|  |  |  |
| --- | --- | --- |
| Amaryllis Framework | September 22  2012 | |
| This document represents a manual that describes the details for developing a software product over the Amaryllis platform as well as the technical details behind it. Also there will be some code coverage and coding guide to develop extra features for this platform. | | Manganese 55 |

[1. Introduction 19](#_Toc345097833)

[1.1. Description for the Chosen Sample 19](#_Toc345097834)

[1.2. Problem Details 19](#_Toc345097835)

[1.3. Motivation and Conclusion 19](#_Toc345097836)

[2. Data Layer 20](#_Toc345097837)

[2.1. Data Tables Grouped by Module 20](#_Toc345097838)

[2.1.1. User Management Module 20](#_Toc345097839)

[2.1.2. Documentation Module 21](#_Toc345097840)

[2.1.3. Class Module 24](#_Toc345097841)

[2.1.4. Constants module 27](#_Toc345097842)

[2.1.5. Controls module 27](#_Toc345097843)

[2.1.6. Desktop Module 28](#_Toc345097844)

[2.1.7. Events Module 29](#_Toc345097845)

[2.1.8. File Module 30](#_Toc345097846)

[2.1.9. Custom Forms Module 31](#_Toc345097847)

[2.1.10. Virtual Drive Module 32](#_Toc345097848)

[2.1.11. Translation Module 33](#_Toc345097849)

[2.1.12. Mappings Module 34](#_Toc345097850)

[2.1.13. Menu Module 35](#_Toc345097851)

[2.1.14. Notifications Module 37](#_Toc345097852)

[2.1.15. Objects Module 38](#_Toc345097853)

[2.1.16. Page Module 40](#_Toc345097854)

[2.1.17. Reference Guide Module 40](#_Toc345097855)

[2.1.18. Reporting Module 40](#_Toc345097856)

[2.1.19. Security Module 42](#_Toc345097857)

[2.1.20. Settings Module 43](#_Toc345097858)

[2.1.21. DataBase Module 44](#_Toc345097859)

[2.2. User Defined Functions and Stored Procedures Grouped by Module 44](#_Toc345097860)

[2.2.1. Documentation Module 44](#_Toc345097861)

[2.2.2. Cache Module 51](#_Toc345097862)

[2.2.3. Class Module 52](#_Toc345097863)

[2.2.4. Constants Module 60](#_Toc345097864)

[2.2.5. DataBase Module 61](#_Toc345097865)

[2.2.6. Desktop Module 61](#_Toc345097866)

[2.2.7. Events Module 65](#_Toc345097867)

[2.2.8. File Module 66](#_Toc345097868)

[2.2.9. Custom Forms Module 68](#_Toc345097869)

[2.2.10. Virtual Drive Module 70](#_Toc345097870)

[2.2.11. Translation Module 74](#_Toc345097871)

[2.2.12. Mappings Module 75](#_Toc345097872)

[2.2.13. Menu Module 77](#_Toc345097873)

[2.2.14. Notification Module 80](#_Toc345097874)

[2.2.15. Objects Module 82](#_Toc345097875)

[2.2.16. Reference Guide Module 90](#_Toc345097876)

[2.2.17. Reporting Module 90](#_Toc345097877)

[2.2.18. Security Module 97](#_Toc345097878)

[2.2.19. Settings Module 101](#_Toc345097879)

[2.2.20. Users Module 102](#_Toc345097880)

[2.2.21. Utility Module 107](#_Toc345097881)

[2.3. Classes, Events, Objects, Mappings and Cache Philosophy 121](#_Toc345097882)

[2.3.1. Class Concept 122](#_Toc345097883)

[2.3.2. Object Concept 124](#_Toc345097884)

[2.3.3. Mapping Concept 128](#_Toc345097885)

[2.3.4. Cache Concept 128](#_Toc345097886)

[2.3.5. Event Concept 130](#_Toc345097887)

[2.4. Conclusions 132](#_Toc345097888)

[3. Communication Layer 133](#_Toc345097889)

[3.1. Settings Class 133](#_Toc345097890)

[3.1.1. Properties 133](#_Toc345097891)

[3.2. SMS Class 133](#_Toc345097892)

[3.2.1. Methods 133](#_Toc345097893)

[3.3. eMail Class 134](#_Toc345097894)

[3.3.1. Methods 134](#_Toc345097895)

[3.4. Dispatch Class 134](#_Toc345097896)

[3.4.1. Methods 134](#_Toc345097897)

[3.5. Database Class 134](#_Toc345097898)

[3.5.1. Properties 135](#_Toc345097899)

[3.5.2. Methods 135](#_Toc345097900)

[3.6. Conclusions 136](#_Toc345097901)

[4. Business Layer 136](#_Toc345097902)

[4.1. Amaryllis Namespace 136](#_Toc345097903)

[4.1.1. Settings Class 136](#_Toc345097904)

[4.1.1.1. Properties 136](#_Toc345097905)

[4.1.2. Page Class 136](#_Toc345097906)

[4.1.2.1. Methods 136](#_Toc345097907)

[4.2. Amaryllis.Controls Namespace 137](#_Toc345097908)

[4.2.1. Bar Control 137](#_Toc345097909)

[4.2.1.1. Methods 137](#_Toc345097910)

[4.2.2. CheckBox Control 138](#_Toc345097911)

[4.2.2.1. Methods 138](#_Toc345097912)

[4.2.2.2. Adjacent Files and Folders 138](#_Toc345097913)

[4.2.3. ComboBox Control 139](#_Toc345097914)

[4.2.3.1. Methods 139](#_Toc345097915)

[4.2.3.2. Adjacent Files and Folders 140](#_Toc345097916)

[4.2.4. Controls Class 140](#_Toc345097917)

[4.2.4.1. Methods 140](#_Toc345097918)

[4.2.4.2. Adjacent Files and Folders 142](#_Toc345097919)

[4.2.5. DatePicker Control 142](#_Toc345097920)

[4.2.5.1. Properties 142](#_Toc345097921)

[4.2.5.2. Methods 142](#_Toc345097922)

[4.2.5.3. Adjacent Files and Folders 143](#_Toc345097923)

[4.2.6. FileUpload Control 143](#_Toc345097924)

[4.2.6.1. Methods 143](#_Toc345097925)

[4.2.6.2. Adjacent Files and Folders 144](#_Toc345097926)

[4.2.7. IconSelector Control 144](#_Toc345097927)

[4.2.7.1. Methods 144](#_Toc345097928)

[4.2.7.2. Adjacent Files and Folders 145](#_Toc345097929)

[4.2.8. ImageUpload Control 145](#_Toc345097930)

[4.2.8.1. Properties 145](#_Toc345097931)

[4.2.8.2. Methods 145](#_Toc345097932)

[4.2.8.3. Adjacent Files and Folders 146](#_Toc345097933)

[4.2.9. Integer Control 146](#_Toc345097934)

[4.2.9.1. Methods 146](#_Toc345097935)

[4.2.9.2. Adjacent Files and Folders 147](#_Toc345097936)

[4.2.10. MultiCheck Control 147](#_Toc345097937)

[4.2.10.1. Methods 147](#_Toc345097938)

[4.2.10.2. Adjacent Files and Folders 148](#_Toc345097939)

[4.2.11. Password Control 148](#_Toc345097940)

[4.2.11.1. Methods 148](#_Toc345097941)

[4.2.11.2. Adjacent Files and Folders 149](#_Toc345097942)

[4.2.12. RadioGroup Control 151](#_Toc345097943)

[4.2.12.1. Methods 151](#_Toc345097944)

[4.2.12.2. Adjacent Files and Folders 151](#_Toc345097945)

[4.2.13. Rating Control 152](#_Toc345097946)

[4.2.13.1. Methods 152](#_Toc345097947)

[4.2.13.2. Adjacent Files and Folders 152](#_Toc345097948)

[4.2.14. RichTextArea Control 152](#_Toc345097949)

[4.2.14.1. Methods 153](#_Toc345097950)

[4.2.14.2. Adjacent Files and Folders 153](#_Toc345097951)

[4.2.15. Select Control 153](#_Toc345097952)

[4.2.15.1. Methods 154](#_Toc345097953)

[4.2.16. Slider Control 155](#_Toc345097954)

[4.2.16.1. Methods 155](#_Toc345097955)

[4.2.16.2. Adjacent Files and Folders 155](#_Toc345097956)

[4.2.17. TextArea Control 155](#_Toc345097957)

[4.2.17.1. Methods 156](#_Toc345097958)

[4.2.18. TextBox Control 156](#_Toc345097959)

[4.2.18.1. Methods 156](#_Toc345097960)

[4.2.19. TimePicker Control 157](#_Toc345097961)

[4.2.19.1. Methods 157](#_Toc345097962)

[4.2.19.2. Adjacent Files and Folders 158](#_Toc345097963)

[4.3. Amaryllis.Data Namespace 158](#_Toc345097964)

[4.3.1. CSV Class 158](#_Toc345097965)

[4.3.1.1. Properties 158](#_Toc345097966)

[4.3.1.2. Methods 158](#_Toc345097967)

[4.3.2. PDF Class 160](#_Toc345097968)

[4.3.2.1. Methods 160](#_Toc345097969)

[4.4. Amaryllis.DataBase Namespace 160](#_Toc345097970)

[4.4.1. Article Class 160](#_Toc345097971)

[4.4.1.1. Properties 160](#_Toc345097972)

[4.4.1.2. Methods 160](#_Toc345097973)

[4.4.1.3. Adjacent Files and Folders 166](#_Toc345097974)

[4.4.2. Class Class 166](#_Toc345097975)

[4.4.2.1. Properties 166](#_Toc345097976)

[4.4.2.2. Methods 166](#_Toc345097977)

[4.4.2.3. Adjacent Files and Folders 173](#_Toc345097978)

[4.4.3. Constant Class 173](#_Toc345097979)

[4.4.3.1. Properties 173](#_Toc345097980)

[4.4.3.2. Methods 173](#_Toc345097981)

[4.4.4. Control Class 174](#_Toc345097982)

[4.4.4.1. Properties 174](#_Toc345097983)

[4.4.4.2. Methods 174](#_Toc345097984)

[4.4.4.3. Adjacent Files and Folders 174](#_Toc345097985)

[4.4.5. Counter Class 174](#_Toc345097986)

[4.4.5.1. Properties 174](#_Toc345097987)

[4.4.5.2. Methods 174](#_Toc345097988)

[4.4.6. Desktop 175](#_Toc345097989)

[4.4.6.1. Properties 175](#_Toc345097990)

[4.4.6.2. Methods 175](#_Toc345097991)

[4.4.6.3. Adjacent Files and Folders 179](#_Toc345097992)

[4.4.7. Event Class 179](#_Toc345097993)

[4.4.7.1. Properties 179](#_Toc345097994)

[4.4.7.2. Methods 179](#_Toc345097995)

[4.4.8. File Class 180](#_Toc345097996)

[4.4.8.1. Properties 180](#_Toc345097997)

[4.4.8.2. Methods 180](#_Toc345097998)

[4.4.8.3. Adjacent Files and Folders 182](#_Toc345097999)

[4.4.9. Form Class 182](#_Toc345098000)

[4.4.9.1. Properties 182](#_Toc345098001)

[4.4.9.2. Methods 182](#_Toc345098002)

[4.4.9.3. Adjacent Files and Folder 184](#_Toc345098003)

[4.4.10. Harddrive Class 184](#_Toc345098004)

[4.4.10.1. Properties 185](#_Toc345098005)

[4.4.10.2. Methods 185](#_Toc345098006)

[4.4.10.3. Adjacent Files and Folders 188](#_Toc345098007)

[4.4.11. Language Class 188](#_Toc345098008)

[4.4.11.1. Properties 189](#_Toc345098009)

[4.4.11.2. Methods 189](#_Toc345098010)

[4.4.12. Mapping Class 189](#_Toc345098011)

[4.4.12.1. Properties 189](#_Toc345098012)

[4.4.12.2. Methods 190](#_Toc345098013)

[4.4.12.3. Adjacent Files and Folders 192](#_Toc345098014)

[4.4.13. Menu Class 192](#_Toc345098015)

[4.4.13.1. Properties 192](#_Toc345098016)

[4.4.13.2. Methods 192](#_Toc345098017)

[4.4.13.3. Adjacent Files and Folders 195](#_Toc345098018)

[4.4.14. Notification Class 195](#_Toc345098019)

[4.4.14.1. Properties 195](#_Toc345098020)

[4.4.14.2. Methods 196](#_Toc345098021)

[4.4.14.3. Adjacent Files and Folders 197](#_Toc345098022)

[4.4.15. Object Class 197](#_Toc345098023)

[4.4.15.1. Properties 197](#_Toc345098024)

[4.4.15.2. Methods 197](#_Toc345098025)

[4.4.15.3. Adjacent Files and Folders 205](#_Toc345098026)

[4.4.16. ReferenceGuide Class 206](#_Toc345098027)

[4.4.16.1. Properties 206](#_Toc345098028)

[4.4.16.2. Methods 206](#_Toc345098029)

[4.4.17. Report Class 206](#_Toc345098030)

[4.4.17.1. Properties 206](#_Toc345098031)

[4.4.17.2. Methods 207](#_Toc345098032)

[4.4.17.3. Adjacent Files and Folders 216](#_Toc345098033)

[4.4.18. Security Class 216](#_Toc345098034)

[4.4.18.1. Properties 216](#_Toc345098035)

[4.4.18.2. Properties 217](#_Toc345098036)

[4.4.18.3. Adjacent Files and Folders 221](#_Toc345098037)

[4.4.19. Setting Class 221](#_Toc345098038)

[4.4.19.1. Properties 221](#_Toc345098039)

[4.4.19.2. Methods 221](#_Toc345098040)

[4.4.20. User Class 222](#_Toc345098041)

[4.4.20.1. Properties 222](#_Toc345098042)

[4.4.20.2. Methods 222](#_Toc345098043)

[4.4.20.3. Adjacent Files and Folders 228](#_Toc345098044)

[4.4.21. Utility Class 228](#_Toc345098045)

[4.4.21.1. Properties 229](#_Toc345098046)

[4.4.21.2. Methods 229](#_Toc345098047)

[4.4.21.3. Adjacent Files and Folders 230](#_Toc345098048)

[4.5. Amaryllis.Objects Namespace 230](#_Toc345098049)

[4.5.1. Border Class 230](#_Toc345098050)

[4.5.1.1. Properties 231](#_Toc345098051)

[4.5.1.2. Methods 231](#_Toc345098052)

[4.5.2. Box Class 231](#_Toc345098053)

[4.5.2.1. Properties 232](#_Toc345098054)

[4.5.2.2. Methods 232](#_Toc345098055)

[4.5.2.3. Adjacent Files and Folders 233](#_Toc345098056)

[4.5.3. Graph Object 233](#_Toc345098057)

[4.5.3.1. Area Type Charts 233](#_Toc345098058)

[4.5.3.2. Bar Type Charts 234](#_Toc345098059)

[4.5.3.3. Box Plot Type Charts 235](#_Toc345098060)

[4.5.3.4. Bubble Type Charts 237](#_Toc345098061)

[4.5.3.5. Doughnut Type Charts 238](#_Toc345098062)

[4.5.3.6. Error Bar Type Charts 239](#_Toc345098063)

[4.5.3.7. Fast Line Type Charts 240](#_Toc345098064)

[4.5.3.8. Fast Point Type Charts 241](#_Toc345098065)

[4.5.3.9. Funnel Type Charts 241](#_Toc345098066)

[4.5.3.10. Kagi Type Charts 242](#_Toc345098067)

[4.5.3.11. Line Type Charts 244](#_Toc345098068)

[4.5.3.12. Pie Type Charts 245](#_Toc345098069)

[4.5.3.13. Point Type Charts 246](#_Toc345098070)

[4.5.3.14. Polar Type Charts 247](#_Toc345098071)

[4.5.3.15. Pyramid Type Charts 248](#_Toc345098072)

[4.5.3.16. Radar Type Charts 249](#_Toc345098073)

[4.5.3.17. Spline Area Type Charts 251](#_Toc345098074)

[4.5.3.18. Stacked Area Type Charts 252](#_Toc345098075)

[4.5.3.19. Stacked Area 100% Type Charts 253](#_Toc345098076)

[4.5.3.20. Stacked Bar Type Charts 254](#_Toc345098077)

[4.5.3.21. Stacked Bar 100% Type Charts 255](#_Toc345098078)

[4.5.3.22. Stacked Column Type Charts 256](#_Toc345098079)

[4.5.3.23. Stacked Column 100% Type Charts 257](#_Toc345098080)

[4.5.3.24. Step Line Type Charts 258](#_Toc345098081)

[4.5.4. Menu Class 259](#_Toc345098082)

[4.5.4.1. Properties 259](#_Toc345098083)

[4.5.4.2. Methods 259](#_Toc345098084)

[4.5.4.3. Adjacent Files and Folders 260](#_Toc345098085)

[4.5.5. MenuBar Class 260](#_Toc345098086)

[4.5.5.1. Properties 261](#_Toc345098087)

[4.5.5.2. Methods 261](#_Toc345098088)

[4.5.5.3. Adjacent Files and Folders 262](#_Toc345098089)

[4.5.6. OverDiv Class 262](#_Toc345098090)

[4.5.6.1. Properties 263](#_Toc345098091)

[4.5.6.2. Methods 263](#_Toc345098092)

[4.5.6.3. Adjacent Files and Folders 263](#_Toc345098093)

[4.5.7. Side Panel Class 263](#_Toc345098094)

[4.5.7.1. Properties 264](#_Toc345098095)

[4.5.7.2. Methods 264](#_Toc345098096)

[4.5.7.3. Adjacent Files and Folders 266](#_Toc345098097)

[4.5.8. Table Class 266](#_Toc345098098)

[4.5.8.1. Properties 268](#_Toc345098099)

[4.5.8.2. Methods 268](#_Toc345098100)

[4.5.9. Tab Panel Class 270](#_Toc345098101)

[4.5.9.1. Properties 271](#_Toc345098102)

[4.5.9.2. Methods 271](#_Toc345098103)

[4.5.9.3. Adjacent Files and Folders 272](#_Toc345098104)

[4.5.10. TaskBar Class 272](#_Toc345098105)

[4.5.10.1. Properties 272](#_Toc345098106)

[4.5.10.2. Methods 273](#_Toc345098107)

[4.5.10.3. Adjacent Files and Folders 273](#_Toc345098108)

[4.6. Amaryllis.Tools Namespace 273](#_Toc345098109)

[4.6.1. Bar Coder Class 273](#_Toc345098110)

[4.6.1.1. Properties 273](#_Toc345098111)

[4.6.1.2. Methods 274](#_Toc345098112)

[4.6.1.3. Adjacent Files and Folders 274](#_Toc345098113)

[4.6.2. Error Class 275](#_Toc345098114)

[4.6.2.1. Properties 275](#_Toc345098115)

[4.6.2.2. Methods 275](#_Toc345098116)

[4.6.3. Form Class 275](#_Toc345098117)

[4.6.3.1. Properties 275](#_Toc345098118)

[4.6.3.2. Methods 275](#_Toc345098119)

[4.6.3.3. Adjacent Files and Folders 276](#_Toc345098120)

[4.6.4. Format Class 276](#_Toc345098121)

[4.6.4.1. Methods 276](#_Toc345098122)

[4.6.5. Graphics Class 278](#_Toc345098123)

[4.6.5.1. Methods 278](#_Toc345098124)

[4.6.6. Image Class 280](#_Toc345098125)

[4.6.6.1. Methods 280](#_Toc345098126)

[4.6.7. LaTeX Class 281](#_Toc345098127)

[4.6.7.1. Methods 281](#_Toc345098128)

[4.6.8. Mapping Class 281](#_Toc345098129)

[4.6.8.1. Methods 281](#_Toc345098130)

[4.6.8.2. Adjacent Files and Folders 282](#_Toc345098131)

[4.6.9. Object Class 282](#_Toc345098132)

[4.6.9.1. Properties 282](#_Toc345098133)

[4.6.9.2. Methods 282](#_Toc345098134)

[4.6.9.3. Adjacent Files and Folders 286](#_Toc345098135)

[4.6.10. Packer Class 287](#_Toc345098136)

[4.6.10.1. Properties 287](#_Toc345098137)

[4.6.10.2. Methods 287](#_Toc345098138)

[4.6.11. Report Class 288](#_Toc345098139)

[4.6.11.1. Methods 288](#_Toc345098140)

[4.6.11.2. Adjacent Files and Folders 290](#_Toc345098141)

[4.6.12. Security Class 290](#_Toc345098142)

[4.6.12.1. Methods 290](#_Toc345098143)

[4.6.13. Amaryllis.Tools.CodeHighlighter Namespace 290](#_Toc345098144)

[4.6.13.1. CLikeFormat Class 291](#_Toc345098145)

[4.6.13.1.1. Properties 291](#_Toc345098146)

[4.6.13.2. CodeFormat Class 291](#_Toc345098147)

[4.6.13.2.1. Properties 291](#_Toc345098148)

[4.6.13.2.2. Methods 291](#_Toc345098149)

[4.6.13.3. CSharpFormat 292](#_Toc345098150)

[4.6.13.3.1. Properties 292](#_Toc345098151)

[4.6.13.4. Highlighter Class 292](#_Toc345098152)

[4.6.13.4.1. Methods 292](#_Toc345098153)

[4.6.13.5. HtmlFormat Class 293](#_Toc345098154)

[4.6.13.5.1. Properties 293](#_Toc345098155)

[4.6.13.5.2. Methods 293](#_Toc345098156)

[4.6.13.6. JavaScriptFormat Class 293](#_Toc345098157)

[4.6.13.6.1. Properties 293](#_Toc345098158)

[4.6.13.7. SourceFormat Class 294](#_Toc345098159)

[4.6.13.7.1. Properties 294](#_Toc345098160)

[4.6.13.7.2. Methods 294](#_Toc345098161)

[4.6.13.8. TSqlFormat Class 295](#_Toc345098162)

[4.6.13.8.1. Properties 295](#_Toc345098163)

[4.7. Amaryllis.Widgets Namespace 295](#_Toc345098164)

[4.7.1. About Class 295](#_Toc345098165)

[4.7.1.1. Properties 295](#_Toc345098166)

[4.7.1.2. Methods 295](#_Toc345098167)

[4.7.1.3. Adjacent Files and Folders 296](#_Toc345098168)

[4.7.2. BarCoder Class 296](#_Toc345098169)

[4.7.2.1. Properties 296](#_Toc345098170)

[4.7.2.2. Methods 296](#_Toc345098171)

[4.7.2.3. Adjacent Files and Folders 296](#_Toc345098172)

[4.7.3. Impersonation Class 297](#_Toc345098173)

[4.7.3.1. Properties 297](#_Toc345098174)

[4.7.3.2. Methods 297](#_Toc345098175)

[4.7.3.3. Adjacent Files and Folders 297](#_Toc345098176)

[4.7.4. Language Class 297](#_Toc345098177)

[4.7.4.1. Properties 297](#_Toc345098178)

[4.7.4.2. Methods 297](#_Toc345098179)

[4.7.4.3. Adjacent Files and Folders 298](#_Toc345098180)

[4.7.5. Notification Class 298](#_Toc345098181)

[4.7.5.1. Properties 298](#_Toc345098182)

[4.7.5.2. Methods 298](#_Toc345098183)

[4.7.5.3. Adjacent Files and Folders 298](#_Toc345098184)

[4.7.6. Zoom Class 299](#_Toc345098185)

[4.7.6.1. Properties 299](#_Toc345098186)

[4.7.6.2. Methods 299](#_Toc345098187)

[4.7.6.3. Adjacent Files and Folders 299](#_Toc345098188)

[5. Client Interface Layer 299](#_Toc345098189)

[5.1. Window Interface 302](#_Toc345098190)

[5.1.1. Programming Classes 302](#_Toc345098191)

[5.1.1.1. Amaryllis.Objects Namespace 302](#_Toc345098192)

[5.1.1.1.1. Page Class 302](#_Toc345098193)

[5.1.1.1.1.1. Properties 303](#_Toc345098194)

[5.1.1.1.1.2. Methods 303](#_Toc345098195)

[5.1.1.1.1.3. Adjacent Files and Folders 304](#_Toc345098196)

[5.1.1.1.2. TreeMenu Class 305](#_Toc345098197)

[5.1.1.1.2.1. Properties 305](#_Toc345098198)

[5.1.1.1.2.2. Methods 305](#_Toc345098199)

[5.1.1.1.2.3. Adjacent Files and Folders 307](#_Toc345098200)

[5.1.1.2. Amaryllis.Widgets Namespace 307](#_Toc345098201)

[5.1.1.2.1. Harddrive Widget 307](#_Toc345098202)

[5.1.1.2.1.1. Properties 307](#_Toc345098203)

[5.1.1.2.1.2. Methods 307](#_Toc345098204)

[5.1.1.2.1.3. Adjacent Files and Folders 307](#_Toc345098205)

[5.1.1.2.2. Harddrive Widget 308](#_Toc345098206)

[5.1.1.2.2.1. Properties 308](#_Toc345098207)

[5.1.1.2.2.2. Methods 308](#_Toc345098208)

[5.1.1.2.2.3. Adjacent Files and Folders 308](#_Toc345098209)

[5.1.1.2.3. TabScroller Widget 308](#_Toc345098210)

[5.1.1.2.3.1. Properties 308](#_Toc345098211)

[5.1.1.2.3.2. Methods 308](#_Toc345098212)

[5.1.1.2.3.3. Adjacent Files and Folders 310](#_Toc345098213)

[5.1.2. Interface Elements 310](#_Toc345098214)

[5.1.2.1. Header Bar 310](#_Toc345098215)

[5.1.2.2. Footer Bar 310](#_Toc345098216)

[5.1.2.3. Content Section 311](#_Toc345098217)

[5.1.2.4. Taskbar 312](#_Toc345098218)

[5.1.2.5. Icon Element 313](#_Toc345098219)

[5.1.2.6. Window Element 313](#_Toc345098220)

[5.1.3. Widgets 316](#_Toc345098221)

[5.1.3.1. Login Widget 316](#_Toc345098222)

[5.1.3.2. Quick Link Widget 317](#_Toc345098223)

[5.1.3.3. Notification Widget 318](#_Toc345098224)

[5.1.3.4. Zoom Widget 318](#_Toc345098225)

[5.1.3.5. Language Widget 319](#_Toc345098226)

[5.1.3.6. About Widget 319](#_Toc345098227)

[5.1.3.7. Impersonation Widget 320](#_Toc345098228)

[5.1.4. Taskbar Section 320](#_Toc345098229)

[5.1.4.1. Start Button 320](#_Toc345098230)

[5.1.4.2. Time and Date Display 320](#_Toc345098231)

[5.1.4.3. Window Adjacent Buttons Zone 320](#_Toc345098232)

[5.1.5. Java Scripts 321](#_Toc345098233)

[5.1.5.1. Controls Folder 321](#_Toc345098234)

[5.1.5.1.1. CheckBox 321](#_Toc345098235)

[5.1.5.1.1.1. Methods 321](#_Toc345098236)

[5.1.5.1.2. ComboBox 321](#_Toc345098237)

[5.1.5.1.2.1. Variables 321](#_Toc345098238)

[5.1.5.1.2.2. Methods 321](#_Toc345098239)

[5.1.5.1.3. Controls 322](#_Toc345098240)

[5.1.5.1.3.1. Methods 322](#_Toc345098241)

[5.1.5.1.4. DatePicker 322](#_Toc345098242)

[5.1.5.1.4.1. Variables 322](#_Toc345098243)

[5.1.5.1.4.2. Methods 323](#_Toc345098244)

[5.1.5.1.4.3. Adjacent Files and Folders 325](#_Toc345098245)

[5.1.5.1.5. FileUpload 325](#_Toc345098246)

[5.1.5.1.5.1. Variables 325](#_Toc345098247)

[5.1.5.1.5.2. Methods 325](#_Toc345098248)

[5.1.5.1.6. IconSelector 325](#_Toc345098249)

[5.1.5.1.6.1. Methods 326](#_Toc345098250)

[5.1.5.1.7. ImageUpload 326](#_Toc345098251)

[5.1.5.1.7.1. Variables 326](#_Toc345098252)

[5.1.5.1.7.2. Methods 326](#_Toc345098253)

[5.1.5.1.8. Integer 327](#_Toc345098254)

[5.1.5.1.8.1. Methods 327](#_Toc345098255)

[5.1.5.1.9. MultiCheck 327](#_Toc345098256)

[5.1.5.1.9.1. Methods 327](#_Toc345098257)

[5.1.5.1.10. Password 328](#_Toc345098258)

[5.1.5.1.10.1. Methods 328](#_Toc345098259)

[5.1.5.1.11. RadioGroup 329](#_Toc345098260)

[5.1.5.1.11.1. Methods 329](#_Toc345098261)

[5.1.5.1.12. Rating 329](#_Toc345098262)

[5.1.5.1.12.1. Methods 329](#_Toc345098263)

[5.1.5.1.13. Slider 330](#_Toc345098264)

[5.1.5.1.13.1. Variables 330](#_Toc345098265)

[5.1.5.1.13.2. Methods 330](#_Toc345098266)

[5.1.5.1.14. TimePicker 331](#_Toc345098267)

[5.1.5.1.14.1. Variables 331](#_Toc345098268)

[5.1.5.1.14.2. Methods 331](#_Toc345098269)

[5.1.5.2. Core Folder 333](#_Toc345098270)

[5.1.5.2.1. Animation 333](#_Toc345098271)

[5.1.5.2.1.1. Variables 333](#_Toc345098272)

[5.1.5.2.1.2. Methods 333](#_Toc345098273)

[5.1.5.2.2. Compat 334](#_Toc345098274)

[5.1.5.2.2.1. Methods 334](#_Toc345098275)

[5.1.5.2.3. Error 336](#_Toc345098276)

[5.1.5.2.3.1. Methods 336](#_Toc345098277)

[5.1.5.2.4. Mouse 336](#_Toc345098278)

[5.1.5.2.4.1. Variables 336](#_Toc345098279)

[5.1.5.2.4.2. Methods 337](#_Toc345098280)

[5.1.5.2.5. Utility 341](#_Toc345098281)

[5.1.5.2.5.1. Methods 341](#_Toc345098282)

[5.1.5.3. Objects Folder 343](#_Toc345098283)

[5.1.5.3.1. Balloon 343](#_Toc345098284)

[5.1.5.3.1.1. Variables 343](#_Toc345098285)

[5.1.5.3.1.2. Methods 343](#_Toc345098286)

[5.1.5.3.2. ContextMenu 344](#_Toc345098287)

[5.1.5.3.2.1. Variables 344](#_Toc345098288)

[5.1.5.3.2.2. Methods 345](#_Toc345098289)

[5.1.5.3.3. Icon 348](#_Toc345098290)

[5.1.5.3.3.1. Variables 348](#_Toc345098291)

[5.1.5.3.3.2. Methods 348](#_Toc345098292)

[5.1.5.3.4. Menu 351](#_Toc345098293)

[5.1.5.3.4.1. Methods 351](#_Toc345098294)

[5.1.5.3.5. MenuBar 352](#_Toc345098295)

[5.1.5.3.5.1. Variables 352](#_Toc345098296)

[5.1.5.3.5.2. Methods 352](#_Toc345098297)

[5.1.5.3.6. ModalBox 354](#_Toc345098298)

[5.1.5.3.6.1. Variables 354](#_Toc345098299)

[5.1.5.3.6.2. Methods 354](#_Toc345098300)

[5.1.5.3.6.3. Adjacent Files and Folders 355](#_Toc345098301)

[5.1.5.3.7. Notification 355](#_Toc345098302)

[5.1.5.3.7.1. Variables 355](#_Toc345098303)

[5.1.5.3.7.2. Methods 355](#_Toc345098304)

[5.1.5.3.8. OverDiv 356](#_Toc345098305)

[5.1.5.3.8.1. Methods 356](#_Toc345098306)

[5.1.5.3.9. Page 357](#_Toc345098307)

[5.1.5.3.9.1. Variables 357](#_Toc345098308)

[5.1.5.3.9.2. Methods 358](#_Toc345098309)

[5.1.5.3.9.3. Adjacent Files and Folders 360](#_Toc345098310)

[5.1.5.3.10. SidePanel 361](#_Toc345098311)

[5.1.5.3.10.1. Methods 361](#_Toc345098312)

[5.1.5.3.11. TabPanel 361](#_Toc345098313)

[5.1.5.3.11.1. Methods 361](#_Toc345098314)

[5.1.5.3.12. TaskBar 361](#_Toc345098315)

[5.1.5.3.12.1. Variables 361](#_Toc345098316)

[5.1.5.3.12.2. Methods 361](#_Toc345098317)

[5.1.5.3.12.3. Adjacent Files and Folders 364](#_Toc345098318)

[5.1.5.3.13. TreeMenu 364](#_Toc345098319)

[5.1.5.3.13.1. Variables 364](#_Toc345098320)

[5.1.5.3.13.2. Methods 364](#_Toc345098321)

[5.1.5.3.13.3. Adjacent Files and Folders 367](#_Toc345098322)

[5.1.5.3.14. Window 367](#_Toc345098323)

[5.1.5.3.14.1. Variables 367](#_Toc345098324)

[5.1.5.3.14.2. Methods 367](#_Toc345098325)

[5.1.5.3.14.3. Adjacent Files and Folders 375](#_Toc345098326)

[5.1.5.4. Resources Folder 375](#_Toc345098327)

[5.1.5.5. Tools Folder 376](#_Toc345098328)

[5.1.5.5.1. Mapping 376](#_Toc345098329)

[5.1.5.5.1.1. Methods 376](#_Toc345098330)

[5.1.5.5.2. Object 376](#_Toc345098331)

[5.1.5.5.2.1. Methods 376](#_Toc345098332)

[5.1.5.5.3. Report 379](#_Toc345098333)

[5.1.5.5.3.1. Methods 379](#_Toc345098334)

[5.1.5.6. WebServices Folder 379](#_Toc345098335)

[5.1.5.6.1. Icon 380](#_Toc345098336)

[5.1.5.6.1.1. Methods 380](#_Toc345098337)

[5.1.5.6.2. Notification 381](#_Toc345098338)

[5.1.5.6.2.1. Variables 381](#_Toc345098339)

[5.1.5.6.2.2. Methods 381](#_Toc345098340)

[5.1.5.6.3. Window 381](#_Toc345098341)

[5.1.5.6.3.1. Methods 382](#_Toc345098342)

[5.1.5.7. Widgets 383](#_Toc345098343)

[5.1.5.7.1. About 383](#_Toc345098344)

[5.1.5.7.1.1. Methods 383](#_Toc345098345)

[5.1.5.7.1.2. Adjacent Files and Folders 383](#_Toc345098346)

[5.1.5.7.2. BarCoder 383](#_Toc345098347)

[5.1.5.7.2.1. Methods 383](#_Toc345098348)

[5.1.5.7.2.2. Adjacent Files and Folders 383](#_Toc345098349)

[5.1.5.7.3. Harddrive 384](#_Toc345098350)

[5.1.5.7.3.1. Methods 384](#_Toc345098351)

[5.1.5.7.3.2. Adjacent Files and Folders 384](#_Toc345098352)

[5.1.5.7.4. Language 384](#_Toc345098353)

[5.1.5.7.4.1. Methods 384](#_Toc345098354)

[5.1.5.7.5. Notification 385](#_Toc345098355)

[5.1.5.7.5.1. Methods 385](#_Toc345098356)

[5.1.5.7.5.2. Adjacent Files and Folders 385](#_Toc345098357)

[5.1.5.7.6. TabScroller 385](#_Toc345098358)

[5.1.5.7.6.1. Variables 385](#_Toc345098359)

[5.1.5.7.6.2. Methods 385](#_Toc345098360)

[5.1.5.7.7. Zoom 386](#_Toc345098361)

[5.1.5.7.7.1. Methods 387](#_Toc345098362)

[5.1.5.7.8. ZoomWindow 388](#_Toc345098363)

[5.1.5.7.8.1. Methods 388](#_Toc345098364)

[5.1.6. Styles 388](#_Toc345098365)

[5.1.7. Tools 388](#_Toc345098366)

[5.1.8. Web Pages 388](#_Toc345098367)

[5.1.8.1. About 388](#_Toc345098368)

[5.1.8.1.1. Adjacent Files and Folders 389](#_Toc345098369)

[5.1.8.2. AccessoryCalc 389](#_Toc345098370)

[5.1.8.2.1. Adjacent Files and Folders 390](#_Toc345098371)

[5.1.8.3. AdminDuplicateObjects 390](#_Toc345098372)

[5.1.8.3.1. Adjacent Files and Folders 390](#_Toc345098373)

[5.1.8.4. AdminMergeObjects 390](#_Toc345098374)

[5.1.8.4.1. Adjacent Files and Folders 391](#_Toc345098375)

[5.1.8.5. AdminSettings 391](#_Toc345098376)

[5.1.8.5.1. Adjacent Files and Folders 392](#_Toc345098377)

[5.1.8.6. AdminUsers 392](#_Toc345098378)

[5.1.8.6.1. Adjacent Files and Folders 393](#_Toc345098379)

[5.1.8.7. AdminUsers\_Insert 393](#_Toc345098380)

[5.1.8.7.1. Adjacent Files and Folders 394](#_Toc345098381)

[5.1.8.8. AdminUsers\_Update 394](#_Toc345098382)

[5.1.8.8.1. Parameters 395](#_Toc345098383)

[5.1.8.8.2. Adjacent Files and Folders 395](#_Toc345098384)

[5.1.8.9. Article 395](#_Toc345098385)

[5.1.8.9.1. Parameters 397](#_Toc345098386)

[5.1.8.9.2. Adjacent Files and Folders 397](#_Toc345098387)

[5.1.8.10. Article\_Insert 397](#_Toc345098388)

[5.1.8.10.1. Adjacent Files and Folders 398](#_Toc345098389)

[5.1.8.11. Article\_Update 398](#_Toc345098390)

[5.1.8.11.1. Parameters 399](#_Toc345098391)

[5.1.8.11.2. Adjacent Files and Folders 399](#_Toc345098392)

[5.1.8.12. ArticleLink 399](#_Toc345098393)

[5.1.8.12.1. Adjacent Files and Folders 400](#_Toc345098394)

[5.1.8.13. ArticleSearch 400](#_Toc345098395)

[5.1.8.13.1. Parameters 401](#_Toc345098396)

[5.1.8.13.2. Adjacent Files and Folders 401](#_Toc345098397)

[5.1.8.14. ArticleStart 401](#_Toc345098398)

[5.1.8.14.1. Adjacent Files and Folders 402](#_Toc345098399)

[5.1.8.15. ArticleTree 402](#_Toc345098400)

[5.1.8.15.1. Adjacent Files and Folders 402](#_Toc345098401)

[5.1.8.16. ArticleTreeFrame 402](#_Toc345098402)

[5.1.8.16.1. Parameters 402](#_Toc345098403)

[5.1.8.16.2. Adjacent Files and Folders 402](#_Toc345098404)

[5.1.8.17. Background 403](#_Toc345098405)

[5.1.8.17.1. Adjacent Files and Folders 403](#_Toc345098406)

[5.1.8.18. BarCode\_Download 403](#_Toc345098407)

[5.1.8.18.1. Parameters 404](#_Toc345098408)

[5.1.8.19. ChangePassword 404](#_Toc345098409)

[5.1.8.19.1. Adjacent Files and Folders 404](#_Toc345098410)

[5.1.8.20. Class 405](#_Toc345098411)

[5.1.8.20.1. Parameters 409](#_Toc345098412)

[5.1.8.20.2. Adjacent Files and Folders 409](#_Toc345098413)

[5.1.8.21. Class\_Insert 409](#_Toc345098414)

[5.1.8.21.1. Adjacent Files and Folders 410](#_Toc345098415)

[5.1.8.22. Class\_InsertAttribute 410](#_Toc345098416)

[5.1.8.22.1. Parameters 413](#_Toc345098417)

[5.1.8.22.2. Adjacent Files and Folders 413](#_Toc345098418)

[5.1.8.23. Class\_UpdateAttribute 413](#_Toc345098419)

[5.1.8.23.1. Parameters 413](#_Toc345098420)

[5.1.8.23.2. Adjacent Files and Folders 413](#_Toc345098421)

[5.1.8.24. Controls\_FileUpdate 413](#_Toc345098422)

[5.1.8.24.1. Adjacent Files and Folders 414](#_Toc345098423)

[5.1.8.25. Controls\_FileUpdate 414](#_Toc345098424)

[5.1.8.25.1. Adjacent Files and Folders 414](#_Toc345098425)

[5.1.8.26. Default 414](#_Toc345098426)

[5.1.8.27. DesktopIcon\_Insert 415](#_Toc345098427)

[5.1.8.27.1. Adjacent Files and Folders 416](#_Toc345098428)

[5.1.8.28. DesktopIcon\_Update 416](#_Toc345098429)

[5.1.8.28.1. Parameters 417](#_Toc345098430)

[5.1.8.28.2. Adjacent Files and Folders 417](#_Toc345098431)

[5.1.8.29. Download 417](#_Toc345098432)

[5.1.8.29.1. Parameters 417](#_Toc345098433)

[5.1.8.30. Error 417](#_Toc345098434)

[5.1.8.30.1. Parameters 417](#_Toc345098435)

[5.1.8.30.2. Adjacent Files and Folders 418](#_Toc345098436)

[5.1.8.31. ExportDownload 418](#_Toc345098437)

[5.1.8.31.1. Parameters 418](#_Toc345098438)

[5.1.8.32. Form\_Insert 418](#_Toc345098439)

[5.1.8.32.1. Adjacent Files and Folders 419](#_Toc345098440)

[5.1.8.33. FormAddMapping 419](#_Toc345098441)

[5.1.8.33.1. Parameters 420](#_Toc345098442)

[5.1.8.33.2. Adjacent Files and Folders 420](#_Toc345098443)

[5.1.8.34. Forms 420](#_Toc345098444)

[5.1.8.34.1. Parameters 422](#_Toc345098445)

[5.1.8.34.2. Adjacent Files and Folders 422](#_Toc345098446)

[5.1.8.35. FormView 422](#_Toc345098447)

[5.1.8.35.1. Parameters 422](#_Toc345098448)

[5.1.8.35.2. Adjacent Files and Folders 422](#_Toc345098449)

[5.1.8.36. Harddrive 423](#_Toc345098450)

[5.1.8.36.1. Parameters 424](#_Toc345098451)

[5.1.8.36.2. Adjacent Files and Folders 424](#_Toc345098452)

[5.1.8.37. HarddriveDownload 424](#_Toc345098453)

[5.1.8.37.1. Parameters 424](#_Toc345098454)

[5.1.8.38. HarddriveFile\_Insert 424](#_Toc345098455)

[5.1.8.38.1. Parameters 425](#_Toc345098456)

[5.1.8.38.2. Adjacent Files and Folders 425](#_Toc345098457)

[5.1.8.39. HarddriveFile\_Update 425](#_Toc345098458)

[5.1.8.39.1. Parameters 425](#_Toc345098459)

[5.1.8.39.2. Adjacent Files and Folders 425](#_Toc345098460)

[5.1.8.40. HarddriveFolder\_Insert 426](#_Toc345098461)

[5.1.8.40.1. Parameters 426](#_Toc345098462)

[5.1.8.40.2. Adjacent Files and Folders 426](#_Toc345098463)

[5.1.8.41. HarddriveFolder\_Update 426](#_Toc345098464)

[5.1.8.41.1. Parameters 427](#_Toc345098465)

[5.1.8.41.2. Adjacent Files and Folders 427](#_Toc345098466)

[5.1.8.42. HarddriveIcons 427](#_Toc345098467)

[5.1.8.42.1. Adjacent Files and Folders 428](#_Toc345098468)

[5.1.8.43. HarddrivePage 429](#_Toc345098469)

[5.1.8.43.1. Parameters 429](#_Toc345098470)

[5.1.8.43.2. Adjacent Files and Folders 429](#_Toc345098471)

[5.1.8.44. HarddriveSearch 429](#_Toc345098472)

[5.1.8.44.1. Adjacent Files and Folders 430](#_Toc345098473)

[5.1.8.45. Login 430](#_Toc345098474)

[5.1.8.45.1. Parameters 432](#_Toc345098475)

[5.1.8.45.2. Adjacent Files and Folders 432](#_Toc345098476)

[5.1.8.46. Login2 432](#_Toc345098477)

[5.1.8.46.1. Parameters 433](#_Toc345098478)

[5.1.8.46.2. Adjacent Files and Folders 433](#_Toc345098479)

[5.1.8.47. Logout 433](#_Toc345098480)

[5.1.8.48. MappingObject 434](#_Toc345098481)

[5.1.8.48.1. Parameters 434](#_Toc345098482)

[5.1.8.48.2. Adjacent Files and Folders 434](#_Toc345098483)

[5.1.8.49. Menu 434](#_Toc345098484)

[5.1.8.49.1. Adjacent Files and Folders 435](#_Toc345098485)

[5.1.8.50. Menu\_InsertCommand 436](#_Toc345098486)

[5.1.8.50.1. Adjacent Files and Folders 436](#_Toc345098487)

[5.1.8.51. Menu\_InsertMenu 436](#_Toc345098488)

[5.1.8.51.1. Adjacent Files and Folders 437](#_Toc345098489)

[5.1.8.52. Menu\_UpdateCommand 437](#_Toc345098490)

[5.1.8.52.1. Parameters 437](#_Toc345098491)

[5.1.8.52.2. Adjacent Files and Folders 437](#_Toc345098492)

[5.1.8.53. Menu\_UpdateMenu 438](#_Toc345098493)

[5.1.8.53.1. Parameters 438](#_Toc345098494)

[5.1.8.53.2. Adjacent Files and Folders 438](#_Toc345098495)

[5.1.8.54. Notification 438](#_Toc345098496)

[5.1.8.54.1. Adjacent Files and Folders 438](#_Toc345098497)

[5.1.8.55. Object\_Insert 439](#_Toc345098498)

[5.1.8.55.1. Parameters: 439](#_Toc345098499)

[5.1.8.55.2. Adjacent Files and Folders 440](#_Toc345098500)

[5.1.8.56. Object\_Update 440](#_Toc345098501)

[5.1.8.56.1. Parameters 444](#_Toc345098502)

[5.1.8.56.2. Adjacent Files and Folders 444](#_Toc345098503)

[5.1.8.57. ObjectList 444](#_Toc345098504)

[5.1.8.57.1. Parameters 445](#_Toc345098505)

[5.1.8.57.2. Adjacent Files and Folders 445](#_Toc345098506)

[5.1.8.58. ObjectLog 445](#_Toc345098507)

[5.1.8.58.1. Parameters 446](#_Toc345098508)

[5.1.8.58.2. Adjacent Files and Folders 446](#_Toc345098509)

[5.1.8.59. QuickLink 446](#_Toc345098510)

[5.1.8.59.1. Adjacent Files and Folders 447](#_Toc345098511)

