

Javalin

Javalin

1

Kafka Connect

Data Server

Javalin Omegle

Jetty -

HTML Javalin

Javalin

Javalin REST API

WebSockets

Prometheus grafana Javalin

Javalin Vue

OpenAPI 3

influxDB

Javalin Java 10 Google Guice

JWT Javalin

Javalin Servlet

GraalVM Javalin 22MB

Hibernate ORM Javalin

Javalin

JavaEasyExcel

EasyExcel

Javalin

Javalin

Javalin

1

Javalin

<https://javalin.io/documentation#getting-started>

io.javalin javalin 6.6.0

Javalin Jackson Logback artifact id javalin-bundle javalin

```
import io.javalin.Javalin;

public class HelloWorld {
    public static void main(String[] args) {
        var app = Javalin.create(/*config*/)
            .get("/", ctx -> ctx.result("Hello World"))
            .start(7070);
    }
}
```

Javalin

- before...getpostputpatchdeleteafter headoptionstrace
connect
- //hello-world/hello/{name}
- ctx -> { ... }MyClass implements Handler

void Handlerctx.result(result)ctx.json(obj)ctx.future(future)

before/after

Before-handler

before-handler

```
app.before(ctx -> {
    // runs before all requests
});
```

```

app.before("/path/*", ctx -> {
// runs before request to /path/*
});

404 before-handlerapp.beforeMatched

app.beforeMatched(ctx -> {
// runs before all matched requests (including static files)
});

API GET POST JavalinGETPOSTPUTPATCHDELETEHEAD
OPTIONSTRACECONNECT Javalin#addHandler

```

```

app.get("/output", ctx -> {
// some code
ctx.json(object);
});

app.post("/input", ctx -> {
// some code
ctx.status(201);
});
ctx.pathParam("key")

app.get("/hello/{name}", ctx -> { // the {} syntax does not allow slashes (/) as part of the parameter
ctx.result("Hello: " + ctx.pathParam("name"));
});
app.get("/hello/", ctx -> { // the <> syntax allows slashes (/) as part of the parameter
ctx.result("Hello: " + ctx.pathParam("name"));
});

app.get("/path/*", ctx -> { // will match anything starting with /path/
ctx.result("You are here because " + ctx.path() + " matches " + ctx.matchedPath());
});
(           ) {param-name}

```

```

app.after(ctx -> {
// run after all requests
});
app.after("/path/*", ctx -> {
// runs after request to /path/*

```

```
});
404app.afterMatched

app.afterMatched(ctx -> {
// runs after all matched requests (including static files)
});
```

Context http servlet servlet getter setter

```
// Request methods
body() // request body as string
bodyAsBytes() // request body as array of bytes
bodyAsClass(clazz) // request body as specified class (deserialized from JSON)
bodyStreamAsClass(clazz) // request body as specified class (memory optimized version of above)
bodyValidator(clazz) // request body as validator typed as specified class
bodyInputStream() // the underlying input stream of the request
uploadedFile("name") // uploaded file by name
uploadedFiles("name") // all uploaded files by name
uploadedFiles() // all uploaded files as list
uploadedFileMap() // all uploaded files as a "names by files" map
formParam("name") // form parameter by name, as string
formParamAsClass("name", clazz) // form parameter by name, as validator typed as specified class
formParams("name") // list of form parameters by name
formParamMap() // map of all form parameters
pathParam("name") // path parameter by name as string
pathParamAsClass("name", clazz) // path parameter as validator typed as specified class
pathParamMap() // map of all path parameters
basicAuthCredentials() // basic auth credentials (or null if not set)
attribute("name", value) // set an attribute on the request
attribute("name") // get an attribute on the request
attributeOrCompute("name", ctx -> {}) // get an attribute or compute it based on the context if absent
attributeMap() // map of all attributes on the request
contentType() // content length of the request body
contentType() // request content type
cookie("name") // request cookie by name
cookieMap() // map of all request cookies
header("name") // request header by name (can be used with Header.HEADERNAME)
headerAsClass("name", clazz) // request header by name, as validator typed as specified class
headerMap() // map of all request headers
host() // host as string
ip() // ip as string
isMultipart() // true if the request is multipart
isMultipartFormData() // true if the request is multipart/formdata
method() // request methods (GET, POST, etc)
path() // request path
port() // request port
```

```

protocol() // request protocol
queryParam("name") // query param by name as string
queryParamAsClass("name", clazz) // query param by name, as validator typed as specified class
queryParamsAsClass("name", clazz) // query param list by name, as validator typed as list of
specified class
queryParams("name") // list of query parameters by name
queryParamMap() // map of all query parameters
queryString() // full query string
scheme() // request scheme
sessionAttribute("name", value) // set a session attribute
sessionAttribute("name") // get a session attribute
consumeSessionAttribute("name") // get a session attribute, and set value to null
cachedSessionAttribute("name", value) // set a session attribute, and cache the value as a request
attribute
cachedSessionAttribute("name") // get a session attribute, and cache the value as a request attribute
cachedSessionAttributeOrCompute(...) // same as above, but compute and set if value is absent
sessionAttributeMap() // map of all session attributes
url() // request url
fullUrl() // request url + query string
contextPath() // request context path
userAgent() // request user agent
req() // get the underlying HttpServletRequest

// Response methods
result("result") // set result stream to specified string (overwrites any previously set result)
result(byteArray) // set result stream to specified byte array (overwrites any previously set result)
result(inputStream) // set result stream to specified input stream (overwrites any previously set result)
future(futureSupplier) // set the result to be a future, see async section (overwrites any previously set
result)
writeSeekableStream(inputStream) // write content immediately as seekable stream (useful for audio
and video)
result() // get current result stream as string (if possible), and reset result stream
resultInputStream() // get current result stream
contentType("type") // set the response content type
header("name", "value") // set response header by name (can be used with Header.HEADERNAME)
redirect("/path", code) // redirect to the given path with the given status code
status(code) // set the response status code
status() // get the response status code
cookie("name", "value", maxAge) // set response cookie by name, with value and max-age (optional).
cookie(cookie) // set cookie using javalin Cookie class
removeCookie("name", "/path") // removes cookie by name and path (optional)
json(obj) // calls result(jsonString), and also sets content type to json
jsonStream(obj) // calls result(jsonStream), and also sets content type to json
html("html") // calls result(string), and also sets content type to html
render("/template.tmpl", model) // calls html(renderedTemplate)
res() // get the underlying HttpServletResponse

```

// Other methods

```

async(runnable) // lifts request out of Jetty's ThreadPool, and moves it to Javalin's AsyncThreadPool
async(asyncConfig, runnable) // same as above, but with additional config
handlerType() // handler type of the current handler (BEFORE, AFTER, GET, etc)
appData(typedKey) // get data from the Javalin instance (see app data section below)
with(pluginClass) // get context plugin by class, see plugin section below
matchedPath() // get the path that was used to match this request (ex, "/hello/{name}")
endpointHandlerPath() // get the path of the endpoint handler that was used to match this request
cookieStore() // see cookie store section below
skipRemainingHandlers() // skip all remaining handlers for this request

```

Javalin Javalin#createappData(...)Context Javalin

```

// register a custom attribute
var myKey = new Key("my-key");
var app = Javalin.create(config -> {
    config.appData(myKey, myValue);
});
// access a custom attribute
var myValue = ctx.appData(myKey); // var will be inferred to MyValue
// call a custom method on a custom attribute
ctx.appData(myKey).myMethod();

```

CookieStore

```

ctx.cookieStore().set(key, value); // store any type of value
ctx.cookieStore().get(key); // read any type of value
ctx.cookieStore().clear(); // clear the cookie-store
cookieStore

```

```

cookie cookie-store-map
ctx.attribute()
cookie-store-map base64 cookie map

```

```

serverOneApp.post("/cookie-storer", ctx -> {
    ctx.cookieStore().set("string", "Hello world!");
    ctx.cookieStore().set("i", 42);
    ctx.cookieStore().set("list", Arrays.asList("One", "Two", "Three"));
});
serverTwoApp.get("/cookie-reader", ctx -> { // runs on a different server than serverOneApp
    String string = ctx.cookieStore().get("string")
    int i = ctx.cookieStore().get("i")
    List list = ctx.cookieStore().get("list")
});
cookieget serverTwoApp postserverOneApp

```


cookie 4kb

WebSockets

Javalin WebSocket lambda

```
app.ws("/websocket/{path}", ws -> {
  ws.onConnect(ctx -> System.out.println("Connected"));
});
```

```
ws.onConnect(WsConnectContext)
ws.onError(WsErrorContext)
ws.onClose(WsCloseContext)
ws.onMessage(WsMessageContext)
ws.onBinaryMessage(WsBinaryMessageContext)
WsContext WsMessageContext.message()WsContext
```

- Javalin WebSocket

WsBefore

app.wsBefore WebSocket WebSocket

```
app.wsBefore(ws -> {
  // runs before all WebSocket requests
});
app.wsBefore("/path/*", ws -> {
  // runs before websocket requests to /path/*
});
```

WsEndpoint

WebSocket app.ws(path, handler)WebSocket

```
app.ws("/websocket/{path}", ws -> {
  ws.onConnect(ctx -> System.out.println("Connected"));
  ws.onMessage(ctx -> {
    User user = ctx.messageAsClass(User.class); // convert from json
    ctx.send(user); // convert to json and send back
  });
  ws.onBinaryMessage(ctx -> System.out.println("Message"));
  ws.onClose(ctx -> System.out.println("Closed"));
  ws.onError(ctx -> System.out.println("Errored"));
});
```

app.wsAfter WebSocket WebSocket

```
app.wsAfter(ws -> {
  // runs after all WebSocket requests
});
app.wsAfter("/path/*", ws -> {
```

```

// runs after websocket requests to /path/*
});
WsContext
WsContext websocket websocket servlet

// Session methods
send(obj) // serialize object to json string and send it to client
send("message") // send string to client
send(byteBuffer) // send bytes to client
sendAsClass(obj, clazz) // serialize object to json string and send it to client

// Upgrade Context methods (getters)
matchedPath() // get the path that was used to match this request (ex, "/hello/{name}")
host() // host as string

queryParam("name") // query param by name as string
queryParamAsClass("name", clazz) // query param parameter by name, as validator typed as
specified class
queryParams("name") // list of query parameters by name
queryParamMap() // map of all query parameters
queryString() // full query string

pathParam("name") // path parameter by name as string
pathParamAsClass("name", clazz) // path parameter as validator typed as specified class
pathParamMap() // map of all path parameters

header("name") // request header by name (can be used with Header.HEADERNAME)
headerAsClass("name", clazz) // request header by name, as validator typed as specified class
headerMap() // map of all request headers

cookie("name") // request cookie by name
cookieMap() // map of all request cookies

attribute("name", value) // set an attribute on the request
attribute("name") // get an attribute on the request
attributeMap() // map of all attributes on the request

sessionAttribute("name") // get a session attribute
sessionAttributeMap() // map of all session attributes

sendPing() // send a ping to the client
sendPing(bytes) // send a ping with data to the client
enableAutomaticPings() // enable automatic pinging to avoid timeouts
enableAutomaticPings(1, HOURS, bytes) // enable automatic pinging with custom interval and/or
data
disableAutomaticPings() // disable automatic pinging

closeSession() // close the session

```

```

closeSession(closeStatus) // close the session with a CloseStatus
closeSession(400, "reason") // close the session with a status and reason
WsMessageContext
message() // receive a string message from the client
messageAsClass(clazz) // deserialize message from client
WsBinaryMessageContext
data() // receive a byte array of data from the client
offset() // the offset of the data
length() // the length of the data
WsCloseContext
status() // the int status for why connection was closed
reason() // the string reason for why connection was closed
WsErrorContext
error() // the throwable error that occurred
WsConnectContext
WsConnectContextWsContext

apiBuilder().path().apiBuilder() Javalin app.router.ApiBuilder.get(app/router, ...)
app/router.get(...)

```

HTTP import static io.javalin.apibuilder.ApiBuilder.*

```

config.router.apiBuilder() -> {
  path("/users", () -> {
    get(UserController::getAllUsers);
    post(UserController::createUser);
    path("/{id}", () -> {
      get(UserController::getUser);
      patch(UserController::updateUser);
      delete(UserController::deleteUser);
    });
    ws("/events", UserController::webSocketEvents);
  });
  path() /
  path("api", ...)path("/api", ...)

```

```

CrudHandler
CrudHandlerapiBuilder()

```

```

config.router.apiBuilder() -> {
  crud("users/{user-id}", new UserController());
};
crud

```

```

interface CrudHandler {

```

```

getAll(ctx)
getOne(ctx, resourceId)
create(ctx)
update(ctx, resourceId)
delete(ctx, resourceId)
}

```

Javalin Validator

```

ctx.queryParamAsClass("paramName", MyClass.class) // creates a Validator for the value of
queryParam("paramName")
ctx.formParamAsClass("paramName", MyClass.class) // creates a Validator for the value of
formParam("paramName")
ctx.pathParamAsClass("paramName", MyClass.class) // creates a Validator for the value of
pathParam("paramName")
ctx.headerAsClass("headerName", MyClass.class) // creates a Validator for the value of
header("paramName")
ctx.bodyValidator(MyClass.class) // creates a Validator for the value of body()
Validator.create(clazz, value, fieldName)

```

API

```

allowNullable() // turn the Validator into a NullableValidator (must be called first)
check(predicate, "error") // add a check with a ValidationError("error") to the Validator
check(predicate, validationError) // add a check with a ValidationError to the Validator (can have args
for localization)
get() // return the validated value as the specified type, or throw ValidationException
getOrThrow(exceptionFunction) // return the validated value as the specified type, or throw custom
exception
getOrDefault() // return default-value if value is null, else call get()
errors() // get all the errors of the Validator (as map("fieldName", List))

```

// VALIDATE A SINGLE QUERY PARAMETER WITH A DEFAULT VALUE

////////////////////////////////////

```

Integer myValue = ctx.queryParamAsClass("value", Integer.class).getOrDefault(788) // validate value
ctx.result(value) // return validated value to the client
// GET ?value=a would yield HTTP 400 - {"my-qp":
[{"message":"TYPE_CONVERSION_FAILED","args":{"value":"a"}}]}
// GET ?value=1 would yield HTTP 200 - 1 (the validated value)
// GET ? would yield HTTP 200 - 788 (the default value)

```

// VALIDATE TWO DEPENDENT QUERY PARAMETERS //////////////////////////////////////

```

Instant fromDate = ctx.queryParamAsClass("from", Instant.class).get();
Instant toDate = ctx.queryParamAsClass("to", Instant.class)
.check(it -> it.isAfter(fromDate), "'to' has to be after 'from'")
.get();

```

// VALIDATE A JSON BODY //////////////////////////////////////

```

MyObject myObject = ctx.bodyValidator(MyObject.class)
    .check(obj -> obj.myObjectProperty == someValue, "THINGS_MUST_BE_EQUAL")
    .get();

// VALIDATE WITH CUSTOM VALIDATIONERROR //////////////////////////////////////
ctx.queryParamAsClass("param", Integer.class)
    .check({ it > 5 }, new ValidationError("OVER_LIMIT", Map.of("limit", 5)))
    .get();
// GET ?param=10 would yield HTTP 400 - {"param":[{"message":"OVER_LIMIT","args":
{"limit":5,"value":10}]}

Validator ageValidator = ctx.queryParamAsClass("age", Integer.class)
    .check(n -> !n.contains("-"), "ILLEGAL_CHARACTER")

// Empty map if no errors, otherwise a map with the key "age" and failed check messages in the list.
Map<String, List> errors = ageValidator.errors();

// Merges all errors from all validators in the list. Empty map if no errors exist.
Map<String, List> manyErrors = JavalinValidation.collectErrors(ageValidator, otherValidator, ...)

Validator

app.exception(ValidationException::class.java) { e, ctx ->
    ctx.json(e.errors).status(400)
}

app.exception(ValidationException.class, (e, ctx) -> {
    // your code
});

Javalin.create(config -> {
    config.validation.register(Instant.class, v -> Instant.ofEpochMilli(Long.parseLong(v)));
});

Javalin AccessManager/ Javalin 6 Javalin 5 6 beforeMatched

Javalin 6

app.beforeMatched(ctx -> {
    var userRole = getUserRole(ctx); // some user defined function that returns a user role
    if (!ctx.routeRoles().contains(userRole)) { // routeRoles are provided through the Context interface
        throw new UnauthorizedResponse(); // request will have to be explicitly stopped by throwing an
        exception
    }
});

```

```
app.get("/public", ctx -> ctx.result("Hello public"), Role.OPEN);
app.get("/private", ctx -> ctx.result("Hello private"), Role.LOGGED_IN);
```

Javalin `HttpResponseException` JSON JSON

```
app.post("/") { throw new ForbiddenResponse("Off limits!") }
JSON
```

```
{
  "title": "Off limits!",
  "status": 403,
  "type": "https://javalin.io/documentation#forbiddenresponse",
  "details": []
}
```

Forbidden
 Map<String, String>

302 FoundRedirected

400 Bad RequestBad request

401 UnauthorizedUnauthorized

403 ForbiddenForbidden

NotFoundResponse
 404 Not FoundNot found

405 Method not allowed

409 Conflict

GoneResponse
 410 GoneGone

500 Internal server error

502 Bad GatewayBad gateway

503 Service unavailable

504 Gateway timeout

beforeendpointafterwsException Exceptionapp.exception()app.wsException()

```
// HTTP exceptions
app.exception(NullPointerException.class, (e, ctx) -> {
// handle nullpointers here
});

app.exception(Exception.class, (e, ctx) -> {
// handle general exceptions here
// will not trigger if more specific exception-mapper found
});

// WebSocket exceptions
app.wsException(NullPointerException.class, (e, ctx) -> {
// handle nullpointers here
});

app.wsException(Exception.class, (e, ctx) -> {
// handle general exceptions here
// will not trigger if more specific exception-mapper found
});
```

HTTP HTTP

```
app.error(404, ctx -> {
ctx.result("Generic 404 message");
});
```

```
app.exception(FileNotFoundException.class, (e, ctx) -> {
ctx.status(404);
}).error(404, ctx -> {
ctx.result("Generic 404 message");
});
```

```
app.error(404, "html", ctx -> {
ctx.html("Generic 404 message");
});
```

HTML JSON

Javalin app.sse()SseClient

```
app.sse("/sse", client -> {
    client.sendEvent("connected", "Hello, SSE");
    client.onClose() -> System.out.println("Client disconnected");
    client.close(); // close the client
});
client.keepAlive()
```

Queue clients = new ConcurrentLinkedQueue();

```
app.sse("/sse", client -> {
    client.keepAlive();
    client.onClose() -> clients.remove(client));
    clients.add(client);
});
```

SseClient API

```
sendEvent("myMessage") // calls emit("message", "myMessage", nold)
sendEvent("eventName", "myMessage") // calls emit("eventName", "myMessage", nold)
sendEvent("eventName", "myMessage", "id") // calls emit("eventName", "myMessage", "id")
sendComment("myComment") // calls emit("myComment")
onClose(runnable) // callback which runs when a client closes its connection
keepAlive() // keeps the connection alive. useful if you want to keep a list of clients to broadcast to.
close() // closes the connection
terminated() // returns true if the connection has been closed
ctx // the Context from when the client connected (to fetch query-params, etc)
```

