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

*NOTE: Mark the correct answers with Yellow highlight*

**Chapter Number 3 – Fluent API**

1. Within the context of .net core entity framework, the Fluent API is used for…

a. Providing in-line documentation and help text.

Reason – incorrect, they are not used for either of those purposes.

b. describing data type of properties, refining relationships between entities, and providing customization that extend what possible with Data Annotations or by Convention.

Reason – Correct, they are used to refine the physical storage, relationships, and other attributes for related database tables and columns, that override and extend the Conventions and Data Annotations capabilities.

c. establishing relationships between classes.

Reason – Incorrect, relationships are just one aspect of class-models that can be described using the Fluent API.

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 three ways to affect the database entity definitions that are mapped to classes and their precedence order?

a. Data Annotations First, Fluent API second, and by Convention last (default).

Reason –Incorrect, Data Annotations do not override the Fluent API settings.

b. by Convention first, Data Annotations second, and the Fluent API last (default).

Reason – Incorrect; this is the reverse order.

c. Fluent API first, Convention second, and Data Annotations last (default).

Reason – Incorrect; By Convention is the default (last) to be applied, preceded by Data Annotations and the Fluent API (takes precedence over all others).

d. Fluent API first, Data Annotations second, and by Convention last (default).

Reason – Correct, the Fluent API is the most powerful and takes precedence over declarations done through Data Annotation, both of which override Convention.

3. Using the Fluent API, how do you specify an alternate key on a table mapped to class?

a. invoke ModelBuilder.Entity<entityName>.CreateAlternateKey(“ColumnName”) from DbContext.OnModelCreating.

Reason – Incorrect; there is no method called CreateAlternateKey (use HasAlternateKey).

b. invoke ModelBuilder.Entity<entityName>.HasAlternateKey(“ColumnName”) from DbContext.OnModelCreating.

Reason – Correct, In our DbContext derived classed, we override the OnModelCreating method and using the ModelBulder class call HasAlternateKey supplying the name of the column used for the alternate key.

c. invoke ModelBuilder.Entity<entityName>.HasAlternateKey(“ColumnName”) from DbContext. OnConfiguring.

Reason – Incorrect; OnConfiguring override is used to configure the database (and other options).

d. add [Key] to the property declaration.

Reason – incorrect; this is a Data-Annotation attribute that designates a property as a primary key.

4.How to you make a class property value optional (not required) in the mapped database table?

a. using the ModelBuilder, use the IsRequired(false) method for a specific class property.

Reason – Correct; IsRequred is used to specify that a column can be null (empty), which mean it is not required.

b. using the ModelBuider, use the Optional(true) method fore a specific class property.

Reason – Incorrect, there is no method called Optional.

c. override the OnConfiguring method of the derived DbContext and call the IsOptional(true) for the specified table and column.

Reason – Incorrect, you do not use OnConfiguring to access the Fluent API (ModelBuilder class).

d. it is not possible, all database column must have a value (special value DBNull is assigned to the mapped table column).

Reason. – Incorrect, you can have nullable columns of any datatype (in fact, it is the default).

5. What is the relationship type between classes that can be defined only through the Fluent API?

a. One-to-One.

Reason – Incorrect; this can be described in both Data Annotations and Fluent API.

b. One-To-Many.

Reason - Incorrect; this can be described in both Data Annotations and Fluent API.

c. Many-To-Many.

Reason – Correct; requires the creation of an intermediary class that contain a composite key composing the primary keys of the two involved tables as foreign keys, which resolve the many-to-many into two one-to-many relationships.

d. None – all types can be established using Data Annotations or through following Conventions.

Reason – Incorrect; establishing a Many-To-Many relationship requires the Fluent API and an additional intermediary class to resolve the relationship into two One-To-Many relationships.