- **Vendor: Microsoft**
- **Exam Code: 70-487**
- **Exam Name: Developing Microsoft Azure and Web Services**
- **Question 41 -- Question 60**

QUESTION 41
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:
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.

QUESTION 42
Data provided by Consolidated Messenger is cached in the HttpContext.Cache object. You need to ensure that the cache is correctly updated when new data arrives. What should you do?

A. Ensure that the EffectivePrivateBytesLimit value is greater than the size of the database file.
B. Change the sliding expiration of the cache item to 12 hours.
C. Use the SqlCacheDependency type configured with a connection string to the database file.
D. Use the CacheDependency type configured to monitor the SFTP target folder.

Answer: D

QUESTION 43
You need to load flight information provided by Consolidated Messenger. Which should you use?

A. SQL Server Data Transformation Services (DTS)
B. EntityTransaction and EntityCommand
C. Office Open XML
D. OleDbConnection and OleDbDataReader

Answer: D

QUESTION 44
Drag and Drop Question
You need to parse flight information from Blue Yonder Airlines. The content of the XML file is shown below.

```
<?xml version="1.0" encoding="utf-8"?>
<AirlineFeed>
  <Flight xmlns="urn:CFI" name="AS515">
    <Seats>123</Seats>
    <Arrival>5/2/2011 12:01:13</Arrival>
  </Flight>
  <Flight name="UN24">
    <Seats>123</Seats>
    <Arrival>5/1/2012 10:17:57 PM +02:00</Arrival>
  </Flight>
  <FlightManifest>
    ...
  </FlightManifest>
</AirlineFeed>
```

Some airlines do not specify the timezone of the arrival time. If the timezone is not specified, then it should be interpreted per the business requirements. You need to implement the LoadFlights() and Parse() methods of the BlueYonderLoader class. What should you do? (To answer, drag the appropriate code segments to the correct location in the answer area. Each segment 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.)

```
var flights = feed.Elements(
 feed.Root.GetPrefixOfNamespace("{urn:CFI}") + "Flight");
```

```
var flights = feed.Descendants().Where(x =>
 x.NodeType != XmlNodeType.XmlDeclaration && (string)x ==
"Flight");
```

```
var flights = feed.Descendants("{urn:CFI}Flight")
 .Concat(feed.Descendants("Flight"));
```

```
fi.Arrival = DateTimeOffset.Parse(arrivalRaw,
 null, System.Globalization.DateTimeStyles.AssumeUniversal);
```

```
fi.Arrival = DateTimeOffset.Parse(arrivalRaw,
 null, System.Globalization.DateTimeStyles.AdjustToUniversal);
```

```
fi.Arrival = XmlConvert.ToDateTimeOffset(arrivalRaw,
 new[] { "Local", "Universal" });
```

```
public IEnumerable<FlightInfo> LoadFlights(XDocument feed)
{
  return flights.Select(x => Parse(x));
}

private FlightInfo Parse(XElement flightElement)
{
  var fi =  new FlightInfo();
  fi.Flight = flightElement.Attribute("name").Value;
  var arrivalRaw = flightElement.Element("Arrival").Value;

  fi.Seats = XmlConvert.ToInt32(flightElement.Element("Seats").Value);
  return fi;
}
```

Answer:

```
var flights = feed.Elements(
 feed.Root.GetPrefixOfNamespace("{urn:CFI}") + "Flight");
```

```
var flights = feed.Descendants().Where(x =>
 x.NodeType != XmlNodeType.XmlDeclaration && (string)x ==
"Flight");
```

```
var flights = feed.Descendants("{urn:CFI}Flight")
 .Concat(feed.Descendants("Flight"));
```

```
fi.Arrival = DateTimeOffset.Parse(arrivalRaw,
 null, System.Globalization.DateTimeStyles.AssumeUniversal);
```

```
fi.Arrival = DateTimeOffset.Parse(arrivalRaw,
 null, System.Globalization.DateTimeStyles.AdjustToUniversal);
```

```
fi.Arrival = XmlConvert.ToDateTimeOffset(arrivalRaw,
 new[] { "Local", "Universal" });
```

```
public IEnumerable<FlightInfo> LoadFlights(XDocument feed)
{
    var flights = feed.Descendants("{urn:CFI}Flight")
     .Concat(feed.Descendants("Flight"));
  return flights.Select(x => Parse(x));
}

private FlightInfo Parse(XElement flightElement)
{
  var fi =  new FlightInfo();
  fi.Flight = flightElement.Attribute("name").Value;
  var arrivalRaw = flightElement.Element("Arrival").Value;
    fi.Arrival = DateTimeOffset.Parse(arrivalRaw,
     null, System.Globalization.DateTimeStyles.AssumeUniversal);
  fi.Seats = XmlConvert.ToInt32(flightElement.Element("Seats").Value);
  return fi;
}
```

QUESTION 45

You are adding a new REST service endpoint to the FlightDataController controller that returns the total number of seats for each airline. You need to write a LINQ to Entities query to extract the required data. Which code segment should you use?

A.
```
var query = from flight in _Context.FlightInfo
 group flight by flight.Seats into agg
 let airline = agg.First()
 select new
 {
   TotalSeats = agg.Key,
   Airline = airline,
 };
```

B.
```
var query = from flight1 in _Context.FlightInfo
 from flight2 in _Context.FlightInfo
 where flight1.Airline == flight2.Airline
 select new
 {
   Airline = flight1.Airline,
   TotalSeats = Math.BigMul(flight1.Seats, flight2.Seats),
 };
```

C.
```
var query = from flight in _Context.FlightInfo
 from airline in flight.Airline
 group airline by airline into agg
 select new
 {
   Airline = agg.Key,
   TotalSeats = agg.Sum(x => Convert.ToInt32(x)),
 };
```

D.
```
var query = from flight in _Context.FlightInfo
 group flight by flight.Airline into agg
 select new
 {
   Airline = agg.Key,
   TotalSeats = agg.Sum(x => x.Seats),
 };
```

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

Answer: D

QUESTION 46
Historical flight information data will be stored in Windows Azure Table Storage using the FlightInfo class as the table entity. There are millions of entries in the table. Queries for historical flight information specify a set of airlines to search and whether the query should return only late flights. Results should be ordered by flight name. You need to specify which properties of the FlightInfo class should be used at the partition and row keys to ensure that query results are returned as quickly as possible. What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

A. Use the WasLate property as the row key.
B. Use the Airline property as the row key.
C. Use the WasLate property as the partition key.
D. Use the Arrival property as the row key.
E. Use the Airline property as the partition key.
F. Use the Flight property as the row key.

