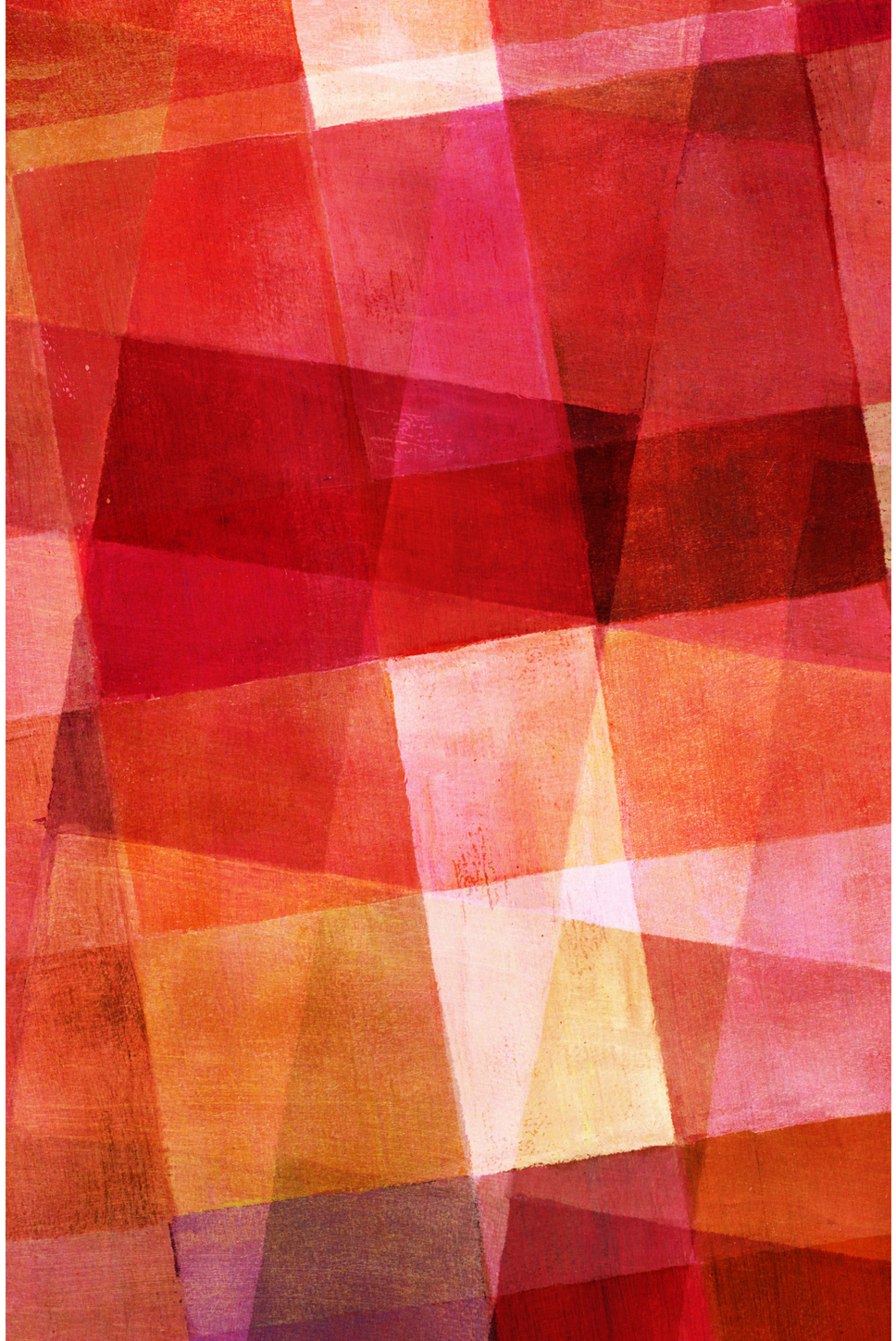
SPRING BOOT VS. .NET CORE

It's all about the APIs



AGENDA

- Prerequisites
- Dependencies management
- Creating a new api
- Create CRUD Controller
- Validation
- Error handling
- Database access
- Call another API
- Scheduled Jobs
- Testing controllers
- Memory footprint
- Swagger
- Docker
- Deploy to heroku

README.md

.Net Core WebApi vs Spring Boot

This is a prezentation about the 2 technologies and it's purpose is to higlight the similarities and differences between them.

The demo project is a Todos API with basic endpoints and a postgresql database. It exxplains the controller, model, service, validation, error handling, api to api call, scheduled jobs, testing, memory footprint and docker files.

Prezentation can be found here: Open slides

Bitly short url for all files: http://bit.ly/core-vs-spring

It also contains 2 projects with the same features for a better comparison.

Here are the projects runing live:

- dotnet core 2/3 REST webapi .net core webapi link
- spring boot 2 REST api spring boot api link

The keynote prezentation link is here: open me

In each project there is a Dockerfile and a Docker-compose file so that it can be executed locally. The docker file is expecting a postgres installation to run on your localhost port 5432. For the docker-compose there is no other requirement than the docke to be prezent.

NJoy

PREREQUISITES

Install Java

https://www.java.com/ES/download/

Install .Net Core

https://dotnet.microsoft.com/download

DEPENDENCIES MANAGEMENT

Apache Maven Project
http://maven.apache.org/

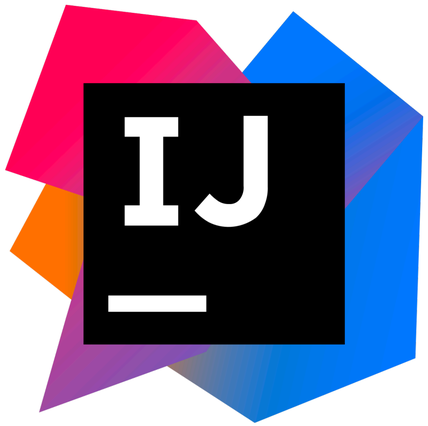
Maven™

Apache / Maven / Welcome to Apache Maven

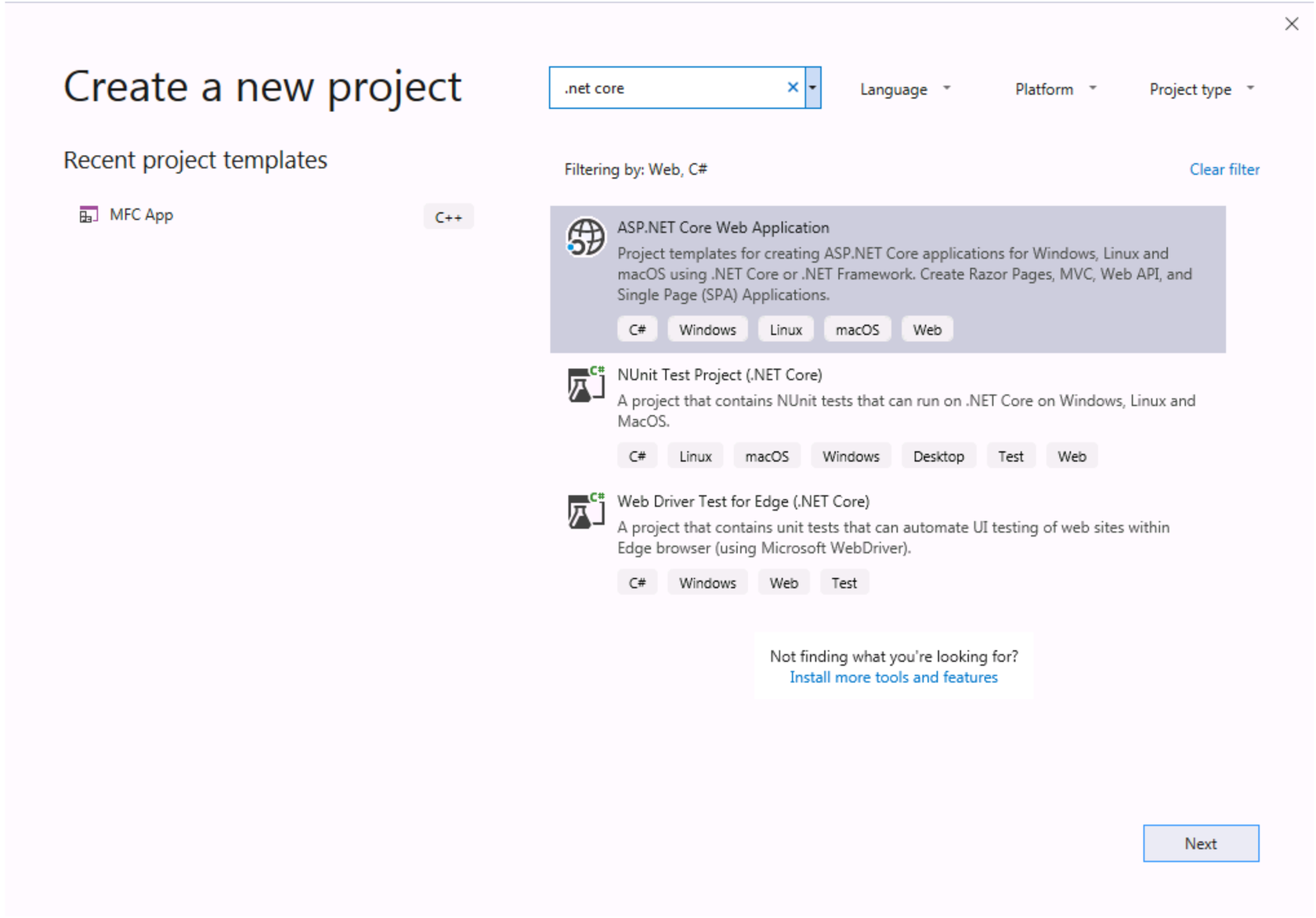
Download | Get Sources | Last Published: 2019-10-08