[5.1.8.60. Recycle 447](#_Toc345098512)

[5.1.8.60.1. Adjacent Files and Folders 447](#_Toc345098513)

[5.1.8.61. RefGuide\_Class 447](#_Toc345098514)

[5.1.8.61.1. Parameters 450](#_Toc345098515)

[5.1.8.61.2. Adjacent Files and Folders 450](#_Toc345098516)

[5.1.8.62. RefGuide\_ClassList 450](#_Toc345098517)

[5.1.8.62.1. Adjacent Files and Folders 450](#_Toc345098518)

[5.1.8.63. Report 451](#_Toc345098519)

[5.1.8.63.1. Parameters 454](#_Toc345098520)

[5.1.8.63.2. Adjacent Files and Folders 454](#_Toc345098521)

[5.1.8.64. Report\_Insert 454](#_Toc345098522)

[5.1.8.64.1. Adjacent Files and Folders 455](#_Toc345098523)

[5.1.8.65. Report\_InsertGraph 455](#_Toc345098524)

[5.1.8.65.1. Parameters 455](#_Toc345098525)

[5.1.8.65.2. Adjacent Files and Folders 456](#_Toc345098526)

[5.1.8.66. Report\_UpdateGraph 456](#_Toc345098527)

[5.1.8.66.1. Parameters 456](#_Toc345098528)

[5.1.8.66.2. Adjacent Files and Folders 456](#_Toc345098529)

[5.1.8.67. ReportGraphView 457](#_Toc345098530)

[5.1.8.67.1. Parameters 457](#_Toc345098531)

[5.1.8.68. ReportParameter\_Insert 457](#_Toc345098532)

[5.1.8.68.1. Parameters 458](#_Toc345098533)

[5.1.8.68.2. Adjacent Files and Folders 459](#_Toc345098534)

[5.1.8.69. ReportParameter\_Update 459](#_Toc345098535)

[5.1.8.69.1. Parameter 459](#_Toc345098536)

[5.1.8.69.2. Adjacent Files and Folders 459](#_Toc345098537)

[5.1.8.70. ReportView 459](#_Toc345098538)

[5.1.8.70.1. Parameters 460](#_Toc345098539)

[5.1.8.70.2. Adjacent Files and Folders 460](#_Toc345098540)

[5.1.8.71. Search 460](#_Toc345098541)

[5.1.8.71.1. Adjacent Files and Folders 460](#_Toc345098542)

[5.1.8.72. Security 460](#_Toc345098543)

[5.1.8.72.1. Parameters 463](#_Toc345098544)

[5.1.8.72.2. Adjacent Files and Folders 464](#_Toc345098545)

[5.1.8.73. Security\_InsertGroup 464](#_Toc345098546)

[5.1.8.73.1. Adjacent Files and Folders 464](#_Toc345098547)

[5.1.8.74. Settings 464](#_Toc345098548)

[5.1.8.74.1. Adjacent Files and Folders 465](#_Toc345098549)

[5.1.8.75. StopImpersonation 465](#_Toc345098550)

[5.1.8.76. Task 465](#_Toc345098551)

[5.1.8.76.1. Adjacent Files and Folders 466](#_Toc345098552)

[5.1.8.77. UserChangeData 466](#_Toc345098553)

[5.1.8.77.1. Adjacent Files and Folders 468](#_Toc345098554)

[5.1.8.78. UtilityCloseWindow 468](#_Toc345098555)

[5.1.8.78.1. Parameters 468](#_Toc345098556)

[5.1.8.79. Welcome 468](#_Toc345098557)

[5.2. Plain Interface 469](#_Toc345098558)

[5.2.1. Programming Classes 469](#_Toc345098559)

[5.2.1.1. Amaryllis.Objects Namespace 469](#_Toc345098560)

[5.2.1.1.1. Page Class 469](#_Toc345098561)

[5.2.1.1.1.1. Properties 469](#_Toc345098562)

[5.2.1.1.1.2. Methods 469](#_Toc345098563)

[5.2.1.1.1.3. Adjacent Files and Folders 471](#_Toc345098564)

[5.2.1.1.2. TreeMenu Class 471](#_Toc345098565)

[5.2.1.1.2.1. Properties 471](#_Toc345098566)

[5.2.1.1.2.2. Methods 471](#_Toc345098567)

[5.2.1.1.2.3. Adjacent Files and Folders 473](#_Toc345098568)

[5.2.2. Interface Elements 473](#_Toc345098569)

[5.2.2.1. Header Bar 473](#_Toc345098570)

[5.2.2.2. Footer Bar 473](#_Toc345098571)

[5.2.2.3. Content Section 473](#_Toc345098572)

[5.2.2.4. Menu 474](#_Toc345098573)

[5.2.2.5. Window Element 474](#_Toc345098574)

[5.2.3. Widgets 474](#_Toc345098575)

[5.2.3.1. Quick Link Widget 474](#_Toc345098576)

[5.2.3.2. Notification Widget 474](#_Toc345098577)

[5.2.3.3. Zoom Widget 474](#_Toc345098578)

[5.2.3.4. Language Widget 474](#_Toc345098579)

[5.2.3.5. About Widget 474](#_Toc345098580)

[5.2.3.6. Impersonation Widget 474](#_Toc345098581)

[5.2.4. Taskbar Section 475](#_Toc345098582)

[5.2.4.1. Window Adjacent Buttons Zone 475](#_Toc345098583)

[5.2.5. Java Scripts 475](#_Toc345098584)

[5.2.5.1. Controls Folder 475](#_Toc345098585)

[5.2.5.2. Core Folder 475](#_Toc345098586)

[5.2.5.2.1. Animation 475](#_Toc345098587)

[5.2.5.2.2. Compat 475](#_Toc345098588)

[5.2.5.2.3. Error 475](#_Toc345098589)

[5.2.5.2.4. Utility 475](#_Toc345098590)

[5.2.5.2.4.1. Methods 475](#_Toc345098591)

[5.2.5.3. Objects Folder 476](#_Toc345098592)

[5.2.5.3.1. Menu 476](#_Toc345098593)

[5.2.5.3.2. MenuBar 476](#_Toc345098594)

[5.2.5.3.3. ModalBox 476](#_Toc345098595)

[5.2.5.3.4. OverDiv 476](#_Toc345098596)

[5.2.5.3.5. Page 476](#_Toc345098597)

[5.2.5.3.5.1. Variables 476](#_Toc345098598)

[5.2.5.3.5.2. Methods 476](#_Toc345098599)

[5.2.5.3.6. SidePanel 477](#_Toc345098600)

[5.2.5.3.7. TabPanel 477](#_Toc345098601)

[5.2.5.3.8. TaskBar 477](#_Toc345098602)

[5.2.5.3.8.1. Methods 477](#_Toc345098603)

[5.2.5.3.9. TreeMenu 478](#_Toc345098604)

[5.2.5.3.9.1. Methods 478](#_Toc345098605)

[5.2.5.3.10. Window 479](#_Toc345098606)

[5.2.5.3.10.1. Variables 479](#_Toc345098607)

[5.2.5.3.10.2. Methods 479](#_Toc345098608)

[5.2.5.4. Resources Folder 481](#_Toc345098609)

[5.2.5.5. Tools Folder 481](#_Toc345098610)

[5.2.5.6. Widgets 481](#_Toc345098611)

[5.2.5.6.1. About 481](#_Toc345098612)

[5.2.5.6.2. BarCoder 481](#_Toc345098613)

[5.2.5.6.3. Language 481](#_Toc345098614)

[5.2.5.6.4. Notification 481](#_Toc345098615)

[5.2.5.6.5. Zoom 481](#_Toc345098616)

[5.2.5.6.6. ZoomWindow 481](#_Toc345098617)

[5.2.6. Styles 482](#_Toc345098618)

[5.2.7. Tools 482](#_Toc345098619)

[5.2.8. Web Pages 482](#_Toc345098620)

[6. Special Syntaxes 482](#_Toc345098621)

[6.1. Specific XML Syntax 482](#_Toc345098622)

[6.2. Specific HTML Syntax 483](#_Toc345098623)

[6.2.1. Class and Report Development Syntax 483](#_Toc345098624)

[6.2.1.1. Language Tag Tokens 483](#_Toc345098625)

[6.2.1.1.1. Single-look and Multi-look 483](#_Toc345098626)

[6.2.1.1.2. Error Message 484](#_Toc345098627)

[6.2.1.2. Class Attributes Tags 484](#_Toc345098628)

[6.2.1.2.1. Composed Tags 487](#_Toc345098629)

[6.2.1.3. Class Tags 487](#_Toc345098630)

[6.2.1.4. Special Tags 488](#_Toc345098631)

[6.2.2. Report Development Syntax 489](#_Toc345098632)

[6.2.2.1. Language Tag Tokens 489](#_Toc345098633)

[6.2.2.2. Report Parameter Tags 489](#_Toc345098634)

[6.2.2.3. Class Tags 489](#_Toc345098635)

[6.2.3. Custom Form Development Syntax 489](#_Toc345098636)

[6.2.3.1. Side Tab Panel Objects 489](#_Toc345098637)

[6.2.3.2. Top Tab Panel Objects 489](#_Toc345098638)

[6.2.3.3. Listing 490](#_Toc345098639)

[6.2.3.4. Object 490](#_Toc345098640)

[6.2.3.5. Mappings 490](#_Toc345098641)

[6.2.3.6. Simple Tags 491](#_Toc345098642)

[6.2.3.6.1. Object Associated Tags 491](#_Toc345098643)

[6.2.3.6.2. Add Mapping Tag 491](#_Toc345098644)

[6.2.3.7. Complex Tags 491](#_Toc345098645)

[6.2.3.7.1. Objects Tag 491](#_Toc345098646)

[6.2.3.7.2. Matrix Tag 492](#_Toc345098647)

[6.2.3.8. Conclusions 492](#_Toc345098648)

[6.2.4. Documentation Syntax 492](#_Toc345098649)

[6.2.4.1. Code Tag 492](#_Toc345098650)

[6.2.4.2. Box and Sidebox Tags 493](#_Toc345098651)

[6.2.4.3. Latex Tag 493](#_Toc345098652)

[6.2.4.4. The HDD Tags 494](#_Toc345098653)

[6.2.4.5. Bullet Tag 494](#_Toc345098654)

[6.3. Specific SQL Syntax 494](#_Toc345098655)

[6.3.1. Class Development Syntax 494](#_Toc345098656)

[6.3.1.1. Language Tags 494](#_Toc345098657)

[6.3.1.2. Class Attribute Tags 494](#_Toc345098658)

[6.3.1.3. Special Tags 495](#_Toc345098659)

[6.3.2. Report Development 496](#_Toc345098660)

[6.3.2.1. Report Parameter Tags 496](#_Toc345098661)

[6.3.2.2. Class Tags 496](#_Toc345098662)

[Annex 1 – Global Entity-Relation Diagram 497](#_Toc345098663)

[Annex 2 – Platform Layers and Interactions 498](#_Toc345098664)

[Annex 3 – Default Web Page Access for the Default Security Groups 499](#_Toc345098665)

[Annex 4 – Web Page Availability for the Interfaces 501](#_Toc345098666)

[Annex A – Platform Changes Tracker 503](#_Toc345098667)

[Version 2.5.55 – Manganese 55 503](#_Toc345098668)

[Version 2.5.54 – Manganese 54 504](#_Toc345098669)

[Annex B – Manual Changes Tracker 505](#_Toc345098670)

# Introduction

Giving the number of applications developed in the IT domain and knowing the fact that many of these applications are personalized, the applications in this department are diverse (design and feature wise), but we can identify a few sets of solutions or common modules between solutions (e.g. most of the IT solutions have a module for user administration). From all the sets of IT solutions found on the market the Amaryllis platform concentrates on the set represented by document and workflow orientated products.

What are document and workflow orientated products? Basically, they are applications in which the users fill in a bunch of forms/field from a bunch of forms (depending on their access clearance) and the saved forms may determine field updates on other forms. For a simple example we will consider one of the simplest applications of this kind, more specifically, an application that manages the employee list for a company.

## Description for the Chosen Sample

Going for an employee managing application, the first question raised is “What are the attributes that identify/characterize a person?”. First answer for this question will be: “Name, family name, PIN[[1]](#footnote-1), birthdate, birth place and address.”, which represent information found on the ID card of each person/employee.

Stepping up, most of the employee management solutions generate some reports in which we can find supplementary information from the ID cards, such as the document series and number, and knowing the fact that a person changes a few ID cards in a lifetime, we challenge our self with the problem of being able to generate past reports with the data we had at that time. At this point we deduce the following: a person may have multiple ID cards along the years, so a person is identified in the present with the data found on his latest ID card, but the application must retain the information found on all the ID cards that a person had.

## Problem Details

For most applications of this kind we have something called a data layer. In this layer each document/form is represented by a table and the table’s columns contain the values for the different fields found on the form. Furthermore, for each document we will have a form/web page that displays the field needed for fata collecting and behind these fields we have the business logic part which reacts to data saving.

Given the above situation, if we need a new field for a document, the work involved in order to accommodate the new field can be very stressing: table modification in the data layer to accommodate the new column, modification on the form/web page to accommodate a new field and changes in the business layer to validate the new field, correlate it with the other fields and pass it on to the data layer. On top of the development and testing the solution with the new field, we have the deployment part which can mean more work.

## Motivation and Conclusion

Reviewing all the developed personalized solutions for different clients, all the document oriented applications have the same logical structure and similar problems when facing data structure changes or new documents/forms. Noticing the similarities between different solutions and their evolution when used by the client, the conclusion was that for fast development we need a platform that supports any kind of document and any workflow imaginable that has a minimal impact on the data layer, business layer and user interface when changing a document format or workflow.

# Data Layer

This chapter will explain in detail the table structure (tables, columns and relations between tables), stored procedures and user defined functions (scope, effects and parameters). We will split the tables in modules just to give you a better view of things and keep the entity-relation schemas to a minimum.

In order to have a big picture regarding the tables and the relations between them you may refer to Annex 1 representing the entity-relation diagram for all the entities. For each described module we will present a diagram containing the specific tables for the module itself and all the level 1 adjacent tables.

## Data Tables Grouped by Module

### User Management Module

This module is very common to most of the applications and its main goal is to give the client the possibility to manage a user list for the platform.



User management module diagram

Note: For this module the File table is an adjacent table so it will be detailed in its own module.

**User**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key[[2]](#footnote-2) |
| PhotoID | INT |  | Foreign key[[3]](#footnote-3) for the File table. Links the file representing the user photo. If the user has no photo, this field is filled with a reference to a default image. |
| IsLocked | INT |  | Represents the number of successive failed authentication attempts. It resets to 0 if the authentication ends with success. |
| NextPasswordChangeDate | DATETIME |  | The date when the user is required to change his password (if the platform is configured for this). |
| Phone | VARCHAR(16) |  | User’s phone number used for SMS notifications. |
| UserName | VARCHAR(64) |  |  |
| FirstName | NVARCHAR(64) |  |  |
| LastName | NVARCHAR(64) |  |  |
| GUID | VARCHAR(64) |  | A GUID[[4]](#footnote-4) generated with the SQL’s NEWID() function used to auto-login a user in the case of direct links. |
| eMail | VARCHAR(128) |  | User’s e-mail used for notifications sent via e-mail. |
| Password | VARBINARY(128) |  | The user’s password hashed using the SQL’s PWDENCRYPT(…) function. |

**UserApartenence**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| UserID | INT |  | Foreign key for the User table. |
| GroupID | INT |  | Foreign key for the UserGroup table. |

**UserGroup**

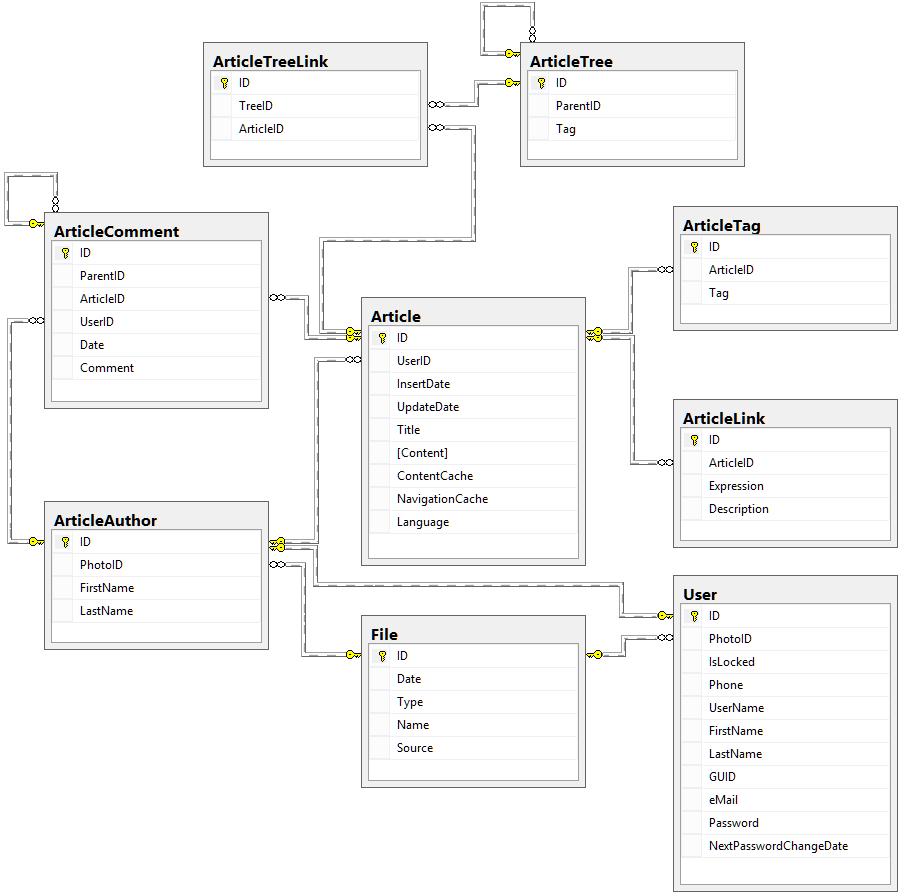
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(64) |  |  |

### Documentation Module

This module was developed in order to be able to provide documentation inside the platform for all the users. As a default for this module, there are some generic articles loaded regarding development and all the basic help to support the developer in building an application over the platform.

The developer may add articles of its own in order to provide help for the final user of the developed application. He can link these articles to object forms and so on in order to give a user localized and context specific documentation.

In order to preserve the author data, the users are duplicated in the ArticleAuthor table. This is done because when a user is deleted its article will not have a valid user entry to link with. By duplicating the user data, even if the user is deleted from the User table, the articles will have an entry to link with and display the proper author information.



Documentation module diagram

NOTE: For this module the File and User tables are adjacent tables so they will be detailed in their own modules.

**Article**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| UserID | INT |  | Foreign key for the ArticleAuthor table. |
| InsertDate | DATETIME |  | The date when the article was created. |
| UpdateDate | DATETIME |  | The date when the article was last modified. |
| Title | NVARCHAR(128) |  | Article title |
| Content | NVARCHAR(MAX) |  | Article content in specific HTML form given by the user. |
| ContentCache | NVARCHAR(MAX) |  | Article content processed and interpreted in pure HTML. This field is computed when the article is accessed and it is empty. This field is reset to NULL when the article is modified. |
| NavigationCache | NVARCHAR(MAX) |  | Table of content for the article processed in pure HTML from the content. This field behaves the same as the ContentCache field. |
| Language | VARCHAR(8) |  | The language code in which the article was written. |

**ArticleAuthor**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| PhotoID | INT |  | Foreign key for the File table. Links the file representing the user photo. If the user has no photo, this field is filled with a reference to a default image. |
| FirstName | NVARCHAR(64) |  |  |
| LastName | NVARCHAR(64) |  |  |

**ArticleComment**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ParentID | INT |  | Foreign key for the ArticleComment table. This is needed in order to keep comments at comments. |
| ArticleID | INT |  | Foreign key for the Article table. |
| UserID | INT |  | Foreign key for the ArticleAuthor table. |
| Date | DATETIME |  | The date when the comment was created. |
| Comment | NVARCHAR(MAX) |  | The comment in pure HTML form. |

**ArticleLink**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ArticleID | INT |  | Foreign key for the Article table. If the expression has an article behind it that can detail the description even further, this field may hold that article’s ID. |
| Expression | NVARCHAR(512) |  | The common expression in text form. |
| Description | NVARCHAR(4000) |  | The description for the common expression. |

**ArticleTag**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ArticleID | INT |  | Foreign key for the Article table. |
| Tag | NVARCHAR(128) |  |  |

**ArticleTree**

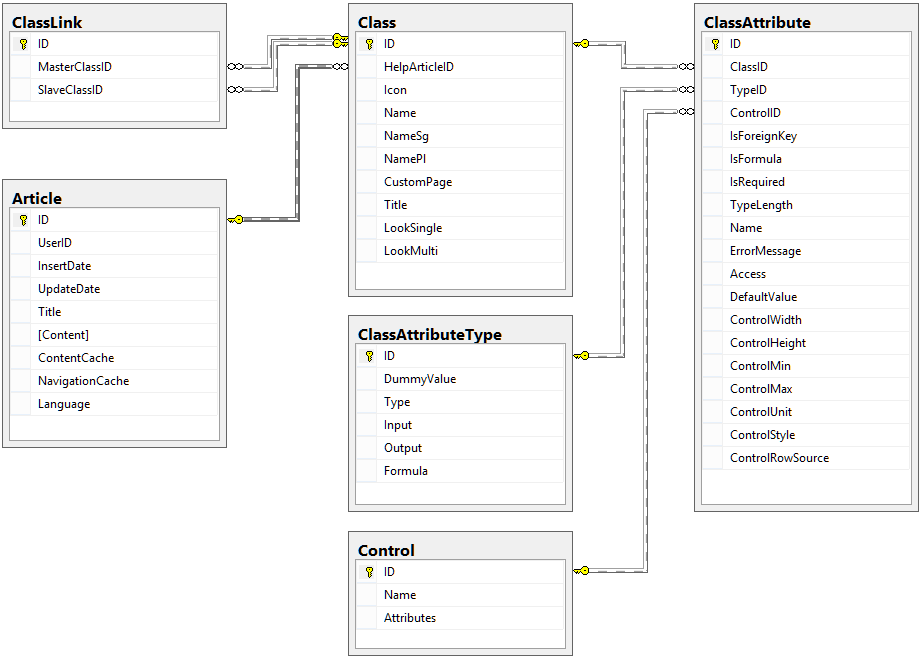
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ParentID | INT |  | Foreign key for the ArticleTree table. This is needed in order to have a hierarchy for a tree structure. |
| Tag | NVARCHAR(128) |  |  |

**ArticleTreeLink**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| TreeID | INT |  | Foreign key for the ArticleTree table. |
| ArticleID | INT |  | Foreign key for the Article table. |

### Class Module

This module is destined to aid the developer in adding and managing the structure for the existing documents and forms that the application client uses. All the platform’s main features revolve around this module so we may say that this module represents the core of the platform.



Class module diagram

NOTE: For this module the Article and Control tables are adjacent tables so they will be detailed in their own modules.

**Class**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| HelpArticleID | INT |  | Foreign key for the Article table. |
| Icon | VARCHAR(128) |  | The web relative path for the icon associated with the class. |
| Name | VARCHAR(128) |  | The internal name for the class used in development. |
| NameSg | VARCHAR(256) |  | The class name in singular form used for the client display. |
| NamePl | VARCHAR(256) |  | The class name in plural form used for the client display. |
| CustomPage | VARCHAR(256) |  | The relative path for the custom web page design for this class, if any. |
| Title | VARCHAR(8000) |  | A tokened[[5]](#footnote-5) SQL specific expression that will determine the title for all the objects in the class. |
| LookSingle | VARCHAR(MAX) |  | A HTML specific description for the form used to modify an object for the class. |
| MookMulti | VARCHAR(MAX) |  | A HTML specific description used to display a listing for the objects in the class. |

**ClassAttribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ClassID | INT |  | Foreign key for the Class table. |
| TypeID | INT |  | Foreign key for the ClassAttributeType table. |
| ControlID | INT |  | Foreign key for the Control table. |
| IsForeignKey | INT |  | Used as a Boolean to determine is the attribute is a foreign key or not. |
| IsFormula | INT |  | Used as a Boolean to specify if the attribute is computed or not. |
| IsRequired | INT |  | Used as a Boolean to specify if the value for the attribute is required or not. |
| TypeLength | VARCHAR(8) |  | For certain attribute types this field is required to determine the maximum length for the value. Sample: NVARCHAR type needs a length between 1 and 4000 or MAX. |
| Name | VARCHAR (128) |  | The attribute’s name used only in development. |
| ErrorMessage | VARCHAR (256) |  | If the attribute is required this field contains the error message to be displayed if the attribute has no value. |
| Access | VARCHAR (MAX) |  | A specific SQL statement that returns ‘READ’ or ‘WRITE’ depending on the user’s access to this field. |
| DefaultValue | VARCHAR (MAX) |  | A specific SQL expression that returns the default value for the attribute or the value for the computed attributes. |
| ControlWidth | INT |  | Depending on the control for this attribute this specifies the display width of the control. |
| ControlHeight | INT |  | Depending on the control for this attribute this specifies the display height of the control. |
| ControlMin | FLOAT |  | Depending on the control for this attribute this specifies the minimum value for the attribute value. |
| ControlMax | FLOAT |  | Depending on the control for this attribute this specifies the maximum value for the attribute value. |
| ControlUnit | FLOAT |  | Depending on the control for this attribute this specifies the fixed increment unit for the attribute value. |
| ControlStyle | VARCHAR (512) |  | Depending on the control for this attribute this specifies the HTML style for the displayed control. |
| ControlRowSource | VARCHAR (MAX) |  | Depending on the control for this attribute this is a specific SQL query that returns the data for the control. |

**ClassAttributeType**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| DummyValue | VARCHAR(32) |  | A fixed value used for specific SQL statements and triggers. |
| Type | VARCHAR (64) |  | The name of the attribute type. |
| Input | VARCHAR (128) |  | A tokened SQL statement used to convert the values from text to the desired SQL data types. |
| Output | VARCHAR (128) |  | A tokened SQL statement used to convert values from SQL data types to text in order to be displayed to the user. |
| Formula | VARCHAR (128) |  | A tokened SQL statement used to convert values to specific format in order to be processed in specific SQL statements. |

**ClassLink**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| MasterClassID | INT |  | Foreign key for the Class table. Determines the master class in the relation between two classes. |
| SlaveClassID | INT |  | Foreign key for the Class table. Determines the slave class in the relation between two classes. |

### Constants module

This module holds the list of parameters/constants needed to configure the application, such as the default interface language or the default image to be displayed in case of no image present.

**Constant**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Constant | VARCHAR(64) |  | The name of the constant. |
| Value | NVARCHAR(64) |  | The value for the constant. |

### Controls module

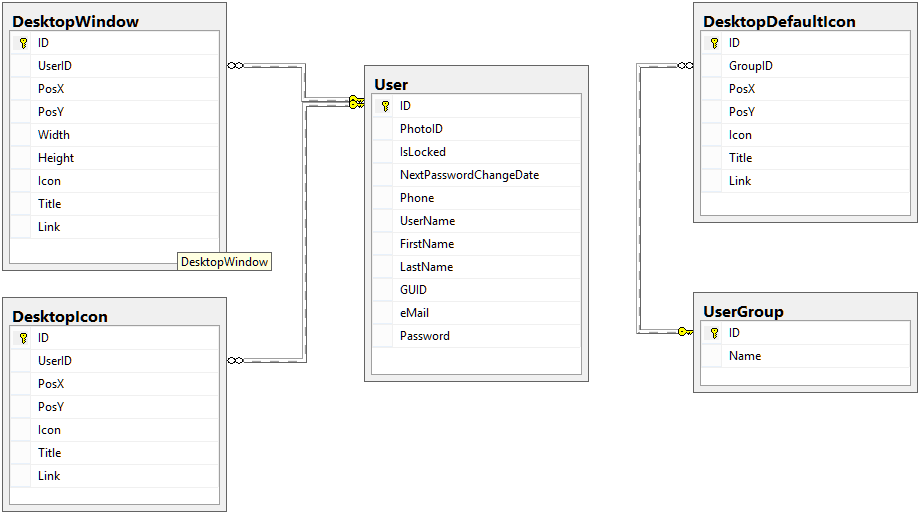
This module contains in a small table used as a list of the controls through which a value for an attribute can be collected.

**Control**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(128) |  | The name of the control. |
| Attributes | VARCHAR(8000) |  | The list of available custom properties for the control separated by the “,” character, such as width or style. |

### Desktop Module

This module is used by the client interface in order to preserve the state of a user’s session. The basic information stored in this module is related to the windows opened by the user and the icons found on the user’s desktop. However we embedded in this module a list of default icons that a user must have depending on the access groups that he belong to.



Desktop module diagram

NOTE: For this module the User and UserGroup tables are adjacent tables so they will be detailed in their own modules.

**DesktopDefaultIcon**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| GroupID | INT |  | Foreign key for the UserGroup table. |
| PosX | INT |  | Horizontal position in pixels on the desktop. |
| PosY | INT |  | Vertical position in pixels on the desktop. |
| Icon | VARCHAR(128) |  | Relative web path for the pictogram to be displayed. |
| Title | NVARCHAR(256) |  | The icon title. |
| Link | VARCHAR(512) |  | The web link that will be opened when double-clicking on the icon. |

**DesktopIcon**

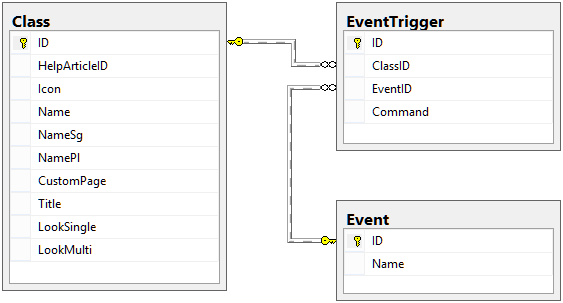
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| UserID | INT |  | Foreign key for the User table. |
| PosX | INT |  | Horizontal position in pixels on the desktop. |
| PosY | INT |  | Vertical position in pixels on the desktop. |
| Icon | VARCHAR(128) |  | Relative web path for the pictogram to be displayed. |
| Title | NVARCHAR(256) |  | The icon title. |
| Link | VARCHAR(512) |  | The web link that will be opened when double-clicking on the icon. |

**DesktopWindow**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| UserID | INT |  | Foreign key for the User table. |
| PosX | INT |  | Horizontal position in pixels on the desktop. |
| PosY | INT |  | Vertical position in pixels on the desktop. |
| Width | INT |  | The window’s width in pixels. |
| Height | INT |  | The window’s height in pixels. |
| Icon | VARCHAR(128) |  | Relative web path for the pictogram to be displayed. |
| Title | NVARCHAR(256) |  | The window title. |
| Link | VARCHAR(512) |  | The web link that will be loaded by the window. |

### Events Module

This module contains the specific SQL queries that are executed for each class when a class object triggers a platform supported event.



Events module diagram

Note: For this module the Class table is an adjacent table so it will be detailed in its own module.

**Event**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(128) |  | The name of the event. |

**EventTrigger**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ClassID | INT |  | Foreign key for the Class table. |
| EventID | INT |  | Foreign key for the Event table. |
| Command | NVARCHAR(MAX) |  | A specific SQL command that must be executed for every object in the referenced class when the referenced event is triggered. |

### File Module

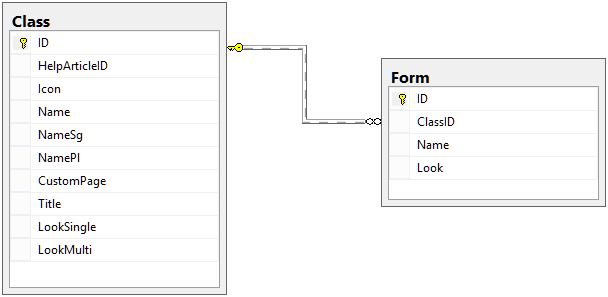
This module consists in a single table that stores all the files uploaded in the application. This is done in order to preserve the files when porting the application from one server to another. The files stored by this module are downloaded on the webserver disk and kept there as a cache. If a file is missing or changed, the file will be downloaded from this module to the webserver’s cache.

**File**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Date | DATETIME |  | The date when the file was last modified. |
| Type | VARCHAR(8) |  | The file extension, such as “.png” or “.pdf”. This is required in case we want to define a full-text index over the stored files. |
| Name | VARCHAR(1024) |  | A virtual name and path for the file. |
| Source | IMAGE |  | Actual file content encoded as binary. |

### Custom Forms Module

This module consists in a single table that stores the information about the custom forms made specifically for a class/object collection.



Custom form module diagram

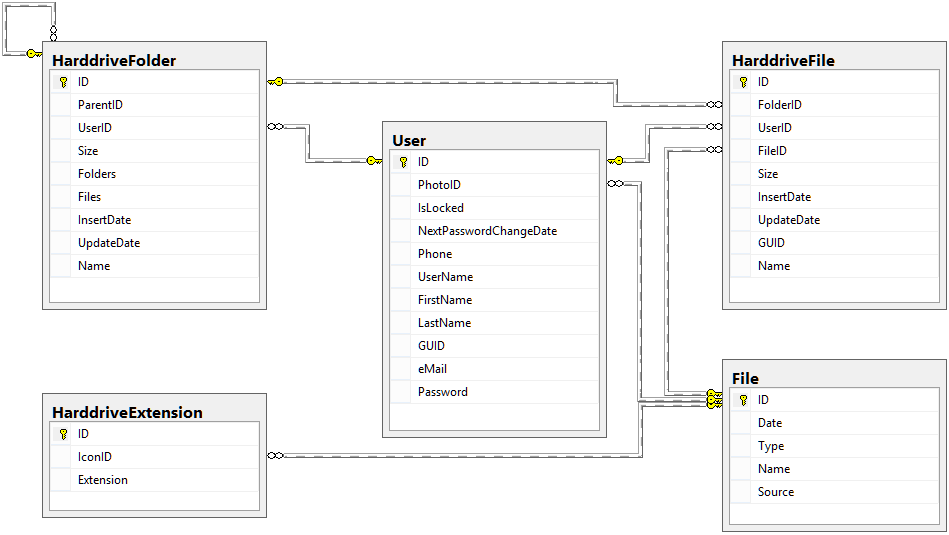
NOTE: For this module the Class table is an adjacent table so it will be detailed in its own module.

**Form**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ClassID | INT |  | Foreign key for the Class table. |
| Name | VARCHAR(128) |  | The name of the custom form needed for development usage only. |
| Look | VARCHAR(MAX) |  | A specific HTML that determines how the custom form looks and what data should it load. |

### Virtual Drive Module

This module gives the users the possibility to keep files organized in folder on the platform itself. Thus, enhances the platform with a small virtual drive capability. The files stored through this module may be integrated in documentation articles by simply referencing them through a unique identifier.



Virtual drive module diagram

NOTE: For this module the File and User tables are adjacent tables so they will be detailed in their own modules.

**HarddriveExtension**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| IconID | INT |  | Foreign key for the File table. |
| Extension | VARCHAR(8) |  | The file extension for which the Icon referenced is for, such as “.png” or “.pdf”. |

**HarddriveFile**

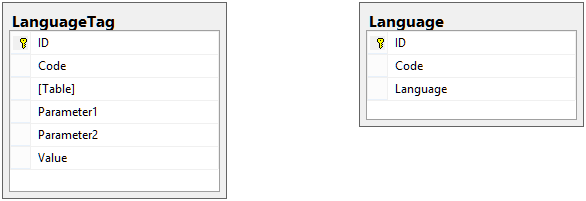
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| FolderID | INT |  | Foreign key for the HarddriveFolder table. |
| UserID | INT |  | Foreign key for the User table. |
| FileID | INT |  | Foreign key for the File table. |
| Size | INT |  | The dimension of the file in bytes. |
| InsertDate | DATETIME |  | The date when the file was created. |
| UpdateDate | DATETIME |  | The date when the file was last modified. |
| GUID | VARCHAR(64) |  | A GUID that uniquely identifies the file. |
| Name | NVARCHAR(256) |  | The name of the file, including the extension. |

**HarddriveFolder**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ParentID | INT |  | Foreign key for the HarddriveFolder table. Needed to in order to have a hierarchical folder structure. |
| UserID | INT |  | Foreign key for the User table. |
| Size | INT |  | The dimension of all the contained files and folders in bytes. |
| Folders |  |  | The number of subfolders. |
| Files |  |  | The number of files contained. |
| InsertDate | DATETIME |  | The date when the file was created. |
| UpdateDate | DATETIME |  | The date when the file was last modified. |
| Name |  |  |  |

### Translation Module

In order to make the platform support multiple languages we need a common module that integrates with all the other modules in order to cover translations. For this purpose we have this module consisting in only two tables, one to hold a list with the supported languages and one to hold the translations for all the texts needed.



Language translation module diagram

**Language**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Code | VARCHAR(8) |  | A code that uniquely identifies the language, such as “EN-UK” or “RO-RO”. |
| Language | NVARCHAR(32) |  | The name of the language. |

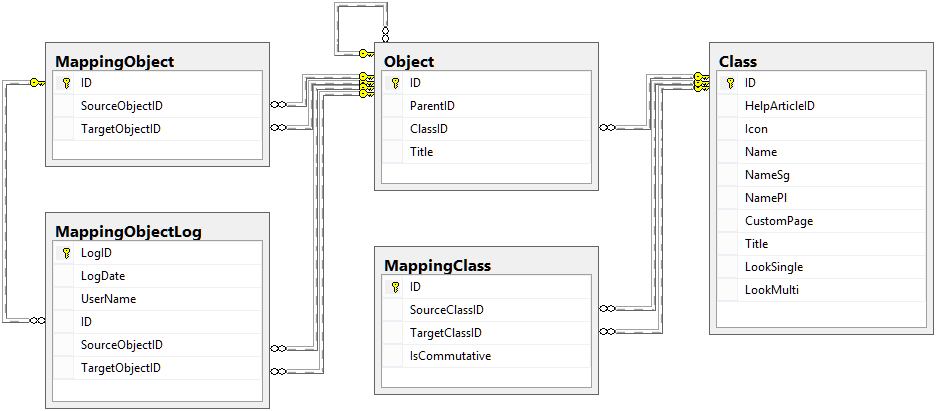
**LanguageTag**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Code | VARCHAR(8) |  | A code that uniquely identifies the language, such as “EN-UK” or “RO-RO”. |
| Table | VARCHAR(64) |  | The name of the table that has a key value that needs to be translated. |
| Parameter1 | VARCHAR(64) |  | A column used to identify the key that needs to be translated. |
| Parameter2 | VARCHAR(64) |  | A column used to identify the key that needs to be translated. |
| Value | NVARCHAR(256) |  | The translation for the key itself. |

### Mappings Module

This module is part of the platforms core. Due to the way in which the data is stored inside the platform and due to the cases that may appear this is an essential module that integrates data as needed. The main goal of this module is to link different objects together in one direction only or both directions. Because the objects are linked in a hierarchical way we need this module in order to give the data a graph like structure where the objects are vertices and the hierarchical links and mapping links between objects are edges.

As a note, the mappings set between objects are logged at every change, which is why the MappingObject table has the corresponding MappingObjectLog table, while the MappingClass table defines what type of objects may be mapped and if they can be one-way or both-ways.

****

Mappings module diagram

NOTE: For this module the Class and Object tables are adjacent tables so they will be detailed in their own modules.

**MappingClass**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| SourceClassID | INT |  | Foreign key for the Class table. References the source class for the mapping. |
| TargetClassID | INT |  | Foreign key for the Class table. References the target class for the mapping. |
| IsCommutative | INT |  | Used as a Boolean to determine if the mapping is from the source class to the target class or if it is both ways. |

**MappingObject**

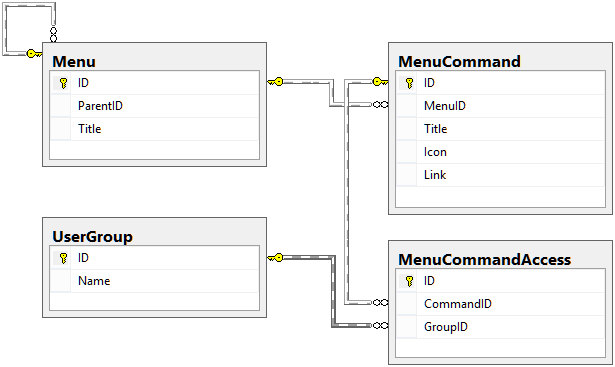
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| SourceClassID | INT |  | Foreign key for the Object table. References the source object for the mapping. |
| TargetClassID | INT |  | Foreign key for the Object table. References the target object for the mapping. |

**MappingObjectLog**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| LogID | INT |  | Primary key |
| LogDate | DATETIME |  | The date when the mapping was created/modified. |
| UserName | NVARCHAR(128) |  | The user’s full name of the user who triggered the creation/modification. |
| ID | INT |  | Foreign key for the MappingObject table. |
| SourceClassID | INT |  | Foreign key for the Object table. References the source object for the mapping. |
| TargetClassID | INT |  | Foreign key for the Object table. References the target object for the mapping. |

### Menu Module

This module is mainly dedicated to the taskbar displaying, so it contains information about the menu hierarchy, menu commands and the groups’ access on commands.



Menu module diagram

NOTE: For this module the UserGroup table is an adjacent table so it will be detailed in its own module.

**Menu**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ParentID | INT |  | Foreign key for the Menu table. This is in place because the menu has a hierarchical tree structure. |
| Title | VARCHAR(128) |  | The menu title. |

**MenuCommand**

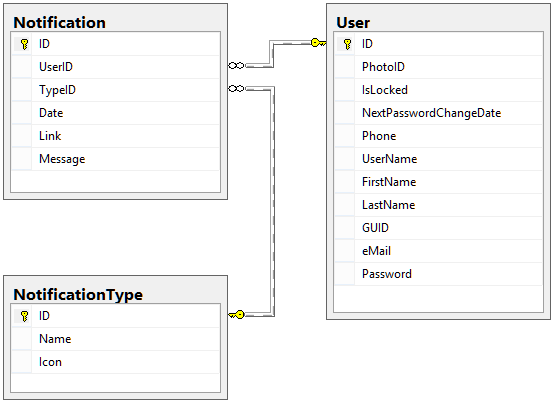
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| MenuID | INT |  | Foreign key for the Menu table. |
| Title | VARCHAR(128) |  | The command title. |
| Icon | VARCHAR(256) |  | The relative web path where the icon associated with the command can be found. |
| Link | VARCHAR(512) |  | The http link that must be opened in a window when the command is triggered. |

**MenuCommandAccess**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| CommandID | INT |  | Foreign key for the MenuCommand table. |
| GroupID | INT |  | Foreign key for the UserGroup table. |

### Notifications Module

This module holds the information regarding the notifications for the users.



Notifications module diagram

NOTE: For this module the User table is an adjacent table so it will be detailed in its own module.

**Notification**

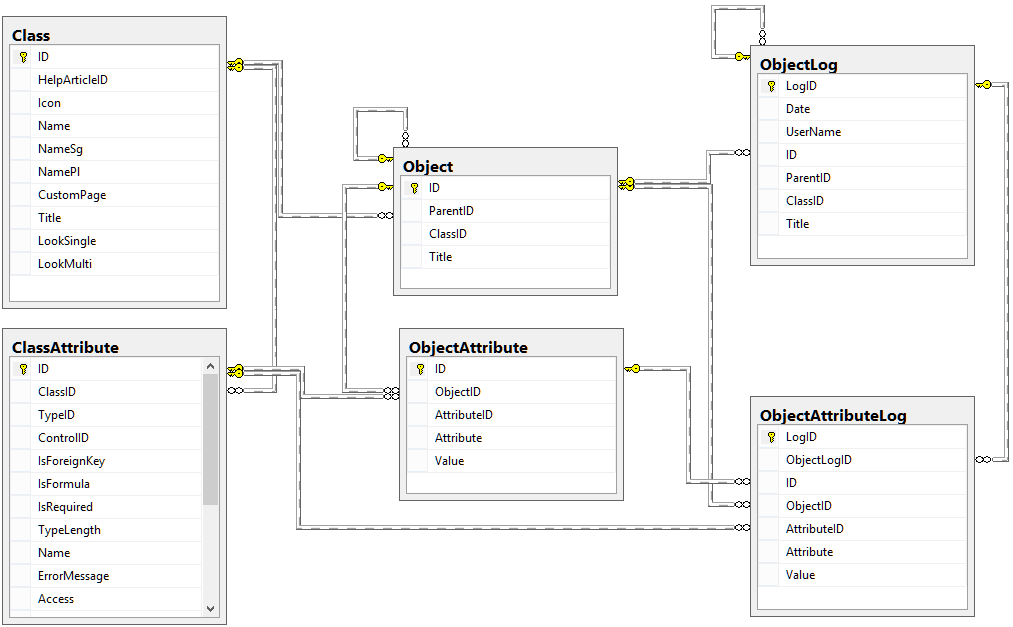
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| UserID | INT |  | Foreign key for the User table. |
| TypeID | INT |  | Foreign key for the NotificationType table. |
| Date | DATETIME |  | The date when the notification was issued to the user. |
| Link | VARCHAR(256) |  | An http link that is relevant to the notification, if any. |
| Message | NVARCHAR(MAX) |  | The message detailing the notification. |

**NotificationType**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(128) |  | The name for the notification type. |
| Icon | VARCHAR(128) |  | The http relative path where the picture associated with the notification type can be found. |

### Objects Module

This module represents a big part of the platforms core and it is responsible for holding the data inserted into the platform. Also, it is another module that has logs attached to its tables due to the nature of its purpose.



Objects module diagram

NOTE: For this module the Class and ClassAttribute tables are adjacent tables so they will be detailed in their own modules.

**Object**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ParentID | INT |  | Foreign key for the Object table. This gives the Object table the hierarchical data structure that we aim for. |
| ClassID | INT |  | Foreign key for the Class table. |
| Title | NVARCHAR(4000) |  | The processed object title tokened with language translation tags if necessary. |

**ObjectAttribute**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ObjectID | INT |  | Foreign key for the Object table. |
| AttributeID | INT |  | Foreign key for the ClassAttribute table. |
| Attribute | VARCHAR(128) |  | The name of the attribute referred. This is a redundancy that is in place to shorten a JOIN in SQL when developing custom stored procedures or user defined functions. |
| Value | NVARCHAR(MAX) |  | The actual value for the object’s attribute. |

**ObjectAttributeLog**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| LogID | INT |  | Primary key |
| ObjectLogID | INT |  | Foreign key for the ObjectLog table. |
| ID | INT |  | Foreign key for the ObjectAttribute table. |
| ObjectID | INT |  | Foreign key for the Object table. |
| AttributeID | INT |  | Foreign key for the ClassAttribute table. |
| Attribute | VARCHAR(128) |  | The name of the attribute referred. This is a redundancy that is in place to shorten a JOIN in SQL when developing custom stored procedures or user defined functions. |
| Value | NVARCHAR(MAX) |  | The actual value for the object’s attribute. |

**ObjectLog**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| LogID | INT |  | Primary key |
| LogDate | DATETIME |  | The date when the object was created/modified. |
| UserName | NVARCHAR(128) |  | The user’s full name of the user who triggered the creation/modification. |
| ID | INT |  | Foreign key for the Object table. |
| ParentID | INT |  | Foreign key for the Object table. This gives the Object table the hierarchical data structure that we aim for. |
| ClassID | INT |  | Foreign key for the Class table. |
| Title | NVARCHAR(4000) |  | The processed object title tokened with language translation tags if necessary. |

### Page Module

This module keeps a reference of the restricted web pages displayed by the user interface and consists in only one table.

