Multi Choice Type Question You are working on an existing ASP.NET Core project where the database schema is already defined. Which approach would you use to integrate EF Core with the existing database schema? Marks : 1 Negative Marks : 0 image Answer here Code-first approach Model-first approach DB-first approach Migrations-first approach

**ChatGPT said:**

The correct answer is:

**DB-first approach** ✅

**Explanation:**

* In a **DB-first approach**, the database already exists, and you generate your EF Core models and DbContext from the existing database schema.
* **Code-first** is used when you define the models first and generate the database from them.
* **Model-first** is more common in older Entity Framework versions, not EF Core.
* **Migrations-first** isn’t a standard EF Core approach; migrations are used with code-first or DB-first to track schema changes.

**You said:**

Multi Choice Type Question You have an existing ASP.NET Core project that uses EF Core with the code-first approach. You want to switch to the DB-First approach and generate the EF Core model from the existing database schema. What is the recommended way to achieve this? Marks : 1 Negative Marks : 0 image Answer here Reverse engineer the database schema using the dotnet ef dbcontext scaffold command. Manually modify the existing code-first model classes to match the database schema. Create a new ASP.NET Core project and choose the DB-First template during project setup. Use the EF Core Migration feature to generate the EF Core model from the existing database schema.

**ChatGPT said:**

The correct answer is:

**Reverse engineer the database schema using the dotnet ef dbcontext scaffold command.** ✅

**Explanation:**

* When switching from **code-first** to **DB-first**, you need to generate EF Core models (DbContext and entity classes) from an existing database.
* The **dotnet ef dbcontext scaffold** command connects to the database and automatically generates the required classes.
* Manually modifying code-first models is error-prone and not recommended.
* There is no DB-First template during project creation in ASP.NET Core.
* EF Core migrations are used to update the database from models, not the other way around.

**You said:**

Multi Choice Type Question You are working on an ASP.NET Core project and need to use Entity Framework Core for database operations. Which of the following statements best describes Entity Framework (EF) Core? Marks : 1 Negative Marks : 0 image Answer here EF Core is an object-relational mapping (ORM) framework that enables developers to work with databases using strongly-typed .NET objects. EF Core is a database management system specifically designed for ASP.NET Core applications. EF Core is a lightweight version of Entity Framework that only supports code-first database development. EF Core is a standalone SQL query builder library for executing raw SQL queries in ASP.NET Core.

**ChatGPT said:**

The correct answer is:

**EF Core is an object-relational mapping (ORM) framework that enables developers to work with databases using strongly-typed .NET objects.** ✅

**Explanation:**

* **EF Core** is an ORM that allows developers to interact with databases using .NET objects instead of writing raw SQL.
* It supports both **code-first** and **DB-first** approaches.
* EF Core is **not a database management system** itself.
* It is **not limited to code-first**; it also supports database-first.
* While EF Core allows executing raw SQL, it is much more than just a SQL query builder.

**You said:**

Multi Choice Type Question You are using the DB-First approach in EF Core and want to specify the output file name for the generated DbContext class. Which option would you use when running the dotnet ef dbcontext scaffold command? Marks : 1 Negative Marks : 0 image Answer here --context --output-file --context-file --generate-file

**ChatGPT said:**

The correct answer is:

**--context** ✅

**Explanation:**

* In **DB-First** EF Core, when using the dotnet ef dbcontext scaffold command, the **--context** option allows you to specify the name of the generated DbContext class.
* --output-dir is used to specify the folder for the generated entity classes.
* There is **no --output-file or --context-file option** in EF Core scaffolding.
* --generate-file is not a valid EF Core CLI option.

**You said:**

Multi Choice Type Question You are developing an ASP.NET Core application that requires access to an existing database. You decide to use EF Core with the DB-First approach. Which of the following options best describes the steps you need to follow? Marks : 1 Negative Marks : 0 image Answer here Design your entity classes, generate migrations, and then scaffold the database from the migrations. Generate entity classes and migrations from an existing database, and then scaffold the database from the migrations. Create a new database using EF Core code-first approach and then generate entity classes from the database. Write SQL scripts to create the database schema, and then generate entity classes and migrations from the schema.

