Microsoft

Exam 70-487

Developing Windows Azure and Web Services

Version: 8.4

[Total Questions: 113]

Topic break down

Topic	No. of Questions
Topic 1: Flight Information	16
Topic 2: ASP.NET MVC	16
Topic 3: Online Bookstore	15
Topic 4: Mix Questions	66

Topic 1, Flight Information

Background

You are developing a flight information consolidation service. The service retrieves flight information from a number of sources and combines them into a single data set. The consolidated flight information is stored in a SQL Server database. Customers can query and retrieve the data by using a REST API provided by the service.

The service also offers access to historical flight information. The historical flight information can be filtered and queried in an ad hoc manner.

The service runs on a Windows Azure Web Role. SSL is not used.

Business Requirements

- A new data source for historical flight information is being developed by a contractor located on another continent.
- If a time zone is not specified, then it should be interpreted as Coordinated Universal Time (UTC).
- When you upgrade a service from a staging deployment to a production deployment, the time that the service is unavailable must be minimized.
- The default port must be used for HTTP.

Technical Requirements

The existing sources of flight information and the mechanism of exchange are listed below.

- Blue Yonder Airlines provides flight information in an XML file.
- Consolidated Messenger provides flight information in a Microsoft Access database that is uploaded every 12 hours to the service using SFTP. The company uses port 22 for SFTP.
- Margie's Travel provides and consumes flight information using serialized ADO.NET DataSets. Data is periodically synced between the service and Margie's Travel.
- Trey Research provides data from multiple sources serialized in proprietary binary formats. The data must be read by using .NET assemblies provided by Trey Research. The assemblies use a common set of dependencies. The current version of the Trey Research assemblies is 1.2.0.0. All assemblies provided by Trey Research are signed with a key pair contained in a file named Trey.snk, which Trey Research also supplies.
- The application specification requires that any third-party assemblies must have strong names.

Application Structure

FlightInfo.cs

```
public class FlightInfo
{
    string DataSource { get; set; }
    public string Airline { get; set; }
    public string Flight { get; set; }
    public DateTimeOffset Arrival { get; set; }
    public int Seats { get; set; }
    public bool WasLate { get; set; }
}
```

BlueYonderLoader.cs

```
public class BlueYonderLoader
{
    public IEnumerable<RawFlightData> LoadFlights(XDocument feed)
    {
        ...
    }

    private RawFlightData Parse(XElement flightElement)
    {
        ...
    }
}
```

HistoricalDataLoader.cs

```
public class HistoricalDataLoader
{
    public static IEnumerable<HistoricalFlightInfo> LoadHistoricalFlights()
    {
        ...
    }

    public void StreamHistoricalFlights(XmlWriter responseWriter, string airline)
    {

        ...
    }

    private XElement ConvertToHistoricalFlight(XElement flight)
    {
        return new XElement("Flight", flight);
    }

    private string GetAirline(XElement flightName)
    {
        return flightName.Value.Substring(0, 2);
    }

    IEnumerable<XElement> RemoteDataStream()
    {
        return XDocument.Load("").Elements();
    }
}
```

MargiesTravelSync.cs

```csharp
public class MargiesTravelSync
{
  public void Sync()
  {
    ...
  }

  private DataSet LoadLocal()
  {
    var dataSet = new DataSet();
    dataSet.ReadXml("local");
    return dataSet;
  }

  private StreamWriter SendStream()
  {
    return new StreamWriter("SendStream");
  }

  private StreamReader ReceiveStream()
  {
    return new StreamReader("ReceiveStream");
  }
}
```

FlightInfoContext.cs

```
public class FlightInfoContext : DbContext
{
  public DbSet<FlightInfo> FlightInfo { get; set; }

  public override int SaveChanges()
  {
    return base.SaveChanges();
  }

  private bool IsTransient(int ex)
  {
    var errors = new[] { 10053, 10054, 64 };
    return errors.Contains(ex);
  }
}
```

FlightDataController.cs

```
public class FlightDataController : ApiController
{
  FlightInfoContext _Context;

  public FlightDataController()
  {
    _Context = new FlightInfoContext();
  }

  [HttpGet]
  public IEnumerable<FlightInfo> GetFlightInfo()
  {
    return _Context.FlightInfo.Select(x => x).AsEnumerable();
  }

  private IEnumerable<HistoricalFlightInfo> LoadHistorical()
  {
    return HistoricalDataLoader.LoadHistoricalFlights();
  }
}
```

Question No : 1 DRAG DROP - (Topic 1)

The service has been deployed to Windows Azure.

Trey Research has provided version 1.3.0.0 of the assembly to support a change in the serialization format. The service must remain available during the transition to the new serialization format.

You need to ensure that the service is using the new assembly.