**Page**

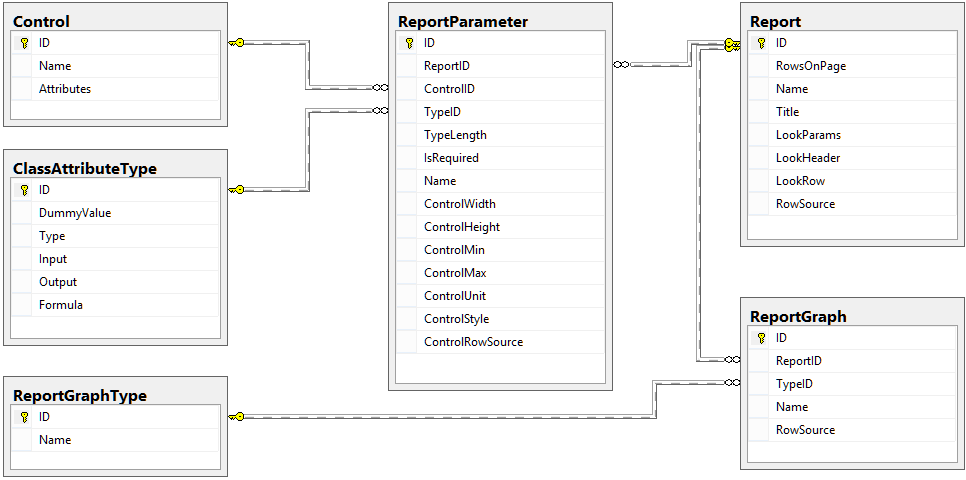
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(128) |  | The file name of the web page, such as “Login.aspx”. |

### Reference Guide Module

This is a pseudo-module and its data comes from other modules but we grouped some procedures into this module because they are out of the scope of others.

### Reporting Module

Given a document/form platform that collects and links data, the requirement for the platform will be for a reporting module. In our case, this module keeps the metadata regarding the reports and graphics that the platform can display.



Reporting module diagram

NOTE: For this module the ClassAttributeType and Control tables are adjacent tables so they will be detailed in their own modules.

**Report**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| RowsOnPage | INT |  | The maximum number of rows that can be displayed on a single result page. |
| Name | VARCHAR(128) |  | The name of the report used in development. |
| Title | NARCHAR(1024) |  | A tokened SQL expression that computes the title for the report. |
| LookParams | NVARCHAR(MAX) |  | A specific HTML that describes the form that collects the parameters for the report. |
| LookHeader | NVARCHAR(MAX) |  | A specific HTML that describes the report’s header. If this header description is missing then the report is considered to be free, else it is considered to be a strict report. |
| LookRow | NVARCHAR(MAX) |  | A specific HTML that describes the report’s row. If this is missing the report will be considered free, else it considered being a strict report. |
| RowSource | NVARCHAR(MAX) |  | A specific SQL statement that retrieves the data for the report. |

**ReportGraph**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ReportID | INT |  | Foreign key for the Report table. |
| TypeID | INT |  | Foreign key for the ReportGraphType table. |
| Name | VARCHAR(64) |  | The name of the graph used in development. |
| RowSource | NVARCHAR(MAX) |  | A specific SQL statement that retrieves the data for the graph. |

**ReportGraphType**

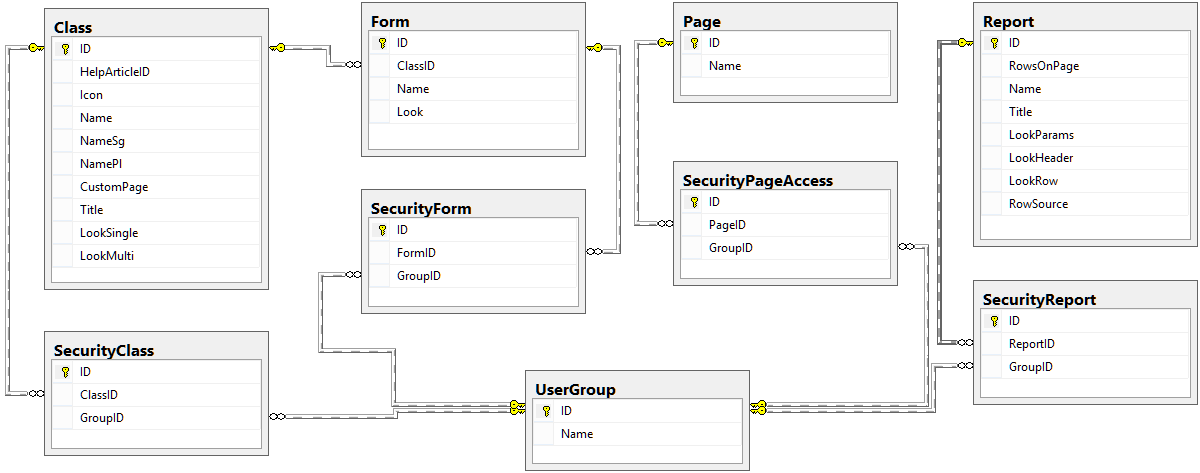
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(32) |  | The name for the graph type. |

**ReportParameter**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ReportID | INT |  | The name for the graph type. |
| ControlID | INT |  | Foreign key for the Control table. |
| TypeID | INT |  | Foreign key for the ClassAttributeType table. |
| TypeLength | INT |  | For certain parameter types this field is required to determine the maximum length for the value. Sample: NVARCHAR type needs a length between 1 and 4000. |
| IsRequired | INT |  | Used as a Boolean to determine if the parameter is required or not. |
| Name | VARCHAR(128) |  | The parameter name used in the development process. |
| ControlWidth | INT |  | Depending on the control for this attribute this specifies the display width of the control. |
| ControlHeight | INT |  | Depending on the control for this attribute this specifies the display height of the control. |
| ControlMin | FLOAT |  | Depending on the control for this attribute this specifies the minimum value for the attribute value. |
| ControlMax | FLOAT |  | Depending on the control for this attribute this specifies the maximum value for the attribute value. |
| ControlUnit | FLOAT |  | Depending on the control for this attribute this specifies the fixed increment unit for the attribute value. |
| ControlStyle | VARCHAR (512) |  | Depending on the control for this attribute this specifies the HTML style for the displayed control. |
| ControlRowSource | VARCHAR (MAX) |  | Depending on the control for this attribute this is a specific SQL query that returns the data for the control. |

### Security Module

As expected from each platform of this kind, the platform has a security module that determines which user can access what. In our case this module covers access to the Classes/Objects, Reporting, Custom forms and Page modules.



Security module diagram

NOTE: For this module the Class, Form, Page, Report and UserGroup tables are adjacent tables so they will be detailed in their own modules.

**SecurityClass**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ClassID | INT |  | Foreign key for the Class table. |
| GroupID | INT |  | Foreign key for the UserGroup table. |

**SecurityForm**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| FormID | INT |  | Foreign key for the Form table. |
| GroupID | INT |  | Foreign key for the UserGroup table. |

**SecurityPageAccess**

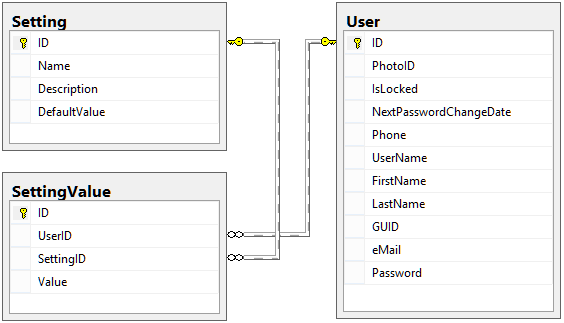
|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| PageID | INT |  | Foreign key for the Page table. |
| GroupID | INT |  | Foreign key for the UserGroup table. |

**SecurityReport**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| ReportID | INT |  | Foreign key for the Report table. |
| GroupID | INT |  | Foreign key for the UserGroup table. |

### Settings Module

This is a module designed to hold personal setting for the user.

****

Settings module diagram

NOTE: For this module the User table is an adjacent table so it will be detailed in its own module.

**Setting**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Name | VARCHAR(64) |  | The setting’s name. |
| Description | VARCHAR(8000) |  | A description for what the setting represents. |
| DefaultValue | VARCHAR(8000) |  | The default value for the setting. |

**SettingValue**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| UserID | INT |  | Foreign key for the User table |
| SettingID | INT |  | Foreign key for the Setting table. |
| Value | VARCHAR(128) |  | The value for the setting and user pair. |

### DataBase Module

This module contains a single table and its main purpose is to keep entries for the first primary key value available for the tables that do not have auto-incremental primary keys.

**DataBaseCounter**

|  |  |  |  |
| --- | --- | --- | --- |
| Column | Data Type  (Length) | Allows NULL | Description |
| ID | INT |  | Primary key |
| Index | INT |  | First available primary key value specified by the counter column. |
| Counter | VARCHAR(32) |  | The name of the database table. |

## User Defined Functions and Stored Procedures Grouped by Module

### Documentation Module

The general prefix for the tables associated with this module is “Article” and the stored procedures’ prefix is “Article\_”.

**Article\_DeleteArticle (SP[[6]](#footnote-6))**

Purpose: deletes an article from the database along with its comments and tags and at the same time it removes the article reference from the ArticleLink table for the deleted article and resets the caches for the articles that refer the deleted article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_DeleteComment (SP)**

Purpose: deletes an article comment from the database along with the corresponding sub-comments.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @CommentID | INT | References the primary key from the ArticleComment table. |

**Article\_DeleteLink (SP)**

Purpose: deletes an expression from the database and resets the caches for the articles that contain the deleted expression.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @LinkID | INT | References the primary key from the ArticleLink table. |

**Article\_DeleteTagForArticle (SP)**

Purpose: deletes a tag for an article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @TagID | INT | References the primary key from the ArticleTag table. |

**Article\_GetArticleCache (SP)**

Purpose: retrieves some information regarding an article that is needed to display the article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_GetArticleInfo (SP)**

Purpose: retrieves the general information regarding an article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_GetArticlesForTreeID (SP)**

Purpose: retrieves the general information regarding all the articles linked to a tree node from the ArticleTree table.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @TreeID | INT | References the primary key from the ArticleTree table. |
| @Language | VARCHAR(8) | The current interface language code needed to provide only the articles corresponding to that language alone. |

**Article\_GetCommentsForArticle (SP)**

Purpose: retrieves all the comments for the article specified ordered by date and organized in the hierarchy given by the ID – ParentID relation.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_GetFilesForArticle (SP)**

Purpose: retrieves all the virtual drive files referenced by the specified article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_GetLinks (SP)**

Purpose: retrieves all the expressions with all the data regarding them.

Parameters: NONE

**Article\_GetLinksForArticle (SP)**

Purpose: retrieves all the expression data contained by the specified article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_GetNewestArticles (SP)**

Purpose: retrieves the latest 3 articles written in the language specified by code.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The current interface language code needed to provide only the articles corresponding to that language alone. |

**Article\_GetSimilarArticles (SP)**

Purpose: retrieves the top 20 articles that are similar with the specified article and written in the same language. The articles are determined to be similar by the tags and expressions that the articles share with the given article.

NOTE: The specified article is ignored by this search.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |
| @Language | VARCHAR(8) | The current interface language code needed to provide only the articles corresponding to that language alone. |

**Article\_GetTagsForArticle (SP)**

Purpose: retrieves all the tag that represents the specified article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |

**Article\_GetTopTags (SP)**

Purpose: retrieves the top 15 tags depending on number of articles that link to them.

Parameters: NONE

**Article\_GetTree (SP)**

Purpose: Retrieves the tree structure that groups articles together. The structure itself is retrieved as a hierarchical menu.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The current interface language code needed to provide only the articles corresponding to that language alone. |

**Article\_GetTreeNodeInfo (SP)**

Purpose: retrieves the information regarding a single article tree node.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @TreeID | INT | References the primary key from the ArticleTree table. |
| @Language | VARCHAR(8) | The current interface language code needed to provide only the articles corresponding to that language alone. |

**Article\_InsertArticle (SP)**

Purpose: inserts a new article in the database and returns the primary key value for the new article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. This represents the author of the article and can’t be updated from the interface later on. Also this will reference the primary key from the ArticleAuthor table. |
| @Title | NVARCHAR(128) | The article’s title. |
| @Content | NVARCHAR(MAX) | The article’s content written in a specific HTML format. |
| @Language | VARCHAR(8) | The code for the language in which the article was written. |

**Article\_InsertComment (SP)**

Purpose: inserts a new comment for the specified article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ParentID | INT | References the primary key from the ArticleComment table. This parameter may be NULL in case the comment refers only to the article and it is not a response for another comment. |
| @ArticleID | INT | References the primary key from the Article table. |
| @UserID | INT | References the primary key from the User table. This represents the author of the article and can’t be updated from the interface later on. Also this will reference the primary key from the ArticleAuthor table. |
| @Comment | NVARCHAR(MAX) | The comment text written in pure HTML format. |

**Article\_InsertLink (SP)**

Purpose: inserts a new expression and its definition.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. The article referenced details the expression and may be passed on as NULL. |
| @Expression | NVARCHAR(512) | The expression itself. |
| @Description | NVARCHAR(4000) | The definition or description for the expression. |

**Article\_InsertTagForArticle (SP)**

Purpose: inserts a new tag for the specified article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. |
| @Tag | NVARCHAR(128) | The tag itself. |

**Article\_Search (SP)**

Purpose: retrieves a list of articles that contain all the words passed as parameters of part of the words ordered by weights. The weight is determined by the number of words found and by the place where they were found. For example, an article is more relevant for the search criteria if the words are found in the article’s title rather than its content.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Words | NVARCHAR(4000) | The search criteria. |
| @Language | VARCHAR(8) | The current interface language code needed to provide only the articles corresponding to that language alone. |

**Article\_UpdateArticle (SP)**

Purpose: updates a database existing article.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. This represents the article that needs to be updated. |
| @Title | NVARCHAR(128) | The article’s title. |
| @Content | NVARCHAR(MAX) | The article’s content written in a specific HTML format. |
| @Language | VARCHAR(8) | The code for the language in which the article was written. |

**Article\_UpdateArticleCache (SP)**

Purpose: updates an articles content cache and navigation cache.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ArticleID | INT | References the primary key from the Article table. This represents the article that needs to be updated. |
| @ContentCache | NVARCHAR(MAX) | The article’s content cache written in pure HTML. |
| @NavigationCache | NVARCHAR(MAX) | The article’s navigation cache written in pure HTML. |

**Article\_UpdateLink (SP)**

Purpose: updates an expression’s definition.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @LinkID | INT | References the primary key from the ArticleLink table. This represents the expression that needs to be updated. |
| @ArticleID | INT | References the primary key from the Article table. The article referenced details the expression and may be passed on as NULL. |
| @Expression | NVARCHAR(512) | The expression itself. |
| @Description | NVARCHAR(4000) | The definition or description for the expression. |

### Cache Module

The general prefix for the tables associated with this module is “CacheTable” and the stored procedures’ prefix is “Cache\_”.

This module has no actual tables but it generates tables based on the need. For each class of objects it generates a table containing the information regarding the object from the targeted class. If we ignore all the platform tables, the cache tables form a basic relational database between all the stored objects.

For each cache table generated for a class this module generates an associated log table that keeps all the objects’ changes for that class.

**Cache\_CreateTablesForClass (SP)**

Purpose: creates the cache tables for the object from the specified class and for their logs, also it creates all the needed foreign key constraints from the created cache table to the other cache tables and vice versa.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Cache\_DeleteObject (SP)**

Purpose: deletes the specified entry from the cache table specific to the referred object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Cache\_DropTablesForClass (SP)**

Purpose: deletes the cache tables for the object from the specified class and for their logs, also it deletes all the foreign key constraints from the cache table to the other cache tables and vice versa.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Cache\_InsertObject (SP)**

Purpose: computes the cache row for an object and inserts it in the proper cache table.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Cache\_ProcessClass (SP)**

Purpose: drops the cache table associated to a class and recreates the cache table including the log.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Cache\_UpdateObject (SP)**

Purpose: computes the cache row for an object and updates it in the proper cache table.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

### Class Module

The general prefix for the tables associated with this module is “Class” and the stored procedures’ prefix is “Class\_”.

**Class\_DeleteAttribute (SP)**

Purpose: deletes the specified attribute and resets the cache for the corresponding cache, also it deletes the data for the specified attribute from the objects pool.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ClassAttribute table. |

**Class\_DeleteClass (SP)**

Purpose: deletes the specified class and the cache for it, also it deletes all the objects belonging to the class, mapping and class link entries that refer the specified class and language tags for the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the Class table. |

**Class\_DeleteLink (SP)**

Purpose: deletes the specified class link.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ClassLink table. |

**Class\_ExistsAttribute (SP)**

Purpose: checks if the referenced class already contains an attribute with the given name.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @Name | VARCHAR(128) | The name of the attribute. |
| @ExceptID | INT | References the primary key from the ClassAttribute table. If an attribute is modified it has to avoid that attribute when checking for duplicate attribute name. |

**Class\_ExistsClass (SP)**

Purpose: checks if there is a class with the specified name.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the class. |
| @ExceptID | INT | References the primary key from the Class table. If class is modified it has to avoid that class when checking for duplicate class name. |

**Class\_GetAttributeInfo (SP)**

Purpose: retrieves all the information regarding the specified class attribute.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ClassLink table. |

**Class\_GetAttributesForClass (SP)**

Purpose: retrieves the list with the attributes for the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Class\_GetAttributeTypes (SP)**

Purpose: retrieves the list with all the attributes types.

Parameters: NONE

**Class\_GetClasses (SP)**

Purpose: retrieves the list with all the classes.

Parameters: NONE

**Class\_GetClassesExt (SP)**

Purpose: retrieves the list with all the classes and an extra column that indicates which are the classes on which the user has access to.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**Class\_GetClassesForUser (SP)**

Purpose: retrieves the list with all the classes on which the specified user has access to.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**Class\_GetClassIDFromName (SP)**

Purpose: retrieves a class’s ID from the class’s name.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The class’s name. |

**Class\_GetClassInfo (SP)**

Purpose: retrieves the class information for the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @Language | VARCHAR(8) | The language code in which the interface is displayed. This is needed to return the information about the class translated. |

**Class\_GetLanguageTagsForClass (SP)**

Purpose: retrieves all the translation tags for the specified class along with the language code.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @Language | VARCHAR(8) | The language code in which the interface is displayed. This is needed to return the information about the class translated. |

**Class\_GetMasterClassesForClass (SP)**

Purpose: retrieves the list of classes that are master classes for the specified class according to the relation defined in the ClassLink table.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Class\_GetSlaveClassesForClass (SP)**

Purpose: retrieves the list of classes that are slave classes for the specified class according to the relation defined in the ClassLink table.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Class\_GetSlaveClassesForClassAndUser (SP)**

Purpose: retrieves the list of classes that are slave classes for the specified class according to the relation defined in the ClassLink table and on which the specified user has access.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @UserID | INT | References the primary key from the User table. |
| @Language | VARCHAR(8) | The language code in which the interface is displayed in order to retrieve the class names translated. |

**Class\_InsertAttribute (SP)**

Purpose: inserts a new attribute for the specified class and resets the cache for that class, also it adds the attribute to all the objects of that class and fills them with the NULL value.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @Type | VARCHAR(64) | The name of the data type. |
| @Control | VARCHAR(128) | The name of the control through which the value of the attribute will be collected. |
| @IsForeignKey | INT | Used as a Boolean to indicate that the attribute’s value will reference another object. |
| @IsFormula | INT | Used as a Boolean to indicate that the attribute will be computed based on a formula when the object is saved. |
| @IsRequired | INT | Used as a Boolean to indicate that the attribute requires a value and can’t be left blank. |
| @TypeLenght | VARCHAR(MAX) | Represents the maximum length for the attribute’s value. It is required for some attribute data type such as NVARCHAR (this must have a length between 1 and 4000 or MAX). |
| @Name | VARCHAR(128) | The name of the attribute. |
| @ErrorMessage | VARCHAR(256) | A tokened string that is thrown as an exception when the attribute value is left blank and the attribute is required. |
| @Access | VARCHAR(MAX) | A specific SQL expression that must return ‘WRITE’ or ‘READ’ depending on the user’s access level or the object’s workflow. |
| @DefaultValue | VARCHAR(MAX) | A specific SQL expression that returns the attribute’s default value or gives the formula used to compute its value in case the @IsFormula parameter is set to true. |
| @ControlWidth | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlHeight | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlMin | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlMax | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlUnit | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlStyle | VARCHAR(512) | For some controls this is an optional parameter that defines the control. |
| @ControlRowSource | VARCHAR(MAX) | For some controls this is a required parameter that defines the control. |

**Class\_InsertClass (SP)**

Purpose: inserts a new class, prepares the cache for the class and returns the primary key value for the new class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Icon | VARCHAR(256) | The relative web path where the icon associated with the new class will be found. |
| @Name | VARCHAR(128) | The name of the class used in development. |
| @NameSg | VARCHAR(128) | The name of the class in singular form used for the interface. This value is considered a language tag and will be translated when displayed to the user. |
| @NamePl | VARCHAR(128) | The name of the class in plural form used for the interface. This value is considered a language tag and will be translated when displayed to the user. |
| @ClassID | INT | References the primary key from the Class table. The new class will be cloned after the class with this primary key, if specified. |

**Class\_InsertLink (SP)**

Purpose: inserts a new master-slave class relation.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @MasteClassID | INT | References the primary key from the Class table. |
| @SlaveClassID | INT | References the primary key from the Class table. |

**Class\_NormalizeQuery (SP)**

Purpose: this is a special stored procedure that parses specialized SQL statement to replace the tokens referencing classes. Also, keep in mind that this procedure restricts access to objects depending on the user for which the specific SQL query is processed for.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Cmd | NVARCHAR(MAX) | This is the specialized SQL query and it is also an OUTPUT parameter, meaning that the processed SQL query will be returned through this parameter. |
| @UserID | INT | References the primary key from the User table. |

**Class\_NormalizeQueryAtDate (SP)**

Purpose: this is a special stored procedure that parses specialized SQL statement to replace the tokens referencing classes. Also, keep in mind that this procedure restricts access to objects depending on the user for which the specific SQL query is processed for and it gets all the data that was available in a specific moment in time in the same form in which they were at that moment in time.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Cmd | NVARCHAR(MAX) | This is the specialized SQL query and it is also an OUTPUT parameter, meaning that the processed SQL query will be returned through this parameter. |
| @UserID | INT | References the primary key from the User table. |
| @Date | VARCHAR(32) | The date representing the point in time in the “DD/MM/YYYY HH:MM” format. |

**Class\_UpdateAttribute (SP)**

Purpose: updates the specified attribute and resets the cache for the corresponding class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ClassAttribute table. |
| @Type | VARCHAR(64) | The name of the data type. |
| @Control | VARCHAR(128) | The name of the control through which the value of the attribute will be collected. |
| @IsForeignKey | INT | Used as a Boolean to indicate that the attribute’s value will reference another object. |
| @IsFormula | INT | Used as a Boolean to indicate that the attribute will be computed based on a formula when the object is saved. |
| @IsRequired | INT | Used as a Boolean to indicate that the attribute requires a value and can’t be left blank. |
| @TypeLenght | VARCHAR(MAX) | Represents the maximum length for the attribute’s value. It is required for some attribute data type such as NVARCHAR (this must have a length between 1 and 4000 or MAX). |
| @Name | VARCHAR(128) | The name of the attribute. |
| @ErrorMessage | VARCHAR(256) | A tokened string that is thrown as an exception when the attribute value is left blank and the attribute is required. |
| @Access | VARCHAR(MAX) | A specific SQL expression that must return ‘WRITE’ or ‘READ’ depending on the user’s access level or the object’s workflow. |
| @DefaultValue | VARCHAR(MAX) | A specific SQL expression that returns the attribute’s default value or gives the formula used to compute its value in case the @IsFormula parameter is set to true. |
| @ControlWidth | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlHeight | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlMin | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlMax | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlUnit | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlStyle | VARCHAR(512) | For some controls this is an optional parameter that defines the control. |
| @ControlRowSource | VARCHAR(MAX) | For some controls this is a required parameter that defines the control. |

**Class\_UpdateClass (SP)**

Purpose: update the specified class and resets the cache for the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the Class table. |
| @HelpArticleID | INT | References the primary key from the Article table. Points to the associated help article for the class, if any. |
| @Icon | VARCHAR(256) | The relative web path where the icon associated with the new class will be found. |
| @Name | VARCHAR(128) | The name of the class used in development. |
| @NameSg | VARCHAR(128) | The name of the class in singular form used for the interface. This value is considered a language tag and will be translated when displayed to the user. |
| @NamePl | VARCHAR(128) | The name of the class in plural form used for the interface. This value is considered a language tag and will be translated when displayed to the user. |
| @CustomPage | VARCHAR(256) | The name of the custom page that will display objects from the specified class, if any. |
| @Title | VARCHAR(8000) | A specific tokened SQL expression that computes the title for all the objects of the specified class. |
| @LookSingle | VARCHAR(MAX) | A specific tokened HTML that triggers the way in which a data collecting form for an object of the specified class will look like. |
| @LookMulti | VARCHAR(MAX) | A specific tokened HTML that triggers the way in which a listing for all the objects in the specified class looks like. |

**Class\_ValidateAttributeChange (SP)**

Purpose: Validates an attribute change from the data type point of view. For example, if we change an attribute from an INT to a NVARCHAR data type its ok, but if we want to change an attribute from a NVARCHAR to an INT data type that might not be ok because the attribute may contain values that can’t be converted to INT, such as the text ‘aaa’.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @AttributeID | INT | References the primary key from the ClassAttribute table. |
| @Type | VARCHAR(128) | The name of the new data type for the specified attribute. |
| @TypeLength | VARCHAR(8) | For some data types a maximum value length is required, such as for the NVARCHAR data type, the type length must be a value between 1 and 4000 or MAX. |

### Constants Module

The general prefix for the tables associated with this module is “Constant” and the stored procedures’ prefix is “Constant\_”.

**Constant\_GetValue (UDF[[7]](#footnote-7))**

Purpose: returns the value for the specified constant through its name.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Constant | VARCHAR(64) | The name of the constant itself. |

**Constant\_GetValueByName (SP)**

Purpose: returns the value for the specified constant through its name.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Constant | VARCHAR(64) | The name of the constant itself. |

**Constant\_Update (SP)**

Purpose: updates the value for the specified constant.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Constant | VARCHAR(64) | The name of the constant itself. |
| @Value | NVARCHAR(64) | The new value for the constant. |

### DataBase Module

The general prefix for the tables associated with this module is “DataBase” and the stored procedures’ prefix is “DataBase\_”.

This module is mainly used by the data layer. Some of the tables do not have auto-incremental primary keys. This module holds the first available values for those tables and in the case of a data insert on those tables this module is called to provide the primary key value.

**DataBase\_GetID (SP)**

In the case of multiple primary keys reservation this procedure returns the first available primary key value but reserves all needed. The primary key value returned is returned as an error code, not in the usual manner (meaning with a select statement).

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Counter | VARCHAR(64) | The name of the table. |
| @Count | INT | The number of primary keys that need to be reserved. The default value for this parameter is 1. |

### Desktop Module

The general prefix for the tables associated with this module is “Desktop” and the stored procedures’ prefix is “Desktop\_”.

The following procedures are used to manage data regarding user windows and desktop items as well as the default icons for a group of users. If a user is assigned to a group, then his desktop will be populated with the default icons linked to that group.

**Desktop\_DeleteDefaultIcon (SP)**

Purpose: deletes the specified desktop default icon assign to a group of users.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @IconID | INT | References the primary key from the DesktopDefaultIcon table. |

**Desktop\_DeleteIcon (SP)**

Purpose: deletes the specified desktop icon assign to a user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @IconID | INT | References the primary key from the DesktopIcon table. |

**Desktop\_DeleteWindow (SP)**

Purpose: deletes the specified window assign to a user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @WindowID | INT | References the primary key from the DesktopWindow table. |

**Desktop\_GetDefaultIconsForGroup (SP)**

Purpose: retrieves the information regarding all the default icons linked to a user group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Desktop\_GetIconInfo (SP)**

Purpose: retrieves the information regarding the specified icon.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @IconID | INT | References the primary key from the DesktopIcon table. |

**Desktop\_GetIconsForUser (SP)**

Purpose: retrieves the information regarding all the icons belonging to the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**Desktop\_GetWindowsForUser (SP)**

Purpose: retrieves the information regarding all the windows belonging to the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**Desktop\_InsertDefaultIcon (SP)**

Purpose: inserts a default icon for a group of users and inserts an equivalent icon for each of the users that belong to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |
| @PosX | INT | Specifies the horizontal icon’s position in pixels. |
| @PosY | INT | Specifies the vertical icon’s position in pixels. |
| @Icon | VARCHAR(128) | The relative web path where the pictogram associated with this icon can be found. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Link | NVARCHAR(512) | The http path of the web page that must be opened when the icon is triggered. |

**Desktop\_InsertIcon (SP)**

Purpose: inserts an icon for a user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @PosX | INT | Specifies the horizontal icon’s position in pixels. |
| @PosY | INT | Specifies the vertical icon’s position in pixels. |
| @Icon | VARCHAR(128) | The relative web path where the pictogram associated with this icon can be found. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Link | NVARCHAR(512) | The http path of the web page that must be opened when the icon is triggered. |

**Desktop\_InsertWindow (SP)**

Purpose: inserts a window for a user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Icon | VARCHAR(128) | The relative web path where the pictogram associated with this icon can be found. |
| @Link | NVARCHAR(512) | The http path of the web page that must be opened when the icon is triggered. |
| @PosX | INT | Specifies the horizontal icon’s position in pixels. |
| @PosY | INT | Specifies the vertical icon’s position in pixels. |
| @Width | INT | The window’s width in pixels. |
| @Height | INT | The window’s height in pixels. |

**Desktop\_UpdateDefaultIcon (SP)**

Purpose: updates the specified default icon for a group of users and updates the equivalent icons for the users that belong to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @IconID | INT | References the primary key from the DesktopDefaultIcon table. |
| @PosX | INT | Specifies the horizontal icon’s position in pixels. |
| @PosY | INT | Specifies the vertical icon’s position in pixels. |
| @Icon | VARCHAR(128) | The relative web path where the pictogram associated with this icon can be found. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Link | NVARCHAR(512) | The http path of the web page that must be opened when the icon is triggered. |

**Desktop\_UpdateIcon (SP)**

Purpose: updates an icon for a user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @IconID | INT | References the primary key from the DesktopIcon table. |
| @PosX | INT | Specifies the horizontal icon’s position in pixels. |
| @PosY | INT | Specifies the vertical icon’s position in pixels. |
| @Icon | VARCHAR(128) | The relative web path where the pictogram associated with this icon can be found. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Link | NVARCHAR(512) | The http path of the web page that must be opened when the icon is triggered. |

**Desktop\_UpdateWindow (SP)**

Purpose: updates a window for a user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @WindowID | INT | References the primary key from the DesktopWindow. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Icon | VARCHAR(128) | The relative web path where the pictogram associated with this icon can be found. |
| @Link | NVARCHAR(512) | The http path of the web page that must be opened when the icon is triggered. |
| @PosX | INT | Specifies the horizontal icon’s position in pixels. |
| @PosY | INT | Specifies the vertical icon’s position in pixels. |
| @Width | INT | The window’s width in pixels. |
| @Height | INT | The window’s height in pixels. |

### Events Module

The general prefix for the tables associated with this module is “Event” and the stored procedures’ prefix is “Event\_”.

**Event\_DeleteObject (SP)**

This procedure has the same parameters as the Object\_DeleteObject procedure but we won’t detail it because it is used just to validate specific SQL event queries.

**Event\_GetTriggersForClass (SP)**

Purpose: retrieves the specific SQL queries for all the triggers of the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Event\_InsertObject (SP)**

This procedure has the same parameters as the Object\_InsertObject procedure but we won’t detail it because it is used just to validate specific SQL event queries.

**Event\_Raise (SP)**

Purpose: computes and executes a specific SQL trigger query.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @ObjectID | INT | References the primary key from the Object table. |
| @ParentID | INT | References the primary key from the Object table. Represents the parent object of the object passed through the @ObjectID parameter. |
| @Attribute | NVARCHAR(MAX) | An XML structure that encapsulates all the attribute values for the object specified using the @ObjectID parameter. |
| @UserID | INT | References the primary key from the User table. |

**Event\_SetTrigger (SP)**

Purpose: updates the specific SQL query for a trigger and class and if it is missing it creates it.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @EventID | INT | References the primary key from the Event table. |
| @Command | NVARCHAR(MAX) | The specific SQL query that will be executed when the specified event is triggered for the specified class. |

**Event\_UpdateAttribute (SP)**

This procedure has the same parameters as the Object\_UpdateAttribute procedure but we won’t detail it because it is used just to validate specific SQL event queries.

**Event\_UpdateObject (SP)**

This procedure has the same parameters as the Object\_UpdateObject procedure but we won’t detail it because it is used just to validate specific SQL event queries.

### File Module

The general prefix for the tables associated with this module is “File” and the stored procedures’ prefix is “File\_”.

This module may trigger the database to grow and data to be scatter. In order to avoid over-crowding the data, the File table may be placed on another database and alter the stored procedures for this module to work with the moved table.

**File\_Delete (SP)**

Purpose: deletes the specified file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the File table. |

**File\_GetFileContent (SP)**

Purpose: retrieves the binary content for the specified file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the File table. |

**File\_GetFileIDs (SP)**

Purpose: retrieves all the primary key value from the File table.

Parameters: NONE

**File\_GetFileInfo (SP)**

Purpose: retrieves all the information regarding the specified file except the binary content of the file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the File table. |

**File\_Insert (SP)**

Purpose: inserts a new file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(1024) | The name of the file including the extension and a virtual path for it. |
| @Extension | VARCHAR(8) | The file’s extension, such as “.png” or “.pdf”. |
| @Source | IMAGE | The binary content of the file. |

**File\_Update (SP)**

Purpose: updates the specified file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the File table. |
| @Name | VARCHAR(1024) | The name of the file including the extension and a virtual path for it. |
| @Extension | VARCHAR(8) | The file’s extension, such as “.png” or “.pdf”. |
| @Source | IMAGE | The binary content of the file. |

### Custom Forms Module

The general prefix for the tables associated with this module is “Form” and the stored procedures’ prefix is “Form\_”.

**Form\_Delete (SP)**

Purpose: deletes the specified custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |

**Form\_ExecQuery (SP)**

Purpose: executes specific SQL queries found in the custom form’s description.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Query | NVARCHAR(MAX) | The specific SQL query that needs to be computed and executed. |
| @UserID | INT | References the primary key from the User table. Specifies the user in whose context is the query executed. |

**Form\_GetFormInfo (SP)**

Purpose: retrieves all the information regarding the specified form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |
| @Language | VARCHAR(8) | The language code in which the custom form must be displayed. Basically it will be the language code in which the interface is displayed. |

**Form\_GetForms (SP)**

Purpose: retrieves the list of all the custom forms translated in the language in which the interface is displayed.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The language code in which the interface is displayed. |

**Form\_GetLanguageTagsForForm (SP)**

Purpose: retrieves the list of all the language tags for the specified custom forms.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |

**Form\_Insert (SP)**

Purpose: inserts a new custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @Name | VARCHAR(128) | The form’s internal name used in development. |
| @Look | NVARCHAR(MAX) | A specific HTML description of the form. |

**Form\_Update (SP)**

Purpose: updates the specified custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |
| @ClassID | INT | References the primary key from the Class table. |
| @Name | VARCHAR(128) | The form’s internal name used in development. |
| @Look | NVARCHAR(MAX) | A specific HTML description of the form. |

### Virtual Drive Module

The general prefix for the tables associated with this module is “Harddrive” and the stored procedures’ prefix is “Harddrive\_”.

**Harddrive\_DeleteExtension (SP)**

Purpose: deletes the specified file extension and the file assign to it.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ExtensionID | INT | References the primary key from the HarddriveExtension table. |

**Harddrive\_DeleteFile (SP)**

Purpose: deletes the specified virtual drive file, the corresponding database file and updates the parent folder information.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the HarddriveFile table. |

**Harddrive\_DeleteFolder (SP)**

Purpose: deletes the specified virtual drive folder and updates the parent folder information.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the HarddriveFile table. |

**Harddrive\_GetExtensionInfo (SP)**

Purpose: retrieves all the information regarding the specified file extension.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Extension | VARCHAR(8) | The extension’s name. |

**Harddrive\_GetExtensions (SP)**

Purpose: retrieves all the information regarding all the file extensions.

Parameters: NONE

**Harddrive\_GetFileInfo (SP)**

Purpose: retrieves all the information regarding the specified virtual file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the HarddriveFile table. |

**Harddrive\_GetFiles (SP)**

Purpose: retrieves all the information regarding the virtual files from the specified virtual folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |

**Harddrive\_GetFolderInfo (SP)**

Purpose: retrieves all the information regarding the specified virtual folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |

**Harddrive\_GetFolderPath (UDF)**

Purpose: returns the virtual drive folder path for the specified folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |

**Harddrive\_GetFolders (SP)**

Purpose: retrieves all the information regarding the virtual folders sub-ordered to the specified virtual folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |

**Harddrive\_GetFolderTreeForUser (SP)**

Purpose: retrieves the virtual folders tree structure for the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**Harddrive\_GetHierarchyForFolder (SP)**

Purpose: retrieves the virtual folders tree structure for the specified virtual folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |

**Harddrive\_GetSize (UDF)**

Purpose: returns a size expressed in bytes into a shorter form containing the measure unit (e.g. KB, MB …).

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Size | INT | The size expressed in bytes. |

**Harddrive\_InsertExtension (SP)**

Purpose: inserts a new virtual file extension.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @IconID | INT | References the primary key from the File table. Represents the file that is associated with the extension. |
| @Extension | VARCHAR(8) | The extension itself. |

**Harddrive\_InsertFile (SP)**

Purpose: inserts a new virtual drive file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |
| @FileID | INT | References the primary key from the File table. |
| @Name | NVARCHAR(256) | The name of the file including the extension. |

**Harddrive\_InsertFolder (SP)**

Purpose: inserts a new virtual drive folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @ParentID | INT | References the primary key from the HarddriveFolder table. |
| @Name | NVARCHAR(256) | The name of the folder itself. |

**Harddrive\_Search (SP)**

Purpose: retrieves all the information regarding the virtual drive files and folders for the specified user that respect the provided search criteria.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Pattern | NVARCHAR(256) | The search criteria. |

**Harddrive\_UpdateFile (SP)**

Purpose: updates the specified virtual drive file.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FileID | INT | References the primary key from the HarddriveFile table. |
| @Name | NVARCHAR(256) | The name of the file including the extension. |

**Harddrive\_UpdateFolder (SP)**

Purpose: updates the specified virtual drive folder.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FolderID | INT | References the primary key from the HarddriveFolder table. |
| @Name | NVARCHAR(256) | The name of the folder itself. |

### Translation Module

The general prefix for the tables associated with this module is “Language” and the stored procedures’ prefix is “Language\_”.

**Language\_ExtractTags (TF[[8]](#footnote-8))**

Purpose: retrieves a table containing all the language tags found in the specified text.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Text | NVARCHAR(MAX) | The text that will be parsed for language tags. |

**Language\_GetLanguages (SP)**

Purpose: retrieves all the information regarding the languages supported by platform.

Parameters: NONE

**Language\_GetTags (SP)**

Purpose: retrieves all the language tags that correspond to the given criteria.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | A language code (e.g. “EN\_UK” or “RO-RO”). |
| @Table | VARCHAR(64) | The name of the table for which the tags are meant. |
| @Parameter1 | VARCHAR(64) | The first parameter that characterizes the language tags. |
| @Parameter2 | VARCHAR(64) | The second parameter that characterizes the language tags. |

**Language\_SetTag (SP)**

Purpose: updates the specified language tag and if it is missing it inserts it.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | A language code (e.g. “EN\_UK” or “RO-RO”). |
| @Table | VARCHAR(64) | The name of the table for which the tags are meant. |
| @Parameter1 | VARCHAR(64) | The first parameter that characterizes the language tags. |
| @Parameter2 | VARCHAR(64) | The second parameter that characterizes the language tags. |
| @Value | NVARCHAR(256) | The language tag value. |

### Mappings Module

The general prefix for the tables associated with this module is “Mapping” and the stored procedures’ prefix is “Mapping\_”.

**Mapping\_DeleteClassMapping (SP)**

Purpose: deletes the specified mapping relation between two classes and the object mappings established using the specified relation.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @MappingID | INT | References the primary key from the MappingClass table. |
| @UserID | INT | References the primary key from the User table. |

**Mapping\_DeleteObjectMapping (SP)**

Purpose: deletes the specified mapping relation between two objects.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @MappingID | INT | References the primary key from the MappingObject table. |
| @UserID | INT | References the primary key from the User table. |

**Mapping\_GetMappingInfo (SP)**

Purpose: retrieves the brief information regarding the objects linked by the specified mapping entry.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the MappingObject table. |

**Mapping\_GetMappingsForClass (SP)**

Purpose: retrieves the mapping relations that involve the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @OnlyFromClass | INT | Used as a Boolean to filter only the relations from the specified class only. |

**Mapping\_GetMappingsForClassAndUser (SP)**

Purpose: retrieves the mapping relations that involve the specified class and on which the specified user has access to.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @OnlyFromClass | INT | Used as a Boolean to filter only the relations from the specified class only. |
| @UserID | INT | References the primary key from the User table. |

**Mapping\_GetMappingsForObject (SP)**

Purpose: retrieves a detailed list with all the objects of the specified class mapped to the specified object on which the specified user has access.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @ObjectID | INT | References the primary key from the Object table. |
| @UserID | INT | References the primary key from the User table. |

**Mapping\_InsertClassMapping (SP)**

Purpose: inserts a mapping relation between the two specified classes.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @SourceClassID | INT | References the primary key from the Class table. Specifies the source class for the mapping relation. |
| @TargetClassID | INT | References the primary key from the Class table. Specifies the target class for the mapping relation. |
| @IsCommutative | INT | Used as a Boolean to specify if the relation is one-way or both-ways. |

**Mapping\_InsertObjectMapping (SP)**

Purpose: links two objects according to the mapping relation defined between their classes.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @SourceObjectID | INT | References the primary key from the Object table. Specifies the source object for the mapping. |
| @TargetObjectID | INT | References the primary key from the Object table. Specifies the target object for the mapping. |
| @UserID | INT | References the primary key from the User table. |

### Menu Module

The general prefix for the tables associated with this module is “Menu” and the stored procedures’ prefix is “Menu\_”.

**Menu\_DeleteAccess (SP)**

Purpose: deletes the access for a group to a menu item.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @CommandID | INT | References the primary key from the MenuCommand table. |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Menu\_DeleteCommand (SP)**

Purpose: deletes the specified menu command and the group access links for it, even the language tags associated to that menu command.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the MenuCommand table. |

**Menu\_DeleteMenu (SP)**

Purpose: deletes the specified menu and the sub-menus and commands sub-ordered to the specified menu, including all the language tags attached to the deleted entries.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the Menu table. |

**Menu\_GetCommandInfo (SP)**

Purpose: retrieves the full information for the specified menu command.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the MenuCommand table. |

**Menu\_GetMenuForGroup (SP)**

Purpose: retrieves the menu tree structure and commands for the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Menu\_GetMenuForGroups (SP)**

Purpose: retrieves the full menu tree structure (including commands) and a column for each defined user group that specifies the commands on which the users assigned to the group have access.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The code for the language in which the menu and command entries must be translated. |

**Menu\_GetMenuForUser (SP)**

Purpose: retrieves the full menu tree structure and commands for the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Language | VARCHAR(8) | The code for the language in which the menu and command entries must be translated. |

**Menu\_GetMenuInfo (SP)**

Purpose: retrieves all the information regarding the specified menu.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the Menu table. |

**Menu\_GetMenuTree (SP)**

Purpose: retrieves the full menu tree structure without commands.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The code for the language in which the menu must be translated. |

**Menu\_InsertAccess (SP)**

Purpose: inserts the access link for a group to a menu item.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @CommandID | INT | References the primary key from the MenuCommand table. |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Menu\_InsertCommand (SP)**

Purpose: inserts a new menu command.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @MenuID | INT | References the primary key from the Menu table. |
| @Title | VARCHAR(128) | The menu command title. |
| @Icon | VARCHAR(256) | The relative web path of the icon associated with the command. |
| @Link | VARCHAR(512) | The http path of the web page that must be opened when |

**Menu\_InsertMenu (SP)**

Purpose: inserts a new menu command.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @MenuID | INT | References the primary key from the Menu table. |
| @Title | VARCHAR(128) | The menu title. |

**Menu\_UpdateCommand (SP)**

Purpose: updates the specified menu command.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the MenuCommand table. |
| @MenuID | INT | References the primary key from the Menu table. |
| @Title | VARCHAR(128) | The menu command title. |
| @Icon | VARCHAR(256) | The relative web path of the icon associated with the command. |
| @Link | VARCHAR(512) | The http path of the web page that must be opened when |

**Menu\_UpdateMenu (SP)**

Purpose: updates the specified menu.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the Menu table. |
| @MenuID | INT | References the primary key from the Menu table. |
| @Title | VARCHAR(128) | The menu title. |

### Notification Module

The general prefix for the tables associated with this module is “Notification” and the stored procedures’ prefix is “Notification\_”.