**ChatGPT said:**

The correct answer is:

**Write SQL scripts to create the database schema, and then generate entity classes and migrations from the schema.** ✅

**Explanation:**

* In **DB-First** EF Core, the database already exists (or is created via SQL scripts), and you generate the **DbContext** and **entity classes** from that existing schema using scaffolding (dotnet ef dbcontext scaffold).
* Designing entity classes and generating migrations is part of the **code-first** approach, not DB-First.
* You do **not** scaffold from migrations; migrations are used to update databases from models, not the other way around.

**You said:**

Multi Choice Type Question You are working on an ASP.NET Core project and need to integrate a database into your application. You want to use the Entity Framework Core (EF Core) with the DB-First approach. Which of the following steps will be part of this process? Marks : 1 Negative Marks : 0 image Answer here Creating entity classes and migrations from an existing database. Defining the database schema using EF Core annotations. Generating a database from your application's entity classes. Executing SQL scripts to create the database schema.

**ChatGPT said:**

The correct answer is:

**Executing SQL scripts to create the database schema.** ✅

**Explanation:**

* In the **DB-First approach**, the database already exists or is created using **SQL scripts**, and EF Core is used to generate **DbContext** and entity classes from this existing schema.
* Creating entity classes and migrations is part of **code-first**, not DB-First.
* Defining the schema using EF Core annotations and generating a database from entity classes are also **code-first** steps.

**You said:**

Multi Choice Type Question You have been given a SQL Server database and you need to generate the EF Core model and entity classes from the existing database schema. Which command would you use to scaffold the EF Core model using the DB-First approach? Marks : 1 Negative Marks : 0 image Answer here dotnet ef database update dotnet ef migrations add Initial dotnet ef dbcontext scaffold dotnet ef database drop

**ChatGPT said:**

The correct answer is:

**dotnet ef dbcontext scaffold** ✅

**Explanation:**

* In **DB-First** EF Core, the dotnet ef dbcontext scaffold command is used to generate the **DbContext** and **entity classes** from an existing database schema.
* dotnet ef database update applies migrations to the database.
* dotnet ef migrations add creates a new migration based on model changes (code-first).
* dotnet ef database drop deletes the database and is unrelated to scaffolding.

**You said:**

Multi Choice Type Question When using Entity Framework Core with the DB-First approach, what option would you utilize to specify the namespace for the generated entity classes while scaffolding the DbContext? Marks : 1 Negative Marks : 0 image Answer here --context-namespace --entity-namespace --namespace --output-namespace

**ChatGPT said:**

The correct answer is:

**--namespace** ✅

**Explanation:**

* When using **DB-First** scaffolding with EF Core (dotnet ef dbcontext scaffold), the **--namespace** option allows you to specify the **namespace** for the generated entity classes.
* --context-namespace is used to set the namespace specifically for the **DbContext** class.
* There are no valid EF Core options called --entity-namespace or --output-namespace.

**You said:**

Multi Choice Type Question When using the dotnet ef dbcontext scaffold command with Entity Framework Core in the Database-First approach, what does the --force option do? Marks : 1 Negative Marks : 0 image Answer here Forces the scaffolding process to overwrite existing files without prompting. Forces the database to update its schema based on the current DbContext configuration. Forces the scaffolding process to include all tables from the database. Forces the DbContext to ignore any existing migrations in the project.

**ChatGPT said:**

The correct answer is:

**Forces the scaffolding process to overwrite existing files without prompting.** ✅

**Explanation:**

* In **DB-First** EF Core, when you run dotnet ef dbcontext scaffold, the **--force** option allows the command to **overwrite existing generated files** (DbContext and entity classes) without asking for confirmation.
* It does **not** update the database schema, include all tables, or ignore migrations.

**You said:**

Multi Choice Type Question In an ASP.NET Core application using Entity Framework Core, what command would you use to add a migration for your DbContext? Marks : 1 Negative Marks : 0 image Answer here dotnet ef update database dotnet ef model add dotnet ef dbcontext add dotnet ef migration add

**ChatGPT said:**

The correct answer is:

**dotnet ef migrations add** ✅

**Explanation:**

* In **EF Core**, the dotnet ef migrations add <MigrationName> command is used to create a **new migration** based on changes in your DbContext and entity classes.
* dotnet ef database update applies migrations to the database.
* dotnet ef model add and dotnet ef dbcontext add are **not valid EF Core CLI commands**.

