

Practical Exam (205)

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'DESKTOP-BBKQSPH (SQL Server 15.0.2080.9 - sa)'. The central query window shows a SQL query executed successfully. The Results pane at the bottom displays the output of the query, which is a table with 5 columns: E_Id, E_Title, E_Type_Event, E_Estimated_Cost, and E_Actual_Cost. The query results show two rows of data.

```
Script for SelectTopNRows command from SSMS *****
SELECT TOP (1000) [E_Id]
, [E_Title]
, [E_Type_Event]
, [E_Estimated_Cost]
, [E_Actual_Cost]
, [E_Date]
FROM [Events_DB].[dbo].[Event_Master]
```

E_Id	E_Title	E_Type_Event	E_Estimated_Cost	E_Actual_Cost	E_Date
1	Naurati	Festive	10,000	12,000	7-10-2021
2	Music Day	cult	5000	6000	4-10-2021

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'DESKTOP-BBKQSPH (SQL Server 15.0.2080.9 - sa)'. The central query window shows the table structure of the 'Event_Master' table. The Column Properties pane at the bottom shows the properties for the 'E_Id' column, which is an identity column.

Column Name	Data Type	Allow Nulls
E_Id	int	<input type="checkbox"/>
E_Title	varchar(50)	<input checked="" type="checkbox"/>
E_Type_Event	varchar(50)	<input checked="" type="checkbox"/>
E_Estimated_Cost	varchar(50)	<input checked="" type="checkbox"/>
E_Actual_Cost	varchar(50)	<input checked="" type="checkbox"/>
E_Date	varchar(50)	<input checked="" type="checkbox"/>

Column Properties

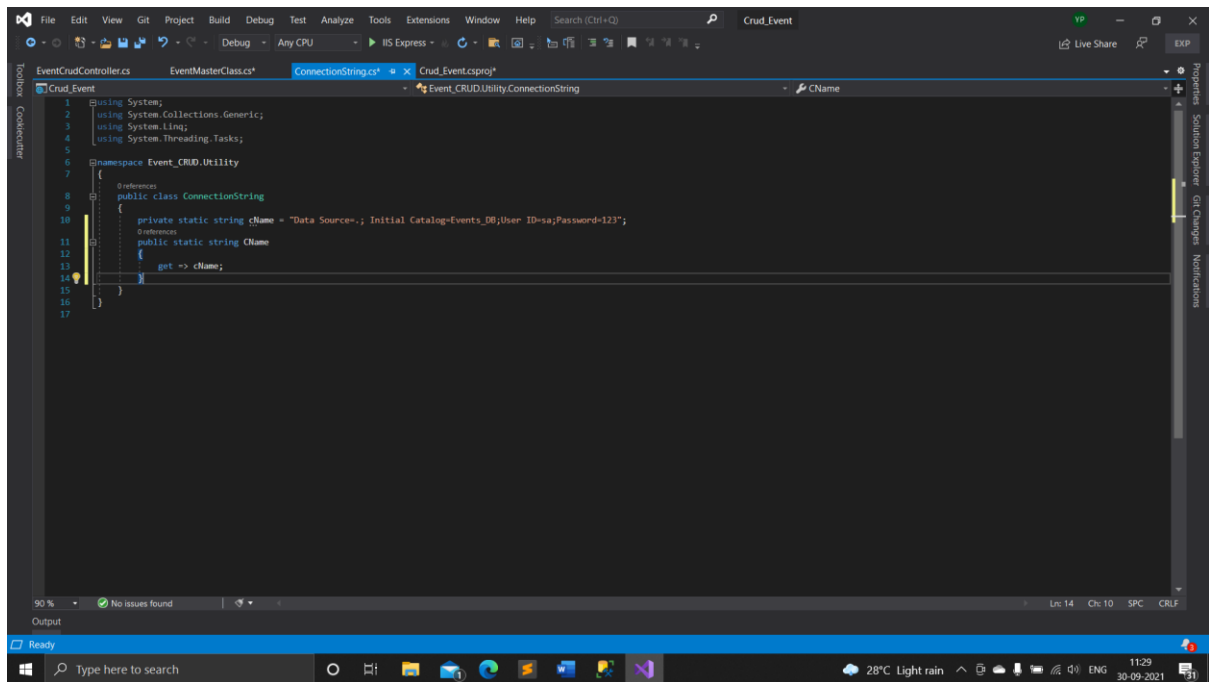
Property	Value
DTS-published	No
Full-text Specification	No
Has Non-SQL Server Subscriber	No
Identity Specification	Yes
(Is Identity)	Yes
Identity Increment	1
Identity Seed	1