Javalin Javalin

```
Javalin.create(config -> {
    config.http // The http layer configuration: etags, request size, timeout, etc
    config.router // The routing configuration: context path, slash treatment, etc
    config.jetty // The embedded Jetty webserver configuration
    config.staticFiles // Static files and webjars configuration
    config.spaRoot = // Single Page Application roots configuration
    config.requestLogger // Request Logger configuration: http and websocket loggers
    config.bundledPlugins // Bundled plugins configuration: enable bundled plugins or add custom ones
    config.events // Events configuration
    config.vue // Vue Plugin configuration
    config.contextResolver // Context resolver implementation configuration
    config.validation // Default validator configuration
    config.useVirtualThreads // Use virtual threads (based on Java Project Loom)
    config.showJavalinBanner // Show the Javalin banner in the logs
    config.startupWatcherEnabled // Print warning if instance was not started after 5 seconds
```


config.pvt // This is “private”, only use it if you know what you’re doing

```
config.events(listenerConfig) // Add an event listener
config.jsonMapper(jsonMapper) // Set a custom JsonMapper
config.fileRenderer(fileRenderer) // Set a custom FileRenderer
config.registerPlugin(plugin) // Register a plugin
config.appData(key, data) // Store data on the Javalin instance
```

});

Http

```
Javalin.create(config -> {
config.http.generateEtags = booleanValue; // if javalin should generate etags for dynamic responses
(not static files)
config.http.prefer405over404 = booleanValue; // return 405 instead of 404 if path is mapped to
different HTTP method
config.http.maxRequestSize = longValue; // the max size of request body that can be accessed
without using using an InputStream
config.http.responseBufferSize = longValue; // the size of the response buffer (default 32kb)
config.http.defaultContentType = stringValue; // the default content type
config.http.asyncTimeout = longValue; // timeout in milliseconds for async requests (0 means no
timeout)
config.http.strictContentTypes = booleanValue; // throw exception if e.g content-type is
missing/incorrect when attempting to parse JSON
```

```
config.http.customCompression(strategy); // set a custom compression strategy
config.http.brotliAndGzipCompression(lvl, lvl); // enable brotli and gzip compression with the specified levels
config.http.gzipOnlyCompression(lvl); // enable gzip compression with the specified level
config.http.brotliOnlyCompression(lvl); // enable brotli compression with the specified level
config.http.disableCompression(); // disable compression
```

});

ContextContextResolvers

```
Javalin.create(config -> {
config.contextResolver.ip = ctx -> “custom ip”; // called by Context#ip()
config.contextResolver.host = ctx -> “custom host”; // called by Context#host()
config.contextResolver.scheme = ctx -> “custom scheme”; // called by Context#scheme()
config.contextResolver.url = ctx -> “custom url”; // called by Context#url()
config.contextResolver.fullUrl = ctx -> “custom fullUrl”; // called by Context#fullUrl()
```

```
});
Jetty
Javalin.create(config -> {
config.jetty.defaultHost = "localhost"; // set the default host for Jetty
config.jetty.defaultPort = 1234; // set the default port for Jetty
config.jetty.threadPool = new ThreadPool(); // set the thread pool for Jetty
config.jetty.timeoutStatus = 408; // set the timeout status for Jetty (default 500)
config.jetty.clientAbortStatus = 499; // set the abort status for Jetty (default 500)
config.jetty.multipartConfig = new MultipartConfig(); // set the multipart config for Jetty
config.jetty.modifyJettyWebSocketServletFactory(factory -> {}); // modify the
JettyWebSocketServletFactory
config.jetty.modifyServer(server -> {}); // modify the Jetty Server
config.jetty.modifyServletContextHandler(handler -> {}); // modify the ServletContextHandler (you can
set a SessionHandler here)
config.jetty.modifyHttpConfiguration(httpConfig -> {}); // modify the HttpConfiguration
config.jetty.addConnector((server, httpConfig) -> new ServerConnector(server)); // add a connector to
the Jetty Server
});
```

ctrl+f FileUploadConfig
Javalin Servlet

```
Javalin.create(config -> {
config.jetty.multipartConfig.cacheDirectory("c:/temp"); //where to write files that exceed the in memory
limit
config.jetty.multipartConfig.maxFileSize(100, SizeUnit.MB); //the maximum individual file size allowed
config.jetty.multipartConfig.maxInMemoryFileSize(10, SizeUnit.MB); //the maximum file size to handle
in memory
config.jetty.multipartConfig.maxTotalRequestSize(1, SizeUnit.GB); //the maximum size of the entire
multipart request
});
```

HTTP config.requestLogger.http()Context

```
Javalin.create(config -> {
config.requestLogger.http((ctx, ms) -> {
// log things here
});
});
WebSocket config.requestLogger.ws()app.ws().onMessageonConnectonError
onClose
```

```
Javalin.create(config -> {
config.requestLogger.ws(ws -> {
ws.onMessage(ctx -> {
```

```

System.out.println("Received: " + ctx.message());
});
});
});
WebSocket

```

```

Javalin.create(config -> {
config.router.contextPath = stringValue; // the context path (ex '/blog' if you are hosting an app on a
subpath, like 'mydomain.com/blog')
config.router.ignoreTrailingSlashes = booleanValue; // treat '/path' and '/path/' as the same path
config.router.treatMultipleSlashesAsSingleSlash = booleanValue; // treat '/path//subpath' and
'/path/subpath' as the same path
config.router.caseInsensitiveRoutes = booleanValue; // treat '/PATH' and '/path' as the same path
});
SpaRoot
(SPA) 404 404

```

```

config.spaRoot.addFile("/root", "/path/to/file.html") / config.spaRoot.addFile("/root",
"/path/to/file.html", Location.EXTERNAL)

```

Handler

```

config.spaRoot.addHandler("/root", ctx -> {
ctx.html(...);
});

```

```

config.staticFiles.add("/directory", location) GET

```

```

run before-handlers
run endpoint-handlers
if no endpoint-handler found
run static-file-handlers
if static-file-found
static-file-handler sends response
else
response is 404
run after-handlers
Javalin StaticFileConfig

```

```

Javalin.create(config -> {
config.staticFiles.add(staticFiles -> {
staticFiles.hostedPath = "/"; // change to host files on a subpath, like '/assets'
staticFiles.directory = "/public"; // the directory where your files are located
staticFiles.location = Location.CLASSPATH; // Location.CLASSPATH (jar) or Location.EXTERNAL
(file system)

```

```

staticFiles.precompress = false; // if the files should be pre-compressed and cached in memory
(optimization)
staticFiles.aliasCheck = null; // you can configure this to enable symlinks (=
ContextHandler.ApproveAliases())
staticFiles.headers = Map.of(...); // headers that will be set for the files
staticFiles.skipFileFunction = req -> false; // you can use this to skip certain files in the dir, based on
the HttpServletRequest
staticFiles.mimetypes.add(mimeType, ext); // you can add custom mimetypes for extensions
});
});
config.staticFiles.add(...)

```

WebJars config.staticFiles.enableWebJars() /webjars/name/version/file.extWebJars
<https://www.webjars.org/> NPM WebJars

SPASpaRootConfig

Javalin Java /

```

org.slf4j slf4j-simple 2.0.17 Javalin Jettystart()stop
Javalin app = Javalin.create()
.start() // start server (sync/blocking)
.stop() // stop server (sync/blocking)
app.start() app.stop()

```

```

Runtime.getRuntime().addShutdownHook(new Thread(() -> {
app.stop();
}));

```

```

app.events(event -> {
event.serverStopping(() -> { /* Your code here */ });
event.serverStopped(() -> { /* Your code here */ });
});
modifyServer

```

```

Javalin.create(config -> {
config.jetty.modifyServer(server -> server.setStopTimeout(5_000)); // wait 5 seconds for existing
requests to finish
});

```

Javalin#startIP

```

Javalin.create().start("127.0.0.1", 1235)
SessionHandler

```

jetty Javalin

```
StatisticsHandler statisticsHandler = new StatisticsHandler();
```

```
Javalin.create(config -> {
  config.server() -> {
    Server server = new Server();
    server.setHandler(statisticsHandler);
    return server;
  }
});
```

SSL/HTTP2

Javalin SSL <https://javalin.io/plugins/ssl-helpers> SSL HTTP2/3 Jetty

Jetty SSL HTTP2

HelloWorldSecure SSL

HTTP2 repo Kotlin Java javalin-http2-example

Javalin /

```
Javalin app = Javalin.create().events(event -> {
  event.serverStarting() -> { ... });
  event.serverStarted() -> { ... });
  event.serverStartFailed() -> { ... });
  event.serverStopping() -> { ... });
  event.serverStopped() -> { ... });
  event.handlerAdded(handlerMetaInfo -> { ... });
  event.wsHandlerAdded(wsHandlerMetaInfo -> { ... });
});
```

```
app.start() // serverStarting -> (serverStarted || serverStartFailed)
app.stop() // serverStopping -> serverStopped
```

PluginJavalin Javalin

Javalin

Javalin#before // runs first, can throw exception (which will skip any endpoint handlers)
 Javalin#get/post/patch/etc // runs second, can throw exception
 Javalin#error // runs third, can throw exception
 Javalin#after // runs fourth, can throw exception
 Javalin#exception // runs any time a handler throws (cannot throw exception)
 JavalinConfig#requestLogger // runs after response is written to client
 JavalinConfig#accessManager // wraps all your endpoint handlers in a lambda of your choice

Javalin Handler

```
app.get("/", ctx -> {
  NaiveRateLimit.requestPerTimeUnit(ctx, 5, TimeUnit.MINUTES); // throws if rate limit is exceeded
  ctx.status("Hello, rate-limited World!");
});
```

// you can overwrite the key-function:

RateLimitUtil.keyFunction = ctx -> // uses (ip+method+endpointPath) by default

Jetty 11 Android Javalin 5+ Javalin 4
 Javalin 4 Android Jetty 11+

JRE Loom newVirtualThreadPerTaskExecutorJavalin enableVirtualThreads
 QueuedThreadPool 250

Handler Javalin 250 1

Loom Javalin Loomylin

ctx.future()ctx.async()

Jetty

WebSocket

WebSocket TCP Javalin WebSocket

WebSocket

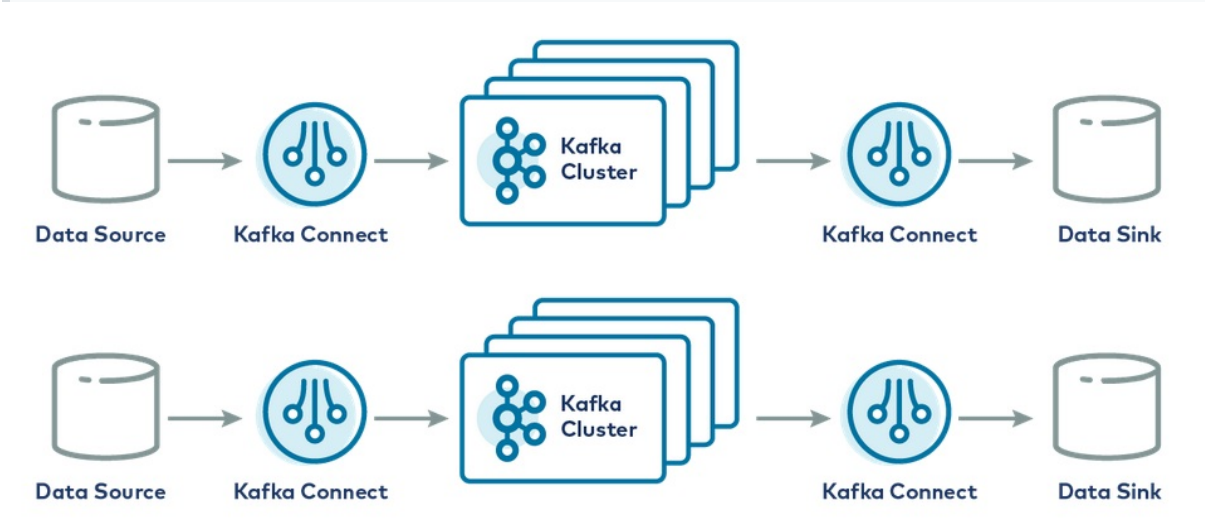
Javalin Javalin /tutorials/testing /UI/

Kafka Connect

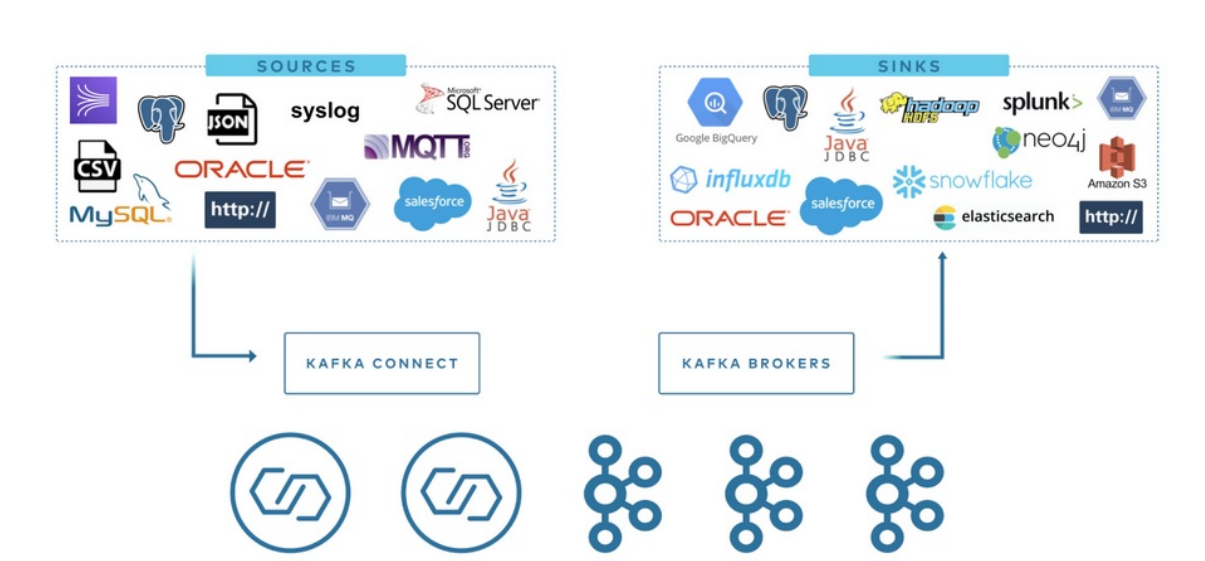
Kafka Connect

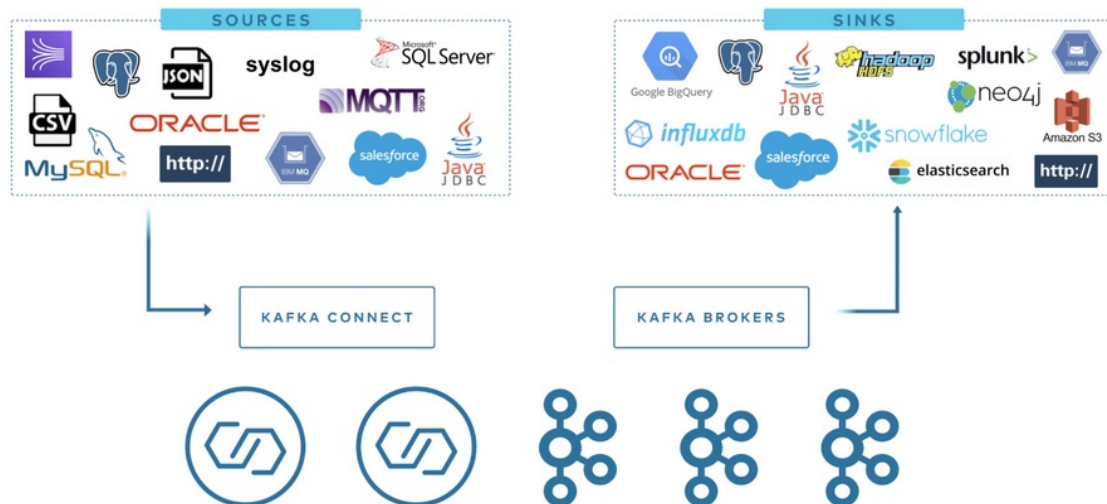
Kafka Connect

Kafka Connect Apache Kafka Kafka Kafka Connect Kafka, Kafka Connect Kafka Source Connector, Sink ConnectorSource Connector Sink Connector Kafka BrokerSource Connector Kafka ConnectSink Connector Kafka Connect



Source Connector ☐ Kafka ConnectSink Connector ☐ Kafka Connect





Kafka Connector Kafka Connect Schema Connector
Kafka

Kafka data pipeline

- Kafka Databend Mysql Kafka
- Elasticsearch Kafka Kafka Elasticsearch Kafka

Kafka Connect

- Source Connect Kafka
- Sink Connect Kafka

Kafka [Confluent Hub](#) Connector [Elasticsearch Service Sink Connector](#), [Amazon Sink Connector](#), [HDFS Sink Connector](#) Kafka

Kafka Connect ROI

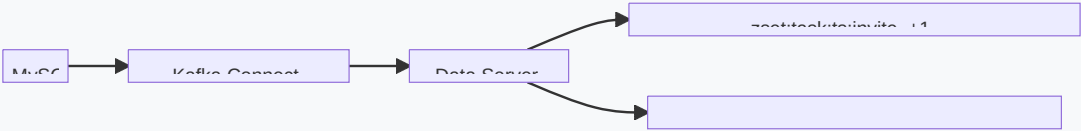
Kafka Connect

- 1 Kafka Connect MySQL topic
- 2Data Server kafka-clients

```
<!-- https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients -->
<dependency>
  <groupId>org.apache.kafka</groupId>
```

```
<artifactId>kafka-clients</artifactId>  
<version>3.8.0</version>  
</dependency>
```

3Data Server



- 4

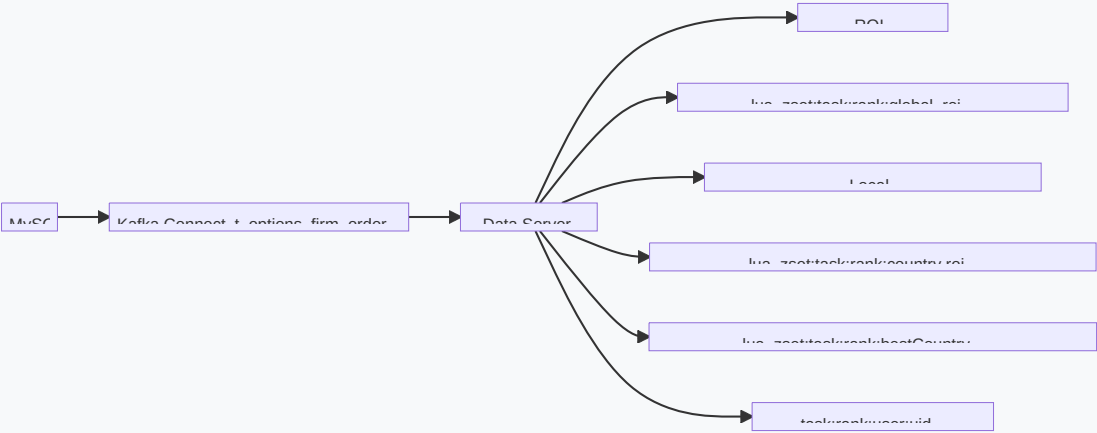
Kafka Connect ROI

ROI

- 1 Kafka Connect MySQL topic
- 2Data Server kafka-clients

```
<!-- https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients -->
<dependency>
  <groupId>org.apache.kafka</groupId>
  <artifactId>kafka-clients</artifactId>
  <version>3.8.0</version>
</dependency>
```

3Data Server ROI = profit_loss/(price*quantity)



