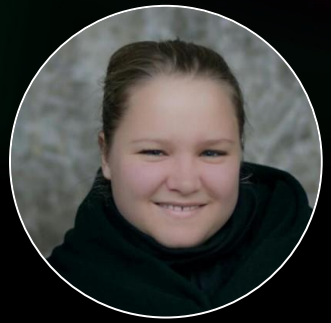
Engineering
Awesome
Conference.

What’s new in Vapor 4



By Heidi Puk Hermann
@HeidiPuk

Engineering Awesome Conference.

Vapor 4: Official release begins

Related Projects Vapor

tanner0101 Tanner Mar 25

Now that Swift 5.2 is here, Vapor 4 has begun releasing official tags for all of its packages. This process starts with the lowest level packages and works up to high level ones.

The first package, AsyncKit, has been tagged:

GitHub 59

vapor/async-kit 59

Sugary extensions for the SwiftNIO library. Contribute to vapor/async-kit development by creating an account on GitHub.

More packages will continue to roll out over the coming days alongside more API and guide docs.

For anyone interested in getting started with Vapor 4 during the final release process, check out this post: [Vapor 4] Release Candidate 495. Release candidate dependencies will automatically upgrade to official releases as they become available.

Check back here for an official post-release getting started guide once all the tags go out.

Prerequisites for Vapor 4
Database models and migrations
Async and database queries
Registering services

Prerequisites to getting started with Vapor 4

A supported OS
Minimum Swift 5.2 installed
Install the new Vapor Toolbox

Database models and migrations

Minimum requirements for a DB model

Engineering Awesome Conference.

```
Import Fluent
Import Vapor

final class User: Model, Content {
    static let schema = "users"

    @ID(key: .id)
    var id: UUID?

    init() {}
}
```

Property wrappers

Engineering Awesome Conference.

```
@propertyWrapper
public final class FieldProperty<Model, Value>
    where Model: FluentKit.Fields, Value: Codable
{
    public let key: FieldKey
    var outputValue: Value?
    var inputValue: DatabaseQuery.Value?

    public var projectedValue: FieldProperty<Model, Value> {
        self
    }

    public var wrappedValue: Value {
        get {
            guard let value = self.value else {
                fatalError("Cannot access field before it is initialized or fetched: \(self.key)")
            }
            return value
        }
        set {
            self.value = newValue
        }
    }

    public init(key: FieldKey) {
        self.key = key
    }
}
```

Bringing life to your models

Engineering Awesome Conference.

@Field / @OptionalField

@TimeStamp
@Enum

@Children
@Parent / @OptionalParent
@Siblings

@Group

```
final class User: Model, Content {
    static let schema = "users"

    @ID
    var id: UUID?

    @Field(key: "username")
    var username: String

    init() {}

    init(id: UUID? = nil, username: String) {
        self.id = id
        self.username = username
    }
}
```

Migrations

Migrations are like a version control system for your database. Each migration defines a change to the database and how to undo it.

Vapor 2

Engineering Awesome Conference.

```
extension User: Preparation {
    /// Prepares a table/collection in the database
    /// for storing Users
    static func prepare(_ database: Database) throws {
        try database.create(self) { builder in
            builder.id()
            builder.string("username")
        }
    }

    /// Undoes what was done in `prepare`
    static func revert(_ database: Database) throws {
        try database.delete(self)
    }
}
```

Vapor 3

Engineering Awesome Conference.

```
extension User: Migration {}
```

Vapor 4

Engineering Awesome

ence.

```
struct CreateUser: Migration {
    let model = User()

    func prepare(on database: Database) -> EventLoopFuture<Void> {
        database.schema(User.schema)
            .id()
            .field(model.$username.key, .string, .required)
            .create()
    }

    func revert(on database: Database) -> EventLoopFuture<Void> {
        database.schema(User.schema).delete()
    }
}
```

