Getting Started with Entity Framework 6 Code First using MVC 5

Chapter 01: Getting Started with Entity Framework 6 Code First using MVC 5

1. What is the navigation property in the entity data model? Why do you use the virtual keyword in navigation properties?
    - Navigation properties hold other entities that are related to this entity.
    - Navigation properties are defined as virtual so that they can take advantage of certain Entity Framework functionality such as lazy loading.
2. What is a data context class in the entity data model?
    - The main class that coordinates Entity Framework functionality for a given data model is the database context class
    - This class is responsible for interacting with the database.
    - It is responsible for the following activities:
        - Querying
        - Change Tracking
        - Persisting Data
        - Caching
        - Manage Relationship
3. What is an entity set (DbSet) in an entity class?
    - For each table in the database is defined as a property of DbSet type
    - An entity set corresponds to a database table, and an entity corresponds to a row in the table.
4. How do you initialize databases with test data with Entity Framework?
    - To initialize a database, you have to create an initializer class and override seed.
    - Wherever entity Framework creates the database. It will call automatically the seed method.
5. Write down the convention used by the Entity Framework during database creation?
    - The pluralized forms of entity class names are used as table names.
    - Entity property names are used for column names.
    - Entity properties that are named ID or classnameID are recognized as primary key properties.
    - A property is interpreted as a foreign key property if it's named <navigation property name><primary key property name> (for example, StudentID for the Student navigation property since the Student entity's primary key is ID).
    - Foreign key properties can also be named the same simply as <primary key property name> (for example, EnrollmentID since the Enrollment entity's primary key is EnrollmentID).

Chapter 02: Implement CRUD Functionality with the Entity Framework in ASP.NET MVC

1. How do you create an URL using helpers?
    - An URL is created using ActionLink helper
    - For example, @HTML.ActionLink("Create", "Create", "Employees")
2. How can you prevent cross-site request forgery?
    - The ValidateAntiForgeryToken attribute helps prevent cross-site request forgery attacks.
    - It requires a corresponding Html.AntiForgeryToken() statement in the view

3. How can you prevent over-posting?
    - The Bind attribute is one way to protect against over-posting in create scenarios.
    - It is applied to Action method parameter with include property
    - For example
    - public ActionResult Create ([Bind(Include="name, course")]Trainee Trainee){...}
4. What are the possible states of an entity object?
    - An entity may be in one of the following states:
        - Added. The entity does not yet exist in the database. The SaveChanges method must issue an INSERT statement.
        - Unchanged. Nothing needs to be done with this entity by the SaveChanges method. When you read an entity from the database, the entity starts out with this status.
        - Modified. Some or all of the entity's property values have been modified. The Savechanges method must issue an UPDATE statement.
        - Deleted. The entity has been marked for deletion. The SaveChanges method must issue a DELETE statement.
        - Detached. The entity isn't being tracked by the database context.

Chapter 05: Use EF Migrations in an ASP.NET MVC app and deploy to Azure

1. What is migration feature in entity Framework?
    - The migrations feature in Entity Framework provides a way to incrementally update the database schema to keep it in sync with the application's data model while preserving existing data in the database.
2. Explain entity Framework migration commands.
    - The migration commands are:
    - enable-migrations - creates migrations folders and save a Configuration file in the folder. This file contains Seed method to populate test data
    - add-migration <migration-name> - create a timestamp_migration-name.cs file that contains Up method to create tables and Down method to delete tables
    - update-database - runs the Up method to create the database and then it runs the Seed method to populate the database

Chapter 07: Read related data with EF in an ASP.NET MVC app

1. What are the ways to load related data in Entity Framework?
    - Lazy loading. When the entity is first read, related data isn't retrieved. However, the first time you attempt to access a navigation property, the data required for that navigation property is automatically retrieved.

      ```
      departments = context.Departments
      foreach (Department d in dep,artments) // Department rows
      {
       foreach (Course C in d.Courses}//Query: Course rows related to
      {
          Department d courseList.Add(d.,ll\lame + c .Title); }
      }
      ```
    - Eager loading. When the entity is read, related data is retrieved along with it. This typically results in a single join query that retrieves all of the data that's needed. You specify eager loading by using the Include method.

      ```
      departments = context .Departments. Include (x => x.Courses)
      ```

```
foreach (Department d in departments) //Department
{
foreach (Course c in d.Courses) //rows and related course
{
courseList.Add(d.Hame + c.Title);
 }
}
```

- Explicit loading. This is similar to lazy loading, except that you explicitly retrieve the related data in code

```
foreach (Department d in departments) // Department rows
{
 foreach (Course C in d.Courses}//Query: Course rows related to
{
    context .Entry(d) .Collection(x => c.Courses).Load() ;
    courseList.Add(d.Name + c.Title)
}
```

Chapter 10: Implement Inheritance with EF in an ASP.NET MVC 5 app

1. What are the available inheritance patterns in Entity Framework?
    - Table per Hierarchy (TPH): This approach suggests one table for the entire class inheritance hierarchy. The table includes a discriminator column which distinguishes between inheritance classes
    - Table per Type (TPT): This approach suggests a separate table for each domain class.
    - Table per Concrete Class (TPC): This approach suggests one table for one concrete class, but not for the abstract class.