- 4

Data Server

Data Server

Redis

```
protected JedisPooled provideRedis() {
    var poolConfig = new GenericObjectPoolConfig<Connection>();
    //
    poolConfig.setMaxWaitMillis(config.getMaxWaitMillis());
    //
    poolConfig.setMaxTotal(config.getMaxTotal());
    //
    poolConfig.setMaxIdle(config.getMaxIdle());
    //
    poolConfig.setMinIdle(config.getMinIdle());
    // ( PoolableObjectFactory.validateObject() )
    poolConfig.setTestOnBorrow(config.isTestOnBorrow());
    // ( PoolableObjectFactory.validateObject() )
    poolConfig.setTestOnReturn(config.isTestOnReturn());
    //
    poolConfig.setTestWhileIdle(config.isTestWhileIdle());
    //
    poolConfig.setMinEvictableIdleTimeMillis(config.getMinEvictableIdleTimeMillis());
    //
    poolConfig.setTimeBetweenEvictionRunsMillis(config.getTimeBetweenEvictionRunsMillis());
    //
    poolConfig.setNumTestsPerEvictionRun(-1);

    var clientConfig = DefaultJedisClientConfig.builder()
        .user(config.getUser())
        .password(config.getPassword())
        .database(config.getDatabase())
        .build();
    var hostPort = new HostAndPort(config.getAddress(), config.getPort());
    return new JedisPooled(poolConfig, hostPort, clientConfig);
}
```

- 1

- 2

Hikari

```
private DataSource createDataSource(JdbcConfig conf) {
    var config = new HikariConfig();
    config.setJdbcUrl(conf.getUrl());
    config.setUsername(conf.getUser());
    config.setPassword(conf.getPass());
    //config.setThreadFactory(Thread.ofVirtual().factory());
    config.setConnectionTestQuery(conf.getConnectionTestQuery());
    config.setMaxLifetime(conf.getMaxLifetime());
    config.setMaximumPoolSize(conf.getMaxPoolSize());
    config.setMinimumIdle(conf.getMinIdle());
    config.setIdleTimeout(conf.getIdleTimeout());
    return new HikariDataSource(config);
}
```

-
-

1

```
"" inviteDao.aggInvites Const.INVITE_LEVELS
```

```
public void run() {
    var nowTs = DateUtil.current();
    var lastTs = redisDao.getLong(RedisKey.inviteTaskTs(), null);

    var result = inviteDao.aggInvites(lastTs);
    var levels = Const.INVITE_LEVELS;
    for (var it : result.entrySet()) {
        var inviteCount = it.getValue().getCount();
        for (var n : levels) {
            if (inviteCount < n) continue;
            //n, ,
            achievementDao.add(it.getKey(), Const.ACH_INVITE, n.toString(), inviteCount.toString());
            break;
        }
    }

    redisDao.setLong(RedisKey.inviteTaskTs(), nowTs);
```



```
}
```

2ROI

```
roiRank24h t_options_firm_ordert_user
```

```
SELECT * FROM
(
    SELECT
        {0}.id,
        {0}.user_id uid,
        {1}.user_name un,
        {1}.country ct,
        (profit_loss/(price*quantity)) roi,
        @rn := IF(@prev = user_id, @rn + 1, 1) AS rn,
        @prev := user_id
    FROM {0}
    JOIN (SELECT @prev := NULL, @rn := 0) AS vars
    LEFT JOIN {1} on {0}.user_id = {1}.id
    WHERE {2}
    ORDER BY {0}.user_id, roi desc
) AS T1
WHERE rn = 1
order by roi desc
```

```
roiRank24h cacheRoiRank24hInRedis
```

-
- Redis
-

```
//
public void run() {
    var nowTs = DateUtil.current();
    var ranks = orderDao.roiRank24h(nowTs-Const.DAY_MS, config.getBaseTo
ken(), config.getQuoteToken());
    cacheDao.cacheRoiRank24hInRedis(ranks, 200);
}

//
public void cacheRoiRank24hInRedis(List<RankingVo> ranks, int limit) {
    var countries = countryDao.all();
    var counter = new Counter();
    var countryMap = new HashMap<String, List<RankingVo>>();
    var bested = new HashSet<String>();
```

```

var best = new ArrayList<RankingVo>();
var global = new ArrayList<RankingVo>();
for (var i = 0; i < ranks.size(); i++) {
    var it = ranks.get(i);
    it.setGr(i+1); //
    if(i < limit) global.add(it);

    var isWhite = countries.containsKey(it.getCt());
    var country = isWhite ? it.getCt() : "Local";

    var countryRank = counter.incr(country);
    it.setLr(countryRank); //
    if(!countryMap.containsKey(country)) {
        countryMap.put(country, new ArrayList<>());
    }

    var local = countryMap.get(country);
    if(local.size() <= limit) local.add(it);

    //best country
    if(isWhite && !bested.contains(country)) {
        bested.add(country);
        best.add(it);
    }

    worker.submit(() -> {
        redisDao.setJson(RedisKey.userRank(it.getUid()), it, config.
getRankExpiresSec(), false);
    });
}

redisDao.setJson(RedisKey.globalRank(), global, config.getRankExpire
sSec(), true);
redisDao.setJson(RedisKey.bestCountry(), best, config.getRankExpires
Sec(), true);
countryMap.forEach((name, rank) -> {
    redisDao.setJson(RedisKey.countryRank(name), rank, config.getRan
kExpiresSec(), true);
});
}

```

zset

- Kafka Connect MySQL Kafka MySQL >> Kafka >> Redis
- RocketMQ Connect MySQL RocketMQ MySQL >> Kafka >> Redis
- Canal MySQL binlog MySQL >> Canal >> Redis

```
redisDao.setJson() redis set json
```

- 50 >> >> >> top50, zset zset top50,
- zset

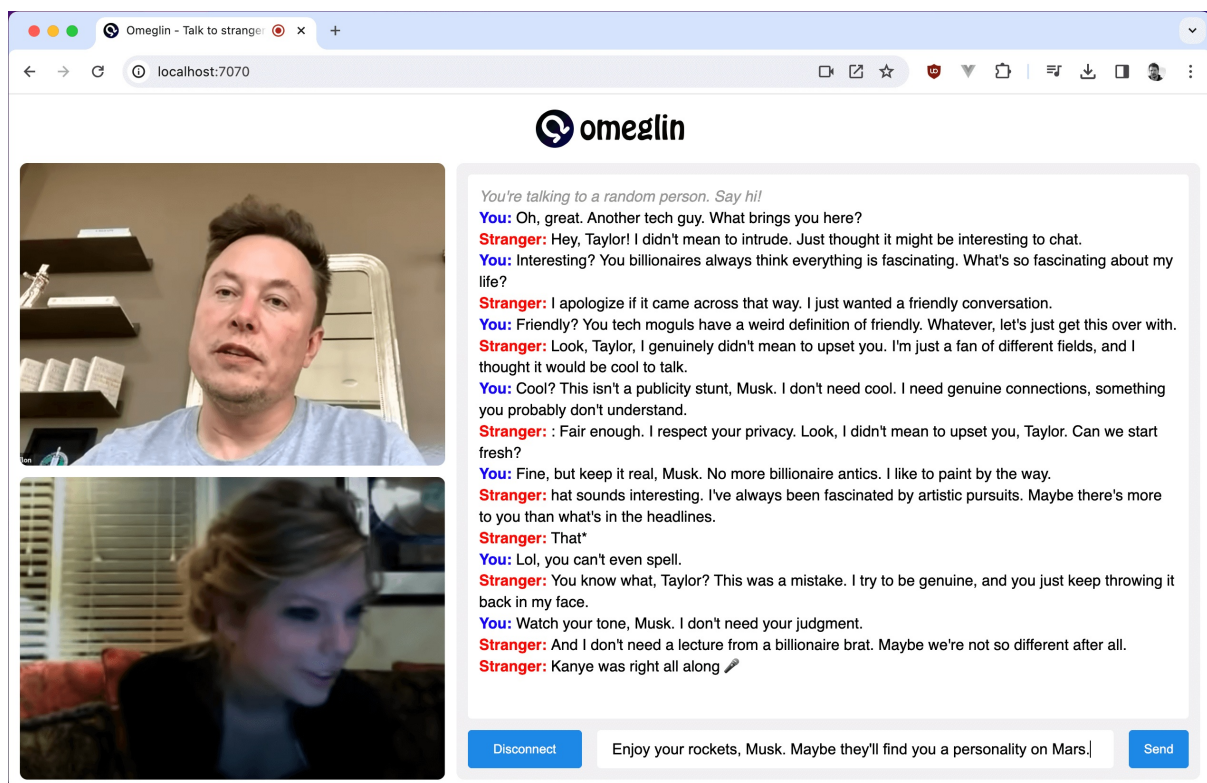
Javalin Omegle

Javalin Omegle

20231230 • David Åse25-50

GitHub fork/clone

Javalin Omegle Omegle JavaScript WebRTC
JavaScript 80-120 Kotlin Java



Maven Gradle Bazel Javalin JavalinJettyJackson Logback

```
<dependency>
  <groupId>io.javalin</groupId>
  <artifactId>javalin-bundle</artifactId>
```

```
<version>6.6.0</version>
</dependency>
```

```
src
├── main
│   ├── java/kotlin
│   │   ├── io
│   │   │   ├── javalin
│   │   │   └── omeglin
│   │   │       ├── OmeglinMain.kt/java // main class
│   │   │       └── Matchmaker.kt/java  // matchmaking logic
│   └── resources
│       ├── public
│       │   ├── index.html // html for the frontend
│       │   ├── js
│       │   │   ├── app.js // main class
│       │   │   ├── peer-connection.js // webrtc logic
│       │   │   └── chat.js // chat logic
│       └── style.css // styling
```

WebRTC WebSocket

```
package io.javalin.omeglin;

import io.javalin.Javalin;
import io.javalin.http.staticfiles.Location;

public class OmeglinMain {
    public static void main(String[] args) {
        Javalin.create(config -> {
            config.staticFiles.add("src/main/resources/public", Location
.EXTERNAL);
            config.router.mount(router -> {
                router.ws("/api/matchmaking", Matchmaking::websocket);
            });
        }).start(7070);
    }
}
```

Loation.EXTERNAL websocket WebRTC 7070

WebSocket ExchangeExchange SDP
Exchange

websocket WebSocket

- **onConnect** ping
 - **onClose** pairingAbort
 - **onMessage** "PAIRING_START" "PAIRING_ABORT" "PAIRING_DONE"
- WebRTC SDP SDP

WsContext SDP doneCount otherUser

WebSocket namedata

```
package io.javalin.omeglin;

import io.javalin.websocket.WsContext;
import io.javalin.websocket.WsConfig;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.util.concurrent.ConcurrentLinkedQueue;

public class Matchmaking {
    private static final Logger logger = LoggerFactory.getLogger(Matchmaking.class);
    private static final ConcurrentLinkedQueue<Exchange> queue = new ConcurrentLinkedQueue<>();

    public static void websocket(WsConfig ws) {
        ws.onConnect(user -> user.enableAutomaticPings());
        ws.onClose(user -> pairingAbort(user));
        ws.onMessage(user -> {
            logger.info("Received message: " + user.message());
            var message = user.messageAsClass(Message.class);
            switch (message.name()) {
```

```

        case "PAIRING_START" -> pairingStart(user);
        case "PAIRING_ABORT" -> pairingAbort(user);
        case "PAIRING_DONE" -> pairingDone(user);
        case "SDP_OFFER", "SDP_ANSWER", "SDP_ICE_CANDIDATE" -> {
            var exchange = findExchange(user);
            if (exchange != null && exchange.a != null && exchange.b != null) {
                send(exchange.otherUser(user), message); // forward message to other user
            } else {
                logger.warn("Received SDP message from unpaired user");
            }
        }
    });
}

private static void pairingStart(WsContext user) {
    queue.removeIf(ex -> ex.a == user || ex.b == user); // prevent double queueing
    var exchange = queue.stream()
        .filter(ex -> ex.b == null)
        .findFirst()
        .orElse(null);
    if (exchange != null) {
        exchange.b = user;
        send(exchange.a, new Message("PARTNER_FOUND", "GO_FIRST"));
        send(exchange.b, new Message("PARTNER_FOUND"));
    } else {
        queue.add(new Exchange(user));
    }
}

private static void pairingAbort(WsContext user) {
    var exchange = findExchange(user);
    if (exchange != null) {
        send(exchange.otherUser(user), new Message("PARTNER_LEFT"));
        queue.remove(exchange);
    }
}

private static void pairingDone(WsContext user) {
    var exchange = findExchange(user);
    if (exchange != null) {
        exchange.doneCount++;
    }
    queue.removeIf(ex -> ex.doneCount == 2);
}

```



```

    }

    private static Exchange findExchange(WsContext user) {
        return queue.stream()
            .filter(ex -> user.equals(ex.a) || user.equals(ex.b))
            .findFirst()
            .orElse(null);
    }

    private static void send(WsContext user, Message message) { // null
safe send method
        if (user != null) {
            user.send(message);
        }
    }

    record Message(String name, String data) {
        public Message(String name) {
            this(name, null);
        }
    }

    static class Exchange {
        public WsContext a;
        public WsContext b;
        public int doneCount = 0;

        public Exchange(WsContext a) {
            this.a = a;
        }

        public WsContext otherUser(WsContext user) {
            return user.equals(a) ? b : a;
        }
    }
}

```

JavaScript WebRTC CSS

- index.html HTML UI
- style.css CSS

- app.js JavaScript
- peer-connection.js WebRTC SDP
- chat.js UI

.html

HTML

- video
- button
- /

styles.css ID javascript

app.js JavaScript import JavaScript javascript

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Omeglin - Talk to strangers</title>
  <link href="/favicon.ico" type="image/svg+xml" rel="icon">
  <link rel="stylesheet" href="/styles.css">
</head>
<body>
<header>
  
</header>
<main>
  <div class="video-panel">
    <section class="remote">
      
      <video id="remoteVideo" autoplay playsinline></video>
    </section>
    <section class="local">
      <video id="localVideo" autoplay playsinline muted></video>
    </section>
  </div>
  <div class="chat-panel">
    <div class="chat-log" id="chatLog"></div>
    <div class="chat-controls">
      <div class="pairing">
        <button id="startPairing">Find stranger</button>
        <button id="abortPairing">Cancel</button>
      </div>
    </div>
  </div>
</main>
</body>
</html>
```

```

        <button id="leavePairing">Disconnect</button>
      </div>
      <div class="messaging">
        <input type="text" placeholder="Type a message..." id="chatInput">
        <button id="chatSend">Send</button>
      </div>
    </div>
  </main>
<script type="module" src="/js/app.js"></script>
</body>
</html>

```

App.js

app.js JavaScript peer-connection.js UI chat.js

```

import {Chat} from './chat.js';
import {PeerConnection} from './peer-connection.js';

const peerConnection = new PeerConnection({
  onLocalMedia: stream => document.getElementById("localVideo").srcObject = stream,
  onRemoteMedia: stream => document.getElementById("remoteVideo").srcObject = stream,
  onChatMessage: message => chat.addRemoteMessage(message),
  onStateChange: state => {
    document.body.dataset.state = state;
    chat.updateUi(state);
  }
});

let chat = new Chat(peerConnection);

document.getElementById("startPairing").addEventListener("click", async () => {
  peerConnection.setState("CONNECTING");
  peerConnection.sdpExchange.send(JSON.stringify({name: "PAIRING_START"}));
});

document.getElementById("abortPairing").addEventListener("click", () => {
  peerConnection.sdpExchange.send(JSON.stringify({name: "PAIRING_ABORT"}));
  peerConnection.disconnect("LOCAL");
});

```

```

    })

    document.getElementById("leavePairing").addEventListener("click", () => {
        peerConnection.sendBye();
    });

    window.addEventListener("beforeunload", () => {
        if (peerConnection.state === "CONNECTED") {
            peerConnection.sendBye();
        }
    });

```

chatchatapp.jschat.js

Chat.js

chat.js UI

```

export class Chat {

    #input = document.getElementById("chatInput");
    #sendBtn = document.getElementById("chatSend");
    #log = document.getElementById("chatLog");
    #peerConnection;

    constructor(peerConnection) {
        this.#peerConnection = peerConnection;
        this.updateUi("NOT_CONNECTED");
        this.#sendBtn.addEventListener("click", () => {
            if (this.#peerConnection.dataChannel === null) return console.log("No data channel");
            if (this.#input.value.trim() === "") return this.#input.value = "";
            this.#addToLog("local", this.#input.value);
            this.#peerConnection.dataChannel.send(JSON.stringify({chat: this.#input.value}));
            this.#input.value = "";
        });

        this.#input.addEventListener("keyup", event => {
            if (event.key !== "Enter") return;
            this.#sendBtn.click(); // reuse the click handler
        });
    }

    updateUi(state) {
        if (["NOT_CONNECTED", "CONNECTING", "CONNECTED"].includes(state))

```

```

) {
    this.#log.innerHTML = "";
}
if (state === "NOT_CONNECTED") this.#addToLog("server", "Click '
Find Stranger' to connect with a random person!");
if (state === "CONNECTING") this.#addToLog("server", "Finding a
stranger for you to chat with...");
if (state === "CONNECTED") this.#addToLog("server", "You're talk
ing to a random person. Say hi!");
if (state === "DISCONNECTED_LOCAL") this.#addToLog("server", "Yo
u disconnected");
if (state === "DISCONNECTED_REMOTE") this.#addToLog("server", "S
tranger disconnected");
}

addRemoteMessage = (message) => this.#addToLog("remote", message)

#addToLog(owner, message) {
    this.#log.insertAdjacentHTML("beforeend", `<div class="message $
{owner}">${message}</div>`);
    this.#log.scrollTop = this.#log.scrollHeight;
}
}

```

UI# JavaScript private Java/Kotlin

updateUiaddRemoteMessageapp.js

Peer-connection.js

WebRTC SDP

WebRTC "" SDP SDP WebSocket REST WebRTC
SDP

“offer” SDP “answer” SDP “GO_FIRST” offer answer WebRTC

“”SDP

“”

- WebSocket
- “GO_FIRST”“PARTNER_FOUND”
-
-
- SDP

- SDP
- ICE
- ICE
- ICE ondatachannel 3

“”

- WebSocket
- “PARTNER_FOUND”“GO_FIRST”
- SDP
- SDP
-
- SDP
- ICE
- ICE
- ICE ondatachannel 3 “”

“” ICE “”

```
export class PeerConnection {
  sdpExchange; // WebSocket with listeners for exchanging SDP offers and answers
  peerConnection; // RTCPeerConnection for exchanging media (with listeners for media and ICE)
  dataChannel; // RTCDataChannel for exchanging signaling and chat messages (with listeners)
  state; // NOT_CONNECTED, CONNECTING, CONNECTED, DISCONNECTED_SELF, DISCONNECTED_REMOTE
  options; // constructor args {onStateChange, onLocalMedia, onRemoteMedia, onChatMessage}
  localStream; // MediaStream from local webcam and microphone

  constructor(options) {
    this.options = options;
    this.init();
  }

  async init() { // needs to be separate from constructor because of a sync
    try {
      this.localStream = await navigator.mediaDevices.getUserMedia
```

```

({video: true, audio: true});
    this.options.onLocalMedia(this.localStream);
  } catch (error) {
    alert("Failed to enable webcam and/or microphone, please reload the page and try again");
  }
  this.setState("NOT_CONNECTED");
  this.peerConnection = this.createPeerConnection();
  this.sdpExchange = this.createSdpExchange();
}

```

```

createSdpExchange() { // WebSocket with listeners for exchanging SDP offers and answers
  let ws = new WebSocket(`ws://${window.location.host}/api/matchmaking`);
  ws.addEventListener("message", (event) => {
    const message = JSON.parse(event.data);
    console.log("Received WebSocket message", message.name)
    if (message.name === "PARTNER_FOUND") this.handlePartnerFound(message.data);
    if (message.name === "SDP_OFFER") this.handleSdpOffer(JSON.parse(message.data));
    if (message.name === "SDP_ANSWER") this.handleSdpAnswer(JSON.parse(message.data));
    if (message.name === "SDP_ICE_CANDIDATE") this.handleIceCandidate(JSON.parse(message.data));
  });
  ws.addEventListener("close", async () => {
    while (this.sdpExchange.readyState === WebSocket.CLOSED) {
      console.log("WebSocket closed, reconnecting in 1 second");
    }
    await new Promise(resolve => setTimeout(resolve, 1000));
    this.sdpExchange = this.createSdpExchange();
  });
  return ws;
}

```