**Notification\_Delete (SP)**

Purpose: deletes the specified notification.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @NotificationID | INT | References the primary key from the Notification table. |

**Notification\_GetLatestNotificationsForUser (SP)**

Purpose: retrieves the latest top 10 notifications for the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Language | VARCHAR(8) | The language code representing the language in which the notification categories should be translated in. |

**Notification\_GetNewNotificationsForUser (SP)**

Purpose: retrieves latest notification primary key value for the specified user if the @LastID parameter has the value “0” or retrieves the information regarding the notifications that appear after the notification specified through the @LastID parameter.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @LastID | INT | References the primary key from the Notification table. |
| @Language | VARCHAR(8) | The language code representing the language in which the notification categories should be translated in. |

**Notification\_GetNotificationsForUser (SP)**

Purpose: retrieves all the notifications for the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Language | VARCHAR(8) | The language code representing the language in which the notification categories should be translated in. |

**Notification\_SendToGroups (SP)**

Purpose: inserts the given notification for all the users belonging to the specified groups.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Groups | VARCHAR(8000) | The list of groups separated by the “,” char. |
| @Type | VARCHAR(32) | The internal name of the notification type. |
| @Link | VARCHAR(256) | An http link that refers to the notification. |
| @Message | NVARCHAR(MAX) | The message text in a specific tokened HTML format. |

**Notification\_SendToUser (SP)**

Purpose: inserts the given notification for the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Type | VARCHAR(32) | The internal name of the notification type. |
| @Link | VARCHAR(256) | An http link that refers to the notification. |
| @Message | NVARCHAR(MAX) | The message text in pure HTML format. |

### Objects Module

The general prefix for the tables associated with this module is “Object” and the stored procedures’ prefix is “Object\_”.

**Object\_CloneObject (SP)**

Purpose: clones the specified object, creates the cache for it in the afferent cache table and returns the primary key value for the new object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @UserID | INT | References the primary key from the User table. |
| @WithOutput | INT | Specifies if the procedure must return the primary key value for the new object or not. |

**Object\_DeleteObject (SP)**

Purpose: deletes the specified object and the specified object’s cache and logs the operation.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetAttributesForObject (SP)**

Purpose: retrieves all the information regarding all the attributes of the specified objects.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Object\_GetAttributesForObjectExt (SP)**

Purpose: retrieves all the information regarding all the attributes of the specified objects and extra fields for access level, computed default value and so on.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @ParentID | INT | References the primary key from the Object table. |
| @ClassID | INT | References the primary key from the Class table. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetAttributesForObjectLog (SP)**

Purpose: retrieves all the information regarding all the attributes of the specified object log.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @LogID | INT | References the primary key from the ObjectLog table. |

**Object\_GetAttributeValueForObject (UDF)**

Purpose: retrieves the value for the specified object and specified attribute.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @Attribute | VARCHAR(128) | The name of the attribute. |

**Object\_GetBreadcrumbs (SP)**

Purpose: retrieves the information regarding the objects situated above the specified object in the hierarchy.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Object\_GetChildObjects (SP)**

Purpose: retrieves the information regarding the child objects of the specified class for the specified object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @RootObjectID | INT | References the primary key from the Object table. |
| @ChildClassID | INT | References the primary key from the Class table. |

**Object\_GetEntry (SP)**

Purpose: retrieves the row assign to an object in the cache module.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Object\_GetLatestChildID (UDF)**

Purpose: returns the last inserted child object’s primary key value for the specified object of the specified class. Supplementary, an attribute equals value criteria may be specified.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @Class | VARCHAR(128) | The name of the class. |
| @Attribute | VARCHAR(128) | The attribute’s name for the restriction criteria. |
| @Value | NVARCHAR(MAX) | The attribute’s value for the restriction criteria. |

**Object\_GetLogsForObject (SP)**

Purpose: retrieves the information regarding all the logs for the specified objects.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Object\_GetObjectClass (UDF)**

Purpose: returns the primary key value of the class of which the object belongs to.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Object\_GetObjectInfo (SP)**

Purpose: retrieves the information regarding the specified object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @Language | VARCHAR(8) | The language code in which the object’s title must be translated. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetObjectListing (SP)**

Purpose: retrieves the list of objects that match the give criteria.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @ParentID | INT | References the primary key from the Object table. |
| @Filter | NVARCHAR(MAX) | Field filters to restrict the result encoded in a special XML string. |
| @Order | VARCHAR(8000) | A list of all the attributes separated using the “,” character that represents the ordering criteria for the result. |
| @MappedToID | INT | References the primary key from the Class table. Filters only the objects that are in a mapping relation with objects from this referred class. |
| @MappingFilter | INT | Specifies the mapping relation with the object from the class specified through the @MappedToID parameter. 0 means ignore relation, 1 means mapped with and -1 means not mapped with. |
| @Search | NVARCHAR(128) | General search criteria field. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetObjectLogInfo (SP)**

Purpose: retrieves the information regarding the specified object log.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @LogID | INT | References the primary key from the ObjectLog table. |

**Object\_GetObjectMatrix (SP)**

Purpose: retrieves an object matrix containing all the children object of the specified class for the specified object displayed in a two dimensional array indexed by the two specified dimensions.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @RootObjectID | INT | References the primary key from the Object table. |
| @ChildClassID | INT | References the primary key from the Class table. |
| @Vertical | VARCHAR(128) | The name of the attribute that will be the vertical dimension. |
| @Horizontal | VARCHAR(128) | The name of the attribute that will be the horizontal dimension. |

**Object\_GetParent (UDF)**

Purpose: returns the primary key value of the object that is the parent of the specified object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |

**Object\_GetPossibleParents (SP)**

Purpose: retrieves the list of all the objects that can be parent objects for the specified object according to the ClassLink table.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @Language | VARCHAR(8) | The language code in which the information should be translated. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetRecycleObjects (SP)**

Purpose: retrieves a list with the last 100 deleted object of the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetRecycleSummary (SP)**

Purpose: retrieves the numbers of objects deleted for each class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Language | VARCHAR(8) | The language code in which the information should be translated. |

**Object\_GetRowSourceForAttribute (SP)**

Purpose: retrieves the result of the computed SQL query provided as row source for the specified attribute.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @AttributeID | INT | References the primary key from the ClassAttribute table. |
| @ObjectID | INT | References the primary key from the Object table. |
| @ObjectParentID | INT | References the primary key from the Object table. |
| @UserID | INT | References the primary key from the User table. |

**Object\_GetSearchResults (SP)**

Purpose: retrieves the full listing information for the objects specified in the list of objects of the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @IDs | VARCHAR(MAX) | The list of object primary key values separated by the “,” character. |

**Object\_InsertObject (SP)**

Purpose: inserts a new object, creates the cache for it in the afferent cache table and returns the primary key value for the new object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @ParentID | INT | References the primary key from the Object table. |
| @Attributes | NVARCHAR(MAX) | The values for all the object’s attributes encode in a specific XML text. |
| @UserID | INT | References the primary key from the User table. |
| @WithOutput | INT | Specifies if the procedure must return the primary key value for the new object or not. |

**Object\_LogObject (SP)**

Purpose: logs the specified objects and creates the appropriate entry in the cache log.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @UserID | INT | References the primary key from the User table. |

**Object\_RestoreObject (SP)**

Purpose: restores the specified deleted object with or without its deleted children.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @WithChildren | INT | Specifies if the children objects will be restored as well. |

**Object\_Search (SP)**

Purpose: retrieves the primary key values for all the objects that are compatible with the given search criteria.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Words | NVARCHAR(4000) | The search criteria. |
| @UserID | INT | References the primary key from the User table. |
| @Language | VARCHAR(8) | The language code for the language in which the information must be translated. |

**Object\_UpdateAttribute (SP)**

Purpose: updates the specified attribute of the specified object with the given value. Also, it will log the change of the entire object and it will change the logs as well.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @Attribute | VARCHAR(128) | The name of the attribute that will be updated. |
| @Value | NVARCHAR(MAX) | The new value for the attribute. |
| @UserID | INT | References the primary key from the User table. |

**Object\_UpdateObject (SP)**

Purpose: updates the specified object and the cache for it in the afferent cache table logging the change as well.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @Attributes | NVARCHAR(MAX) | The values for all the object’s attributes encode in a specific XML text. |
| @UserID | INT | References the primary key from the User table. |

**Object\_UpdateObjectParent (SP)**

Purpose: updates the parent for the specified object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @ParentID | INT | References the primary key from the Object table. |

### Reference Guide Module

The general prefix for the stored procedures’ prefix is “ReferenceGuide\_”. This module doesn’t have any adjacent tables.

**ReferenceGuide\_GetAdjacentClasses**

Purpose: retrieves the adjacent classes for the specified class (mapped, master or slave).

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**ReferenceGuide\_GetAttributesForClass**

Purpose: retrieves the list of attributes for the specified class.

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**ReferenceGuide\_GetClasses**

Purpose: retrieves the list of classes.

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The language code for the language in which the information must be translated. |

### Reporting Module

The general prefix for the tables associated with this module is “Report” and the stored procedures’ prefix is “Report\_”.

**Report\_DeleteGraph (SP)**

Purpose: deletes the specified graph.

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ReportGraph table. |

**Report\_DeleteParameter (SP)**

Purpose: deletes the specified parameter.

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ParamID | INT | References the primary key from the ReportParameter table. |

**Report\_DeleteReport (SP)**

Purpose: deletes the specified report with all the afferent graphs and parameters.

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |

**Report\_ExistsParameter (SP)**

Purpose: checks if the specified parameter exists for the given report.

Parameter:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @Name | VARCHAR(128) | The name of the parameter. |
| @ExceptID | INT | References the primary key from the ReportParameter table. Represents a parameter that will be avoided in the search and it is used when a parameter is updated. |

**Report\_ExistsReport (SP)**

Purpose: checks if the specified report exists.

Parameter:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the report. |
| @ExceptID | INT | References the primary key from the Report table. Represents a report that will be avoided in the search and it is used when a report is updated. |

**Report\_GetGraphData (SP)**

Purpose: retrieves the data for the given graph.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GraphID | INT | References the primary key from the ReportGraph table. |
| @UserID | INT | References the primary key from the User table. |
| @Params | NVARCHAR(MAX) | The parameter values encoded in a specific XML structure. |

**Report\_GetGraphDataAtDate (SP)**

Purpose: retrieves the data at the specified moment in time for the given graph.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GraphID | INT | References the primary key from the ReportGraph table. |
| @UserID | INT | References the primary key from the User table. |
| @Params | NVARCHAR(MAX) | The parameter values encoded in a specific XML structure. |
| @Date | VARCHAR(32) | The date in the “DD/MM/YYYY HH:MM” format. |

**Report\_GetGraphInfo (SP)**

Purpose: retrieves the information regarding a graph.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ReportGraph table. |
| @Language | VARCHAR(8) | The language code for the language in which the information must be translated. |

**Report\_GetGraphsForReport (SP)**

Purpose: retrieves the information all the graphs linked to the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the Report table. |
| @Language | VARCHAR(8) | The language code for the language in which the information must be translated. |

**Report\_GetGraphTypes (SP)**

Purpose: retrieves the list of all the supported graph types.

Parameters: NONE

**Report\_GetLanguageTagsForReport (SP)**

Purpose: retrieves the list of all the language tags linked to the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |

**Report\_GetParameterInfo (SP)**

Purpose: retrieve the information regarding the specified parameter.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ParamID | INT | References the primary key from the ReportParameter table. |

**Report\_GetParametersForReport (SP)**

Purpose: retrieves the list with all the parameters regarding the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |

**Report\_GetReportData (SP)**

Purpose: retrieves the data for the given report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @UserID | INT | References the primary key from the User table. |
| @Params | NVARCHAR(MAX) | The parameter values encoded in a specific XML structure. |

**Report\_GetReportDataAtDate (SP)**

Purpose: retrieves the data at the specified moment in time for the given report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @UserID | INT | References the primary key from the User table. |
| @Params | NVARCHAR(MAX) | The parameter values encoded in a specific XML structure. |
| @Date | VARCHAR(32) | The date in the “DD/MM/YYYY HH:MM” format. |

**Report\_GetReportInfo (SP)**

Purpose: retrieves the information for the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @Language | VARCHAR(8) | The language code for the language in which the information must be translated. |

**Report\_GetReportList (SP)**

Purpose: retrieves the list with all the reports.

Parameters: NONE

**Report\_GetRowSourceForParameter (SP)**

Purpose: retrieves the data regarding a parameter row source.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ParamID | INT | References the primary key from the ReportParameter table. |
| @UserID | INT | References the primary key from the User table. |

**Report\_InsertGraph (SP)**

Purpose: inserts a new graph.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @Name | INT | The name of the graph. |
| @TypeID | INT | References the primary key from the ReportGraphType table. |
| @RowSource | NVARCHAR(MAX) | A specific SQL query for that retrieves the graph’s data. |

**Report\_InsertParameter (SP)**

Purpose: inserts a new parameter for the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @Control | VARCHAR(128) | The name of the control through which the parameter value will be collected. |
| @Type | VARCHAR(64) | The name of the parameter’s data type. |
| @TypeLength | INT | The maximum length of the parameter’s value. Required only for some parameter data types such as NVARCHAR. |
| @IsRequired | INT | Used as a Boolean to specify if the parameter requires a value or not. |
| @ControlWidth | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlHeight | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlMin | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlMax | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlUnit | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlStyle | VARCHAR(512) | For some controls this is an optional parameter that defines the control. |
| @ControlRowSource | VARCHAR(MAX) | For some controls this is a required parameter that defines the control. |

**Report\_InsertReport (SP)**

Purpose: inserts a new report and returns the primary key value of the new report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the report. |
| @Title | VARCHAR(1024) | The report’s title. |

**Report\_UpdateGraph (SP)**

Purpose: updates the specified graph.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ReportGraph table. |
| @Name | INT | The name of the graph. |
| @TypeID | INT | References the primary key from the ReportGraphType table. |
| @RowSource | NVARCHAR(MAX) | A specific SQL query for that retrieves the graph’s data. |

**Report\_UpdateParameter (SP)**

Purpose: updates the specified parameter.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the ReportParameter table. |
| @Control | VARCHAR(128) | The name of the control through which the parameter value will be collected. |
| @Type | VARCHAR(64) | The name of the parameter’s data type. |
| @TypeLength | INT | The maximum length of the parameter’s value. Required only for some parameter data types such as NVARCHAR. |
| @IsRequired | INT | Used as a Boolean to specify if the parameter requires a value or not. |
| @ControlWidth | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlHeight | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlMin | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlMax | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlUnit | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlStyle | VARCHAR(512) | For some controls this is an optional parameter that defines the control. |
| @ControlRowSource | VARCHAR(MAX) | For some controls this is a required parameter that defines the control. |

**Report\_UpdateReport (SP)**

Purpose: updates the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @RowsOnPage | INT | Specifies the maximum numbers of entries that may be displayed on a single page. |
| @Name | VARCHAR(128) | The name of the report. |
| @Title | VARCHAR(1024) | The report’s title. |
| @LookParams | NVARCHAR(MAX) | A specific tokened HTML that describes the form that collects the parameters for the report. |
| @LookHeader | NVARCHAR(MAX) | A specific tokened HTML that describes the header for the report. |
| @LookRow | NVARCHAR(MAX) | A specific tokened HTML that describes a single entry/data row for the report. |
| @RowSource | NVARCHAR(MAX) | A specific SQL query that retrieves the data for the report. |

### Security Module

The general prefix for the tables associated with this module is “Security” and the stored procedures’ prefix is “Security\_”.

**Security\_AddClassAccess (SP)**

Purpose: gives access for the specified class to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Security\_AddFormAccess (SP)**

Purpose: gives access for the specified custom form to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Security\_AddReportAccess (SP)**

Purpose: gives access for the specified report to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Security\_CrossSiteScripting (UDF)**

Purpose: parses a text to eliminate special characters in order to avoid the cross-site-scripting vulnerability.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Text | NVARCHAR(MAX) | The text that needs to be parsed. |

**Security\_GetClassesForGroup (SP)**

Purpose: retrieves the list of classes on which the specified group has access.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Security\_GetFormsForGroup (SP)**

Purpose: retrieves the list of custom forms on which the specified group has access.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Security\_GetGroupsForClass (SP)**

Purpose: retrieves the list of the groups that have access to the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Security\_GetGroupsForForm (SP)**

Purpose: retrieves the list of the groups that have access to the specified custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |

**Security\_GetGroupsForReport (SP)**

Purpose: retrieves the list of the groups that have access to the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |

**Security\_GetReportsForGroup (SP)**

Purpose: retrieves the list of reports on which the specified group has access.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**Security\_HasAccessOnClass (SP)**

Purpose: determines if the specified user has access to the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |
| @UserID | INT | References the primary key from the User table. |

**Security\_HasAccessOnForm (SP)**

Purpose: determines if the specified user has access to the specified custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @FormID | INT | References the primary key from the Form table. |
| @UserID | INT | References the primary key from the User table. |

**Security\_HasAccessOnObject (SP)**

Purpose: determines if the specified user has access to the specified object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID | INT | References the primary key from the Object table. |
| @UserID | INT | References the primary key from the User table. |

**Security\_HasAccessOnPage (SP)**

Purpose: determines if the specified user has access to the specified web page.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Page | VARCHAR(128) | The name of the web page including the “aspx” extension. |
| @UserID | INT | References the primary key from the User table. |

**Security\_HasAccessOnReport (SP)**

Purpose: determines if the specified user has access to the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ReportID | INT | References the primary key from the Report table. |
| @UserID | INT | References the primary key from the User table. |

**Security\_IsUserCompatibleWithGroups (UDF)**

Purpose: returns an integer that is used as a Boolean in order to say if the specified user belongs to at least one access group from the given group list.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Groups | VARCHAR(MAX) | List of group names separated by the “,” character. |

**Security\_IsUserMemberOfGroups (SP)**

Purpose: returns an integer that is used as a Boolean in order to say if the specified user belongs to at least one access group from the given group list.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Groups | VARCHAR(MAX) | List of group names separated by the “,” character. |

**Security\_RevokeClassAccess (SP)**

Purpose: revokes access for the specified group on the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the SecurityClass table. |

**Security\_RevokeFormAccess (SP)**

Purpose: revokes access for the specified group on the specified custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the SecurityForm table. |

**Security\_RevokeReportAccess (SP)**

Purpose: revokes access for the specified group on the specified report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the SecurityReport table. |

### Settings Module

The general prefix for the tables associated with this module is “Setting” and the stored procedures’ prefix is “Setting\_”.

**Setting\_GetSettingsForUser (SP)**

Purpose: retrieves the values for all the settings for the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**Setting\_GetValueForSetting (SP)**

Purpose: retrieves the value for the specified setting and the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Setting | VARCHAR(64) | The name of the setting. |
| @UserID | INT | References the primary key from the User table. |

**Setting\_Update (SP)**

Purpose: sets the value for the specified setting and the specified user with a new value.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Setting | VARCHAR(64) | The name of the setting. |
| @UserID | INT | References the primary key from the User table. |
| @Value | NVARCHAR(4000) | The new value for the setting. |

### Users Module

The general prefix for the tables associated with this module is “User” and the stored procedures’ prefix is “User\_”.

**User\_DeleteApartenence (SP)**

Purpose: deletes a user appurtenance to a group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ApartenenceID | INT | References the primary key from the UserApartenence table. |

**User\_DeleteGroup (SP)**

Purpose: deletes a user group and the link for it (class, report and custom form access and user appurtenance).

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**User\_DeleteUser (SP)**

Purpose: deletes the specified user and its attachment information (desktop windows, desktop icons and appurtenance).

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**User\_GetAdministrativeStatistic (SP)**

Purpose: retrieves statistical data regarding the platform status.

Parameters: NONE

**User\_GetApartenencesForUser (SP)**

Purpose: retrieves the list of all the groups the user belongs to.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**User\_GetGroupInfo (SP)**

Purpose: retrieves the information regarding the specified user group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**User\_GetGroups (SP)**

Purpose: retrieves the list of all the user groups.

Parameters: NONE

**User\_GetGUIDFromUserID (SP)**

Purpose: retrieves the specified user’s GUID.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ID | INT | References the primary key from the User table. |

**User\_GetIDForGroup (SP)**

Purpose: retrieves the primary key value for the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Group | VARCHAR(64) | The name of the group. |

**User\_GetUserIDFromGUID (SP)**

Purpose: retrieves the primary key value for the user specified through its GUID.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GUID | VARCHAR(64) | The user’s GUID. |

**User\_GetUserInfo (SP)**

Purpose: retrieves all the information regarding the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**User\_GetUserPhoto (SP)**

Purpose: retrieves the primary key value for the file that represents the user’s photo for the specified user.

Parameter:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserName | VARCHAR(64) | The user name used for authentication. |

**User\_GetUsers (SP)**

Purpose: retrieves the users list with all the information regarding them.

Parameters: NONE

**User\_GetUsersForGroup (SP)**

Purpose: retrieves the list of users that belong to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |

**User\_InsertApartenence (SP)**

Purpose: links the specified user to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @GroupID | INT | References the primary key from the UserGroup table. |

**User\_InsertGroup (SP)**

Purpose: inserts a new group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(64) | The group’s name. |

**User\_InsertUser (SP)**

Purpose: inserts a new user, links it to the default group and returns the primary key value for the new user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @PhotoID | INT | References the primary key from the File table. |
| @Phone | VARCHAR(16) | The user’s phone number. |
| @UserName | VARCHAR(64) | The user’s user name used for authentication. |
| @FirstName | NVARCHAR(64) | The user’s first name. |
| @LastName | NVARCHAR(64) | The user’s last name. |
| @Password | VARCHAR(64) | The user’s password used for authentication. |
| @eMail | VARCHAR(128) | The user’s e-mail address. |
| @MustChangePassword | INT | Used as a Boolean to specify the fact that the user needs to change its password at the next logon. |
| @WithOutput | INT | Used as a Boolean to specify if the procedure must return the primary key value for the new user or not. |

**User\_IsMemberOf (SP)**

Purpose: returns an integer used as a Boolean to specify if the specified user belongs to the specified group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Group | VARCHAR(64) | The group’s name. |

**User\_IsUserNameAvailable (SP)**

Purpose: returns as an integer used as a Boolean to specify if the specified user name exists.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserName | VARCHAR(64) | The user’s user name. |
| @AvoidUserID | INT | References the primary key from the User table. The user’s primary key value that needs to be avoided in case of an update. |

**User\_LockUser (SP)**

Purpose: marks the specified user’s account as being locked.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**User\_LogIn (SP)**

Purpose: Authenticates a user and returns the primary key value for the authenticated user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserName | VARCHAR(64) | The user’s user name. |
| @Password | VARCHAR(64) | The user’s password. |

**User\_MustChangePassword (SP)**

Purpose: returns an integer used as a Boolean specifying if the specified user must change its password or not.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**User\_UnlockUser (SP)**

Purpose: removes the locked marker for a user if the user account is marked as locked.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |

**User\_UpdateGroup (SP)**

Purpose: updates a group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @GroupID | INT | References the primary key from the UserGroup table. |
| @Name | VARCHAR(64) | The group’s name. |

**User\_UpdateUser (SP)**

Purpose: updates the information regarding the specified user.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @UserID | INT | References the primary key from the User table. |
| @Phone | VARCHAR(16) | The user’s phone number. |
| @UserName | VARCHAR(64) | The user’s user name used for authentication. |
| @FirstName | NVARCHAR(64) | The user’s first name. |
| @LastName | NVARCHAR(64) | The user’s last name. |
| @Password | VARCHAR(64) | The user’s password used for authentication. |
| @eMail | VARCHAR(128) | The user’s e-mail address. |

### Utility Module

The general prefix for the tables associated with this module is “Utility” and the stored procedures’ prefix is “Utility\_”.

**Utility\_BinaryToChar (UDF)**

Purpose: converts a stream of binary data to the corresponding char stream.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Data | VARBINARY(MAX) | The binary stream that needs to be parsed into a char stream. |

**Utility\_DeleteClasses (SP)**

Purpose: deletes all the classes, reports, custom forms and event data with the security links for the groups.

Parameters: NONE

**Utility\_DeleteObjects (SP)**

Purpose: deletes all the object related data including mapping information and logs.

Parameters: NONE

**Utility\_DeleteObjectsForClass (SP)**

Purpose: deleted all the objects information that belongs to the specified class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ClassID | INT | References the primary key from the Class table. |

**Utility\_ExportClass (SP)**

Purpose: export all the relevant data in order to replicate the classes on another instance.

Parameters: NONE

**Utility\_ExportClassAttribute (SP)**

Purpose: export all the relevant data in order to replicate the class attributes on another instance.

Parameters: NONE

**Utility\_ExportClassAttributeType (SP)**

Purpose: export all the relevant data in order to replicate the class attribute types on another instance.

Parameters: NONE

**Utility\_ExportClassLink (SP)**

Purpose: export all the relevant data in order to replicate the class links on another instance.

Parameters: NONE

**Utility\_ExportConstant (SP)**

Purpose: export all the relevant data in order to replicate the constants on another instance.

Parameters: NONE

**Utility\_ExportControl (SP)**

Purpose: export all the relevant data in order to replicate the controls on another instance.

Parameters: NONE

**Utility\_ExportDesktopDefaultIcon (SP)**

Purpose: export all the relevant data in order to replicate the desktop default icons on another instance.

Parameters: NONE

**Utility\_ExportEvent (SP)**

Purpose: export all the relevant data in order to replicate the events on another instance.

Parameters: NONE

**Utility\_ExportEventTrigger (SP)**

Purpose: export all the relevant data in order to replicate the event triggers on another instance.

Parameters: NONE

**Utility\_ExportForm (SP)**

Purpose: export all the relevant data in order to replicate the custom forms on another instance.

Parameters: NONE

**Utility\_ExportLanguage (SP)**

Purpose: export all the relevant data in order to replicate the languages on another instance.

Parameters: NONE

**Utility\_ExportLanguageTag (SP)**

Purpose: export all the relevant data in order to replicate the language tags on another instance.

Parameters: NONE

**Utility\_ExportMappingClass (SP)**

Purpose: export all the relevant data in order to replicate the mapping class relations on another instance.

Parameters: NONE

**Utility\_ExportMenu (SP)**

Purpose: export all the relevant data in order to replicate the menus on another instance.

Parameters: NONE

**Utility\_ExportMenuCommand (SP)**

Purpose: export all the relevant data in order to replicate the menu commands on another instance.

Parameters: NONE

**Utility\_ExportMenuCommandAccess (SP)**

Purpose: export all the relevant data in order to replicate the menu command access rights on another instance.

Parameters: NONE

**Utility\_ExportPage (SP)**

Purpose: export all the relevant data in order to replicate the web pages on another instance.

Parameters: NONE

**Utility\_ExportReport (SP)**

Purpose: export all the relevant data in order to replicate the reports on another instance.

Parameters: NONE

**Utility\_ExportReportGraph (SP)**

Purpose: export all the relevant data in order to replicate the report graphs on another instance.

Parameters: NONE

**Utility\_ExportReportGraphType (SP)**

Purpose: export all the relevant data in order to replicate the report graph types on another instance.

Parameters: NONE

**Utility\_ExportReportParameter (SP)**

Purpose: export all the relevant data in order to replicate the report parameters on another instance.

Parameters: NONE

**Utility\_ExportSecurityClass (SP)**

Purpose: export all the relevant data in order to replicate the security for the classes on another instance.

Parameters: NONE

**Utility\_ExportSecurityForm (SP)**

Purpose: export all the relevant data in order to replicate the security for the custom forms on another instance.

Parameters: NONE

**Utility\_ExportSecurityPageAccess (SP)**

Purpose: export all the relevant data in order to replicate the page access rights on another instance.

Parameters: NONE

**Utility\_ExportSecurityReport (SP)**

Purpose: export all the relevant data in order to replicate the security for the reports on another instance.

Parameters: NONE

**Utility\_ExportSetting (SP)**

Purpose: export all the relevant data in order to replicate the settings on another instance.

Parameters: NONE

**Utility\_ExportStoredProcedure (SP)**

Purpose: export all the relevant data in order to replicate the stored procedures and user defined functions on another instance.

Parameters: NONE

**Utility\_ExportUserGroup (SP)**

Purpose: export all the relevant data in order to replicate the user groups on another instance.

Parameters: NONE

**Utility\_GenerateDataScript (SP)**

Purpose: export all the relevant data in order to replicate an instance data on another instance.

Parameters: NONE

**Utility\_GetPossibleObjectDuplicates (SP)**

Purpose: retrieves data regarding all the objects that may be duplicated.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Language | VARCHAR(8) | The language code representing the language in which the data will be translated. |

**Utility\_ImportClass (SP)**

Purpose: Imports a single class.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @HelpArticleID | INT | References the primary key from the Article table. Points to the associated help article for the class, if any. |
| @Icon | VARCHAR(256) | The relative web path where the icon associated with the new class will be found. |
| @Name | VARCHAR(128) | The name of the class used in development. |
| @NameSg | VARCHAR(128) | The name of the class in singular form used for the interface. This value is considered a language tag and will be translated when displayed to the user. |
| @NamePl | VARCHAR(128) | The name of the class in plural form used for the interface. This value is considered a language tag and will be translated when displayed to the user. |
| @CustomPage | VARCHAR(256) | The name of the custom page that will display objects from the specified class, if any. |
| @Title | VARCHAR(8000) | A specific tokened SQL expression that computes the title for all the objects of the specified class. |
| @LookSingle | VARCHAR(MAX) | A specific tokened HTML that triggers the way in which a data collecting form for an object of the specified class will look like. |
| @LookMulti | VARCHAR(MAX) | A specific tokened HTML that triggers the way in which a listing for all the objects in the specified class looks like. |

**Utility\_ImportClassAttribute (SP)**

Purpose: Imports a single class attribute.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Class | VARCHAR(64) | The internal name of the class. |
| @Type | VARCHAR(64) | The name of the data type. |
| @Control | VARCHAR(128) | The name of the control through which the value of the attribute will be collected. |
| @IsForeignKey | INT | Used as a Boolean to indicate that the attribute’s value will reference another object. |
| @IsFormula | INT | Used as a Boolean to indicate that the attribute will be computed based on a formula when the object is saved. |
| @IsRequired | INT | Used as a Boolean to indicate that the attribute requires a value and can’t be left blank. |
| @TypeLenght | VARCHAR(MAX) | Represents the maximum length for the attribute’s value. It is required for some attribute data type such as NVARCHAR (this must have a length between 1 and 4000 or MAX). |
| @Name | VARCHAR(128) | The name of the attribute. |
| @ErrorMessage | VARCHAR(256) | A tokened string that is thrown as an exception when the attribute value is left blank and the attribute is required. |
| @Access | VARCHAR(MAX) | A specific SQL expression that must return ‘WRITE’ or ‘READ’ depending on the user’s access level or the object’s workflow. |
| @DefaultValue | VARCHAR(MAX) | A specific SQL expression that returns the attribute’s default value or gives the formula used to compute its value in case the @IsFormula parameter is set to true. |
| @ControlWidth | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlHeight | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlMin | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlMax | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlUnit | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlStyle | VARCHAR(512) | For some controls this is an optional parameter that defines the control. |
| @ControlRowSource | VARCHAR(MAX) | For some controls this is a required parameter that defines the control. |

**Utility\_ImportClassAttributeType (SP)**

Purpose: Imports a single class attribute type.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @DummyValue | VARCHAR(32) | A fixed value used for specific SQL statements and triggers. |
| @Type | VARCHAR(64) | The name of the attribute type. |
| @Input | VARCHAR(128) | A tokened SQL statement used to convert the values from text to the desired SQL data types. |
| @Output | VARCHAR(128) | A tokened SQL statement used to convert values from SQL data types to text in order to be displayed to the user. |
| @Formula | VARCHAR(128) | A tokened SQL statement used to convert values to specific format in order to be processed in specific SQL statements. |

**Utility\_ImportClassLink (SP)**

Purpose: inserts a single class link relation.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @MasterClass | VARCHAR(128) | The internal name of the master class. |
| @SlaveClass | VARCHAR(128) | The internal name of the slave class. |

**Utility\_ImportConstant (SP)**

Purpose: inserts a single constant.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Constant | VARCHAR(64) | The constant’s name. |
| @Value | NVARCHAR(64) | The value for the constant. |

**Utility\_ImportControl (SP)**

Purpose: inserts a single control.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The control’s name. |
| @Attributes | VARCHAR(MAX) | The list of supported properties separated by the “,” character. |

**Utility\_ImportDesktopDefaultIcon (SP)**

Purpose: inserts a single default desktop icon.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Group | VARCHAR(64) | The name of the group for which the icon is destined. |
| @PosX | INT | The horizontal position in pixels of the icon. |
| @PosY | INT | The vertical position in pixels of the icon. |
| @Icon | VARCHAR(128) | The relative web path of the pictogram associated with the icon. |
| @Title | NVARCHAR(256) | The icon’s title. |
| @Link | VARCHAR(512) | The http web address of the web page that must be opened when the icon is triggered. |

**Utility\_ImportEvent (SP)**

Purpose: inserts a single event.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the event. |

**Utility\_ImportEventTrigger (SP)**

Purpose: inserts a single event.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Class | VARCHAR(128) | The internal name of the class. |
| @Event | VARCHAR(128) | The name of the event. |
| @Command | NVARCHAR(MAX) | The specific SQL query representing the event trigger. |

**Utility\_ImportForm (SP)**

Purpose: inserts a single custom form.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Class | VARCHAR(128) | The name of the class. |
| @Name | VARCHAR(128) | The internal name for the custom form. |
| @Look | NVARCHAR(MAX) | The specific tokened HTML describing the custom form. |

**Utility\_ImportLanguage (SP)**

Purpose: inserts a single language.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Code | VARCHAR(8) | The code representing the language. |
| @Language | NVARCHAR(32) | The language’s name. |

**Utility\_ImportLanguageTag (SP)**

Purpose: inserts a single language tag.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Code | VARCHAR(8) | The code representing the language. |
| @Table | VARCHAR(64) | The table for which the language tag is destined. |
| @Parameter1 | VARCHAR(64) | The first parameter characterising the tag. |
| @Parameter2 | VARCHAR(64) | The second parameter characterising the tag. |
| @Value | NVARCHAR(256) | The language tag value. |

**Utility\_ImportMappingClass (SP)**

Purpose: inserts a single mapping relation between two classes.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @SourceClass | VARCHAR(128) | The internal name of the source class. |
| @TargetClass | VARCHAR(128) | The internal name of the target class. |
| @IsCommutative | INT | Used as a Boolean to specify if the relation is one-way or both-ways. |

**Utility\_ImportMenu (SP)**

Purpose: inserts a single menu.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Title | VARCHAR(128) | The menu’s title. |
| @Path | VARCHAR(MAX) | The menu’s full virtual path in the menus tree structure. |

**Utility\_ImportMenuCommand (SP)**

Purpose: inserts a single menu command.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Path | VARCHAR(MAX) | The command’s full virtual path in the menus tree structure. |
| @Title | VARCHAR(128) | The menu command’s title. |
| @Icon | VARCHAR(256) | The relative web path for the icon associated with the menu command. |
| @Link | VARCHAR(512) | The http path for the web page that must be opened when the menu command is triggered. |

**Utility\_ImportMenuCommandAccess (SP)**

Purpose: inserts a single menu command access link.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Path | VARCHAR(MAX) | The command’s full virtual path in the menus tree structure. |
| @Command | VARCHAR(128) | The title for the command. |
| @Group | VARCHAR(64) | The name of the group. |

**Utility\_ImportPage (SP)**

Purpose: inserts a single page.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the page including the “aspx” extension. |

**Utility\_ImportReport (SP)**

Purpose: inserts a single report.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @RowsOnPage | INT | Specifies the maximum number of entries that may be displayed on a single report page. |
| @Name | VARCHAR(128) | The internal name for the report. |
| @Title | VARCHAR(128) | The report’s title. |
| @LookParams | VARCHAR(MAX) | A specific tokened HTML description for the parameter collection form. |
| @LookHeader | VARCHAR(MAX) | A specific tokened HTML description for the report’s header. |
| @LookRow | VARCHAR(MAX) | A specific tokened HTML description for a single report entry. |
| @RowSource | VARCHAR(MAX) | A specific SQL query that returns the data for the report. |

**Utility\_ImportReportGraph (SP)**

Purpose: inserts a single graph.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Report | VARCHAR(128) | The internal report name. |
| @Type | VARCHAR(32) | The type of the graph. |
| @Name | VARCHAR(64) | The graph’s name. |
| @RowSource | VARCHAR(MAX) | A specific SQL query that returns the data for the graph. |

**Utility\_ImportReportGraphType (SP)**

Purpose: inserts a single graph type.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(32) | The name of the graph type. |

**Utility\_ImportReportParameter (SP)**

Purpose: inserts a single report parameter.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Report | VARCHAR(128) | The name of the report. |
| @Control | VARCHAR(128) | The name of the control through which the parameter value will be collected. |
| @Type | VARCHAR(64) | The name of the parameter’s type. |
| @TypeLength | INT | The length for the parameter’s value. Some data types require a length, such as the NVARCHAR data type. |
| @IsRequired | INT | Used as a Boolean to specify if the parameter requires a value or not. |
| @Name | VARCHAR(128) | The name of the parameter. |
| @ControlWidth | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlHeight | INT | For some controls this is a required or optional parameter that defines the control. |
| @ControlMin | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlMax | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlUnit | FLOAT | For some controls this is a required parameter that defines the control. |
| @ControlStyle | VARCHAR(512) | For some controls this is an optional parameter that defines the control. |
| @ControlRowSource | VARCHAR(MAX) | For some controls this is a required parameter that defines the control. |

**Utility\_ImportSecurityClass (SP)**

Purpose: inserts a single class access link for a group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Class | VARCHAR(128) | The internal name of the class. |
| @Group | VARCHAR(64) | The name of the group. |

**Utility\_ImportSecurityForm (SP)**

Purpose: inserts a single custom form access link for a group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Form | VARCHAR(128) | The name of the form. |
| @Group | VARCHAR(64) | The name of the group. |

**Utility\_ImportSecurityPageAccess (SP)**

Purpose: inserts a single web page access link for a group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Page | VARCHAR(128) | The name of the web page. |
| @Group | VARCHAR(64) | The name of the group. |

**Utility\_ImportSecurityReport (SP)**

Purpose: inserts a single report access link for a group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Report | VARCHAR(128) | The name of the report. |
| @Group | VARCHAR(64) | The name of the group. |

**Utility\_ImportSetting (SP)**

Purpose: inserts a single setting.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the setting. |
| @Description | VARCHAR(8000) | The description for the setting. |
| @DefaultValue | VARCHAR(8000) | The default value for the setting. |

**Utility\_ImportStoredProcedure (SP)**

Purpose: inserts a single stored procedure or user defined function.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the stored procedure or user defined function. |
| @Type | VARCHAR(32) | Specifies if the code is for a stored procedure or user defined function. |
| @Code | VARCHAR(MAX) | The SQL code for the stored procedure or user defined function. |

**Utility\_ImportUserGroup (SP)**

Purpose: inserts a single group.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @Name | VARCHAR(128) | The name of the group. |

**Utility\_InitDatabase (SP)**

Purpose: resets the database to a blank platform instance.

Parameters: NONE

**Utility\_MergeObjects (SP)**

Purpose: merges two objects into one.

Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Data Type  (Length) | Description |
| @ObjectID1 | INT | The primary key value of the primary object. |
| @ObjectID2 | INT | The primary key value of the secondary object. This object will be deleted after the merger. |
| @UserID | INT | References the primary key for the User table. |
| @IgnoreDifferences | INT | Used as a Boolean to ignore the differences between the two objects regarding attributes’ values. |
| @IgnoreClass | INT | Used as a Boolean to ignore the fact that the objects are of a different class. |
| @WithLogPreserve | INT | Used as a Boolean to specify if the log for the second object must be preserved or not. |
| @Language | VARCHAR(8) | The code for the language in which the possible errors will be translated. |

**Utility\_ResetCache (SP)**

Purpose: triggers a cache reset for all the classes.

Parameters: NONE

**Utility\_ResetUserGUIDs (SP)**

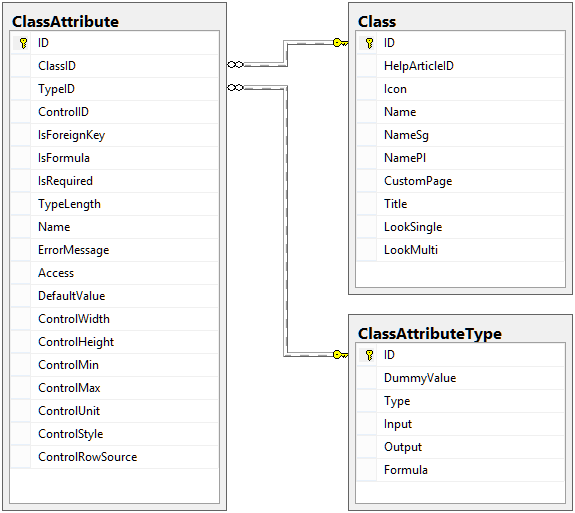
Purpose: resets the GUIDs for all the users.

Parameters: NONE

## Classes, Events, Objects, Mappings and Cache Philosophy

As said before, for every application that has a database behind it we are faced with difficulties when adding a new table column in the final stages of development. Because of this, the platform has a fixed structure that is able to retain every kind of data structures.

Now, in order to retain data for every kind of structure the first thing is to know/store the data regarding the data, meaning the metadata. This part is covered by the Class module which retains the metadata regarding the data structures and the relations between them.

****

Basic class module diagram

The above three tables hold the metadata regarding the data structures for which data will be stored. We can see here a similarity with the special system views named sys.objects, sys.columns and sys.types which return the metadata regarding the tables, stored procedures and user defined functions from the database. However, in our case these tables hold a bit more information regarding the data structures.

### Class Concept

To better illustrate this concept we can consider the example on which we started earlier, the employee evidence. Thus, for an employee we will start with the minimum information: first name, last name, PIN, birthdate, birth place and address. In order to define the employee data structure we will start with creating the data structure. This is done by adding a new row in the Class table through the Class\_InsertClass procedure. After the following procedure call we have a base for the new data structure.

EXEC Class\_InsertClass 'Images/Icons/User.png', 'Employee', 'Employee', 'Employees', 0

After calling this line of code we will have the following information in the Class table:



Preliminary information

What this means is that we have a data structure/entity, called “Employee” with no help article linked, no icon, no custom page, no title, no single look description and no multi look description. This is the preliminary information for the new entity. In order to fully describe an entity we have to have the list of attributes for the entity because the single look, multi look and title are based on those.

In order to attach the attributes to the newly created entity we can use the Class\_InsertAttribute procedure. After executing the following script, we have the attributes attached to the new entity.

EXEC Class\_InsertAttribute 3, 'NVARCHAR', 'TextBox', 0, 0, 1, '32', 'FirstName',

'You forgot to fill in the first name of the person.', '''WRITE''', '',

0, 0, 0, 0, 0, 'width: 200px;', ''

EXEC Class\_InsertAttribute 3, 'NVARCHAR', 'TextBox', 0, 0, 1, '32', 'LastName',

'You forgot to fill in the last name of the person.', '''WRITE''', '',

0, 0, 0, 0, 0, 'width: 200px;', ''

EXEC Class\_InsertAttribute 3, 'NVARCHAR', 'TextBox', 0, 0, 1, '16', 'PIN',

'You forgot to fill in the PIN for the person.', '''WRITE''', '',

0, 0, 0, 0, 0, 'width: 100px;', ''

EXEC Class\_InsertAttribute 3, 'DATETIME', 'DatePicker', 0, 0, 0, '', 'Birthdate',

'', '''WRITE''', '', 0, 0, 0, 0, 0, '', ''

EXEC Class\_InsertAttribute 3, 'NVARCHAR', 'TextBox', 0, 0, 0, '256', 'BirthPlace',

'', '''WRITE''', '', 0, 0, 0, 0, 0, 'width: 300px;', ''

EXEC Class\_InsertAttribute 3, 'NVARCHAR', 'TextArea', 0, 0, 0, '512', 'Address',

'', '''WRITE''', '', 0, 0, 0, 0, 0, 'width: 300px; height: 80px;', ''

****

Class attributes for the employee class

After adding the attributes for a class we can complete the class definition by adding the title, multi look and single look information. For these three pieces of information we will provide the following:

Title description:

[FirstName] + ' ' + [LastName]

Multi look description: Single look description:

|  |  |
| --- | --- |
| <header>  <th>FirstName</th>  <th>LastName</th>  <th>PIN</th>  </header>  <sort>  <th>[LastName]</th>  <th>[FirstName]</th>  <th></th>  </sort>  <filter>  <th>[LastName]</th>  <th>[FirstName]</th>  <th>[PIN]</th>  </filter>  <tr>  <td>[LastName]</td>  <td>[FirstName]</td>  <td>[PIN]</td>  </tr> | <h3>Identification</h3>  <table>  <tr>  <td>First name:</td>  <td>[FirstName]</td>  </tr>  <tr>  <td>Last name:</td>  <td>[LastName]</td>  </tr>  <tr>  <td>PIN:</td>  <td>[PIN]</td>  </tr>  </table>  <h3>Birth Date and Place</h3>  <table>  <tr>  <td>Birthdate:</td>  <td>[Birthdate]</td>  </tr>  <tr>  <td>Birth place:</td>  <td>[BirthPlace]</td>  </tr>  </table>  <h3>Residence</h3>  [Address] |

We can add this information for the Employee class by using the Class\_UpdateClass stored procedure. See the script bellow:

EXEC Class\_UpdateClass 3, 0, 'Images/Icons/User.png', 'Employee', 'Employee', 'Employees', '',

'[FirstName] + '' '' + [LastName]',

'<h3>Identification</h3>

<table>

<tr><td>First name:</td><td>[FirstName]</td></tr>

<tr><td>Last name:</td><td>[LastName]</td></tr>

<tr><td>PIN:</td><td>[PIN]</td></tr>

</table>

<h3>Birth Date and Place</h3>

<table>

<tr><td>Birthdate:</td><td>[Birthdate]</td></tr>

<tr><td>Birth place:</td><td>[BirthPlace]</td></tr>

</table>

<h3>Residence</h3>

[Address]',

'<header><th>FirstName</th><th>LastName</th><th>PIN</th></header>

<sort><th>[LastName]</th><th>[FirstName]</th><th></th></sort>

<filter><th>[LastName]</th><th>[FirstName]</th><th>[PIN]</th></filter>

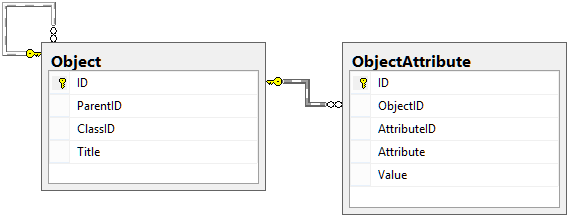
<tr><td>[LastName]</td><td>[FirstName]</td><td>[PIN]</td></tr>'

### Object Concept

After defining a class/entity the platform knows what attribute values to collect in order to save a valid class entry called in the following documentation “object”. For our Employee class, an object is composed of five strings and one date, each of these representing the value for the corresponding attribute. Thus, for an object of the employee class we may consider the following information:

|  |  |
| --- | --- |
| Attribute | Value |
| FirstName | Smith |
| LastName | John |
| PIN | 1840122323966 |
| Birthdate | 22/01/1984 |
| BirthPlace | Bucharest Sector 1 |
| Address | #61 Icoanei street, 1st floor |

In order to store this information, the platform uses the following table structure:



Objects storage data structure diagram

In the Object table we have the essential information regarding the object: unique identification key (ID), parent object’s unique identifier (ParentID) and object class reference (ClassID) and the object’s title (Title). In the ObjectAttribute table we have the values for the object attributes. For each object attribute we have a unique identifier (ID), an object reference (ObjectID), an attribute reference (AttributeID), the attribute name duplicated just for fast query processing when needed (Attribute) and the attribute’s value stored as a string.

