2h 55min

Submit Task

1
2
3

Task 2

Your task is to create a simple RESTful API that will allow a collection of articles to be managed. It should provide four operations, described in detail below. Your API should accept requests and return responses in JSON format. Additionally, it should be available under the address: */api/articles*.

Your goal is to implement just a controller using an injected repository:

```
[ApiController]
public class ArticlesController: ControllerBase
{
    private IRepository _repository;

    public ArticlesController(IRepository repository)
    {
        _repository = repository;
    }
}
```

The repository provides the following methods:

```
public interface IRepository
{
    // Returns a found article or null.
    Article Get(Guid id);
    // Returns the identifier of a created article.
    // Throws an exception if an article is null.
    // Throws an exception if a title is null or empty.
    Guid Create(Article article);
    // Returns true if an article was deleted or false if it was not possible to find it.
    bool Delete(Guid id);
    // Returns true if an article was updated or false if it was not possible to find it.
    // Throws an exception if an articleToUpdate is null.
    // Throws an exception if a title is null or empty.
    bool Update(Article articleToUpdate);
}
```

The `Article` class looks as follows (you do not need to create or change it):

```
public class Article
{
    public Guid Id { get; set; }
    public string Title { get; set; }
    public string Text { get; set; }
}
```

Operation 1 – Create a new article

- **HTTP Verb**: POST
- **Response Code On Success**: 201 (CREATED)
- **Response Code On Failure**: 400 (BAD REQUEST) if a title was not provided.
- **Operation URI**: */api/articles*
- On success, the API should return a *Location* header pointing to a created resource and additionally a created entity in the body. The entity should have the id updated to the one returned by the `Create` method.

Sample request:

```
POST /api/articles HTTP/1.1
Content-Type: application/json
...

{
    title: 'Some article'
    text: 'Hello World!'
}
```

Sample response in the case of success:

```
HTTP/1.1 201 Created
Location: /api/articles/c99a1a99-7296-46ca-b267-503e33f05b7b
...
{
    id: 'c99a1a99-7296-46ca-b267-503e33f05b7b'
    title: 'Some article'
    text: 'Hello World!'
}
```

Operation 2 – Delete an article with a given identifier

- **HTTP Verb**: DELETE
- **Response Code On Success**: 200 (OK)
- **Response Code On Failure**: 404 (NOT FOUND) if the article to delete does not exist.
- **Operation URI**: */api/articles/<GUID>*
- If the article to delete is not found, the API should return the following error message in the response body: *Article not found.*

Operation 3 – Update an article by updating its identifier to the given one

- **HTTP Verb**: PUT
- **Response Code On Success**: 200 (OK)
- **Response Code On Failure**: 404 (NOT FOUND) if the article to update does not exist, 400 (BAD REQUEST) if a title was not provided.
- **Operation URI**: */api/articles/<GUID>*
- This operation MUST NOT create new articles.
- This operation MUST NOT return an updated article in the response body.
- If the article to update was not found, the API should return the following error message in the response body: *Article not found.*

Sample request:

```
PUT /api/articles/c99a1a99-7296-46ca-b267-503e33f05b7b HTTP/1.1
Content-Type: application/json
...

{
    title: 'New title'
    text: 'Hello World!!!!!!!!!!!!!!!!'
}
```

Operation 4 – Get an article with a given identifier

- **HTTP Verb**: GET
- **Request Body**: N/A
- **Response Code On Success**: 200 (OK)
- **Response Code On Failure**: 404 (NOT FOUND) if the article with the given ID was not found.
- **Operation URI**: */api/articles/<GUID>*
- If the article was not found, the API should return the following error message in the response body: *Article not found.*

Sample response on success:

```
HTTP/1.1 200 OK
...
{
    id: 'c99a1a99-7296-46ca-b267-503e33f05b7b'
    title: 'Some article'
    body: 'Hello World!'
}
```

Hints

- You can assume that the API is available to everybody (no authorisation and authentication required).
- Technology (framework) that should be used: .NET Core 3.1.
- If the tests return NotFound status, verify if the attributes take into account parameter binding.

Solution C# 6.0 with .NET 4.7 (Mono 6.12)

```
using System;
using Microsoft.AspNetCore.Mvc;

namespace simple_rest_api
{
    [ApiController]
    public class ArticlesController: ControllerBase
    {
        private IRepository _repository;

        public ArticlesController(IRepository repository)
        {
            // Console.WriteLine("Sample debug output");
            _repository = repository;
        }
    }
}

/*
public interface IRepository
```

To leave editor use Ctrl + M

Test Output

Run Code

All changes saved

Any problems with the editor? Switch to basic editor

Give Feedback