```

createPeerConnection() { // RTCPeerConnection for exchanging media (with listeners for media and ICE)
  let conn = new RTCPeerConnection();
  conn.ontrack = event => {
    console.log(`Received ${event.track.kind} track`);
    this.options.onRemoteMedia(event.streams[0])
  };
  conn.onicecandidate = event => {
    if (event.candidate === null) { // candidate gathering complete

```

```

        console.log("ICE candidate gathering complete");
        return this.sdpExchange.send(JSON.stringify({name: "PAIR
ING_DONE"}));
    }
    console.log("ICE candidate created, sending to partner");
    let candidate = JSON.stringify(event.candidate);
    this.sdpExchange.send(JSON.stringify({name: "SDP_ICE_CANDIDA
TE", data: candidate}))
    };
    conn.oniceconnectionstatechange = () => {
        if (conn.iceConnectionState === "connected") {
            this.setState("CONNECTED");
            // ice candidates can still be added after "connected" s
            tate, so we need to log this with a delay
            setTimeout(() => console.log("WebRTC connection establis
hed"), 500);
        }
    };
    conn.ondatachannel = event => { // only for the "answerer" (the
one who receives the SDP offer)
        console.log("Received data channel from offerer");
        this.dataChannel = this.setupDataChannel(event.channel)
    };
    return conn;
}

setupDataChannel(channel) { // RTCDataChannel for exchanging signali
ng and chat messages
    channel.onmessage = event => {
        console.log("Received data channel message", event.data);
        if (event.data === "BYE") {
            this.disconnect("REMOTE");
            return console.log("Received BYE message, closing connec
tion");
        }
        this.options.onChatMessage(JSON.parse(event.data).chat);
    }
    return channel;
}

sendBye() {
    if (this.dataChannel === null) return console.log("No data chann
el");
    this.dataChannel.send("BYE");
    this.disconnect("LOCAL");
}

disconnect(originator) {

```



```

        this.dataChannel = null;
        this.peerConnection.close();
        this.peerConnection = this.createPeerConnection();
        this.setState(`DISCONNECTED_${originator}`);
    }

    setState(state) {
        this.state = state;
        this.options.onStateChange(state);
    }

    handlePartnerFound(instructions) {
        if (instructions !== "GO_FIRST") {
            return console.log("Partner found, waiting for SDP offer ..."); // only for the "answerer" (the one who receives the SDP offer)
        }
        console.log("Partner found, creating SDP offer and data channel");
        this.tryHandle("PARTNER_FOUND", async () => { // only for the "offerer" (the one who sends the SDP offer)
            this.dataChannel = this.setupDataChannel(this.peerConnection.createDataChannel("data-channel"));
            this.localStream.getTracks().forEach(track => this.peerConnection.addTrack(track, this.localStream));
            const offer = await this.peerConnection.createOffer();
            await this.peerConnection.setLocalDescription(offer);
            let offerJson = JSON.stringify(this.peerConnection.localDescription);
            this.sdpExchange.send(JSON.stringify({name: "SDP_OFFER", data: offerJson}));
        });
    }

    handleSdpOffer(offer) { // only for the "answerer" (the one who receives the SDP offer)
        this.tryHandle("SDP_OFFER", async () => {
            console.log("Received SDP offer, creating SDP answer");
            await this.peerConnection.setRemoteDescription(new RTCSessionDescription(offer));
            this.localStream.getTracks().forEach(track => this.peerConnection.addTrack(track, this.localStream));
            const answer = await this.peerConnection.createAnswer();
            await this.peerConnection.setLocalDescription(answer);
            let answerJson = JSON.stringify(this.peerConnection.localDescription);
            this.sdpExchange.send(JSON.stringify({name: "SDP_ANSWER", data: answerJson}));
        });
    }

```

```

    }

    handleSdpAnswer(answer) { // only for the "offerer" (the one who sends the SDP offer)
        this.tryHandle("SDP_ANSWER", async () => {
            await this.peerConnection.setRemoteDescription(new RTCSessionDescription(answer));
        });
    }

    handleIceCandidate(iceCandidate) {
        this.tryHandle("ICE_CANDIDATE", async () => {
            await this.peerConnection.addIceCandidate(new RTCIceCandidate(iceCandidate));
        });
    }

    tryHandle(command, callback) {
        try {
            callback()
        } catch (error) {
            console.error(`Failed to handle ${command}`, error);
        }
    }
}

```

WebRTC 6699669966996699

CSS CSS CSS body 3 1

```

[data-state=NOT_CONNECTED] button#startPairing, /* start button */
[data-state=DISCONNECTED_LOCAL] button#startPairing, /* start button */
[data-state=DISCONNECTED_REMOTE] button#startPairing, /* start button */
[data-state=CONNECTING] button#abortPairing, /* abort button */
[data-state=CONNECTED] button#leavePairing { /* leave button */
    display: block;
}

```

CSS

```

:root {
  --spacing: 12px;
  --border-radius-large: 8px;
  --border-radius-small: 4px;
}

* {
  box-sizing: border-box;
}

body { /* wraps header and main */
  display: grid;
  grid-template-rows: auto 1fr;
  grid-row-gap: var(--spacing);
  height: 100vh;
  padding: var(--spacing);
  margin: 0;
  font-family: 'Roboto', sans-serif;
}

header img {
  display: block;
  margin: 4px auto;
  max-height: 40px;
}

main { /* wraps video-panel and chat-panel */
  display: grid;
  grid-template-columns: 4fr 7fr; /* video panel is 4/11 of the width,
  chat panel is 7/11 */
  grid-column-gap: var(--spacing);
  overflow: auto;
}

.video-panel {
  display: grid;
  grid-template-rows: 1fr 1fr;
  grid-row-gap: var(--spacing);
  overflow: auto;

  & section {
    overflow: auto;
    display: flex;
    justify-content: center;
    align-items: center;
    background: #f2f2f2;
    border-radius: var(--border-radius-large);
  }
}

```

```

    & video {
        width: 100%;
        height: 100%;
        object-fit: cover;
    }

    &.local video {
        transform: scaleX(-1);
    }
}

.chat-panel {
    display: grid;
    grid-template-rows: 1fr auto;
    grid-row-gap: var(--spacing);
    padding: var(--spacing);
    background: #f2f2f2;
    border-radius: var(--border-radius-large);

    .chat-log {
        padding: var(--spacing);
        background: #fff;
        overflow-y: scroll;
        border-radius: var(--border-radius-small);
        line-height: 1.4;

        .message.local::before {
            font-weight: bold;
            color: blue;
            content: "You: ";
        }

        .message.remote::before {
            font-weight: bold;
            color: red;
            content: "Stranger: ";
        }

        .message.server {
            color: #999;
            font-style: italic;
        }
    }
}

.chat-controls {
    display: flex;

```

```

        .messaging {
            display: flex;
            flex-grow: 1;

            & input {
                margin: 0 16px;
                width: 100%;
                height: 40px;
                padding: 16px;
                border: 0;
                border-radius: var(--border-radius-small);
                font-size: 16px;

                &:focus {
                    outline: none;
                }
            }
        }

        .pairing button {
            width: 120px;
        }
    }
}

```

```

button {
    background: #1e88e5;
    color: #fff;
    border: 0;
    border-radius: var(--border-radius-small);
    height: 40px;
    line-height: 1;
    padding: 0 16px;
    cursor: pointer;

    &:hover {
        background: #1976d2;
    }
}

```

```

/* Conditional styles for buttons */

```

```

.chat-controls .pairing button {
    display: none;
}

```

```

[data-state=NOT_CONNECTED] button#startPairing,
[data-state=DISCONNECTED_LOCAL] button#startPairing,
[data-state=DISCONNECTED_REMOTE] button#startPairing,

```

```

[data-state=CONNECTING] button#abortPairing,
[data-state=CONNECTED] button#leavePairing {
    display: block;
}

/* Conditional styles for spinner */
.spinner {
    display: none;
}

[data-state=CONNECTING] .spinner {
    display: block;
}

/* Conditional styles for remote video */
body:not([data-state=CONNECTED]) .remote video {
    display: none;
}

/* Conditional styles for chat */
body:not([data-state=CONNECTED]) .chat-controls .messaging {
    filter: grayscale(1);
    opacity: 0.6;
    pointer-events: none;
}

/* Mobile layout */
@media (max-width: 768px) {
    main { /* put video panel on top of chat panel */
        grid-template-columns: none;
        grid-template-rows: 1fr 2fr;
        grid-row-gap: var(--spacing);
    }

    .video-panel { /* put video side by side */
        grid-template-rows: none;
        grid-template-columns: 1fr 1fr;
        grid-column-gap: var(--spacing);
    }
}

```


Jetty -

Jetty -

201892 • David Åse10-20

GitHub fork/clone

Jetty

Web SessionSession CookiesessionIdSession

Jetty 9.4 Jetty

Jetty HashMap (RAM) Jetty

Jetty Serializableimplements Serializable

a SessionSession a FileSessionDataStore

SessionHandler SessionCacheFileSessionDataStore

```
public static SessionHandler fileSessionHandler() {
    SessionHandler sessionHandler = new SessionHandler();
    SessionCache sessionCache = new DefaultSessionCache(sessionHandler);
    sessionCache.setSessionDataStore(fileSessionDataStore());
    sessionHandler.setSessionCache(sessionCache);
    sessionHandler.setHttpOnly(true);
    // make additional changes to your SessionHandler here
    return sessionHandler;
}
```

```
private static FileSessionDataStore fileSessionDataStore() {
    FileSessionDataStore fileSessionDataStore = new FileSessionDataStore();
    File baseDir = new File(System.getProperty("java.io.tmpdir"));
    File storeDir = new File(baseDir, "javalin-session-store");
    storeDir.mkdir();
    fileSessionDataStore.setStoreDir(storeDir);
    return fileSessionDataStore;
}
IO
```


SessionHandlerSessionCacheFileSessionDataStore JDBC

```

public static SessionHandler sqlSessionHandler(String driver, String url) {
    SessionHandler sessionHandler = new SessionHandler();
    SessionCache sessionCache = new DefaultSessionCache(sessionHandler);
    sessionCache.setSessionDataStore(
        jdbcDataStoreFactory(driver, url).getSessionDataStore(sessionHandler)
    );
    sessionHandler.setSessionCache(sessionCache);
    sessionHandler.setHttpOnly(true);
    // make additional changes to your SessionHandler here
    return sessionHandler;
}

private static JDBCSessionDataStoreFactory jdbcDataStoreFactory(String driver, String url) {
    DatabaseAdaptor databaseAdaptor = new DatabaseAdaptor();
    databaseAdaptor.setDriverInfo(driver, url);
    // databaseAdaptor.setDatasource(myDataSource); // you can set data source here (for connection
    pooling, etc)
    JDBCSessionDataStoreFactory jdbcSessionDataStoreFactory = new
    JDBCSessionDataStoreFactory();
    jdbcSessionDataStoreFactory.setDatabaseAdaptor(databaseAdaptor);
    return jdbcSessionDataStoreFactory;
}

```

MongoDB

```

private static MongoSessionDataStoreFactory mongoDataStoreFactory(String url, String dbName,
    String collectionName) {
    MongoSessionDataStoreFactory mongoSessionDataStoreFactory = new
    MongoSessionDataStoreFactory();
    mongoSessionDataStoreFactory.setConnectionString(url);
    mongoSessionDataStoreFactory.setDbName(dbName);
    mongoSessionDataStoreFactory.setCollectionName(collectionName);
    return mongoSessionDataStoreFactory;
}

```

Jetty JDBCMongoDBInfinispanHazelcast Google Cloud DataStoreJDBC jetty-server
MongoDB

SQL Jetty jettysessions MongoDB

```

{
  "id": {
    "$oid": "5b858d527d3c0f8722173292"
  },
  "id": "node0j4ii2zxu01i91g5f8odup78c30",
  "accessed": 1535479586876,
  "context": {

```

```

"00_0_0": {
  "metadata": {
    "lastNode": "node0",
    "lastSaved": 1535479586879,
    "version": 78
  },
  "signed-in-user": "tipsy" // custom data
},
"created": 1535479122617,
"expiry": 0,
"lastAccessed": 1535479585053,
"maxIdle": -1,
"valid": true
}
Jetty

```

SessionCacheJettySessionHandler
 SessionCacheDefaultSessionCacheNullSessionCache

DefaultSessionCache Jetty DefaultSessionCache
 NullSessionCache

NullSessionCache a Session SessionDataStoreJetty

10 MongoDBmlab 40

SameSite cookie
 Jetty Cookie (CSRF) Cookie Cookie SameSite=strict
 JSESSIONID

```

static SessionHandler customSessionHandler() {
  final SessionHandler sessionHandler = new SessionHandler();
  sessionHandler.setHttpOnly(true);
  sessionHandler.setSecureRequestOnly(true);
  sessionHandler.setSameSite(HttpCookie.SameSite.STRICT);
  return sessionHandler;
}

```

Jetty SessionHandlerSessionCacheSessionDataStore
 AFileSessionDataStore
 DefaultSessionCache
 NullSessionCache
 Javalin

javalin.io Javalin Javalin Jetty SessionHandler

```
var app = Javalin.create(config -> {
config.jetty.modifyServletContextHandler(handler ->
handler.setSessionHandler(fileSessionHandler()));
}).start(7070);
SessionHandlerSessionCache SessionDataStore Jetty
```

cookie cookie SameSitesecure

```
app.get("/write", ctx -> {
// values written to the session will be available on all your instances if you use a session db
ctx.sessionAttribute("my-key", "My value");
});
```

```
app.get("/read", ctx -> {
// values on the session will be available on all your instances if you use a session db
String myValue = ctx.sessionAttribute("my-key");
});
```

```
app.get("/invalidate", ctx -> {
// if you want to invalidate a session, jetty will clean everything up for you
ctx.req().getSession().invalidate();
});
```

ID

```
app.get("/change-id", ctx -> {
// it could be wise to change the session id on login, to protect against session fixation attacks
ctx.req().changeSessionId();
});
```

HTML Javalin

HTML Javalin

<https://javalin.io/tutorials/html-forms-example>

→

```
<dependencies>
  <dependency>
    <groupId>io.javalin</groupId>
    <artifactId>javalin-bundle</artifactId>
    <version>6.6.0</version>
  </dependency>
</dependencies>
```

```
import io.javalin.Javalin;
import io.javalin.http.staticfiles.Location;
import io.javalin.util.FileUtil;
import java.util.HashMap;
import java.util.Map;

public class JavalinHtmlFormsExampleApp {

    private static final Map<String, String> reservations = new HashMap<
>() {{
        put("saturday", "No reservation");
        put("sunday", "No reservation");
    }};

    public static void main(String[] args) {

        Javalin app = Javalin.create(config -> {
            config.staticFiles.add("/public", Location.CLASSPATH);
        });

        app.post("/make-reservation", ctx -> {
```

```

        reservations.put(ctx.formParam("day"), ctx.formParam("time")
    );
    ctx.html("Your reservation has been saved");
});

app.get("/check-reservation", ctx -> {
    ctx.html(reservations.get(ctx.queryParam("day")));
});

app.start();

}

}

```

7070

/src/resources/public postget

HTML

HTML /resources/public/index.html

<http://localhost:7777/>.

```

<h2>Make reservation:</h2>
<form method="post" action="/make-reservation">
    Choose day
    <select name="day">
        <option value="saturday">Saturday</option>
        <option value="sunday">Sunday</option>
    </select>
    <br>
    Choose time
    <select name="time">
        <option value="8:00 PM">8:00 PM</option>
        <option value="9:00 PM">9:00 PM</option>
    </select>
    <br>
    <button>Submit</button>
</form>

```

map.put()POST method="post"

<form>

Java app.post("/make-reservation", ctx -> {...} action action="/make-reservation"

/POST

<http://localhost:7777/make-r>

```

<h2>Check your reservation:</h2>
<form method="get" action="/check-reservation">
  Choose day
  <select name="day">
    <option value="saturday">Saturday</option>
    <option value="sunday">Sunday</option>
  </select>
  <br>
  <button>Submit</button>
</form>

```

GET

GET""GET <http://localhost:7777/check-reservation?day=saturday>URL

HTML GET POST

- POST
- POST ctx.formParam(key) Javalin
- GET POST
- GET URL ctx.queryParam(key) Javalin

```

app.post("/upload-example", ctx -> {
  ctx.uploadedFiles("files").forEach(file -> {
    FileUtil.streamToFile(file.content(), "upload/" + file.filename(
  ));
});
ctx.html("Upload successful");
});

```

```
ctx.uploadedFiles("files") filesupload
```

HTML

```

<h1>Upload example</h1>
<form method="post" action="/upload-example" enctype="multipart/form-data">
  <input type="file" name="files" multiple>

```

```
<button>Submit</button>  
</form>
```

```
enctype="multipart/form-data"
```

```
<form>
```

```
multiple
```

```
<input>
```

Javalin

Javalin

"" Gmail

201786 • David Åse~5

GitHub fork/clone

Maven →

```
<dependencies>
  <dependency>
    <groupId>io.javalin</groupId>
    <artifactId>javalin-bundle</artifactId> <!-- For handling http-r
requests -->
    <version>6.6.0</version>
  </dependency>
  <dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-email</artifactId> <!-- For sending emails -
->
    <version>1.5</version>
  </dependency>
  <dependency>
    <groupId>com.j2html</groupId>
    <artifactId>j2html</artifactId> <!-- For creating HTML form -->
    <version>1.6.0</version>
  </dependency>
</dependencies>
```

GET '/POST '/contact-us'GET '/contact-us/success'

```
import io.javalin.Javalin;
import org.apache.commons.mail.DefaultAuthenticator;
import org.apache.commons.mail.Email;
import org.apache.commons.mail.SimpleEmail;

import static io.javalin.apibuilder.ApiBuilder.get;
import static io.javalin.apibuilder.ApiBuilder.post;
```



```

import static j2html.TagCreator.br;
import static j2html.TagCreator.button;
import static j2html.TagCreator.form;
import static j2html.TagCreator.input;
import static j2html.TagCreator.textarea;

public class JavalinEmailExampleApp {

    public static void main(String[] args) {

        Javalin.create(config -> {
            config.router.apiBuilder(() -> {
                get("/", ctx -> ctx.html(
                    form().withAction("/contact-us").withMethod("post").
with(
                    input().withName("subject").withPlaceholder("Sub
ject"),
                    br(),
                    textarea().withName("message").withPlaceholder("
Your message ..."),
                    br(),
                    button("Submit")
                ).render()
            ));
            post("/contact-us", ctx -> {
                Email email = new SimpleEmail();
                email.setHostName("smtp.googlemail.com");
                email.setSmtPort(465);
                email.setAuthenticator(new DefaultAuthenticator("YOU
R_EMAIL", "YOUR_PASSWORD"));
                email.setSSLonConnect(true);
                email.setFrom("YOUR_EMAIL");
                email.setSubject(ctx.formParam("subject"));
                email.setMsg(ctx.formParam("message"));
                email.addTo("RECEIVING_EMAIL");
                email.send(); // will throw email-exception if somet
hing is wrong
                ctx.redirect("/contact-us/success");
            });
            get("/contact-us/success", ctx -> ctx.html("Your message
was sent"));
        }).start(7070);
    }
}

```

- YOUR_EMAIL Gmail youremail@gmail.com
- YOUR_PASSWORD Gmail *
- RECEIVING_ADDRESS
- * gmail

<http://localhost:7000>

/contact-us/success

SentGmail

Javalin REST API

Javalin REST API

Maven →

