Inhalt

[Dictionary 1](#_Toc5864255)

[Code Snippets 2](#_Toc5864256)

[Server 2](#_Toc5864257)

[GET 2](#_Toc5864258)

[POST 2](#_Toc5864259)

[Logic - Businesslogik zwischen Server und Datenbank 2](#_Toc5864260)

[Validierung 2](#_Toc5864261)

[Novum interne Datenstruktur auf OrderSolutions umwandeln 2](#_Toc5864262)

[Database 3](#_Toc5864263)

[Caché Klassenmethodenaufruf mit Retourwert 3](#_Toc5864264)

[Caché Klassenmethodenaufruf ohne Retourwert 3](#_Toc5864265)

[Caché SQL-Aufruf 3](#_Toc5864266)

[Data 4](#_Toc5864267)

[Table 4](#_Toc5864268)

# Dictionary

|  |  |  |
| --- | --- | --- |
| Novum Code | Novacom Code, Bedeutung | Werte |
| ClientId | FA, Firma | 1001 |
| PosId | Kassa | RK2 |
| ServiceAreaId | VKO, Verkaufsort | RES |
| ServiceAreaName | Verkaufsortbezeichnung | Stube |
| PriceLevelId | Preisebene, VPE | 1 |
| WaiterId | Key | 1 |
| WaiterName | Kellnername | Markus Kogler |
| Table.Id | Tisch | 1010.1 |
| Table.Name | Tischname | 10.1 |
| Table.Comment | Tischkommentar | Stammtisch |
| Table.Amount | Tischbetrag | 12,23 |
| Article.Id | Artikelnummer | 10005 |
| Article.Name | Artikelbezeichnung | Frankfurter |
| Order.ArticleId | Artikelnummer | 10005 |
| Order.Name | Bestellungstext | Schnitzel in Herzform |
| Order.Quantity | Menge | 2 |
| Order.UnitPrice | Einzelpreis | 7,89 |
| Order.TotalPrice | Gesamtpreis | 15,78 |

# C# Convention

## Programmiersprache ist Englisch

Alle Klassen-, Methoden-, Variablennamen sind ausnahmslos englisch.

## Objektnamenschreibweise

|  |  |  |
| --- | --- | --- |
| **Object** | **Casing** | **Beispiel** |
| Klassenname | PascalCase | class **Table**{} |
| Methodenname | PascalCase | bool **HasValue**() {} |
| Methodenparameter | camelCase | void SetValue(int **value**) {} |
| lokale Variablen | camelCase | List<string> **openTableIds**; |
| Propertynamen | PascalCase | string **Name** { get; set; } |
| Enumnamen | PascalCase | private enum **Index** { Id = 0, Name = 1} |
| Konstanten | PascalCase | public const string **MyConst** = "MyConst"; |

## Objektnamensvergebung

|  |  |  |
| --- | --- | --- |
| **Object** | **Casing** | **Beispiel** |
| Klassenname | Nomen Singular | class **Table**{} |
| Interface | Nomen Singular  mit Prefix **I** | interface **ITable** {} |
| Enum | Nomen Singular  kein Pre-Suffix Enum oder Flag | //Correct  enum **Color** { Red, Green}  //Avoid  enum **ColorEnum** { Red, Green } |
| boolesche Felder | Frage muss lesbar sein und mit ja/nein beantwortbar sein | bool **valueExists**;  if (**valueExists**) {} |
| boolesche Methoden | Frage muss lesbar sein und mit ja/nein beantwortbar sein | bool **RequestNeedsAuthorization**(HttpRequest request) {}  if (**RequestNeedsAuthorization**) {} |
| Exceptions | Suffix Exception | public class **ReaderException** : System.Exception {} |
| Handler | Suffix Handler  zwei Parameter sender und e | public void **ReaderEventHandler**(object sender, ReadEventArgs e)  {} |

## Objektreiheinfolge

Die Objektreihenfolge ist nach dem ausgerichtet, dass Objekte weiter unten Objekte von oben verwenden. Ausnahme sind hier die privaten Methoden.

|  |  |  |
| --- | --- | --- |
| Reihenfolge |  |  |
| 1 | Enums |  |
| 2 | Konstanten |  |
| 3 | Variablen |  |
| 4 | Properties |  |
| 5 | Konstruktor |  |
| 6 | **public** Methoden |  |
| 7 | **private** Methoden |  |

