Go lernen für einen Symfony Websocket Proxy

Benjamin Eberlei @beberlei

29.04.2016, Symfony Live Cologne

passion for software quality

SymfonyLive COLOGNE 2016 27-29 APRIL

tideways.io

Qafoo

passion for software quality

Introduction

Polygot Programming:

Why even consider another language?

php

Shared Nothing Architecture

Simple Deployment

Fast Prototyping

Huge Ecosystem (Composer)

php

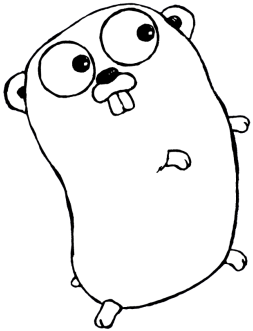
PHP - The Ugly

PHP is a bad choice for

- Asynchroneous I/O
- Threads/Parallel Processing
- Very High Throughput
- Websockets

Go

- Announced by Google in 2009
- Stable 1.0 release in 2012
- Currently at version 1.6
- Huge standard library
- `http://golang.org`



Language Properties

- Compiled Language
- Imperative/Procedural Code
- Strongly/Statically Typed

Feels like writing PHP4

What are we using it for?

- Monitoring
  - StatsD Client
  - Syslog UDP to TCP
- Tideways
  - Local Daemon
  - CLI
  - Websockets
- Microservices (less than 5000 LOC)
  - Timeseries Database

Hello World

```php
<?php
// helloworld.php
echo "Hi!";
```

```go
// helloworld.go
package main

import "fmt"

func main() {
  fmt.Println("Hi!")
}
```

- C-ish syntax
- Abbrevations for many keywords
- Program starts in function and package `main`
- semicolon statement termination is optional
- Functions are always namespaced

How to run it?

```
$ sudo apt-get install golang

$ go run helloworld.go
Hi!

$ go build helloworld.go
$ ./helloworld
Hi!
```

- Compiled into single binaries
- Includes **ALL** runtime dependencies
- Compiler is extremly strict
- Available on Windows, Linux, Mac (+more)

How do I organize it?

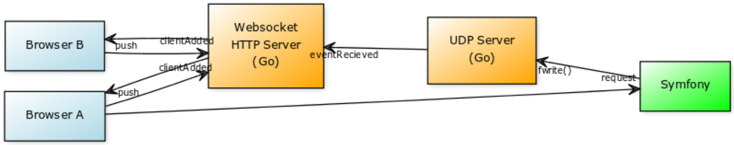
```
helloworld
|-- helloworld
|-- helloworld.go
\-- .gitignore
```

- Start with everything in a single file
- Learn about `$GOPATH` later
  - `$GOPATH` is like PHPs include path

Lets built something useful

- A websocket proxy
- Browsers start websocket connection to our Go proxy
- Our Symfony application sends messages to Proxy via UDP
- Use-cases
  - Flash Messages
  - Messages between users (Chat, ...)
  - Task-Based UIs and Event Sourcing

Architecture



The Code

`https://github.com/beberlei/websocket-proxy-example`

passion for software quality

Symfony Side: Messaging Service

```php
<?php

class Messaging
{
  public function send(User $user, $name, $args)
  {
    $addr = "udp://127.0.0.1:8081";
    $fp = @stream_socket_client($addr);
    fwrite($fp, json_encode([
        'UserId' => $user->getId(),
        'Name' => $name,
        'Payload' => $args,
    ]));
    fclose($fp);
  }
}
```

Qafoo
passion for software quality

Symfony Side: Messaging Service

```php
<?php

class BlogController extends Controller
{
    public function createPostAction()
    {
        // ...
        $messaging = $this->get('messaging');
        $messaging->send(
                $user,
                'BlogPostCreated',
                ['title' => $post->getTitle()]
        );
    }
}
```

passion for software quality

Building our Go program in steps

1. Declare global variables and types we need
2. Build the Websocket HTTP server
3. Build the UDP Server endpoint
4. Integrate UDP+Websocket servers
5. Multi-threading with Goroutines

Variables in Go

```go
package main

var udpAddr = "127.0.0.1:8125"
var httpAddr string
var httpAddr string = ":8080"
```

- Go is statically typed and Compiler can infer types
- Type of a variable can **never** change
- * prefix denotes a pointer, like PHP references

What about Memory Management?