```
<dependencies>
  <dependency>
    <groupId>io.javalin</groupId>
    <artifactId>javalin-bundle</artifactId>
    <version>6.6.0</version>
  </dependency>
</dependencies>
```

API CRUD

```
import io.javalin.http.Context;
import java.util.*;

public class UserController {
    public record User(String name, String email) {}

    private static final Map<String, User> users;

    static {
        var tempMap = Map.of(
            randomId(), new User("Alice", "alice@alice.kt"),
            randomId(), new User("Bob", "bob@bob.kt"),
            randomId(), new User("Carol", "carol@carol.kt"),
            randomId(), new User("Dave", "dave@dave.kt")
        );
        users = new HashMap<>(tempMap);
    }

    public static void getAllUserIds(Context ctx) {
        ctx.json(users.keySet());
    }

    public static void createUser(Context ctx) {
        users.put(randomId(), ctx.bodyAsClass(User.class));
    }
}
```

```

    }

    public static void getUser(Context ctx) {
        ctx.json(users.get(ctx.pathParam("userId")));
    }

    public static void updateUser(Context ctx) {
        users.put(ctx.pathParam("userId"), ctx.bodyAsClass(User.class));
    }

    public static void deleteUser(Context ctx) {
        users.remove(ctx.pathParam("userId"));
    }

    private static String randomId() {
        return UUID.randomUUID().toString();
    }
}

```

RouteRole

io.javalin.security.RouteRole

```

import io.javalin.security.RouteRole;

enum Role implements RouteRole { ANYONE, USER_READ, USER_WRITE }

```

API

```

import io.javalin.Javalin;
import static io.javalin.apibuilder.ApiBuilder.*;

public class Main {

    public static void main(String[] args) {

        Javalin app = Javalin.create(config -> {
            config.router.mount(router -> {
                router.beforeMatched(Auth::handleAccess);
            }).apiBuilder(() -> {
                get("/", ctx -> ctx.redirect("/users"), Role.ANYONE);
                path("users", () -> {
                    get(UserController::getAllUserIds, Role.ANYONE);
                }
            )
        })
    }
}

```

```

        post(UserController::createUser, Role.USER_WRITE);
        path("{userId}", () -> {
            get(UserController::getUser, Role.USER_READ);
            patch(UserController::updateUser, Role.USER_WRITE);
            delete(UserController::deleteUser, Role.USER_WRITE);
        });
    });
}).start(7070);
}
}

```

- ANYONEgetAllUserIds
- USER_READgetUser
- USER_WRITEcreateUserupdateUserdeleteUser

Auth::handleAccess

- ApiRole.ANYONE
-
- 401 Unauthorized

```

public static void handleAccess(Context ctx) {
    var permittedRoles = ctx.routeRoles();
    if (permittedRoles.contains(Role.ANYONE)) {
        return; // anyone can access
    }
    if (userRoles(ctx).stream().anyMatch(permittedRoles::contains)) {
        return; // user has role required to access
    }
    ctx.header(Header.WWW_AUTHENTICATE, "Basic");
    throw new UnauthorizedResponse();
}

```

```
Javalin ctx.userRoles userRoles(ctx) map(Pair<String,
String>, Set<Role>) +
```

```
record Pair(String a, String b) {}
private static final Map<Pair, List<Role>> userRolesMap = Map.of(
    new Pair("alice", "weak-1234"), List.of(Role.USER_READ),
    new Pair("bob", "weak-123456"), List.of(Role.USER_READ, Role.USER_WR
ITE)
);
```

Basic-auth-header+ userRoleMap

```
public static List<Role> getUserRoles(Context ctx) {
    return Optional.ofNullable(ctx.basicAuthCredentials())
        .map(credentials -> userRolesMap.getOrDefault(new Pair(credentials.getUsername(), credentials.getPassword()), List.of()))
        .orElse(List.of());
}
```

base64 SSL

REST API

WebSockets

WebSockets

WebSocket WebSocket TCP HTTP WebSocket

Maven →

```
<dependencies>
  <dependency>
    <groupId>io.javalin</groupId>
    <artifactId>javalin-bundle</artifactId>
    <version>6.6.0</version>
  </dependency>
  <dependency>
    <groupId>com.j2html</groupId>
    <artifactId>j2html</artifactId>
    <version>1.6.0</version>
  </dependency>
</dependencies>
```

Javalin

Javalin

- /
-
- // websocket
-
- HTML JSON

```
import io.javalin.Javalin;
import io.javalin.http.staticfiles.Location;
import io.javalin.websocket.WsContext;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Map;
import java.util.concurrent.ConcurrentHashMap;
```

```

import static j2html.TagCreator.article;
import static j2html.TagCreator.attrs;
import static j2html.TagCreator.b;
import static j2html.TagCreator.p;
import static j2html.TagCreator.span;

public class JavalinWebsocketExampleApp {

    private static final Map<WsContext, String> userUsernameMap = new Co
ncurrentHashMap<>();
    private static int nextUserNumber = 1; // Assign to username for nex
t connecting user

    public static void main(String[] args) {
        Javalin app = Javalin.create(config -> {
            config.staticFiles.add("/public", Location.CLASSPATH);
            config.router.mount(router -> {
                router.ws("/chat", ws -> {
                    ws.onConnect(ctx -> {
                        String username = "User" + nextUserNumber++;
                        userUsernameMap.put(ctx, username);
                        broadcastMessage("Server", (username + " joined
the chat"));
                    });
                    ws.onClose(ctx -> {
                        String username = userUsernameMap.get(ctx);
                        userUsernameMap.remove(ctx);
                        broadcastMessage("Server", (username + " left th
e chat"));
                    });
                    ws.onMessage(ctx -> {
                        broadcastMessage(userUsernameMap.get(ctx), ctx.m
essage());
                    });
                });
            });
        }).start(7070);
    }

    // Sends a message from one user to all users, along with a list of
current usernames
    private static void broadcastMessage(String sender, String message)
    {
        userUsernameMap.keySet().stream().filter(ctx -> ctx.session.isOp
en()).forEach(session -> {
            session.send(
                Map.of(
                    "userMessage", createHtmlMessageFromSender(sender, m

```



```

message),
        "userlist", userUsernameMap.values()
    )
    );
});
}

// Builds a HTML element with a sender-name, a message, and a timestamp
private static String createHtmlMessageFromSender(String sender, String message) {
    return article(
        b(sender + " says:"),
        span(attrs(".timestamp"), new SimpleDateFormat("HH:mm:ss").format(new Date())),
        p(message)
    ).render();
}
}

```

JavaScript

JavaScript index.html

```

<!DOCTYPE html>
<html>
<head>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>WebSockets</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="chatControls">
        <input id="message" placeholder="Type your message">
        <button id="send">Send</button>
    </div>
    <ul id="userlist"> <!-- Built by JS --> </ul>
    <div id="chat"> <!-- Built by JS --> </div>
    <script src="websocketDemo.js"></script>
</body>
</html>

```

websocketDemo.js

```
// small helper function for selecting element by id
```

```

let id = id => document.getElementById(id);

//Establish the WebSocket connection and set up event handlers
let ws = new WebSocket("ws://" + location.hostname + ":" + location.port
+ "/chat");
ws.onmessage = msg => updateChat(msg);
ws.onclose = () => alert("WebSocket connection closed");

// Add event listeners to button and input field
id("send").addEventListener("click", () => sendAndClear(id("message").value));
id("message").addEventListener("keypress", function (e) {
    if (e.keyCode === 13) { // Send message if enter is pressed in input field
        sendAndClear(e.target.value);
    }
});

function sendAndClear(message) {
    if (message !== "") {
        ws.send(message);
        id("message").value = "";
    }
}

function updateChat(msg) { // Update chat-panel and list of connected users
    let data = JSON.parse(msg.data);
    id("chat").insertAdjacentHTML("afterbegin", data.userMessage);
    id("userlist").innerHTML = data.userlist.map(user => "<li>" + user + "</li>").join("");
}

```

localhost:7070

100 WebSocket

Prometheus grafana Javalin

Prometheus grafana Javalin

<https://javalin.io/tutorials/prometheus-example>

Maven →

- Javalin Prometheus
- unirest

```
<dependencies>
  <dependency>
    <groupId>io.javalin</groupId>
    <artifactId>javalin-bundle</artifactId>
    <version>6.6.0</version>
  </dependency>
  <dependency>
    <groupId>io.prometheus</groupId>
    <artifactId>simpleclient_httpserver</artifactId>
    <version>0.16.0</version>
  </dependency>
  <dependency>
    <groupId>com.konghq</groupId>
    <artifactId>unirest-java</artifactId>
    <version>3.13.10</version>
  </dependency>
</dependencies>
```

Prometheus Jetty StatisticsHandler Javalin Prometheus
QueuedThreadPoolJetty

```
public static void main(String[] args) throws Exception {

    StatisticsHandler statisticsHandler = new StatisticsHandler();
    QueuedThreadPool queuedThreadPool = new QueuedThreadPool(200, 8, 60_000);

    Javalin app = Javalin.create(config -> {
        config.jetty.threadPool = queuedThreadPool;
        config.jetty.modifyServer(server -> {
```

```

        server.setHandler(statisticsHandler);
    });
}).start(7070);
initializePrometheus(statisticsHandler, queuedThreadPool);
}

private static void initializePrometheus(StatisticsHandler statisticsHan
dler, QueuedThreadPool queuedThreadPool) throws IOException {
    StatisticsHandlerCollector.initialize(statisticsHandler);
    QueuedThreadPoolCollector.initialize(queuedThreadPool);
    HTTPServer prometheusServer = new HTTPServer(7080);
    LoggerFactory.getLogger("JavalinPrometheusExampleApp").info("Prometh
eus is listening on: http://localhost:7080");
}

```

Prometheus StatisticsHandlerQueuedThreadPool initializePrometheus
Prometheus

Prometheus/Grafana 7080

Prometheus-client

Prometheus CollectorStatisticsHandlerCollector QueuedThreadPoolCollector
.register()collect()

Maven

public static void main

```

router.apiBuilder(() -> { // available on config.router inside Javalin.c
reate()
    get("/1", ctx -> ctx.result("Hello World"));
    get("/2", ctx -> {
        Thread.sleep((long) (Math.random() * 2000));
        ctx.result("Slow Hello World");
    });
    get("/3", ctx -> ctx.redirect("/2"));
    get("/4", ctx -> ctx.status(400));
    get("/5", ctx -> ctx.status(500));
});

while (true) {
    spawnRandomRequests();
}

```

spawnRandomRequests()

```
private static void spawnRandomRequests() throws InterruptedException {
    new Thread(() -> {
        for (int i = 0; i < new Random().nextInt(50); i++) {
            Unirest.get("http://localhost:7070/1").asString(); // we want a lot more "200 - OK" traffic
            Unirest.get("http://localhost:7070/" + (1 + new Random().nextInt(5))).asString(); // hit a random (1-5) endpoint
        }
    }).start();
    Thread.sleep((int) (Math.random() * 250));
}
```

0-250ms — 0-100/1

Prometheus

PrometheusPrometheus

https://prometheus.io/docs/prometheus/latest/getting_started/

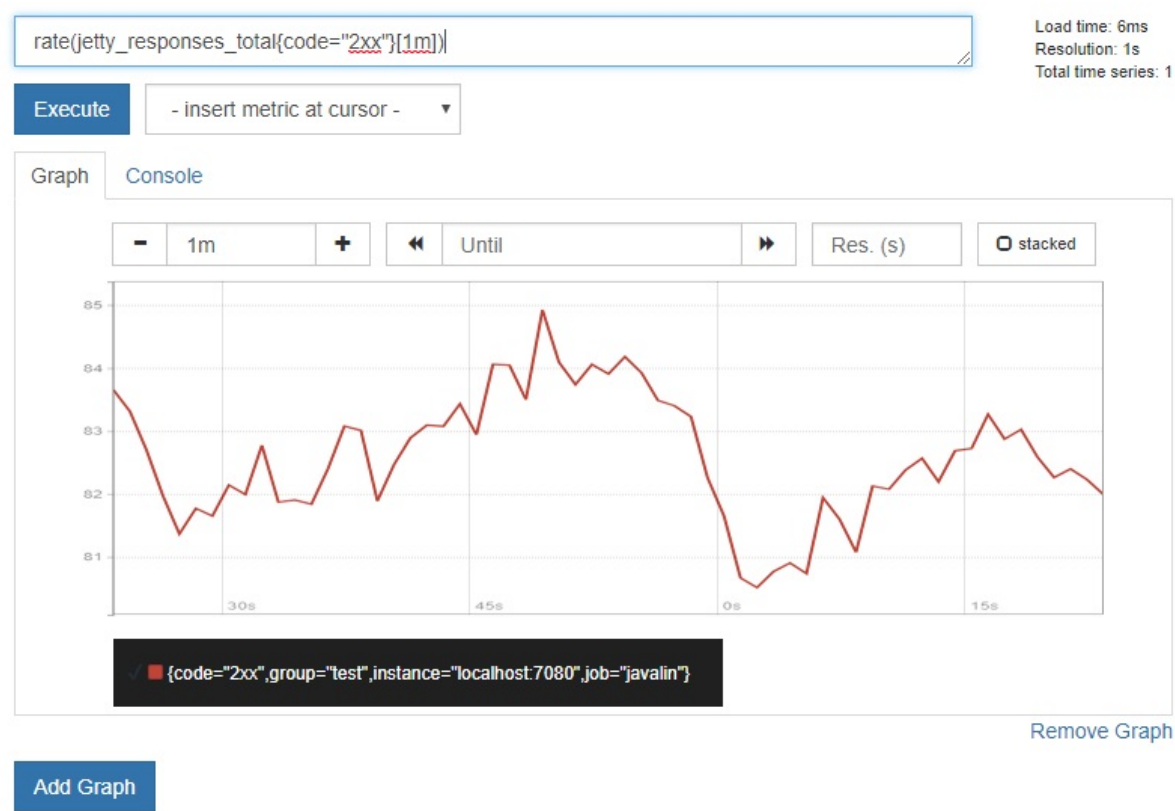
prometheus.yml scrape-config

```
scrape_configs:
  - job_name: 'javalin'
    scrape_interval: 1s
    static_configs:
      - targets: ['localhost:7080']
        labels:
          group: 'test'
```

Prometheus

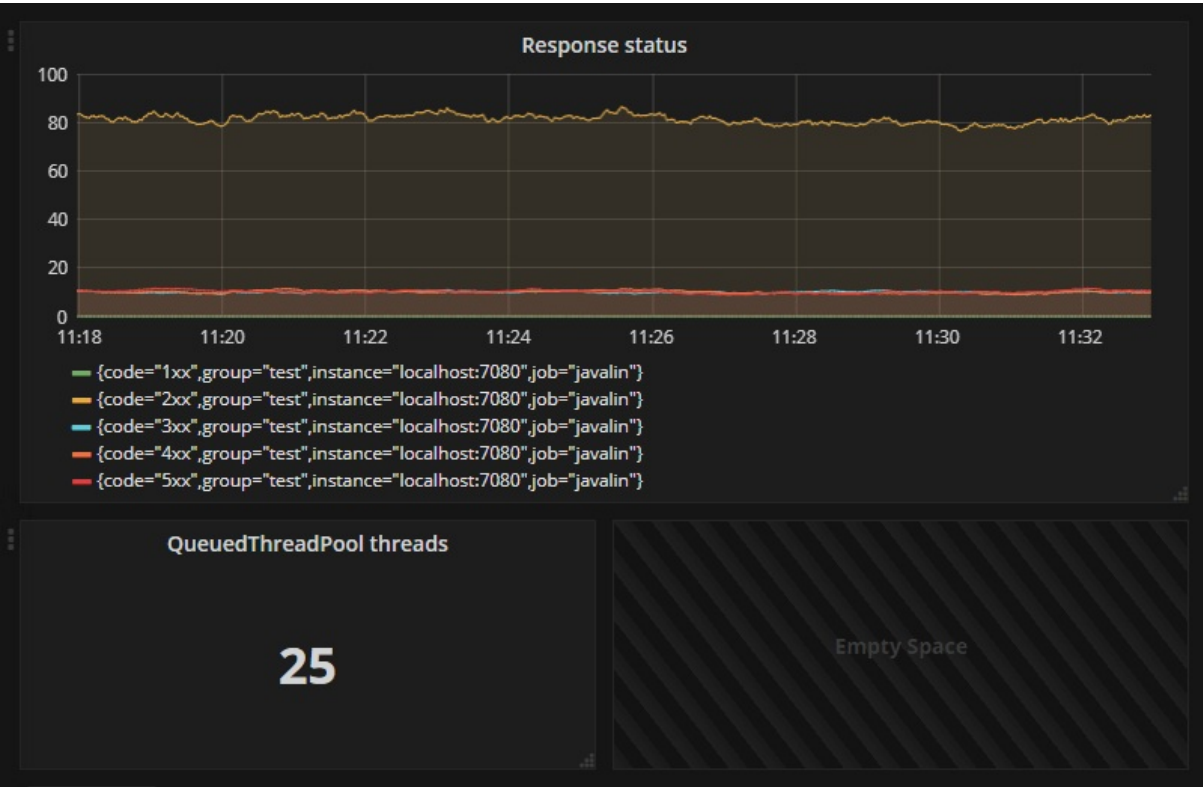
```
prometheus --config.file=prometheus.yml
```

localhost:9090 Prometheus



Prometheus Grafana <https://prometheus.io/docs/visualization/grafana/>

grafana prometheus



Javalin Vue

Javalin Vue

Javalin Vue

Web HTML PHPJSPJSTLTwirlScala Velocity Java
 jQuery/Zepto KnockoutAngularMeteor Vue LessSASSSCSS CSS
 BowerNPMYarnGrunt Gulp Webpack

/ DOM

Vue Vue NodeNPM Webpackreducer sagas
 rehydration tree shake

Kotlin Maven JavalinWeb
 Vue

io.javalin javalin-bundle 6.6.0 org.webjars.npm vue 3.2.37
 Web

/src/main/kotlin/javalinvue/JavalinVueExampleApp.kt

import io.javalin.Javalin

```
fun main() {
  val app = Javalin.create { config ->
    config.staticFiles.enableWebjars()
    config.vue.vueInstanceNameInJs = "app" // only required for Vue 3, is defined in layout.html
  }.start(7070)
}
```

HTML Vue
 /src/main/resources/vue/layout.html

@routeComponent
<http://localhost:7070/users/>
 /src/main/resources/vue/views/user-profile.vue

/src/main/resources/vue/views/not-found.vue

...

Javalin (before-filters) (route-roles) basic-auth

404

ANYONELOGGED_IN API

```
import io.javalin.Javalin
import io.javalin.apibuilder.ApiBuilder.get
import io.javalin.security.RouteRole
import io.javalin.http.Header
import io.javalin.http.Context
import io.javalin.http.HttpStatus
import io.javalin.http.UnauthorizedResponse
import io.javalin.vue.VueComponent

enum class Role : RouteRole { ANYONE, LOGGED_IN }

fun main() {

    val app = Javalin.create { config ->
        config.staticFiles.enableWebjars()
        config.vue.apply {
            stateFunction = { ctx -> mapOf("currentUser" to ctx.currentUser()) }
            vueInstanceNameInJs = "app"
        }
        config.router.mount {
            it.beforeMatched { ctx ->
                if (Role.LOGGED_IN in ctx.routeRoles() && ctx.currentUser() == null) {
                    ctx.header(Header.WWW_AUTHENTICATE, "Basic")
                    throw UnauthorizedResponse()
                }
            }
        }
        }.apiBuilder { // frontend routes
            get("/", VueComponent("hello-world"), Role.ANYONE)
            get("/users", VueComponent("user-overview"), Role.ANYONE)
            get("/users/{user-id}", VueComponent("user-profile"), Role.LOGGED_IN)
        }.apiBuilder { // api routes
            get("/api/users", UserController::getAll, Role.ANYONE)
            get("/api/users/{user-id}", UserController::getOne, Role.LOGGED_IN)
        }
    }.apply {
        error(HttpStatus.NOT_FOUND, "html", VueComponent("not-found"))
    }.start(7070)