## Abkürzung vermeiden

Ausnahmen bilden bekannte Abkürzungen wie Id, Ftp, Uri, etc. Diese werden in CamelCase geschrieben.  
Eine weiter Ausnahme sind EventHandler und Exceptions

|  |
| --- |
| // Correct  UserGroup userGroup;  Assignment employeeAssignment;  // Avoid  UserGroup usrGrp;  Assignment empAssignment**;**  FTPHelper FTPHelper**;**  // Exceptions  CustomerId customerId;  **XmlDocument** xmlDocument;  FtpHelper ftpHelper;  UriPart uriPart;  public delegate void ReadBarcodeEventHandler(object sender, ReadBarcodeEventArgs e);  catch (**Exception** e) {} |

## Keine Unterstrich verwenden

Ausnahmen bilden **private** Daten.

|  |
| --- |
| // Correct  public DateTime clientAppointment;  public TimeSpan timeLeft;  // Avoid  public DateTime client\_Appointment;  public TimeSpan time\_Left;  // Exception (Class field)  private DateTime \_registrationDate; |

## Var verwenden bei lokalen Zuweisungen

Ausnahmen sind primitive Typen wie int, string, double, etc.

|  |
| --- |
| var stream = File.Create(path);  var customers = new Dictionary();  // Exceptions  int index = 100;  string timeSheet;  bool isCompleted; |

## geschwungene Klammern

Eine geschwungene Klammer bekommt eine eigene Zeile!

|  |
| --- |
| // Correct  class Program  {  static void Main(string[] args)  {  //...  }  } |

# Code Snippets

## Server

### GET

|  |
| --- |
| [HttpGet]  [Route("/api/v2/data/ServiceAreas")]  public IActionResult GetServiceAreas()  {  var serviceAreas = Novum.Logic.Os.Data.GetServiceAreas();  return new OkObjectResult(serviceAreas);  } |

### POST

|  |
| --- |
| [HttpPost]  [Route("/api/v2/actions/Auth/Login")]  public IActionResult AuthLogin([FromBody][Required]LoginUser loginUser)  {  var session = Data.Sessions.GetSession(Request);  try  {  session.WaiterId = loginUser.Id;  Logic.Os.Registration.Login(session, loginUser);  Data.Sessions.SetSession(session);  //204 - No Content  return new NoContentResult();  }  catch (Exception ex)  {  session.WaiterId = "";  Data.Sessions.SetSession(session);  var osError = new OsError();  osError.ErrorMsg = ex.Message;  //401 - Unauthorized  return new UnauthorizedObjectResult(osError);  }  } |

## Logic - Businesslogik zwischen Server und Datenbank

### Validierung

|  |
| --- |
| public static void Login(Session session, LoginUser loginUser)  {  bool validWaiter = DB.Api.Waiter.ValidWaiter(session, loginUser.Password);  if (!validWaiter)  throw new Exception(string.Format("user {0} not valid", loginUser.Id));  DB.Api.Waiter.Login(session);  } |

### Novum interne Datenstruktur auf OrderSolutions umwandeln

|  |
| --- |
| public static List<Novum.Data.Os.Category> GetCategories(string menuId)  {  var osCategories = new List< Novum.Data.Os.Category>();  var mainMenus = DB.Api.Menu.GetMainMenu(menuId);  var handledMenuIds = new List<string>();  foreach (Novum.Data.Menu menu in mainMenus.Values)  {  var category = new Novum.Data.Os.Category();  category.Name = menu.Name;  var contentEntries = GetCategoryContent(menu.Id, ref handledMenuIds);  category.Content = contentEntries;  categories.Add(category);  }  return categories;  } |

## Database

### Caché Klassenmethodenaufruf mit Retourwert

|  |
| --- |
| public Dictionary<string, Novum.Data.Table> GetTables()  {  var tables = new Dictionary<string, Novum.Data.Table>();  var dbString = Interaction.CallClassMethod("cmNT.Tisch", "GetTischListeAll", Data.Department);  var tablesString = new Novum.Data.Utils.DataString(dbString);  var tablesArray = tablesString.SplitByDoublePipes();  foreach (string tableString in tablesArray)  {  if (string.IsNullOrEmpty(tableString))  continue;  var table = new Novum.Data.Table(tableString);  tables.Add(table.Id, table);  }  return tables;  } |
|  |