INTEGRATED DEVELOPMENT ENVIRONMENT: DOT NET

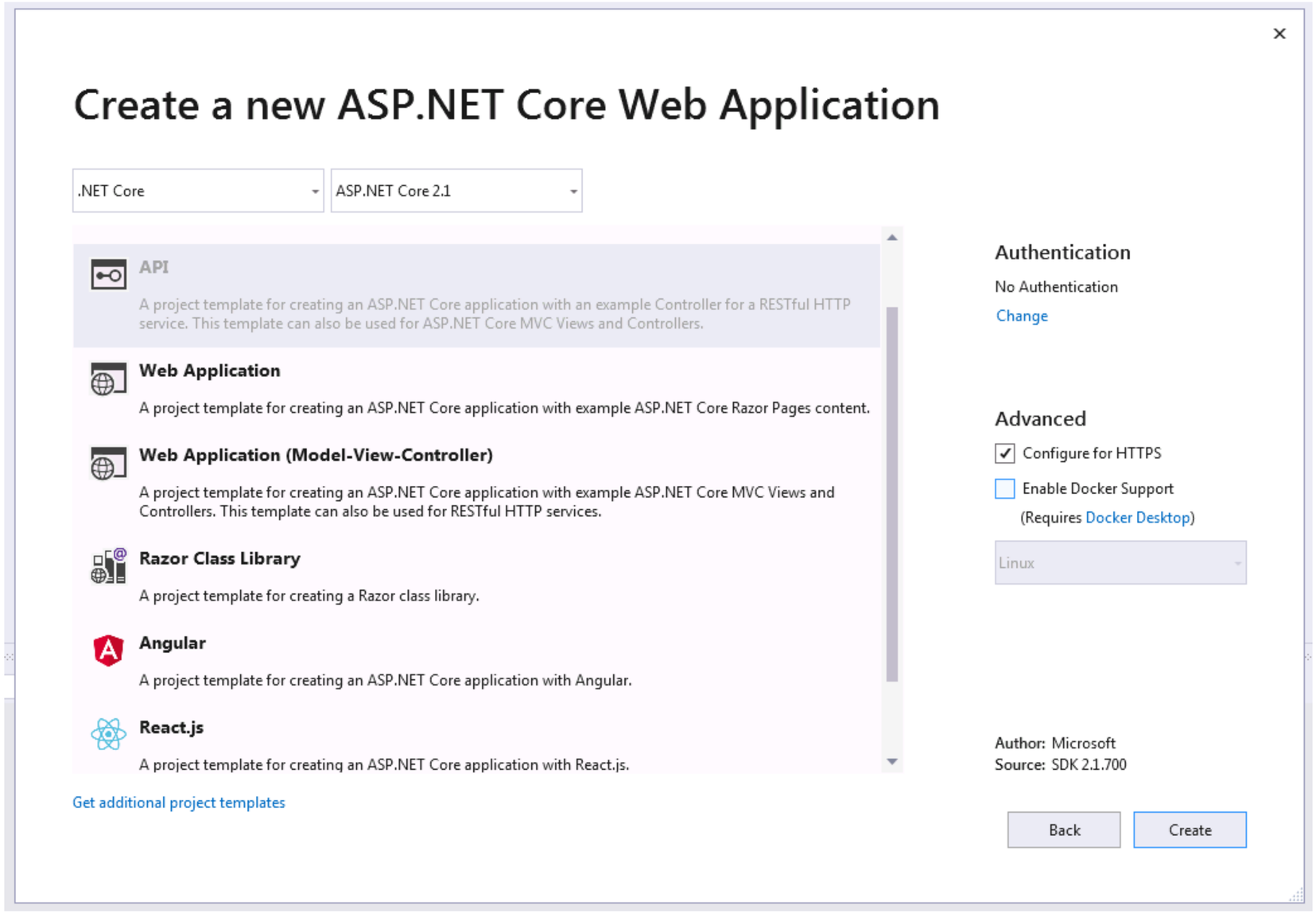
INTEGRATED DEVELOPMENT ENVIRONMENT: JAVA



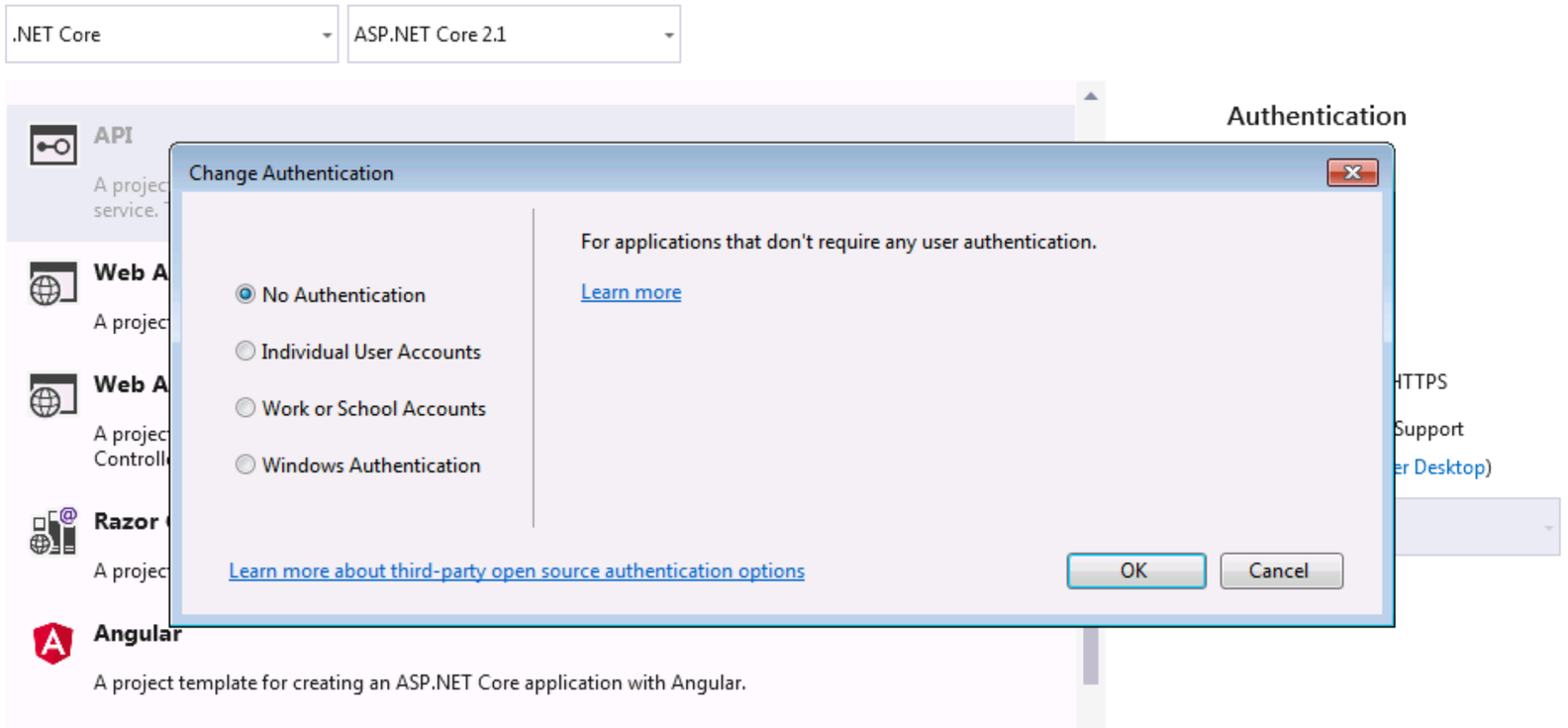
CREATE NEW API – .NET CORE



CREATE NEW API – .NET CORE



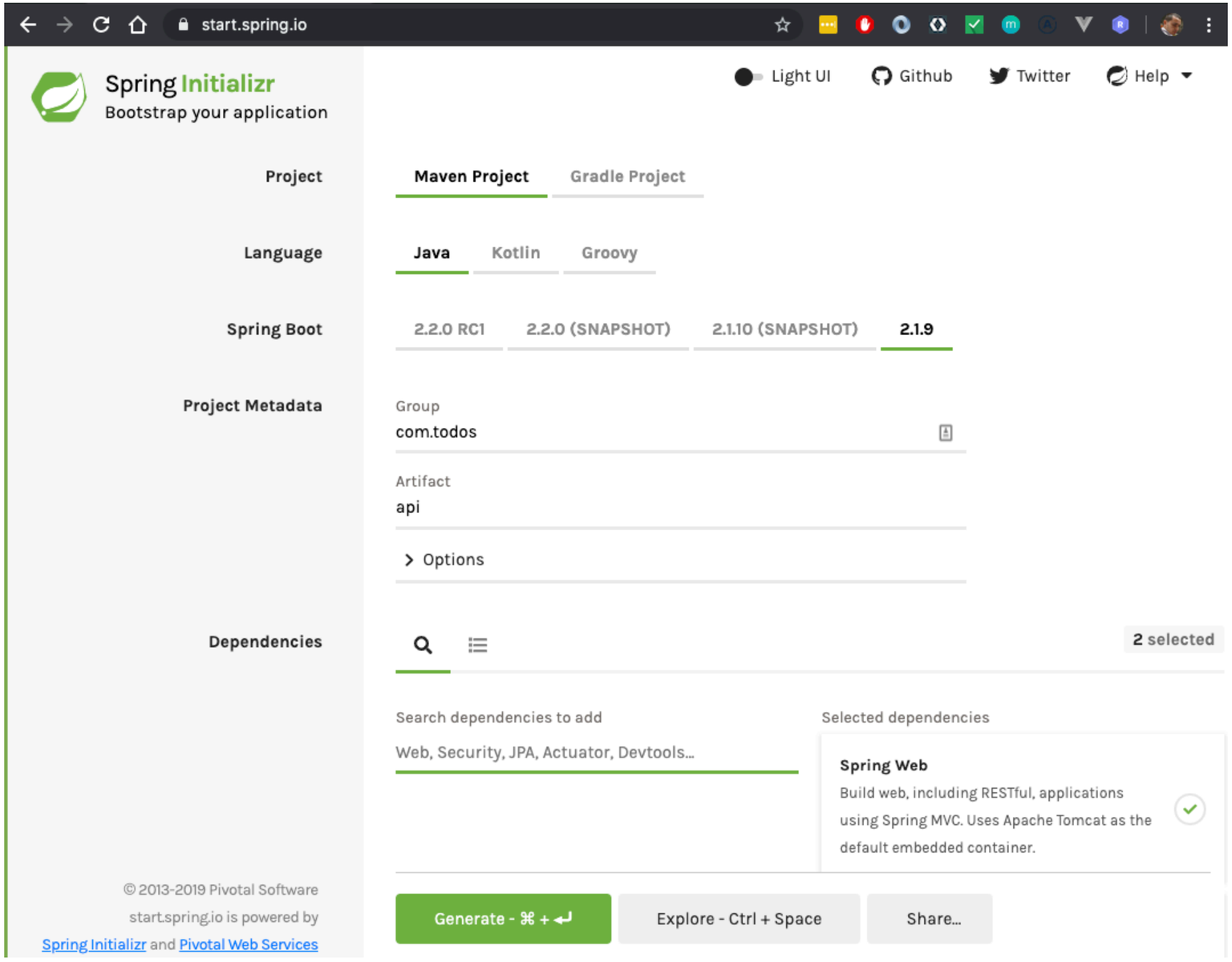
CREATE NEW API – .NET CORE



CREATE NEW API - .NET CORE

```
➜  ~ dotnet new
Usage: new [options]

Options:
  -h, --help          Displays help for this command.
  -l, --list          Lists templates containing the specified name. If no name is specified, lists all templates.
  -n, --name          The name for the output being created. If no name is specified, the name of the current directory is used.
  -o, --output        Location to place the generated output.
  -i, --install       Installs a source or a template pack.
  -u, --uninstall     Uninstalls a source or a template pack.
  --nuget-source      Specifies a NuGet source to use during install.
  --type              Filters templates based on available types. Predefined values are "project", "item" or "other".
  --dry-run           Displays a summary of what would happen if the given command line were run if it would result in a template creation.
  --force             Forces content to be generated even if it would change existing files.
  -lang, --language   Filters templates based on language and specifies the language of the template to create.
  --update-check      Check the currently installed template packs for updates.
  --update-apply      Check the currently installed template packs for update, and install the updates.


Templates                                 Short Name               Language          Tags
--------------------------------------------------------------------------------------------------------------------
Console Application                       console                  [C#], F#, VB      Common/Console
Class library                             classlib                 [C#], F#, VB      Common/Library
WPF Application                           wpf                      [C#]              Common/WPF
WPF Class library                         wpflib                   [C#]              Common/WPF
WPF Custom Control Library                wpfcustomcontrollib      [C#]              Common/WPF
WPF User Control Library                  wpfusercontrollib        [C#]              Common/WPF
Windows Forms (WinForms) Application      winforms                 [C#]              Common/WinForms
Windows Forms (WinForms) Class library    winformslib              [C#]              Common/WinForms
Worker Service                            worker                   [C#]              Common/Worker/Web
Unit Test Project                         mstest                   [C#], F#, VB      Test/MSTest
NUnit 3 Test Project                      nunit                    [C#], F#, VB      Test/NUnit
NUnit 3 Test Item                         nunit-test               [C#], F#, VB      Test/NUnit
xUnit Test Project                        xunit                    [C#], F#, VB      Test/xUnit
Razor Component                           razorcomponent           [C#]              Web/ASP.NET
Razor Page                                page                     [C#]              Web/ASP.NET
```

CREATE NEW API – SPRING BOOT: START.SPRING.IO



DEPENDENCIES

Selected dependencies

Spring Web
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

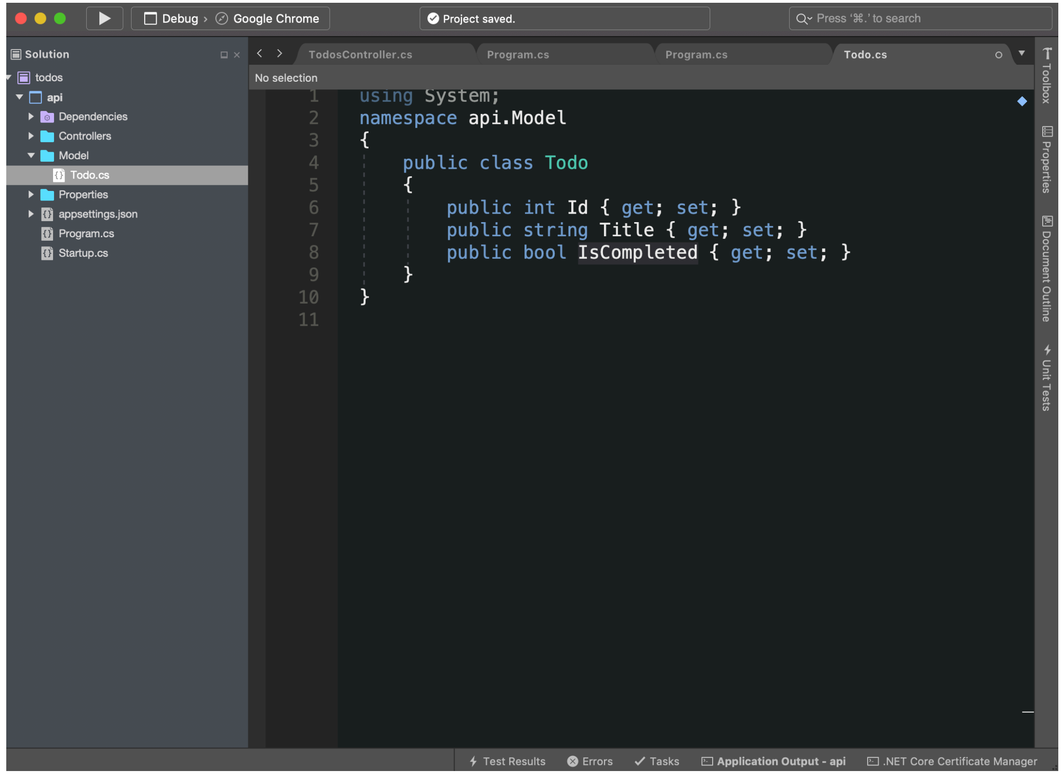
Spring Data JPA
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