Async - Vapor 3

Engineering Awesome Conference.

```
func postalOfficesHandler(req: Request) -> Future<View> {
    PostalOffice
        .query(on: req)
        .paginate(for: req)
        .map { postalOffices in
            try PostalOfficesContext(
                title: "Postal Offices",
                postalOffices: postalOffices,
                isLoggedIn: req.isAuthenticated(User.self),
                user: req.authenticated(User.self)
            )
        }
        .flatMap { viewContext in
            try req.leaf().render("postalOffices", viewContext)
        }
}
```

Async - Vapor 4

Engineering Awesome Conference.

method	argument	description
`map`	`(T) -> U`	Maps a future value to a different value.
`flatMapThrowing`	`(T) throws -> U`	Maps a future value to a different value or an error.
`flatMap`	`(T) -> EventLoopFuture<U>`	Maps a future value to different *future* value.
`transform`	`U`	Maps a future to an already available value.

Async - Vapor 4

Engineering Awesome Conference.

```
func create(req: Request) throws -> EventLoopFuture<User> {
    let user = try req.content.decode(User.self)
    return User.query(on: req.db)
        .filter(\.$username, .equal, user.username)
        .first()
        .flatMapThrowing { existingUser -> Void in
            guard existingUser == nil else {
                throw Abort(.badRequest, reason: "Username already taken")
            }
            return ()
        }
        .flatMap { _ in
            user.save(on: req.db)
        }
        .transform(to: user)
}
```

Database queries

Engineering Awesome Conference.

```
func create(req: Request) throws -> EventLoopFuture<User> {
    let user = try req.content.decode(User.self)
    return User.query(on: req.db)
        .filter(\.$username, .equal, user.username)
        .first()
        .flatMapThrowing { existingUser -> Void in
            guard existingUser == nil else {
                throw Abort(.badRequest, reason: "Username already taken")
            }
            return ()
        }
        .flatMap { _ in
            user.save(on: req.db)
        }
        .transform(to: user)
}
```

Database Queries

Engineering Awesome Conference.

```
func index(req: Request) -> EventLoopFuture<Page<User>> {
    User.query(on: req.db)
        .with(\.$todos)
        .paginate(for: req)
}
```

Database Queries

Engineering Awesome Conference.

```
// Fetches all planets with a star named Sun.
Planet.query(on: database)
    .join(Star.self, on: \Planet.$star.$id == \Star.$id)
    .filter(Star.self, \.$name == "Sun")
    .all()

// Accessing joined model from query result.
let planet: Planet = ...
let star = try planet.joined(Star.self)
```

How to register services

Engineering Awesome Conference.

```
struct MyAPI {
    let client: Client
    func foos() -> EventLoopFuture<[String]> { ... }
}

extension Request {
    var myAPI: MyAPI {
        .init(client: self.client)
    }
}

req.myAPI.foos()
```

Honorable mentions

XCTVapor - https://docs.vapor.codes/4.0/testing/

Queues - https://docs.vapor.codes/4.0/queues/

Leaf

Update to the toolbox - https://github.com/vapor/toolbox

Logging - https://github.com/apple/swift-log

Engineering
Awesome
Conference.

Thank you

By Heidi Puk Hermann
@HeidiPuk

Vapor docs: https://docs.vapor.codes/4.0/
Vapor discord channel: https://discord.com/invite/vapor

References:

https://docs.vapor.codes/4.0/install/macos/
https://docs.vapor.codes/4.0/install/linux/
https://github.com/apple/swift-evolution/blob/master/proposals/0258-property-wrappers.md
https://docs.vapor.codes/4.0/fluent/model/
https://docs.vapor.codes/4.0/fluent/migration/
https://docs.vapor.codes/4.0/async
https://apple.github.io/swift-nio/docs/current/NIO/Classes/EventLoopFuture.html
https://docs.vapor.codes/4.0/fluent/query/
https://docs.vapor.codes/4.0/services/
https://docs.vapor.codes/4.0/testing/
https://docs.vapor.codes/4.0/queues/
https://github.com/vapor/toolbox
https://github.com/apple/swift-log