Answer: EF

QUESTION 47
Transformed historical flight information provided by the RemoteDataStream() method must be written to the response stream as a series of XML elements named Flight within a root element named Flights. Each Flight element has a child element named FlightName that contains the flight name that starts with the two-letter airline prefix. You need to implement the StreamHistoricalFlights() method so that it minimizes the amount of memory allocated. Which code segment should you use as the body of the StreamHistoricalFlights() method in the HistoricalDataLoader.es file?

A.
```
responseWriter.WriteStartElement("Flights");
var flights = RemoteDataStream()
 .OrderBy(x => GetAirline(x.Element("FlightName")));
var filteredFlights = flights
 .SkipWhile(x => GetAirline(x.Element("FlightName")) != airline);
foreach (var f in filteredFlights)
{
  var flight = ConvertToHistoricalFlight(f);
  flight.WriteTo(responseWriter);
}
responseWriter.WriteEndElement();
```

B.
```
responseWriter.WriteStartElement("Flights");
var flights = RemoteDataStream().Select(x =>
 {
   if (GetAirline(x) == airline)
   {
     return ConvertToHistoricalFlight(x);
   }
   return null;
 });
flights.TakeWhile(x =>
 {
   x.WriteTo(responseWriter);
   return x != null;
 });
responseWriter.WriteEndElement();
```

C.
```
var data = RemoteDataStream().ToDictionary(x =>
 GetAirline(x.Element("FlightName")),
 x => new XStreamingElement("Flights", ConvertToHistoricalFlight(x).Descendants()));
data[airline].WriteTo(responseWriter);
```

D.
```
var flights = new XStreamingElement("Flights",
 from flight in RemoteDataStream()
 where GetAirline(flight.Element("FlightName")) == airline
 select ConvertToHistoricalFlight(flight));
flights.WriteTo(responseWriter);
```

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

D. Option D

Answer: D
Explanation:
http://msdn.microsoft.com/en-us/library/system.xml.linq.xstreamingelement.aspx
http://msdn.microsoft.com/en-us/library/bb551307.aspx

QUESTION 48
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?

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

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

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

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

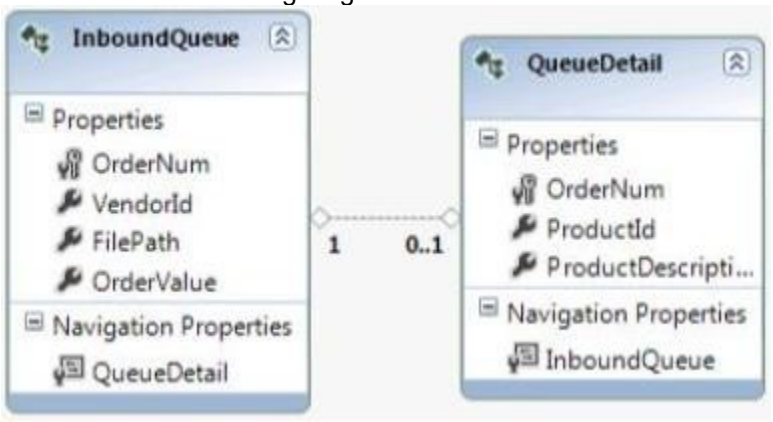
Explanation:
- EntitySqlException: Represents errors that occur when parsing Entity SQL command text. This exception is thrown when syntactic or semantic rules are violated.
- SqlException: The exception that is thrown when SQL Server returns a warning or error. This class cannot be inherited.
- EntityCommandExecutionException: Represents errors that occur when the underlying storage provider could not execute the specified command. This exception usually wraps a provider-specific exception.

Case Study 2 - ASP.NET MVC (QUESTION 49 - QUESTION 63)

Background
You are developing an ASP.NET MVC application in Visual Studio 2012 that will be used to process orders.
Business Requirements
The application contains the following three pages.
- A page that queries an external database for orders that are ready to be processed. The user can then process the order.
- A page to view processed orders.
- A page to view vendor information.

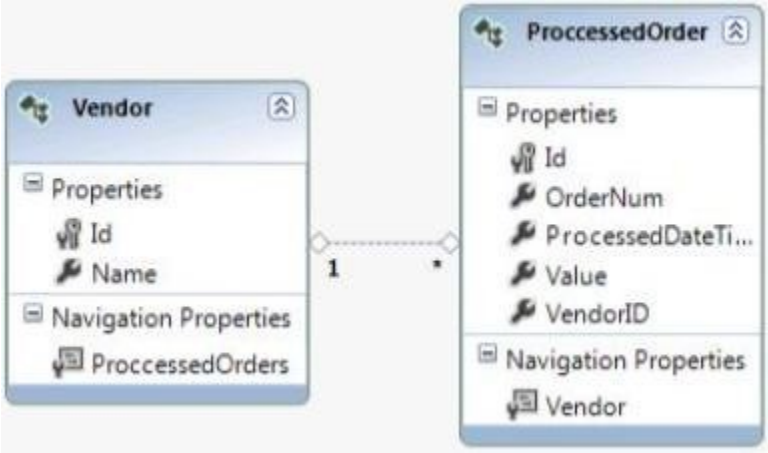
The application consumes three WCF services to retrieve external data.
Technical Requirements
Visual Studio Solution:
The solution contains the following four projects.
- ExternalQueue: A WCF service project used to communicate with the external order database.
- OrderProcessor: An ASP.NET MVC project used for order processing and logging order metadata.
- OrderUpload: A WCF service project used to submit order data to an external data source.
- Shipping: A WCF service project used to acquire shipping information.

ExternalQueue Project:
Entity Framework is used for data access. The entities are defined in the ExternalOrders.edmx file as shown in the following diagram.



The project contains two services defined in the following files.
- IExternalQueueService.es
- ExternalQueueService.svc

The ExternalQueue.Helpers namespace contains a definition for a class named OrderNotFound Exception.
OrderProcessor Project:
Entity Framework is used for data access. The entities are defined in the ProcessedOrders.edmx file as shown in the following diagram.



The classes are contained in the OrderProcessor.Entities namespace. The project contains the following two controllers.