**You said:**

Multi Choice Type Question You are using the DB-First approach with EF Core and want to include additional database tables in your model that were not included in the initial scaffold. Which command should you use to add these tables to your existing model? Marks : 1 Negative Marks : 0 image Answer here Re-run the dotnet ef dbcontext scaffold command with the new tables included. Manually create new entity classes for the additional tables. Use the dotnet ef migrations add command to include the new tables. Update the existing DbContext class to include the new tables.

**ChatGPT said:**

The correct answer is:

**Re-run the dotnet ef dbcontext scaffold command with the new tables included.** ✅

**Explanation:**

* In **DB-First** EF Core, to include additional tables that were not part of the initial scaffold, you **re-run the dotnet ef dbcontext scaffold command**, specifying the new tables.
* You can use the --force option to overwrite existing files if needed.
* Manually creating entity classes or updating DbContext is **possible but not recommended**, as it can lead to inconsistencies with the database schema.
* dotnet ef migrations add is used in **code-first**, not DB-First.

**You said:**

Multi Choice Type Question When using Entity Framework Core with the DB-First approach, what is the purpose of the OnModelCreating method in the DbContext class? Marks : 1 Negative Marks : 0 image Answer here To seed initial data into the database. To configure the model and relationships between entities. To create migrations for the database schema. To update the database schema based on model changes.

**ChatGPT said:**

The correct answer is:

**To configure the model and relationships between entities.** ✅

**Explanation:**

* In **EF Core**, the OnModelCreating method in the DbContext class is used to **configure entity properties, relationships, table mappings, keys, constraints, and other aspects of the model**.
* While you *can* seed data in OnModelCreating, its primary purpose is **model configuration**.
* Creating migrations or updating the database schema is done using **EF Core CLI commands**, not OnModelCreating.

**You said:**

Multi Choice Type Question In the context of EF Core's DB-First approach, what is the role of the OnModelCreating method within the DbContext class? Marks : 1 Negative Marks : 0 image Answer here It is used to configure the database connection string and other settings. It allows defining the model's schema using Fluent API, including relationships, constraints, and configurations. It generates migration scripts to update the database schema based on the entity classes. It is responsible for executing raw SQL queries directly against the database.

**ChatGPT said:**

The correct answer is:

**It allows defining the model's schema using Fluent API, including relationships, constraints, and configurations.** ✅

**Explanation:**

* In **EF Core**, the OnModelCreating method in the DbContext class is used to **configure the model** using the **Fluent API**. This includes setting up **relationships, constraints, table names, keys, and other configurations**.
* Configuring the connection string is done in the DbContext constructor or via dependency injection.
* Generating migrations and executing raw SQL are handled outside OnModelCreating.

**You said:**

Multi Choice Type Question When using the DB-First approach with EF Core, how does the DbContext class facilitate interaction with the database schema? Marks : 1 Negative Marks : 0 image Answer here It automatically updates the database schema based on the entity classes. It provides a set of DbSet properties that map to the database tables and allows querying and saving data. It generates SQL scripts to create the database schema from the entity classes. It requires manual synchronization of the entity classes with the database schema.

**ChatGPT said:**

The correct answer is:

**It provides a set of DbSet properties that map to the database tables and allows querying and saving data.** ✅

**Explanation:**

* In **DB-First EF Core**, the DbContext class acts as a **bridge between your application and the database**.
* The **DbSet properties** represent tables in the database, enabling you to **query, insert, update, and delete data** using strongly-typed entities.
* The DbContext does **not automatically update the database schema**—that is handled by migrations in code-first, not DB-first.
* It also does **not generate SQL scripts** from entity classes in DB-First.

**You said:**

Multi Choice Type Question In an ASP.NET Core project using the DB-First approach, you want to exclude certain tables from being scaffolded into your EF Core model. Which option should you use with the dotnet ef dbcontext scaffold command? Marks : 1 Negative Marks : 0 image Answer here --exclude-tables --skip-tables --tables-exclude --exclude

**ChatGPT said:**

The correct answer is:

**--exclude-tables** ✅

**Explanation:**

* When using **DB-First scaffolding** with EF Core, the --exclude-tables option allows you to **exclude specific tables** from being generated in the DbContext and entity classes.
* Other options like --skip-tables, --tables-exclude, or --exclude are **not valid EF Core CLI options**.