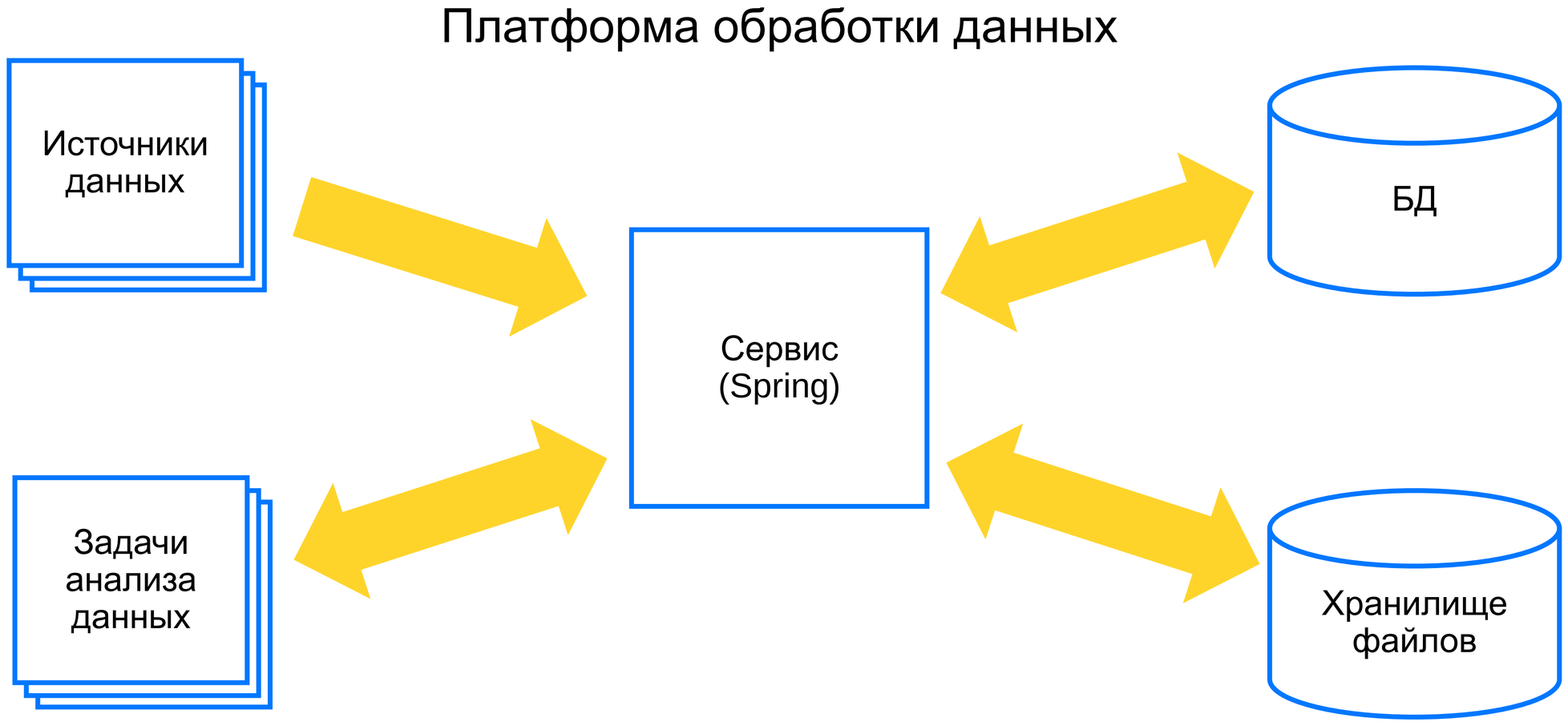
Spring REST гигантомания

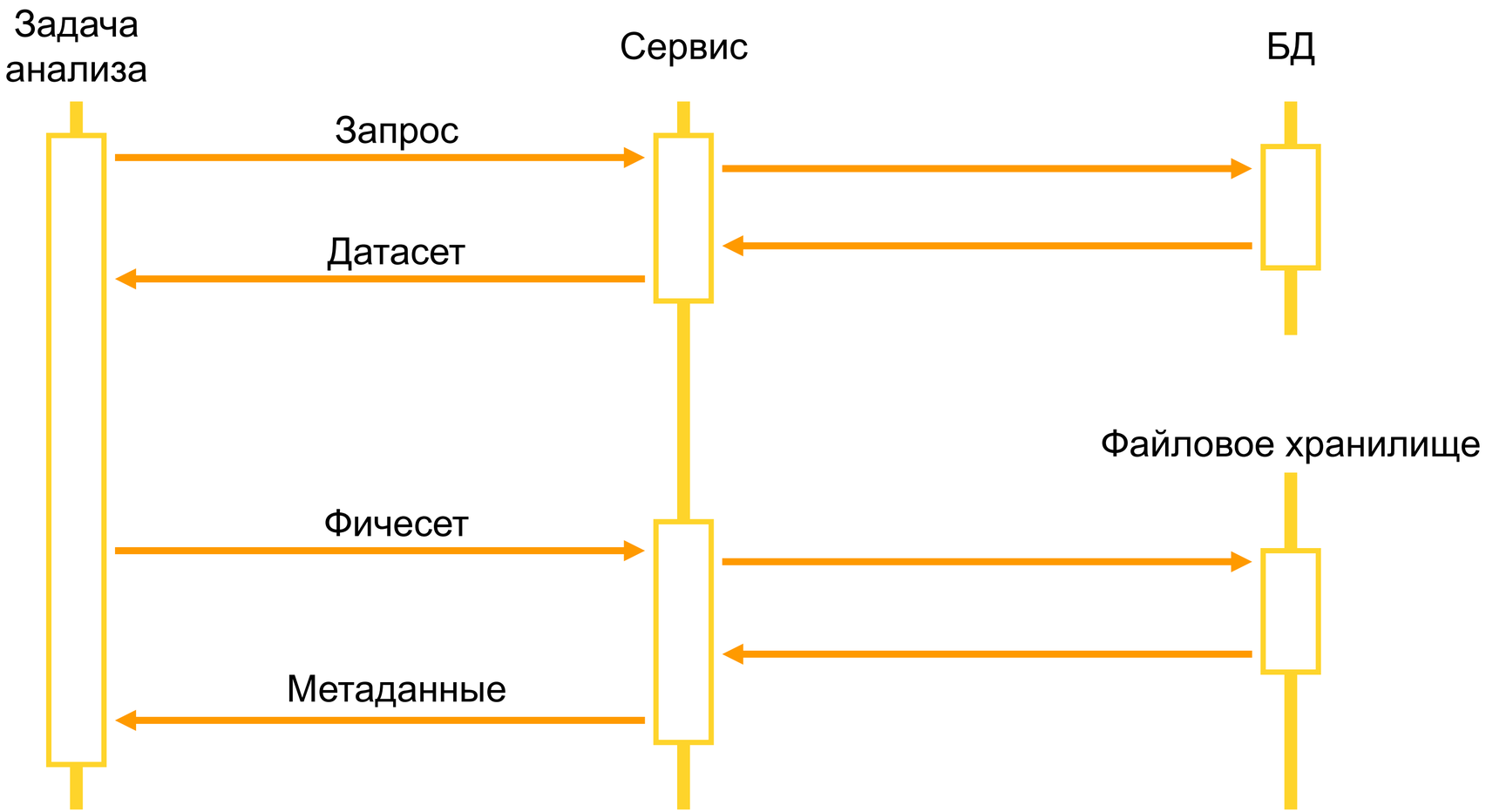