- InboundQueueController.es
- ProcessedOrderController.es

WCF service proxies to the ExternalQueue, Shipping and OrderUpload services have been generated by using the command prompt. The ExecuteCommandProcedure() method in the ExternalQueueService.svc file must run asynchronously. The ProcessedOrderController controller has the following requirements. The GetVendorPolicy() method must enforce a 10 minute absolute cache expiration policy. The GetProcessedOrders() method must return a view of the 10 most recently processed orders.

OrderUpload Project:

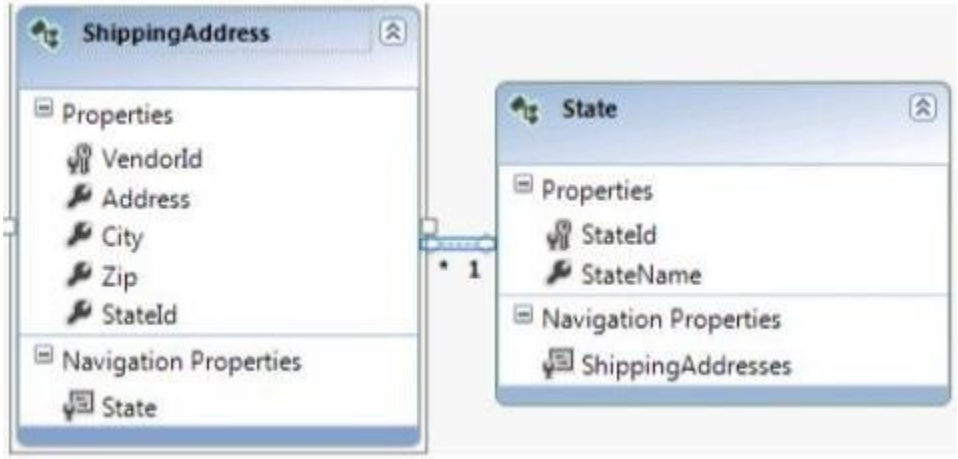
The project contains two services defined in the following files.

- IUploadCallbackService.es
- UploadCallbackService.svc

Data Access is maintained in a file named UploadOrder.es.

Shipping Project:

Entity Framework is used for data access. The entities are defined in the ExternalOrders.edmx file as shown in the following diagram.



The Custom Tool property for ExternalOrders.edmx has been removed. POCO classes for the Entity Model are located in the ShippingAddress.es file. The POCO entity must be loaded by using lazy loading. The project contains two services defined in the following files.

- IShippingService.es
- ShippingService.svc

The IShippingService contract must contain an operation that receives an order number as a parameter. The operation must return a class named ShippingInfo that inherits from a class named

State.

Application Structure

ExternalQueue\IExternalQueueService.cs

```
IQ01 using System.Collections.Generic;
IQ02 using System.ServiceModel;
IQ03 using ExternalQueue.Helpers;
IQ04
IQ05 namespace ExternalQueue
IQ06 {
IQ07   [ServiceContract]
IQ08   public interface IExternalQueueService
IQ09   {
IQ10     [OperationContract]
IQ11     List<Entities.InboundQueue> GetExternalOrders();
IQ12
IQ13     [FaultContract(typeof(OrderNotFoundException))]
IQ14     [OperationContract]
IQ15     void DeleteExternalOrder(int orderNum);
IQ16
IQ17     [OperationContract]
IQ18     Entities.InboundQueue GetExternalOrder(int orderNum);
IQ19   }
IQ20 }
```

ExternalQueue\ProcessedOrderController.cs

```
PC01 using System;
PC02 using System.Collections.Generic;
PC03 using System.Linq;
PC04 using System.Runtime.Caching;
PC05 using System.Web.Mvc;
PC06 using OrderProcessor.Entities;
PC07 using OrderProcessor.Helpers;
PC08 using System.Configuration;
PC09
PC10 namespace OrderProcessor.Controllers
PC11 {
PC12   public class ProcessedOrderController : Controller
PC13   {
PC14     public ActionResult GetProcessedOrders()
PC15     {
PC16       using (var context = new ProcessedOrders())
PC17       {
PC18         List<Entities.ProccessedOrder> orders = new List<ProccessedOrder>();
PC19         return View(orders);
PC20       }
PC21     }
PC22
PC23     private ObjectCache cache {get { return MemoryCache.Default; }}
PC24
PC25     public ActionResult GetVendors()
PC26      {
PC27        List<Entities.Vendor> vendors = cache.Get
("vendorKey") as List<Entities.Vendor>;
PC28        if (vendors == null)
PC29        {
PC30          using (var context = new ProcessedOrders())
PC31          {
PC32            vendors = context.Vendors.ToList();
PC33
PC34          }
PC35        }
PC36        return View(vendors);
PC37      }
PC38
PC39      private CacheItemPolicy GetVendorPolicy()
PC40      {
PC41        CacheItemPolicy vendorPolicy = new CacheItemPolicy();
PC42
PC43        return vendorPolicy;
PC44      }
PC45
PC46      private List<string> GetTriggerPaths()
PC47      {
PC48        List<string> triggerPath = new List<string>();
PC49        triggerPath.Add(@"c:\triggers\vendortrigger.txt");
PC50        return triggerPath;
PC51      }
PC52   }
PC53 }
```

OrderProcessor\InboundQueueController.cs

```
IC01 using System;
IC02 using System.Collections.Generic;
IC03 using System.Web.Mvc;
IC04 using OrderProcessor.Entities;
IC05 using ExternalQueue.Entities;
IC06 using System.ServiceModel;
IC07 using System.Collections;
IC08 using ExternalQueue.Helpers;
IC09 using OrderProcessor.Helpers;
IC10 using System.Linq;
IC11
IC12 namespace OrderProcessor.Controllers
IC13 {
IC14   public class InboundQueueController : Controller
IC15   {
IC16     public ActionResult GetQueueItems()
IC17     {
IC18       IEnumerable<InboundQueue> inboundOrders = Enumerable.Empty<InboundQueue>();
IC19       return View(inboundOrders);
IC20     }
IC21
IC22     public ActionResult ProcessOrder(int orderNum)
IC23     {
IC24       ExternalQueueServiceClient qService = new ExternalQueueServiceClient();
IC25       InboundQueue externalOrder = qService.GetExternalOrder(orderNum);
IC26       if (externalOrder != null)
IC27       {
IC28         using (var context = new ProcessedOrders())
IC29         {
IC30           ProccessedOrder order = new ProccessedOrder();
IC31           order.OrderNum = externalOrder.OrderNum;
IC32           order.Value = Convert.ToDouble(externalOrder.OrderValue);
IC33           order.VendorID = Convert.ToInt32(externalOrder.VendorId);
IC34           order.ProcessedDateTime = DateTime.Now;
IC35           context.ProccessedOrders.Add(order);
IC36           context.SaveChanges();
IC37         }
IC38         qService.DeleteExternalOrder(orderNum);
IC39       }
IC40       return RedirectToAction("GetQueueItems");
IC41     }
IC42
IC43     public ActionResult ViewShippingInfo(int orderNum)
IC44     {
IC45       ShippingServiceClient shipService = new ShippingServiceClient();
IC46       var info = shipService.GetShippingInfo(orderNum);
IC47       return View(info);
IC48     }
IC49   }
IC50 }
```