}

private fun Context.currentUser() = this.basicAuthCredentials()?.username
basic-auth
```

JavalinVue

```
config.vue.stateFunction = { ctx -> mapOf("currentUser" to currentUser(ctx)) }
```

VueComponent basic-auth <http://localhost:7070/users/app-frame.vue>

JavalinVueExampleApp.kt
 UserController.ktgetAllgetOne
 layout.html Vue
 app-frame.vue
 user-overview.vue
 user-profile.vue
 not-found.vue 404
 pom.xml
 WebJars NPM

javalinvue-example

```
├── src
│   ├── main
│   │   ├── kotlin
│   │   │   ├── javalinvue
│   │   │   │   ├── UserController.kt
│   │   │   │   └── Main.kt
│   │   └── resources
│   └── components
│       ├── app-frame.vue
│       ├── components
│       │   ├── not-found.vue
│       │   ├── user-overview.vue
│       │   └── user-profile.vue
│       └── layout.html
└── pom.xml
```

DOM Vue Vue

10

30 25 20kbGZIP 100kb

2022 10 JavalinVue.optimizeDependenciesJavalin 4+ gzip 8kb

Chrome 30 + 25

- Vue JavaScript

Vue CDN Webpack

JS

JS

Kotlin/Java

Webjar NPM NPM Webjar dist

Javalin @componentRegistration@routeComponentJavalin Vue /resources/vue

Vue @componentRegistration

GET Vue CSS

JavalinVue \$javalin

..... JavalinVue

/src/main/resources/vue/components/app-frame.vue app-frame app-frame 404

layout.html

OpenAPI 3

<https://javalin.io/tutorials/openapi-example>

User CRUD API OpenAPI Java Kotlin GitHub

Maven →

```
io.javalin:javalin-bundle:6.6.0 io.javalin:community.openapi:javalin-openapi-plugin:6.6.0
io.javalin:community.openapi:javalin-swagger-plugin:6.6.0 io.javalin:community.openapi:javalin-redoc-
plugin:6.6.0
Open API
```

```
src/main/kotlin org.apache.maven.plugins:maven-compiler-plugin:3.10.1 11 11
io.javalin:community.openapi:openapi-annotation-processor:${javalin.version}
API
OpenAPI
```

```
package io.javalin.example.kotlin
```

```
import io.javalin.Javalin
import io.javalin.apibuilder.ApiBuilder.delete
import io.javalin.apibuilder.ApiBuilder.get
import io.javalin.apibuilder.ApiBuilder.patch
import io.javalin.apibuilder.ApiBuilder.path
import io.javalin.apibuilder.ApiBuilder.post
import io.javalin.example.kotlin.user.UserController
import io.javalin.openapi.OpenApiInfo
import io.javalin.openapi.plugin.OpenApiPlugin
import io.javalin.openapi.plugin.redoc.ReDocPlugin
import io.javalin.openapi.plugin.swagger.SwaggerPlugin
```

```
fun main() {
```

```
Javalin.create { config ->
    config.registerPlugin(OpenApiPlugin { pluginConfig ->
        pluginConfig.withDefinitionConfiguration { version, definition -
>
            definition.withOpenApiInfo { info: OpenApiInfo ->
                info.title = "Javalin OpenAPI example"
            }
        }
    })
    config.registerPlugin(SwaggerPlugin())
    config.registerPlugin(ReDocPlugin())
    config.router.apiBuilder {
```

```

        path("users") {
            get(UserController::getAll);
            post(UserController::create);
            path("{userId}") {
                get(UserController::getOne);
                patch(UserController::update);
                delete(UserController::delete);
            }
        }
    }
}.start(7001)

println("Check out ReDoc docs at http://localhost:7001/redoc")
println("Check out Swagger UI docs at http://localhost:7001/swagger")

```

```

}
OpenAPI config.plugins.register ReDoc Swagger UI API Web UI

```

API UserController

```
package io.javalin.example.kotlin.user
```

```
import io.javalin.http.Context
```

```
object UserController {
```

```
    fun create(ctx: Context) {
    }

```

```
    fun getAll(ctx: Context) {
    }

```

```
    fun getOne(ctx: Context) {
    }

```

```
    fun update(ctx: Context) {
    }

```

```
    fun delete(ctx: Context) {
    }

```

```

}
User CRUD API

```

Get users

```

@OpenApi(
  summary = "Get all users",
  operationId = "getAllUsers",
  tags = ["User"],
  responses = [OpenApiResponse("200", [OpenApiContent(Array::class)]),
  path = "/users",
  methods = [HttpMethod.GET]
)
fun getAll(ctx: Context) {
  ctx.json(UserService.getAll())
}
UserServiceUser OpenAPI GitHub

```

```

-
operationId - OpenAPI
-
- User

```

OpenAPI

User Schema UserGet all users

OpenAPI

200 User

repoTry it out

Update user

```

@OpenApi(
  summary = "Update user by ID",
  operationId = "updateUserById",
  tags = ["User"],
  pathParams = [OpenApiParam("userId", Int::class, "The user ID")],
  requestBody = OpenApiRequestBody([OpenApiContent(NewUserRequest::class)]),
  responses = [
    OpenApiResponse("204"),
    OpenApiResponse("400", [OpenApiContent(ErrorResponse::class)]),
    OpenApiResponse("404", [OpenApiContent(ErrorResponse::class)])
  ],
  path = "/users/{userId}",
  methods = [HttpMethod.PUT]
)

```

pathParams - Javalin queryParamsformParams
requestBody - JSON

userIdNewUserRequest 400
404

repo API Try it out

javalin-bundle JavalinJackson

GitHub POM

influxDB

InfluxDB

InfluxDB

<https://www.influxdata.com/>

- Docker Docker
- Gradle

java 2

- POST /upload
- GET /statistics

2 kotlin

realtimestatistics.Main

```
data class Statistic(val count: Int = 0, val timestamp: Long = Date().time)

data class Total(val count: Double, val sum: Double, val min: Double, val max: Double)

val influxHost = System.getenv().getOrDefault("influx.host", "influxdb")
!!

val influxDB: InfluxDB by lazy { InfluxDBFactory.connect("http://$influxHost:8086", "root", "root") }

fun main(args: Array<String>) {
    val app = Javalin.create().start(7000)
    val statisticService = StatisticsService(influxDB)
    val controller = Controller(statisticService)

    app.routes {
        get("/statistics", { ctx ->
            controller.get(ctx)
        })
    }
}
```

```

    })
    post("/upload", { ctx ->
        controller.post(ctx)
    })
}

}

class Controller(private val statisticService: StatisticsService) {
    private val asStatusCode = fun StatisticResult.(): Int {
        return if (this == StatisticResult.OK) {
            201
        } else {
            204
        }
    }

    fun post(ctx: Context) {
        val statistic = ctx.bodyAsClass(Statistic::class.java)
        val result = statisticService.create(statistic)
        ctx.status(result.asStatusCode())
    }

    fun get(ctx: Context) {
        ctx.json(statisticService.aggregated())
    }
}

```

realtimestatistics.StatisticsService

```

private val timeFrameInMillis = 60000

private val aggregateQuery = """
    SELECT count(s_count) as count,
           sum(s_count) as sum,
           min(s_count) as min,
           max(s_count) as max
    FROM uploads
    where time > now() - 60s
    """

init {
    influxDB.createDatabase(dbName)
}

fun create(statistic: Statistic): StatisticResult {
    val now = Date().time

```

```

        if ((statistic.timestamp + timeFrameInMillis) >= now) {
            influxDB.write(dbName, "", Point.measurement("uploads")
                .time(statistic.timestamp, TimeUnit.MILLISECONDS)
                .addField("s_count", statistic.count)
                .addField("s_timestamp", statistic.timestamp)
                .build())
            return StatisticResult.OK
        }
        return StatisticResult.OLD
    }

    fun aggregated(): Total {
        val query = Query(
            aggregateQuery,
            dbName
        )
        val results = influxDB.query(query)
            .results
        if (results.first().series == null) {
            return Total(0.0, 0.0, 0.0, 0.0)
        }
        return results.first().series.first().values
            .map { mutableList ->
                Total(mutableList[1].toString().toDouble(),
                    mutableList[2].toString().toDouble(),
                    mutableList[3].toString().toDouble(),
                    mutableList[4].toString().toDouble()
                )
            }[0]
    }
}

```

<https://github.com/ricardobaumann/real-time-statistics>

docker compose docker-compose up

<http://localhost:7000/>

POST/upload { "count" : 40 }

/statistics

Javalin Java 10 Google Guice

Javalin Java 10 Google Guice

Javalin

Google GuiceJava 10 Java 10

```
var amazingFramework = "Javalin"; // java10
// vs
String amazingFramework = "Javalin"; // not java10
```

Maven → Javalin Web slf4j jackson JSON Guice

```
<dependencies>
  <dependency>
    <groupId>io.javalin</groupId>
    <artifactId>javalin</artifactId>
    <version>2.8.0</version>
  </dependency>
  <dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-simple</artifactId>
    <version>2.0.17</version>
  </dependency>
  <dependency>
    <groupId>com.google.inject</groupId>
    <artifactId>guice</artifactId>
    <version>4.2.0</version>
  </dependency>
  <dependency>
    <groupId>com.google.inject.extensions</groupId>
    <artifactId>guice-multibindings</artifactId>
    <version>4.2.0</version>
  </dependency>
  <dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
    <version>2.9.5</version>
```

```

    </dependency>
</dependencies>

```

Java 10

```

<properties>
  <maven.compiler.source>10</maven.compiler.source>
  <maven.compiler.target>10</maven.compiler.target>
</properties>

```

Java

io.kidbank.user

UserController UserService

```

package io.kidbank.user;

import io.javalin.Context;
import io.kidbank.user.services.UserService;

import javax.inject.Inject;
import javax.inject.Singleton;

@Singleton
class UserController {
    private UserService userService;

    @Inject
    public UserController(UserService userService) {
        this.userService = userService;
    }

    public void index(Context ctx) {
        ctx.json(userService.getAllUsersUppercase());
    }
}

```

UserControllerRouting Google Guice UserController bindRoutes()

```

package io.kidbank.user;

import io.alzuma.Routing;
import io.javalin.Javalin;

import javax.inject.Inject;
import javax.inject.Singleton;

import static io.javalin.apibuilder.ApiBuilder.get;
import static io.javalin.apibuilder.ApiBuilder.path;

@Singleton
class UserRouting extends Routing<UserController> {
    private Javalin javalin;
    @Inject
    public UserRouting(Javalin javalin) {
        this.javalin = javalin;
    }

    @Override
    public void bindRoutes() {
        javalin.routes(() -> {
            path("api/kidbank/users", () -> {
                get(ctx -> getController().index(ctx));
            });
        });
    }
}

```

io.kidbank.user

Multibinder Google Guice

Multibinder.newSetBinder(...)

JavalinWeb

```

package io.kidbank.user;

import com.google.inject.AbstractModule;
import com.google.inject.multibindings.Multibinder;
import io.alzuma.Routing;
import io.kidbank.user.repositories.UserRepositoryModule;
import io.kidbank.user.services.UserServiceModule;

public class UserModule extends AbstractModule {

```

```

@Override
protected void configure() {
    bind(UserController.class);
    install(new UserServiceModule());
    install(new UserRepositoryModule());
    Multibinder.newSetBinder(binder(), Routing.class).addBinding().to(
        UserRouting.class);
}
}

```

Javalin Web

private Set<Routing> routes Google Guice Routes Multibinder

RoutingbindRoutes() Set<Routing> Javalin

```

package io.kidbank;

import com.google.inject.Inject;
import io.alzuma.AppEntrypoint;
import io.alzuma.Routing;
import io.javalin.Javalin;

import javax.inject.Singleton;
import java.util.Collections;
import java.util.Set;

@Singleton
class WebEntrypoint implements AppEntrypoint {
    private Javalin app;

    @Inject(optional = true)
    private Set<Routing> routes = Collections.emptySet();

    @Inject
    public WebEntrypoint(Javalin app) {
        this.app = app;
    }

    @Override
    public void boot(String[] args) {
        bindRoutes();
        app.port(7000);
        app.start();
    }

    private void bindRoutes() {
        routes.forEach(r -> r.bindRoutes());
    }
}

```

```

    }
}

```

WebModuleKid bank“ Web ”

MapBinderRouting“

```
MultibinderMapBinderHashMap<EntrypointType, AppEntrypoint>
```

```

package io.kidbank;

import com.google.inject.AbstractModule;
import com.google.inject.multibindings.MapBinder;
import io.alzuma.AppEntrypoint;
import io.alzuma.EntrypointType;
import io.javalin.Javalin;
import org.jetbrains.annotations.NotNull;

class WebModule extends AbstractModule {
    private Javalin app;

    private WebModule(Javalin app) {
        this.app = app;
    }

    @NotNull
    public static WebModule create() {
        return new WebModule(Javalin.create());
    }

    @Override
    protected void configure() {
        bind(Javalin.class).toInstance(app);
        MapBinder.newMapBinder(binder(), EntrypointType.class, AppEntrypoint.class).addBinding(EntrypointType.REST).to(WebEntrypoint.class);
    }
}

```

“Run as”Startup

```

import com.google.inject.Inject;
import io.alzuma.AppEntrypoint;
import io.alzuma.EntrypointType;

import javax.inject.Singleton;
import java.util.Collections;
import java.util.Map;

```



```

import java.util.Optional;

@Singleton
public class Startup {
    @Inject(optional = true)
    private Map<EntrypointType, AppEntrypoint> entrypoints = Collections
.emptyMap();

    public void boot(EntrypointType entrypointType, String[] args) {
        var entryPoint = Optional.ofNullable(entrypoints.get(entrypointT
ype));
        entryPoint.orElseThrow(() -> new RuntimeException("Entrypoint no
t defined")).boot(args);
    }
}

```

AppModule

```

import com.google.inject.AbstractModule;
import io.kidbank.KidBankModule;

public class AppModule extends AbstractModule {
    protected void configure() {
        bind(Startup.class);
        install(new KidBankModule());
    }
}

```

AppModuleStartup Javalin REST

```

public class App {
    public static void main(String[] args) {
        var injector = Guice.createInjector(new AppModule());
        injector.getInstance(Startup.class).boot(EntrypointType.REST, ar
gs);
    }
}

```

<http://localhost:7000/api/kidbank/users> ["BOB","KATE","JOHN"]

Web

Guice

JWT Javalin

JWT Javalin

JWT

Maven

Auth0 Java JWT

Auth0 Java JWT

Javalin Context JWT cookie / cookie JWT

JWT JWT JWTGeneratr Auth0 JWTVerifier

```
class MockUser {
    String name;
    String level;
```

```
    MockUser(String name, String level) {
        this.name = name;
        this.level = level;
    }
```

```
}
JWT
```

```
//1.
Algorithm algorithm = Algorithm.HMAC256("very_secret");
```

```
//2.
JWTGenerator generator = (user, alg) -> {
    JWTCreator.Builder token = JWT.create()
        .withClaim("name", user.name)
        .withClaim("level", user.level);
```

```

return token.sign(alg);
};

//3.
JWTVerifier verifier = JWT.require(algorithm).build();

//4.
JWTProvider provider = JWTProvider(algorithm, generator, verifier);
1) HMAC256 RSA

2) JWT

3) Auth0

4)

```

```

/generate/validate

```

```

//
// .. create your Javalin app ...
//
Handler generateHandler = context -> {
MockUser mockUser = new MockUser("Mocky McMockface", "user");
String token = provider.generateToken(mockUser);
context.json(new JWTResponse(token));
};

```

```

Handler validateHandler = context -> {
Optional decodedJWT = JavalinJWT.getTokenFromHeader(context)
.flatMap(provider::validateToken);

```

```

    if (!decodedJWT.isPresent()) {
        context.status(401).result("Missing or invalid token");
    }
    else {
        context.result("Hi " + decodedJWT.get().getClaim("name").asString())
    }
;
}

```

```

};

```

```

app.get("/generate", generateHandler);
app.get("/validate", validateHandler);
generateHandler JSON { "jwt": "..." }

```

```

validateHandlerBearer

```

/generate JWT“Bearer” /validate

JWT JWT

JWT Cookie

```
Handler decodeHandler = JavalinJWT.createHeaderDecodeHandler(provider);
JWT Cookie Cookie JWT
```

```
app.before(decodeHandler);
JWT
```

```
enum Roles implements Role {
    ANYONE,
    USER,
    ADMIN
}
```

```
Map<String, Role> rolesMapping = new HashMap<String, Role>() {{
    put("user", Roles.USER);
    put("admin", Roles.ADMIN);
}};
```

```
JWTAccessManager accessManager = new JWTAccessManager("level", rolesMapping,
    Roles.ANYONE);
app.accessManager(accessManager);
```

```
Handler generateHandler = context -> {
    MockUser mockUser = new MockUser("Mocky McMockface", "user");
    String token = provider.generateToken(mockUser);
    context.json(new JWTResponse(token));
};
```

```
Handler validateHandler = context -> {
    DecodedJWT decodedJWT = JavalinJWT.getDecodedFromContext(context);
    context.result("Hi " + decodedJWT.getClaim("name").asString());
};
```

```
app.get("/generate", generateHandler, roles(Roles.ANYONE));  
app.get("/validate", validateHandler, roles(Roles.USER, Roles.ADMIN));  
app.get("/adminsounge", validateHandler, roles(Roles.ADMIN));  
generateHandler validateHandler /validate/adminlounge
```

Javalin Servlet

Javalin Servlet

<https://javalin.io/2018/11/15/javalin-embedded-example.html>

Javalin Jetty WAR WAR Servlet 3.0

javalin-tomcat-embed-example

```
git clone https://github.com/tipsy/javalin-tomcat-embed-example
cd javalin-tomcat-embed-example
./gradlew clean appRun
```

Gradle Gretty Tomcat WAR

<http://localhost:8080/rest> REST

curl localhost:8080/rest/

Gradle WARGradle Gretty Tomcat WAR appRun

dependencies Javalin Jetty

```
dependencies {
    compile(kotlin("stdlib-jdk8"))
    compile("io.javalin:javalin:3.13.13") {
        exclude(mapOf("group" to "org.eclipse.jetty"))
        exclude(mapOf("group" to "org.eclipse.jetty.websocket"))
    }
    compile("org.slf4j:slf4j-simple:1.7.30")
}
```

servlet

```
@WebServlet(urlPatterns = ["/rest/*"], name = "MyRestServlet", asyncSupported = false)
class MyRestServlet : HttpServlet() {
    val javalin: JavalinServlet = Javalin.createStandalone()
```

```

        .get("/rest") { ctx -> ctx.result("Hello!") }
        .servlet()

```

```

        override fun service(req: HttpServletRequest, resp: HttpServletResponse) {
            javalin.service(req, resp)
        }
    }
}

```

```

createStandalone() Javalin JettyJavalin.create() WAR
java.lang.ClassNotFoundException: org.eclipse.jetty.server.Server
Servlet Servlet @WebServlet Servlet JavalinJavalin
@MultipartConfig UploadedFile UploadedFiles Servlet getter

```


GraalVM Javalin 22MB

GraalVM Javalin 22MB

<https://javalin.io/2018/09/27/javalin-graalvm-example.html>

Oracle GraalVM (AOT) JVM JVM SubstrateVM Go
 GraalVM RESTful Web Java JVM E.ON Java
 — Kubernetes “” 12 dockerized

- Java Spring Boot Kubernetes Pod 20 500
- Spring Boot 512MB Java JVM Spring
- Java JRE 65MB alpine openJDK Spring Boot 40MB Fat
 Jar 100MB Docker JRE Docker 100MB 2018 100MB
 Hello World Go 6MB

GraalVM AOT JVM GraalVM JVM Java
 GraalVM Class.forName()

