Community

256 yegor256.com

28 Comments • Created 2 months ago

Java Web App Architecture In Takes Framework

Takes is a pure object-oriented and immutable Java web framework that turns the design and development of web applications into a pleasant and fun process.

(yegor256.com)

28 Comments

Recommend Share Sort by Newest

Join the discussion…

Suseika • 2 months ago

This is a very interesting framework, especially because it incorporates your OO-thinking ideas that I'm really fond of but haven't yet seen much examples of.

But why use an Iterator in Fork when there is a perfectly suitable Optional class that can expressively wrap 0 or 1 value?

• Reply • Share ›

Yegor Bugayenko author ➦ Suseika • 2 months ago

Because we're in Java6, not in Java7 :)

• Reply • Share ›

Suseika ➦ Yegor Bugayenko • 2 months ago

But Optional is really easy to implement yourself if you're so conservative. Iterator seems to be a completely wrong metaphor in this case. It even implies unrelated behavior (.remove()). We're not iterating anything, we're checking if there was some result or if there wasn't, right?

checking if there was some result or if there wasn't, right?

• Reply • Share ›

Yegor Bugayenko author ➦ Suseika • 2 months ago

You're absolutely right, we'll fix this in the next version, see https://github.com/yegor256/ta...

• Reply • Share ›

Suseika ➦ Yegor Bugayenko • 2 months ago

Yay, glad to be of use.

• Reply • Share ›

Михаил Бобруцков • 2 months ago

how much requests per second this thing can handle?

• Reply • Share ›

Yegor Bugayenko author ➦ Михаил Бобруцков • 2 months ago

The "thing" itself doesn't do almost anything. It just dispatches incoming sockets to your objects, in a few parallel threads. The more CPUs you have, the more threads you can have, the more hits per second we can dispatch. The amount of code in Takes framework is so small, comparing to, say, Spring, that it's definitely should work faster. I haven't done any comparisons yet.

• Reply • Share ›

Marcos Douglas Santos ➦ Yegor Bugayenko • 19 days ago

Do you have more tests? Something changed?

• Reply • Share ›

Yegor Bugayenko author ➦ Marcos Douglas Santos • 19 days ago

Not yet, will do something soon... I'll post my results here

1 • Reply • Share ›

Marcos Douglas…

Мих • 2 months ago

ok, ed your example:

```
new FtBasic(
new TkFork(new FkRegex("/", "hello, world!")), 8080
).start(Exit.NEVER);
```

and spring in jetty:

```
@RequestMapping("/")
@ResponseBody
public String hello(ModelMap model) {
return "hello, world!";
}
```

```
ab -n 100000 -c 32 http://localhost:8080/
```

Takes - 6726.03 rps

Spring - 11912.42 rps (15582.29 with keep-alive)

also if I increase connections number to 64 Takes sometimes shuts down with exception like 'java.net.SocketException: Broken pipe'

^ | ˅ • Reply • Share ›

Yegor Bugayenko author ➦ Михаил Бобруцков • 2 months ago

In your example Takes works in a single thread, maybe that's why so slow. Try to run this:

```
new FtBasic(
  new BkParallel(
    new BkBasic(
      new TkFork(new FkRegex("/", "hello, world!"))
    ),
    100 // number of threads
  ),
  8080
).start(Exit.NEVER);
```

^ | ˅ • Reply • Share ›

Михаил Бобруцков ➦ Yegor Bugayenko • 2 months ago

~11.4k, it is still slower than spring

^ | ˅ • Reply • Share ›

Yegor Bugayenko author ➦ Михаил Бобруцков • 2 months ago

Hm... interesting, thanks for the catch. I will try to experiment with it a bit and will make sure we're faster than everybody else :) I'll update you here soon.

^ | ˅ • Reply • Share ›

Marcos Douglas Santos ➦ Yegor Bugayenko • a month ago

Some data about performance and frameworks http://www.infoq.com/news/2015...

^ | ˅ • Reply • Share ›

Михаил Бобруцков ➦ Михаил Бобруцков • 2 months ago

btw, undertow handles about 130k rps on the same laptop

^ | ˅ • Reply • Share ›

Armen Arzumanyan • 2 months ago

Hi, it is perfect! Anyway JSF world is other:) , but, Yegor can we understand takes framework as as middle ware or integration framework?

^ | ˅ • Reply • Share ›

John Fielder • 2 months ago

Hello Yegor. This is a really intuitive framework you've built here, especially after you get your head around the class names. I do have one concern however: can this use multiple your head around the class names. I do have one concern however: can this use multiple cpu cores at the moment? It looks like it can only handle one request at a time, and that wouldn't be very scalable.

^ | ⌄ • Reply • Share ›

Yegor Bugayenko author ➦ John Fielder • 2 months ago

It works with multiple threads, you just need to use BkParallel for that. Default BkBasic is single threaded, but BkParallel decorator makes it multi-threaded. FtCLI uses it by default, if you specify "--threads" command line option. Did I answer the question you want to see a Java example?