In order to save an Employee object with the data shown before we can use the following script:

EXEC Object\_InsertObject 3, 0,

'<Object>

<Attribute Name="FirstName" Value="Smith" />

<Attribute Name="LastName" Value="John" />

<Attribute Name="PIN" Value="1840122323966" />

<Attribute Name="Birthdate" Value="22/01/1984" />

<Attribute Name="BirthPlace" Value="Bucharest Sector 1" />

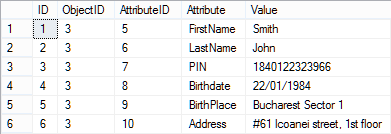
<Attribute Name="Address" Value="#61 Icoanei street, 1st floor" />

</Object>', 1, NULL

After running the above script we get the following results regarding the Object and ObjectAttribute tables.



Object table result



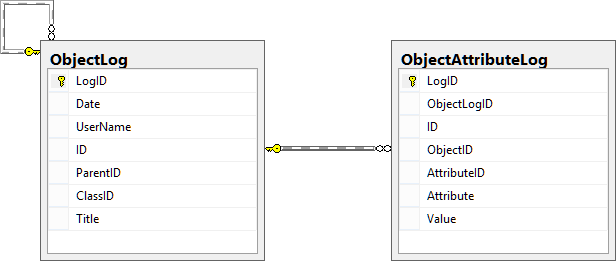
ObjectAttribute table result

As you can see in the script used to add the Employee object to the data collection we used an XML string to pass all the attributes’ values through a single parameter. In the chapters before we mentioned that some stored procedure parameters are “specific XML texts”. It is now time to detail a bit this simple XML structure.

The specific XML structure used to pass multiple values through a single parameter has a single root node called “Object” which has no XML attributes and contains all the Attribute XML nodes. An Attribute XML node has the two mandatory XML attributes and cannot have any child nodes. The two mandatory XML attributes are, as you can see, Name and Value, named accordingly to what they represent.

NOTE: any XML string passed as a parameter for the SQL stored procedures will have this XML structure.

Another issue that we must cover here is the log concept. Every time an object is created, altered or deleted, the platform will generate a log for that operation. The log will save some extra information regarding the user that triggered the log operation and the time and date when the log operation was triggered. For the logging part we have the following diagram. Keep in mind that the object is logged along with its attributes values.



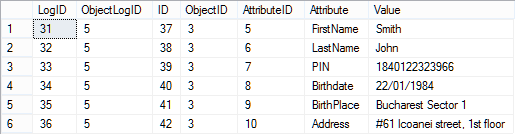
The object logging structure diagram

As you can see, on top of the regular object information we have a couple of more information. The primary keys for these tables are named LogID. For the ObjectLog table we have the information regarding the user that triggered the log and the log date (Date and UserName fields) and for the ObjectAttributeLog we have a new field that references the log for which the log attribute is for. When an object is logged, all of its attributes are logged as well, so we need to know all the attributes value for the object at that time.

For our Employee object that we inserted via the script illustrated before, we have the following information in the log tables.



ObjectLog table data



ObjectAttributeLog table data

Now, we will run the following script that changes the Employee’s first name from “Smith” to “Doe”. After we will present the new data found in the log tables just to illustrate the mechanism of the platform.

EXEC Object\_UpdateObject 3,

'<Object>

<Attribute Name="FirstName" Value="Doe" />

<Attribute Name="LastName" Value="John" />

<Attribute Name="PIN" Value="1840122323966" />

<Attribute Name="Birthdate" Value="22/01/1984" />

<Attribute Name="BirthPlace" Value="Bucharest Sector 1" />

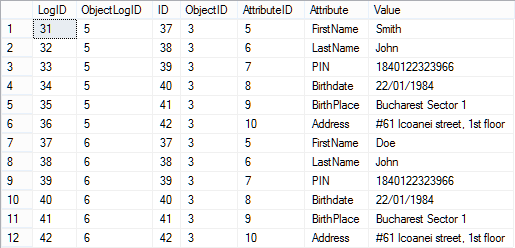
<Attribute Name="Address" Value="#61 Icoanei street, 1st floor" />

</Object>', 1

EXEC Object\_LogObject 3,1



New ObjectLog table data



New ObjectAttributeLog table data

### Mapping Concept

As you can observe, the objects have some default attributes that are not defined by the user. The default attributes are the following: ID, ParentID and Title. The ID represents the primary key and the title is the processed title according to the title specifications from the class. The default attribute that has a very important role for this part is the ParentID attribute.

The ParentID attribute represents the parent object for an object, thus giving the data a hierarchical tree structure. On top of this attribute we can define attributes that refer different object and this gives the tree structure a graph-like structure. The only problem with these relations is that they are one-to-many[[9]](#footnote-9) relations and we may need a many-to-many[[10]](#footnote-10) relation from time to time. A many-to-many relation is done using an intermediate table that pairs primary key values for the liked objects. In our case we can work around by creating an intermediate class but this has some limitations and may be rendered inefficient.

For a simple many-to-many relation we can use the mapping module. This module simulates the intermediate table used to have many-to-many relations, but on top of that it can hold two different types on relations: one-way and both-ways, thus giving the data an oriented graph structure.

Similar to the object module, this module logs any mapping created or deleted between two objects. The alteration of a mapping is at the moment unusable because you do not have anything to change for a mapping except the object references and by doing that we obtain a new mapping.

### Cache Concept

As you saw before, all the data regarding the saved objects is stored into two tables which makes it very hard to retrieve in case we want a list with all the objects of a certain class. In the above example, if we want a list with all the Employee objects and convert each attribute value to its corresponding data type we will need an SQL script that looks something like this:

SELECT O.ID,O.ParentID,O.Title,

OA1.Value AS FirstName,

OA2.Value AS LastName,

OA3.Value AS PIN,

CONVERT(DATETIME,OA4.Value,103) AS Birthdate,

OA5.Value AS BirthPlace,

OA6.Value AS [Address]

FROM [Object] AS O

INNER JOIN ObjectAttribute AS OA1 ON O.ID=OA1.ObjectID

AND OA1.Attribute='FirstName'

INNER JOIN ObjectAttribute AS OA2 ON O.ID=OA1.ObjectID

AND OA2.Attribute='LastName'

INNER JOIN ObjectAttribute AS OA3 ON O.ID=OA1.ObjectID

AND OA3.Attribute='PIN'

INNER JOIN ObjectAttribute AS OA4 ON O.ID=OA1.ObjectID

AND OA4.Attribute='Birthdate'

INNER JOIN ObjectAttribute AS OA5 ON O.ID=OA1.ObjectID

AND OA5.Attribute='BirthPlace'

INNER JOIN ObjectAttribute AS OA6 ON O.ID=OA1.ObjectID

AND OA6.Attribute='Address'

WHERE O.ClassID=3

The above script is fairly simple and retrieves the needed data, but if we are faced with a class with lots of attributes then we have a lot of joins to write and this complicates things. On top of that, considering that the Object and ObjectAttribute tables will grow in data over time, the above script will run slower and slower and it will become inefficient. We also have to consider what happens if we want to join data from two entities, we will be faced with an SQL script that joins two sub-query results which are not indexed.



The result for the above SQL script

To solve this problem we introduced in the platform the caching module. This module keeps the data for each class in a separated pre-processed table that looks like the result for the above SQL script. The table that retains the data for the Employee class will be called “CacheTable3”, where “CacheTable” is the generic prefix for cache tables and “3” represents the primary key value for the Employee class. Thus, we can replace the above SQL script with the following script that will have the same result:

SELECT \* FROM CacheTable3

Simpler and much faster to write and execute.

NOTE: The cache table is always kept up to date, but in the case of data missing or old data we have the Cache\_ProcessClass stored procedure at our disposal, or we can use the Utility\_ResetCache stored procedure if we want to reprocess the cache for all the classes.

Regarding the logs for the Employee class objects, the cache module generates a corresponding log table. In our case, the log table associated with the Employee class will be named “CacheTable3\_Log”. Notice that the only thing that changed in the log table’s name is the “\_Log” suffix to determine that this is the log table for the “CacheTable3” table. At this point the log table holds the following information:



The cache log table data for the Employee class

Also, the cache log tables are kept up to date, but in the case of a sync issue we can trigger the Cache\_ProcessClass stored procedure to re-cache the data for the needed class. The Cache\_ProcessClass stored procedure re-caches the log as well.

### Event Concept

As you are aware, multiple data storage applications are faced with the issue regarding the data integrity. If we save bad data we get bad results and we want to filter all the bad data before it is even saved. On top of that, for some data that we save, we need to update or insert other data. Because we wanted to keep things as generic and flexible as possible we thought that events can help developers to solve the above two issues.

The platform supports the following event set:

|  |  |
| --- | --- |
| Event Name | Description |
| PreInsert | Is triggered when a user opens an object insert form of any kind. Its role is to prevent the user from saving data if he is not allowed to do that. |
| BeforeInsert | Is triggered after the user entered the data for a new object and starts the save procedure. Its main role is to validate data before it is even saved. |
| AfterInsert | Is triggered after the data for a new object was saved successfully and its main role is to alter other objects if needed. |
| NotifInsert | Is triggered after the AfterInsert event and it is used to process a set of notifications that must be presented to the interested users. |
| BeforeUpdate | Is triggered after the user entered the new data for an existing object and starts the save procedure. Its main role is to validate data before it is even saved. |
| AfterUpdate | Is triggered after the data for an existing object was saved successfully and its main role is to alter other objects if needed. |
| NotifUpdate | Is triggered after the AfterUpdate event and it is used to process a set of notifications that must be presented to the interested users. |
| BeforeDelete | Is triggered when a user starts a delete operation. Its main purpose is to validate the delete operation. |
| AfterDelete | Is triggered after an object was deleted, in case that other data needs to be updated accordingly. |
| NotifDelete | Is triggered after the AfterDelete event and it is used to process a set of notifications that must be presented to the interested users. |
| OnSelect | This, so called event, is used when computing specific SQL queries. Its main purpose is to limit the returned objects for each class according to the specified user’s access rights. |
| BeforeMappingInsert | It is triggered before a mapping between two objects is established. Its main purpose is to validate the mapping. |
| AfterMappingInsert | It is triggered after a mapping was created between two objects. |
| NotifMappingInsert | It is triggered after the AfterMappingInsert event and it is used to process a set of notifications that must be presented to the interested users. |
| BeforeMappingDelete | It is triggered before the mapping between two objects is deleted in order to validate the delete operation. |
| AfterMappingDelete | It is triggered after a mapping between two objects was deleted. |
| NotifMappingDelete | It is triggered after the AfterMappingDelete event and it is used to process a set of notifications that must be presented to the interested users. |

A platform event is computed for any object of a class if it is defined. An event consists in a specific SQL query that is computed for an object when the event is triggered. The SQL query attached to an event respects the SQL language except a few kinds of tokens that may appear and are replaced with valid SQL code before running. The tokens that may appear in an event’s query represent language tags, object attributes and class object collections.

**Language Tag Tokens**

If a query contains a language tag, the language tag will be enclosed using the “[#” and “]” sequences. Keep in mind that a language tag will be replaced with the value of the language tag for the needed language and it may generate errors if it is not enclosed using the “ ’ ” characters. For example, we may have a line of code inside the event’s query that can look like this:

SET @Var='[#language\_tag]'

If the value for the specified language tag is “xxx” the above line will be transformed as follows:

SET @Var='xxx'

**Object Attribute Tokens**

An event query is run in the context of one object so we are able to use in the query tokens for the attributes values. For our Employee example we can use the value for the FirstName attribute by enclosing it in square brackets (like “[FirstName]”). These kinds of tokens are replaced using the Formula field’s value from the ClassAttributeType table and the attribute value. Considering the bellow example:

SET @Var1=[FirstName]

SET @Var2=[Birthdate]

This example will be translated in the following form for our object:

SET @Var1=CONVERT(NVARCHAR(32),N'Doe')

SET @Var2=CONVERT(VARCHAR,NULLIF('22/01/1984',''),103)

**Class Object Collection Tokens**

In order to confront data with object collections we introduced the ability to specify the needed collection/class enclosed in braces and use it as a regular SQL table. Working on our example we can write SQL commands that look like the following:

SELECT \*

FROM {Employee} AS E

SELECT \*

FROM {Employee#} AS E

As you can see, in the second SELECT statement we placed a “#” character after the class’s name. This signals the platform’s interpreter to ignore the OnSelect limitation on the Employee class (if any) and retrieve all the rows. For the first SELECT statement the OnSelect limitation will be used (if any) and for the translation we will consider that there is a limitation placed on the Employee class.

CREATE TABLE #IDs3(ID INT)

SELECT @Cmd=REPLACE(ET.Command,'[@UserID]',CONVERT(NVARCHAR,@UserID))

FROM EventTrigger AS ET WHERE ET.ID=CONVERT(NVARCHAR,ET.ID)

SELECT @Cmd=REPLACE(@Cmd,'{'+[Name]+'}','CacheTable'+CONVERT(VARCHAR,ID))

FROM Class

EXEC (@Cmd)

INSERT INTO #IDs3

SELECT ID

FROM #IDs

TRUNCATE TABLE #IDs

SELECT \*

FROM (SELECT \* FROM CacheTable3 WHERE ID IN (SELECT ID FROM #IDs3)) AS E

DROP TABLE #IDs3

SELECT \*

FROM CacheTable3 AS E

As you can see, the second SELECT statement was translated as a simple select from the CacheTable3. However, the first SELECT statement was modified into something more different. The OnSelect restriction returns the list of IDs on which the user has access and places them into a temporary table created outside the scope called “#IDs”. If we have multiple restricted tables it is not recommended to keep all the IDs together in the same table, so for each class the interpreter will create a new temporary table composed from the “#IDs” prefix and the primary key value for the class. After this, the OnSelect restriction statement is prepared and executed storing the result in the #IDs table. The next step is to move the results in the particular temporary restriction table and truncate the general #IDs table. Before dropping the particular temporary restriction table the transformed SELECT statement is executed with the restriction in place.

NOTE: The general #IDs temporary restriction table is created and deleted outside the event scope.

## Conclusions

This chapter described the database layer detailing the tables and the relations between them. Also the user defined functions and stored procedures are detailed briefly. For more details regarding the user defined functions and stored procedures you may consult the online help which will be updated permanently to cover the changes.

The specific XML, HTML and SQL syntaxes will be detailed in further chapters when they are more relevant, but for the time being they were shown just as a sample in order to understand some of the stored procedures and to prepare you for understanding their effects.

# Communication Layer

This layer consists in a separate C# project that handles the communication between the business layer and the database layer. Also, this layer handles email and SMS notifications and holds the global settings for the platform.

We will start the presentation with the easiest class and continue with the rest of them in no particular order.

## Settings Class

This class holds the operational settings for the platform. At this point in holds the settings regarding the database connection string and the email server data and credentials. This class is a static class and doesn’t have any methods.

### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ConnectionString | Constant string | Holds the connection string which is used to configure the database server connection. |
| Mail\_SMTP | Constant string | Holds the SMTP server’s address which will be responsible with the notification sending. |
| Mail\_Box | Constant string | Holds the email box through which the notifications will be sent. |
| Mail\_UserName | Constant string | Holds the user name in order to authenticate with the SMTP server. |
| Mail\_Password | Constant string | Holds the password in order to authenticate with the SMTP server. |

The Mail\_userName and Mail\_Password may be left blank if the specified SMTP server does not require authentication.

It is recommended that this class should hold the SMS service settings. At this point they are missing because the platform comes with a dummy SMS sending method. This is because each implementation over this platform that uses a web2sms service must be configured accordingly to the used service.

## SMS Class

This class is static and contains a single dummy procedure that is called by the platform when needed. This dummy method must be rewritten in order to integrate the platform with a web2sms service. If the application developed over this platform does not require a service like this, the SMS class must be left as it is in order to avoid complications.

### Methods

**Send**

Purpose: sends the supplied text as an SMS to the provided phone number.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Phone | String | The phone number that will receive the SMS. |
| Message | String | The SMS text. |

## eMail Class

This is a static class responsible for the notifications sending via email. Just like the SMS class, this contains a single method but it is fully written.

### Methods

**Send**

Purpose: sends the supplied notification via email.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| To | String | The list of email addresses for which the message is destined separated by the “;” character. |
| BCC | String | The list of blind courtesy copy recipients separated by the “;” character. |
| CC | String | The list of courtesy copy recipients separated by the “;” character. |
| Subject | String | The email’s subject. |
| Body | String | The email message encoded as HTML. |
| Attachments | String | The list of full file paths for the files that need to be attached to the email separated by the “;” character. |

## Dispatch Class

This is a static class responsible for dispatching the messages via email and/or SMS.

### Methods

**DispatchNotifications**

Purpose: dispatches the received notifications and sends them via the designated channels.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| DT | System.Data.DataTable | A data table containing the notifications. The columns for this table are: eMail, Phone, Subject, Message, Link (the representative link for the notification), WithSMS (int used as Boolean, specifies that the notification must sent via SMS), WitheMail (int used as Boolean, specifies that the notification must sent via email). |

## Database Class

Considering how the database layer looks like and how it is structured, the interface has to trigger stored procedures called in order to interact with the data. For all the applications of this kind, in order to call a stored procedure from the database a connection has to be opened for the server, the procedure must be called with the needed list of parameters, the result must be returned and stored and finally close the connection for the server. To write similar code over and over in order to call stored procedures is time consuming so we developed a single class in the platform that handles stored procedure calls.

Just as a heads up, this communication layer is a single C# class found in the project in the Communication project under the path DataBase\DataBase.cs and it is referenced by the BusinessLogic project. As a logical path we can find this class in the Amaryllis.DataBase namespace.

The DataBase class contains a short set of methods that are used in order to interact with the database layer through the stored procedures and it is static in order for it to be shared and work with all the classes from the business layer without generating multiple database connections and overwhelm the database server. Without any delays we will detail these methods bellow.

### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GlobalConnection | System.Data.SqlClient.SqlConnection | Holds an SQL open connection to the database server. If this property is null then a connection will be opened and close when a transaction is required. |

### Methods

**OpenConnection**

Purpose: initializes the GlobalConnection member.

Returns: NOTHING

Parameters: NONE

**CloseConnection**

Purpose: disposes of the GlobalConnection member.

Returns: NOTHING

Parameters: NONE

**ExecuteCommand**

Purpose: connects to the database server and executes the given stored procedure with the give parameters.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ProcedureName | String | The name of the stored procedure to be executed. |
| Params | object[] (optional) | An object vector containing pairs of stored procedure parameter name and value. Default value is “null”. |

## Conclusions

This layer is used by the business layer directly and helps the business layer communicate with the database layer and outside systems (email and SMS). Because of this layer we consider the platform to be a four-layer module and you can see this in Annexe 2.

# Business Layer

This layer represents the link between the communication layer and the interface layer and it handles data validation before sending it to the communication layer. The composition for this layer is based on C# classes, mostly static, which relay to a module in the data layer or that are used as utilities in order to display information or group information in a standardized design item.

The classes for this layer are grouped in namespaces according to their functionality, so for this layer we will detail the information according to namespace appurtenance.

## Amaryllis Namespace

### Settings Class

The main purpose for this class is to hold information regarding the platform’s version and version tag. At this point there are no other settings to be retained here. In case of other settings regarding custom modules, is best to store them here just for the generality of it all.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Version | Constant string | Holds the full version of the platform. |
| VersionTag | Constant string | Holds the tag for the platform. |

### Page Class

Derived from the System.Web.UI.Page class, it is used as base for every web page in the client interface. Its main purpose is to set the interface language and keep a database connection opened until the web page ends its execution.

#### Methods

**Page**

Purpose: this is the class constructor and it links the Page\_PreLoad method to the page PreLoad event and the Page\_Unload method to the Unload, Error and AbortTransaction events.

Returns: NOTHING

Parameters: NONE

**Page\_PreLoad**

Purpose: sets the language interface according to the current user or default value, opens a database connection checks the current user’s access rights to the accessed page redirecting him if his access is denied.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| sender | object | The object which triggered the event. |
| e | EventArgs | A structure containing the event arguments. |

**Page\_Unload**

Purpose: closes the database connection opened by the Page\_PreLoad method.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| sender | object | The object which triggered the event. |
| e | EventArgs | A structure containing the event arguments. |

## Amaryllis.Controls Namespace

This namespace contains all the classes regarding user controls through which database object attributes values can be collected.

### Bar Control

This is a display only control and can’t connect any value and it is mostly used to display a value graphically.



Bar control display

#### Methods

**GetDisplay**

Purpose: retrieves the HTML code that displays the bar control.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | float | The value that needs to be represented. |
| Max | Int | The maximum value for the bar control. |
| Width | Int | The maximum width in pixels for the bar control. |

**Display**

Purpose: writes the HTML code that displays the bar control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | float | The value that needs to be represented. |
| Max | Int | The maximum value for the bar control. |
| Width | Int | The maximum width in pixels for the bar control. |

### CheckBox Control



Check box control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the check box control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |
| ReadOnly | Bool (optional) | Specifies if the control value is read only or not. Default value if “false”. |
| Checked | Amaryllis.Controls.CheckBoxValue (optional) | Specified the control’s state. Default value is Amaryllis.Controls.CheckBoxValue.Undefined. |

**GetBoxValue**

Purpose: retrieves a check box control’s value.

Returns: Amaryllis.Controls.CheckBoxValue

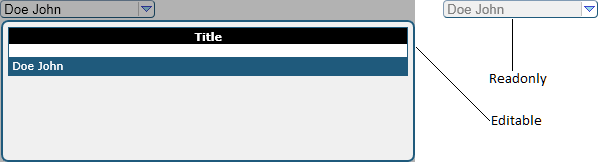
Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/CheckBox/Checked.png | Image |
| Images/Controls/CheckBox/CheckedError.png | Image |
| Images/Controls/CheckBox/CheckedReadonly.png | Image |
| Images/Controls/CheckBox/Unchecked.png | Image |
| Images/Controls/CheckBox/UncheckedError.png | Image |
| Images/Controls/CheckBox/UncheckedReadonly.png | Image |
| Scripts/Controls/CheckBox.js | Script |

### ComboBox Control



Combo box control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the combo box control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |
| DT | System.Data.DataTable | The data source for the control. |
| ShownCells | string | The list of columns that must be displayed in the drop list separated by the “,” character. |
| ColumnHeaders | string | The captions for the columns displayed by the drop list separated by the “,” character. |
| DisplayColumnID | string | The name of the column that represents the value to be displayed for the selected row. |
| ValueColumnID | string | The name of the column that contains the value needed for the attribute. |
| ReadOnly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| Width | Int | The width in pixels of the drop box. |
| Height | Int | The height in pixels of the drop box. |
| Style | string | CSS style for the combo box. |
| SelectedValue | String (optional) | The value that must be selected by default. |

**GetBoxValue**

Purpose: retrieves a combo box control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/ComboBox/Arrow.png | Image |
| Scripts/Controls/ComboBox.js | Script |

### Controls Class

This class links all the controls in order to minimize switches and cases from the code.

#### Methods

**DisplayReportControl**

Purpose: calls the appropriate Display function for the specified control. This method is used in order to display report parameter forms.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Control | String | The name of the control. |
| FieldName | String | Represents the control’s HTML id. |
| Width | Int | The width in pixels of the control. |
| Height | Int | The height in pixels of the control. |
| Min | Float | The minimum value for the control. |
| Max | Float | The maximum value for the control. |
| Unit | Float | The rounding unit for the control’s value. |
| Style | String | The CSS style of the control. |
| Value | String | The default value for the control. |
| ParameterID | Int | The primary key value for the report parameter. |

**DisplayObjectControl**

Purpose: calls the appropriate Display function for the specified control. This method is used in order to display report parameter forms.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Control | String | The name of the control. |
| FieldName | String | Represents the control’s HTML id. |
| IsForeignKey | bool | Specifies the fact that the control represents a foreign key attribute. |
| IsReadonly | bool | Specifies if the control will be displayed in read-only mode or not. |
| Width | Int | The width in pixels of the control. |
| Height | Int | The height in pixels of the control. |
| Min | Float | The minimum value for the control. |
| Max | Float | The maximum value for the control. |
| Unit | Float | The rounding unit for the control’s value. |
| Style | String | The CSS style of the control. |
| Value | String | The value for the control. |
| DisplayValue | String | The value that the control should display. |
| DefaultValue | String | The default value for the control. |
| ParentID | Int | The primary key value for the parent object. |
| ObjectID | Int | The primary key value of the object. |
| AttributeID | Int | The primary key value of the class attribute for which the control is displayed. |

**GetValue**

Purpose: retrieves the value for the named control.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Control | String | The name of the control. |
| FieldName | String | Represents the control’s HTML id. |
| IsForeignKey | bool | Specifies the fact that the control represents a foreign key attribute. |

**GetDummyValue**

Purpose: retrieves the dummy value for the named control.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Control | String | The name of the control. |
| FieldName | String | Represents the control’s HTML id. |
| IsForeignKey | bool | Specifies the fact that the control represents a foreign key attribute. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Controls/Controls.resx | Resource |
| Properties/Controls/Controls.RO-RO.resx | Resource |
| Scripts/Controls/Controls.js | Script |

### DatePicker Control



Date picker control display

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Months | String[] | A vector containing the calendar months’ names. The first value is a sentinel. |
| MonthDays | Int[] | A vector containing the number of days of each calendar month. The first value is a sentinel. |

#### Methods

**Display**

Purpose: writes the HTML code that displays the date picker control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |
| Readonly | Bool (optional) | Specifies if the control value is read only or not. Default value if “false”. |
| Date | String (optional) | The default value to be displayed by the control. |

**GetValue**

Purpose: retrieves a date picker control’s value.

Returns: string

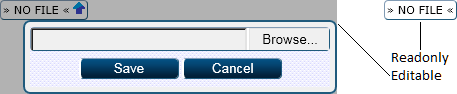
Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Controls/DatePicker.resx | Resource |
| Properties/Controls/DatePicker.RO-RO.resx | Resource |
| Images/Controls/DatePicker/Calendar.png | Image |
| Images/Controls/DatePicker/Next.png | Image |
| Images/Controls/DatePicker/Previous.png | Image |
| Scripts/Controls/DatePicker.js | Script |
| Scripts/Resources/EN-UK/Controls\_DatePicker.js | Script |
| Scripts/Resources/RO-RO/Controls\_DatePicker.js | Script |

### FileUpload Control



File upload control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the file upload control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| FileID | String | The default file to be displayed by the control. |

**GetValue**

Purpose: retrieves a file upload control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Controls/FileUpload.resx | Resource |
| Properties/Controls/FileUpload.RO-RO.resx | Resource |
| App\_GlobalResources/Pages/Controls\_FileUpload/Page\_Controls\_FileUpload.resx | Resource |
| App\_GlobalResources/Pages/Controls\_FileUpload/Page\_Controls\_FileUpload.RO-RO.resx | Resource |
| Images/Controls/FileUpload/Download.png | Image |
| Images/Controls/FileUpload/Upload.png | Image |
| Scripts/Controls/FileUpload.js | Script |
| Controls\_FileUpload.aspx | Web Page |

### IconSelector Control



Icon selector control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the icon selector control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Width | Int | Specifies the width in pixels of the control. |
| Value | String (optional) | The default icon to be displayed by the control. |

**GetValue**

Purpose: retrieves the icon selector control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Icons | Folder |

### ImageUpload Control



Image upload control display

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| NoImageID | Int | Contains the primary key value of the default image to be displayed. |

#### Methods

**Display**

Purpose: writes the HTML code that displays the image upload control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| Width | Int | The width of the control in pixels. |
| Height | Int | The height of the control in pixels. |
| FileID | String | The default file to be displayed by the control. |

**GetValue**

Purpose: retrieves the image upload control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Controls/ImageUpload.resx | Resource |
| Properties/Controls/ImageUpload.RO-RO.resx | Resource |
| App\_GlobalResources/Pages/Controls\_ImageUpload/Page\_Controls\_ImageUpload.resx | Resource |
| App\_GlobalResources/Pages/Controls\_ImageUpload/Page\_Controls\_ImageUpload.RO-RO.resx | Resource |
| Images/Controls/ImageUpload/NoImage.png | Image |
| Scripts/Controls/ImageUpload.js | Script |
| Controls\_ImageUpload.aspx | Web Page |

### Integer Control



Interger control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the integer control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| Min | Int | The minimum value for the control’s value. |
| Max | Int | The maximum value for the control’s value. |
| Value | Int | The default value for the control. |

**GetValue**

Purpose: retrieves the integer control’s value.

Returns: int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/Integer/Minus.png | Image |
| Images/Controls/Integer/Plus.png | Image |
| Scripts/Controls/Integer.js | Script |

### MultiCheck Control



Multi check control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the multi check control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| Source | System.Data.DataTable | The source data table for the control. |
| OptionColumnID | String | The name of the column representing the option. |
| Value | String | The default value for the control. |

**GetValue**

Purpose: retrieves the multi check control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/MultiCheck/Checked.png | Image |
| Images/Controls/MultiCheck/CheckedError.png | Image |
| Images/Controls/MultiCheck/CheckedReadonly.png | Image |
| Images/Controls/MultiCheck/Unchecked.png | Image |
| Images/Controls/MultiCheck/UncheckedError.png | Image |
| Images/Controls/MultiCheck/UncheckedReadonly.png | Image |
| Scripts/Controls/MultiCheck.js | Script |

### Password Control

****

Password control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the password control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| Style | String | The CSS style for the control. |
| Value | String (optional) | The default value for the control. |

**GetValue**

Purpose: retrieves the password control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/Password/0.png | Image |
| Images/Controls/Password/1.png | Image |
| Images/Controls/Password/2.png | Image |
| Images/Controls/Password/3.png | Image |
| Images/Controls/Password/4.png | Image |
| Images/Controls/Password/5.png | Image |
| Images/Controls/Password/6.png | Image |
| Images/Controls/Password/7.png | Image |
| Images/Controls/Password/8.png | Image |
| Images/Controls/Password/9.png | Image |
| Images/Controls/Password/10.png | Image |
| Images/Controls/Password/11.png | Image |
| Images/Controls/Password/12.png | Image |
| Images/Controls/Password/13.png | Image |
| Images/Controls/Password/14.png | Image |
| Images/Controls/Password/15.png | Image |
| Images/Controls/Password/16.png | Image |
| Images/Controls/Password/17.png | Image |
| Images/Controls/Password/18.png | Image |
| Images/Controls/Password/19.png | Image |
| Images/Controls/Password/20.png | Image |
| Images/Controls/Password/21.png | Image |
| Images/Controls/Password/22.png | Image |
| Images/Controls/Password/23.png | Image |
| Images/Controls/Password/24.png | Image |
| Images/Controls/Password/25.png | Image |
| Images/Controls/Password/26.png | Image |
| Images/Controls/Password/27.png | Image |
| Images/Controls/Password/28.png | Image |
| Images/Controls/Password/29.png | Image |
| Images/Controls/Password/30.png | Image |
| Images/Controls/Password/31.png | Image |
| Images/Controls/Password/32.png | Image |
| Images/Controls/Password/33.png | Image |
| Images/Controls/Password/34.png | Image |
| Images/Controls/Password/35.png | Image |
| Images/Controls/Password/36.png | Image |
| Images/Controls/Password/37.png | Image |
| Images/Controls/Password/38.png | Image |
| Images/Controls/Password/39.png | Image |
| Images/Controls/Password/40.png | Image |
| Images/Controls/Password/41.png | Image |
| Images/Controls/Password/42.png | Image |
| Images/Controls/Password/43.png | Image |
| Images/Controls/Password/44.png | Image |
| Images/Controls/Password/45.png | Image |
| Images/Controls/Password/46.png | Image |
| Images/Controls/Password/47.png | Image |
| Images/Controls/Password/48.png | Image |
| Images/Controls/Password/49.png | Image |
| Images/Controls/Password/50.png | Image |
| Images/Controls/Password/51.png | Image |
| Images/Controls/Password/52.png | Image |
| Images/Controls/Password/53.png | Image |
| Images/Controls/Password/54.png | Image |
| Images/Controls/Password/55.png | Image |
| Images/Controls/Password/56.png | Image |
| Images/Controls/Password/57.png | Image |
| Images/Controls/Password/58.png | Image |
| Images/Controls/Password/59.png | Image |
| Images/Controls/Password/60.png | Image |
| Images/Controls/Password/61.png | Image |
| Images/Controls/Password/62.png | Image |
| Images/Controls/Password/63.png | Image |
| Images/Controls/Password/64.png | Image |
| Images/Controls/Password/65.png | Image |
| Images/Controls/Password/66.png | Image |
| Images/Controls/Password/67.png | Image |
| Images/Controls/Password/68.png | Image |
| Images/Controls/Password/69.png | Image |
| Images/Controls/Password/70.png | Image |
| Images/Controls/Password/71.png | Image |
| Images/Controls/Password/72.png | Image |
| Images/Controls/Password/73.png | Image |
| Images/Controls/Password/74.png | Image |
| Images/Controls/Password/75.png | Image |
| Images/Controls/Password/76.png | Image |
| Images/Controls/Password/77.png | Image |
| Images/Controls/Password/78.png | Image |
| Images/Controls/Password/79.png | Image |
| Images/Controls/Password/80.png | Image |
| Images/Controls/Password/81.png | Image |
| Images/Controls/Password/82.png | Image |
| Images/Controls/Password/83.png | Image |
| Images/Controls/Password/84.png | Image |
| Images/Controls/Password/85.png | Image |
| Images/Controls/Password/86.png | Image |
| Images/Controls/Password/87.png | Image |
| Images/Controls/Password/88.png | Image |
| Images/Controls/Password/89.png | Image |
| Images/Controls/Password/90.png | Image |
| Images/Controls/Password/91.png | Image |
| Images/Controls/Password/92.png | Image |
| Images/Controls/Password/93.png | Image |
| Images/Controls/Password/94.png | Image |
| Images/Controls/Password/95.png | Image |
| Images/Controls/Password/96.png | Image |
| Images/Controls/Password/97.png | Image |
| Images/Controls/Password/98.png | Image |
| Images/Controls/Password/99.png | Image |
| Images/Controls/Password/100.png | Image |
| Scripts/Controls/Password.js | Script |

### RadioGroup Control



Radio group control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the radio group control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. Default value if “false”. |
| Source | System.Data.DataTable | The source data table for the control. |
| OptionColumnID | String | The name of the column representing the option. |
| Value | String | The default value for the control. |

**GetValue**

Purpose: retrieves the radio group control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/RadioGroup/Checked.png | Image |
| Images/Controls/RadioGroup/CheckedError.png | Image |
| Images/Controls/RadioGroup/CheckedReadonly.png | Image |
| Images/Controls/RadioGroup/Unchecked.png | Image |
| Images/Controls/RadioGroup/UncheckedError.png | Image |
| Images/Controls/RadioGroup/UncheckedReadonly.png | Image |
| Scripts/Controls/RadioGroup.js | Script |

### Rating Control

****

Rating control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the rating control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool (optional) | Specifies if the control value is read only or not. Default value if “false”. |
| Min | Int (optional) | The minimum value accepted by the control. |
| Max | Int (optional) | The maximum value accepted by the control. |
| Value | String (optional) | The default value for the control. |

**GetValue**

Purpose: retrieves the rating control’s value.

Returns: string

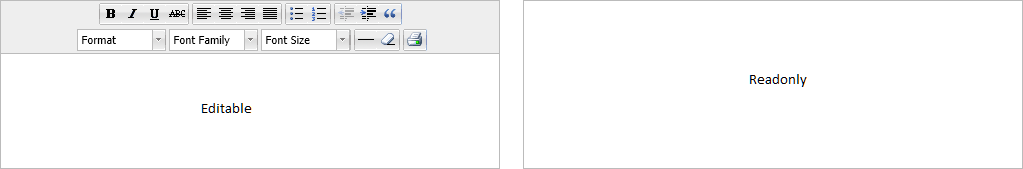
Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Controls/Rating/Empty.png | Image |
| Images/Controls/Rating/EmptyBlink.png | Image |
| Images/Controls/Rating/Full.png | Image |
| Images/Controls/Rating/FullBlink.png | Image |
| Scripts/Controls/Rating.js | Script |

### RichTextArea Control



Rich text area control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the rich text area control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. |
| Style | String (optional) | The CSS style of the control. |
| Value | String (optional) | The default value for the control. |

**GetValue**

Purpose: retrieves the rich text area control’s value.

Returns: string

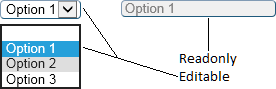
Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Controls/RichTextArea | Folder |

### Select Control



Select control display

#### Methods

**Display – 1st Form**

Purpose: writes the HTML code that displays the select control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |
| ReadOnly | Bool | Specifies if the control value is read only or not. |
| Source | String | The list of values for the select list separated by the “|” character. |
| Style | string | CSS style for the select control. |
| Value | String | The value that must be selected by default. |

**Display – 2nd Form**

Purpose: writes the HTML code that displays the select control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | string | Represents the control’s HTML id. |
| ReadOnly | Bool | Specifies if the control value is read only or not. |
| Source | System.Data.DataTable | The list of values for the select list. |
| DescriptionColumn | String | The name of the column representing the select list values. |
| IDColumn | String | The name of the column representing the values that will be passed as the attribute value. |
| Style | string | CSS style for the select control. |
| Value | String (optional) | The value that must be selected by default. |

**GetValue**

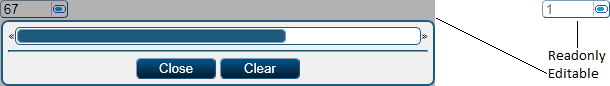
Purpose: retrieves the select control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

### Slider Control



Slider control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the slider control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. |
| Min | Float | The minimum value accepted by the control. |
| Max | Float | The maximum value accepted by the control. |
| Unit | Float | The incremental step for the control. |
| Value | String (optional) | The default value for the control. |

**GetValue**

Purpose: retrieves the slider control’s value.

Returns: float

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Controls/Slider.resx | Resource |
| Properties/Controls/Slider.RO-RO.resx | Resource |
| Images/Controls/Slider/Slider.png | Image |
| Scripts/Controls/Slider.js | Script |

### TextArea Control

****

Text area control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the text area control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. |
| Style | String | The CSS style for the control. |
| Value | String (optional) | The default value for the control. |

**GetValue**

Purpose: retrieves the text area control’s value.

Returns: float

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

### TextBox Control



Text box control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the text box control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool | Specifies if the control value is read only or not. |
| Style | String | The CSS style for the control. |
| Value | String (optional) | The default value for the control. |

**GetValue**

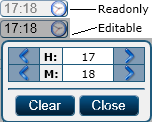
Purpose: retrieves the text box control’s value.

Returns: float

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

### TimePicker Control



Time picker control display

#### Methods

**Display**

Purpose: writes the HTML code that displays the time picker control on the desired web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |
| Readonly | Bool (optional) | Specifies if the control value is read only or not. The default value is “false”. |
| Time | String (optional) | The default value for the control. |

**GetValue**

Purpose: retrieves the time picker control’s value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | Represents the control’s HTML id. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Controls/TimePicker.resx | Resource |
| Properties/Controls/TimePicker.RO-RO.resx | Resource |
| Images/Controls/TimePicker/Clock.png | Image |
| Images/Controls/TimePicker/Next.png | Image |
| Images/Controls/TimePicker/Previous.png | Image |
| Scripts/Controls/TimePicker.js | Script |

## Amaryllis.Data Namespace

This module contains all the classes that are used to export data from the platform.

### CSV Class

This is a static class responsible for exporting data to CSV formatted files.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Separator | Char | The character used as a separator for cells. |
| Marker | Char | The character that delimits cell values. |

#### Methods

**DecodeNextToken**

Purpose: returns the next cell value from the specified CSV line shortening the line itself.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Line | String reference | The CSV line from which the token must be extracted. |

**EncodeNextToken**

Purpose: encodes the specified value in order to be written in a CSV file.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Token | String | The value that needs to be encoded. |

**ImportFromFile**

Purpose: imports the data from a CSV file in a data table.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FilePath | String | The full absolute path of the file from which the data must be imported. |

**ExportReportToFile**

Purpose: exports the data for the specified report to a CSV file.

Returns: string (representing the absolute full file path)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value for the report. |
| Parameters | String | The parameters for the report encoded in a specific XML structure. |
| LogDate | String | The point in time until the data must be processed in the “DD/MM/YYYY HH:MM” format. |

**ExportToFile**

Purpose: exports the specified data table to a CSV file.

Returns: string (representing the absolute full file path)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| DT | System.Data.DataTable | The data table that needs to be exported. |
| WithHeaders | Bool | Specifies if the data table header must be exported as well. |

### PDF Class

This is a static class responsible for exporting data to PDF formatted files.

#### Methods

**HTML2PDF**

Purpose: exports an HTML file to a PDF file.

Returns: string (representing the absolute full file path)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLFilePath | String | The full absolute path of the HTML file that must be converted in a PDF format. |

## Amaryllis.DataBase Namespace

### Article Class

This is the class assigned to the documentation module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**CheckArticleContent**

Purpose: checks if the article content is in the correct HTML format.

Returns: string (representing the error message)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Content | String | The article content that needs to be validated. |

**DeleteArticle**

Purpose: deletes the specified article and adjacent data.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**DeleteComment**

Purpose: deletes a single comment along with its sub-comments.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CommentID | Int | The primary key value of the comment. |

**DeleteLink**

Purpose: deletes the specified expression.

Returns: NONE

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LinkID | Int | The primary key value of the expression. |

**DeleteTagForArticle**

Purpose: deletes the specified tag.

Returns: NOTING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| TagID | Int | The primary key value of the tag. |

**GetArticleCache**

Purpose: computes, updates and returns the content and navigation cache for the specified article.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetArticlesForTreeID**

Purpose: returns the result of the Article\_GetArticlesForTreeID stored procedure.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| TreeID | Int | The primary key value of the tree node. |

**GetArticleInfo**

Purpose: retrieves the information regarding the specified article.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetCommentsForArticle**

Purpose: retrieves the list of comments for the specified article.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetFilesForArticle**

Purpose: retrieves the list of files from the virtual drive referred by the article.

Returns: System.Data.DataTable

Purpose:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetLinks**

Purpose: retrieves the list of all the expressions.

Returns: System.Data.DataTable

Parameters: NONE

**GetNewestArticle**

Purpose: returns the result of the Article\_GetNewestArticle stored procedure.

Returns: System.Data.DataTable

Parameters: NONE

**GetLinksForArticle**

Purpose: retrieves the list of expressions referred by the article.

Returns: System.Data.DataTable

Purpose:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetSimilarArticles**

Purpose: retrieves a short list of the articles that are similar to the specified article.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetTagsForArticle**

Purpose: retrieves the list of tags for the specified article.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**GetTopTags**

Purpose: retrieves the list of the most used article tags.

Returns: System.Data.DataTable

Parameters: NONE

**GetTree**

Purpose: returns the result of the Article\_Gettree stored procedure.

Returns: System.Data.DataTable

Parameters: NONE

**GetTreeNodeInfo**

Purpose: returns the result of the Article\_GetNodeInfo stored procedure.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| TreeID | Int | The primary key value of the tree node. |

**InsertArticle**

Purpose: inserts a new article.

Returns: int (representing the primary key value of the attribute)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The article’s title. |
| Content | String | The article’s content encoded in a specific HTML syntax. |
| Language | String | The language code for the language in which the article is written. |

**InsertComment**

Purpose: inserts a new comment for the specified article.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ParentID | Int | The primary key value of the parent comment. |
| ArticleID | Int | The primary key value of the article. |
| Comment | String | The comment encoded in pure HTML. |

**InsertLink**

Purpose: inserts a new expression.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |
| Expression | String | The expression itself. |
| Description | String | The description for the expression. |

**InsertTagForArticle**

Purpose: inserts the specified tag for the specified article.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |
| Tag | String | The tag’s name. |

**Search**

Purpose: retrieves the list of articles that match the given criteria.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Words | String | The search criteria. |

**UpdateArticleCache**

Purpose: updates the content cache and navigation cache for the specified article.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |
| Content | String | The content cache in pure HTML form. |
| Navigation | String | The navigation cache in pure HTML form. |

**UpdateLink**

Purpose: updates an expression and reset the cache for the articles containing that expression.

Returns: NOTING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LinkID | Int | The primary key value of the expression. |
| ArticleID | Int | The primary key value of the article. |
| Expression | String | The expression itself. |
| Description | String | The description for the expression. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Article.resx | Resource |
| Properties/DataBase/Article.RO-RO.resx | Resource |

### Class Class

This class represents the Class module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**DeleteAttribute**

Purpose: deletes the specified attribute.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AttributeID | Int | The primary key value of the attribute. |

**DeleteClass**

Purpose: deletes the specified class.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**DeleteLink**

Purpose: deletes a relation between two classes.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LinkID | Int | The primary key value of the class relation. |

**ExistsAttribute**

Purpose: checks if the specified attribute exists for the given class.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| Name | String | The name of the attribute. |
| ExceptID | Int | The primary key value of the attribute that must be excluded from the check. |

**ExistsClass**

Purpose: checks if the specified class exists.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The internal name of the class. |
| ExceptID | Int | The primary key value of the class that must be excluded from the check. |

**GetAttributeInfo**

Purpose: retrieves the information for the given attribute.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AttributeID | Int | The primary key value of the attribute. |

**GetAttributeTypes**

Purpose: retrieves the list of attribute types.

Returns: System.Data.DataTable

Parameters: NONE

**GetAttributesForClass**

Purpose: retrieves the list of attributes for the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetClassInfo**

Purpose: retrieves the information for the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetClasses**

Purpose: retrieves the list of classes.

Returns: System.Data.DataTable

Parameters: NONE

**GetClassesExt**

Purpose: retrieves the list of classes with extended information.

Returns: System.Data.DataTable

Parameters: NONE

**GetClassIDFromName**

Purpose: retrieves the class primary key value from its name.

Returns: int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetClassesForUser**

Purpose: retrieves the list of classes on which the current logged on user has access.

