**AJAX - Asynchronous JavaScript and XML**  
Technika tworzenia aplikacji internetowych, w których interakcja użytkownika z serwerem odbywa się bez przeładowywania całego dokumentu, w sposób asynchroniczny. Ma to umożliwiać bardziej dynamiczną interakcję z użytkownikiem niż w tradycyjnym modelu, w którym każde żądanie nowych danych wiąże się z przesłaniem całej strony HTML

Dzięki Ajax będziemy usuwać pozycje faktury i faktury bez odświeżania strony.

Wywołujemy akcję w kontrolerze bez odświeżania strony

Do deleteInvoice() musimy przekazać Id faktury, którą zamierzamy usunąć.

|  |
| --- |
| <td>  <button  class="btn btn-danger btn-sm"  onclick="deleteInvoice('@invoice.Id', this)">  Usuń Fakturę  </button>  </td> |
| @section scripts  {  <script type="text/javascript">  function deleteInvoice(id) {  if (!confirm("Czy na pewno chcesz usunąć fakturę ?"))  return;  }  </script>  } |

Po stronie html’a jako parametr w cudzysłowie przełączamy się na C# ca pomocą znacznika@  
i podajemy invoiceId.

W javascripcie w definicji funkcji nie oczekujemy konkretnego typu, tylko oczekujemy, że będzie przekazane id.  
Następnie za pomocą AJAX usuniemy z bazy danych fakturę o podanym Id, dzięki wywołaniu Akcji HttpPost Delete() z kontrolera Home.

|  |
| --- |
| @section scripts  {  <script type="text/javascript">  // --------------------------------------------  function deleteInvoice(id) {  if (!confirm("Czy na pewno chcesz usunąć fakturę ?"))  return;  $.ajax({  type: 'POST',  url: "@Url.Action("DeleteInvoice","Home")",  data: {  invoiceId: id  },  success: function (data) {  if (data.Success) {  // usuwanie z tabeli w html'u  }  else {  alert(data.Message);  }  },  error: function (data) {  alert(data.Message);  },  dataType: 'json'    });  // --------------------------------------------  }  </script> |
| [HttpPost]  public ActionResult DeleteInvoice(int invoiceId)  {  try  {  var userId = User.Identity.GetUserId();  \_invoiceRepository.Delete(invoiceId, userId);  }  catch (Exception exception)  {  // TODO: logowanie do pliku niepowodzenie usunięcia faktury  return Json(new { Success = false, Message = exception.Message });  }  return Json(new { Success = true });  } |

W sekcji type: Przekazujemy typ akcji POST, następnie podajemy w sekcji url:  
za pomocą c# i helpera Url.Action podajemy nazwę akcji i kontrolera.  
Czyli chcemy aby została wywołana Akcja DeleteInvoice() w kontrolerze Home.   
Następnie w sekcji data: przekazujemy nazwę parametru. Jak przejdziemy do Akcji w kontrolerze,  
to Akcja DeleteInvoice oczekuje parametru o nazwie invoiceId, tak więc musimy go nazwać.  
Następnie mówimy co ma się wydarzyć , jak wszystko zostanie zakończone sukcesem  
zostanie wywołana funkcja, która jako parametr dostanie data – czyli JSON’a zwracanego z kontrolera. Jeżeli dostanie informacje, że wszystko ok, to usuniemy pozycję z tabeli w HTML.  
W przeciwnym wypadku zostanie wyświetlony komunikat jaki wygenerujemy w kontrolerze

W zdarzeniu onClick() oprócz id faktury, przekazujemy parametr javasrcipt this, dzięki czemu przekazujemy cały obiekt przycisku i będziemy mogli się dobrać do jego właściwości.

|  |
| --- |
| <td>  <button  class="btn btn-danger btn-sm"  onclick="deleteInvoice('@invoice.Id', this)">  Usuń Fakturę  </button>  </td> |

btn.parentNode.parentNode – pierwszy node to <td>, drugi to <tr>  
potem przechodzimy jeszcze jeden poziom wyżej czyli do <body> i usuwamy potomka

|  |
| --- |
| if (data.Success) {  // usuwanie z tabeli w html'u  var row = btn.parentNode.parentNode;  row.parentNode.removeChild(row);  } |