особенности работы с мультигигабайтными POST запросами

Алексей Рагозин

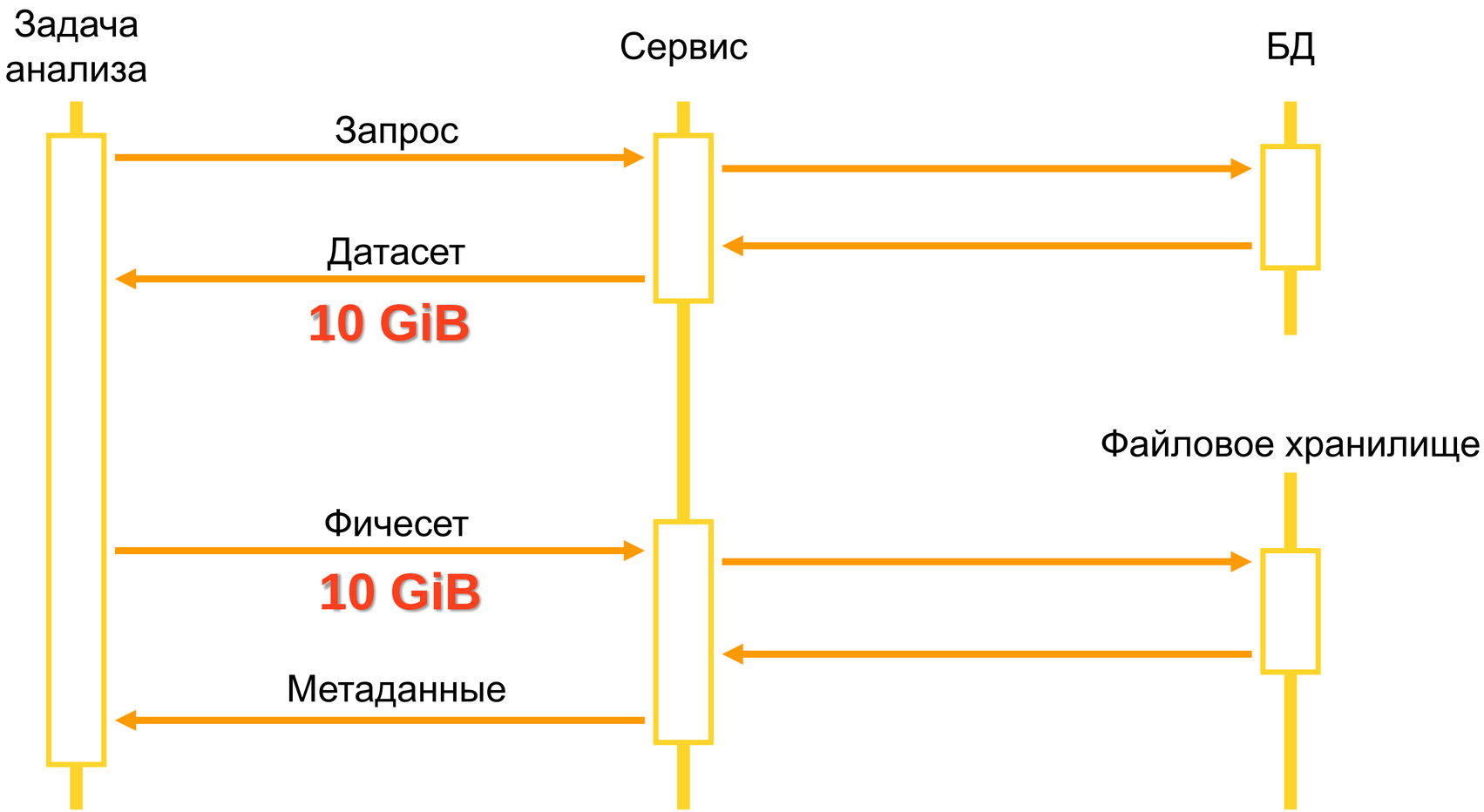
Как я дошёл до жизни такой?