Returns: System.Data.DataTable

Parameters: NONE

**GetLanguageTagsForClass**

Purpose: retrieves the list of language tags for the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetMasterClassesForClass**

Purpose: retrieves the list of master classes for the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetRealClassInfo**

Purpose: retrieves the information for the given class with no translation applied.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetSlaveClassesForClass**

Purpose: retrieves the list of slave classes for the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetSlaveClassesForClass**

Purpose: retrieves the list of slave classes for the given class on which the logged on user has access to.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**InsertAttribute**

Purpose: inserts a new class attribute.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| Name | String | The name of the attribute. |
| Type | String | The name for the attribute’s type. |
| TypeLength | String | The length for the attribute’s value. For some attribute types this is required. |
| IsForeignKey | bool | Specifies if the attribute is a foreign key or not. |
| IsFormula | bool | Specifies if the attribute is computed using a formula or not. |
| IsRequired | bool | Marks the attribute of being mandatory. |
| ErrorMessage | String | A tokened string representing the error message to be given when the attribute is required and left blank. |
| Access | String | A specific SQL expression that returns “READ” or “WRITE” to determine the access to the field. |
| DefaultValue | String | A specific SQL expression that represents the default value for the field or the formula that is used to compute its value. |
| Control | String | The name of the control through which the attribute’s value will be collected. |
| ControlWidth | Int | The control’s width in pixels. |
| ControlHeight | Int | The control’s height in pixels. |
| ControlMin | Float | The minimum value for the attribute. |
| ControlMax | Float | The maximum value for the attribute. |
| ControlUnit | Float | The increment unit for the value. |
| ControlStyle | String | The CSS style for the control. |
| ControlRowSource | String | A specific SQL query that retrieves the row data for the control. |

**InsertClass**

Purpose: inserts a new class.

Returns: int (representing the primary key value for the new class)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The internal name of the class. |
| Icon | String | The http relative path of the icon associated with this class. |
| NameSg | String | The class display name in singular form. |
| NamePl | String | The class display name in plural form. |
| CloneClassID | Int | The primary key value for the class that will be used as a model class. |

**InsertLink**

Purpose: inserts a master-slave relation between two classes.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MasterClassID | Int | The primary key value of the master class in the relation. |
| SlaveClassID | Int | The primary key value of the slave class in the relation. |

**UpdateAttribute**

Purpose: updates and existing class attribute.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AttributeID | Int | The primary key value of the class attribute. |
| Name | String | The name of the attribute. |
| Type | String | The name for the attribute’s type. |
| TypeLength | String | The length for the attribute’s value. For some attribute types this is required. |
| IsForeignKey | bool | Specifies if the attribute is a foreign key or not. |
| IsFormula | bool | Specifies if the attribute is computed using a formula or not. |
| IsRequired | bool | Marks the attribute of being mandatory. |
| ErrorMessage | String | A tokened string representing the error message to be given when the attribute is required and left blank. |
| Access | String | A specific SQL expression that returns “READ” or “WRITE” to determine the access to the field. |
| DefaultValue | String | A specific SQL expression that represents the default value for the field or the formula that is used to compute its value. |
| Control | String | The name of the control through which the attribute’s value will be collected. |
| ControlWidth | Int | The control’s width in pixels. |
| ControlHeight | Int | The control’s height in pixels. |
| ControlMin | Float | The minimum value for the attribute. |
| ControlMax | Float | The maximum value for the attribute. |
| ControlUnit | Float | The increment unit for the value. |
| ControlStyle | String | The CSS style for the control. |
| ControlRowSource | String | A specific SQL query that retrieves the row data for the control. |

**UpdateClass**

Purpose: updates an existing class.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| Name | String | The internal name of the class. |
| Icon | String | The http relative path of the icon associated with this class. |
| NameSg | String | The class display name in singular form. |
| NamePl | String | The class display name in plural form. |
| CustomPage | String | The name of the custom web page for the class. |
| HelpArticleID | Int | The primary key value of the help article. |
| Title | String | A tokened specific SQL expression that determines the title for the objects of the class. |
| LookSingle | String | A tokened specific HTML description for the data collecting form. |
| LookMulti | String | A tokened specific HTML description for the object listing. |

**ValidateAttributeChange**

Purpose: validates if it is ok for an attribute to change its data type.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AttributeID | Int | The primary key value of the class attribute. |
| Type | String | The name for the attribute’s type. |
| TypeLength | String | The length for the attribute’s value. For some attribute types this is required. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Class.resx | Resource |
| Properties/DataBase/Class.RO-RO.resx | Resource |

### Constant Class

This class represents the Constants module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetValue**

Purpose: retrieves the value for the specified constant.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Constant | String | The constant’s name. |

**Update**

Purpose: updates the value for the specified constant.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Constant | String | The constant’s name. |
| Value | String | The new value for the constant. |

### Control Class

This class represents the Control module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetControls**

Purpose: retrieves the list of controls.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| OnlyReportControls | Bool (optional) | This specifies if the list will be limited to the controls supported by report parameters or not. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Control.resx | Resource |
| Properties/DataBase/Control.RO-RO.resx | Resource |

### Counter Class

This class represents the DataBase module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetNextID**

Purpose: retrieves the first primary key value available for the specified table.

Returns: int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Counter | Amaryllis.DataBase.CounterName | Specifies the table for which the primary key value is needed. |
| ReserveCount | int | Specifies the number of primary key values that need to be reserved. |

### Desktop

This class represents the Desktop module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**DeleteDefaultIcon**

Purpose: deletes the specified default icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The primary key value of the default icon. |

**DeleteIcon**

Purpose: deletes the specified icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The primary key value of the icon. |

**DeleteWindow**

Purpose: deletes the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The primary key value of the window. |

**GetDefaultIconsForGroup**

Purpose: retrieves the list of the default icons assigned to the specified group.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**GetIconsForUser**

Purpose: retrieves the list of icons for the current logged user.

Returns: System.Data.DataTable

Parameters: NONE

**GetIconInfo**

Purpose: retrieves the information regarding the specified icon.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The primary key value of the icon. |

**GetWindowsForUser**

Purpose: retrieves the list of windows for the current logged user.

Returns: System.Data.DataTable

Parameters: NONE

**InsertDefaultIcon**

Purpose: inserts a default icon for a group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |
| PosX | Int | Represents the horizontal position in pixels. |
| PosY | Int | Represents the vertical position in pixels. |
| Icon | String | The http relative path of the pictogram associated with the icon. |
| Title | String | The icon’s title. |
| Link | String | The http path of the web page that must be displayed when the icon is triggered. |

**InsertIcon**

Purpose: inserts an icon for the current logged user.

Returns: int (representing the primary key value for the new icon).

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The icon’s title. |
| Icon | String | The http relative path of the pictogram associated with the icon. |
| Link | String | The http path of the web page that must be displayed when the icon is triggered. |
| PosX | Int | Represents the horizontal position in pixels. |
| PosY | Int | Represents the vertical position in pixels. |

**InsertWindow**

Purpose: inserts a window for the current logged user.

Returns: int (representing the primary key value for the new window).

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The window’s title. |
| Icon | String | The http relative path of the pictogram associated with the window. |
| Link | String | The http path of the web page that must be displayed by the window. |
| PosX | Int | Represents the horizontal position in pixels. |
| PosY | Int | Represents the vertical position in pixels. |
| Width | Int | The width of the window in pixels. |
| Height | Int | The height of the window in pixels. |

**UpdateDefaultIcon**

Purpose: updates the specified default icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The primary key value of the default icon. |
| PosX | Int | Represents the horizontal position in pixels. |
| PosY | Int | Represents the vertical position in pixels. |
| Icon | String | The http relative path of the pictogram associated with the icon. |
| Title | String | The icon’s title. |
| Link | String | The http path of the web page that must be displayed when the icon is triggered. |

**UpdateIcon**

Purpose: updates the specified icon.

Returns: NOTHING.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The primary key value of the icon. |
| Title | String | The icon’s title. |
| Icon | String | The http relative path of the pictogram associated with the icon. |
| Link | String | The http path of the web page that must be displayed when the icon is triggered. |
| PosX | Int | Represents the horizontal position in pixels. |
| PosY | Int | Represents the vertical position in pixels. |

**UpdateWindow**

Purpose: updates the data for the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The primary key value of the window. |
| Title | String | The window’s title. |
| Icon | String | The http relative path of the pictogram associated with the window. |
| Link | String | The http path of the web page that must be displayed by the window. |
| PosX | Int | Represents the horizontal position in pixels. |
| PosY | Int | Represents the vertical position in pixels. |
| Width | Int | The width of the window in pixels. |
| Height | Int | The height of the window in pixels. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Desktop.resx | Resource |
| Properties/DataBase/Desktop.RO-RO.resx | Resource |

### Event Class

This class represents the Events module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetTriggersForClass**

Purpose: retrieves the list of triggers with the existing queries for the specified class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**Raise**

Purpose: raises the specified event for the given object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the object’s class. |
| ObjectID | Int | The primary key value of the object. |
| ParentID | Int | The primary key value of the object’s parent. |
| Attributes | String | A specific XML that encapsulates all the object’s attribute values. |
| EventName | String | The name of the event. |

**SetTrigger**

Purpose: assigns a query for a class and event combo.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| EventID | Int | The primary key value of the event. |
| Command | String | A specific SQL query that represents the event for the class. |

### File Class

This class represents the File module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |
| ThumbnailSize | Int | The size for the thumbnail icon adjacent to image files. |

#### Methods

**Delete**

Purpose: deletes a file from the database and the cache saved on disk including thumbnail.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the file. |

**DownloadFile**

Purpose: downloads a file from the database and stores it in the “Files” folder. Also it tries to process the thumbnail for the file (if the file is not an image the thumbnail won’t be processed).

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the file. |

**GetFileIDs**

Purpose: retrieves a list of all the primary key values for the files in the database.

Returns: System.Data.DataTable

Parameters: NONE

**GetFileInfo**

Purpose: retrieves the information regarding the specified file.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the file. |

**GetFilePath**

Purpose: retrieves the relative http path for the specified file.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the file. |

**GetThumbnailPath**

Purpose: retrieves the thumbnail’s relative http path for the specified file. If no thumbnail exists, it will retrieve the relative http path of the file.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the file. |

**InsertFile**

Purpose: inserts a new file and downloads it in the cache folder creating a thumbnail as well (if possible).

Returns: int (representing the primary key value of the new file)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SourcePath | String | The absolute disk path of the file. |
| VirtualPath | String | A virtual path for the file containing the file’s name and extension. |

**UpdateFile**

Purpose: updates the specified file, file cache and thumbnail.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the file. |
| SourcePath | String | The absolute disk path of the file. |
| VirtualPath | String | A virtual path for the file containing the file’s name and extension. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/File.resx | Resource |
| Properties/DataBase/File.RO-RO.resx | Resource |
| Files | Folder |

### Form Class

This represents the custom form module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**Delete**

Purpose: deletes the specified custom form.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |

**ExecQuery**

Purpose: executes a specified query and retrieves the data.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Query | String | The specific SQL query that must be interpreted and executed. |

**GetFormInfo**

Purpose: retrieves the information regarding the custom form.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |

**GetForms**

Purpose: retrieves the lists of custom forms.

Retrieves: System.Data.DataTable

Parameters: NONE

**GetLanguageTagsForForm**

Purpose: retrieves the list of tags linked to the given custom form.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |

**GetRealFormInfo**

Purpose: retrieves the information regarding the custom form without translating it.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |

**GetRealForms**

Purpose: retrieves the lists of custom forms without translating the information.

Retrieves: System.Data.DataTable

Parameters: NONE

**Insert**

Purpose: inserts a new custom form.

Returns: int (representing the primary key value for the new custom form)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The internal name for the custom form. |
| Look | String | A specific tokened HTML that describes the custom form look. |
| ClassID | Int | The primary key value of the class. |

**Update**

Purpose: updates the specified custom form.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |
| Name | String | The internal name for the custom form. |
| Look | String | A specific tokened HTML that describes the custom form look. |
| ClassID | Int | The primary key value of the class. |

#### Adjacent Files and Folder

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Form.resx | Resource |
| Properties/DataBase/Form.RO-RO.resx | Resource |

### Harddrive Class

This class represents the virtual drive module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |
| IconMaxDimension | int | Specifies the maximum dimension for the icons assigned to extensions. |

#### Methods

**DeleteExtension**

Purpose: deletes the specified file extension.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ExtensionID | Int | The primary key value of the extension. |

**DeleteFile**

Purpose: deletes the specified virtual drive file.

Returns: int (representing the primary key value of the parent virtual folder)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the virtual file. |

**DeleteFolder**

Purpose: deletes the specified virtual folder.

Returns: int (representing the primary key value of the parent virtual folder)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FolderID | Int | The primary key value of the virtual folder. |

**GetExtensionInfo**

Purpose: returns the information regarding the specified extension.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Extension | String | The file extension. |

**GetExtensions**

Purpose: retrieves the list of registered extensions.

Returns: System.Data.DataTable

Parameters: NONE

**GetFileIDFromGUID**

Purpose: retrieves the primary key value of a virtual file from its GUID.

Returns: int (representing the primary key value for the virtual file)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GUID | String | The virtual file’s GUID. |

**GetFileInfo**

Purpose: retrieves the information regarding the specified virtual file.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the virtual file. |

**GetFolders**

Purpose: retrieves the list of sub-folders for the specified folder.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FolderID | Int | The primary key value of the virtual folder. |

**GetFolderTreeForUser**

Purpose: retrieves the tree folder structure for the current logged on user.

Returns: System.Data.DataTable

Parameters: NONE

**GetHierarchyForFolder**

Purpose: retrieves the hierarchical tree structure for the specified virtual folder.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FolderID | Int | The primary key value of the virtual folder. |

**InsertExtension**

Purpose: inserts a new extension or updates an extension with a corresponding icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Extension | String | The file extension. |
| Icon | System.Web.UI.WebControls.FileUpload | The FileUpload object that encapsulates the icon. |

**InsertFile**

Purpose: inserts a new virtual file.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FolderID | Int | The primary key value of the parent virtual folder. |
| Name | String | The file’s name, including the extension. |
| Source | System.Web.UI.WebControls.FileUpload | The FileUpload object that encapsulates the file. |

**InsertFolder**

Purpose: inserts a new virtual drive folder.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FolderID | Int | The primary key value of the parent virtual folder. |
| Name | String | The folder’s name. |

**Search**

Purpose: retrieves the list of files and folders matching the given search criteria.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Pattern | String | The search criteria. |

**UpdateFile**

Purpose: updates the specified virtual file.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileID | Int | The primary key value of the virtual file. |
| Name | String | The file’s name, including the extension. |
| Source | System.Web.UI.WebControls.FileUpload | The FileUpload object that encapsulates the file. |

**UpdateFolder**

Purpose: updates the specified virtual folder.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FolderID | Int | The primary key value of the virtual folder. |
| Name | String | The folder’s name. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Harddrive.resx | Resource |
| Properties/DataBase/Harddrive.RO-RO.resx | Resource |

### Language Class

This class represents the Language module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetLanguages**

Purpose: retrieves the list of available languages.

Returns: System.Data.DataTable

Parameters: NONE

**GetTags**

Purpose: retrieves the list of language tags that correspond to the given criteria.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Language | String | The language’s code. |
| Table | String | The table associated with the language tags. |
| Parameter1 | String | The first parameter for the language tags. |
| Parameter2 | String | The second parameter for the language tags. |

**SetTag**

Purpose: sets the value for a language tag.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Language | String | The language’s code. |
| Table | String | The table associated with the language tag. |
| Parameter1 | String | The first parameter for the language tag. |
| Parameter2 | String | The second parameter for the language tag. |
| Value | String | The language tag’s value. |

### Mapping Class

This class represents the Mapping module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**DeleteClassMapping**

Purpose: deletes the specified mapping between two classes and the mappings between the objects of those classes logging the operation.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MappingID | Int | The primary key value of the class mapping. |

**DeleteObjectMapping**

Purpose: deletes the specified mapping between two objects and logs the operation.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MappingID | Int | The primary key value of the object mapping. |

**GetMappingInfo**

Purpose: retrieves the information regarding the specified object mapping.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MappingID | Int | The primary key value of the object mapping. |

**GetMappingsForClass**

Purpose: retrieves the list of mapping that involves the specified class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| OnlyFromClass | Bool (optional) | Specifies that only the classes targeted by the specified class must be returned. |

**GetMappingsForClassAndUser**

Purpose: retrieves the list of mapping that involves the specified class and on which the current logged user has access to.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| OnlyFromClass | Bool (optional) | Specifies that only the classes targeted by the specified class must be returned. |

**GetMappingsForObject**

Purpose: retrieves the list of object mapped with the specified object that belong to the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| ObjectID | Int | The primary key value of the object. |

**InsertClassMapping**

Purpose: insert a new class mapping relation.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SourceClassID | Int | The primary key value of the source class. |
| TargetClassID | Int | The primary key value of the target class. |
| IsCommutative | Bool | Specifies if the mapping relation is commutative or not. |

**InsertObjectMapping**

Purpose: inserts a new mapping between two objects.

Returns: int (representing the primary key value for the new mapping)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SourceObjectID | Int | The primary key value of the source object. |
| TargetObjectID | Int | The primary key value of the target object. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Mapping.resx | Resource |
| Properties/DataBase/Mapping.RO-RO.resx | Resource |

### Menu Class

This class represents the Menu module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**DeleteAccess**

Purpose: deletes the access for the specified group on the given command.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CommandID | Int | The primary key value of the menu command. |
| GroupID | Int | The primary key value of the group. |

**DeleteCommand**

Purpose: deletes the specified command and the access links to it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CommandID | Int | The primary key value of the menu command. |

**DeleteMenu**

Purpose: deletes the specified menu and its sub-menus and sub-commands with the access links.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The primary key value of the menu. |

**GetCommandInfo**

Purpose: retrieves the specified menu command’s information.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | Int | The primary key value of the menu command. |

**GetMenuForGroup**

Purpose: retrieves the menu tree structure for the specified group.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**GetMenuFromGroups**

Purpose: retrieves the full menu tree structure with command mapped to all the access groups in order to see which groups have access to the commands.

Returns: System.Data.DataTable

Parameters: NONE

**GetMenuForUser**

Purpose: retrieves the menu tree structure for the logged on user.

Returns: System.Data.DataTable

Parameters: NONE

**GetMenuInfo**

Purpose: retrieves the information for the specified menu.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The primary key value of the menu. |

**GetMenuTree**

Purpose: retrieves the menu tree structure without commands.

Returns: System.Data.DataTable

Parameters: NONE

**InsertAccess**

Purpose: inserts a new access link between the specified menu command and group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CommandID | Int | The primary key value of the menu command. |
| GroupID | Int | The primary key value of the group. |

**InsertCommand**

Purpose: inserts a new menu command.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The primary key value of the parent menu. |
| Title | String | The command’s title. |
| Icon | String | The relative http path of the icon associated with the command. |
| Link | String | The http path of the page that must be displayed when the command is triggered. |

**InsertMenu**

Purpose: inserts a new menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The primary key value of the parent menu. |
| Title | String | The menu’s title. |

**UpdateCommand**

Purpose: updates the specified menu command.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | Int | The primary key value of the menu command. |
| MenuID | Int | The primary key value of the parent menu. |
| Title | String | The command’s title. |
| Icon | String | The relative http path of the icon associated with the command. |
| Link | String | The http path of the page that must be displayed when the command is triggered. |

**UpdateMenu**

Purpose: updates the specified menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | Int | The primary key value of the menu. |
| MenuID | Int | The primary key value of the parent menu. |
| Title | String | The menu’s title. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Menu.resx | Resource |
| Properties/DataBase/Menu.RO-RO.resx | Resource |

### Notification Class

This class represent the Notification module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**Delete**

Purpose: deletes the specified notification.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| NotificationID | Int | The primary key value of the notification. |

**GetLatestNotificationsForUser**

Purpose: retrieves the latest notifications destined for the current logged on user.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int (optional) | The primary key value of the user. |
| Language | String (optional) | The language code representing the language in which the information should be translated. |

**GetNewNotificationForUser**

Purpose: retrieves the newest notifications for the current logged on user.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LastID | Int | The primary key value of the notification. The latest notification retrieved. |

**GetNotificationsForUser**

Purpose: retrieves all the notifications for the current logged on user.

Returns: System.Data.DataTable

Parameters: NONE

**SendToGroups**

Purpose: sends a notification to all the users in the specified groups.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Groups | String | The list of groups separated by the “,” character. |
| Type | String | The name of the notification type. |
| Link | String | The link that details the notification, if any. |
| Message | String | An HTML representing the message. |

**SendToUser**

Purpose: sends a notification to the current logged on user.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Type | String | The name of the notification type. |
| Link | String | The link that details the notification, if any. |
| Message | String | An HTML representing the message. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Notification.resx | Resource |
| Properties/DataBase/Notification.RO-RO.resx | Resource |

### Object Class

This class represents the Object module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**CheckBigInt**

Purpose: checks if the given value is a valid big integer.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckBit**

Purpose: checks if the given value is a valid bit.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckDateTime**

Purpose: checks if the given value is a valid date/time.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckFloat**

Purpose: checks if the given value is a valid float.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckInt**

Purpose: checks if the given value is a valid integer.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckVarBinary**

Purpose: checks if the given value is a valid variable binary stream.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckVarChar**

Purpose: checks if the given value is a valid variable character strem.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**DeleteObject**

Purpose: validates and deletes the object displayed by the given form.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | Display form identifier for the object. |

**EncodeError**

Purpose: encodes the specified error message for the specified field in the specific XML form.

Returns: String

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Field | String | The field name. |
| Message | String | The error message. |

**GetAttributesForObject**

Purpose: retrieves the list of attributes for the given object.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**GetAttributesForObjectExt**

Purpose: retrieves an extended list of attributes for the specified object.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**GetAttribuitesForObjectLog**

Purpose: retrieves the list of attributes for the specified logged instance.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LogID | Int | The primary key value of the object log. |

**GetBreadcrumbs**

Purpose: retrieves the object hierarchy above the given object.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**GetChildObjects**

Purpose: retrieves the list of child objects form the specified object and the given class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| RootObjectID | Int | The primary key value of the object. |
| ChildClassID | Int | The primary key value of the class. |

**GetEntry**

Purpose: returns a single row from the cache representing the given object.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**GetLogsForObject**

Purpose: retrieves the list of logs generated for the given object.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**GetObjectInfo**

Purpose: retrieves the brief information for the specified object.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**GetObjectListing**

Purpose: retrieves the list of object for the specified class that match the criteria described by the rest of the parameters.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| ParentID | Int | The primary key value of the parent object. |
| Filter | String | A specific XML specifying restriction for different attributes. |
| Order | String | The list of attributes separated by the “,” character determining the way in which the objects must be ordered. |
| MappedToID | Int | The primary key value of the class that has objects in a mapping relation with the retrieved objects. |
| MappingFilter | Int | -1 – not mapped with  0 – no mapping relation  1 – mapped with |
| SearchCriteria | String | A generalized search criteria if needed. |

**GetObjectLogInfo**

Purpose: retrieves the brief information regarding the specified log instance.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LogID | Int | The primary key value of the object log. |

**GetObjectMatrix**

Purpose: retrieves a matrix of object primary key values coordinated by the specified attributes.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| RootObjectID | Int | The primary key value of the parent object. |
| ChildClassID | Int | The primary key value of the class. |
| Horizontal | String | The name of the attribute that will give the horizontal dimension. |
| Vertical | String | The name of the attribute that will give the vertical dimension. |

**GetPossibleParents**

Purpose: retrieves the list of the possible parent for the given class of objects.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetRecycleObjects**

Purpose: retrieves the list of object belonging to the specified class that were deleted.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetRecycleSummary**

Purpose: retrieves the brief information regarding the objects that are deleted.

Returns: System.Data.DataTable

Parameters: NONE

**GetRowSourceForAttribute**

Purpose: computes the row source for the specified attribute.

Returns: System.Data.DataTable

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassAttributeID | Int | The primary key value of the class’s attribute. |
| ObjectID | Int | The primary key value of the object. |
| ParentObjectID | Int | The primary key value of the parent object. |

**GetSearchResult**

Purpose: retrieves the list of objects found in the specified ID list that belong to the specified class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| IDs | String | List of object primary key values separated by the “,” character. |

**InsertObject**

Purpose: inserts a new object in the database collecting the attribute values from the specified form.

Returns: int (representing the primary key value for the newly inserted object)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | Display form identifier for the object. |

**LogObject**

Purpose: triggers a log operation for the given object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**RestoreObject**

Purpose: restores a deleted object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |
| WithChildren | Bool | Specifies that the object will be restore with all its deleted children. |

**Search**

Purpose: retrieves the list of objects that match the given criteria.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Criteria | String | The search criteria. |

**UpdateObject**

Purpose: updates the object specified by the form specified through the given identifier.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | Display form identifier for the object. |

**UpdateObjectParent**

Purpose: updates the parent of an object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |
| ParentID | Int | The primary key value of the new parent object. |

**ValidateAttribute**

Purpose: validates the data type of an attribute’s value.

Returns: string (representing the error message to be displayed)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| DataType | String | The data type name for the attribute. |
| Value | String | The value for the attribute. |

**ValidateAttributes**

Purpose: validates the attributes’ values for an object of the specified class.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| Attributes | String | The attribute values encapsulated in a specific XML structure. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Object.resx | Resource |
| Properties/DataBase/Object.RO-RO.resx | Resource |

### ReferenceGuide Class

This class represents the Reference Guide module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetAdjacentClasses**

Purpose: retrieves the list of the adjacent classes (master, slave or mapped) for the specified classes.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetAttributesForClass**

Purpose: retrieves the attribute list for the specified class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetClasses**

Parameters: retrieves the list of classes translated in the current interface language.

Returns: System.Data.DataTable

Parameters: NONE

### Report Class

This class represents the Report module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**CheckBigInt**

Purpose: checks if the given value is a valid big integer.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckBit**

Purpose: checks if the given value is a valid bit.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckDateTime**

Purpose: checks if the given value is a valid date/time.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckFloat**

Purpose: checks if the given value is a valid float.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckInt**

Purpose: checks if the given value is a valid integer.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckVarBinary**

Purpose: checks if the given value is a valid variable binary stream.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**CheckVarChar**

Purpose: checks if the given value is a valid variable character strem.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Value | String | The value that needs to be checked. |

**DeleteGraph**

Purpose: deletes the specified graph.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GraphID | Int | The primary key value of the graph. |

**DeleteParameter**

Purpose: deletes the specified report parameter.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ParameterID | Int | The primary key value of the report parameter. |

**DeleteReport**

Purpose: deletes the specified report with the adjacent graphs and parameters.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**EncodeError**

Purpose: encodes the specified error message for the specified field in the specific XML form.

Returns: String

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Field | String | The field name. |
| Message | String | The error message. |

**ExistsParameter**

Purpose: checks the parameter existence for a report.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| Name | String | The name of the parameter. |
| ExceptID | Int (optional) | The primary key value of the report parameter that will be excluded from the search. |

**ExistsReport**

Purpose: checks the report existence.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The name of the report. |
| ExceptID | Int (optional) | The primary key value of the report that will be excluded from the search. |

**GetGraphData**

Purpose: retrieves the data for the specified graph.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GraphID | Int | The primary key value of the report graph. |
| Params | String | The parameters values encoded in a specific XML structure. |

**GetGraphDataAtDate**

Purpose: retrieves the data for the specified graph at the specified moment in time.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GraphID | Int | The primary key value of the report graph. |
| Params | String | The parameters values encoded in a specific XML structure. |
| Date | String | The moment in time in the “DD/MM/YYYY HH:MM” format. |

**GetGraphInfo**

Purpose: retrieves the information for the given graph.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GraphID | Int | The primary key value of the graph. |

**GetGraphTypes**

Purpose: retrieves the list of graph types.

Returns: System.Data.DataTable

Parameters: NONE

**GetGraphsForReport**

Purpose: retrieves the list of graphs that belong to the given report.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetLanguageTagsForReport**

Purpose: retrieves the language tags assigned to the specified report.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetParameterInfo**

Purpose: retrieves de information regarding the specified report parameter.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report parameter. |

**GetParametersForReport**

Purpose: retrieves the list of parameters for the specified report.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report parameter. |

**GetRealGraphInfo**

Purpose: retrieves the information regarding the specified graph without translating it.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GraphID | Int | The primary key value of the graph. |

**GetRealGraphsForReport**

Purpose: retrieves the list of report graphs without translating it.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetRealReportInfo**

Purpose: retrieves the information for the given report without translating it.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetReportData**

Purpose: retrieves the data for the given report.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| Params | String | The parameters values encoded in a specific XML structure. |

**GetReportDataAtDate**

Purpose: retrieves the data for the given report at the specified moment in time.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| Params | String | The parameters values encoded in a specific XML structure. |
| Date | String | The moment in time in the “DD/MM/YYYY HH:MM” format. |

**GetReportInfo**

Purpose: retrieves the information for the specified report.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetReportList**

Purpose: retrieves the list of reports.

Returns: System.Data.DataTable

Parameters: NONE

**GetRowSourceForParameter**

Purpose: computes the row source for the specified report parameter.

Returns: System.Data.DataTable

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ParamID | Int | The primary key value of the report parameter. |

**InsertGraph**

Purpose: inserts a new graph.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| Name | String | The internal name for the graph. |
| TypeID | Int | The primary key value of the report graph type. |
| RowSource | String | A specific SQL query that retrieves the data for the report graph. |

**InsertParameter**

Purpose: inserts a new report parameter.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| Name | String | The internal name for the graph. |
| IsRequired | Bool | Specifies the fact that the parameter requires a value or not. |
| Type | String | The parameter type. |
| TypeLength | Int | The maximum length of the value for the parameter. Some data types require this. |
| Control | String | The name of the control through which the parameter’s value will be collected. |
| ControlWidth | Int | The control’s width in pixels. |
| ControlHeight | Int | The control’s height in pixels. |
| ControlMin | Float | The minimum value for the parameter. |
| ControlMax | Float | The maximum value for the parameter. |
| ControlUnit | Float | The increment unit for the value. |
| ControlStyle | String | The CSS style for the control. |
| ControlRowSource | String | A specific SQL query that retrieves the row data for the control. |

**InsertReport**

Purpose: inserts a new report.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The report’s internal name. |
| Title | String | The title for the report. |

**UpdateGraph**

Purpose: updates the specified report graph.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GraphID | Int | The primary key value of the report graph. |
| Name | String | The internal name for the graph. |
| TypeID | Int | The primary key value of the report graph type. |
| RowSource | String | A specific SQL query that retrieves the data for the report graph. |

**UpdateParameter**

Purpose: updates the specified report parameter.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ParamID | Int | The primary key value of the report parameter. |
| Name | String | The internal name for the graph. |
| IsRequired | Bool | Specifies the fact that the parameter requires a value or not. |
| Type | String | The parameter type. |
| TypeLength | Int | The maximum length of the value for the parameter. Some data types require this. |
| Control | String | The name of the control through which the parameter’s value will be collected. |
| ControlWidth | Int | The control’s width in pixels. |
| ControlHeight | Int | The control’s height in pixels. |
| ControlMin | Float | The minimum value for the parameter. |
| ControlMax | Float | The maximum value for the parameter. |
| ControlUnit | Float | The increment unit for the value. |
| ControlStyle | String | The CSS style for the control. |
| ControlRowSource | String | A specific SQL query that retrieves the row data for the control. |

**UpdateReport**

Purpose: updates the specified report.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| RowsOnPage | Int | Specifies the maximum number of rows for a single page. |
| Name | String | The report’s internal name. |
| Title | String | The title for the report. |
| LookParams | String | A specific tokened HTML describing the look for the parameter form. |
| LookHeader | String | A specific tokened HTML describing the look for the report header. |
| LookRow | String | A specific tokened HTML describing the way a report entry looks. |
| RowSource | String | A specific SQL query that retrieves the data for the report. |

**ValidateParameter**

Purpose: validates the data type of a parameter’s value.

Returns: string (representing the error message to be displayed)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| DataType | String | The data type name for the parameter. |
| Value | String | The value for the parameter. |

**ValidateParameters**

Purpose: validates the parameters’ values for the specified report.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| Params | String | The parameter values encapsulated in a specific XML structure. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Report.resx | Resource |
| Properties/DataBase/Report.RO-RO.resx | Resource |

### Security Class

This class represents the Security module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Properties

**AddClassAccess**

Purpose: gives access for the specified group to the specified class.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| GroupID | Int | The primary key value of the group. |

**AddFormAccess**

Purpose: gives access for the specified group to the specified custom form.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |
| GroupID | Int | The primary key value of the group. |

**AddReportAccess**

Purpose: gives access for the specified group to the specified report.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |
| GroupID | Int | The primary key value of the group. |

**GetClassesForGroup**

Purpose: retrieves the list of classes on which the specified group has access.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**GetFormsForGroup**

Purpose: retrieves the list of custom forms on which the specified group has access.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**GetGroupsForClass**

Purpose: retrieves the list of groups that have access to the specified class.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**GetGroupsForForm**

Purpose: retrieves the list of groups that have access to the specified custom form.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |

**GetGroupsForReport**

Purpose: retrieves the list of groups that have access to the specified report.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetReportsForGroup**

Purpose: retrieves the list of reports on which the specified group has access.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**HasAccessOnClass**

Purpose: specifies if the current logged on user has access to the specified class.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**HasAccessOnForm**

Purpose: specifies if the current logged on user has access to the specified custom form.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |

**HasAccessOnObject**

Purpose: specifies if the current logged on user has access to the specified object.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**HasAccessOnPage**

Purpose: specifies if the current logged on user has access to the specified web page.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Page | String | The name of the web page, including the “aspx” extension. |

**HasAccessOnReport**

Purpose: specifies if the current logged on user has access to the specified report.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**IsAllowedOnPage**

Purpose: check to see if the current logged on user has access on the specified page.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Page | String | The name of the web page, including the “aspx” extension. |

**IsUserMemberOfGroups**

Purpose: checks if the current logged on user belongs to at least one groups specified.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Groups | String | The name of the groups separated by the “,” character. |

**RevokeClassAccess**

Purpose: deletes the specified access link for the class and group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AccessID | Int | The primary key value of the class access link. |

**RevokeFormAccess**

Purpose: deletes the specified access link for the custom form and group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AccessID | Int | The primary key value of the custom form access link. |

**RevokeReportAccess**

Purpose: deletes the specified access link for the report and group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| AccessID | Int | The primary key value of the report access link. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Security.resx | Resource |
| Properties/DataBase/Security.RO-RO.resx | Resource |

### Setting Class

This class represents the Setting module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |

#### Methods

**GetSettingsForUser**

Purpose: retrieves the list of settings for the current logged on user.

Returns: System.Data.DataTable

Parameters: NONE

**GetValueForSetting**

Purpose: retrieves the value for the specified setting and user.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Setting | String | The name of the setting. |
| UserID | Int (optional) | The primary key value of the user. The default is the current logged on user. |

**Update**

Purpose: updates the specified setting for the specified user.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Setting | String | The name of the setting. |
| Value | String | The new setting’s value. |
| UserID | Int (optional) | The primary key value of the user. The default is the current logged on user. |

### User Class

This class represents the User module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |
| PhotoMaxDimension | Int | The maximum dimension for a user photo. |
| eMailPattern | String | A regular expression for validating email addresses. |

#### Methods

**DeleteApartenence**

Purpose: deletes an appurtenance of a user to a group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ApartenenceID | Int | The primary key value of the user-group link. |

**DeleteGroup**

Purpose: deletes the group with all the links for it (classes, reports, custom forms, users and web pages).

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**DeleteUser**

Purpose: deletes the specified user with its links to the groups.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |

**GetAdministrativeStatistic**

Purpose: retrieves a brief statistic regarding new objects, articles and files.

Returns: System.Data.DataTable

Parameters: NONE

**GetApartenencesForUser**

Purpose: retrieves the list of groups that are linked with the specified user.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |

**GetGroupInfo**

Purpose: retrieves the information regarding the specified group.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**GetGroups**

Purpose: retrieves the list of all groups.

Returns: System.Data.DataTable

Parameters: NONE

**GetGUIDFromUserID**

Purpose: retrieves the specified user’s GUID from its primary key value.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | Int | The primary key value of the user. |

**GetIDForGroup**

Purpose: retrieves the primary key value for the specified group.

Returns: int (representing the primary key value of the group)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Group | String | The group’s name. |

**GetUserIDFromGUID**

Purpose: retrieves the primary key value for the user from its GUID.

Returns: int (representing the user’s primary key value)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GUID | String | The GUID of the user. |

**GetUserPhoto**

Purpose: retrieves the relative web path for the specified user’s photo.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserName | String | The user’s name used for authentication. |

**GetUserInfo**

Purpose: retrieves the information regarding the specified user.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |

**GetUsers**

Purpose: retrieves the list of users.

Returns: System.Data.DataTable

Parameters: NONE

**GetUsersForGroup**

Purpose: retrieves the list of users that belong to the specified group.

Returns: System.Data.DataTable

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |

**InsertApartenence**

Purpose: links the specified user to the specified group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |
| GroupID | Int | The primary key value of the group. |

**InsertGroup**

Purpose: inserts a new group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The group’s name. |

**InsertUser**

Purpose: inserts a new user.

Returns: int (representing the primary key value for the new user)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FirstName | String | The user’s first name. |
| LastName | String | The user’s last name. |
| eMail | String | The user’s e-mail address. |
| Phone | String | The user’s phone number (must be a cell phone number in order to receive SMSs). |
| UserName | String | The user’s name used for authentication. |
| Password | String | The user’s password used for authentication. |
| Confirmation | String | The password confirmation. |
| Photo | System.UI.WebControls.FileUpload | The FileUpload control that contains the user’s photo. |
| MustChangePassword | bool | Specifies if the user must change his password at the first login. |

**IsMemberOf**

Purpose: checks if the specified user belongs to the specified group.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |
| Group | String | The group’s name. |

**IsUserNameAvailable**

Purpose: checks if the specified user name is available.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserName | String | The user’s name used for authentication. |
| AvoidUserID | Int (optional) | The primary key value of the user that must be avoided from the check. |

**LockUser**

Purpose: locks the account for the specified user.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |

**LogIn**

Purpose: authenticates a user.

Returns: int (representing the primary key value for the user authenticated with the provided credentials)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserName | String | The user’s name used for authentication. |
| Password | String | The user’s password used for authentication. |

**MustChangePassword**

Purpose: checks if the specified user must change his password.

Returns: bool

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |

**UnlockUser**

Purpose: unlocks the account for the specified user.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |

**UpdateGroup**

Purpose: updates the specified group.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| GroupID | Int | The primary key value of the group. |
| Name | String | The group’s name. |

**UpdateUser**

Purpose: updates the specified user.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| UserID | Int | The primary key value of the user. |
| FirstName | String | The user’s first name. |
| LastName | String | The user’s last name. |
| eMail | String | The user’s e-mail address. |
| Phone | String | The user’s phone number (must be a cell phone number in order to receive SMSs). |
| UserName | String | The user’s name used for authentication. |
| Password | String | The user’s password used for authentication. |
| Confirmation | String | The password confirmation. |
| Photo | System.UI.WebControls.FileUpload | The FileUpload control that contains the user’s photo. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/User.resx | Resource |
| Properties/DataBase/User.RO-RO.resx | Resource |

### Utility Class

This class represents the Utility module from the data layer.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Prefix | String | Represents the prefix for all the stored procedures called from the data layer. |
| Tables | String[] | A list of procedure names that are linked to table. These procedures are called in the specified order. |

#### Methods

**ExportDataTable**

Purpose: exports the given data table to the specified file as the specified item type.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SW | Ref System.IO.StreamWriter | A reference for the stream writer that writes the information to the desired physical file. |
| DT | Ref System.Data.DataTable | The data table that must be exported to the file. |
| ItemType | String | The name of the item type that characterizes the given data. |

**ExportMetaData**

Purpose: exports all the data retrieved by the procedures enumerated in the Tables vector to a file on disk and returns the path for the file.

Returns: string

Parameters: NONE

**GetPossibleObjectDuplicates**

Purpose: identifies and retrieves the list of possible duplicated objects.

Returns: System.Data.DataTable

Parameters: NONE

**ImportMetaData**

Purpose: imports the data from the specified file.

Returns: NONE

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FilePath | String | The absolute disk path of the file that must be imported. |

**MergeObjects**

Purpose: merges two objects.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID1 | Int | The primary key value of the primary object. |
| ObjectID2 | Int | The primary key value of the secondary object. |
| IgnoreDifferences | Bool | Makes the process ignore the differences between the object’s attributes values. |
| IgnoreClass | Bool | Makes the process ignore the class difference of the two objects, if any. |
| WithLogPreserve | Bool | Makes the process preserve the logs for the secondary object and make them the primary object’s logs. |

**ResetUserGUIDs**

Purpose: resets the GUIDs for all the users.

Returns: NOTHING

Parameters: NONE

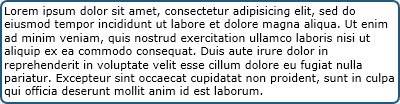
#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/DataBase/Utility.resx | Resource |
| Properties/DataBase/Utility.RO-RO.resx | Resource |

## Amaryllis.Objects Namespace

### Border Class

This class is responsible for placing bordered containers on web pages. The calls for this class can be former as HTML tags and the HTML objects that must be displayed inside the bordered container must be placed between the begin code and end code.



Border object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.Border BD = new Amaryllis.Objects.Border(); %>

<%BD.BeginBorder(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor

incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute

irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla

pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia

deserunt mollit anim id est laborum.

</p>

<%BD.EndBorder(); %>

If the design needs to be changed in order to customize the platform, this is one of the classes that may be changed in order to alter the interface.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |

#### Methods

**Border**

Purpose: (constructor) initializes the Response member.

**BeginBorder**

Purpose: begins the encapsulating border object.

Parameters: NONE

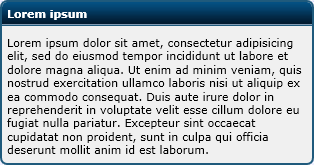
**EndBorder**

Purpose: closes the encapsulating border object.

Parameters: NONE

### Box Class

This class is responsible for placing floating boxes on web pages. The methods for this class can be used similar to HTML tags in order to compose a box object.



Box object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.Box B = new Amaryllis.Objects.Box(); %>

<%B.BeginBox("left", "Lorem ipsum", 300); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor

incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud

exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute

irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla

pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia

deserunt mollit anim id est laborum.

</p>

<%B.EndBox(); %>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |

#### Methods

**Box**

Purpose: (constructor) initializes the Response member.

**BeginBox**

Purpose: begins the encapsulating box object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FloatPosition | String | The floating position for the box object (left or right). |
| Width | Int (optional) | The width of the box object in pixels. |
| Title | String (optional) | The title for the box object |

**EndBox**

Purpose: closes the encapsulating box object.

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Objects/Box/TitleBar.png | Image |

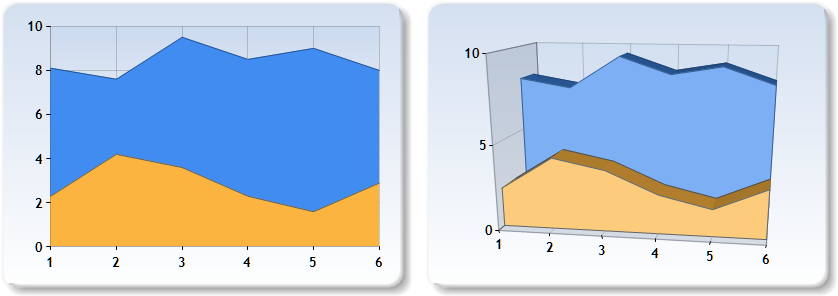
### Graph Object

This class is responsible for processing and composing images for the different graph types supported by the platform. The report module is responsible for keeping data regarding graphs but they are rendered using this class.

For this class each sub-chapter will present a method because each method is responsible for a single type of graph. In order for the graphs to be displayed correctly each method described will require the data send through the data table parameter to be in a certain form. This means that at the end of a sub-chapter we will describe the format of the data table needed.

#### Area Type Charts

For this type of graphs we have the Area method available which is described below.



Area charts in 2D and 3D forms

**Area Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

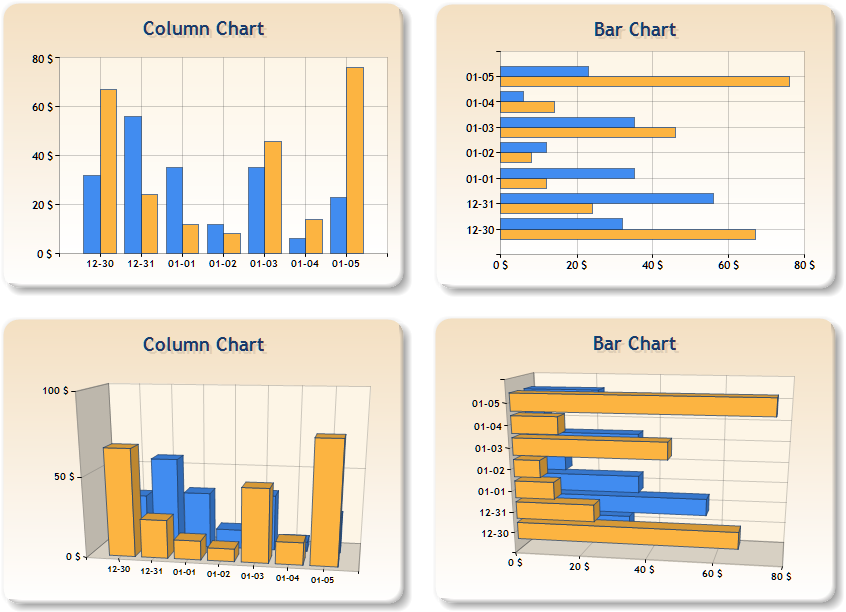
**Source Data Table**

|  |  |  |
| --- | --- | --- |
| Number | Yellow | Blue |
| 1 | 2.30 | 8.10 |
| 2 | 4.20 | 7.60 |
| 3 | 3.60 | 9.50 |
| 4 | 2.30 | 8.50 |
| 5 | 1.60 | 9.00 |
| 6 | 2.90 | 8.00 |

* The captions for the columns, except the first column, represent the names of the series that will be displayed in the legend.
* The first column’s values represent the indexes for the horizontal dimension.
* The rest of the columns, represent the series’ values for the corresponding horizontal dimension value found on the first column.

#### Bar Type Charts

For this type of graphs we have the Bar method available which is described below.

****Bar charts in bar and column form as well as 2D and 3D form

**Bar Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| IsVertical | Bool | Specifies if the chart will be displayed in column mode or bar mode. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

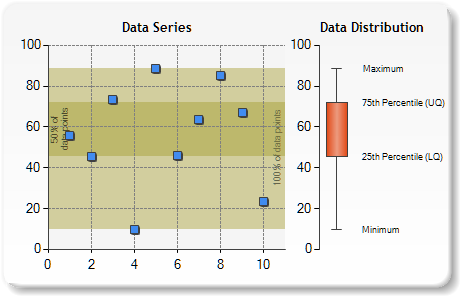
**Source Data Table**

|  |  |  |
| --- | --- | --- |
| Number | Yellow | Blue |
| 12-30 | 64 | 33 |
| 12-31 | 22 | 58 |
| 01-01 | 16 | 37 |
| 01-02 | 15 | 17 |
| 01-03 | 43 | 38 |
| 01-04 | 27 | 7 |
| 01-05 | 78 | 22 |

* The captions for the columns, except the first column, represent the names of the series that will be displayed in the legend.
* The first column’s values represent the indexes for the horizontal dimension.
* The rest of the columns, represent the series’ values for the corresponding horizontal dimension value found on the first column.