This screenshot shows the Visual Studio IDE with the `EventCrudController.cs` file open. The code defines a `Controller` class with several actions: `Index()`, `Details(int id)`, `Create()`, and `Create(IFormCollection collection)`. The `Create(IFormCollection collection)` method includes a `try` block that calls `RedirectToAction` with `nameof(Index)`. The interface includes a menu bar, a toolbar, and a sidebar with the Solution Explorer and Properties windows. The status bar at the bottom shows 'Ready' and system information.

```
1 using Microsoft.AspNetCore.Http;
2 using Microsoft.AspNetCore.Mvc;
3 using System;
4 using System.Collections.Generic;
5 using System.Linq;
6 using System.Threading.Tasks;
7
8 namespace Crud_Event.Controllers
9 {
10     [Route("")]
11     public class EventCrudController : Controller
12     {
13         // GET: EventCrudController
14         public ActionResult Index()
15         {
16             return View();
17         }
18
19         // GET: EventCrudController/Details/5
20         public ActionResult Details(int id)
21         {
22             return View();
23         }
24
25         // GET: EventCrudController/Create
26         public ActionResult Create()
27         {
28             return View();
29         }
30
31         // POST: EventCrudController/Create
32         [HttpPost]
33         [ValidateAntiForgeryToken]
34         public ActionResult Create(IFormCollection collection)
35         {
36             try
37             {
38                 return RedirectToAction(nameof(Index));
39             }
40             catch
41             {
37             }
42         }
43     }
44 }
```

This screenshot shows the Visual Studio IDE with the `EventMasterClass.cs` file open. The code defines a `EventMasterClass` class with several properties: `E_Id`, `E_Type_Event`, `E_Estimated_Cost`, `E_Actual_Cost`, and `E_Date`. Each property has a `get; set;` pair. The interface includes a menu bar, a toolbar, and a sidebar with the Solution Explorer and Properties windows. The status bar at the bottom shows 'Ready' and system information.

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Threading.Tasks;
5
6 namespace Crud_Event.Models
7 {
8     [Route("")]
9     public class EventMasterClass
10     {
11         // GET: EventMasterClass
12         public int E_Id { get; set; }
13
14         // GET: EventMasterClass
15         public string E_Type_Event { get; set; }
16
17         // GET: EventMasterClass
18         public string E_Estimated_Cost { get; set; }
19
20         // GET: EventMasterClass
21         public string E_Actual_Cost { get; set; }
22
23         // GET: EventMasterClass
24         public string E_Date { get; set; }
25     }
26 }
```



EventCrudController.cs

```
using Microsoft.AspNetCore.Http;
using Microsoft.AspNetCore.Mvc;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Crud_Event.Models;

namespace Crud_Event.Controllers
{
    public class EventCrudController : Controller
    {
        EventDataAccessLayer eventDataAccessLayer = new EventDataAccessLayer();
        // GET: EventCrudController
        public ActionResult Index()
        {
            EventDataAccessLayer eventDataAccessLayer = new EventDataAccessLayer();

            IEnumerable<EventMasterClass> ev = eventDataAccessLayer.GetAllEvent();
            return View(ev);
        }

        // GET: EventCrudController/Details/5
        public ActionResult Details(int id)
        {
            return View();
        }

        // GET: EventCrudController/Create
```

```

public ActionResult Create()
{
    return View();
}

// POST: EventCrudController/Create
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(IFormCollection collection)
{
    try
    {
        return RedirectToAction(nameof(Index));
    }
    catch
    {
        return View();
    }
}

// GET: EventCrudController/Edit/5
public ActionResult Edit(int id)
{
    return View();
}

// POST: EventCrudController/Edit/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Edit(int id, IFormCollection collection)
{
    try
    {
        return RedirectToAction(nameof(Index));
    }
    catch
    {
        return View();
    }
}

// GET: EventCrudController/Delete/5
public ActionResult Delete(int id)
{
    return View();
}

// POST: EventCrudController/Delete/5
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Delete(int id, IFormCollection collection)
{
    try
    {
        return RedirectToAction(nameof(Index));
    }
    catch
    {
        return View();
    }
}
}

```