Типичный сценарий использования



Типичный сценарий использования



Постановка задачи

- Лимит памяти для сервиса – 512 MiB
- “Условное” ограничение размера запроса – 10 GiB
- Зоопарк клиентов: Java, Python, curl
- API менять нельзя
- До сдачи проекта – 9 недель

Ключевые подзадачи

Входящие данные по HTTP

Сервер | Клиент

? | ?

Исходящие данные по HTTP

Сервер | Клиент

? | ?

Запись данных в PostgreSQL

?

Чтение данных из PostgreSQL

?

HTTP POST Запрос

```
POST http://wp.loadlab.ragozin.info/wp-login.php HTTP/1.1
Host: wp.loadlab.ragozin.info
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:66.0) Gecko/20100101 Firefox...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 110
Referer: http://wp.loadlab.ragozin.info/wp-login.php
Connection: keep-alive
Cookie: wp-settings-time-1=1557258334; wordpress_test_cookie=WP+Cookie+check; wordp ...
Upgrade-Insecure-Requests: 1

log=boss&pwd=boss&wp-submit=Log+In&redirect_to=http%3A%2F%2Fwp.loadlab.ragozin.info%...
```

HTTP Ответ

```
HTTP/1.1 200 OK
Server: nginx/1.16.0
Date: Thu, 16 May 2019 13:45:24 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
X-Powered-By: PHP/7.3.5
Link: <http://wp.loadlab.ragozin.info/wp-json/>; rel="https://api.w.org/"

1f6a
<!DOCTYPE html>
<html lang="en-US" class="no-js">
<head>
...
```

Начнём с простого
– RestTemplate

RestTemplate - отправка

Как отправить 10 GiB через RestTemplate?

Специальная поддержка

- **`File, Path, Resource`**
- **`MultiValueMap`**

Для менее тривиальных сценариев
используем – **`HttpMessageConverter`**

RestTemplate - отправка

```
public void postUnlimited(String path, InputStream source) throws IOException {

        RestTemplate template = ...

        ...

        RequestEntity<Resource> request = new RequestEntity<>(new InputStreamResource(source),
                        HttpMethod.POST, URI.create(serverUrl + path));

        ResponseEntity<String> resp = template.exchange(request, String.class);

        ...
}
```

RestTemplate - отправка

```
public void postUnlimited(String path, OutputStreamCallback dataProducer) {
        RestTemplate template = ...
        ...
        template.getMessageConverters().add(
                new AbstractGenericHttpMessageConverter<OutputStreamCallback>() {
                ...
                @Override
                protected void writeInternal(OutputStreamCallback callback,
                                             Type type, HttpOutputMessage outputMessage) {
                        callback.process(outputMessage.getBody());
                }
        });

        RequestEntity<OutputStreamCallback> request = new RequestEntity<>(
                        dataProducer, HttpMethod.POST, URI.create(serverUrl + path));
        ResponseEntity<String> resp = template.exchange(request, String.class);
}
```

RestTemplate - получение

```
public void getUnlimited(String path) throws IOException {
        ...
        RestTemplate template = ...

        ...
        RequestEntity<Void> request = new RequestEntity<>(HttpMethod.GET,
                                                          URI.create(serverUrl + path));

        ResponseEntity<Resource> resp = template.exchange(request, Resource.class);

        process(resp.getBody().getInputStream());
}
```

RestTemplate - получение

```
public void getUnlimited(String path) throws IOException {
        ...
        RestTemplate template = ...

        ...
        RequestEntity<Void> request = new RequestEntity<>(HttpMethod.GET,
                                                          URI.create(serverUrl + path));

        ResponseEntity<Resource> resp = template.exchange(request, Resource.class);

        process(resp.getBody().getInputStream());
}
```

Так не работает – получим OutOfMemoryError

RestTemplate - получение

```
public void getUnlimited(String path) throws IOException {

        RestTemplate template = ...

        ...
        template.execute(URI.create(serverUrl + path), HttpMethod.GET,
                req -> {},
                rsp -> {
                        process(rsp.getBody());
                        return null;
        });
}
```

А что на серверной стороне?

Отправка “неограниченого” ответа

Отправка не проблема - StreamingResponseBody

```
@RequestMapping(...)
public ResponseEntity<StreamingResponseBody> getStreamUnlim(...) throws IOException {
        return ResponseEntity.ok(outputStream -> {

            ...

        });
}
```

Servlet API

Сигнатура HTTP обработчика в Servlet API

```
void service(HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException;
```

Можем воспользоваться этим в Spring

```
@RequestMapping("...")
public void handle(
        HttpServletRequest request, HttpServletResponse response)
                                                throws IOException {
        InputStream is = request.getInputStream();
        ...
        response.setStatus(200);
        ...
        OutputStream os = response.getOutputStream();
        ...
}
```