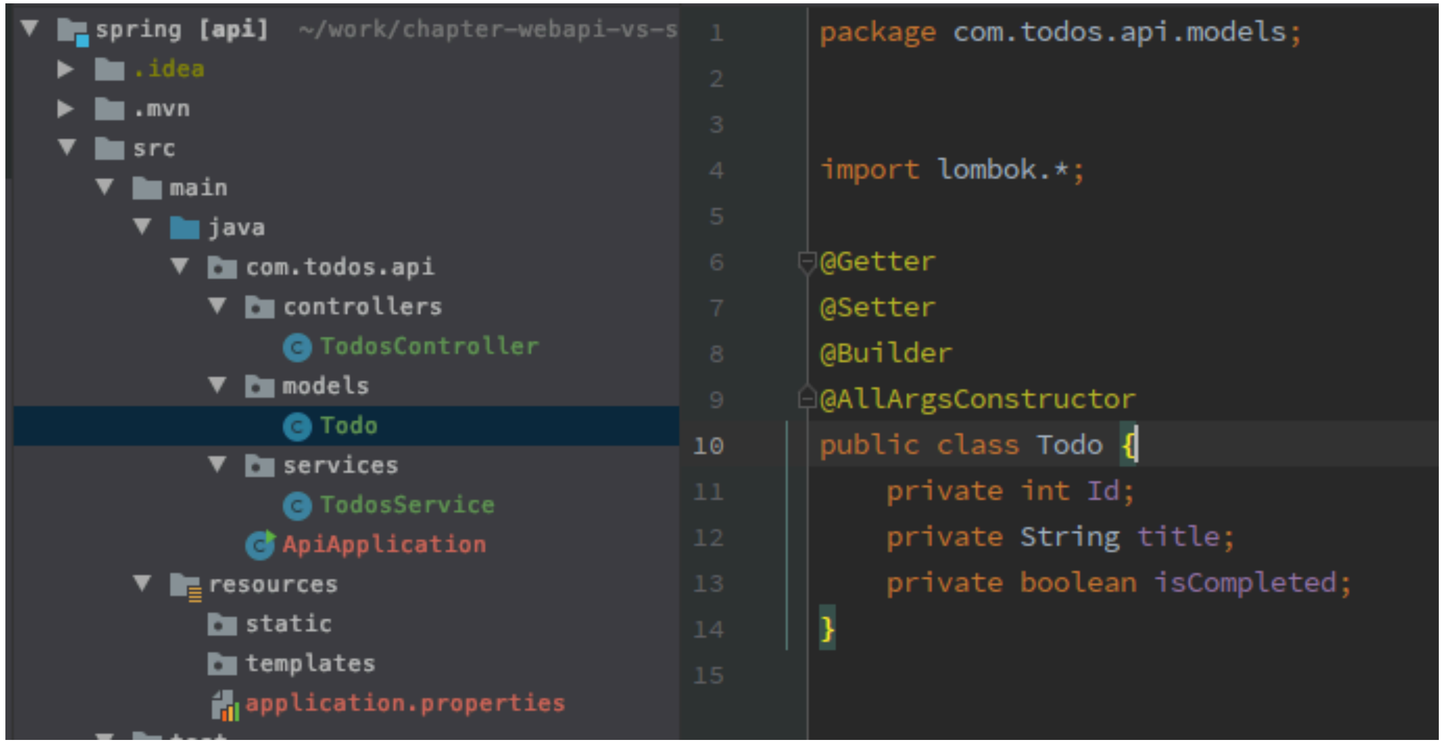
Spring Boot DevTools
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

PostgreSQL Driver
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

Lombok
Java annotation library which helps to reduce boilerplate code.

TODO MODEL C# | JAVA





DOMAIN VALIDATION C#

```
public class Todo
{
    [Required]
    10 references
    public int Id { get; set; }
    [Required]
    [MinLength(2, ErrorMessage = "Minimum length allowed is 2")]
    8 references
    public string Title { get; set; }
```

```
// POST api/todos
[HttpPost]
0 references
public ActionResult Post([FromBody]Todo todo)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);
    _service.Create(todo);
    return Created("", todo);
}
```

Built-in attributes

Here are some of the built-in validation attributes:

- `[CreditCard]`: Validates that the property has a credit card format.
- `[Compare]`: Validates that two properties in a model match.
- `[EmailAddress]`: Validates that the property has an email format.
- `[Phone]`: Validates that the property has a telephone number format.
- `[Range]`: Validates that the property value falls within a specified range.
- `[RegularExpression]`: Validates that the property value matches a specified regular expression.
- `[Required]`: Validates that the field is not null. See [Required] attribute for details about this attribute's behavior.
- `[StringLength]`: Validates that a string property value doesn't exceed a specified length limit.
- `[Url]`: Validates that the property has a URL format.
- `[Remote]`: Validates input on the client by calling an action method on the server. See [Remote] attribute for details about this attribute's behavior.

```
public class ValidEmailDomainAttribute : ValidationAttribute
{
    private readonly string allowedDomain;

    public ValidEmailDomainAttribute(string allowedDomain)
    {
        this.allowedDomain = allowedDomain;
    }

    public override bool IsValid(object value)
    {
        string[] strings = value.ToString().Split('@');
        return strings[1].ToUpper() == allowedDomain.ToUpper();
    }
}
```

DOMAIN VALIDATION JAVA

JSR 380 is a specification of the Java API for bean validation, part of JavaEE and JavaSE, which ensures that the properties of a bean meet specific criteria, using annotations such as *@NotNull*, *@Min*, and *@Max*.

```java
public class Todo {
    @Id @GeneratedValue
    private int Id;
    @NotNull
    @Min(value = 2,message = "Minimum length allowed is 2")
    private String title;
```

```java
@PostMapping(value = "/api/todos")
public ResponseEntity<?> post(@Valid @RequestBody Todo todo){
    _service.create(todo);
    return  ResponseEntity.created( location: null).body(todo);
}
```


```
{
  "timestamp": "2019-11-20T15:04:53.583+0000",
  "status": 400,
  "error": "Bad Request",
  "errors": [
    {
      "codes": [],
      "arguments": [],
      "defaultMessage": "Minimum length allowed is 2",
      "objectName": "todo",
      "field": "title",
      "rejectedValue": "s",
      "bindingFailure": false,
      "code": "Min"
    }
  ],
  "message": "Validation failed for object='todo'. Error count: 1",
```


Per the JSR 380 specification, the *validation-api* dependency contains the standard validation APIs.

Hibernate Validator is the reference implementation of the validation API.

All of the annotations used in the example are standard JSR annotations:

- ***@NotNull*** – validates that the annotated property value is not *null*
- ***@AssertTrue*** – validates that the annotated property value is *true*
- ***@Size*** – validates that the annotated property value has a size between the attributes *min* and *max*; can be applied to *String*, *Collection*, *Map*, and array properties
- ***@Min*** – *V*alidates that the annotated property has a value no smaller than the *value* attribute
- ***@Max*** – validates that the annotated property has a value no larger than the *value* attribute
- ***@Email*** – validates that the annotated property is a valid email address

Some annotations accept additional attributes, but the *message* attribute is common to all of them. This is the message that will usually be rendered when the value of the respective property fails validation.

Some additional annotations that can be found in the JSR are:

- ***@NotEmpty*** – validates that the property is not null or empty; can be applied to *String*, *Collection*, *Map* or *Array* values
- ***@NotBlank*** – can be applied only to text values and validated that the property is not null or whitespace
- ***@Positive*** and ***@PositiveOrZero*** – apply to numeric values and validate that they are strictly positive, or positive including 0
- ***@Negative*** and ***@NegativeOrZero*** – apply to numeric values and validate that they are strictly negative, or negative including 0
- ***@Past*** and ***@PastOrPresent*** – validate that a date value is in the past or the past including the present; can be applied to date types including those added in Java 8
- ***@Future*** *and* ***@FutureOrPresent*** – validates that a date value is in the future, or in the future including the present

CUSTOM DOMAIN VALIDATION JAVA

4. The New Annotation

Let's create a new *@interface* to define our annotation:

```
@Documented
@Constraint(validatedBy = ContactNumberValidator.class)
@Target( { ElementType.METHOD, ElementType.FIELD })
@Retention(RetentionPolicy.RUNTIME)
public @interface ContactNumberConstraint {
    String message() default "Invalid phone number";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

5. Creating a Validator

Let's now create a validator class that enforces rules of our validation:

```
public class ContactNumberValidator implements
  ConstraintValidator<ContactNumberConstraint, String> {

    @Override
    public void initialize(ContactNumberConstraint contactNumber) {
    }

    @Override
    public boolean isValid(String contactField,
      ConstraintValidatorContext cxt) {
        return contactField != null && contactField.matches("[0-9]+")
          && (contactField.length() > 8) && (contactField.length() < 14);
    }

}
```

5. Programmatic Validation

Some frameworks – such as Spring – have simple ways of triggering the validation process by just using annotations. This is mainly so that we don't have to interact with the programmatic validation API.

Let's now go the manual route and set things up programmatically:

```
ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
Validator validator = factory.getValidator();
```

```
Set<ConstraintViolation<User>> violations = validator.validate(user);
```

By iterating over the violations, we can get all the violation messages by using the *getMessage* method.

```
for (ConstraintViolation<User> violation : violations) {
    log.error(violation.getMessage());
}
```

TODO SERVICE C# | JAVA

```csharp
namespace api.Servcies
{
    public class TodosService
    {
        List<Todo> _todos = new List<Todo> {
        new Todo{Id=1, Title="First todo!", IsCompleted = false} };

        public List<Todo> GetAll() => _todos;
        public Todo Get(int id) => _todos.FirstOrDefault(x => x.Id == id);
        public void Create(Todo todo) {
            todo.Id = _todos.Count + 1;
            _todos.Add(todo);
        }

        public void Update(int id,Todo todo)
        {
            var td = _todos.FirstOrDefault(x => x.Id == id);
            if (td != null)
            {
                td.Title = todo.Title;
                td.IsCompleted = todo.IsCompleted;
            }
        }
        public void Delete(int id) =>
            _todos.Remove(_todos.Single(x => x.Id == id));
    }
```

```java
@Service
public class TodosService {
    private List<Todo> _todos = new ArrayList<>();

    {
        _todos. add(new Todo( Id: 1, title: "First todo",  isCompleted: false));;
    };

    public List<Todo> GetAll(){ return _todos;}
    public Todo Get(int id){
        return _todos.stream().filter(x->x.getId() == id).findFirst().get();}

    public Todo create(Todo todo) {
        todo.setId(_todos.size() + 1);
        _todos.add(todo);
        return todo;
    }

    public Todo update(int id, Todo todo) {

        Todo td = _todos.stream().filter(x-> x.getId() == id)
                .findFirst().get();
        if (td != null){
            td.setCompleted(todo.isCompleted());
            td.setTitle(todo.getTitle());
        }
        return td;
    }

    public void delete(int id){
        _todos.remove(
                _todos.stream().filter(x-> x.getId() == id)
                .findFirst().get());
    }
}
```

TODOS CONTROLLER C# | JAVA

```c#
[Route("api/[controller]")]
public class TodosController : Controller
{
    public TodosService _service { get; }

    public TodosController(TodosService service)
        => _service = service;

    // GET: api/todos
    [HttpGet]
    public IEnumerable<Todo> Get() => _service.GetAll();

    // GET api/todos/5
    [HttpGet("{id}")]
    public ActionResult<Todo> Get(int id) => Ok(_service.Get(id));

    // POST api/todos
    [HttpPost]
    public ActionResult Post([FromBody]Todo todo)
    {
        _service.Create(todo);
        return Created("", todo);
    }

    // PUT api/todos/5
    [HttpPut("{id}")]
    public ActionResult Put(int id, [FromBody]Todo todo)
    {
        _service.Update(id,todo);
        return NoContent();
    }

    // DELETE api/todos/5
    [HttpDelete("{id}")]
    public ActionResult Delete(int id)
    {
        _service.Delete(id);
        return NoContent();
    }
```

```java
@RestController
//@RequestMapping()
public class TodosController {

    @Autowired
    TodosService _service;

    @GetMapping(value = "/api/todos")
    public ResponseEntity< List<Todo>> get(){
        return  ResponseEntity.ok(  _service.GetAll());
    }
    @GetMapping(value = "/api/todos/{id}")
    public ResponseEntity<?> get(@PathVariable Integer id){
        return  ResponseEntity.ok(_service.Get(id));
    }