```
}
```

EventDataAccessLayer.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Threading.Tasks;
using Event_CRUD.Utility;

namespace Crud_Event.Models
{
    public class EventDataAccessLayer
    {
        string connectionString = ConnectionString.CName;
        public IEnumerable<EventMasterClass> GetAllEvent()
        {
            List<EventMasterClass> lstStudent = new List<EventMasterClass>();
            using (SqlConnection con = new SqlConnection(connectionString))
            {
                SqlCommand cmd = new SqlCommand("", con);
                cmd.CommandType = CommandType.StoredProcedure;
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();

                while (rdr.Read())
                {
                    EventMasterClass ev = new EventMasterClass();
                    ev.E_Id = Convert.ToInt32(rdr["E_Id"]);
                    ev.E_Title = rdr["E_Title"].ToString();
                    ev.E_Type_Event = rdr["E_Type_Event"].ToString();
                    ev.E_Estimated_Cost = rdr["E_Estimated_Cost"].ToString();
                    ev.E_Actual_Cost = rdr["E_Actual_Cost"].ToString();
                    ev.E_Date = rdr["E_Date"].ToString();

                    lstStudent.Add(ev);
                }
                con.Close();
            }
            return lstStudent;
        }
        public void AddEvent(EventMasterClass ev)
        {
            using (SqlConnection con = new SqlConnection(connectionString))
            {
                SqlCommand cmd = new SqlCommand("", con);
                cmd.CommandType = CommandType.StoredProcedure;

                cmd.Parameters.AddWithValue("@E_Id", ev.E_Title);
                cmd.Parameters.AddWithValue("@E_Type_Event", ev.E_Type_Event);
                cmd.Parameters.AddWithValue("@E_Estimated_Cost", ev.E_Estimated_Cost);
                cmd.Parameters.AddWithValue("@E_Actual_Cost", ev.E_Actual_Cost);
                cmd.Parameters.AddWithValue("@E_Date", ev.E_Date);
                con.Open();
                cmd.ExecuteNonQuery();
                con.Close();
            }
        }
        public void UpdateEvent(EventMasterClass ev)
```

```

{
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        SqlCommand cmd = new SqlCommand("", con);
        cmd.CommandType = CommandType.StoredProcedure;

        cmd.Parameters.AddWithValue("@E_Id", ev.E_Title);
        cmd.Parameters.AddWithValue("@E_Type_Event", ev.E_Type_Event);
        cmd.Parameters.AddWithValue("@E_Estimated_Cost", ev.E_Estimated_Cost);
        cmd.Parameters.AddWithValue("@E_Actual_Cost", ev.E_Actual_Cost);
        cmd.Parameters.AddWithValue("@E_Date", ev.E_Date);
        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
    }
}

public EventMasterClass GetEvent(int? id)
{
    EventMasterClass ev = new EventMasterClass();

    using (SqlConnection con = new SqlConnection(connectionString))
    {
        string sqlQuery = "SELECT * FROM Event_Master WHERE E_Id= " + id;
        SqlCommand cmd = new SqlCommand(sqlQuery, con);
        con.Open();
        SqlDataReader rdr = cmd.ExecuteReader();

        while (rdr.Read())
        {
            ev.E_Id = Convert.ToInt32(rdr["E_Id"]);
            ev.E_Title = rdr["E_Title"].ToString();
            ev.E_Type_Event = rdr["E_Type_Event"].ToString();
            ev.E_Estimated_Cost = rdr["E_Estimated_Cost"].ToString();
            ev.E_Actual_Cost = rdr["E_Actual_Cost"].ToString();
            ev.E_Date = rdr["E_Date"].ToString();
        }
    }
    return ev;
}

public void DeleteEvent(int? id)
{
    using (SqlConnection con = new SqlConnection(connectionString))
    {
        SqlCommand cmd = new SqlCommand("", con);
        cmd.CommandType = CommandType.StoredProcedure;
        cmd.Parameters.AddWithValue("@E_Id", id);
        con.Open();
        cmd.ExecuteNonQuery();
        con.Close();
    }
}
}
}

```

EventMasterClass.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

```

```

namespace Crud_Event.Models
{
    public class EventMasterClass
    {
        public int E_Id { get; set; }
        public string E_Title { get; set; }
        public string E_Type_Event { get; set; }
        public string E_Estimated_Cost { get; set; }
        public string E_Actual_Cost { get; set; }
        public string E_Date { get; set; }
    }
}

```

ConnectionString.cs

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Event_CRUD.Utility
{
    public class ConnectionString
    {
        private static string cName = "Data Source=.; Initial Catalog=Events_DB;User
ID=sa;Password=123";
        public static string CName
        {
            get => cName;
        }
    }
}

```

Index.chnml

```

@model IEnumerable<Crud_Event.Models.EventMasterClass>

@{
    ViewData["Title"] = "Index";
}

<h1>Index</h1>

<p>
    <a asp-action="Create">Create New</a>
</p>
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.E_Id)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.E_Title)
            </th>

```

```

        </th>
        <th>
            @Html.DisplayNameFor(model => model.E_Type_Event)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.E_Estimated_Cost)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.E_Actual_Cost)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.E_Date)
        </th>
        <th></th>
    </tr>
</thead>
<tbody>
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.E_Id)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.E_Title)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.E_Type_Event)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.E_Estimated_Cost)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.E_Actual_Cost)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.E_Date)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |
            @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ })
        </td>
    </tr>
}
</tbody>
</table>

```