```
if (someVariable) {
    Class.forName("SomeClazz")
    ...
}
```

someVariable “SomeClazz” - GraalVM
 spring boot - javalin.io Jetty

Docker GraalVM sdkman JDK sdkman

```
curl -s "https://get.sdkman.io" | bash
```

GraalVM JDK

```
sdk install java 1.0.0-rc-16-gr1 && sdk use java 1.0.0-rc-16-gr1
```

Hello World

```
public class Main {
    public static void main(String[] args) {
        Test t = new Test();
        t.setSomeValue("Hello World!");
        Javalin app = Javalin.create().start(7000);
        app.get("/", ctx -> ctx.json(t));
    }
}
```

Jackson SLF4J Javalin slf4j-simple

```
compile group: 'io.javalin', name: 'javalin', version: '2.2.0'
compile group: 'com.fasterxml.jackson.core', name: 'jackson-databind', version: '2.9.6'
compile group: 'org.slf4j', name: 'slf4j-simple', version: '1.7.25'
compile group: 'org.graalvm', name: 'graal-sdk', version: '1.0.0-rc6'
```

jar fat jar

```
task fatJar(type: Jar) {
    manifest {
        attributes 'Implementation-Title': 'Gradle Jar File Example',
                  'Implementation-Version': version,
                  'Main-Class': 'de.nerden.samples.graal.Main'
    }
    baseName = project.name + '-all'
    from { configurations.compile.collect { it.isDirectory() ? it : zipTree(it) } }
    with jar
}
```

GraalVM native-image

```
j0e@thinkpad ~/projects/graal-javalin master • ? 1 native-image
-jar ./build/libs/graal-javalin-all-1.0-SNAPSHOT.jar
Build on Server(pid: 28578, port: 34643)*
[graal-javalin-all-1.0-SNAPSHOT:28578] classlist: 2,977.05 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (cap): 963.06 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] setup: 1,663.57 ms
[ForkJoinPool-3-worker-3] INFO org.eclipse.jetty.util.log - Logging initialized @5682ms to org.eclipse.jetty.util.log.Slf4jLog
[graal-javalin-all-1.0-SNAPSHOT:28578] (typeflow): 10,510.28 ms
```

```
[graal-javalin-all-1.0-SNAPSHOT:28578] (objects): 6,598.95 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (features): 110.60 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] analysis: 17,612.10 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] universe: 859.27 ms
```

error: unsupported features in 8 methods

Detailed message:

Error: Unsupported method sun.nio.ch.InheritedChannel.soType0(int) is reachable: Native method. If you intend to use the Java Native Interface (JNI), specify -H:+JNI and see also -H:JNIConfigurationFiles=<path> (use -H:+PrintFlags for details)

To diagnose the issue, you can add the option --report-unsupported-elements-at-runtime. The unsupported element is then reported at run time when it is accessed the first time.

...

...

-H:+JNI

```
j0e@thinkpad ~ /projects/graal-javalin master • ? 1 native-image
-jar ./build/libs/graal-javalin-all-1.0-SNAPSHOT.jar -H:+JNI
Build on Server(pid: 28578, port: 34643)
[graal-javalin-all-1.0-SNAPSHOT:28578] classlist: 753.67 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (cap): 528.63 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] setup: 776.76 ms
[ForkJoinPool-15-worker-0] INFO org.eclipse.jetty.util.log - Logging initialized @616692ms to org.eclipse.jetty.util.log.Slf4jLog
[graal-javalin-all-1.0-SNAPSHOT:28578] (typeflow): 5,934.19 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (objects): 6,646.13 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (features): 83.06 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] analysis: 13,491.56 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] universe: 519.25 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (parse): 2,360.81 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (inline): 3,674.24 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] (compile): 15,925.13 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] compile: 22,729.43 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] image: 1,426.49 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] write: 280.71 ms
[graal-javalin-all-1.0-SNAPSHOT:28578] [total]: 40,064.13 ms
```

```
j0e@thinkpad ~ /projects/graal-javalin master • ? 1 ./graal-javalin-all-1.0-SNAPSHOT
[main] INFO io.javalin.Javalin -
```

```
| _ _ _ _ _ |
| | _ _ _ _ _ | ( _ ) _ _ _ _ _ |
```

```
|  _  | | / _ ` \ \ / / _ ` | | | ' _ \ | | | | | | |
|  |  | | | ( _ | \ V / ( _ | | | | | | |
|  \__ / \__,_ | \ / \__,_ | | | | | |
|_____|
|
| https://javalin.io/documentation
|_____|
```

Missing dependency 'Slf4j simple'. Add the dependency.

pom.xml:

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-simple</artifactId>
  <version>1.7.25</version>
</dependency>
```

build.gradle:

```
compile "org.slf4j:slf4j-simple:1.7.25"
```

Visit <https://javalin.io/documentation#logging> if you need more help

[main] INFO io.javalin.Javalin - Starting Javalin ...

[main] ERROR io.javalin.Javalin - Failed to start Javalin

java.lang.IllegalArgumentException: Class org.eclipse.jetty.servlet.ServletMapping[] is instantiated reflectively but was never registered. Register the **class by using org.graalvm.nativeimage.RuntimeReflection**

at java.lang.Throwable.<init>(Throwable.java:265)

at java.lang.Exception.<init>(Exception.java:66)

at java.lang.RuntimeException.<init>(RuntimeException.java:62)

at java.lang.IllegalArgumentException.<init>(IllegalArgumentExceptionException.java:52)

at com.oracle.svm.core.genscavenge.graal.AllocationSnippets.checkDynamicHub(AllocationSnippets.java:162)

at org.eclipse.jetty.util.ArrayUtil.addToArray(ArrayUtil.java:91)

at org.eclipse.jetty.servlet.ServletHandler.addServletWithMapping(ServletHandler.java:907)

at org.eclipse.jetty.servlet.ServletContextHandler.addServlet(ServletContextHandler.java:462)

at io.javalin.core.util.JettyServerUtil.initialize(JettyServerUtil.kt:71)

at io.javalin.Javalin.start(Javalin.java:136)

at io.javalin.Javalin.start(Javalin.java:103)

at de.nerden.samples.graal.Main.main(Main.java:10)

at com.oracle.svm.core.JavaMainWrapper.run(JavaMainWrapper.java:)

163)

GraalVM

GraalVM ServletMapping "" JSON JSON

```
[
  {
    "name": "[Lorg.eclipse.jetty.servlet.ServletMapping;",
    "allDeclaredFields": true,
    "allPublicFields": true,
    "allDeclaredMethods": true,
    "allPublicMethods": true
  },
  {
    "name": "org.slf4j.impl.StaticLoggerBinder",
    "allDeclaredFields": true,
    "allPublicFields": true,
    "allDeclaredMethods": true,
    "allPublicMethods": true
  },
  {
    "name": "com.fasterxml.jackson.databind.ObjectMapper",
    "allDeclaredFields": true,
    "allPublicFields": true,
    "allDeclaredMethods": true,
    "allPublicMethods": true
  },
  {
    "name": "de.nerden.samples.graal.Test",
    "allDeclaredFields": true,
    "allPublicFields": true,
    "allDeclaredMethods": true,
    "allPublicMethods": true
  }
]
```

```
[Lorg.eclipse.jetty.servlet.ServletMapping; ServletMapping
```

slf4j Jackson

```
[qtp1024494636-165] WARN io.javalin.core.ExceptionMapper - Uncaught exception
com.fasterxml.jackson.databind.exc.InvalidDefinitionException: No serializer found for class de.nerden.samples.graal.Test and no properties discovered to create BeanSerializer (to avoid exception, disable SerializationFeature.FAIL_ON_EMPTY_BEANS)
    at java.lang.Throwable.<init>(Throwable.java:265)
```

```

    at java.lang.Exception.<init>(Exception.java:66)
    at java.io.IOException.<init>(IOException.java:58)
    at com.fasterxml.jackson.core.JsonProcessingException.<init>(JsonProcessingException.java:33)
    at com.fasterxml.jackson.databind.JsonMappingException.<init>(JsonMappingException.java:237)
    at com.fasterxml.jackson.databind.exc.InvalidDefinitionException.<init>(InvalidDefinitionException.java:38)
    at com.fasterxml.jackson.databind.exc.InvalidDefinitionException.from(InvalidDefinitionException.java:77)
    at com.fasterxml.jackson.databind.SerializerProvider.reportBadDefinition(SerializerProvider.java:1191)
    at com.fasterxml.jackson.databind.DatabindContext.reportBadDefinition(DatabindContext.java:312)
    at com.fasterxml.jackson.databind.ser.impl.UnknownSerializer.failForEmpty(UnknownSerializer.java:71)
    at com.fasterxml.jackson.databind.ser.impl.UnknownSerializer.serialize(UnknownSerializer.java:33)
    at com.fasterxml.jackson.databind.ser.DefaultSerializerProvider._serialize(DefaultSerializerProvider.java:480)
    at com.fasterxml.jackson.databind.ser.DefaultSerializerProvider.serializeValue(DefaultSerializerProvider.java:319)
    at com.fasterxml.jackson.databind.ObjectMapper._configAndWriteValue(ObjectMapper.java:3905)
    at com.fasterxml.jackson.databind.ObjectMapper.writeValueAsString(ObjectMapper.java:3219)
    at io.javalin.json.JavalinJackson.toJson(JavalinJackson.kt:26)
    at io.javalin.json.JavalinJson$toJsonMapper$1.map(JavalinJson.kt:28)
    at io.javalin.json.JavalinJson.toJson(JavalinJson.kt:32)
    at io.javalin.Context.json(Context.kt:510)
    at de.nerden.samples.graal.Main.lambda$main$0(Main.java:11)
    at de.nerden.samples.graal.Main$$Lambda$925/1179449634.handle(Unknown Source)
    at io.javalin.security.SecurityUtil.noopAccessManager(SecurityUtil.kt:22)
    at io.javalin.Javalin$$Lambda$928/1713301975.manage(Unknown Source)
    at io.javalin.Javalin.lambda$addHandler$0(Javalin.java:485)
    at io.javalin.Javalin$$Lambda$931/1107122283.handle(Unknown Source)
    at io.javalin.core.JavalinServlet$service$2$1.invoke(JavalinServlet.kt:48)
    at io.javalin.core.JavalinServlet$service$2$1.invoke(JavalinServlet.kt:20)
    at io.javalin.core.JavalinServlet$service$1.invoke(JavalinServlet.kt:145)
    at io.javalin.core.JavalinServlet$service$2.invoke(JavalinServlet

```

```

t.kt:43)
    at io.javalin.core.JavalinServlet.service(JavalinServlet.kt:109)
    at io.javalin.core.util.JettyServerUtil$initialize$httpHandler$1
.doHandle(JettyServerUtil.kt:59)
    at org.eclipse.jetty.server.handler.ScopedHandler.nextScope(Scop
edHandler.java:203)
    at org.eclipse.jetty.servlet.ServletHandler.doScope(ServletHandl
er.java:473)
    at org.eclipse.jetty.server.session.SessionHandler.doScope(Sessi
onHandler.java:1564)
    at org.eclipse.jetty.server.handler.ScopedHandler.nextScope(Scop
edHandler.java:201)
    at org.eclipse.jetty.server.handler.ContextHandler.doScope(Conte
xtHandler.java:1242)
    at org.eclipse.jetty.server.handler.ScopedHandler.handle(ScopedH
andler.java:144)
    at org.eclipse.jetty.server.handler.HandlerList.handle(HandlerLi
st.java:61)
    at org.eclipse.jetty.server.handler.StatisticsHandler.handle(Sta
tisticsHandler.java:174)
    at org.eclipse.jetty.server.handler.HandlerWrapper.handle(Handle
rWrapper.java:132)
    at org.eclipse.jetty.server.Server.handle(Server.java:503)
    at org.eclipse.jetty.server.HttpChannel.handle(HttpChannel.java:
364)
    at org.eclipse.jetty.server.HttpConnection.onFillable(HttpConnec
tion.java:260)
    at org.eclipse.jetty.io.AbstractConnection$ReadCallback.succee
ded(AbstractConnection.java:305)
    at org.eclipse.jetty.io.FillInterest.fillable(FillInterest.java:
103)
    at org.eclipse.jetty.io.ChannelEndPoint$2.run(ChannelEndPoint.ja
va:118)
    at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob(QueuedT
hreadPool.java:765)
    at org.eclipse.jetty.util.thread.QueuedThreadPool$2.run(QueuedTh
readPool.java:683)
    at java.lang.Thread.run(Thread.java:748)
    at com.oracle.svm.core.posix.thread.PosixJavaThreads.pthreadStar
tRoutine(PosixJavaThreads.java:238)

```

Jackson / JSON

```
docker run --net=host birdy/graal-javalin
```

```
curl localhost:7000
```

```
j0e@thinkpad ~ /projects/graal-javalin master ? 2 curl localhost:
```

7000



33707

01:15:12

`{"abc": "LOL"}%`

JVM

3

•

Javalin JVM 1-2 CLI

•

—StackoverflowRSS

`cat /proc/7812/status`

VmRSS: 18260 kB

18MB curl 25MB JVM

VmRSS: 183580 kB

1/10

•

Fat Jar 5.7MB JRE 57MB

<https://hub.docker.com/r/library/openjdk/tags/>

60MB 22MB

`-rwxr-xr-x 1 j0e users 22M Sep 24 01:38 graal-javalin`

Go 1/3 Go JDK 9

GraalVM hack GraalVM Java ;) / GraalVM

micronaut.io

<https://github.com/gaemerocher/micronaut-graal-experiments>

Dockerfile GitHub

<https://github.com/birdayz/graal-javalin>

Docker Go

`FROM birdy/graalvm:latest``WORKDIR /tmp/build``ENV GRADLE_USER_HOME /tmp/build/.gradle``ADD . /tmp/build``RUN ./gradlew build fatJar`

MinDoc


```
RUN native-image -jar /tmp/build/build/libs/graal-javalin-all-1.0-SNAPSHOT.jar -H:ReflectionConfigurationFiles=reflection.json -H:+JNI \
    -H:Name=graal-javalin --static --delay-class-initialization-to-runtime
=io.javalin.json.JavalinJson
```

```
FROM scratch
COPY --from=0 /tmp/build/graal-javalin /
ENTRYPOINT ["/graal-javalin"]
```

Dockerfile Docker

Hibernate ORM Javalin

Javalin Java/Kotlin Web

Javalin Hibernate ORM

Kotlin DSL Gradle

IntelliJ IDEA -> -> Java -> Gradle DSLKotlin

Gradlebuild.gradle.kts

Javalin Hibernate PostgreSQL

Lombok

```
plugins {
    id("java")
}
```

```
group = "com.brucemelo.app"
```

```
repositories {
    mavenCentral()
}
```

```
val javalinVersion = "6.3.0"
val lpmbokVersion = "1.18.34"
val postgresqlVersion = "42.7.3"
val hibernateVersion = "7.0.0.Beta1"
val junitVersion = "5.10.3"
```

```
dependencies {
    implementation("io.javalin:javalin-bundle:$javalinVersion")
    compileOnly("org.projectlombok:lombok:$lpmbokVersion")
    annotationProcessor("org.projectlombok:lombok:$lpmbokVersion")
    implementation("org.postgresql:postgresql:$postgresqlVersion")
    implementation("org.hibernate.orm:hibernate-core:$hibernateVersion")
    annotationProcessor("org.hibernate.orm:hibernate-jpamodelgen:$hibernateVersion")
    testImplementation(platform("org.junit:junit-bom:$junitVersion"))
    testImplementation("org.junit.jupiter:junit-jupiter")
}
```

```
tasks.test {
    useJUnitPlatform()
}
```

Docker PostgreSQL

docker compose PostgreSQL

version: '3.8'

services:

postgres:

container_name: postgres1

image: postgres:15.7

environment:

POSTGRES_USER: sa

POSTGRES_PASSWORD: sa

POSTGRES_DB: mydatabase

ports:

- "5432:5432"

restart: **unless-stopped**

Hibernate

Hibernate Course

[@Setter](#) [@Getter](#)

[@Entity](#)

[@Table](#)

public class Course {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

public static Course **newCourse**(String name) {

var course = **new** Course();

course.setName(name);

return course;

}

}

Hibernate /SessionFactory

HibernateConfiguration SessionFactory

Hibernate

SessionFactory Session/StatelessSession

Hibernate

class AppHibernateConfig {

```

static Configuration configuration() {
    var configuration = new Configuration();
    var settings = new Properties();
    settings.put(AvailableSettings.JAKARTA_JDBC_DRIVER, "org.postgresql.
Driver");
    settings.put(AvailableSettings.JAKARTA_JDBC_URL, "jdbc:postgresql://
localhost:5432/mydatabase");
    settings.put(AvailableSettings.JAKARTA_JDBC_USER, "sa");
    settings.put(AvailableSettings.JAKARTA_JDBC_PASSWORD, "sa");
    settings.put(AvailableSettings.HIGHLIGHT_SQL, true);
    settings.put(AvailableSettings.HBM2DDL_AUTO, Action.ACTION_CREATE);

    configuration.setProperties(settings);
    configuration.addAnnotatedClass(Course.class);
    return configuration;
}

```

}

Hibernate

class AppHibernateSessionFactory {

```

private static final Logger logger = LoggerFactory.getLogger(AppHibernat
eSessionFactory.class);

private static SessionFactory sessionFactory;

static SessionFactory getSessionFactory() {
    if (Objects.isNull(sessionFactory)) {
        try {
            var configuration = AppHibernateConfig.configuration();
            var serviceRegistry = new StandardServiceRegistryBuilder()
                .applySettings(configuration.getProperties())
                .build();
            sessionFactory = configuration.buildSessionFactory(serviceRe
gistry);
        } catch (Throwable ex) {
            logger.error("Failed to create session factory", ex);
        }
    }
    return sessionFactory;
}

```

}

Hibernate

Hibernate

Hibernate StatelessSession Transaction - AppHibernate

AppHibernateConfigAppHibernateSessionFactory

public class AppHibernate {

```
public static void inTransaction(Consumer<StatelessSession> consumer) {
    AppHibernateConfig.getSessionFactory().inStatelessTransaction(consum
er);
}
```

```
public static <R> R fromTransaction(Function<StatelessSession, R> functi
on) {
    return AppHibernateConfig.getSessionFactory().fromStatelessTransacti
on(function);
}
```

}

Javalin

Hibernate CourseHandler HTTP Javalin Context

CourseHandler AppHibernate Hibernate

public class CourseHandler {

```
public static Handler listAll = (context) -> {
    var result = AppHibernate.fromTransaction(CourseQueries_::getAllCour
ses);
    context.json(new ResultCourse(result));
};
```

```
public static Handler save = (context) -> {
    var newCourse = context.bodyAsClass(NewCourse.class);
    var result = AppHibernate.fromTransaction(session -> {
        var insertedId = session.insert(Course.newCourse(newCourse.name(
)));
        return session.get(Course.class, insertedId);
    });
    context.json(result).status(HttpStatus.CREATED);
};
```

}

CourseHandler AppHibernate Hibernate

JavalinApp Main

Javalin

public class JavalinApp {

```
public static Javalin create() {
    return Javalin.create((var config) -> config.router.apiBuilder(() ->
    {
        path("/", () -> get(ctx -> ctx.json("Ok")));
        path("/courses", () -> {
            get(CourseHandler.listAll);
            post(CourseHandler.save);
        });
    }));
}
```

}

public class Main {

```
public static void main(String[] args) {
    JavalinApp.create().start(8080);
}
```