OrderUpload\IUploadCallbackService.cs

```
IU01 using System.ServiceModel;
IU02
IU03 namespace OrderUpload
IU04 {
IU05   [ServiceContract(CallbackContract = typeof(IUploadCallback))]
IU06   public interface IUploadCallbackService
IU07   {
IU08     [OperationContract]
IU09     void UploadOrder(int orderNum);
IU10   }
IU11
IU12   public interface IUploadCallback
IU13   {
IU14     [OperationContract]
IU15     decimal GetOrderValue(int orderNum);
IU16   }
IU17 }
```

OrderUpload\UploadCallbackService.svc

```
US01 using System.ServiceModel;
US02
US03 namespace OrderUpload
US04 {
US05   public class UploadCallbackService : IUploadCallbackService
US06   {
US07     public void UploadOrder(int orderNum)
US08     {
US09     }
US10   }
US11 }
```

Shipping\IShippingService.cs

```
IS01 using System.Runtime.Serialization;
IS02 using System.ServiceModel;
IS03
IS04 namespace Shipping
IS05 {
IS06   public interface IShippingService
IS07   {
IS08
IS09   }
IS10 }
```

Shipping\ShippingAddress.cs

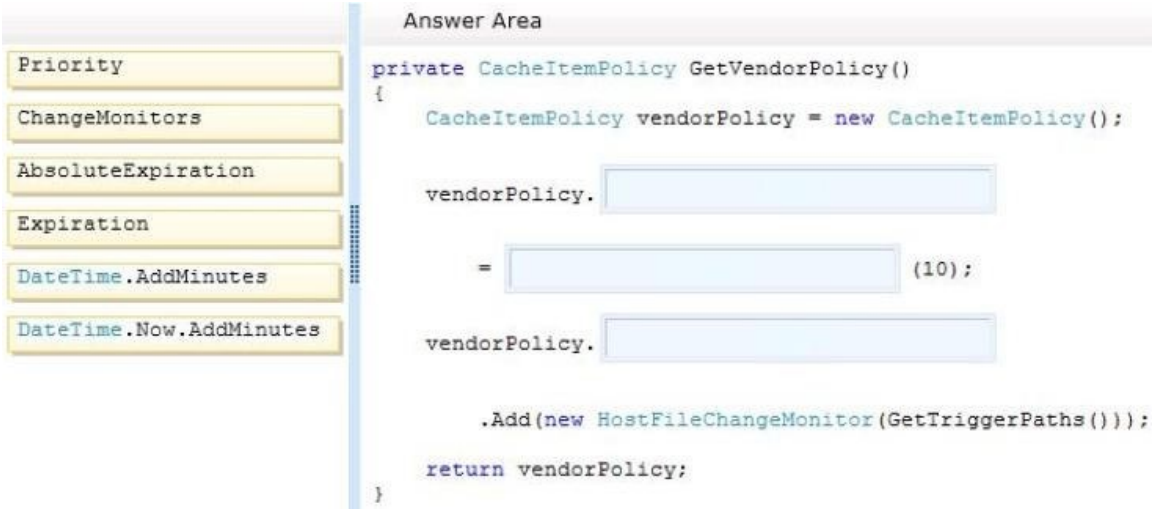
```
SA01 using System.Collections.Generic;
SA02 using System.Data.Objects;
SA03
SA04 namespace Shipping.POCO
SA05 {
SA06   public class ShippingAddress
SA07   {
SA08     public int VendorId { get; set; }
SA09     public string Address { get; set; }
SA10     public string City { get; set; }
SA11     public int StateId { get; set; }
SA12     public string Zip { get; set; }
SA13     public State State { get; set; }
SA14   }
SA15
SA16   public class State
SA17   {
SA18     public int StateId { get; set; }
SA19     public string StateName { get; set; }
SA20     public List<ShippingAddress> ShippingAddresses { get; set; }
SA21   }
SA22 }
```

QUESTION 49
The QueueDetail entity type must inherit from the InboundQueue entity type in the ExternalQueue service project using table-per-type inheritance. You need to modify the entities in the designer. What should you do? (Each correct answer presents part of the solution. Choose all that apply.)

A. Remove the OrderNum property in InboundQueue.
B. Remove the OrderNum property in QueueDetail.
C. Set the QueueDetail BaseType to InboundQueue.
D. Remove the association between the entities.
E. Right-click the entities and validate the table mapping.
F. Set the InboundQueue BaseType to QueueDetail.

Answer: BCDE
Explanation:
http://www.robbagby.com/entity-framework/entity-framework-modeling-table-per-type-inheritance/

QUESTION 50
Drag and Drop Question
The GetVendorPolicy() private method in the ProcessedOrderController controller is returning a CacheItemPolicy object with default values. The returned policy must expire if the external file located at C:\Triggers\VendorTrigger.txt has been modified or the timeout outlined in the technical requirements is reached. You need to return the policy. How should you build the method? (To answer, drag the appropriate code segments to the correct location or locations in the answer area. Each code segment 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.)



Answer:

QUESTION 51

The GetExternalOrder() method in the ExternalQueueService service is throwing a runtime error. The method must query the database for a record that matches the orderNum parameter passed to the method. You need to modify the queryString string to retrieve the record. With which code segment should you replace line EQ64?