Which configuration setting should you add to the web.config? (To answer, drag the appropriate configuration elements to the correct location or locations in the answer area. Each configuration element may be used once, more than once, or not at all. You may need to drag the split bar between panes or scroll to view content.)

```
codeBase version="1.3.0.0" href="Trey.Serialization.dll"

bindingRedirect oldVersion="1.2.5.0" newVersion="1.3.0.0"

bindingRedirect oldVersion="1.2.0.0" newVersion="1.3.0.0"

runtime

location
```

```
<      >
```

```
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
  <dependentAssembly>
    <assemblyIdentity name="Trey.Serialization" />

    <

  </dependentAssembly>
</assemblyBinding>
```

```
</      >
```

Answer:

```
codeBase version="1.3.0.0" href="Trey.Serialization.dll"

bindingRedirect oldVersion="1.2.5.0" newVersion="1.3.0.0"

bindingRedirect oldVersion="1.2.0.0" newVersion="1.3.0.0"

runtime

location
```

```
< runtime >

  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <dependentAssembly>
      <assemblyIdentity name="Trey.Serialization" />


      < codeBase version="1.3.0.0" href="Trey.Serialization.dll"

    </dependentAssembly>
  </assemblyBinding>

</ runtime >
```

Question No : 2 - (Topic 1)

You are adding a new REST service endpoint to the FlightDataController controller. It returns flights from the consolidated data sources only for flights that are late.

You need to write a LINQ to Entities query to extract the required data.

Which code segment should you use?

```
A.  var historical = LoadHistorical();
    var query = _Context.FlightInfo.AsQueryable()
     .Join(historical, x => x.Flight, y => y.Flight, (x, y) => new { Current = x,
    Historical = y })
     .Where(x => x.Historical.WasLate)
     .Select(x => x.Current);

B.  var historical = LoadHistorical();
    var query = _Context.FlightInfo.AsEnumerable()
     .Where(x => historical.All(y => y.WasLate && x.Flight == y.Flight))
     .Select(x => x);

C.  var historical = LoadHistorical();
    var query = _Context.FlightInfo.AsQueryable()
     .Where(x => historical.Select(y => y.Flight).Contains(x.Flight))
     .Where(x => historical.Any(y => y.WasLate))
     .Select(x => x);

D.  var historical = LoadHistorical();
    var query = _Context.FlightInfo.AsEnumerable()
     .Join(historical, x => x.Flight, y => y.Flight, (x, y) => new { Current = x,
    Historical = y })
     .Where(x => x.Historical.WasLate)
     .Select(x => x.Current);
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: D

Explanation: Explanation/Reference:

D is right because you send result as REST so if you use "AsQueryable" the result is deferred to the next enumeration of your result.

D is not optimized but will works.

A will break at runtime.

Credits to Rem

Question No : 3 - (Topic 1)

Errors occasionally occur when saving data using the FlightInfoContext ADO.NET Entity Framework context. Updates to the data are being lost when an error occurs.

You need to ensure that data is still saved when an error occurs by retrying the operation. No more than five retries should be performed.

With which code segment should you replace the body of the SaveChanges() method in

---

the FlightInfoContext.es file?

```
C A.   var result = FlightInfo.SqlQuery("UPDATE WITH RETRY", FlightInfo, "IsTransient", 5);
       if (result.Count() > 5)
       {
         result.AsNoTracking();
         return -1;
       }
       return 0;


C B.   try
       {
         return base.SaveChanges();
       }
       catch (EntityCommandExecutionException ex)
       {
         if (ex.Data.Keys.Cast<int>().Any(x => IsTransient(x)))
         {
           return 5 & SaveChanges();
         }
         return -1;
       }

C C.   for (var i = 0; i < 5; i++)
       {
         try
         {
           return base.SaveChanges();
         }
         catch (SqlException ex)
         {
           if (IsTransient(ex.Number))
           {
             continue;
           }
         }
       }
       return base.SaveChanges();


C D.   var exception = new EntitySqlException();
       while (exception.HResult != 0 && exception.Data.Count < 5)
       {
         try
         {
           return base.SaveChanges();
         }
         catch (EntitySqlException ex)
         {
           if (IsTransient(ex.HResult))
           {
             exception = ex;
           }
         }
       }
       return base.SaveChanges();
```

A. Option A
B. Option B
C. Option C
D. Option D

Answer: C

Explanation: Explanation/Reference:

EntitySqlException: Represents errors that occur when parsing Entity SQL command text.
This exception is thrown when syntactic or semantic rules are violated.

SqlException: The exception that is thrown when SQL Server returns a warning or error.
This class cannot be inherited.

EntityCommandExecutionException : Represents errors that occur when the underlying