Можно миксовать с @RequestParam и другими фишками Spring MVC

Но есть нюанс …

```
curl -F featureId=… -F format=csv -F file=@0001.zip $UPLOAD_URL
```

```
POST /features/upload HTTP/1.1
Host: 127.0.0.1:8080
User-Agent: curl/7.63.0
Accept: */*
Transfer-Encoding: chunked
Content-Type: multipart/form-data; boundary=------------------------ea9fcb2b8436a092
Expect: 100-continue

--------------------------ea9fcb2b8436a092
Content-Disposition: form-data; name="featureId"

...
--------------------------ea9fcb2b8436a092
Content-Disposition: form-data; name="file"; filename="0001.zip"
Content-Type: application/octet-stream

PK........{..X.9^.v.......... C.s..F.WM..0...W.......Y. . .....
W7q..c..iwY.□g.$u........)..f.f.....y!.k..W..d...1..J...M>□z..L.
...
--------------------------ea9fcb2b8436a092--
```

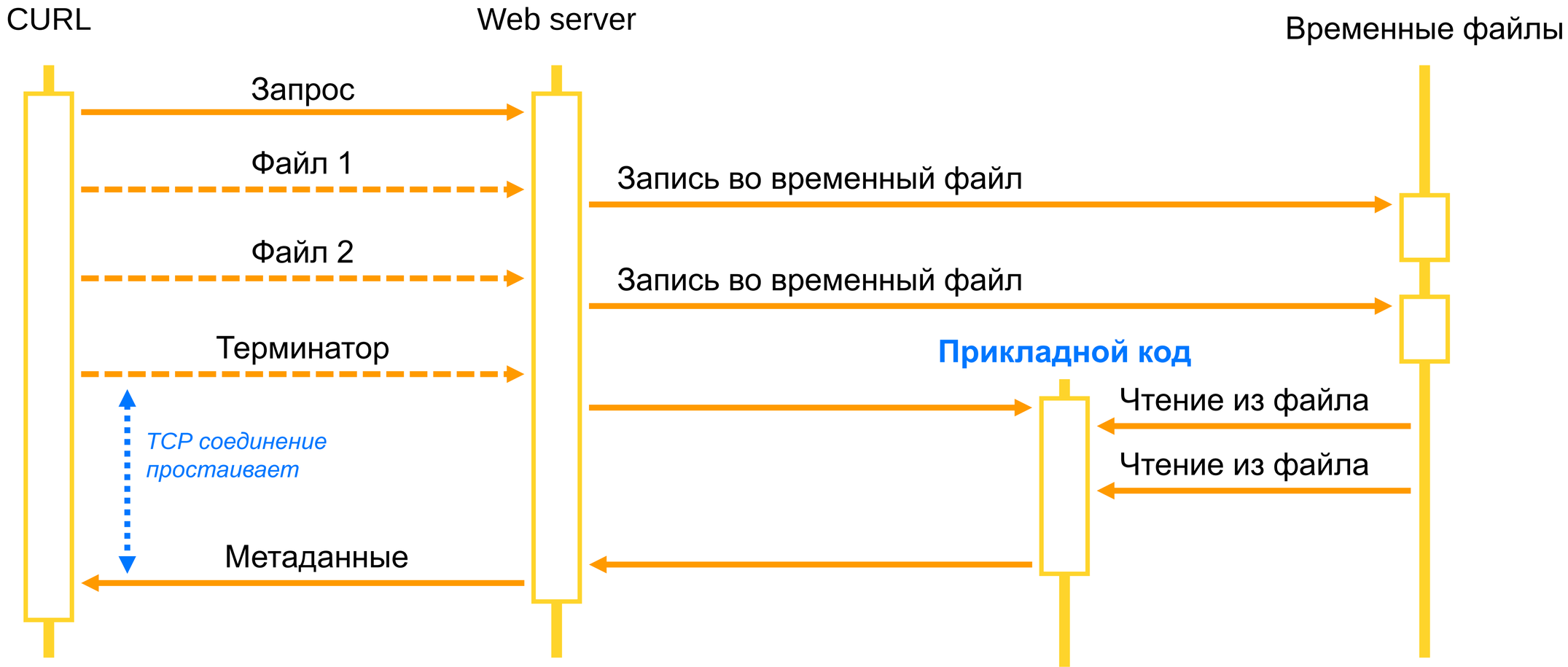
Настройка работы multipart в Spring

application.properties

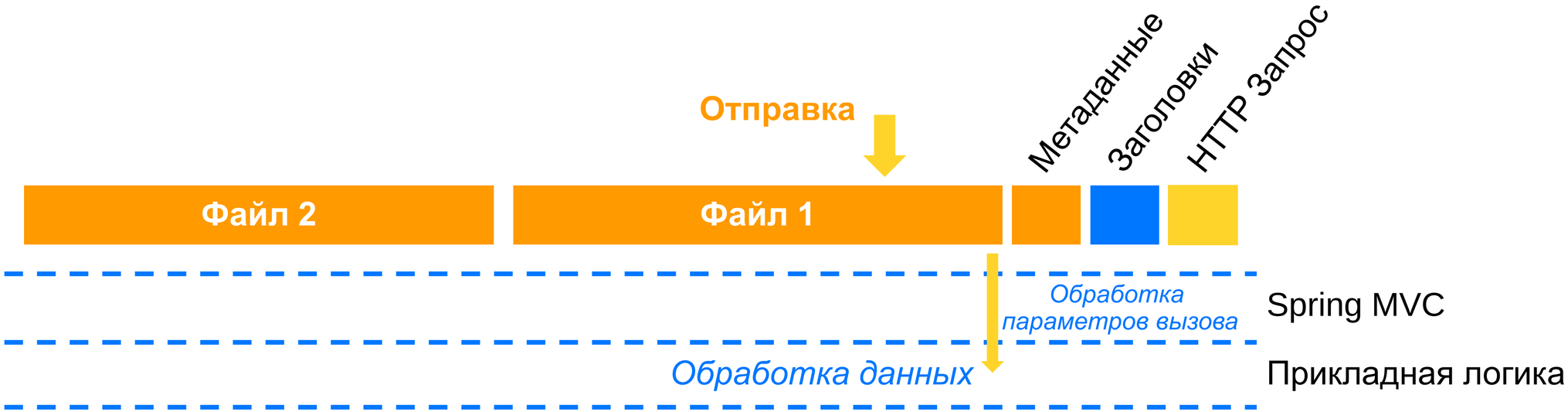
```
spring.servlet.multipart.max-file-size: 10240MB
spring.servlet.multipart.max-request-size: 10240MB


server.tomcat.max-http-form-post-size=-1
server.jetty.max-http-form-post-size=-1
~~server.undertow.max-http-form-post-size=-1~~
```