A.
```
string queryString = @"SELECT VALUE q FROM ExternalOrdersEntities.InboundQueues AS q
  WHERE q.OrderNum = @orderNum";
```

B.
```
string queryString = @"SELECT VALUE * FROM ExternalOrdersEntities.InboundQueues
  WHERE OrderNum = @orderNum";
```

C.
```
string queryString = @"SELECT q.OrderNum, q.VendorId, q.FilePath, q.OrderValue
  FROM ExternalOrdersEntities AS q WHERE q.OrderNum = @orderNum";
```

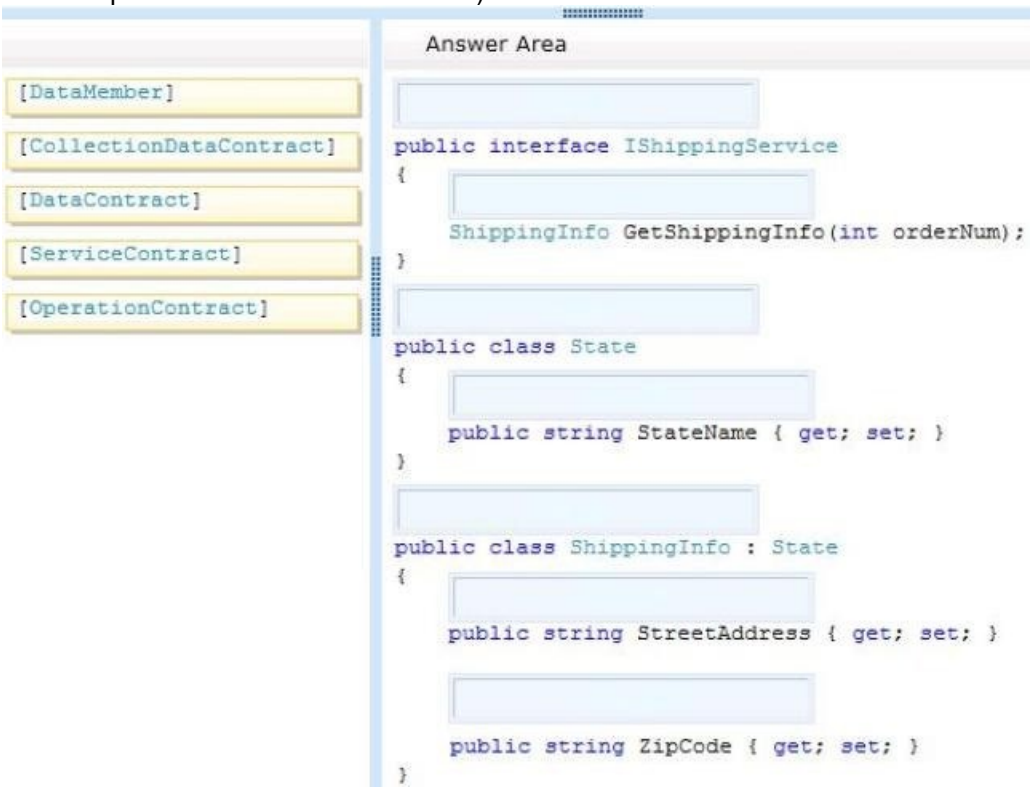
D.
```
string queryString = @"SELECT q FROM ExternalOrdersEntities.InboundQueues
  WHERE q.OrderNum = @orderNum";
```

A. Option A
B. Option B

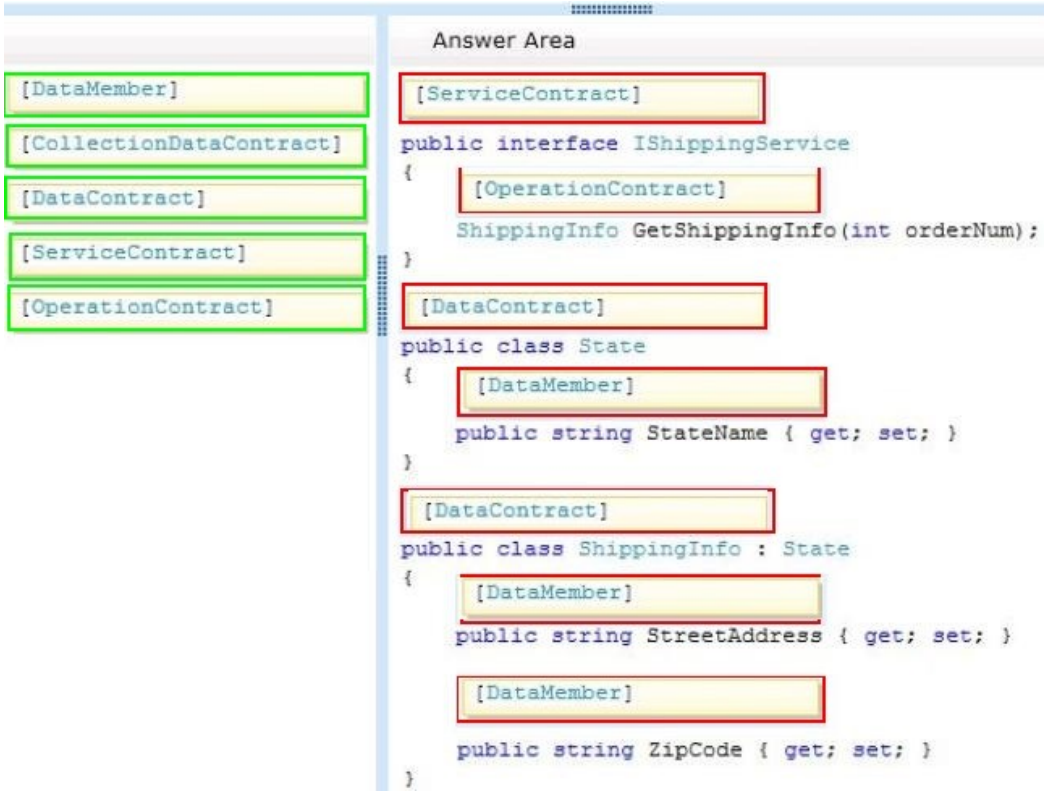
C. Option C
D. Option D

Answer: A
Explanation:
http://www.entityframeworktutorial.net/Querying-with-EDM.aspx

QUESTION 52
Drag and Drop Question
You add a class named ShippingInfo. You need to modify the IShippingService interface and the ShippingInfo class to meet the technical requirements. What should you do? (To answer, drag the appropriate code segments to the correct location or locations in the answer area. Each code segment 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.)