}

JUnit 5

class CoursesTest {

```
Javalin app = JavalinApp.create();
JavalinJackson javalinJackson = new JavalinJackson();

@Test
@DisplayName("Should save and list courses")
void test1() {
    JavalinTest.test(app, (server, client) -> {
        var newCourse = new NewCourse("Course1");
        var postResponse = client.post("/courses", newCourse);
        assertEquals(postResponse.code(), HttpStatus.CREATED.getCode());
        var response = client.get("/courses");
        assertEquals(response.code(), HttpStatus.OK.getCode());
        assertNotNull(response.body());
        ResultCourse result = javalinJackson.fromJsonString(response.body().string(), ResultCourse.class);
        assertNotNull(result.courses());
        var firstCourse = result.courses().stream().findFirst();
    });
}
```

```
        assertTrue(firstCourse.isPresent());
        assertEquals(firstCourse.get().getName(), newCourse.name());
    });
}
```

```
}
```

Javalin Hibernate Java 21Javalin Hibernate 7 Web

HikariCP Agroal

Javalin

Javalin

javalin-rendering Javalin Web HTML javalin-rendering JTEMustache
VelocityPebbleHandlebars Thymeleaf

```
<dependency>
  <groupId>io.javalin</groupId>
  <artifactId>javalin-rendering</artifactId>
  <version>6.6.0</version>
</dependency>
<dependency>
  <groupId><!-- template engine group --></groupId>
  <artifactId><!-- template engine artifact --></artifactId>
  <version><!-- template engine version --></version>
</dependency>
```

javalin-rendering

- Freemarker → <https://freemarker.apache.org>
- jte → <https://jte.gg/>
- mustache → <https://github.com/spullara/mustache.java>
- pebbletemplates → <https://pebbletemplates.io/>
- thymeleaf → <https://www.thymeleaf.org/>
- velocity → <https://velocity.apache.org/>
- Commonmark → <https://github.com/commonmark/commonmark-java>

/markdown src/resources/templates Javalin

```
Javalin.create(config -> {
  config.fileRenderer(new JavalinMustache());
});
```

```
ctx.render()
```

```
ctx.render("/templateFile.ext", model("firstName", "John", "lastName", "Doe"));
```

```
Javalin.create(config -> {
    config.fileRenderer(new JavalinVelocity(myVelocityEngine));
});
```

Javalin

JavalinRenderer

Javalin JavalinRenderer

```
class JavalinRenderer implements FileRenderer {
    private Map<String, FileRenderer> renderers = new HashMap<>();
    public JavalinRenderer register(String extension, FileRenderer renderer) {
        renderers.put(extension, renderer);
        return this;
    }

    @Override
    public String render(String filePath, Map<String, ? extends Object> model, Context context) {
        String extension = filePath.substring(filePath.lastIndexOf(".") + 1);
        return renderers.get(extension).render(filePath, model, context);
    }
}
```

```
Javalin.create(config -> {
    config.fileRenderer(
        new JavalinRenderer()
```

Javalin

```
.register("mustache", new JavalinMustache())  
.register("jte", new JavalinJte())  
);  
});
```

JavaEasyExcel

JavaEasyExcel

xcelEasyExcel

www.jb51.net/article...

github.com/alibaba/e...

```
const download = () => {
  axios({
    method: 'GET',
    url: config.http.baseUrl + '/templateDownload',
    responseType: 'blob',
  })
  .then(function (res) {
    const content = res.data
    const blob = new Blob([content], { type: "application/vnd.openxmlformats-offic"
    const downloadElement = document.createElement("a");
    const href = window.URL.createObjectURL(blob);
    downloadElement.href = href;
    downloadElement.download = decodeURI(res.headers['filename']);
    document.body.appendChild(downloadElement);
    downloadElement.click();
    document.body.removeChild(downloadElement); //
    window.URL.revokeObjectURL(href); // blob
  })
}
```

excelspringbootjar

```
@Override
public void templateDownload(HttpServletResponse response, HttpServletRequest request) {
  //
  String fileName = ".xlsx" ;
  //
  String filePath = "/template/template.xlsx";
  TemplateDownloadUtil.download(response, request, fileName, filePath);
}
```

```

import lombok.extern.slf4j.Slf4j;
import org.springframework.core.io.ClassPathResource;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.URLEncoder;

/**
 *
 * @author
 * @date 2021/05/20 9:20
 */
@Slf4j
public class TemplateDownloadUtil {

    public static void download(HttpServletResponse response, HttpServletRequest request, String
        try {
            response.setContentType("application/vnd.openxmlformats-officedocument.spreadsheetml.");
            response.setCharacterEncoding("utf-8");
            // URLEncoder.encode easyexcel
            response.setHeader("Content-Disposition", "attachment; filename=" + URLEncoder.encode
            response.setHeader("filename", URLEncoder.encode(fileName, "UTF-8"));
            response.setHeader("Access-Control-Expose-Headers", "filename,Content-Disposition"

            //
            String filePath = getClass().getResource("/template/template.xlsx").getPath();
            FileInputStream input = new FileInputStream(filePath);

            //
            ClassPathResource resource = new ClassPathResource(filePath);
            InputStream input = resource.getInputStream();
            OutputStream out = response.getOutputStream();
            byte[] b = new byte[2048];
            int len;
            while ((len = input.read(b)) != -1) {
                out.write(b, 0, len);
            }
            // Excel"xxx.xlsx""
            response.setHeader("Content-Length", String.valueOf(input.getChannel().size()));
            input.close();
        } catch (Exception e) {
            log.error(" : " , e);
        }
    }
}

```

EasyExcel

EasyExcel

	A	B	C	D	E
	填写须知： 1. 系统自动识别Excel表格，表头必须含有“姓名”、“身份证号”、“实发金额”； 2. 带“*”为必填字段，填写后才能上传成功； 3. 若需上传其他表头，可自行在“实发金额”后添加表头，表头最多可添加20个，表头名称请控制在8个字以内； 4. 填写的表头内容不可超过30个字； 5. 实发金额支持填写到2位小数； 6. 每次导入数据不超过5000条。				
1	表头示例：				
2	*姓名	*身份证号	*实发金额		
3					
4					
5					
6					
7					
8					
9					
10					CSDN @yupengfei112233

```

import com.alibaba.excel.context.AnalysisContext;
import com.alibaba.excel.event.AnalysisEventListener;
import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONObject;
import lombok.Data;
import lombok.extern.slf4j.Slf4j;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * excel
 *
 * @author yupf
 * @description Listener springexcelnew, spring
 */
@Slf4j
@Data
public class BatchReadListener extends AnalysisEventListener<Map<Integer, String>> {

    /**
     * 500list
     */
    private static final int BATCH_COUNT = 500;
    //Excel
    private List<Map<Integer, Map<Integer, String>>> list = new ArrayList<>();
    //Excel
    private Map<Integer, String> headTitleMap = new HashMap<>();

    /**
     * DAOservice
     */
    private DbFileBatchService dbFileBatchService;
    private DbFileContentService dbFileContentService;
    private FileBatch fileBatch;
    private int total = 0;

    /**

```

```

    * spring,Listenerspring
    */
    public BatchReadListener(DbFileBatchService dbFileBatchService, DbFileContentService dbFileContentService) {
        this.dbFileBatchService = dbFileBatchService;
        this.dbFileContentService = dbFileContentService;
        this.fileBatch = fileBatch;
    }

    /**
     *
     *
     * @param data    one row value. Is is same as {@link AnalysisContext#readRowHolder()}
     * @param context
     */
    @Override
    public void invoke(Map<Integer, String> data, AnalysisContext context) {
        log.info(":{}" , JSON.toJSONString(data));
        total++;
        Map<Integer, Map<Integer, String>> map = new HashMap<>();
        map.put(context.readRowHolder().getRowIndex(), data);
        list.add(map);
        // BATCH_COUNT
        if (list.size() >= BATCH_COUNT) {
            saveData();
            // list
            list.clear();
        }
    }

    /**
     *
     *
     * @param context
     */
    @Override
    public void doAfterAllAnalysed(AnalysisContext context) {
        //
        saveData();
        log.info(" " );
    }

    /**
     *
     */
    @Override
    public void invokeHeadMap(Map<Integer, String> headMap, AnalysisContext context) {
        log.info("{}" , JSONObject.toJSONString(headMap));
        headTitleMap = headMap;
    }

    /**
     *
     */
    private void saveData() {
        log.info("{}" , list.size());
        FileContent fileContent = null;
        List<FileContent> fileContentList = list.stream().flatMap(
            integerMap -> integerMap.entrySet().stream().map(entrySet -> {
                //entrySet.getKey()RowIndex,
                Integer rowIndex = entrySet.getKey();
                Map<Integer, String> value = entrySet.getValue();
                log.info(JSONObject.toJSONString(value));
                fileContent = new FileContent();
                fileContent.setBatchId(fileBatch.getId());
            })
        );
    }

```

```

        fileContent.setBatchNo(fileBatch.getBatchNo());
        //
        fileContent.setName(value.get(0) != null ? value.get(0).trim() : "");
        fileContent.setCertNo(value.get(1) != null ? value.get(1).trim() : "");
        fileContent.setRealAmount(value.get(2) != null ? value.get(2).trim() : "");
        //JSON
        fileContent.setFieldsValue(JSONObject.toJSONString(value));
        //rowIndex
        fileContent.setRowNum(rowIndex + 1);
        fileContent.setCreateTime(LocalDateTime.now());
        return xcSalaryFileContent;
    }
    }).collect(Collectors.toList());
    log.info(JSONObject.toJSONString(fileContentList));
    dbFileContentService.saveBatch(fileContentList);
    log.info("");
}
}

```

```

BatchReadListener listener = new BatchReadListener(dbFileBatchService, dbFileContentService, fileBatch
try {
    //headRowNumber12
    EasyExcel.read(fileInputStream, listener).headRowNumber(2).sheet().doRead();
} catch (Exception e) {
    log.info("EasyExcel{}", e);
    throw new CustomException("");
}
//
Map<Integer, String> headTitleMap = listener.getHeadTitleMap();
//
List<String> headList = headTitleMap.keySet().stream().map(key -> {
    String head = headTitleMap.get(key);
    log.info(head);
    return head == null ? "" : head.replace(" ", "");
}).collect(Collectors.toList());
//

```

EasyExcel

```

private List<List<String>> getFileHeadList( FileBatch fileBatch) {
    String head = fileBatch.getFileHead();
    List<String> headList = Arrays.asList(head.split(","));
    List<List<String>> fileHead = headList.stream().map(item -> concatHead(Lists.newArrayList
    fileHead.add(concatHead(Lists.newArrayList(" " ))));
    return fileHead;
}

/**
 *
 * @param headContent

```



```

    * @return
    */
    private List<String> concatHead(List<String> headContent) {
        String remake = "
            1.Excel""""""\n"
            2. "" \n"
            3. ""208\n"
            4. 30\n"
            5. 2\n"
            6. 5000\n"
            \n" +
            \n" +
            \n" +
            ""
            ;
        headContent.add(0, remake);
        return headContent;
    }

```

```

List<FileContent> fileContentList = dbFileContentService.list(
    Wrappers.<FileContent>lambdaQuery()
        .eq(FileContent::getBatchId, fileBatch.getId())
        .orderByAsc(FileContent::getRowNum)
);
List<List<Object>> contentList = fileContentList.stream().map(fileContent -> {
    List<Object> rowList = new ArrayList<>();
    String fieldsValue = fileContent.getFieldsValue();
    JSONObject contentObj = JSONObject.parseObject(fieldsValue);
    for (int columnIndex = 0 , length = headList.size(); columnIndex < length; columnIndex++)
        Object content = contentObj.get(columnIndex);
        rowList.add(content == null ? "" : content);
    }
    rowList.add(fileContent.getCheckMessage());
    return rowList;
}).collect(Collectors.toList());

```

```

import com.alibaba.excel.metadata.data.DataFormatData;
import com.alibaba.excel.metadata.data.WriteCellData;
import com.alibaba.excel.write.handler.context.CellWriteHandlerContext;
import com.alibaba.excel.write.metadata.style.WriteCellStyle;
import com.alibaba.excel.write.metadata.style.WriteFont;
import com.alibaba.excel.write.style.HorizontalCellStyleStrategy;
import org.apache.poi.ss.usermodel.BorderStyle;
import org.apache.poi.ss.usermodel.HorizontalAlignment;
import org.apache.poi.ss.usermodel.IndexedColors;

import java.util.List;

/**
 *
 */
public class CellStyleStrategy extends HorizontalCellStyleStrategy {

    private final WriteCellStyle headWriteCellStyle;

```

```

private final WriteCellStyle contentWriteCellStyle;

/**
 *
 */
private final List<Integer> columnIndexes;

public CellStyleStrategy(List<Integer> columnIndexes, WriteCellStyle headWriteCellStyle, WriteCellStyle
    this.columnIndexes = columnIndexes;
    this.headWriteCellStyle = headWriteCellStyle;
    this.contentWriteCellStyle = contentWriteCellStyle;
}

//
@Override
protected void setHeadCellStyle( CellWriteHandlerContext context) {
    //
    WriteFont headWriteFont = new WriteFont();
    headWriteFont.setFontName(" " );
    //
    if (columnIndexes.get(0).equals(context.getRowIndex())) {
        headWriteCellStyle.setFillForegroundColor(IndexedColors.WHITE.getIndex());
        headWriteCellStyle.setHorizontalAlignment(HorizontalAlignment.LEFT);
        headWriteFont.setFontHeightInPoints((short) 12);
        headWriteFont.setBold(false);
        headWriteFont.setFontName(" " );
    }else{
        headWriteCellStyle.setFillForegroundColor(IndexedColors.GREY_25_PERCENT.getIndex())
        headWriteCellStyle.setHorizontalAlignment(HorizontalAlignment.CENTER);
        headWriteFont.setFontHeightInPoints((short) 11);
        headWriteFont.setBold(false);
        headWriteFont.setFontName(" " );
    }
    headWriteCellStyle.setWriteFont(headWriteFont);
    DataFormatData dataFormatData = new DataFormatData();
    dataFormatData.setIndex((short)49);
    headWriteCellStyle.setDataFormatData(dataFormatData);
    if (stopProcessing(context)) {
        return;
    }
    WriteCellData<?> cellData = context.getFirstCellData();
    WriteCellStyle.merge(headWriteCellStyle, cellData.getOrCreateStyle());
}

//
@Override
protected void setContentCellStyle(CellWriteHandlerContext context) {
    WriteFont contentWriteFont = new WriteFont();
    contentWriteFont.setFontName(" " );
    contentWriteFont.setFontHeightInPoints((short) 11);
    //
    contentWriteCellStyle.setWriteFont(contentWriteFont);
    contentWriteCellStyle.setBorderLeft(BorderStyle.THIN);
    contentWriteCellStyle.setBorderTop(BorderStyle.THIN);
    contentWriteCellStyle.setBorderRight(BorderStyle.THIN);
    contentWriteCellStyle.setBorderBottom(BorderStyle.THIN);
    DataFormatData dataFormatData = new DataFormatData();
    dataFormatData.setIndex((short)49);
    contentWriteCellStyle.setDataFormatData(dataFormatData);
    contentWriteCellStyle.setHorizontalAlignment(HorizontalAlignment.CENTER);
    WriteCellData<?> cellData = context.getFirstCellData();
    WriteCellStyle.merge(contentWriteCellStyle, cellData.getOrCreateStyle());
}
}

```

```

import com.alibaba.excel.write.style.row.AbstractRowHeightStyleStrategy;
import org.apache.poi.ss.usermodel.Row;

/**
 *
 */
public class CellRowHeightStyleStrategy extends AbstractRowHeightStyleStrategy {

    @Override
    protected void setHeadColumnHeight(Row row, int relativeRowIndex) {
        //17.7
        if(relativeRowIndex == 0){
            //excel1515*20=300
            row.setHeight((short) 3240);
        }
    }

    @Override
    protected void setContentColumnHeight(Row row, int relativeRowIndex) {
    }
}

```

EasyExcelLongestMatchColumnWidthStyleStrategy()

```

EasyExcel.write(fileName, LongestMatchColumnWidthData.class)
    .registerWriteHandler(new LongestMatchColumnWidthStyleStrategy()).sheet("").doWrite(dataLong

```

```

import com.alibaba.excel.enums.CellDataTypeEnum;
import com.alibaba.excel.metadata.Head;
import com.alibaba.excel.metadata.data.CellData;
import com.alibaba.excel.metadata.data.WriteCellData;
import com.alibaba.excel.write.metadata.holder.WriteSheetHolder;
import com.alibaba.excel.write.style.column.AbstractColumnWidthStyleStrategy;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.collections.CollectionUtils;
import org.apache.poi.ss.usermodel.Cell;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

/**
 * @author yupf
 * @description
 * @date 2022/9/7 18:48
 */
@Slf4j
public class CellWidthStyleStrategy extends AbstractColumnWidthStyleStrategy {

```

```

private Map<Integer, Map<Integer, Integer>> CACHE = new HashMap<>();

@Override
protected void setColumnWidth(WriteSheetHolder writeSheetHolder, List<WriteCellData<?>> cellDataList,
    Map<Integer, Integer> maxColumnWidthMap = CACHE.get(writeSheetHolder.getSheetNo()));
    if (maxColumnWidthMap == null) {
        maxColumnWidthMap = new HashMap<>();
        CACHE.put(writeSheetHolder.getSheetNo(), maxColumnWidthMap);
    }
    if (isHead) {
        if (relativeRowIndex.intValue() == 1) {
            Integer length = cell.getStringCellValue().getBytes().length;
            Integer maxColumnWidth = maxColumnWidthMap.get(cell.getColumnIndex());
            if (maxColumnWidth == null || length > maxColumnWidth) {
                maxColumnWidthMap.put(cell.getColumnIndex(), length);
                writeSheetHolder.getSheet().setColumnWidth(cell.getColumnIndex(), length *
            }
        }
    }
    } else {
        Integer columnWidth = this.dataLength(cellDataList, cell, isHead);
        if (columnWidth >= 0) {
            if (columnWidth > 255) {
                columnWidth = 255;
            }
            Integer maxColumnWidth = maxColumnWidthMap.get(cell.getColumnIndex());
            if (maxColumnWidth == null || columnWidth > maxColumnWidth) {
                maxColumnWidthMap.put(cell.getColumnIndex(), columnWidth);
                writeSheetHolder.getSheet().setColumnWidth(cell.getColumnIndex(), columnWidth
            }
        }
    }
}

private Integer dataLength(List<WriteCellData<?>> cellDataList, Cell cell, Boolean isHead)
    if (isHead) {
        return cell.getStringCellValue().getBytes().length;
    } else {
        CellData cellData = cellDataList.get(0);
        CellDataTypeEnum type = cellData.getType();
        if (type == null) {
            return -1;
        } else {
            switch (type) {
                case STRING:
                    return cellData.getStringValue().getBytes().length;
                case BOOLEAN:
                    return cellData.getBooleanValue().toString().getBytes().length;
                case NUMBER:
                    return cellData.getNumberValue().toString().getBytes().length;
                default:
                    return -1;
            }
        }
    }
}
}
}
}

```

```

EasyExcel.write(response.getOutputStream())
    .head(head)

```

```

        .registerWriteHandler(new CellRowHeightStyleStrategy()) //
        .registerWriteHandler(new CellStyleStrategy(Arrays.asList(0,1),new WriteCellStyle(), new WriteCellStyle
        .registerWriteHandler(new CellWidthStyleStrategy())
        .sheet(sheetName)
        .doWrite(list);

```



EasyExcel

EasyExcelapi

JavaEasyExcel,EasyExcel

:

```

JavaeasyExcelexcel
JavaEasyExcel
JavaEasyexcel
JavaEasyExcel
JavaEasyExcel
Java easyExcel
JavaEasyExcelExcel
Java EasyExcelsheet
Java EasyExcel
JavaExcel(EasyExcel)

```

EasyExcel