Artificial Intelligence

The Super Human Guide

The Developer's Edition: Explaining and Researching a Technical Topic!

You can use ChatGPT to search for technical topics or ask it to explain technical issues simply by providing specific queries or questions related to the topic. ChatGPT has been trained on a large text corpus and can generate informative and relevant responses. Still, it's essential to remember that its answers may only sometimes be entirely accurate or up-to-date on highly specialized or niche technical subjects. Therefore, verifying the information with additional sources is always a good idea to ensure its reliability and relevance.

In this article, we take you through our journey as we use ChatGPT to search for - and explain- topics. The report contains several use cases that will significantly benefit developers looking to fully unlock the potential offered by ChatGPT and other Generative AI tools.

Sample Prompt 1

Why is AsEnumerable() used in the below code?
var GrpResult = from P in Context.Products.Include(p => p.Category).AsEnumerable() group P by P.Category.CategoryName;

Generated Response:

The AsEnumerable() method is used in the above code to explicitly indicate that the remainder of the LINQ query should be executed on the client-side rather than being translated to SQL and executed on the server side.

In the given code, Context.Products.Include(p => p.Category) specifies an eager loading operation to retrieve the Product entities and their associated Category entities. However, the group by operation cannot be translated to SQL, so the AsEnumerable() method is used to bring the results of the Include operation into memory as an IEnumerable<Product> collection.

By invoking AsEnumerable(), the query is now executed in memory rather than being translated to SQL and executed on the server side. The group by the operation is then performed on the IEnumerable<Product> collection returned by AsEnumerable(), and the results are returned as a sequence of IGrouping<string, Product> objects, where each group represents a distinct CategoryName and contains a collection of Product entities belonging to that category.

Sample Prompt 2

Compare Django REST vs. FastAPI in Python.

Generated Response:

Django and FastAPI are both Python web frameworks used to build APIs, but they have some differences. Here are a few comparisons between the two frameworks:

- Performance: FastAPI is faster than Django regarding request handling, thanks to its asynchronous nature and the use of Pydantic for request and response validation.
- Ease of Use: Django has a steep learning curve, but it provides a lot of built-in features such as an ORM, user authentication, and admin panel. FastAPI, on the other hand, is easier to set up and has a simpler syntax, but it relies on external libraries for many features.
- Flexibility: FastAPI is more flexible in terms of customizing the request and response handling, but Django provides a lot of customization options for authentication and authorization.
- Community: Django has a large and active community, with many third-party packages available. FastAPI is a relatively new framework, but its popularity is growing rapidly.
- Scalability: Both Django and FastAPI are scalable, but FastAPI has an advantage in terms of performance, which makes it a good choice for high-traffic applications.

Ultimately, the choice between Django and FastAPI depends on the specific needs of the project. Django is a good choice for complex projects with many built-in features, while FastAPI is a good choice for simple, high-performance projects.

Sample Prompt 3

. Please explain repository patterns for a developer giving a simple example?

Generated Response:

Certainly! Let me give you a simple example of how the Repository pattern can be used in a software application: Suppose you are developing a web application that allows users to create, read, update, and delete (CRUD) products in a database. You could create a ProductsRepository class that acts as an intermediary between the application logic and the database.

Here's how it could work:

- The ProductsRepository class would contain methods for retrieving, inserting, updating, and deleting products from the database. For example, you could have methods like GetAllProducts, GetProductById, InsertProduct, UpdateProduct, and DeleteProduct.

- The application logic would call the appropriate methods in the ProductsRepository to retrieve or modify products. For example, when a user wants to view a list of all products, the application logic would call the GetAllProducts method in the ProductsRepository.

- The ProductsRepository would handle the details of how the data is stored and retrieved from the database. It would use database-specific code (such as SQL statements) to interact with the database.

Using the Repository pattern in this way provides several benefits. It separates the data access logic from the application logic, making the code easier to understand and maintain. It also makes it easier to switch between different data sources (such as different databases) in the future because all the data access logic is contained within the ProductsRepository class. Overall, the Repository pattern is useful for managing data access in a software application. By creating a separate repository class to handle data access, developers can make their code more modular, maintainable, and scalable.

Sample Prompt 4

. Can you give a simple code(example using C# for the previous explanation?

Generated Response:

Sure, here's an example code snippet in C# demonstrating the repository pattern using Entity Framework Core as the data source.

*The entirely generated code and full response can be seen in the upcoming pages.

```
csharp
// Define the repository interface
public interface IProductRepository
{
    IEnumerable<Product> GetAllProducts();
    Product GetProductById(int productId);
    void AddProduct(Product product);
    void UpdateProduct(Product product);
    void DeleteProduct(int productId);
}

// Implement the repository using Entity Framework Core
public class ProductRepository : IProductRepository
{
    private readonly MyDbContext _context;

    public ProductRepository(MyDbContext context)
    {
        _context = context;
    }

    public IEnumerable<Product> GetAllProducts()
    {
        return _context.Products.ToList();
    }

    public Product GetProductById(int productId)
    {
        return _context.Products.FirstOrDefault(p => p.Id == productId);
    }

    public void AddProduct(Product product)
    {
        _context.Products.Add(product);
        _context.SaveChanges();
    }

    public void UpdateProduct(Product product)
    {
        _context.Products.Update(product);
        _context.SaveChanges();
    }
}
}
 }
}

}
```

```
// Use the repository in your application code
var optionsBuilder = new DbContextOptionsBuilder<MyDbContext>();
optionsBuilder.UseSqlServer("your connection string here");

using (var context = new MyDbContext(optionsBuilder.Options))
{
    var repository = new ProductRepository(context);

    // Get all products
    var products = repository.GetAllProducts();

    // Get a product by ID
    var product = repository.GetProductById(1);

    // Add a new product
    repository.AddProduct(new Product { Name = "New Product", Price = 9.99 });

    // Update an existing product
    var existingProduct = repository.GetProductById(1);
    existingProduct.Price = 14.99;
    repository.UpdateProduct(existingProduct);

    // Delete a product
    repository.DeleteProduct(2);
}
```

In this example, we create an interface IProductRepository that defines the methods for interacting with the product data and then create a concrete implementation ProductRepository that uses Entity Framework Core to perform the actual database operations. In the application code, we create a new instance of MyDbContext and pass it to the ProductRepository constructor, and then use the repository methods to perform CRUD (create, read, update, delete) operations on the product data.

ITWORX®

We make the world a better place for everyone!

https://www.itworx.com/

Contact Us!