Answer:



QUESTION 53

Drag and Drop Question

You need to create the ShippingContext class in the ShippingAddress.es file to meet the requirements. What should you do? (To answer, drag the appropriate code segments to the correct location or locations in the answer area. Each code segment 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.)

Answer:

QUESTION 54

You need to modify the ExecuteCommandProcedure() method to meet the technical requirements.
Which code segment should you use?

A.
```
private async Task ExecuteCommandProcedure(EntityCommand command)
{
  using (EntityConnection connection = new EntityConnection
("name=ExternalOrdersEntities"))
  {
    command.Connection = connection;
    await connection.OpenAsync();
    await command.ExecuteNonQueryAsync();
  }
}
```

B.
```
private void ExecuteCommandProcedure(EntityCommand command)
{
  using (EntityConnection connection = new EntityConnection
("name=ExternalOrdersEntities"))
  {
    command.Connection = connection;
    command.ExecuteNonQueryAsync();
  }
}
```

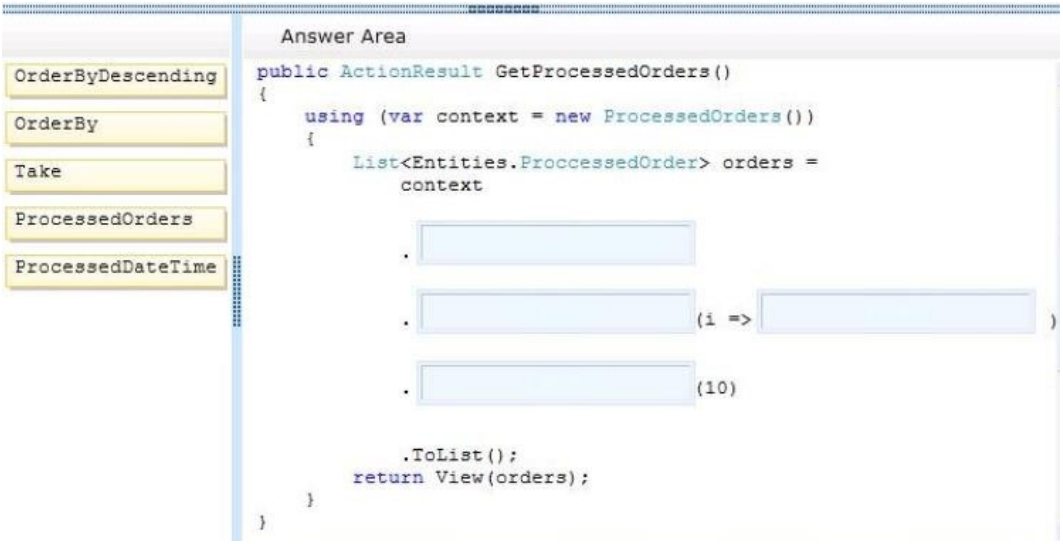
C.
```
private void ExecuteCommandProcedure(EntityCommand command)
{
  using (EntityConnection connection = new EntityConnection
("name=ExternalOrdersEntities"))
  {
    command.Connection = connection;
    connection.OpenAsync();
    command.ExecuteNonQueryAsync();
  }
}
```

D.
```
private async Task ExecuteCommandProcedure(EntityCommand command)
{
  using (EntityConnection connection = new EntityConnection
("name=ExternalOrdersEntities"))
  {
    command.Connection = connection;
    connection.OpenAsync();
    command.ExecuteNonQueryAsync();
  }
}
```

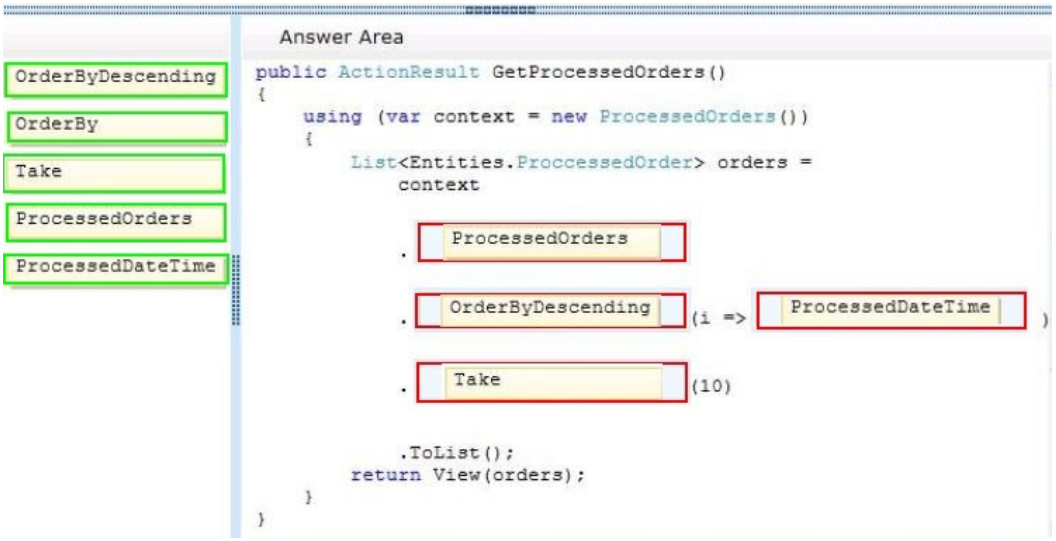
A. Option A
B. Option B
C. Option C
D. Option D

Answer: A

QUESTION 55
Drag and Drop Question
You need to complete the GetProcessedOrders() action in the ProcessedOrderController controller to meet the requirements. What should you do? (To answer, drag the appropriate code segments to the correct location or locations in the answer area. Each code segment 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.)



