Экспортирование выходных данных в HTML-документ

.NET поставляется с большим количеством встроенных классов для рендеринга HTML-контента. Эти классы еще существуют как часть сборки System.Web.dll.

Для оперирования таблицами в .NET предусмотрен класс DataTable из System.Data.dll.

Первым делом реализуем метод для преобразования объекта класса DataTable в готовый HTML-документ:

1. **using** System;
2. **using** System.Data;
3. **using** System.IO;
4. **using** System.Text;
5. **using** System.Web.UI;
6. **using** System.Web.UI.HtmlControls;
8. **namespace** DataTableExport
9. {
10. **public** **static** **class** Extensions
11. {
12. **public** **static** String GetHtml(**this** DataTable dataTable)
13. {
14. StringBuilder sbControlHtml = **new** StringBuilder();
15. **using** (StringWriter stringWriter = **new** StringWriter())
16. {
17. **using** (HtmlTextWriter htmlWriter = **new** HtmlTextWriter(stringWriter))
18. {
19. **using** (var htmlTable = **new** HtmlTable())
20. {
21. // Add table header row
22. **using** (var headerRow = **new** HtmlTableRow())
23. {
24. **foreach** (DataColumn dataColumn **in** dataTable.Columns)
25. {
26. **using** (var htmlColumn = **new** HtmlTableCell())
27. {
28. htmlColumn.InnerText = dataColumn.ColumnName;
29. headerRow.Cells.Add(htmlColumn);
30. }
31. }
32. htmlTable.Rows.Add(headerRow);
33. }
35. // Add data rows
36. **foreach** (DataRow row **in** dataTable.Rows)
37. {
38. **using** (var htmlRow = **new** HtmlTableRow())
39. {
40. **foreach** (DataColumn column **in** dataTable.Columns)
41. {
42. **using** (var htmlColumn = **new** HtmlTableCell())
43. {
44. htmlColumn.InnerText = row[column].ToString();
45. htmlRow.Cells.Add(htmlColumn);
46. }
47. }
48. htmlTable.Rows.Add(htmlRow);
49. }
50. }
51. htmlTable.RenderControl(htmlWriter);
52. sbControlHtml.Append(stringWriter.ToString());
53. }
54. }
55. }
56. **return** sbControlHtml.ToString();
57. }
58. }
59. }

Теперь, чтобы сделать тест, нам сначала нужно добавить использование пространства имен DataTableExport в код нашего приложения, чтобы сделать метод расширения доступным для экземпляра класса DataTable. Следующим шагом является заполнение образца экземпляра DataTable образцами данных.

1. **using** System;
2. **using** System.Data;
3. **using** DataTableExport;
5. **namespace** SampleApplication
6. {
7. **class** Program
8. {
9. **static** **void** Main(**string**[] args)
10. {
11. var dataTable = **new** DataTable();
12. dataTable.Columns.Add("id");
13. dataTable.Columns.Add("Name");
14. dataTable.Columns.Add("Email");
16. var row = dataTable.NewRow();
17. row["id"] = Guid.NewGuid().ToString();
18. row["Name"] = "Bruce Wayne";
19. row["Email"] = "batman@superheroes.com";
20. dataTable.Rows.Add(row);
22. row = dataTable.NewRow();
23. row["id"] = Guid.NewGuid().ToString();
24. row["Name"] = "Clark Kent";
25. row["Email"] = "superman@superheroes.com";
26. dataTable.Rows.Add(row);
28. row = dataTable.NewRow();
29. row["id"] = Guid.NewGuid().ToString();
30. row["Name"] = "Peter Parker";
31. row["Email"] = "spiderman@superheroes.com";
32. dataTable.Rows.Add(row);
34. dataTable.AcceptChanges();
36. var html = dataTable.GetHtml();
37. }
38. }
39. }

Результатом вызова метода расширения GetHtml будет красиво отформатированная HTML-строка таблицы с данными из экземпляра класса DataTable.

1. **<table>**
2. **<tr>**
3. **<td>**id**</td>**
4. **<td>**Name**</td>**
5. **<td>**Email**</td>**
6. **</tr>**
7. **<tr>**
8. **<td>**b5beceae-88b9-48cc-8813-adfade92ad45**</td>**
9. **<td>**Bruce Wayne**</td>**
10. **<td>**batman@superheroes.com**</td>**
11. **</tr>**
12. **<tr>**
13. **<td>**1e7a12f6-2c01-4991-82d7-cbda23dd7a6a**</td>**
14. **<td>**Clark Kent**</td>**
15. **<td>**superman@superheroes.com**</td>**
16. **</tr>**
17. **<tr>**
18. **<td>**74875789-853f-481e-8e7d-94b95e662d10**</td>**
19. **<td>**Peter Parker**</td>**
20. **<td>**spiderman@superheroes.com**</td>**
21. **</tr>**
22. **</table>**

Это базовая реализация, которая будет генерировать только строку HTML таблицы без какого-либо стиля CSS, но классы в System.Web.UI.HtmlControls предоставляют методы и свойства для применения стилей CSS, а также стандартных и пользовательских атрибутов ко всем классам представления тегов HTML.