#### Box Plot Type Charts

For this type of graphs we have the BoxPlot method available which is described below.



Box plot chart

**BoxPlot Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

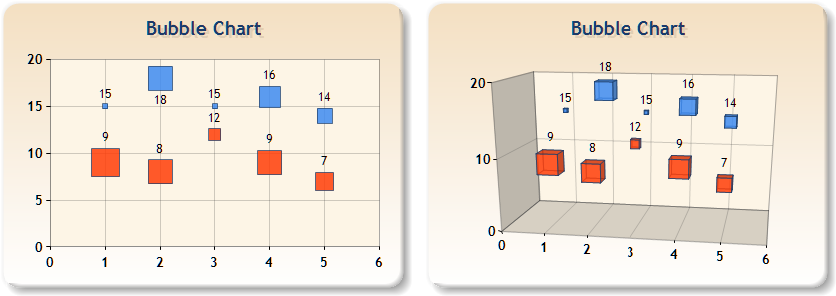
**Source Data Table**

|  |  |
| --- | --- |
| Number | Blue |
| 1 | 57 |
| 2 | 42 |
| 3 | 75 |
| 4 | 10 |
| 5 | 89 |
| 6 | 43 |
| 7 | 62 |
| 8 | 84 |
| 9 | 68 |
| 10 | 22 |

* The captions for the columns, except the first column, represent the names of the series that will be displayed in the legend.
* The first column’s values represent the indexes for the horizontal dimension.
* The rest of the columns, represent the series’ values for the corresponding horizontal dimension value found on the first column.

#### Bubble Type Charts

For this type of graphs we have the Bubble method available which is described below.

****

Bubble charts in 2D and 3D form

**Bubble Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

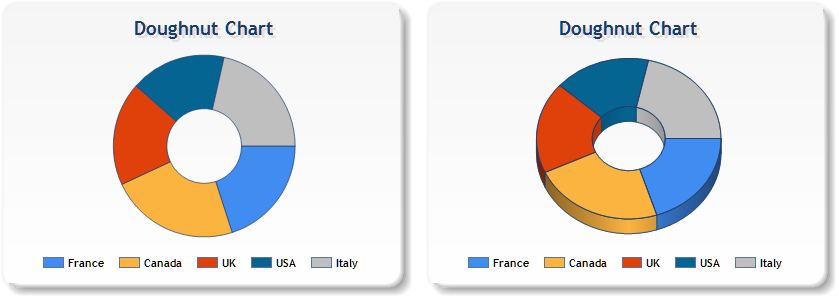
**Source Data Table**

|  |  |  |
| --- | --- | --- |
| Number | Blue | Red |
| 1 | 15 | 9 |
| 2 | 18 | 8 |
| 3 | 15 | 12 |
| 4 | 16 | 9 |
| 5 | 14 | 7 |

* The captions for the columns, except the first column, represent the names of the series that will be displayed in the legend.
* The first column’s values represent the indexes for the horizontal dimension.
* The rest of the columns, represent the series’ values for the corresponding horizontal dimension value found on the first column.
* The values for the series must be between 0 and 100 (representing presents).

#### Doughnut Type Charts

For this type of graphs we have the Doughnut method available which is described below.

****

Doughnut charts in 2D and 3D form

**Doughnut Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

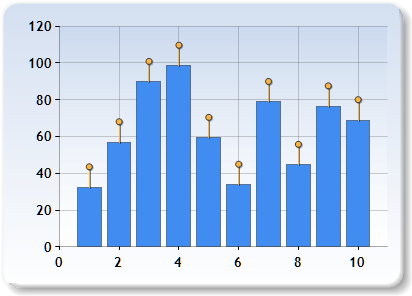
**Source Data Table**

|  |  |
| --- | --- |
| Countries | Countries Values |
| France | 65.62 |
| Canada | 75.54 |
| UK | 60.45 |
| USA | 34.73 |
| Italy | 85.42 |

* A series is represented by two columns, thus column 1 and 2 represent the first series, columns 3 and 4 represent to second series and so on.
* The names of the series are taken from the captions of the odd columns.
* The first column for a series contains the labels and the second column for a series represents the values.

#### Error Bar Type Charts

For this type of graphs we have the ErrorBar method available which is described below.

****

Error bar charts form

**ErrorBar Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

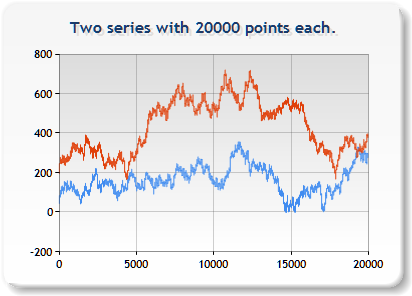
**Source Data Table**

|  |
| --- |
| Values |
| 32.4 |
| 56.9 |
| 89.7 |
| 98.5 |
| 59.3 |
| 33.8 |
| 78.8 |
| 44.6 |
| 76.4 |
| 68.9 |

* The data table must contain only one column which caption represents the series name.
* The values are given by the values found in that column.

#### Fast Line Type Charts

For this type of graphs we have the FastLine method available which is described below.

****

Fast line charts form

**FastLine Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

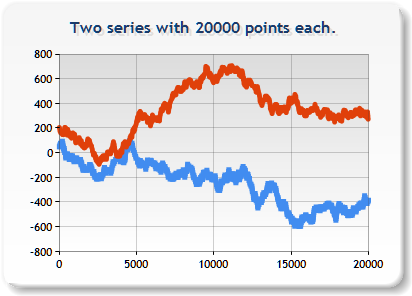
**Source Data Table**

|  |  |
| --- | --- |
| Red | Blue |
| 22.78 | 65.62 |
| 23.01 | 64.98 |
| 22.54 | 65.43 |
| … | … |
| 45.21 | 85.42 |

* A series is represented by one column.
* The names of the series are taken from the captions of the representing column.
* Each column has the consecutive point values.

#### Fast Point Type Charts

For this type of graphs we have the FastPoint method available which is described below.

****

Fast point charts form

**FastPoint Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

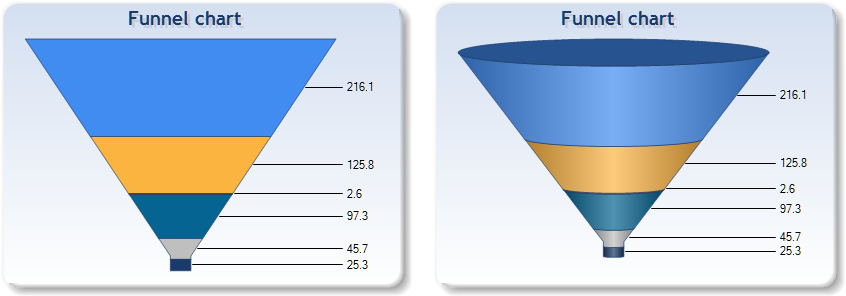
**Source Data Table**

|  |  |
| --- | --- |
| Red | Blue |
| 22.78 | 65.62 |
| 23.01 | 64.98 |
| 22.54 | 65.43 |
| … | … |
| 45.21 | 85.42 |

* A series is represented by one column.
* The names of the series are taken from the captions of the representing column.
* Each column has the consecutive point values.

#### Funnel Type Charts

For this type of graphs we have the Funnel method available which is described below.

****

Funnel charts in 2D and 3D form

**Funnel Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

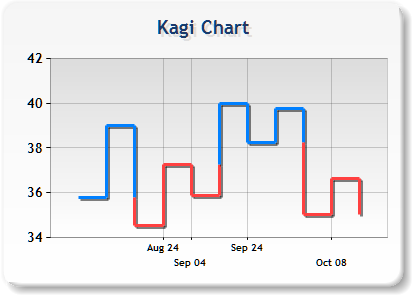
**Source Data Table**

|  |
| --- |
| Default |
| 216.1 |
| 125.8 |
| 2.6 |
| 97.3 |
| 45.7 |
| 25.3 |

* A series is represented by one column.
* The names of the series are taken from the captions of the representing column.
* Each column contains the series’ values.

#### Kagi Type Charts

For this type of graphs we have the Kagi method available which is described below.

****

Kagi charts form

**Kagi Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

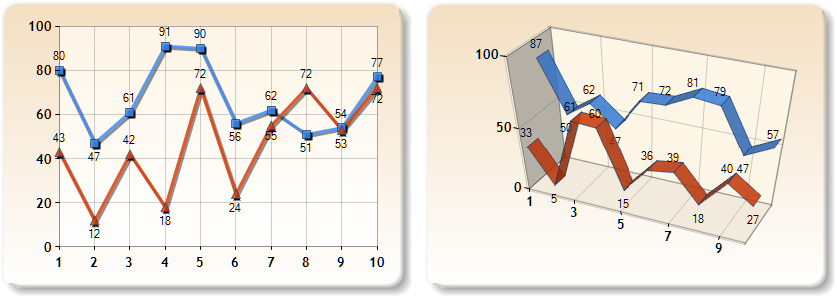
**Source Data Table**

|  |  |
| --- | --- |
| Evolution | Evolution Values |
| Aug 06 | 35.8 |
| Aug 11 | 39.1 |
| Aug 16 | 35.6 |
| Aug 21 | 36.2 |
| Aug 26 | 34.5 |
| Aug 31 | 37.3 |
| Sep 04 | 35.9 |
| Sep 09 | 39.4 |
| Sep 16 | 38.7 |
| Sep 20 | 40.0 |
| Sep 24 | 38.1 |
| Sep 29 | 39.8 |
| Oct 02 | 35.0 |
| Oct 07 | 36.8 |
| Oct 12 | 35.0 |

* A series is represented by two columns, thus column 1 and 2 represent the first series, columns 3 and 4 represent to second series and so on.
* The names of the series are taken from the captions of the odd columns.
* The first column for a series contains the labels and the second column for a series represents the values.

#### Line Type Charts

For this type of graphs we have the Line method available which is described below.

****

Line charts in 2D and 3D form

**Line Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

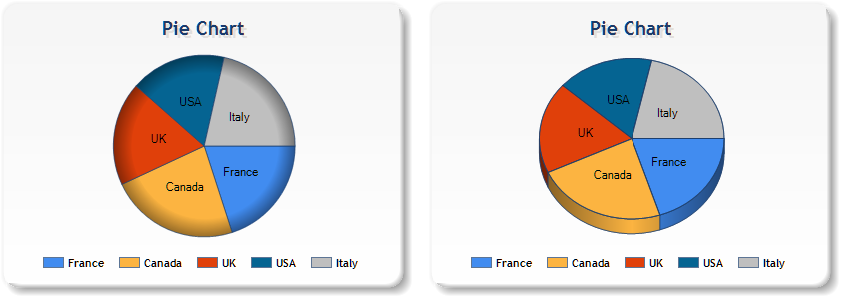
**Source Data Table**

|  |  |
| --- | --- |
| Red | Blue |
| 28 | 77 |
| 33 | 82 |
| 6 | 75 |
| 44 | 83 |
| 70 | 78 |
| 47 | 67 |
| 22 | 75 |
| 63 | 65 |
| 65 | 79 |
| 29 | 64 |

* A series is represented by one column.
* The names of the series are taken from the captions of the representing column.
* Each column has the consecutive point values.

#### Pie Type Charts

For this type of graphs we have the Pie method available which is described below.

****

Pie charts in 2D and 3D form

**Pie Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

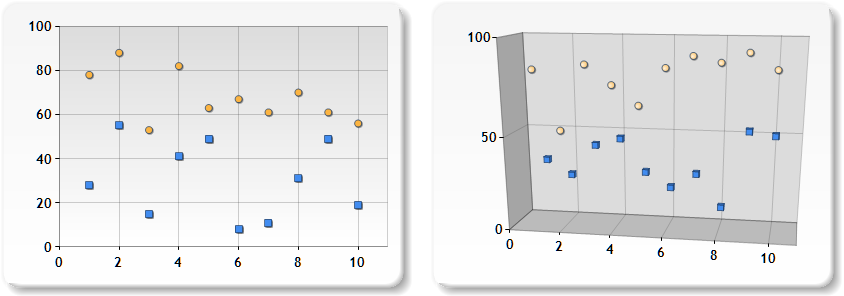
**Source Data Table**

|  |  |
| --- | --- |
| Countries | Countries Values |
| France | 65.62 |
| Canada | 75.54 |
| UK | 60.45 |
| USA | 34.73 |
| Italy | 85.42 |

* A series is represented by two columns, thus column 1 and 2 represent the first series, columns 3 and 4 represent to second series and so on.
* The names of the series are taken from the captions of the odd columns.
* The first column for a series contains the labels and the second column for a series represents the values.

#### Point Type Charts

For this type of graphs we have the Point method available which is described below.

****

Point charts in 2D and 3D form

**Point Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

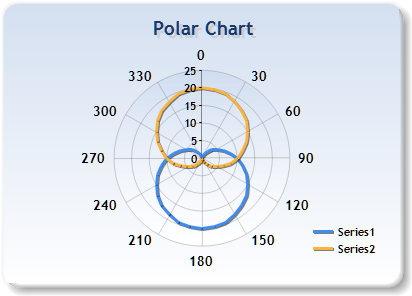
**Source Data Table**

|  |  |
| --- | --- |
| Blue | Yellow |
| 24 | 79 |
| 56 | 87 |
| 17 | 55 |
| 41 | 81 |
| 49 | 63 |
| 8 | 67 |
| 9 | 60 |
| 16 | 72 |
| 49 | 60 |
| 19 | 57 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* Each column represents the series’ values.

#### Polar Type Charts

For this type of graphs we have the Polar method available which is described below.

****

Polar charts form

**Polar Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

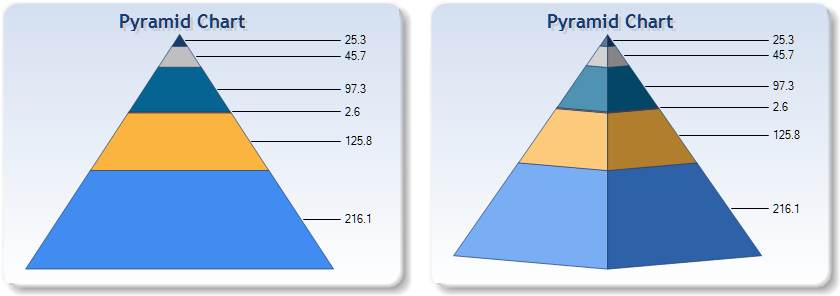
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Yellow | Yellow Values | Blue | Blue Value |
| 0 | 10.00 | 0 | 20.00 |
| 10 | 11.74 | 10 | 19.85 |
| 20 | 13.42 | 20 | 19.40 |
| 30 | 15.00 | 30 | 18.66 |
| 40 | 16.43 | 40 | 17.66 |
| 50 | 17.66 | 50 | 16.43 |
| 60 | 18.66 | 60 | 15.00 |
| 70 | 19.40 | 70 | 13.42 |
| 80 | 19.85 | 80 | 11.74 |
| 90 | 20.00 | 90 | 10.00 |
| 100 | 19.85 | 100 | 8.26 |
| 110 | 19.40 | 110 | 6.58 |
| 120 | 18.66 | 120 | 5.00 |
| 130 | 17.66 | 130 | 3.57 |
| 140 | 16.43 | 140 | 2.34 |
| 150 | 15.00 | 150 | 1.34 |
| 160 | 13.42 | 160 | 0.60 |
| 170 | 11.74 | 170 | 0.15 |
| 180 | 10.00 | 180 | 0.00 |
| 190 | 8.26 | 190 | 0.15 |
| 200 | 6.58 | 200 | 0.60 |
| 210 | 5.00 | 210 | 1.34 |
| 220 | 3.57 | 220 | 2.34 |
| 230 | 2.34 | 230 | 3.57 |
| 240 | 1.34 | 240 | 5.00 |
| 250 | 0.60 | 250 | 6.58 |
| 260 | 0.15 | 260 | 8.26 |
| 270 | 0.00 | 270 | 10.00 |
| 280 | 0.15 | 280 | 11.74 |
| 290 | 0.60 | 290 | 13.42 |
| 300 | 1.34 | 300 | 15.00 |
| 310 | 2.34 | 310 | 16.43 |
| 320 | 3.57 | 320 | 17.66 |
| 330 | 5.00 | 330 | 18.66 |
| 340 | 6.58 | 340 | 19.40 |
| 350 | 8.26 | 350 | 19.85 |
| 360 | 10.00 | 360 | 20.00 |

* A series is represented by two columns, thus column 1 and 2 represent the first series, columns 3 and 4 represent to second series and so on.
* The names of the series are taken from the captions of the odd columns.
* The first column for a series contains the labels and the second column for a series represents the values.

#### Pyramid Type Charts

For this type of graphs we have the Pyramid method available which is described below.

****

Pyramid charts in 2D and 3D form

**Pyramid Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

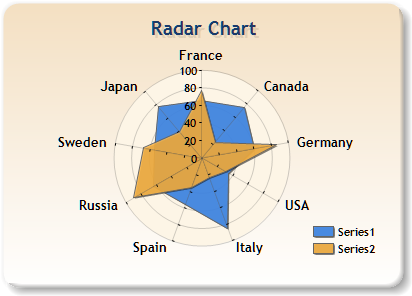
**Source Data Table**

|  |
| --- |
| Default |
| 216.1 |
| 125.8 |
| 2.6 |
| 97.3 |
| 45.7 |
| 25.3 |

* A series is represented by one column.
* The names of the series are taken from the captions of the representing column.
* Each column contains the series’ values.

#### Radar Type Charts

For this type of graphs we have the Radar method available which is described below.

****

Polar charts form

**Radar Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |

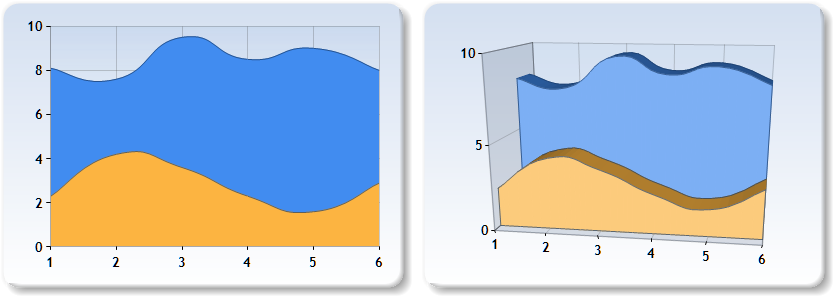
**Source Data Table**

|  |  |  |
| --- | --- | --- |
| Label | Yellow | Blue |
| France | 79.50 | 65.62 |
| Canada | 11.15 | 75.54 |
| Germany | 85.30 | 60.45 |
| USA | 16.73 | 34.73 |
| Italy | 12.14 | 85.42 |
| Spain | 18.45 | 55.90 |
| Russia | 89.20 | 63.60 |
| Sweden | 68.34 | 55.20 |
| Japan | 39.94 | 77.10 |

* The captions for the columns, except the first column, represent the names of the series that will be displayed in the legend.
* The first column’s values represent the indexes for the radial dimension.
* The rest of the columns, represent the series’ values for the corresponding horizontal dimension value found on the first column.

#### Spline Area Type Charts

For this type of graphs we have the SplineArea method available which is described below.



Spline area charts in 2D and 3D forms

**Spline Area Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

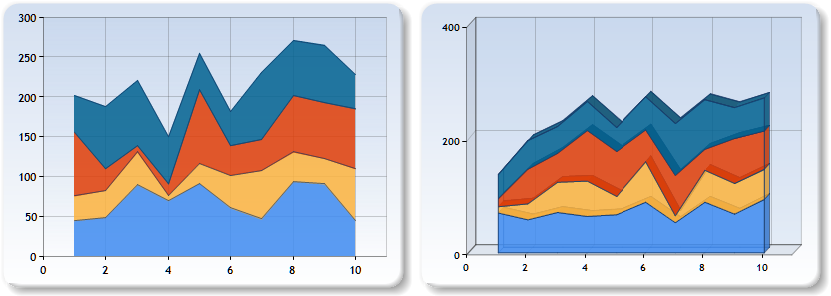
**Source Data Table**

|  |  |  |
| --- | --- | --- |
| Number | Yellow | Blue |
| 1 | 2.30 | 8.10 |
| 2 | 4.20 | 7.60 |
| 3 | 3.60 | 9.50 |
| 4 | 2.30 | 8.50 |
| 5 | 1.60 | 9.00 |
| 6 | 2.90 | 8.00 |

* The captions for the columns, except the first column, represent the names of the series that will be displayed in the legend.
* The first column’s values represent the indexes for the horizontal dimension.
* The rest of the columns, represent the series’ values for the corresponding horizontal dimension value found on the first column.

#### Stacked Area Type Charts

For this type of graphs we have the StackedArea method available which is described below.



Stacked area charts in 2D and 3D forms

**StackedArea Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

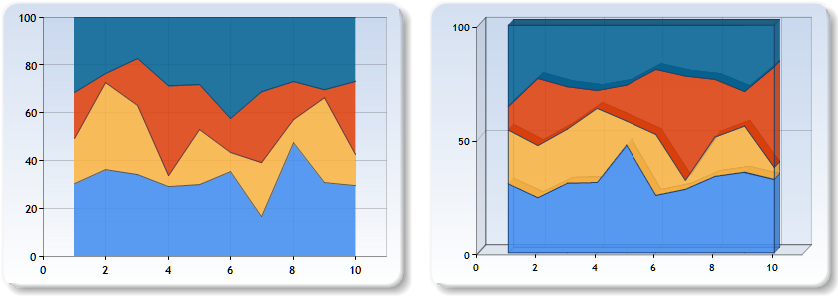
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Light Blue | Yellow | Red | Dark Blue |
| 48 | 10 | 78 | 48 |
| 50 | 10 | 27 | 82 |
| 84 | 14 | 8 | 82 |
| 70 | 2 | 20 | 61 |
| 86 | 8 | 88 | 45 |
| 62 | 15 | 32 | 46 |
| 49 | 26 | 32 | 74 |
| 96 | 28 | 72 | 71 |
| 95 | 22 | 73 | 73 |
| 48 | 58 | 76 | 39 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* The columns represent the series’ values.

#### Stacked Area 100% Type Charts

For this type of graphs we have the StackedArea100 method available which is described below.



Stacked area 100% charts in 2D and 3D forms

**StackedArea100 Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

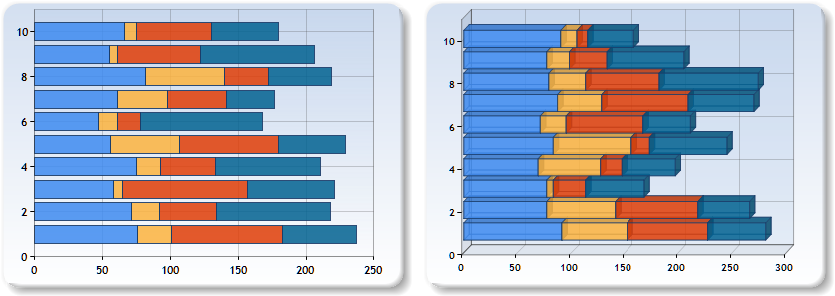
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Light Blue | Yellow | Red | Dark Blue |
| 48 | 10 | 78 | 48 |
| 50 | 10 | 27 | 82 |
| 84 | 14 | 8 | 82 |
| 70 | 2 | 20 | 61 |
| 86 | 8 | 88 | 45 |
| 62 | 15 | 32 | 46 |
| 49 | 26 | 32 | 74 |
| 96 | 28 | 72 | 71 |
| 95 | 22 | 73 | 73 |
| 48 | 58 | 76 | 39 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* The columns represent the series’ values.

#### Stacked Bar Type Charts

For this type of graphs we have the StackedBar method available which is described below.



Stacked bar charts in 2D and 3D forms

**StackedBar Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

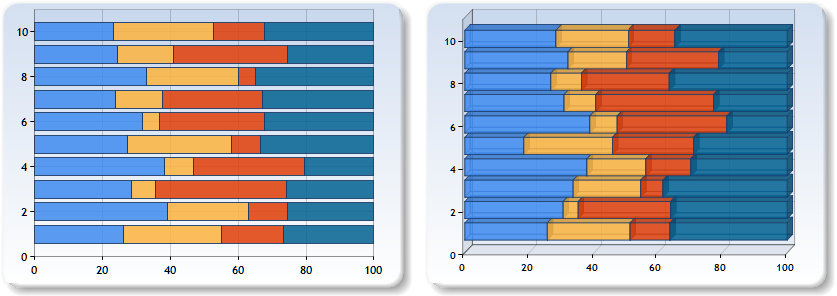
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Light Blue | Yellow | Red | Dark Blue |
| 48 | 10 | 78 | 48 |
| 50 | 10 | 27 | 82 |
| 84 | 14 | 8 | 82 |
| 70 | 2 | 20 | 61 |
| 86 | 8 | 88 | 45 |
| 62 | 15 | 32 | 46 |
| 49 | 26 | 32 | 74 |
| 96 | 28 | 72 | 71 |
| 95 | 22 | 73 | 73 |
| 48 | 58 | 76 | 39 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* The columns represent the series’ values.

#### Stacked Bar 100% Type Charts

For this type of graphs we have the StackedBar100 method available which is described below.



Stacked bar 100% charts in 2D and 3D forms

**StackedBar100 Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

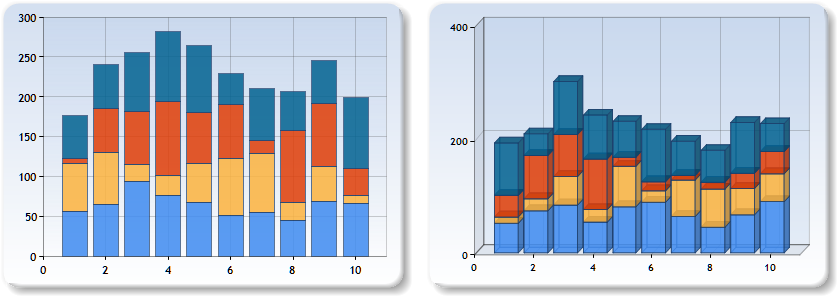
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Light Blue | Yellow | Red | Dark Blue |
| 48 | 10 | 78 | 48 |
| 50 | 10 | 27 | 82 |
| 84 | 14 | 8 | 82 |
| 70 | 2 | 20 | 61 |
| 86 | 8 | 88 | 45 |
| 62 | 15 | 32 | 46 |
| 49 | 26 | 32 | 74 |
| 96 | 28 | 72 | 71 |
| 95 | 22 | 73 | 73 |
| 48 | 58 | 76 | 39 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* The columns represent the series’ values.

#### Stacked Column Type Charts

For this type of graphs we have the StackedColumn method available which is described below.



Stacked column charts in 2D and 3D forms

**StackedColumn Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

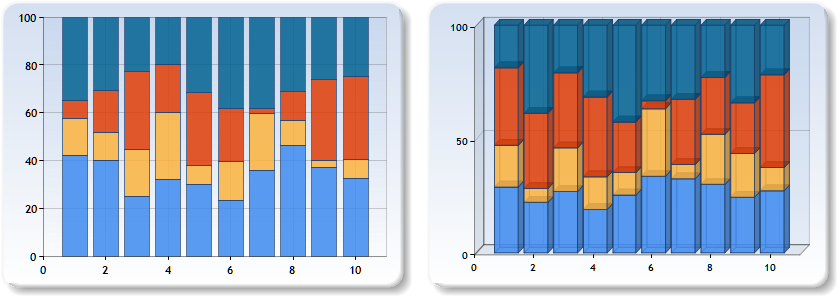
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Light Blue | Yellow | Red | Dark Blue |
| 48 | 10 | 78 | 48 |
| 50 | 10 | 27 | 82 |
| 84 | 14 | 8 | 82 |
| 70 | 2 | 20 | 61 |
| 86 | 8 | 88 | 45 |
| 62 | 15 | 32 | 46 |
| 49 | 26 | 32 | 74 |
| 96 | 28 | 72 | 71 |
| 95 | 22 | 73 | 73 |
| 48 | 58 | 76 | 39 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* The columns represent the series’ values.

#### Stacked Column 100% Type Charts

For this type of graphs we have the StackedColumn100 method available which is described below.



Stacked column 100% charts in 2D and 3D forms

**StackedColumn100 Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

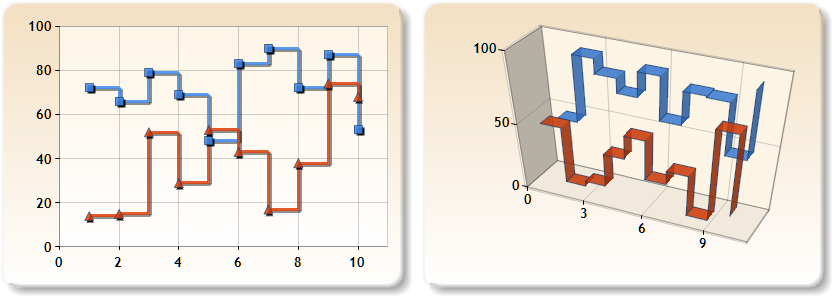
**Source Data Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Light Blue | Yellow | Red | Dark Blue |
| 48 | 10 | 78 | 48 |
| 50 | 10 | 27 | 82 |
| 84 | 14 | 8 | 82 |
| 70 | 2 | 20 | 61 |
| 86 | 8 | 88 | 45 |
| 62 | 15 | 32 | 46 |
| 49 | 26 | 32 | 74 |
| 96 | 28 | 72 | 71 |
| 95 | 22 | 73 | 73 |
| 48 | 58 | 76 | 39 |

* The captions for the columns represent the names of the series that will be displayed in the legend.
* The columns represent the series’ values.

#### Step Line Type Charts

For this type of graphs we have the StepLine method available which is described below.

****

Step line charts in 2D and 3D form

**StepLine Method**

Purpose: generates a graph images considering the given data.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference to the chart object from the web page. |
| DT | System.Data.DataTable | The data needed to display the graph. |
| Style3D | Bool | Specifies if the chart will be generated in 2D or 3D form. |

**Source Data Table**

|  |  |
| --- | --- |
| Red | Blue |
| 28 | 77 |
| 33 | 82 |
| 6 | 75 |
| 44 | 83 |
| 70 | 78 |
| 47 | 67 |
| 22 | 75 |
| 63 | 65 |
| 65 | 79 |
| 29 | 64 |

### Menu Class

This class is responsible for placing simple menus on web pages. The methods for this class can be used similar to HTML tags in order to compose a menu object.



Menu object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.Menu M = new Amaryllis.Objects.Menu(); %>

<%M.BeginMenu(300); %>

<%M.AddOption("Lorem ipsum dolor", "window.location=\"\";"); %>

<%M.AddOption("Lorem ipsum dolor", ""); %>

<%M.AddSeparator(); %>

<%M.AddOption("Lorem ipsum dolor", ""); %>

<%M.AddOption("Lorem ipsum dolor", ""); %>

<%M.AddOption("Lorem ipsum dolor", ""); %>

<%M.EndMenu(); %>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |
| OptionCount | Int | Stores the option count for the displayed options and contributes to the options’ HTML ids. |
| Width | Int | The width in pixels of the displayed menu. |

#### Methods

**Menu**

Purpose: (constructor) initializes the Response, OptionCount and Width properties.

**BeginMenu**

Purpose: begins the encapsulating menu object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Width | Int (optional) | The width of the box object in pixels. |

**AddOption**

Purpose: adds a new option in the menu object that triggers the specified java script.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Command | String | The name displayed for the option. |
| OnClickScript | String | The java script code that must be run when the option is triggered. |

**AddSeparator**

Purpose: adds a separator line to the menu object.

Parameters: NONE

**EndMenu**

Purpose: closes the encapsulating box object.

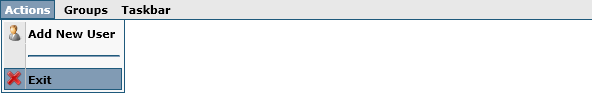
Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Objects/Menu.js | Script |

### MenuBar Class

This class is responsible for placing simple menu bars on web pages. The methods for this class can be used similar to HTML tags in order to compose a menu object.



Menu bar object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.MenuBar MB = new Amaryllis.Objects.MenuBar(); %>

<%

MB.BeginMenuBar();

MB.BeginMenu("Actions");

MB.AddCommand("Add New User", "Images/Icons/User.png", "utilitySetLoadingStatus(); window.location=\"AdminUsers\_Insert.aspx\";");

MB.AddSpace();

MB.AddCommand("Exit","Images/Icons/Close.png","utilityClose();");

MB.EndMenu();

MB.BeginMenu("Groups");

MB.AddCommand("New Group", "Images/Icons/Add.png", "utilitySetLoadingStatus(); window.location=\"Security\_InsertGroup.aspx\";");

MB.AddCommand("Group List", "Images/Icons/Security.png", "utilitySetLoadingStatus(); window.location=\"Security.aspx\";");

MB.EndMenu();

MB.BeginMenu("Taskbar");

MB.AddCommand("Edit Menu", "Images/Icons/Taskbar.png", "utilitySetLoadingStatus(); window.location=\"Menu.aspx\";");

MB.EndMenu();

MB.EndMenuBar();

%>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |
| MenuCount | Int | Stores the menu count for the displayed menus and contributes to the menu commands’ HTML ids. |
| CommandCount | Int | The width in pixels of the displayed menu. |

#### Methods

**Menu**

Purpose: (constructor) initializes the Response, MenuCount and CommandCount properties.

**BeginMenuBar**

Purpose: begins the encapsulating menu bar object.

Parameters: NONE

**BeginMenu**

Purpose: begins an encapsulating menu sub-object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title for the menu object. |

**AddCommand**

Purpose: adds an option to the encapsulating menu object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title for the menu command. |
| Icon | String | The relative HTTP path for the pictogram associated with the command. |
| Script | String | The java script code that must be run when the menu command is triggered. |

**AddSpace**

Purpose: adds a separator bar to the encapsulating menu object.

Parameters: NONE

**EndMenu**

Purpose: closes the encapsulating menu object.

Parameters: NONE

**EndMenuBar**

Purpose: closes the encapsulating menu bar object.

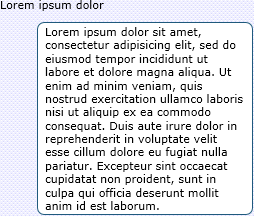
Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Icons | Folder |
| Scripts/Objects/MenuBar.js | Script |

### OverDiv Class

This class is responsible for placing simple menu bars on web pages. This class must be initialized at the beginning of the page and each object must be registered to this class.



OverDiv object display and reaction

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.OverDiv OD = new Amaryllis.Objects.OverDiv(); %>

<a id="object\_name">Lorem ipsum dolor</a>

<%

OD.RegisterObject("object\_name","Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.");

%>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |

#### Methods

**OverDiv**

Purpose: (constructor) initializes the Response property and places the HTML code that is used to display the tooltip.

**RegisterObject**

Purpose: register an HTML object with the OverDiv class in order to activate the tooltip with the specified message when the mouse cursor hovers over it.

Parameters:

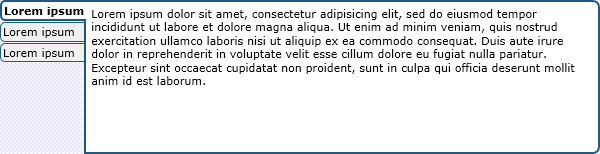
|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The HTML object id that must be registered. |
| Description | String | The tooltip for the registered object. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Objects/OverDiv.js | Script |

### Side Panel Class

This class is responsible for placing side tab panel objects on web pages. The methods for this class can be used similar to HTML tags in order to compose a side tab panel object.



Side tab panel object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.SidePanel SP = new Amaryllis.Objects.SidePanel(); %>

<%SP.BeginSidePanel(140); %>

<%SP.BeginHeader(); %>

<%SP.AddHeader("Lorem&nbsp;ipsum"); %>

<%SP.AddHeader("Lorem&nbsp;ipsum"); %>

<%SP.AddHeader("Lorem&nbsp;ipsum"); %>

<%SP.EndHeader(); %>

<%SP.BeginContent(); %>

<%SP.BeginPanel(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<%SP.EndPanel(); %>

<%SP.BeginPanel(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<%SP.EndPanel(); %>

<%SP.BeginPanel(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<%SP.EndPanel(); %>

<%SP.EndContent(); %>

<%SP.EndSidePanel(); %>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |
| TabCount | Int | Stores the side tab count for the displayed options and contributes to the panels’ HTML ids. |
| PanelCount | Int | Stores the panel count for the displayed options and contributes to the panels’ HTML ids. |
| Height | Int | The minimum height of the side tab panel in pixels. |
| IDs | System.Collections.Stack | Stores the IDs for all the side tabs objects placed on the web page. |

#### Methods

**SidePanel**

Purpose: (constructor) initializes the Response, TabCount, PanelCount, Height and IDs properties.

**BeginSidePanel**

Purpose: begins the encapsulating side tab object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Height | Int (optional) | The minimum height of the object in pixels. |

**BeginHeader**

Purpose: begins the header part for the side tab object.

Parameters: NONE

**AddHeader**

Purpose: adds a panel label in the object’s header.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title of the panel. |

**EndHeader**

Purpose: closes the header for the object.

Parameters: NONE

**BeginContent**

Purpose: begins the content section for the side tab object.

Parameters: NONE

**BeginPanel**

Purpose: begins the panel for the corresponding header label.

Parameters: NONE

**EndPanel**

Purpose: closes the panel for the corresponding header label.

Parameters: NONE

**EndContent**

Purpose: closes the content section for the side tab object.

Parameters: NONE

**EndSidePanel**

Purpose: closes the side panel object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ActivePanel | Int (optional) | The number of the panel that must be activated. The default is the first one. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Objects/SidePanel/Border.png | Image |
| Scripts/Objects/SidePanel.js | Script |

### Table Class

This class is responsible for placing table objects on web pages. The methods for this class can be used similar to HTML tags in order to compose a table object.

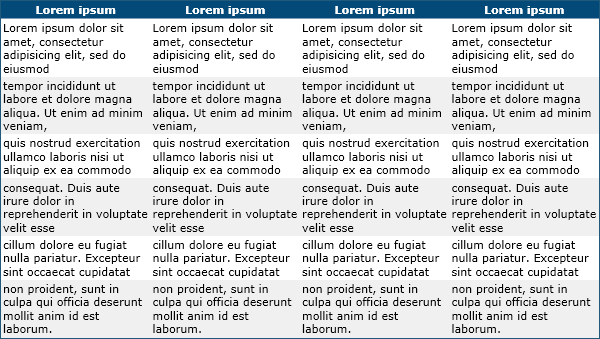


Table object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.Table T = new Amaryllis.Objects.Table(); %>

<%T.BeginTable("100%"); %>

<%T.BeginRow(); %>

<%T.BeginHeader(); %>

Lorem ipsum

<%T.EndHeader(); %>

<%T.BeginHeader(); %>

Lorem ipsum

<%T.EndHeader(); %>

<%T.BeginHeader(); %>

Lorem ipsum

<%T.EndHeader(); %>

<%T.BeginHeader(); %>

Lorem ipsum

<%T.EndHeader(); %>

<%T.EndRow(); %>

<%T.BeginRow(true); %>

<%T.BeginCell(); %>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

<%T.EndCell(); %>

<%T.BeginCell(); %>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

<%T.EndCell(); %>

<%T.BeginCell(); %>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

<%T.EndCell(); %>

<%T.BeginCell(); %>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

<%T.EndCell(); %>

<%T.EndRow(); %>

<%T.BeginRow(false); %>

<%T.BeginCell(); %>

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

<%T.EndCell(); %>

<%T.BeginCell(); %>

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

<%T.EndCell(); %>

<%T.BeginCell(); %>

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

<%T.EndCell(); %>

<%T.BeginCell(); %>

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

<%T.EndCell(); %>

<%T.EndRow(); %>

<%T.BeginRow(true); %>

<%T.BeginCell(); %>

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

<%T.EndCell(); %>

<%T.BeginCell(); %>

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

<%T.EndCell(); %>

<%T.BeginCell(); %>

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

<%T.EndCell(); %>

<%T.BeginCell(); %>

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

<%T.EndCell(); %>

<%T.EndRow(); %>

<%T.BeginRow(false); %>

<%T.BeginCell(); %>

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

<%T.EndCell(); %>

<%T.BeginCell(); %>

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

<%T.EndCell(); %>

<%T.BeginCell(); %>

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

<%T.EndCell(); %>

<%T.BeginCell(); %>

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

<%T.EndCell(); %>

<%T.EndRow(); %>

<%T.BeginRow(true); %>

<%T.BeginCell(); %>

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

<%T.EndCell(); %>

<%T.BeginCell(); %>

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

<%T.EndCell(); %>

<%T.BeginCell(); %>

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

<%T.EndCell(); %>

<%T.BeginCell(); %>

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

<%T.EndCell(); %>

<%T.EndRow(); %>

<%T.BeginRow(false); %>

<%T.BeginCell(); %>

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

<%T.EndCell(); %>

<%T.BeginCell(); %>

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

<%T.EndCell(); %>

<%T.BeginCell(); %>

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

<%T.EndCell(); %>

<%T.BeginCell(); %>

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

<%T.EndCell(); %>

<%T.EndRow(); %>

<%T.EndTable(); %>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |
| IsOdd | Bool | Retains the information about the next row that is processed. |

#### Methods

**Table**

Purpose: (constructor) initializes the Response property.

**BeginTable**

Purpose: begins a table object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Width | String | The width of the table object in pixels or percentage. |

**BeginRow**

Purpose: begins a table row.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IsOdd | Bool (optional) | Marks the row as odd or even. The default for this is “true” marking the row as odd. |

**BeginHeader**

Purpose: begins a header cell.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Align | String (optional) | The text alignment for the cell. |
| ColSpan | Int (optional) | The column span attribute for the cell. |
| RowSpan | Int (optional) | The row span attribute for the cell. |

**EndHeader**

Purpose: closes a header cell.

Parameters: NONE

**BeginCell**

Purpose: begins a cell.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Align | String (optional) | The text alignment for the cell. |
| ColSpan | Int (optional) | The column span attribute for the cell. |
| RowSpan | Int (optional) | The row span attribute for the cell. |

**EndCell**

Purpose: closes a cell.

Parameters: NONE

**EndRow**

Purpose: closes a row.

Parameters: NONE

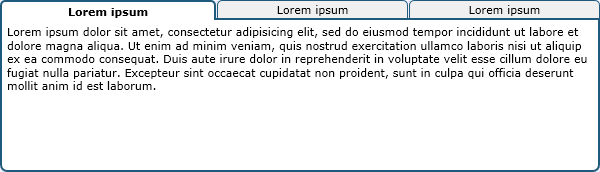
**EndTable**

Purpose: closes a table object.

Parameters: NONE

### Tab Panel Class

This class is responsible for placing top tab panel objects on web pages. The methods for this class can be used similar to HTML tags in order to compose a side tab panel object.



Top tab panel object display

You may use this class as follows to display containers like in the above figure.

<%Amaryllis.Objects.TabPanel TP = new Amaryllis.Objects.TabPanel(); %>

<%TP.BeginTabPanel(140); %>

<%TP.BeginHeader(); %>

<%TP.AddHeader("Lorem&nbsp;ipsum"); %>

<%TP.AddHeader("Lorem&nbsp;ipsum"); %>

<%TP.AddHeader("Lorem&nbsp;ipsum"); %>

<%TP.EndHeader(); %>

<%TP.BeginContent(); %>

<%TP.BeginPanel(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<%TP.EndPanel(); %>

<%TP.BeginPanel(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<%TP.EndPanel(); %>

<%TP.BeginPanel(); %>

<p>

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod

tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,

quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo

consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse

cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat

non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

</p>

<%TP.EndPanel(); %>

<%TP.EndContent(); %>

<%TP.EndTabPanel(); %>

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |
| TabCount | Int | Stores the side tab count for the displayed options and contributes to the panels’ HTML ids. |
| PanelCount | Int | Stores the panel count for the displayed options and contributes to the panels’ HTML ids. |
| Height | Int | The minimum height of the side tab panel in pixels. |
| IDs | System.Collections.Stack | Stores the IDs for all the side tabs objects placed on the web page. |

#### Methods

**TabPanel**

Purpose: (constructor) initializes the Response, TabCount, PanelCount, Height and IDs properties.

**BeginTabPanel**

Purpose: begins the encapsulating side tab object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Height | Int (optional) | The minimum height of the object in pixels. |

**BeginHeader**

Purpose: begins the header part for the side tab object.

Parameters: NONE

**AddHeader**

Purpose: adds a panel label in the object’s header.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title of the panel. |

**EndHeader**

Purpose: closes the header for the object.

Parameters: NONE

**BeginContent**

Purpose: begins the content section for the side tab object.

Parameters: NONE

**BeginPanel**

Purpose: begins the panel for the corresponding header label.

Parameters: NONE

**EndPanel**

Purpose: closes the panel for the corresponding header label.

Parameters: NONE

**EndContent**

Purpose: closes the content section for the side tab object.

Parameters: NONE

**EndSidePanel**

Purpose: closes the side panel object.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ActivePanel | Int (optional) | The number of the panel that must be activated. The default is the first one. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Objects/SidePanel.js | Script |

### TaskBar Class

This class is used internally in order to place the necessary HTML code for the task bar (not the taskbar menu).



Task bar display

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | Stores the http response object for the current http context. |

#### Methods

**TaskBar**

Purpose: (constructor) initializes the Response property.

**PlaceTaskBar**

Purpose: places the base HTML code on the page.