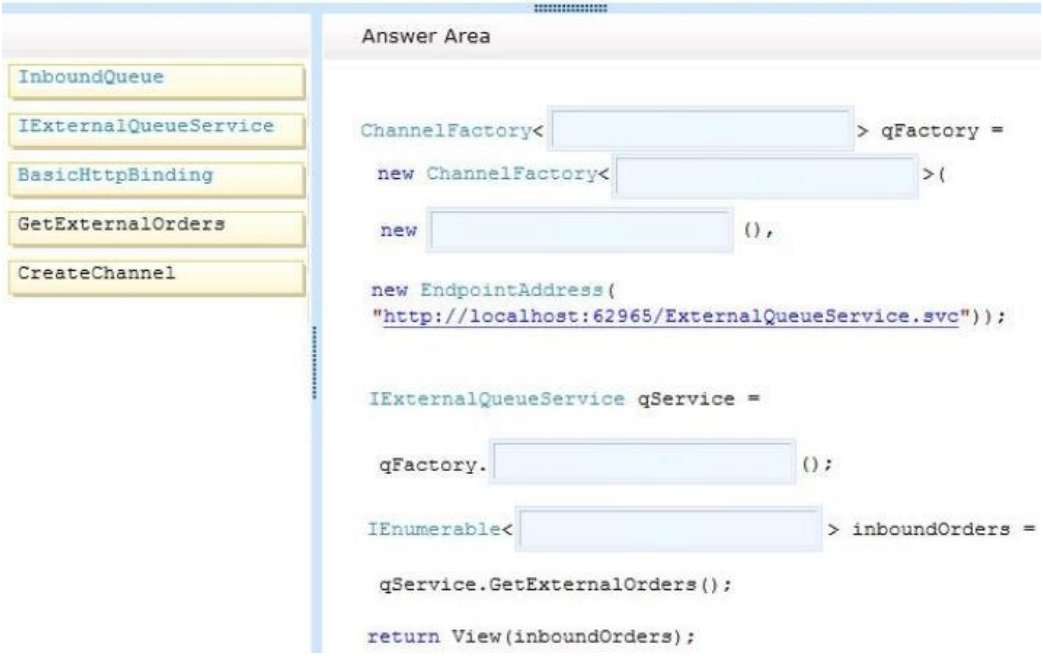
Answer:



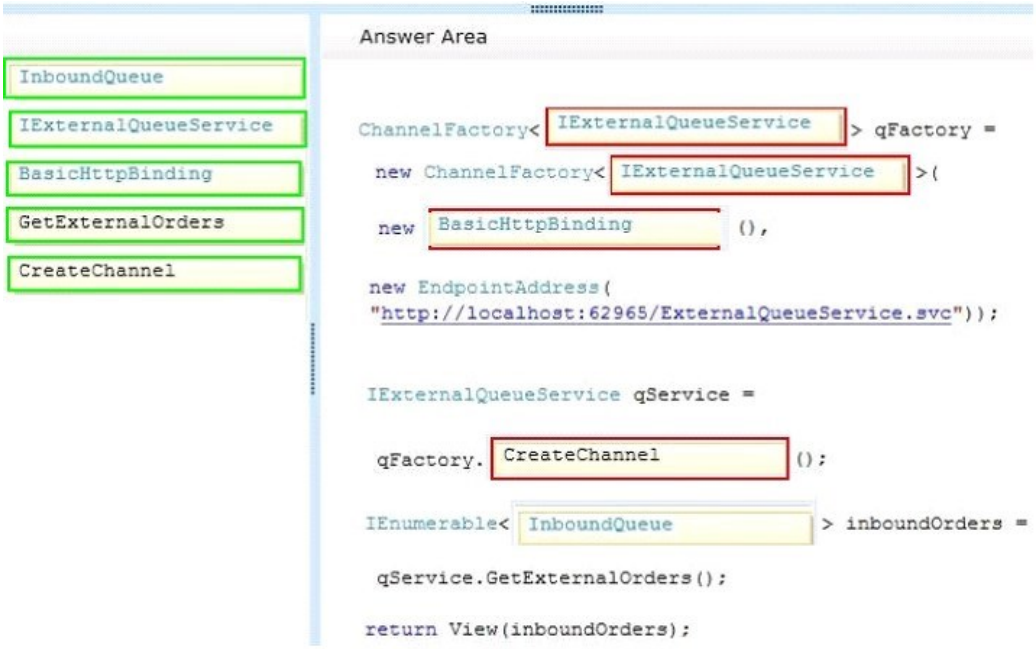
QUESTION 56

Drag and Drop Question

The GetQueueItems() action in the InboundQueueController controller is not populating the view with data. The action must populate the view with data by calling the GetExternalOrders() method in the ExternalQueueService service using the ChannelFactory class. You need to modify the action to populate the view with data. What should you do? (To answer, drag the appropriate code segments to the correct location or locations in the answer area. Each code segment 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.)



Answer:



QUESTION 57

The DeleteExternalOrder() method in the ExternalQueueService service is not throwing a FaultException exception as defined by the FaultContractAttribute attribute in the IExternalQueueService.cs file. You need to throw the FaultException exception. Which code segments can you insert at line EQ45 to achieve this goal? (Each correct answer presents a complete solution. Chose all that apply)

A.
```
throw new FaultException<OrderNotFoundException>(ex.ExceptionMessage);
```

B.
```
throw new FaultException<OrderNotFoundException>(ex, new
 FaultReason("Order not found."));
```

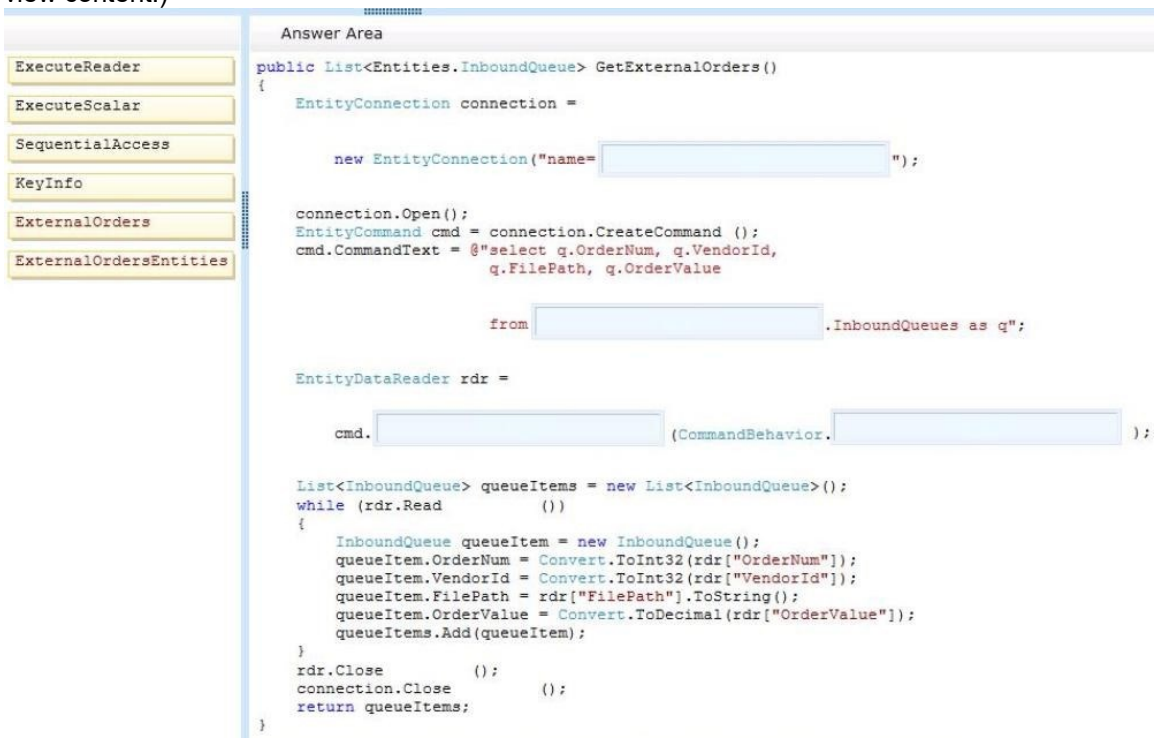
C.
```
throw new FaultException<OrderNotFoundException>(ex);
```