Теперь запросы проходят, но данные записываются во временные файлы и `InputStream` не доступен в `HttpServletRequest`

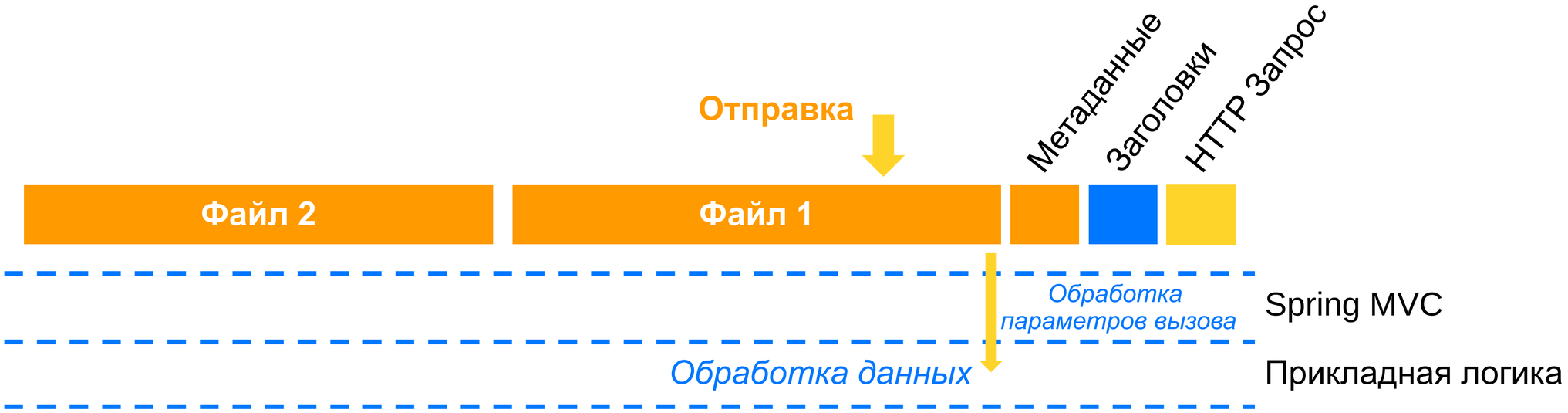
Обработка multipart в Spring



Можно ли обойтись без диска?



Можно ли обойтись без диска?

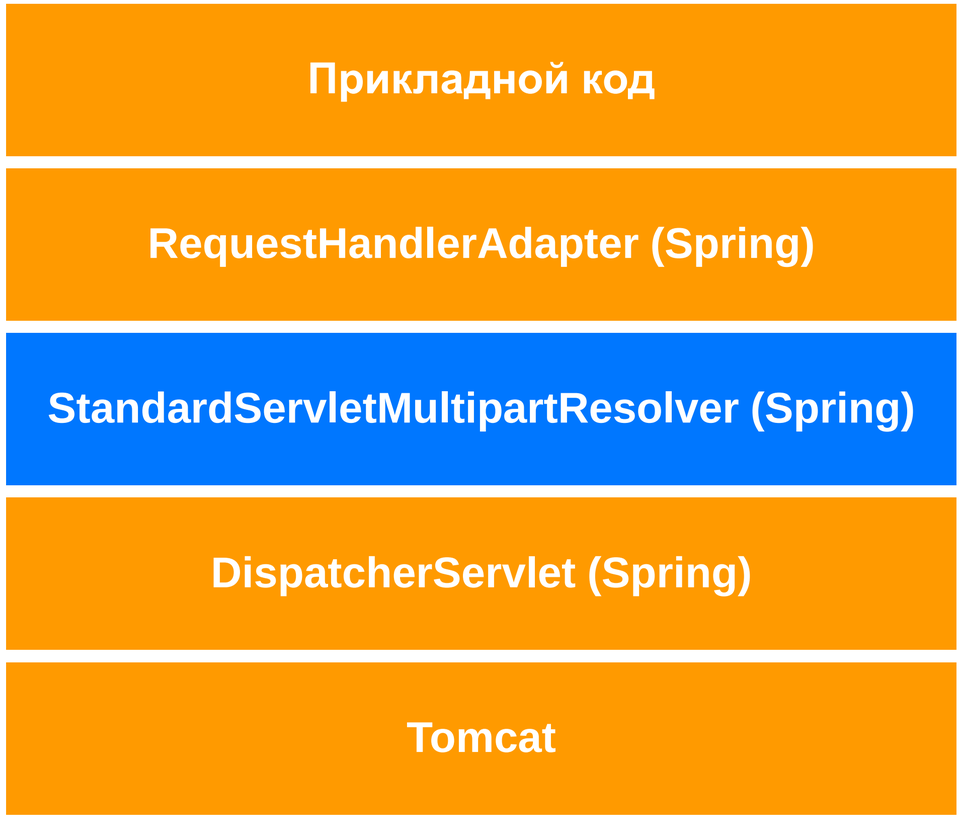


Диалектика

- Нам нужно распарсить поля для инициализации параметров хэндлера
- Нам нельзя читать данные частей “file” до входа в бизнес логику

Но части “file” на практике идут физически после полей мета данных!

