwnd, msg, wParam, lParam);  
    }  
}
```

<https://learn.microsoft.com/en-us/windows/win32/learnwin32/your-first-windows-program>

“While working on Dota 2 in 2012, several engineers at Valve joined the team to help. I was working on something that used a lot of Window's API calls and I was grumpy and said "ugh **Windows API sucks.**" One of the engineers said, "You know, **I wrote USER.DLL, I was 19** at the time."”

D. Kirsch

<https://twitter.com/ZoidCTF/status/1726719694919569410>

Другие интерфейсы

- **Графические**
 - DirectX (DirectDraw/Direct3D)
 - Vulkan
 - OpenGL
- **Платформенные**
 - Android/iOS
 - Graph API
 - CUDA
- **Сетевые**
 - gRPC
 - GraphQL
 - Web API

Например

```
"meta": {
  "build_time":
    "2021-06-01T07:03:25.055Z",
  "license": "CC-BY-4.0",
  "version": "2.0-beta",
  "field_definitions": [
    {
      "name": "Total test results",
      "field": "tests.pcr.total",
      "deprecated": false,
      "prior_names": [
        "totalTestResults"
      ]
    }
  ],
},
```

```
"data": {
  "date": "2021-01-02",
  "states": 56,
  "cases": {
    "total": {
      "value": 20327598,
      "calculated": {
        "population_percent":
          6.1451,
        "change_from_prior_day":
          280318,
        "seven_day_change_percent":
          7.6
      }
    }
  }
},
```

<https://covidtracking.com/data/api/version-2>

XXIII. Модель Ричардсона*

Richardson maturity model

* <https://www.crummy.com/>

L0. Plain 'ol XML

```
POST /appointmentService HTTP/1.1
[various other headers]
<openSlotRequest date = "2010-01-04" doctor = "mjones"/>
...
HTTP/1.1 200 OK
[various headers]
<openSlotList>
  <slot start = "1400" end = "1450">
    <doctor id = "mjones"/>
  </slot>
  <slot start = "1600" end = "1650">
    <doctor id = "mjones"/>
  </slot>
</openSlotList>
```

<https://martinfowler.com/articles/richardsonMaturityModel.html>

L0. Plain 'ol XML

```
POST /appointmentService HTTP/1.1
```

```
[various other headers]
```

```
<appointmentRequest>
```

```
  <slot doctor = "mjones" start = "1400" end = "1450"/>
```

```
  <patient id = "jsmith"/>
```

```
</appointmentRequest>
```

L1. Resources

```
POST /doctors/mjones HTTP/1.1
```

```
[various other headers]
```

```
<openSlotRequest date = "2010-01-04"/>
```

```
...
```

```
HTTP/1.1 200 OK
```

```
[various headers]
```

```
<openSlotList>
```

```
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
```

```
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
```

```
</openSlotList>
```

Вместо одного “класса”, который отвечает за все поведение программы, делается много “классов”, каждый из которых отвечает за свою специализацию (“single responsibility”)

L1. Resources

```
POST /slots/1234 HTTP/1.1  
[various other headers]  
<appointmentRequest>  
  <patient id = "jsmith"/>  
</appointmentRequest>
```

Аналогично объектам, ресурс позволяет отождествить поведение системы с конкретной сущностью, как метод с экземпляром

L2. HTTP methods

```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
```

```
Host: royalhope.nhs.uk
```

```
...
```

```
POST /slots/1234 HTTP/1.1
```

```
[various other headers]
```

```
<appointmentRequest>
```

```
  <patient id = "jsmith"/>
```

```
</appointmentRequest>
```

В предыдущих примерах метод POST использовался из-за необходимости соответствовать протоколу HTTP

L2. HTTP methods

HTTP/1.1 201 Created

Location: slots/1234/appointment

[various headers]

<appointment>

 <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>

 <patient id = "jsmith"/>

</appointment>

В этом случае “глаголы” HTTP используются в строгом соответствии со стандартом, GET – безопасная операция (ничего не изменяет), а значит – позволяет кэшировать результат

<https://www.rfc-editor.org/rfc/rfc9110.html#name-method-definitions>

L2. HTTP methods

HTTP/1.1 409 Conflict

[various headers]

<openSlotList>

 <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>

</openSlotList>

Второй инструмент – коды ошибок, например, в диапазоне 400-499, для того, чтобы не включать сообщение об ошибке в сам ответ

L3. HATEOAS

```
HTTP/1.1 200 OK
```

```
[various headers]
```

```
<openSlotList>
```

```
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450">
```

```
    <link rel = "/linkrels/slot/book"  
          uri = "/slots/1234"/>
```

```
  </slot>
```

```
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650">
```

```
    <link rel = "/linkrels/slot/book"  
          uri = "/slots/5678"/>
```

```
  </slot>
```

```
</openSlotList>
```

Hypertext As The Engine Of Application State – наличие “подсказок” о дальнейших действиях

L3. HATEOAS

HTTP/1.1 201 Created

Location: <http://royalhope.nhs.uk/slots/1234/appointment>

[various headers]

<appointment>

 <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>

 <patient id = "jsmith"/>

 <link rel = "/linkrels/appointment/cancel"

 uri = "/slots/1234/appointment"/>

 <link rel = "/linkrels/appointment/changeTime"

 uri = "/doctors/mjones/slots?date=20100104&status=open"/>

 ...

</appointment>

Это, например, позволяет изменять схему без ведома клиентов, улучшает “видимость” ресурсов (discoverability)

Representational State Transfer (REST)

“If the engine of application state (and hence the API) is not being driven by hypertext, then it cannot be **RESTful** and cannot be a REST API.”

L1. Борьба с усложнением архитектуры с помощью принципа “разделяй и властвуй” (разбиваем страницу на множество ресурсов)

L2. Стандартные термины (HTTP-методы в примере) позволяют унифицировать запросы со стороны клиентов

L3. Включение доступных ресурсов в тело ответа позволяет сделать ресурсы видимыми для клиентов, играя роль документации/спецификации

https://ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

RESTful API

- ... не зависит от конкретного протокола передачи данных (identification != interaction, **separation of concerns**)
- ... не должен изменять протокол, но может дополнять его, если спецификация отсутствует, напр., **PATCH**, **link** (**generic interface**)
- ... не требует от клиента предварительного знания, вся информация доступна из самого интерфейса (**hypertext-driven**, “code-on-demand”)
- ... не привязывает клиентов к конкретной схеме, позволяя серверу менять структуру, при условии, что новая информация доступна в интерфейсе (**decoupling** of client and server)