    @PostMapping(value = "/api/todos")
    public ResponseEntity<?> post(@RequestBody Todo todo){
        _service.create(todo);
        return  ResponseEntity.created( location: null).body(todo);
    }

    @PutMapping(value = "/api/todos/{id}")
    public ResponseEntity<Todo> put(@PathVariable Integer id, @RequestBody Todo todo){
        Todo td = _service.update(id,todo);
        return ResponseEntity.ok(td);
    }

    @DeleteMapping(value = "/api/todos/{id}")
    public ResponseEntity<?> delete(@PathVariable Integer id){
        _service.delete(id);
       return ResponseEntity.accepted().build();
    }
}
```

TODOS ERROR HANDLING: C#

```
1 usage
public class Startup
{
    public Startup(IConfiguration configuration){...}
    1 usage
    public IConfiguration Configuration { get; }
    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services){...}
    // This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
    public void Configure(IApplicationBuilder app, IHostingEnvironment env)
    {

        if (env.IsDevelopment())
            app.UseDeveloperExceptionPage();

        app.ConfigureExceptionHandler();
```

```
1 usage
public static void ConfigureExceptionHandler(this IApplicationBuilder app)
{
    app.UseExceptionHandler(appError =>
    {
        appError.Run( handler: async context =>
        {
            context.Response.StatusCode = (int)HttpStatusCode.InternalServerError;
            context.Response.ContentType = "application/json";

            var contextFeature = context.Features.Get<IExceptionHandlerFeature>();
            if (contextFeature != null)
            {
                Trace.WriteLine( message: $"Something went wrong: {contextFeature.Error}");

                await context.Response.WriteAsync(
                    text: JsonConvert.SerializeObject(new {
                                StatusCode = (int)context.Response.StatusCode,
                                Message = "Internal Server Error."
                            })
                        );
            }
        });
    });
```

TODOS ERROR HANDLING: JAVA

2. Solution 1 – The Controller level *@ExceptionHandler*

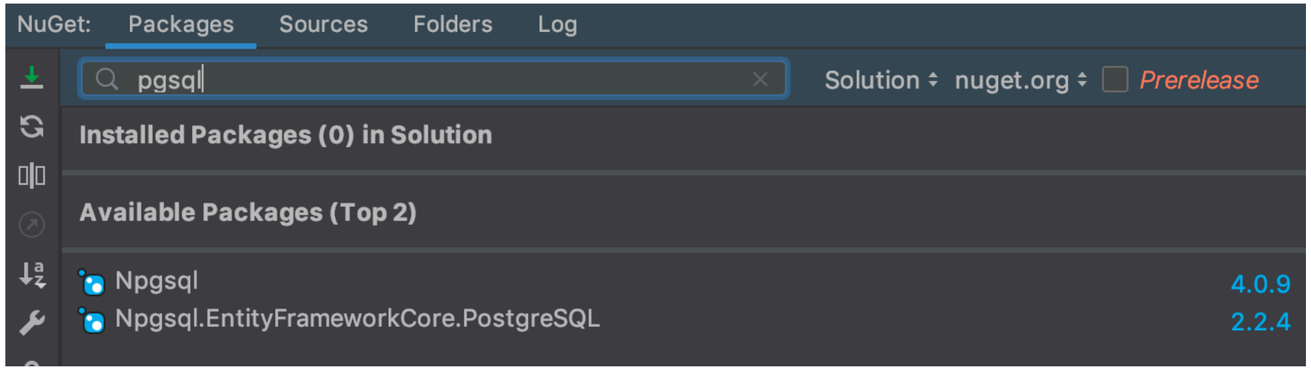
The first solution works at the *@Controller* level – we will define a method to handle exceptions, and annotate that with *@ExceptionHandler*:

```
public class FooController{

    //...
    @ExceptionHandler({ CustomException1.class, CustomException2.class })
    public void handleException() {
        //
    }
}
```

```
@Data
@Builder
class JSONException{
    private String message;
}

@RestControllerAdvice
public class TodosControllerAdvice {

    @ExceptionHandler(TodoException.class)
    @ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)
    public ResponseEntity<?> handleNonExistingHero(TodoException exp) {
        return new ResponseEntity<>( new JSONException( exp.getMessage()), HttpStatus.INTERNAL_SERVER_ERROR);
    }
```

AUTHENTICATION & AUTHORIZATION

DATABASE ACCESS C#




```
2 usages
public class AppDb: DbContext
{
    public AppDb(DbContextOptions<AppDb> options): base(options){}

    public DbSet<Todo> Todos { get; set; }
}
```

```
1 usage
public class Startup
{
    public Startup(IConfiguration configuration){...}
    2 usages
    public IConfiguration Configuration { get; }
    // This method gets called by the runtime. Use this method to add services to the container.
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddDbContext<AppDb>( optionsAction: options =>
            options.UseNpgsql(Configuration.GetConnectionString( name: "DefaultConnection")));
```

DATABASE ACCESS JAVA

pom.xml

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
    <groupId>org.postgresql</groupId>
    <artifactId>postgresql</artifactId>
    <scope>runtime</scope>
</dependency>
```

application.properties

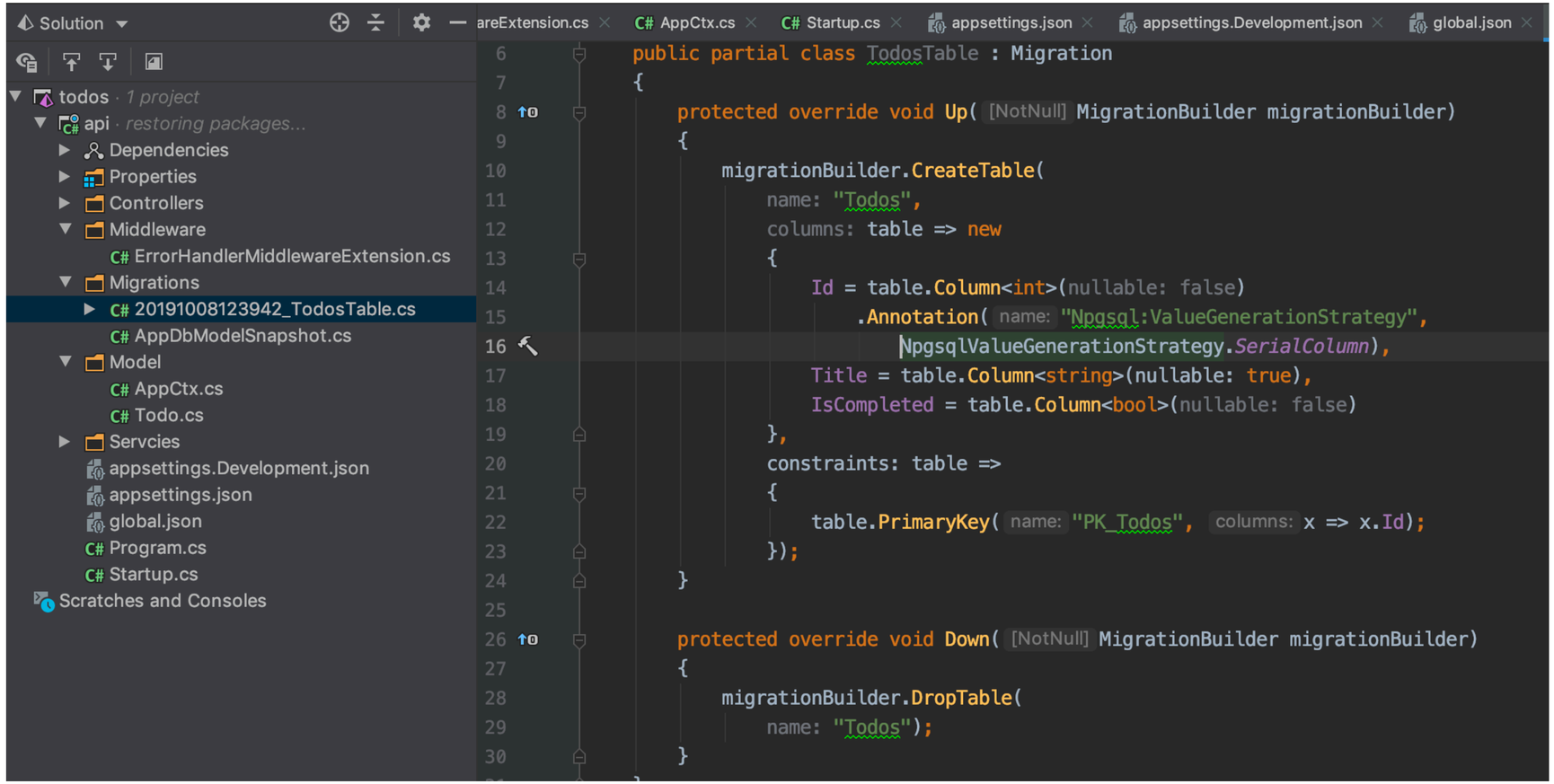
```
spring.datasource.url= jdbc:postgresql://localhost:5432/todos_java
spring.datasource.username=postgres
spring.datasource.password=postgres
spring.jpa.hibernate.ddl-auto=update
spring.datasource.driver-class-name=org.postgresql.Driver

spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
```

```java
package com.todos.api.models;


import lombok.*;
import javax.persistence.*;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@Entity
public class Todo {
    @Id @GeneratedValue
    private int Id;
    private String title;
    private boolean isCompleted;
}

```

```java
package com.todos.api.repositories;

import com.todos.api.models.Todo;
import org.springframework.data.jpa.repository.JpaRepository;

public interface TodosRepository extends JpaRepository<Todo,Integer> {
}
```

CREATE MIGRATIONS: C#

➜ *dotnet ef migrations add TodosTable*



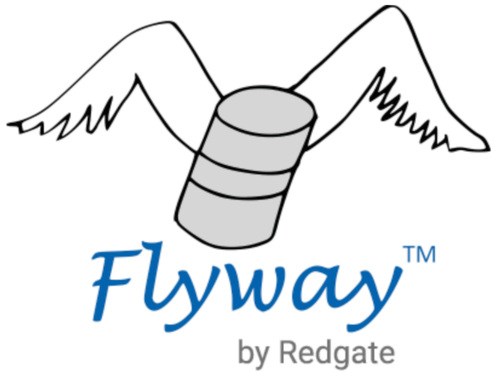
➜ *dotnet ef database update*

AUTOUPDATE DATABASE: C#

Inside the Startup class

```
  // update database schema
using (var serviceScope = app.ApplicationServices.GetRequiredService<IServiceScopeFactory>().CreateScope())
{
    if (serviceScope.ServiceProvider.GetService<AppDb>() == null) return;
    var ctx = serviceScope.ServiceProvider.GetService<AppDb>();
    new DatabaseFacade(ctx).Migrate();
}
```

DATABASE MIGRATIONS: JAVA



Version control for your database.

Robust schema evolution across all your environments.

With ease, pleasure and plain SQL.

LIQUIBASE

The leading open source tool for database change and deployment management.

Which Option is Right for You?

