**Course Name: .Net 6 Core Entity Framework: How-To Guide For Professionals**

*NOTE: Mark the correct answers with Yellow highlight*

**Chapter Number 1 - Migrations**

1. Migrations are used for…

a. Keeping code in sync with database changes.

Reason – incorrect, it does not move database schema changes into the models.

b. Pushing Business Logic into the database Server.

Reason – Incorrect, business logic typically is within middle-tier code and does not reside in the database unless it is contained within a stored procedure. Stored procedures should be used for data (set) based logic only.

c. Pushing Classes (Models) changes into the database.

Reason – Correct, Migrations are used to moved identified classes (models) into the database schema as tables, views, or map results for stored procedure invocations.

d. Pulling database schema definition into class models.

Reason – Incorrect, Migrations are used to move models into the database; another technique is used to move database schema elements into models (re: Scaffolding).

2. What are the types of DbContext and how are they used?

a. Execute and Compile, to run and build Migration code.

Reason – Incorrect, these don’t exist as concepts in Migrations.

b. Generate and Run, to create the database entities and then map them to classes.

Reason – Incorrect, while conceptually correct (Generate is close to Design-Time), however that also incudes mappings.

c. Design-Time and Run-Time; used to push class definitions to the database and then access their stored content.

Reason – Correct, Design-Time DbContext is used to perform migration of class definitions into database schema entities while Run-Time is used during application execution to return content (instance data) for class definitions.

d. Design-Time and Deploy-Time.

Reason – Incorrect, Design-Time is a correct type of DbContext used by migrations to define the changes needed to the database schema, the Design-Time context is also used (via Update-database migration command) to deploy those changes into the database.

3. How do you specify the project folder to place generated migrations into?

a. Add-Migration MyFolder/MigrationName.

Reason – Incorrect, folder specification is not included on the path for the migration name.

b. Create-Migration MigrationName -OutputDir MyFolder/Subfolder.

Reason – Incorrect, trick question – there is no Create-Migration command.

c. Add-Migration MigrationName -OutputDir MyFolder/Subfolder.

Reason – Correct, folder and subfolder are specified with the -OutputDir parameter. Without this parameter, migrations are placed by default into the “Migrations” folder (created when fist migration is generated).

d. Add-Migration MigrationName FolderName/SubFolder.

Reason – Incorrect, the -**OutputDir** parameter must be used to identify the folder and subfolder for generated migrations (folder specification follows the parameter).

4. A Visual Studio project specific for Migrations is…

a. required in order to use migrations to manage schema generation.

Reason Incorrect – migrations can be included in any project or a project specific for migration only.

b. not required, as they are created outside the project folder structure and reference on the command line.

Reason – Incorrect; migrations are part of the project folder structure and must be located in one or more projects.

c. Is by default the startup project, but can be targeted to a specific Visual Studio project file using the -project command parameter; re: “add-migration MyNewMigration -project WebApplication.Migrations”

Reason – Correct, the -Project parameter is used to specify the Migrations Visual Studio project, “WebApplication.Migrations”

d. Is the startup Project only, and cannot be changed.

Reason – Incorrect; the project to contain migrations can be specified on the “Add-Migration” command using parameter -project.

5. A Code-First perspective is…

a. the only way to map classes to database tables.

Reason – Incorrect, classes can be mapped to views, stored procedures and custom SQL; a **Database-First** perspective is also possible wherein classes are mapped to preexisting database structures (this is also known as reverse engineering).

b. where classes are modeled in code and the needed storage structures (schema) are generated at design time.

Reason – Correct; code-first means that classes are create with persistent properties that are mapped to database structures defined through generated & custom migration logic through using the **MigrationBuilder** class.

c. is used to generate c# classes and their corresponding table entities based on design models.

Reason – Incorrect, in .net core 6 entity framework there is no design surface (as there was in prior versions).

d. how you create applications when you don’t have an object model or application design.

Reason – Incorrect, both an object model and application design can be used to support the code-first development paradigm.