Где происходит магия?



`HttpServletRequest`

↓

`StandardMultipartHttpServletRequest`

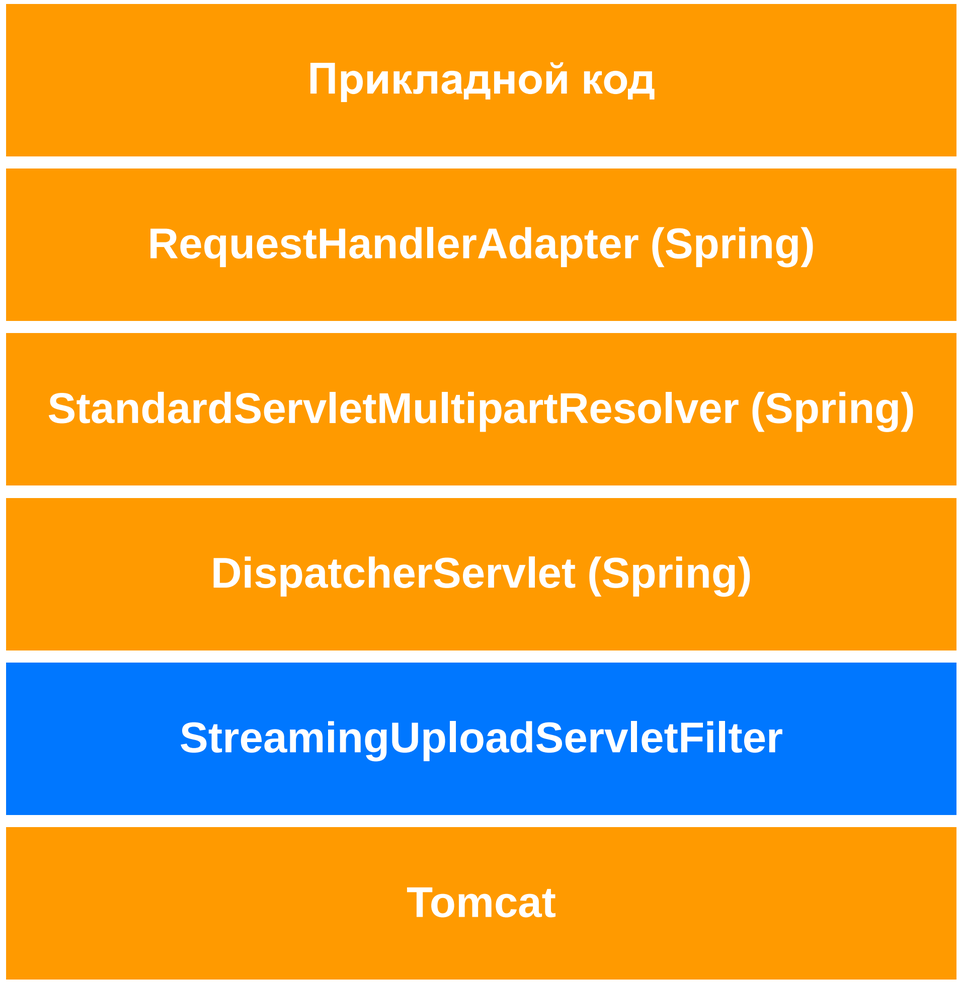
Процессинг тела запроса и
сохранение файлов на диске

Это можно отключить опций

`spring.servlet.multipart.resolve-lazily: true`

но тогда мы теряем обработку @RequestParam

Будем действовать хирургически



Оборачиваем `HttpServletRequest` в адаптер.
Адаптер блокирует чтение из потока,
при достижении первого файла.

StreamingUploadServletFilter

```
@Component @Order(1)
public class StreamingUploadServletFilter implements Filter {

    private static final String IMPORT_END_POINT = "/import";

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
                                                                    throws IOException, ServletException {
        HttpServletRequest hreq = (HttpServletRequest) request;
        ServletRequest filteredRequest = request;
        if (HttpMethod.POST.matches(hreq.getMethod())) {
            String path = hreq.getServletPath();
            if (IMPORT_END_POINT.equals(path)) {
                String type = hreq.getContentType();
                if (!type.startsWith("multipart/form-data")) {
                    ((HttpServletResponse)response).sendError(
                            HttpStatus.BAD_REQUEST.value(), "multipart/form-data content type is required");
                    return;
                }
                filteredRequest = new ImportHttpServletRequest(hreq);
            }
        }
        chain.doFilter(filteredRequest, response);
    }
}
```

Реализация

```
@PostMapping("import")
public ResponseEntity<ImportStatus> importFiles(
          ImportRequest params,
          HttpServletRequest httpRequest) throws IOException {
    // Часть тела запроса обработана
    // для заполнения ImportRequest структуры
    ImportHttpServletRequest freq = (ImportHttpServletRequest) httpRequest;

    Iterator<MultipartFile> files = freq.getUploadedFiles();
    // файлы могут быть обработаны строго последовательно
    // InputStream каждого из MultipartFile объектов проксирует данные из входного запроса

    return processFiles(params, files);
}
```

Промежуточный итог

Входящие данные по HTTP

Сервер	Клиент
✓ Потоковые данные `HttpServletRequest`	`RestTemplate` ✓
✓ Multipart Буферизация на диск Потоковая обработка	

Исходящие данные по HTTP

Сервер	Клиент
✓ `StreamingResponseBody`	`RestTemplate` ✓
✓ `HttpServletResponse`	

Запись данных в PostgreSQL

?