^ | ⌄ • Reply • Share ›

John Fielder ➦ Yegor Bugayenko • 2 months ago

That's alright, I understood it. Thanks!

^ | ⌄ • Reply • Share ›

Marcos Douglas Santos • 2 months ago

Hi. I was waiting for this post. Until now beautiful and simple code.

I have some questions:
1- You always said that a class needs to implement a contract (interface) and all public methods of these classes comes from contract, right?
"...RqMethod will help you get the method name from the first line of the header"
Why RqMethod has more public methods than Request interface? This broke your own rules.

2- You said to use inheritance only from abstract classes, but you have many "XxWrap classes", not abstract, and others that inherits it.
Just to not duplicate the "this.origin ..." code?

EDIT:
3- What you say about inheritance of interfaces, you use?
You could creates ResquestMethod, inherited from Request, and code to RqMethod implement it? Don't resolves question #2?

Thanks.

1 ^ | ⌄ • Reply • Share ›

Yegor Bugayenko author ➦ Marcos Douglas Santos • 2 months ago

Dude, you're reading too much of my blog :) Stop it.

1. Yes, you're right, RqMethod violates the rule. And it has to be refactored. It has to be turned into an interface with a class inside: "RqMethod.Base". So, you're absolutely right and thanks for reminding me about it. I'll do this refactoring soon. Or you can submit a pull request.

2. You're right again, these XxWrap classes are against my own principles. But they are some sort of a pattern I haven't found a name for, yet. It's a very convenient pattern that combines embedded anonymous classes and inheritance. I will write about them a bit latter. Now I'm just experimenting, in this framework

about them a bit latter. Now I'm just experimenting, in this framework.

1 ⌃ | ⌄ • Reply • Share ›

Marcos Douglas Santos ➦ Yegor Bugayenko • 2 months ago

Stop reading your blog, serious? :)

I understood XxWrap classes now. At the first time I thought "this.origin" was implemented as protected and used on subclasses but not, origin is private! Very clever.

⌃ | ⌄ • Reply • Share ›

Breno • 2 months ago

Hm... I totally agree with your idea and never liked the fact that Sprint and other frameworks depend so much on XML and reflection. Your solution is also pretty good and I like that, although I have to ask.. have you checked Undertow.io out? It pretty much calls itself an "embeddable" web server and does what you described.
The difference might be that Undertow is non-blocking. You have strong arguments, though.
I just want to know the difference. Other people might want too.

⌃ | ⌄ • Reply • Share ›

Yegor Bugayenko author ➦ Breno • 2 months ago

Yes, I've seen Undertow and many other similar micro HTTP servers. They are nice, but they are not suitable for bigger applications. And they are not object-oriented, as I understand it. They are based on "Handlers" and other anti-patterns. I wanted to create something that would be truly immutable, scalable and composable. Not just yet another Java web server.

1 ⌃ | ⌄ • Reply • Share ›

Oliver Doepner ➦ Yegor Bugayenko • 2 months ago

Yegor, when you say "nice, but they are not suitable for bigger applications" you don't know what you are talking about. Undertow is the HTTP engine in Wildfly, which is the new name of what was once called JBoss Application Server. The JBoss guys have completely thrown out their Tomcat clone that they used to use as HTTP / servlet engine in JBoss AS, and replaced it with Undertow.

If you really think the JBoss team would be stupid enough to pick something that is "not suitable for bigger applications", maybe it is time for a reality check.

⌃ | ⌄ • Reply • Share ›

Oliver Doepner ➦ Oliver Doepner • 2 months ago

By the way, Undertow is neatly split into a core and a separate servlet engine Maven module.

⌃ | ⌄ • Reply • Share ›

Brady ➦ Yegor Bugayenko • 2 months ago

I don't know if I would call "Handlers" an anti-pattern, if you change the

Servlet interface to `HttpResponse handle(HttpRequest)` with immutable request/response it looks a lot like `Take`.

I might be biased since I wrote https://github.com/bradyo/comm... which is all about Handlers.... =)

I don't know much about Undertow, but I think it is more of an issue of how typical servlet applications are put together in non-composable ways using mutable Request/Response objects. I've been porting my PHP code to spray.io and it has a very good immutable HTTP layer build on top of the servlet API: http://spray.io/documentation/...

^ | ⌄ • Reply • Share ›

Oliver Doepner ➦ Brady • 2 months ago

As far as I have seen in the source code, Undertow uses immutable requests. It actually combines the request/response pair as an "exchange" object, similar to Apache Camel. The methods that access the request are purely read-only.

^ | ⌄ • Reply • Share ›

More from yegor256.com

256 yegor256.com recently published

Constructors Must Be Code-Free

68 Comments Recommend

256 yegor256.com recently published

How to Implement an Iterating Adapter

13 Comments Recommend

256 yegor256.com recently published

How Cookie-Based Authentication Works in the Takes Framework

11 Comments Recommend