USE THE DATABASE: TODOSSERVICE C# | JAVA

```csharp
public class TodosService
{
    private readonly AppDb _db;

    usages  overrides  dragosh
    public TodosService(){}
    usages  overrides  dragosh
    public TodosService(AppDb db) => _db = db;

    usages  overrides  dragosh
    public virtual List<Todo> GetAll() =>
        _db.Todos.OrderByDescending(x=>x.Id).ToList();
    usages  overrides  dragosh
    public Todo Get(int id) =>
        _db.Todos.Single(x => x.Id == id);

    usages  overrides  dragosh
    public void Create(Todo todo) {
        _db.Todos.Add(todo);
        _db.SaveChanges();
    }

    usages  overrides  dragosh
    public Todo Update(int id,Todo todo)
    {
        var td = _db.Todos.FirstOrDefault(x => x.Id == id);
        if (td != null)
        {
            td.Title = todo.Title;
            td.IsCompleted = todo.IsCompleted;
            _db.SaveChanges();
        }
        return td;
    }

    usages  overrides  dragosh
    public void Delete(int id)
    {
        _db.Todos.Remove(new Todo {Id = id});
        _db.SaveChanges();
    }
}
```

```java
@Service
public class TodosService {

    private final TodosRepository _db;

    public TodosService(TodosRepository repository) { _db = repository; }

    public List<Todo> GetAll(){ return _db.findAll();}
    public Optional<Todo> Get(int id){
        return _db.findById(id);}

    public Todo create(Todo todo) {
        todo = _db.saveAndFlush(todo);
        return todo;
    }

    public Todo update(int id, Todo todo) {

        Optional<Todo> td =  _db.findById(id);
        if (td.isPresent()){
            td.get().setCompleted(todo.isCompleted());
            td.get().setTitle(todo.getTitle());
            _db.saveAndFlush(td.get());
        }
        return td.get();
    }

    public void delete(int id){
        _db.deleteById(id);
    }
}
```

CALL EXTERNAL API: C# | JAVA

```csharp
public class ShowsController : Controller
{       dragosh, 4 hours ago • webapi wip
    [HttpGet]
    0 references
    public async Task<ActionResult> Get()
    {
        var client = new HttpClient();
        var result = await client.GetAsync("http://jsnoise.herokuapp.com/api/showslist");
        return Ok(await result.Content.ReadAsStringAsync());
    }
}
```

```java
package com.todos.api.controllers;

import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.client.RestTemplate;

@RestController
public class ShowsController {

    @GetMapping(value = "api/shows")
    public ResponseEntity<?> get(){
        RestTemplate restTemplate = new RestTemplate();
        return ResponseEntity.ok(
                restTemplate
                .getForEntity( url: "http://jsnoise.herokuapp.com/api/showslist", String.class)
                .getBody());
    }
}
```

SCHEDULED JOBS: C# | JAVA

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddHostedService<SimpleJob>();
```

```
1 reference
public class SimpleJob : IHostedService, IDisposable
{
    3 references
    Timer _timer;

    0 references
    public Task StartAsync(CancellationToken cancellationToken)
    {
        _timer = new Timer(DoWork, null, TimeSpan.Zero,
            TimeSpan.FromSeconds(5));

        return Task.CompletedTask;
    }

    1 reference
    private void DoWork(object state)
    {
        Console.WriteLine("Job executed at: " + DateTime.Now);
    }

    0 references
    public Task StopAsync(CancellationToken cancellationToken)
    {
        _timer?.Change(Timeout.Infinite, 0);
        return Task.CompletedTask;

    }
```

```
@SpringBootApplication
@EnableScheduling
public class ApiApplication {

    Run | Debug
    public static void main(final String[] args) {
```

```
@Component
public class SimpleJob {
    //vod return type && no parameters
    //@Scheduled(cron = "0 15 10 15 * ?")
    @Scheduled( fixedRate = 1000, initialDelay = 100)
    public void Execute(){
        System.out.println("Job executed at: " + new Date());
    }
}
```

TESTING CONTROLLERS C#

Using xUnit and Moq (for mocking :D)

```
[Fact]
public void Todos_CanGetAllTodos()
{
    // Arrange
    var todoServiceMock = new Mock<TodosService>();
    var todo = new Todo{Id=1,Title = "Mocked todo"};

    todoServiceMock.Setup( expression: s=> s.GetAll())
        .Returns(new List<Todo>{todo})
        .Verifiable();
    var ctrl = new TodosController(todoServiceMock.Object);

    // Act
    var result =  ctrl.Get() as ActionResult<List<Todo>>;
    var data = result.Value as IEnumerable<Todo>;

    // Assert
    Assert.NotNull(data);
    Assert.True( condition: data.Any());
    Assert.Equal( expected: data.First().Title , actual: todo.Title);
    todoServiceMock.Verify();
}
```

TESTING CONTROLLERS JAVA

Using jUnit5 and Mockito (for mocking :D)

```java
@RunWith(SpringRunner.class)
@WebMvcTest(TodosController.class)
//@TestPropertySource(
//        locations = "classpath:application-integrationtest.properties")
public class TodosControllerTest {
    @Autowired private MockMvc mvc;
    @MockBean private TodosService service;

    @Test
   public void ca_call_get() throws Exception {
        Todo todo =  Todo.builder() .Id(1).title("first todo").build();
        List<Todo> todos = Arrays.asList(todo);
        given(service.getAll()).willReturn(todos);
        mvc.perform(get( urlTemplate: "/api/todos")
                .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andExpect(MockMvcResultMatchers.jsonPath( expression: "$", hasSize(1)))
                .andExpect(MockMvcResultMatchers.jsonPath( expression: "$[0].title", Is.is(todo.getTitle())));;
    }
}
```

MEMORY FOOTPRINT: C#

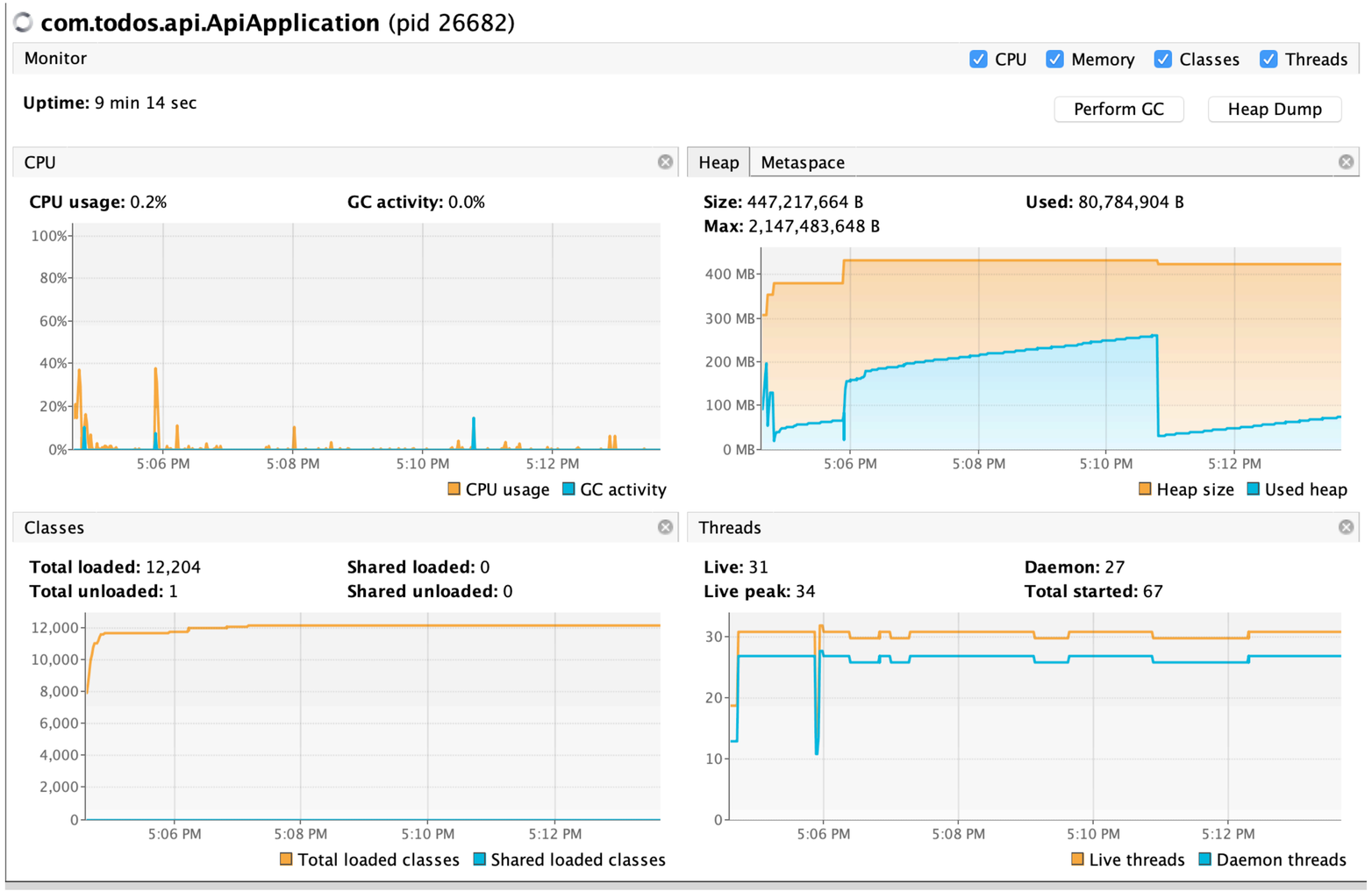
My process used working set 82.016 MB of working set

My process used working set 83.532 MB of working set

My process used working set 65.396 MB of working set

My process used working set 53.072 MB of working set

MEMORY FOOTPRINT: JAVA



SWAGGER C# WEBAPI

dotnet add package Swashbuckle.AspNetCore --version 5.0.0-rc4

//Inside the startup class add:

using Microsoft.OpenApi.Models;

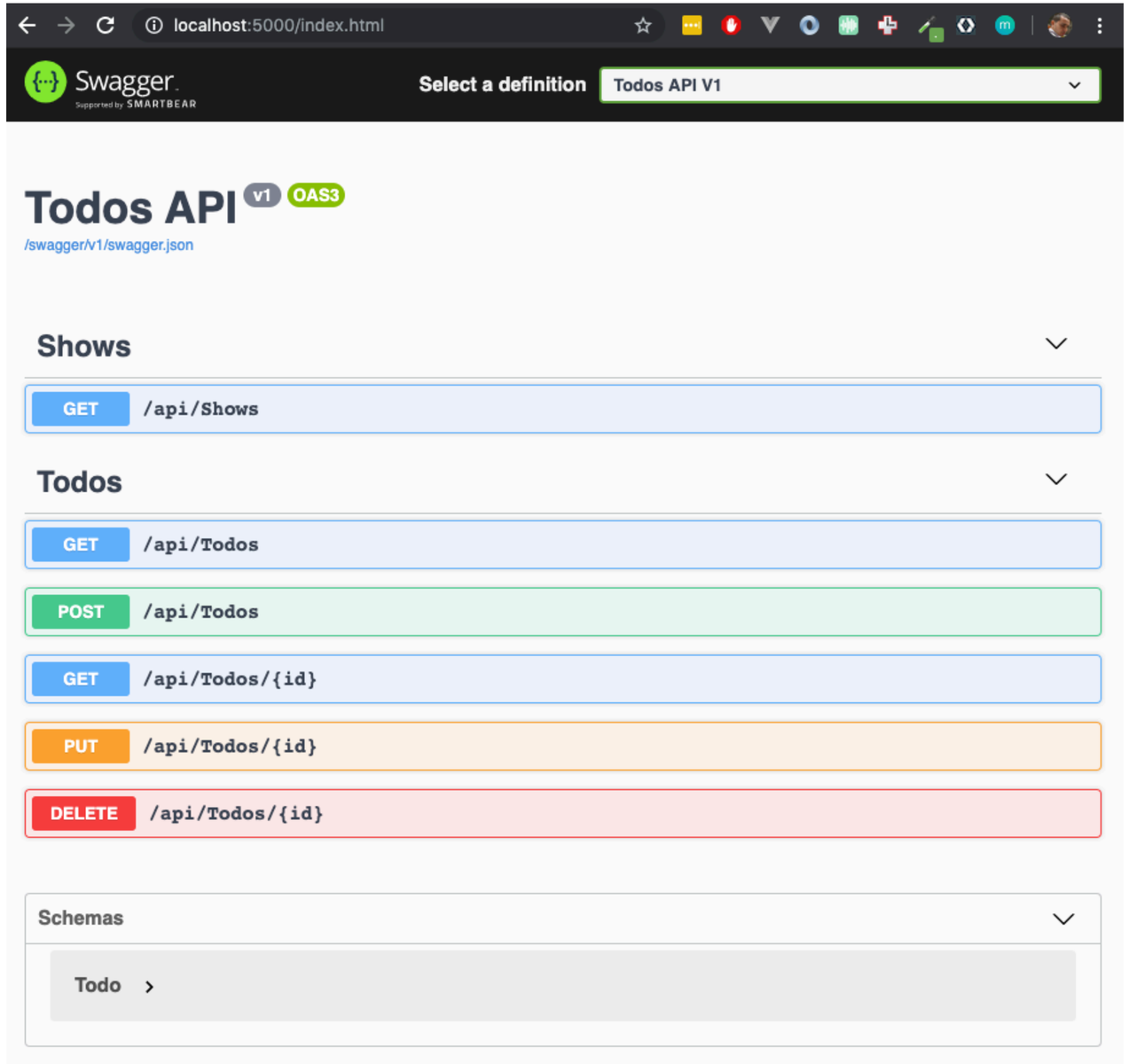
Inside the ConfigureServices method:

```
  // Register the Swagger generator, defining 1 or more Swagger documents
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "Todos API", Version = "v1" });
    });
```

Inside the Configure method:

```
/ Enable middleware to serve generated Swagger as a JSON endpoint.
    app.UseSwagger();

    // Enable middleware to serve swagger-ui (HTML, JS, CSS, etc.),
    // specifying the Swagger JSON endpoint.
    app.UseSwaggerUI(c =>
    {
        c.SwaggerEndpoint("/swagger/v1/swagger.json", "Todos API V1”);
        //next line to use the swagger as root page
        c.RoutePrefix = string.Empty;