Чтение данных из PostgreSQL

?

Почему буферит JDBC?

Почему драйвер PG буферизирует ResultSet?

- auto commit – on
- fetch size = 0
- fetch direction ≠ ResultSet.FETCH_FORWARD

Нужно настроить JDBC Connection …

Почему буферит JDBC?

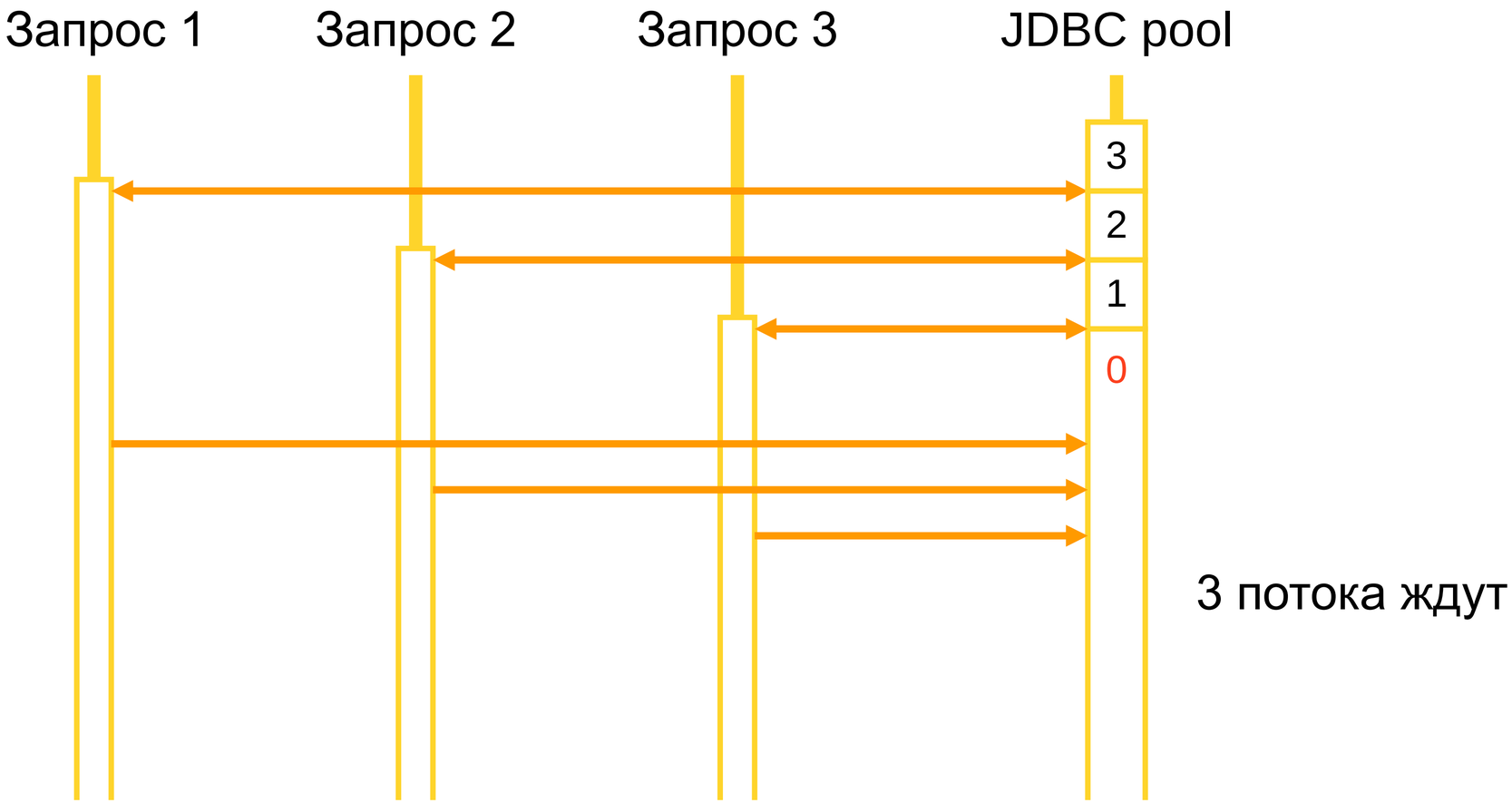
Почему драйвер PG буферизирует ResultSet?

- auto commit – on
- fetch size = 0
- fetch direction ≠ ResultSet.FETCH_FORWARD

Нужно настроить JDBC Connection …

Но у нас Spring JDBC template!

Как получить клинч на одном JDBC пуле?