### Caché Klassenmethodenaufruf ohne Retourwert

|  |
| --- |
| public void Logout(Session session)  {  var posId = DB.Api.Pos.GetPosId(session.SerialNumber);  Interaction.CallVoidClassMethod("cmNT.Kellner", "Kellnerlogout", session.Department, posId, session.WaiterId);  } |

### Caché SQL-Aufruf

|  |
| --- |
| public Dictionary<string, Novum.Data.ServiceArea> GetServiceAreas()  {  var serviceAreas = new Dictionary<string, Novum.Data.ServiceArea>();  var sql = new StringBuilder();  sql.Append(" SELECT VKO, bez, vkebene ");  sql.Append(" FROM WW.VKO ");  sql.Append(" WHERE FA = ").Append(Data.ClientId);  sql.Append(" AND passiv > ").Append(Interaction.SqlToday);  var dataTable = Interaction.GetDataTable(sql.ToString());  foreach (DataRow dataRow in dataTable.Rows)  {  var serviceArea = new Novum.Data.ServiceArea();  serviceArea.Id = DataObject.GetString(dataRow, "VKO");  serviceArea.Name = DataObject.GetString(dataRow, "bez");  serviceArea.PriceLevel = DataObject.GetString(dataRow, "vkebene");  serviceAreas.Add(serviceArea.Id, serviceArea);  }  return serviceAreas;  } |
| public bool ValidWaiter(Session session, string code)  {  var waiters = new Dictionary<string, Novum.Data.Waiter>();  var sql = new StringBuilder();  sql.Append(" SELECT PNR, code, name");  sql.Append(" FROM NT.Pers ");  sql.Append(" WHERE FA = ").Append(session.ClientId);  sql.Append(" AND PNR = ").Append(Interaction.SqlQuote(session.WaiterId));  sql.Append(" AND code = ").Append(Interaction.SqlQuote(code));  sql.Append(" AND passiv > ").Append(Interaction.SqlToday);  var dataTable = Interaction.GetDataTable(sql.ToString());  if (dataTable.Rows.Count == 1)  return true;  else  return false;  } |

## Data

### Table

|  |
| --- |
| namespace Novum.Data  {  public class Table  {  private enum Index  {  Id = 0,  Name = 1,  Amount = 2,  Comment = 3,  AmmountWithPrebookings = 5,  Status = 6,  WaiterId = 7,  WaiterName = 8,  Opened = 9,  Updated = 10,  Room = 11,  Guests = 13,  LeftTableId = 21,  RightTableId = 22,  AssignmentType = 23  }  #region Constructor  public Table(string dbString)  {  if (string.IsNullOrEmpty(dbString))  return;  var dataString = new Utils.DataString(dbString);  var dataList = new Utils.DataList(dataString.SplitByChar96());  this.Id = dataList.GetString((int)Index.Id);  this.Name = dataList.GetString((int)Index.Name);  this.Amount = dataList.GetDecimal((int)Index.Amount);  this.Comment = dataList.GetString((int)Index.Comment);  this.WaiterId = dataList.GetString((int)Index.WaiterId);  this.WaiterName = dataList.GetString((int)Index.WaiterName);  this.Opend = dataList.GetDateTime((int)Index.Opened);  this.Updated = dataList.GetDateTime((int)Index.Updated);  this.Room = dataList.GetString((int)Index.Room);  this.Guests = dataList.GetUInt((int)Index.Guests);  this.LeftTableId = dataList.GetString((int)Index.LeftTableId);  this.RightTableId = dataList.GetString((int)Index.RightTableId);  }  #endregion  #region Properties    public string Id { get; set; }  public string Name { get; set; }  public decimal Amount { get; set; }  public string WaiterId { get; set; }  public string WaiterName { get; set; }  public string Comment { get; set; }  public DateTime Opend { get; set; }  public DateTime Updated { get; set; }  public string Room { get; set; }  public uint Guests { get; set; }  public string LeftTableId { get; set; }  public string RightTableId { get; set; }  }  } |