    });
```

SWAGGER C# WEBAPI

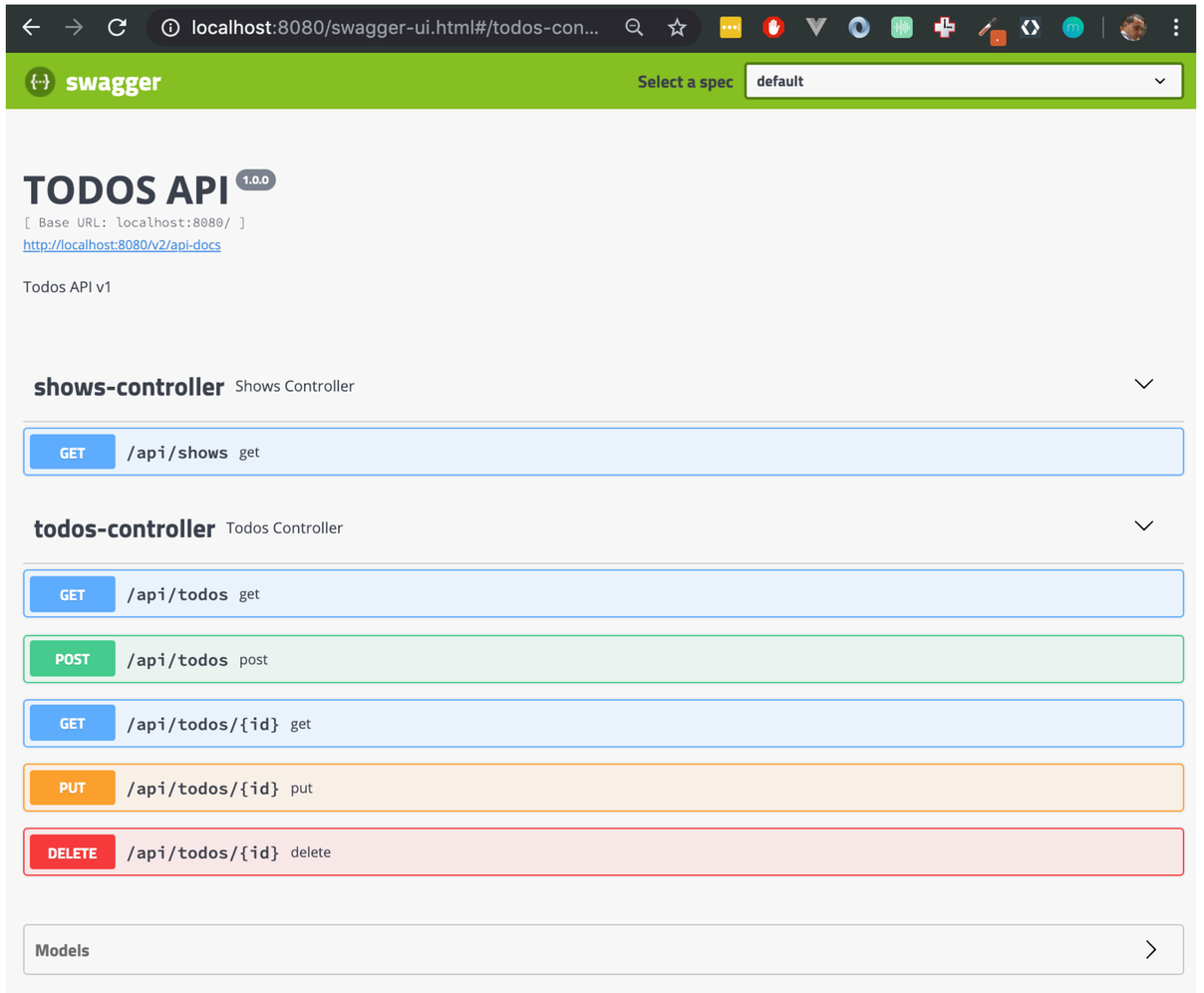


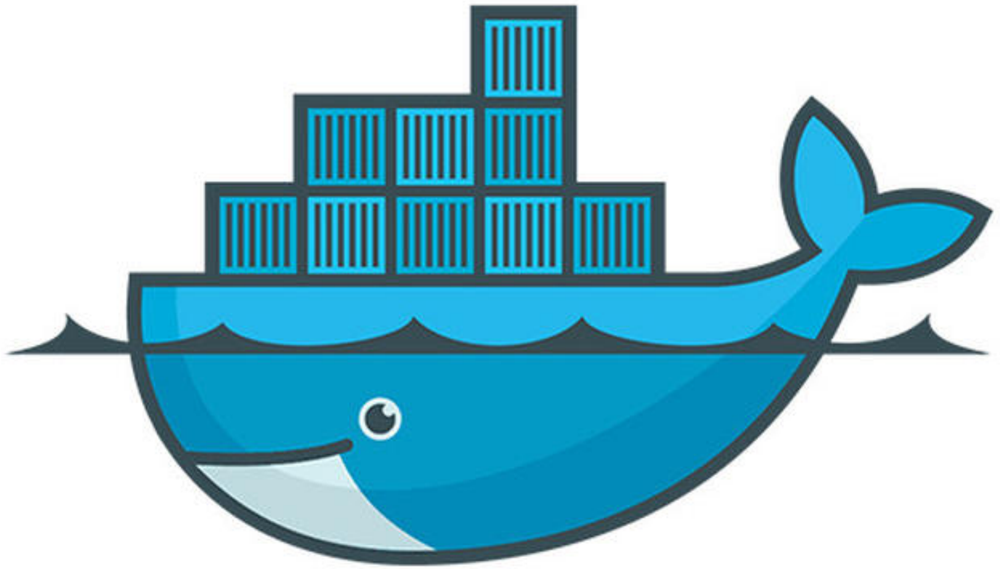
SWAGGER JAVA SPRING BOOT

Inside pom.xml add the Springfox dependencies

```xml
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>2.8.0</version>
    </dependency>

    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
        <version>2.8.0</version>
    </dependency>
</dependencies>
```

SWAGGER JAVA SPRING BOOT





docker

DOCKERFILE: C# | JAVA

docker build -t webapi .

docker run -p 5000:80 webapi

```
FROM mcr.microsoft.com/dotnet/core/sdk:2.2 AS build-env
WORKDIR /app

# Copy csproj and restore as distinct layers
COPY *.csproj ./
RUN dotnet restore

# Copy everything else and build
COPY . ./
RUN dotnet publish -c Release -o out

# Build runtime image
FROM mcr.microsoft.com/dotnet/core/aspnet:2.2
WORKDIR /app
COPY --from=build-env /app/out .
ENTRYPOINT ["dotnet", "api.dll"]
```

docker build -t spring .

docker run -p 8080:8080 spring

```
FROM maven:3.5.2-jdk-8-alpine AS MAVEN_BUILD

COPY pom.xml /build/
COPY src /build/src/
WORKDIR /build/
RUN mvn package -P prod

FROM openjdk:8-jre-alpine
WORKDIR /app
COPY --from=MAVEN_BUILD /build/target/api-0.0.1-SNAPSHOT.jar /app/
ENTRYPOINT ["java","-Dspring.profiles.active=prod ", "-jar", "api-0.0.1-SNAPSHOT.jar"]
```

DOCKER-COMPOSE: C# | JAVA

docker-compose up

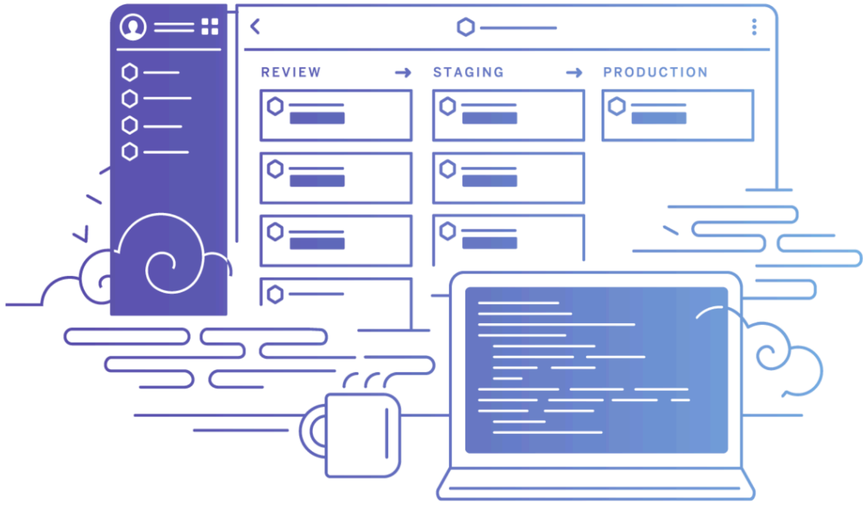
```yaml
version: "3"
services:
    db:
        image: "postgres"
        # ports:
        #       - "5432:5432"
        environment:
          POSTGRES_USER: postgres
          POSTGRES_PASSWORD: postgres
        volumes:
          - ./postgres-data:/var/lib/postgresql/data
        networks:
          - cs-network

    web:
        build: .
        ports:
            - "5000:80"
        depends_on:
            - db
        networks:
          - cs-network
        environment:
          DATABASE_URL: Host=db;Port=5432;Username=postgres;Password=password;Database=todos_cs;
networks:
  cs-network:
    driver: bridge
```

docker-compose up

```yaml
version: "3"
services:
    db:
        image: "postgres"
        environment:
          POSTGRES_DB: postgres
          POSTGRES_USER: postgres
          POSTGRES_PASSWORD: postgres

        volumes:
          - ./postgres-data:/var/lib/postgresql/data
          #- ./init.sql:/docker-entrypoint-initdb.d/10-db.sql

        networks:
          - java-network

    web:
        #build: .
        image: spring
        ports:
            - "8080:8080"
        depends_on:
            - db
        environment:
          - SPRING_DATASOURCE_URL= jdbc:postgresql://db:5432/postgres
          - SPRING_DATASOURCE_USERNAME=postgres
          - SPRING_DATASOURCE_PASSWORD=postgres
        networks:
          - java-network