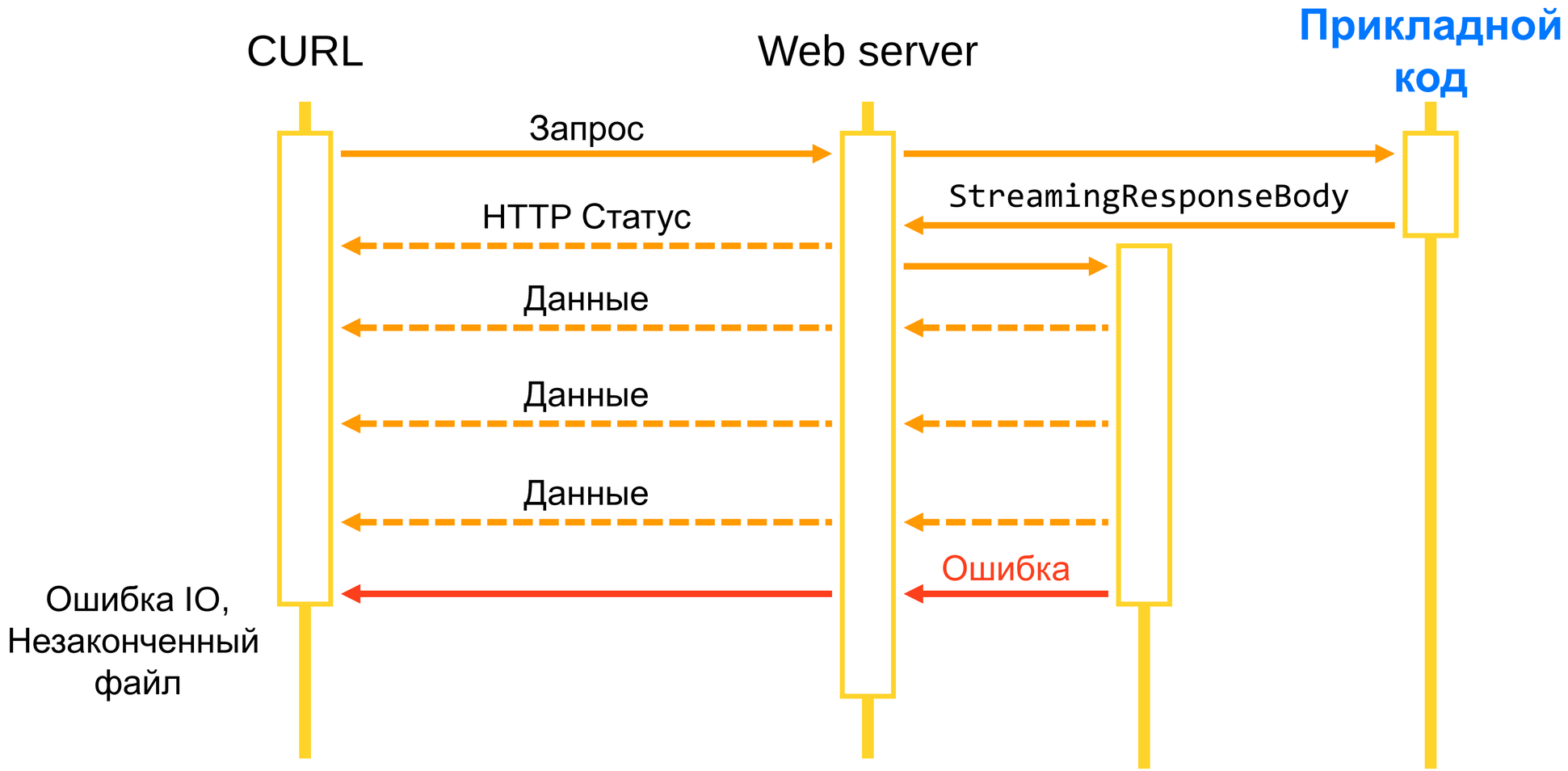


3 потока ждут

Подстраиваем connection в JDBC template

```
public void executeQuery(String query, ResultSetExtractor<?> extractor) {
    StatementCallback<Void> callback = s -> executeQuery(s, query, extractor);
    context.getStorage().getJdbc().execute(callback);
}

...

private Void executeQuery(Statement statement, String query, ResultSetExtractor<?> extractor)
                                                                        throws SQLException {
    statement.getConnection().setAutoCommit(false);
    try {
        statement.setPoolable(false);
        statement.setFetchSize(1000);
        statement.setFetchDirection(ResultSet.FETCH_FORWARD);
        extractor.extractData(statement.executeQuery(query));
    } finally {
        statement.getConnection().rollback();
        statement.getConnection().setAutoCommit(true);
    }
    return null;
}
```

Особенности обработки ошибок сервера



Окончательный итог

Входящие данные по HTTP

Сервер	Клиент
✓ Потоковые данные `HttpServletRequest`	`RestTemplate` ✓
✓ Multipart Буферизация на диск Потоковая обработка	

Исходящие данные по HTTP

Сервер	Клиент
✓ `StreamingResponseBody`	`RestTemplate` ✓
✓ `HttpServletResponse`	

Запись данных в PostgreSQL

Подводных камней не обнаружено

✓

Чтение данных из PostgreSQL

Настройка соединения и курсора

```
statement.getConnection().setAutoCommit(false);
statement.setFetchSize(…);
statement.setFetchDirection(…FETCH_FORWARD);
```

✓

Заключение

Spring вполне готов к “безразмерным” запросам!

- Нужно слегка дорабатывать напильником
- HTTP статус уходит клиенту до блока данных
- multipart имеет особые правила обработки, усложняет потоковую обработку
- “Потоковая” обработка
  – паттерн имеющий право на существование!

Спасибо

Алексей Рагозин

alexey.ragozin@gmail.com