- Go memory management ”similar” to PHP
- No manual memory management/cleanup
- Automatic garbage collection
- Local variables get cleaned up when function ends
- Global variables stick around until program ends
- Supports both ”Pass by Value” and ”Pass by Reference”

Arrays in Go (Hashmaps)

```go
package main

var clients  map[int]*Client

func main() {
	clients = make(map[int]*Client)
}
```

- Maps in Go are reference types, which means they require allocation and initialization
- With `make()` the required memory is allocated

Communicate through Channels

```
var clientAdded          chan *Client
var clientDisconnected chan *Client
var eventReceived        chan *Event

func main() {
    // ...
    clientAdded =          make(chan *Client)
    clientDisconnected = make(chan *Client)
    eventReceived =      make(chan *Event, 100)
}
```

- Channels are in-memory queues at the language level
- Allow safe communication between different ”threads”

passion for software quality

Client and Event Struct Types

```go
type Client struct {
	id        int
	ws        *websocket.Conn
	sendEvent chan *Event
}

type Event struct {
	UserId  int
	Name    string
	Payload interface{}
}
```

- Go has structs, roughly similar to PHP classes.
- Properties can be public (Uppercase) or private (Lowercase)
- Structs can be passed by value or by reference (as pointer)

Writing the HTTP/Websocket server

```
import "http"
import "log"
import "golang.org/x/net/websocket"

func ListenHttp() {
    log.Println("Listening server...")

    http.Handle("/ws", websocket.Handler(HandleWsRequest))
    http.Handle("/", http.FileServer(http.Dir(".")))
    http.ListenAndServe(httpAddr, nil)
}
```

Namespacing, Imports and Vendoring

- Go has a large standard library
- Functionality is grouped by packages
- You write `import json` to import package
- You can import remote code from Github or other SCMs
- Support for Composer like Vendoring with help of tools

Wrting the Request Handler Callback

```go
func HandleWsRequest(ws *websocket.Conn) {
	defer ws.Close()
	maxId++ // this is not thread-safe

	client := &Client{
		id: maxId,
		ws: ws,
		sendEvent: make(chan *Event, 10),
	}

	client.Start()
}
```

passion for software quality

Methods on Structs

```go
func (c *Client) ListenEvents() {
    log.Println("Client %d Listening to events", c.id)
    // ...
}
```

Writing the UDP listener

```go
package main

import "net"
import "log"

var udpAddr string = "127.0.0.1:8081"

func ListenUdp() {
    serverAddr, _ := net.ResolveUDPAddr("udp", udpAddr)
    conn, _ := net.ListenUDP("udp", serverAddr)

    for {
        buf := make([]byte, 1024)
        n, _, _ := conn.ReadFromUDP(buf)
        log.Printf("Received UDP msg %s\n", buf[0:n])
    }
}
```

Return Values and Errors

```
serverAddr, _ := net.ResolveUDPAddr("udp", udpAddr)
conn, err := net.ListenUDP("udp", serverAddr)

if (err != nil) {
    log.Printf("Cannot listen to %s\n", udpAddr)
    os.Exit(1)
}
```

- Go allows multiple return values
- Declared, Unused variables lead to compiler Error
- Variables named "underscore" are ignored: _
- Explicit error handling required

UDP loop: Create Events from JSON

```go
for {
    n, _, err := conn.ReadFromUDP(buf)

    if err != nil { continue }

    var event *Event
    err = json.Unmarshal(buf[0:n], &event)

    if err != nil { continue }

    eventReceived <- event
}
```

Working with Channels

How do we make `clientAdded`, `clientDisconnected` and `eventReceived` channels interact with each other to send messages from UDP to connected Websocket clients?

Writing to a Channel

```
eventReceived <- event
```

Reading from Channels

```
func EventLoop() {
	for {
		select {
		case event := <-eventReceived:
			if client, ok := clients[event.UserId]; ok {
				select {
				case client.sendEvent <- event:
				default:
				}
			}
		case c := <-clientAdded:
			clients[c.id] = c
		case c := <-clientDisconnected:
			delete(clients, c.id)
		}
	}
}
```

Qafoo
passion for software quality

Reading from Channels 2

```go
func (c *Client) Start() {
	log.Println("Listening to events")
	clientAdded <- c
	for {
		select {
		case event := <-c.sendEvent:
			log.Printf("Send to %d: %s\n", c.id, event)
			err := websocket.JSON.Send(c.ws, event)

			if err != nil {
				clientDisconnected <- c
				return
			}
		}
	}
}
```

Responsibilities

ListenHttp():

- Send `clientAdded`
- Send `clientDisconnected`
- Block for Websocket

ListenUdp():

- Send `eventReceived`
- Block for UDP connections

EventLoop():

- Listen to `eventReceived`
- Listen to `clientAdded`
- Listen to `clientDisconnected`
- Block for channels

Goroutines

```
func main() {
    // ...
    go ListenHttp()
    go ListenUdp()

    EventLoop()
}
```

- go keyword starts function in a ”lightweight thread”.
- Go runtime executes all goroutines on several threads
- Scheduler automatically selects the goroutines to run
- Beware of concurrent access to shared state

How do I deploy it?