networks:
  java-network:
    driver: bridge
```

HEROKU

HEROKU Products Marketplace Pricing Documentation Support More



DEVELOPERS

Focus on your apps

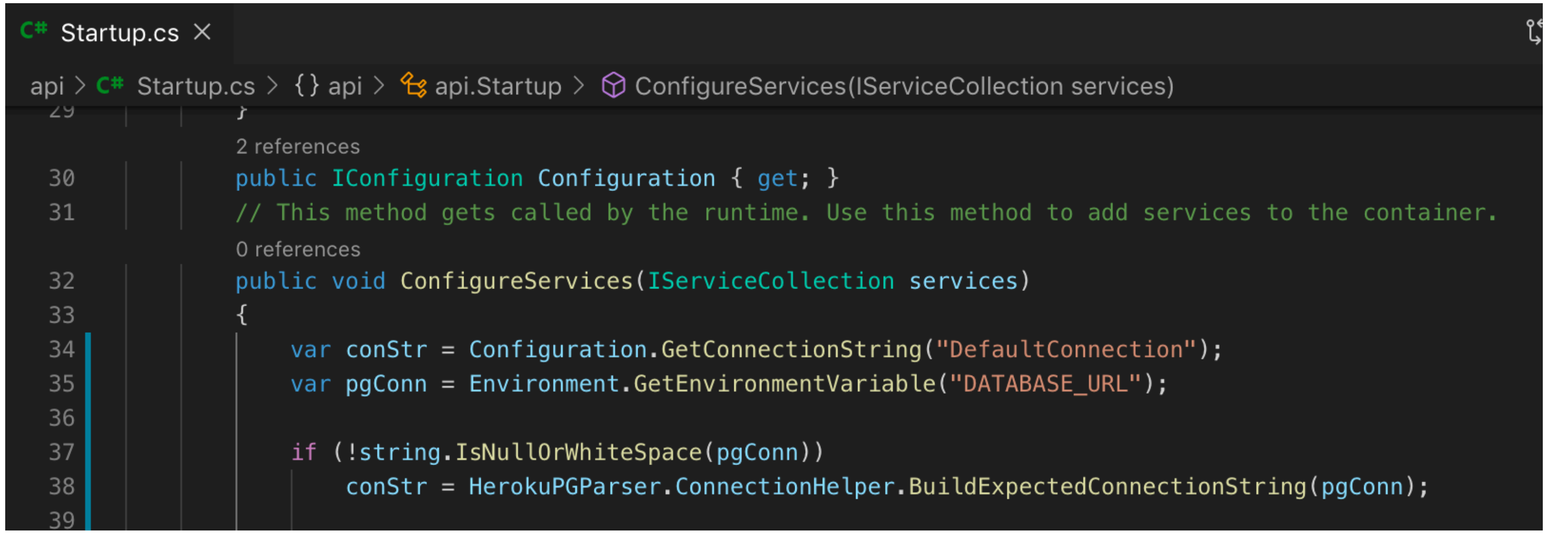
Invest in apps, not ops. Heroku handles the hard stuff — patching and upgrading, 24/7 ops and security, build systems, failovers, and more — so your developers can stay focused on building great apps.

SIGN UP FOR FREE

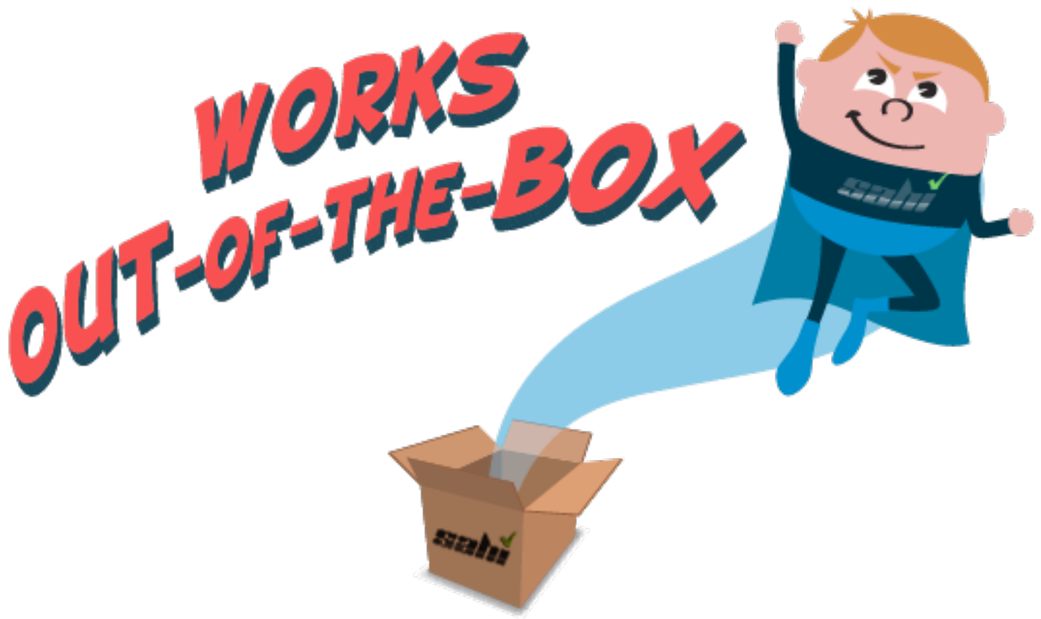
Explore the Heroku Platform

HEROKU WEBAPI

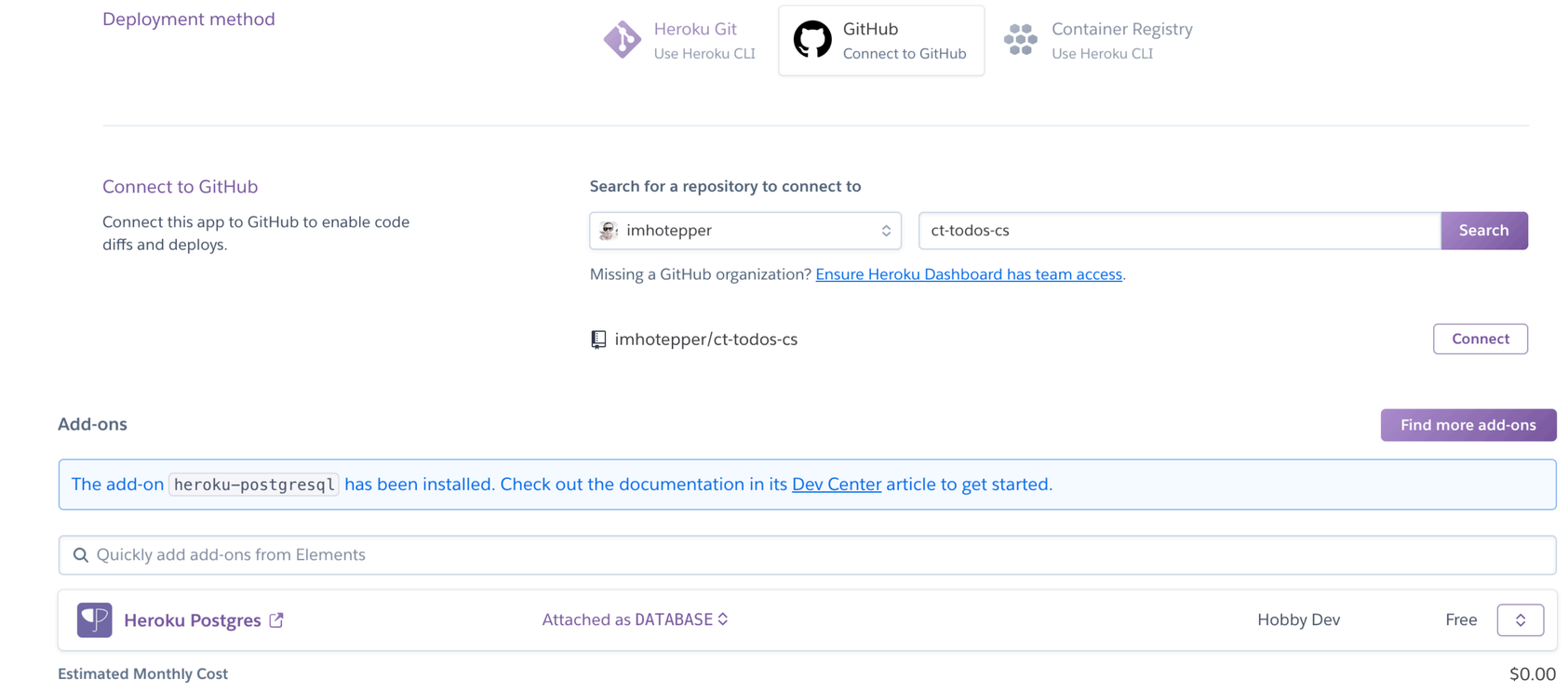
https://github.com/imhotepper/HerokuPgSqlConnectionParser



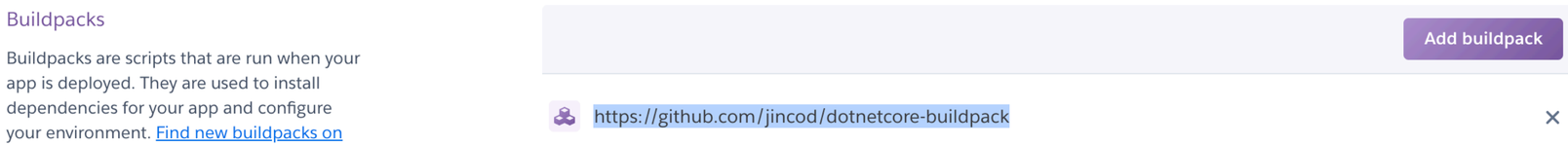
HEROKU SPRING-BOOT



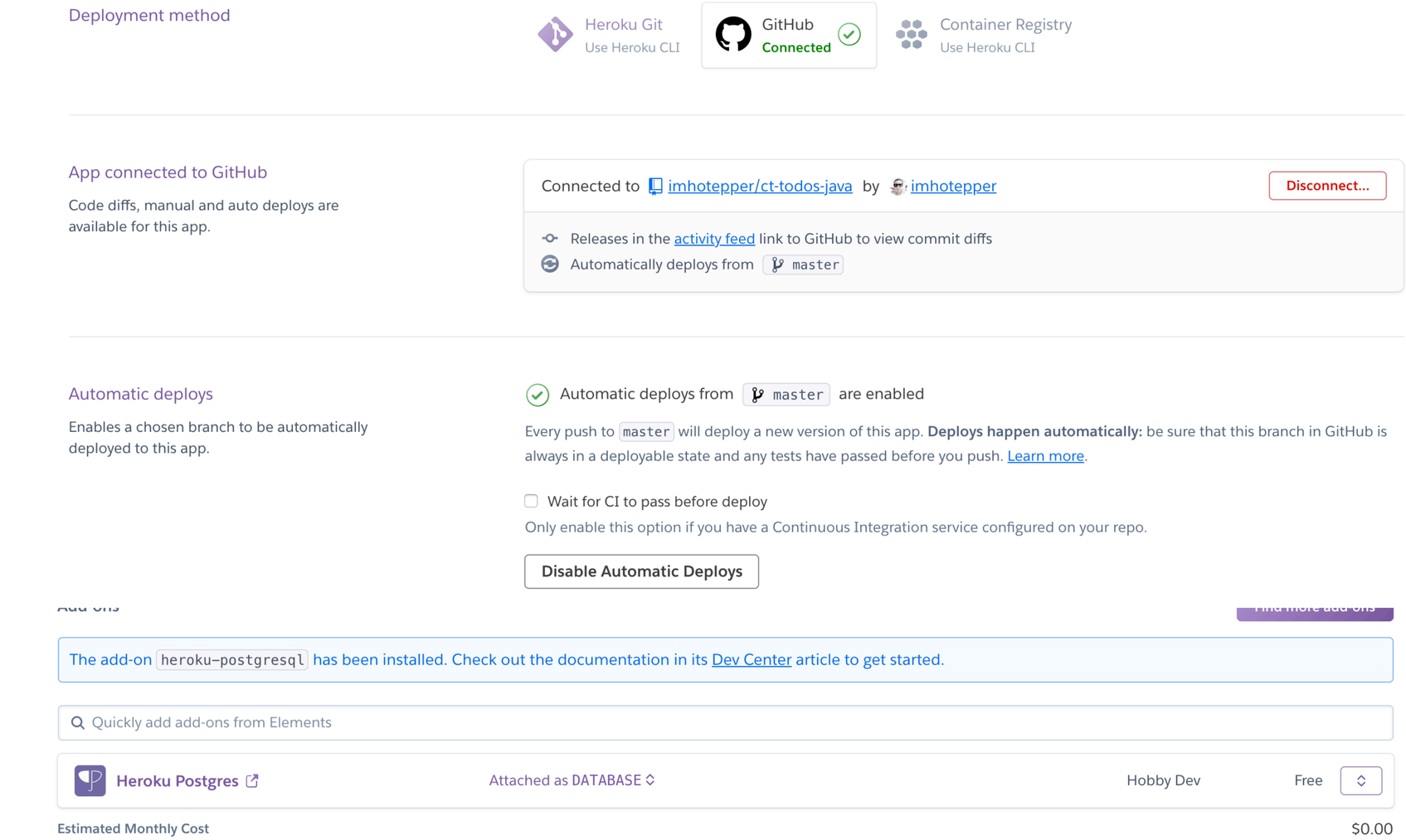
https://ct-todos-cs.herokuapp.com/



https://github.com/jincod/dotnetcore-buildpack



https://ct-todos-java.herokuapp.com/swagger-ui.html



SIZE & MEMORY FOOTPRINT

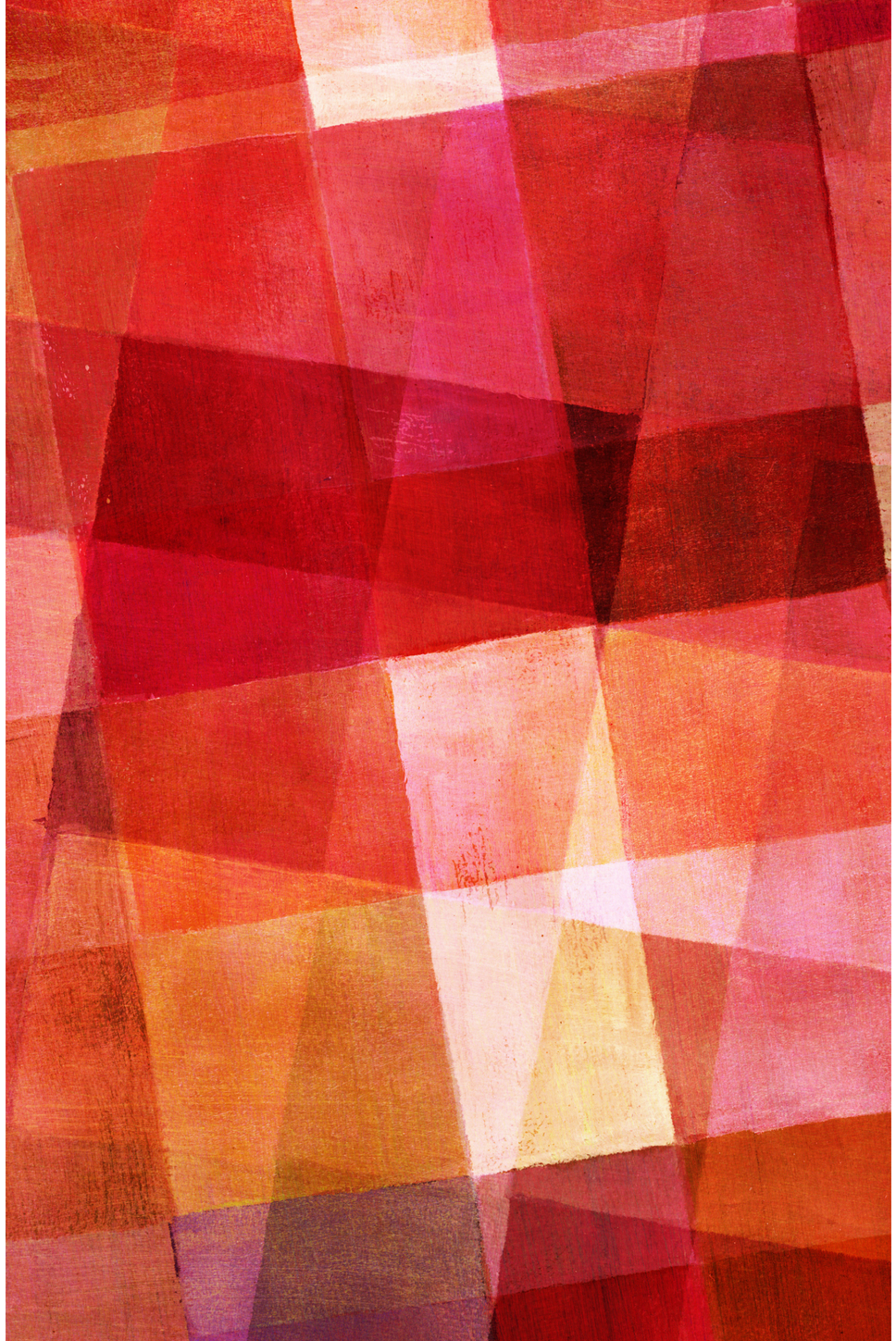
```
-----> Discovering process types
       Procfile declares types -> web
-----> Compressing...
       Done: 68.5M
-----> Launching...
       Released v7
       https://ct-todos-cs.herokuapp.com/ deployed to Heroku
```