D.
```
throw new FaultException
 (new OrderNotFoundException(new Exception(ex.ExceptionMessage)), "Order not
found.");
```

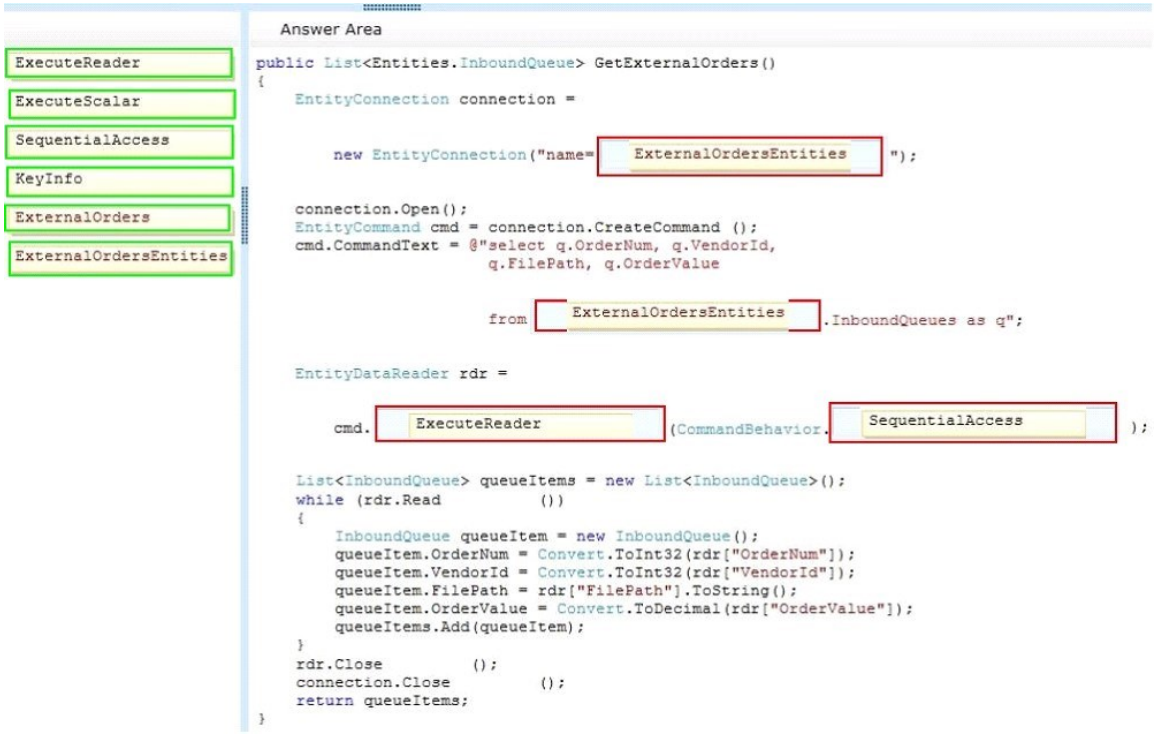
A. Option A
B. Option B
C. Option C
D. Option D

Answer: BC

QUESTION 58
Drag and Drop Question
The GetExternalOrders() method must use members of the EntityClient namespace to query the database for all records in the InboundQueue entity. You need to modify the GetExternalOrders() method to return the correct data. What should you do? (To answer, drag the appropriate code segments to the correct location or locations in the answer area. Each code segment 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.)



Answer:



QUESTION 59
You need to regenerate the service proxies to include task-based asynchronous method signatures. Which command should you use?

A. `aspnet_regiis.exe /t:code http://localhost:62965/UploadCallbackService.svc`
B. `svcutil.exe /t:code http://localhost:62965/UploadCallbackService.svc`
C. `aspnet_compiler.exe /t:code http://localhost:62965/UploadCallbackService.svc`
D. `aspnet_regiis.exe /t:code http://localhost:62965/UploadService.svc`
E. `svcutil.exe /t:code http://localhost:62965/UploadService.svc`

Answer: B
Explanation:
http://msdn.microsoft.com/en-us/library/aa347733.aspx

QUESTION 60
The DeleteExternalOrder() method in the ExternalQueueService service is not throwing a FaultException exception as defined by the FaultContractAttribute attribute in the IExternatQueueService.cs file. You need to throw the FaultException exception. Which code segment can you insert at line EQ45 to achieve this goal? (Each correct answer presents a complete solution. Chose all that apply.)

A.
```
string queryString = @"SELECT q.OrderNum, q.VendorId, q.FilePath, q.OrderValue
    FROM ExternalOrdersEntities.InboundQueues AS q WHERE q.OrderNum = @orderNum";
```

B.
```
string queryString = @"SELECT * FROM ExternalOrdersEntities.InboundQueues
    WHERE OrderNum = @orderNum";
```

C.
```
string queryString = @"SELECT VALUE q FROM ExternalOrdersEntities.InboundQueues AS q
    WHERE q.OrderNum = @orderNum";
```

D.
```
string queryString = @"SELECT VALUE FROM ExternalOrdersEntities.InboundQueues
    WHERE OrderNum = @orderNum";
```

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

Answer: C