- Use `supervisord` or `monitor` for monitoring
- `scp` the new binary to production servers
- Restart program

Using the Proxy

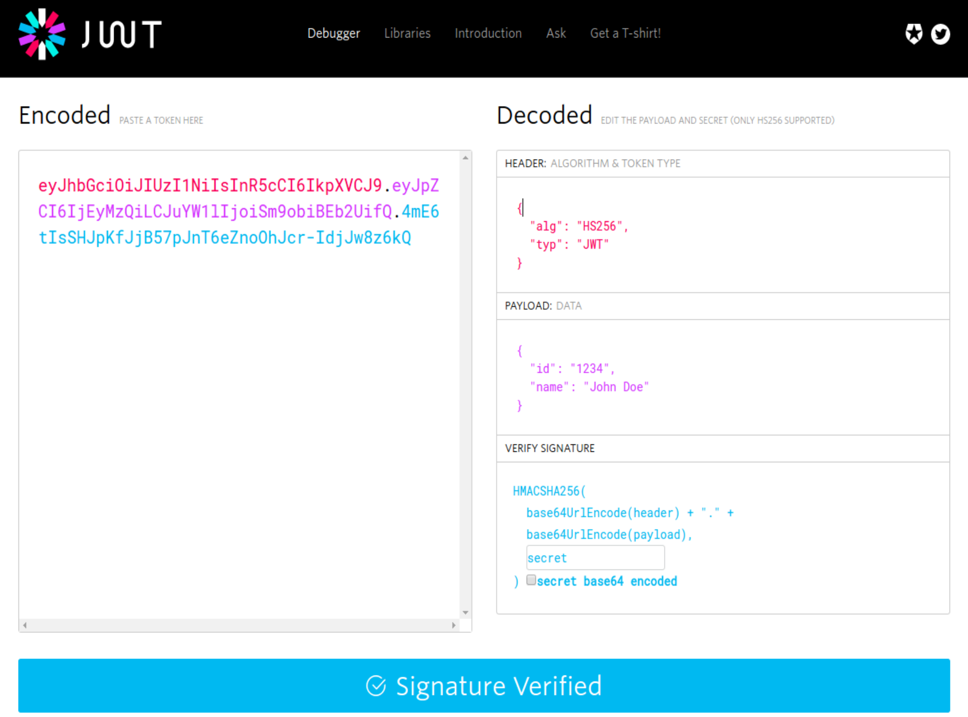
```
<script type="text/javascript" language="javascript">
    var ws = new WebSocket("ws://localhost:8080/ws")
    ws.onmessage = function (message) {
        var event = JSON.parse(message.data)

        switch (event.Name) {
            case "Message":
                document.getElementById("message").
                    innerHTML = event.Payload.text;
                break;
            case "Background":
                document.getElementsByTagName("body")[0].
                    style.backgroundColor = event.Payload.
                    color;
                break;
        }
    }
</script>
```

passion for software quality

Security with JSON Web Token (JWT)

1. Generate a JWT when logging into Symfony app
2. Send JWT token to the client/Browser
3. Browser passes JWT Token over Websocket to authenticate



Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZ
CI6IjEyMzQiLCJuYW1lIjoiSm9obiBEb2UifQ.4mE6
tIsSHJpKfJjB57pJnT6eZnoOhJcr-IdjJw8z6kQ
```

Decoded EDIT THE PAYLOAD AND SECRET (ONLY HS256 SUPPORTED)

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "id": "1234",
  "name": "John Doe"
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret
) ☐ secret base64 encoded
```

Credits

No ElePHPants were harmed during the preparation of this talk.

- Elephpant in the grass by Ulf Wendel
- Elephpant in the trash by Ivo Jansch
- Gopher by Renée French (http://reneefrench.blogspot.de/)

https://joind.in/17011

passion for software quality

THANK YOU

Rent a quality expert
qafoo.com