```
-----> Discovering process types
       Procfile declares types     -> (none)
       Default types for buildpack -> web
-----> Compressing...
       Done: 90M
-----> Launching...
       Released v5
       https://ct-todos-java.herokuapp.com/ deployed to Heroku
```

```
heroku labs:enable log-runtime-metrics -a [appname]
```

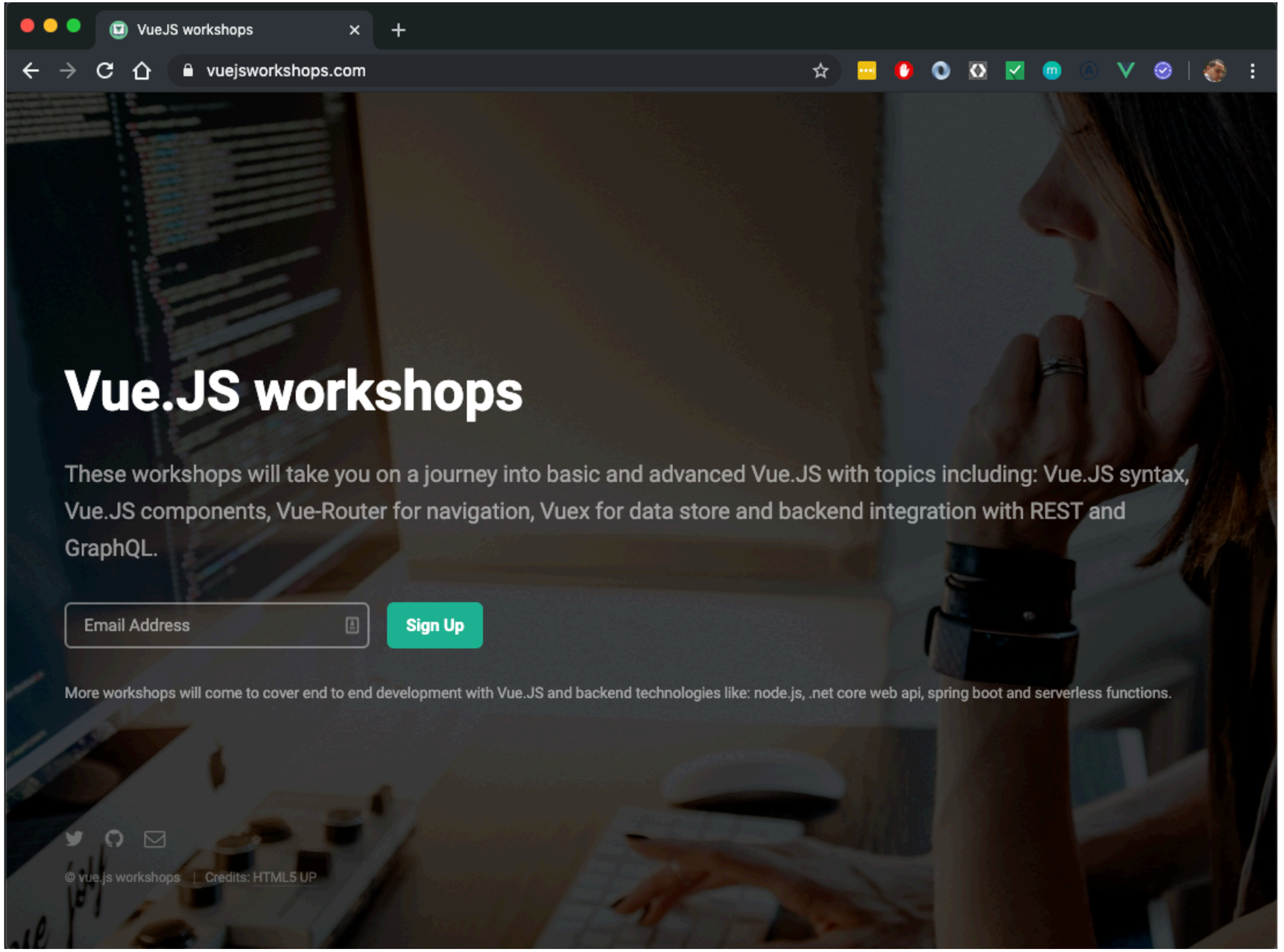
```
2019-11-25T15:09:25.858722+00:00 heroku[web.1]: source=web.1 dyno=heroku.153885564.daf60bcd-41d2-4e4c-830c-efea0060e81e sample#load_avg_1m=0.01
sample#load_avg_5m=0.03
2019-11-25T15:09:25.858722+00:00 heroku[web.1]: source=web.1 dyno=heroku.153885564.daf60bcd-41d2-4e4c-830c-efea0060e81e sample#memory_total=56.16MB
sample#memory_rss=50.55MB sample#memory_cache=5.61MB sample#memory_swap=0.00MB sample#memory_pgpgin=173325pages sample#memory_pgpgout=158948pages
sample#memory_quota=512.00MB
```

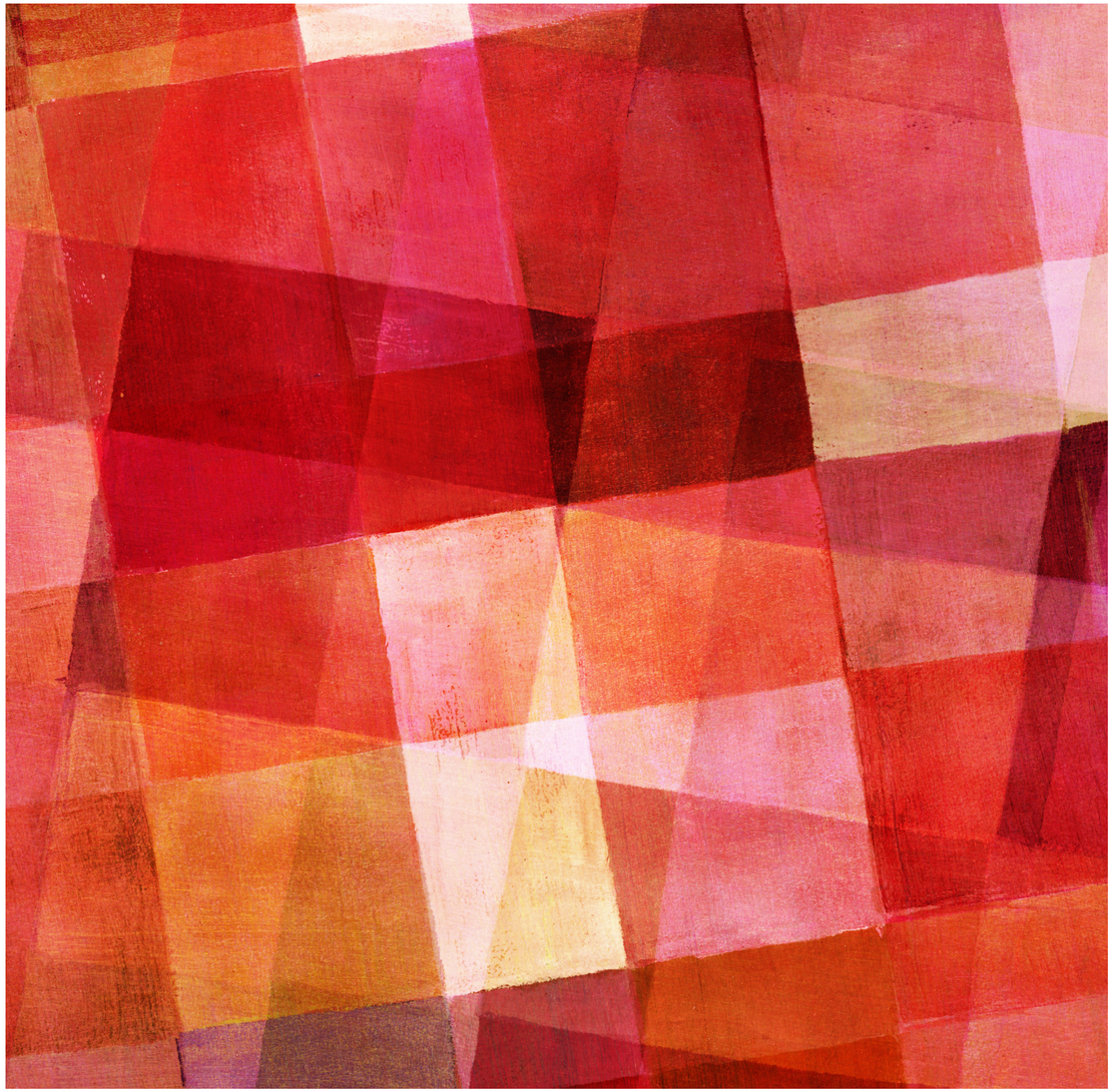
```
2019-11-25T15:07:45.303183+00:00 heroku[web.1]: source=web.1 dyno=heroku.153886854.22ddfbf8-4f5f-4596-b672-b185e3e411b1 sample#load_avg_1m=0.00
2019-11-25T15:07:45.303284+00:00 heroku[web.1]: source=web.1 dyno=heroku.153886854.22ddfbf8-4f5f-4596-b672-b185e3e411b1 sample#memory_total=301.38MB
sample#memory_rss=301.35MB sample#memory_cache=0.03MB sample#memory_swap=0.00MB sample#memory_pgpgin=88632pages sample#memory_pgpgout=13522pages
sample#memory_quota=512.00MB
```

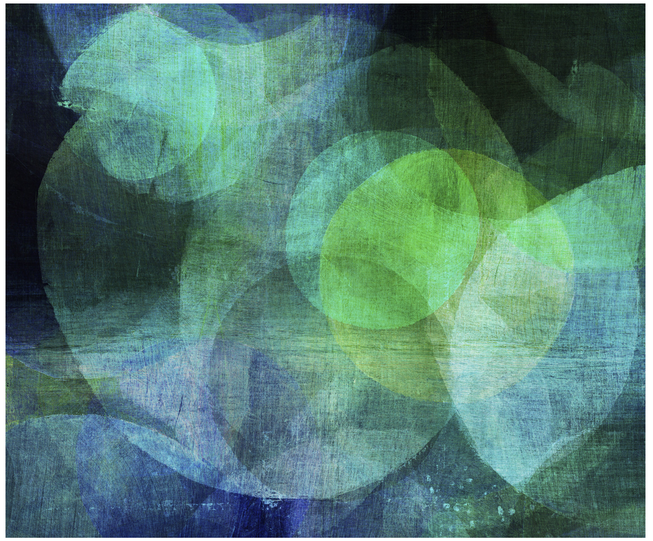


RECAP

- Prerequisites
- Dependencies management
- Creating a new api
- Create CRUD Controller
- Validation
- Error handling
- Database access
- Call another API
- Scheduled Jobs
- Testing controllers
- Memory footprint
- Swagger
- Docker
- Deploy to heroku


VueJS workshops
vuejsworkshops.com
Vue.JS workshops
These workshops will take you on a journey into basic and advanced Vue.JS with topics including: Vue.JS syntax, Vue.JS components, Vue-Router for navigation, Vuex for data store and backend integration with REST and GraphQL.
Email Address
Sign Up
More workshops will come to cover end to end development with Vue.JS and backend technologies like: node.js, .net core web api, spring boot and serverless functions.
© vue.js workshops | Credits: HTML5 UP





Thank you for your time & attention!