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Objects/TaskBar.resx | Resource |
| Properties/Objects/TaskBar.RO-RO.resx | Resource |
| Images/Objects/TaskBar/Arrow\_Left.png | Image |
| Images/Objects/TaskBar/Arrow\_Right.png | Image |
| Images/Objects/TaskBar/ButtonComponent\_1.png | Image |
| Images/Objects/TaskBar/ButtonComponent\_2.png | Image |
| Images/Objects/TaskBar/ButtonComponent\_3.png | Image |
| Scripts/Objects/TaskBar.js | Script |

## Amaryllis.Tools Namespace

### Bar Coder Class

This class is used internally to generate barcodes for every object saved in the database. The generated barcodes represent numbers between 1 and 999.999.999.999. For each code the platform generates an image which is saved in the Files/BarCodes folder.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| EncodingsLeft | Int[] | The array of 7bit codes for each of the 10 digits for the left side of the barcode. |
| EncodingsRight | Int[] | The array of 7bit codes for each of the 10 digits for the right side of the barcode. |
| Positions | Int | The image positions for each of the 12 digits in the barcode. |

The digit codes are as follows:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Digit | Left Code | Left Binary | Left Image | Right Code | Right Binary | Right Image |
| 0 | 13 | 0001101 |  | 114 | 1110010 |  |
| 1 | 25 | 0011001 |  | 102 | 1100110 |  |
| 2 | 19 | 0010011 |  | 108 | 1101100 |  |
| 3 | 61 | 0111101 |  | 66 | 1000010 |  |
| 4 | 35 | 0100011 |  | 92 | 1011100 |  |
| 5 | 49 | 0110001 |  | 78 | 1001110 |  |
| 6 | 47 | 0101111 |  | 80 | 1010000 |  |
| 7 | 59 | 0111011 |  | 68 | 1000100 |  |
| 8 | 55 | 0110111 |  | 72 | 1001000 |  |
| 9 | 11 | 0001011 |  | 85 | 1010101 |  |

#### Methods

**GetCodeFromImage**

Purpose: retrieves the code represented in the barcode.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FilePath | String | The absolute path for the file that contains the barcode. |

**GetImageForCode**

Purpose: generates the barcode image and returns the relative HTTP path for the generated image.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Code | Int | The code without the leading zeroes. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Tools/BarCoder.resx | Resource |
| Properties/Tools/BarCoder.RO-RO.resx | Resource |
| Images/Tools/BarCoder/Blank.png | Image |
| Images/Tools/BarCoder/Digit0.png | Image |
| Images/Tools/BarCoder/Digit1.png | Image |
| Images/Tools/BarCoder/Digit2.png | Image |
| Images/Tools/BarCoder/Digit3.png | Image |
| Images/Tools/BarCoder/Digit4.png | Image |
| Images/Tools/BarCoder/Digit5.png | Image |
| Images/Tools/BarCoder/Digit6.png | Image |
| Images/Tools/BarCoder/Digit7.png | Image |
| Images/Tools/BarCoder/Digit8.png | Image |
| Images/Tools/BarCoder/Digit9.png | Image |
| Scripts/Objects/TaskBar.js | Script |

### Error Class

This class is used internally in order to pack error messages together.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Message | String | The error messages packed. |

#### Methods

**Error**

Purpose: (constructor) initializes the Message property with the empty string.

**ThrowError**

Purpose: throws an exception based on the Message property if it contains a value different then the empty string.

Returns: NOTHING

Parameters: NONE

**AddMessage**

Purpose: encodes a new error message in the Message property.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Message | String | The error messages to be encoded. |

### Form Class

This class is responsible for rendering custom forms.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| SP | Amaryllis.Objects.SidePanel | Side panel class instance needed for rendering custom forms. |
| TP | Amaryllis.Objects.TabPanel | Top tab panel class instance used to render custom forms. |
| Identifier | Int | A number used to specify different identifiers for object listings and displays. |
| IDs | String | Contains the composed list of object IDs displayed. |
| MappingDeleteFunc | String | Retains the java script that must be used to delete a mapping. |
| LastClass | String | Keeps the name of the last encountered class’s name. |
| LastClassID | Int | Keeps the name of the last encountered class primary key value. |

#### Methods

**ParseNode**

Purpose: parses the specified XML node and substitutes HTML code for it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| XN | System.Xml.XmlNode | The XML node that needs to be parsed. |

**DisplayForm**

Purpose: displays the custom form according to the specified parameters.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FormID | Int | The primary key value of the custom form. |
| ObjectID | Int | The primary key value of the primary object. |
| MappingDeleteFunction | String | The java script that must be run when a mapping needs to be deleted. |
| MappingInsertFunction | String | The java script that must be run when a new object must be inserted. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Tools/Form.resx | Resource |
| Properties/Tools/Form.RO-RO.resx | Resource |
| FormAddMapping.aspx | Web page |

### Format Class

This class is responsible for formatting or encoding specific items.

#### Methods

**DataTableToXML**

Purpose: packs a data table into an XML structure.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| DT | System.Data.DataTable | The data table that must be encoded in XML form. |
| FileCols | String (optional) | The name of the columns from the data table that contain database files separated by the “,” character. |
| LinkCols | String (optional) | The name of the columns from the data table that contain http links separated by the “,” character. |

**StringToBase16String**

Purpose: converts a regular string into a base 16 representation.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The text that must be encoded. |

**Base16StringToString**

Purpose: converts a base 16 encoded string into a regular string.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The text that must be dencoded. |

**StringToBase64String**

Purpose: converts a regular string into a base 64 representation.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The text that must be encoded. |

**Base64StringToString**

Purpose: converts a base 64 encoded string into a regular string.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The text that must be decoded. |

**Int**

Purpose: formats an integer grouping digits in packs of 3. Sample: for 1234654 it will return 1 234 654.

Returns: String

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Number | Int | The number that must be formatted. |

### Graphics Class

This class is responsible for image processing and composing. Its functions are basic but usefull.

#### Methods

**PasteImage**

Purpose: copies the source image to the destination image at the given position.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Destination | Ref System.Drawing.Bitmap | A reference to the destination image. |
| Source | Ref System.Drawing.Bitmap | A reference to the source image. |
| X | Int | The horizontal position in pixels where the source image must be placed. |
| Y | Int | The vertical position in pixels where the source image must be placed. |

**PutPixel**

Purpose: modifies the specified pixel from the given image.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| X | Int | The horizontal position of the pixel that will be altered. |
| Y | Int | The vertical position of the pixel that will be altered. |
| C | System.Drawing.Color | The new color for the given pixel. |
| Transparency | Byte (optional) | The transparency for the given pixel. |

**DrawLine**

Purpose: draws a line on the specified image between the two specified points.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| X1 | Int | The horizontal position of the first pixel. |
| Y1 | Int | The vertical position of the first pixel. |
| X2 | Int | The horizontal position of the second pixel. |
| Y2 | Int | The vertical position of the second pixel. |
| C | System.Drawing.Color | The color for the line. |
| Transparency | Byte (optional) | The transparency for the line. |

**DrawRectangle**

Purpose: draws a rectangle considering the two specified points as diagonally oppose.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| X1 | Int | The horizontal position of the first pixel. |
| Y1 | Int | The vertical position of the first pixel. |
| X2 | Int | The horizontal position of the second pixel. |
| Y2 | Int | The vertical position of the second pixel. |
| C | System.Drawing.Color | The color for the rectangle’s border. |
| FillColor | System.Drawing.Color (optional) | The color that will fill the rectangle. |
| Transparency | Byte (optional) | The transparency for the rectangle. |

**DrawCircle**

Purpose: draws a circle on the given image.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| X | Int | The horizontal position of the circle’s center. |
| Y | Int | The vertical position of the circle’s center. |
| Radius | Int | The radius dimension in pixels. |
| C | System.Drawing.Color | The color for the circle’s border. |
| FillColor | System.Drawing.Color (optional) | The color that will fill the circle. |
| Transparency | Byte (optional) | The transparency for the circle. |

**DrawShape**

Purpose: draws a random shape on the given image.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| Points | Int[] | The vector of point coordonates. |
| C | System.Drawing.Color | The color for the circle’s border. |
| Transparency | Byte (optional) | The transparency for the circle. |

**Fill**

Purpose: fills a polygon limited by pixels of a given color with a specified color.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| X | Int | The horizontal position of the start pixel. |
| Y | Int | The vertical position of the start pixel. |
| C | System.Drawing.Color | The color for the circle’s border. |

**TransparencyKey**

Purpose: makes all the pixels of the specified color transparend according to the provided transparency key.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| B | Ref System.Drawing.Bitmap | A reference to the destination image. |
| C | System.Drawing.Color | The color that must be altered. |
| Transparency | Byte | The transparency key. |

### Image Class

This class is responsible with the rendering the images on the platform.

#### Methods

**ReduceImage**

Purpose: reduces the specified image to the dimension specified.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FilePath | String | The absolute path of the image file. |
| MaxDimension | Int | The maximum dimension of the image after shrinking. |

**GetImageHTML**

Purpose: retrieves the HTML code that displays the picture on a web page.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FilePath | String | The absolute path of the image file. |
| MaxDimension | Int (optional) | The maximum dimension for the image. |
| WithZoom | Bool (optional) | Integrates the image with the zoom feature of the platform. |

### LaTeX Class

This class is responsible for transforming LaTeX formulas into images in order for the platform to be able to display them.

#### Methods

**LaTeX2HTML**

Purpose: transforms the LaTeX formula into an image.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LaTeXFormula | String | The LaTeX formula. |
| ImagePath | String | The relative http path of the file that will contain the formula. |

### Mapping Class

This class is used to display mapped objects and mapped search results.

#### Methods

**PlaceScripts**

Purpose: writes a specific HTML line in order to include the needed java script on a web page.

Returns: NOTHING

Parameters: NONE

**DisplayMappingSearch**

Purpose: displays the list of objects of the given class that correspond to the given search criteria.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed objects. |
| ClassID | Int | The primary key value of the class. |
| SearchCriteria | String | The search criteria. |
| SelectFunction | String | The java script that must be run when a found object is selected. |

**DisplayMappingsForObjectAndClass**

Purpose: displays the list of objects of the given class mapped with the given object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |
| ObjectID | Int | The primary key value of the primary object. |
| DeleteFunction | String | The java script that must be run when a mapping needs to be deleted. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Tools/Mapping.resx | Resource |
| Properties/Tools/Mapping.RO-RO.resx | Resource |
| Scripts/Tools/Mapping.js | Script |

### Object Class

This class is used to display object listings and object forms on web pages.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HiddensFlag | Bool | Marks the fact that the hidden fields need to restore objects were displayed or not. The start value is “false”. |

#### Methods

**PlaceScripts**

Purpose: writes a specific HTML line in order to include the needed java script on a web page.

Returns: NOTHING

Parameters: NONE

**EncodeFieldValue**

Purpose: encodes a token field/attribute in the specified HTML form.

Returns: string (representing the HTML form description with the field/attribute token encoded)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Script | String | The description of the form in specific tokened HTML. |
| Field | String | The name of the field. |
| Value | object | Generic field value. |

**DisplaySearchResultForClass**

Purpose: displays the list of objects of the given class specified in the list of primary key values given.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the primary class. |
| IDs | String | The primary key values of all the objects that must be displayed separated by the “,” character. |
| Weights | String | The weights for the objects that must be listed. |

**DisplayRecyclerBinForClass**

Purpose: displays the list of deleted objects for the give class.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the class. |

**DisplayMultiLookListForClass**

Purpose: displays the list of objects for the given class and parameters.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed object list. |
| ClassID | Int | The primary key value of the primary class. |
| ParentID | Int | The primary key value of the parent object. |
| WithSorting | Bool | Specify if the sorting bar will be displayed or not. |
| WithFiltering | Bool | Specify if the filtering bar will be displayed or not. |
| WithMapping | Bool | Specify if the mapping bar will be displayed or not. |
| WithPagination | Bool | Specify if the pagination bar will be displayed or not. If no pagination bar is displayed, only the first page will be shown. |

**GetObjectErrorID**

Purpose: returns the error hidden field ID for the object form marked with the given identifier.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed object form. |

**SetObjectError**

Purpose: sets the error for the specified form.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed object form. |
| ErrorMessage | String | The error message. |

**ResetObjectError**

Purpose: resets the error for the object form specified.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed object form. |

**GetObjectError**

Purpose: returns the error message for the object form marked with the given identifier.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed object form. |

**ExportToHTML**

Purpose: exports the form for the given object to a file.

Returns: string (representing the file’s absolute path)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | Int | The primary key value of the object. |

**DisplaySingleInstanceForObject**

Purpose: displays the form for the given object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The basic identifier for the displayed object form. |
| ObjectID | Int | The primary key value of the object. |
| ParentID | Int | The primary key value of the object’s parent. |
| ClassID | Int | The primary key value of the object’s class. |
| WithBarCode | Bool (optional) | Specifies if the object form will display the barcode as well. The default value is “true”. |

**DisplaySingleInstanceForObjectLog**

Purpose: displays an object form in read-only mode for an object log instance.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LogID | Int | The primary key value of the object’s log. |

**DisplayLogTitle**

Purpose: displays the given object’s log title.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| LogID | Int | The primary key value of the object’s log. |

**DisplayTitle**

Purpose: displays the given object’s title.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ClassID | Int | The primary key value of the object’s class. |
| ObjectID | Int | The primary key value of the object. |
| ParentID | Int | The primary key value of the object’s parent. |

**DisplayHelpButton**

Purpose: displays a help button that links the specified article.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ArticleID | Int | The primary key value of the article. |

**ThrowError**

Purpose: adds the specified error to the list of error for the active session.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Error | String | The error message(s). |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Tools/Object.resx | Resource |
| Properties/Tools/Object.RO-RO.resx | Resource |
| Scripts/Tools/Object.js | Script |

### Packer Class

This class is used to pack attributes and values in the specific XML structure.

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Pack | String | The pack of attributes and values without the root node. |

#### Methods

**Packer**

Purpose: (constructor) initializes the Pack member with the empty string.

**EncodeText**

Purpose: encodes the given text in order to encapsulate it in an XML structure.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The text that needs to be encoded. |

**Append**

Purpose: appends a new attribute and value to the pack.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Attribute | String | The name of the attribute. |
| Value | String | The value for the attribute. |

**GetPack**

Purpose: retrieves the pack XML structure.

Returns: string

Parameters: NONE

**Clear**

Parameters: clears the packed attributes and values.

Returns: NOTHING

Parameters: NONE

### Report Class

This class is responsible for displaying report results, report graphs and report parameter forms.

#### Methods

**PlaceScripts**

Purpose: writes a specific HTML line in order to include the needed java script on a web page.

Returns: NOTHING

Parameters: NONE

**GetReportLogDate**

Purpose: retrieves the report history date.

Returns: string (represents the date and time in the “DD/MM/YYYY HH:MM” format)

Parameters: NONE

**DisplayGraph**

Purpose: processes the given graph and displays it using the specified Chart object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| C | Ref System.Web.UI.DataVizualization.Charting.Chart | A reference for the Chart object. |
| GraphID | Int | The primary key value of the graph. |
| Params | String | The parameters and values encoded in the specific XML structure. |
| Date | String | The date and time in the “DD/MM/YYYY HH:MM” format. |

**DisplayReportData**

Purpose: displays the report data on the web page.

Returns: bool (specifies if the report has data to display or not)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetParametersForReport**

Purpose: retrieves the specific XML structure containing the parameters and values for the given report.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**DisplayReportDataFree**

Purpose: triggered by the DisplayReportData method for the reports which do not impose a strict display of their entries.

Returns: bool (specifies if the report has data to display or not)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**DisplayReportDataStrict**

Purpose: triggered by the DisplayReportData method for the reports which impose a strict display of their entries.

Returns: bool (specifies if the report has data to display or not)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**DisplayReportParameterForm**

Purpose: displays the parameter form for the specified report.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

**GetReportTitle**

Purpose: retrieves the report’s title.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ReportID | Int | The primary key value of the report. |

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Tools/Report.resx | Resource |
| Properties/Tools/Report.RO-RO.resx | Resource |
| Scripts/Tools/Report.js | Script |

### Security Class

This class is used to filter texts in order to avoid the cross site scripting vulnerability.

#### Methods

**CrossSiteScripting**

Purpose: encodes the specified text in order to avoid cross site scripting.

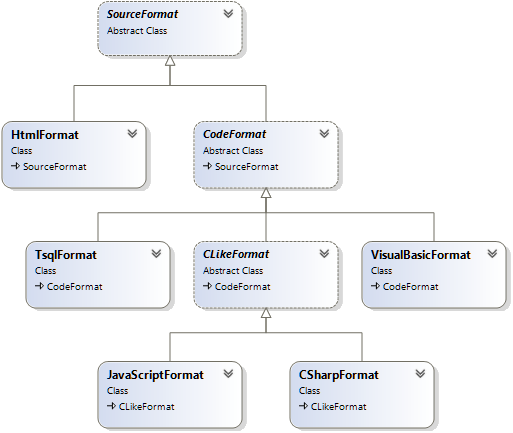
Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The text that needs to be encoded. |
| WithTrimming | Bool (optional) | Specifies if the text should be trimmed or not. The default is “false”. |

### Amaryllis.Tools.CodeHighlighter Namespace

This namespace hold the classes that are used to format programming code in articles. The classes from this namespace are derived from one another, so for a better understanding we provided the schema bellow.



Class diagram for the CodeHighlighting module

#### CLikeFormat Class

This class derives from the CodeFormat class and it is used in order to format C/C++ like languages code.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CommentRegEx | String | The regular expression for recognising code comments. |
| StringRegEx | String | The regular expression for recognising string values. |

#### CodeFormat Class

This class is derives from the SourceFormat classa it is responsible for four language formatting.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Keywords | Abstract String | The list of keywords separated by spaces. |
| Preprocessors | Virtual String | The list of pre-processed keywords separated by spaces. |
| StringRegEx | Abstract String | The regular expression for recognising string values. |
| CommentRegEx | Abstract String | The regular expression for recognising code comments. |
| CaseSensitive | Virtual Bool | Marks the language as being case sensitive. |

##### Methods

**CodeFormat**

Purpose: (constructor) setups the regular expressions parsers from the parent class.

**MatchEval**

Purpose: returns the next token in the source file encoded in HTML determining the way that the token will be displayed.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| match | System.Text.RegularExpressions.Match |  |

#### CSharpFormat

This class is derived from the CLikeFormat class and it is used to format C# code sequences.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Keywords | String | The list of key words for the C# language. |
| Preprocessors | String | The list of pre-processed key words for the C# language. |

#### Highlighter Class

This class is the main class for this namespace and it is used to format all programming languages according to the specified language.

##### Methods

**Highlight**

Purpose: formats the given code sequence for the specified language.

Returns: string (representing the code sequence formatted for HTML display)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Code | String | The code sequence. |
| Language | Amaryllis.Tools.CodeHighlighter.Languages | The language for which the code is parsed. |

**GetLanguageTypeFromString**

Purpose: gets the language type from a string.

Returns: Amaryllis.Tools.CodeHighlighter.Languages

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Language | String | The language name. |

#### HtmlFormat Class

This class is responsible for formatting HTML code blocks.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| csf | Amaryllis.Tools.CodeHighlighter.CSharpFormat | A C# formatting class needed for the server side code. |
| jsf | Amaryllis.Tools.CodeHighlighter.JavaScriptFormat | A java script formatting class needed for the java script code. |
| attribRegex | System.Text.RegularExpression.Regex | A regular expression object needed for parsing the code block. |

##### Methods

**HtmlFormat**

Purpose: (constructor) initializes the private members for the class.

**AttributeMatchEval**

Purpose: identifies and encodes the next HTML attribute token in the code block.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| match | System.Text.RegularExpressions.Match |  |

**MatchEval**

Purpose: identifies and encodes the next java script token in the code block.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| match | System.Text.RegularExpressions.Match |  |

#### JavaScriptFormat Class

This class is derived from the CLikeFormat class and it is used to format java script code sequences.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Keywords | String | The list of key words for the JavaScript language. |
| Preprocessors | String | The list of pre-processed key words for the JavaScript language. |

#### SourceFormat Class

This is the base class for all the language classes.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| TabSpaces | Byte | The number of spaces that replace a tab. |
| LineNumbers | Bool | Specifies if the line numbers must be displayed. |
| Alternate | Bool | Triggers the alternation of row color. |
| EmbededStyleSheet | Bool |  |
| CodeRegex | System.Text.RegularExpression.Regex | A regular expression object. |

##### Methods

**SourceFormat**

Purpose: (constructor) initializes the properties for the class.

**FormatCode**

Purpose: formats the given code sequence.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| source | System.IO.Stream or String | The code sequence or the stream that contains the code sequence. |
| lineNumbers | Bool | Specifies if the line numbers must be displayed. |
| alternate | Bool | Triggers the alternation of row color. |
| embededStyleSheet | Bool |  |
| subCode | Bool | Marks the sequence as being a sub code sequence. |

**FormatSubCode**

Purpose: formats a sequence of code.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| source | String | The code sequence that contains the code sequence. |

**MatchEval**

Purpose: the prototype design for the MatchEval method.

Returns: string

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| match | System.Text.RegularExpressions.Match |  |

#### TSqlFormat Class

This class is derived from the CodeFormat class and it is used to format SQL code sequences.

##### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CommentRegEx | String | The regular expression for comments. |
| StringRegEx | String | The regular expression for string values. |
| CaseSensitive | Bool | Marks the language as not being case sensitive. |
| Keywords | String | The list of keywords for the SQL language. |
| Preprocessors | String | The list of pre-processed keywords for the SQL language. |

## Amaryllis.Widgets Namespace

This namespace contains all the classes that are responsible for the widget displays on the platform’s title bar.

### About Class

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

#### Methods

**About**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

**PlaceWindow**

Purpose: writes the HTML code used by the widget in order to display the window associated to it.

Returns: NOTHING

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Widgets/About.resx | Resource |
| Properties/Widgets/About.RO-RO.resx | Resource |
| Images/Widgets/About/Icon.png | Image |
| Scripts/Resources/EN-UK/Widgets\_About.js | Script |
| Scripts/Resources/RO-RO/Widgets\_About.js | Script |
| Scripts/Widgets/About.js | Script |

### BarCoder Class

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

#### Methods

**BarCoder**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

**PlaceWindow**

Purpose: writes the HTML code used by the widget in order to display the window associated to it.

Returns: NOTHING

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Widgets/BarCoder.resx | Resource |
| Properties/Widgets/BarCoder.RO-RO.resx | Resource |
| Images/Widgets/BarCoder/Icon.png | Image |
| Scripts/Resources/EN-UK/Widgets\_BarCoder.js | Script |
| Scripts/Resources/RO-RO/Widgets\_BarCoder.js | Script |
| Scripts/Widgets/BarCoder.js | Script |

### Impersonation Class

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

#### Methods

**Impersonation**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Widgets/Impersonation.resx | Resource |
| Properties/Widgets/Impersonation.RO-RO.resx | Resource |
| Images/Widgets/Impersonation/Icon.png | Image |

### Language Class

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

#### Methods

**Language**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

**PlaceWindow**

Purpose: writes the HTML code used by the widget in order to display the window associated to it.

Returns: NOTHING

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Widgets/Language.resx | Resource |
| Properties/Widgets/Language.RO-RO.resx | Resource |
| Images/Widgets/Language/Icon.png | Image |
| Scripts/Widgets/Language.js | Script |

### Notification Class

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

#### Methods

**Notification**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

**PlaceWindow**

Purpose: writes the HTML code used by the widget in order to display the window associated to it.

Returns: NOTHING

Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Widgets/Notification.resx | Resource |
| Properties/Widgets/Notification.RO-RO.resx | Resource |
| Images/Widgets/Notification/Icon.png | Image |
| Scripts/Resources/EN-UK/Widgets\_Notification.js | Script |
| Scripts/Resources/RO-RO/Widgets\_Notification.js | Script |
| Scripts/Widgets/Notification.js | Script |

### Zoom Class

#### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

#### Methods

**Zoom**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

**PlaceWindow**

Purpose: writes the HTML code used by the widget in order to display the window associated to it.

Returns: NOTHING

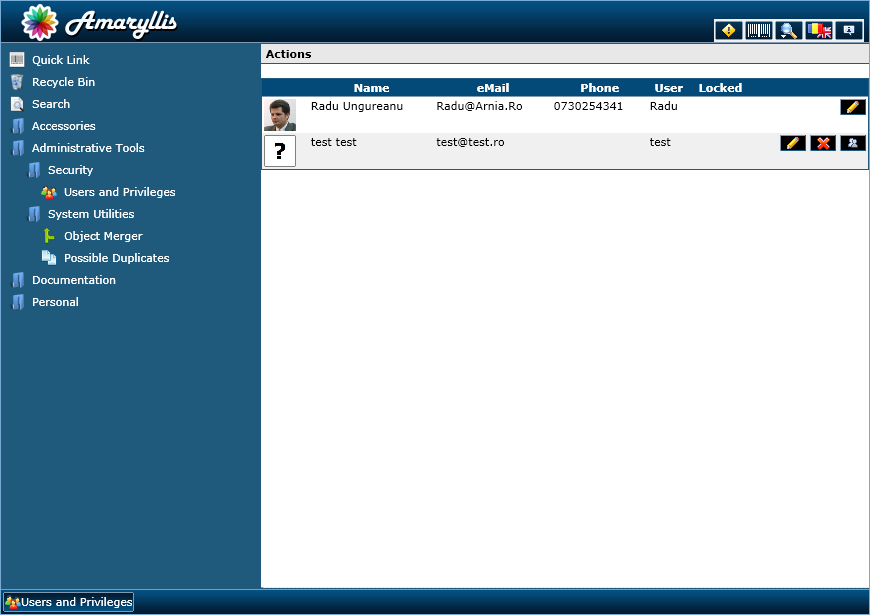
Parameters: NONE

#### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Properties/Widgets/Zoom.resx | Resource |
| Properties/Widgets/Zoom.RO-RO.resx | Resource |
| Images/Widgets/Zoom/Icon.png | Image |
| Scripts/Widgets/Zoom.js | Script |
| Scripts/Widgets/ZoomWindow.js | Script |

# Client Interface Layer

For this part we have two projects created, the PlainInterface and WindowInterface projects. The two projects represent two possible interfaces for the platform, but only the WindowInterface project offers support for class, report and custom form development. The PlainInterface project may offer them as well but the web pages for this part were not ported because the code for this part may change and there is no need to keep the code in two places.

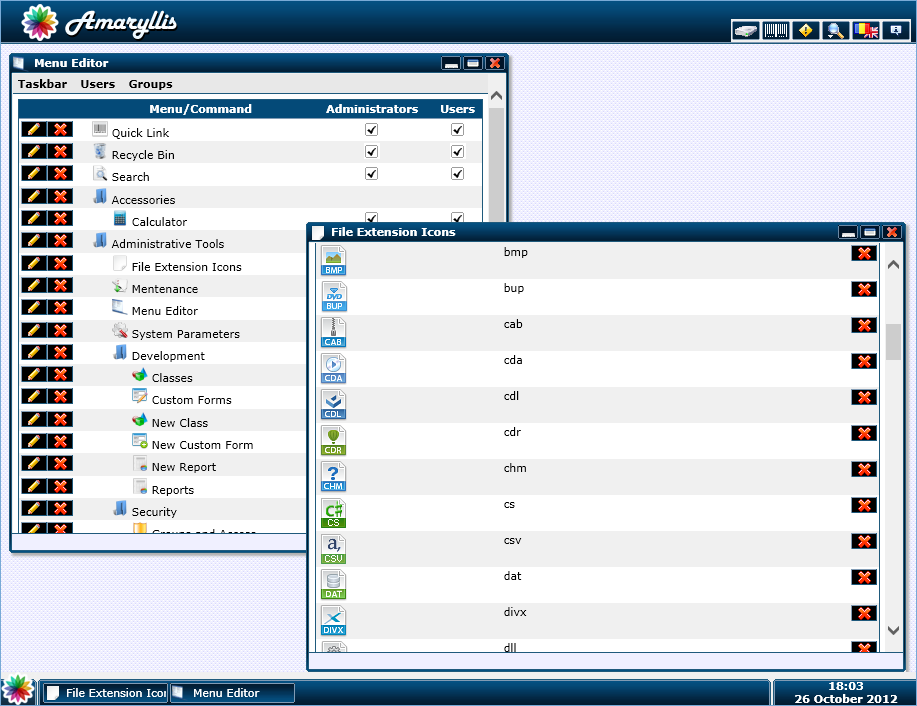


Plain interface display

The plain interface is a classical web interface (header, side menu-content, footer layout). The header displays the platform’s logo and the widgets buttons. The menu is displayed on the right side in a tree structure and each link opens a page in the content section. On the footer bar we have something similar to breadcrumbs; if a page is opened from a loaded page it will keep the breadcrumb for the old page on the footer bar.

As said, the functionality for this interface is limited to only the functionality destined for the final user. This is an interface destined for the users that are not familiar or find it difficult to work with windowed web interfaces.

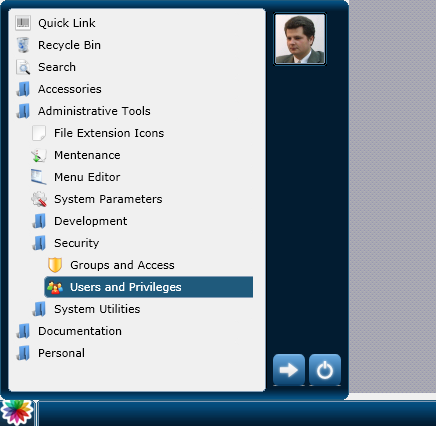
In the figure above you can see the “Users and Privileges” web page opened. Also you can see the tree menu opened on a few levels just to get the idea and the breadcrumb start on the footer bar. Next we will describe a bit the window interface as seen in the figure bellow.



Window interface display

As you can see, this interface is a bit elaborated. The header bar displays the same things but the footer bar differs in functionality. The footer bar is made similar to the task bar in Windows; you can see in the left corner the start button and the time and date in the right corner. The content section is at this point a surface on which windows are displayed. As you can see above, the “Menu Editor” and “File Extension Icons” are displayed and they are also shown in the footer bar bellow.

The displayed windows behave similar to the ones displayed in Windows. The user can maximize them, minimize them and move them around. The menu displayed on the plain interface on the left side can be triggered on the windows interface by clicking the start button found on the lower left corner as seen in the figure bellow.



Menu triggered in the window interface

The first interface that we wish to detail is the window interface because it covers all the web pages and functionality for the web pages thus the plain interface will be explain at the same time leaving us with the option to reference chapters from the window interface when we explain the plain interface.

## Window Interface

In order to properly describe this interface we will start by describing the interface’s elements. After this we will pass through all the web pages and detail their functionality.

### Programming Classes

This chapter covers the different development classes that are particular to this interface and that are outside the business logic layer. The classes described here will be grouped by namespace.

#### Amaryllis.Objects Namespace

##### Page Class

This class is responsible for the web pages and window pages design. All of the methods for this class can be called similar to html tags just to keep a clear structure of the web pages. Bellow you can see the way to call the methods for a main web page and a window web page.

|  |  |
| --- | --- |
| <%Amaryllis.Objects.Page P = new Amaryllis.Objects.Page(); %>  <%P.BeginPage(); %>  <%P.EndPage(); %> | <%Amaryllis.Objects.Page P = new Amaryllis.Objects.Page(); %>  <%P.BeginWindow(); %>  <%P.EndWindow(); %> |
| Main page | Window page |

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |
| UserID | Int | The primary key value for the currently logged in user. |
| Months | String[] | The array of calendar months. |

###### Methods

**Page**

Purpose: (constructor) initializes the Response and UserID properties.

**BeginPage**

Purpose: writes the HTML code that begins the enclosing main web page.

Parameters: NONE

**EndPage**

Purpose: writes the HTML code that closes the enclosing main web page.

Parameters: NONE

**Taskbar8**

Purpose: writes the HTML code that displays the Windows 8 style taskbar.

Parameters: NONE

**Taskbar7**

Purpose: writes the HTML code that displays the Windows 7 style taskbar.

Parameters: NONE

**BeginWindow**

Purpose: writes the HTML code that begins the enclosing window web page.

Parameters: NONE

**EndWindow**

Purpose: writes the HTML code that closes the enclosing window web page.

Parameters: NONE

**DisplayErrors**

Purpose: writes the code that displays the encountered errors on the web page.

Parameters: NONE

**ThrowError**

Purpose: adds a new set of errors on the existing error stack.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Errors | String | The error messages separated by the “|” character. |

**PlacePageScripts**

Purpose: writes the HTML code in order to include the needed java scripts on the main web page.

Parameters: NONE

**PlaceWindowScripts**

Purpose: writes the HTML code in order to include the needed java scripts on the window web page.

Parameters: NONE

**PlaceResourceScript**

Purpose: writes the HTML code that includes the given resource java script on the web page for the currently interface language.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileName | String | The name of resource java script file that must be included on the web page. |

**PlaceResourcesPageScript**

Purpose: triggers the platform to include the proper resources java script file on the current web page.

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Objects/Page/Objects\_Page.resx | Resource |
| App\_GlobalResources/Objects/Page/Objects\_Page.RO-RO.resx | Resource |
| Images/Objects/Page/Background.png | Image |
| Images/Objects/Page/Bullet.png | Image |
| Images/Objects/Page/FooterBar.png | Image |
| Images/Objects/Page/FooterCornerLeft.png | Image |
| Images/Objects/Page/FooterCornerRight.png | Image |
| Images/Objects/Page/HeaderBar.png | Image |
| Images/Objects/Page/LogoutButton.png | Image |
| Images/Objects/Page/ShutdownButton.png | Image |
| Images/Objects/Page/StartButton.png | Image |
| Images/Objects/Page/Taskbar\_Component\_1\_1.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_1\_2.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_1\_3.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_1\_4.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_1\_5.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_2\_1.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_2\_2.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_2\_3.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_3\_1.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_3\_2.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_3\_3.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_3\_4.png | Image |
| Images/Objects/Page/ Taskbar\_Component\_3\_5.png | Image |
| Images/Objects/Page/ Taskbar\_PictureFrame.png | Image |
| Images/Objects/Page/ Taskbar8BG.png | Image |
| Scripts/Objects/Page.js | Script |
| Scripts/Resources/EN-UK/Objects\_Page.js | Script |
| Scripts/Resources/RO-RO/Objects\_Page.js | Script |

##### TreeMenu Class

This class is responsible for the tree structure displayed on the taskbar and the virtual drive folder structure. The methods for this class are used internally so we will only concentrate on the properties and methods without showing an example on how to aggregate them.

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |
| Level | Int | Holds the hierarchy level for the tree menu. |
| Count | Int | Holds the option count and contributes to the created objects HTML IDs. |
| Version | String | Holds the style for the tree structure that must be contructed. |

###### Methods

**TreeMenu**

Purpose: (constructor) initializes the properties.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Version | String (optional) | Specifies the tree menu version. |

**Reset**

Purpose: resets the Level property to zero.

Parameters: NONE

**BeginMenu**

Purpose: starts a new encapsulating menu object with the specified title.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The menu’s title. |

**AddScriptCommand**

Purpose: adds a new menu command that runs the specified java script code fragment when it is triggered.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The command’s title. |
| JavaScriptCode | String | The java script code fragment that must be run when the command is triggered. |

**AddCommand**

Purpose: adds a new command to the menu.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The command’s title. |
| Icon | String (optional) | The relative HTTP path for the pictogram associated with the command. |
| Link | String | The HTTP path for the web page that must be opened when the command is triggered. |

**EndMenu**

Purpose: closes the current encapsulating menu.

Parameters: NONE

**DisplayMenuForUser8**

Purpose: generates the HTML code for a Windows 8 style menu for the current logged on user.

Parameters: NONE

**DisplayMenuForUser7**

Purpose: generates the HTML code for a Windows 7 style menu for the current logged on user.

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Objects/TreeMenu.js | Script |
| Scripts/Resources/EN-UK/Objects\_TreeMenu.js | Script |
| Scripts/Resources/RO-RO/Objects\_TreeMenu.js | Script |

#### Amaryllis.Widgets Namespace

##### Harddrive Widget

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

###### Methods

**Harddrive**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

**PlaceWindow**

Purpose: writes the HTML code used by the widget in order to display the window associated to it.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Widgets/Harddrive/Widgets\_Harddrive.resx | Resource |
| App\_GlobalResources/Widgets/Harddrive/Widgets\_Harddrive.RO-RO.resx | Resource |
| Images/Widgets/Harddrive/FolderClosed.png | Image |
| Images/Widgets/Harddrive/FolderOpened.png | Image |
| Images/Widgets/Harddrive/Icon.png | Image |
| Images/Widgets/Harddrive/Minus.png | Image |
| Images/Widgets/Harddrive/None.png | Image |
| Images/Widgets/Harddrive/Plus.png | Image |
| Scripts/Resources/EN-UK/Widgets\_Harddrive.js | Script |
| Scripts/Resources/RO-RO/Widgets\_Harddrive.js | Script |
| Scripts/Widgets/Harddrive.js | Script |

##### Harddrive Widget

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |

###### Methods

**Login**

Purpose: (constructor) initializes the Response property.

**PlaceButton**

Purpose: writes the HTML code that displays the widget’s button.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Widgets/Harddrive/Icon.png | Image |

##### TabScroller Widget

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |
| Item | Int | Counter for the number of objects from a tabscroller object. |
| Count | Int | Counter for the number of tabscroller objects. |
| BigWidth | Int | The width in pixels for a full size panel. |
| SmallWidth | Int | The width in pixels for a compacted panel. |
| Height | Int | The height in pixels for a panel. |

###### Methods

**TabScroller**

Purpose: (constructor) initializes the Response, Item and Count properties.

**PlaceScripts**

Purpose: writes the HTML code in order to include the proper java scripts files on the web page.

**BeginObject**

Purpose: writes the HTML code that starts an encapsulating object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| DivBigWidth | Int | The width in pixels for a full size panel. |
| DivSmallWidth | Int | The width in pixels for a compact panel. |
| DivHeight | Int | The height in pixels for a panel. |

**BeginItem**

Purpose: writes the HTML code that begins an encapsulating panel object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Name | String | The title for the panel. |
| HorizontalImage | String | The relative HTTP path for the title image displayed when the panel is fully displayed. |
| VerticalLeftImage | String | The relative HTTP path for the title image displayed when the panel is compacted on the left side of the scroller object. |
| VerticalRightImage | String | The relative HTTP path for the title image displayed when the panel is compacted on the right side of the scroller object. |

**EndItem**

Purpose: writes the HTML code that closes an encapsulating panel object.

Returns: NOTHING

Parameters: NONE

**EndObject**

Purpose: writes the HTML code that closes an encapsulating tabscroller object.

Returns: NOTHING

Parameters: NONE

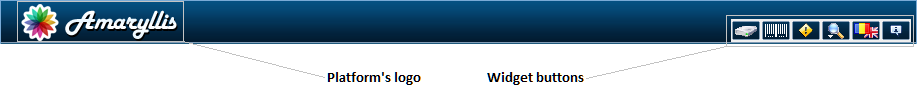
###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Widgets/TabScroller.js | Script |

### Interface Elements

#### Header Bar

As describe above, we have the following items on the title bar.

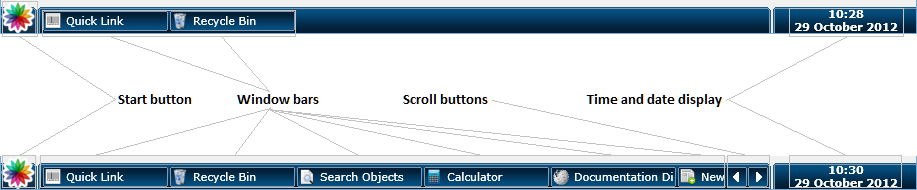


Header bar and elements

We have the logo in the left side of the header bar and on the right corner we have the widget buttons. All of these items have to fit on a 43 pixels height space.

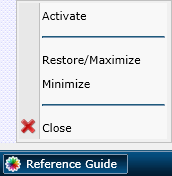
#### Footer Bar

For this interface the footer bar acts similar to the taskbar from Windows. It displays the time and date on the right side and a start button on the left side. The center part is, as shown below, used to display the window bars for all the opened windows. If the window bars do not fit on the given space, scrolling buttons will be displayed.



Footer bar and elements

Each of the window bars has a context menu assign to it that contains the options available for it and the associated window on the content section.



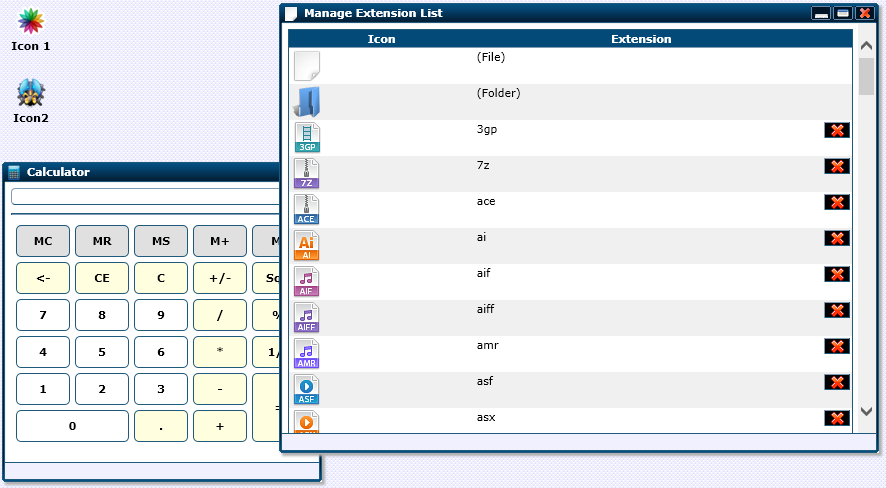
Context menu associated with the window bar

The options for an active window are the ones seen in the figure above:

* Activate: will bring a window to front and restore it to its dimensions if it is minimized
* Restore/Maximize: maximizes a window or restores it to its original dimensions before it was maximized depending on the window state
* Minimize: minimizes a window to the taskbar
* Close: closes the window and removes it from the taskbar.

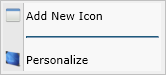
#### Content Section

The content section is only a desktop on which icons and windows can be displayed as shown in the figure bellow.



Content section and elements

The content section has its own context menu just like in Windows in order for the user to change preferences or add icons to his desktop.



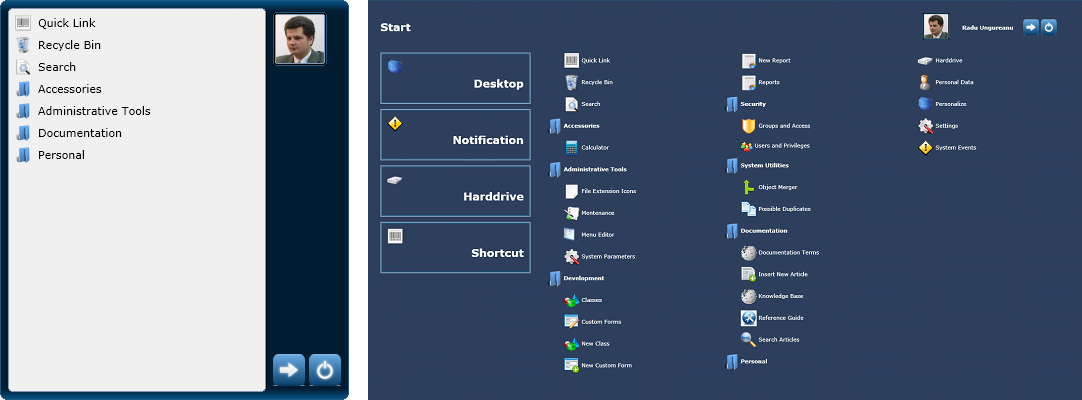
The content section context menu

The options seen above are the following:

* Add New Icon: triggeres a new window that helps the user add a new icon to his content section
* Personalize: opens a new window that helps the user change his preferences regarding the taskbar display and background image for the content section.

#### Taskbar

The taskbar element for this interface can be configured in two ways design wise. The first and simplest form is similar to the Windows 7 form and the second one is similar to the Windows 8 form. This was done just to have the option of configuring the interface to something more familiar to the users but each user can choose its own display way.

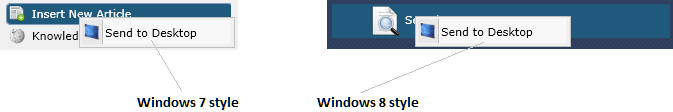


Taskbar displays in Windows 7 and 8 styles

The Windows 7 style displays the menu in a tree structure and shows the logout and exit buttons in the lower right corner. Similar to the Windows 7 taskbar, the user image/pictogram is displayed in the right top corner and triggers the window that displays the user’s data (detailed in a later chapter).

The Windows 8 style displays a few options on the left side and the rest are displayed as a grouping structure on the right side. The logout and exit buttons are on the right upper corner as well as the user’s image/pictogram that triggers the user’s data web page.

Each option in the taskbar opens a new window on the content section with the title and icon with which it is registered on the taskbar, also each option in the taskbar has a context menu assign to it as seen in the figure bellow.



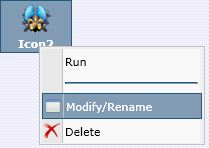
Context menu display for the taskbar options

The context’s menu options are the following:

* Send to Desktop: adds a new icon to the content section with the same pictogram and title as the taskbar menu option that triggers the same window.

#### Icon Element

The simple icon element is taken from the Windows operating system and ported to the platform. Each icon triggers a new window opening the specified web page on the double-click option. Similar to the known functionality of a desktop icon, the platform’s icons can be dragged and moved on the content section and also possess a context menu as seen in the figure bellow.



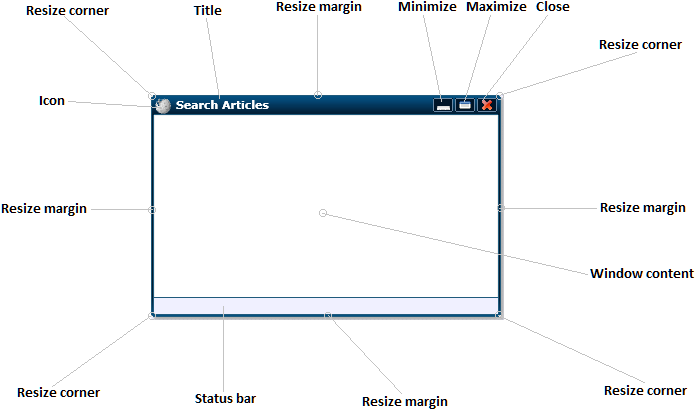
The context menu for an icon

The icon context menu items are the following:

* Run: triggers the icon to open a new window with the specified web page
* Modify/Rename: opens a new window with the web page design to facilitate the alteration for the icon
* Delete: removes the icon from the user’s content section

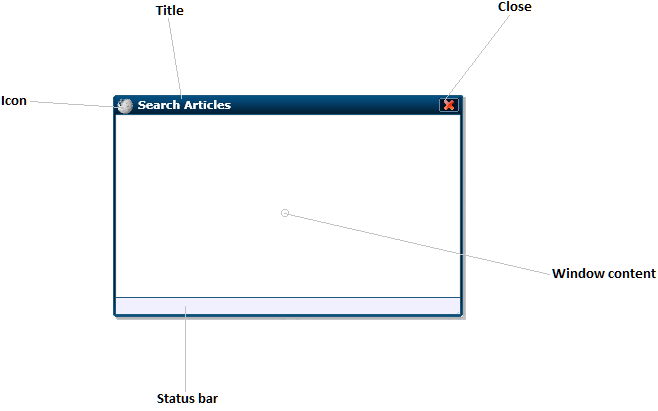
#### Window Element

A very important element for the content section is the window element which loads web pages. The figure bellow illustrates the window element with all of its buttons and triggers.



Regular window element features

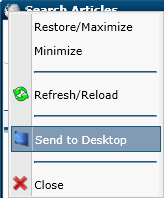
Complementing the regular window above we have the fixed size window which excludes some of the functionalities and buttons of the regular window.



Fixed border window element features

Window elements can be moved around on the content section and minimized to the footer section just like on a regular operating system. As an exception, and a natural one, window elements can’t be moved around if they are maximized.

As expected, the window element has a context menu assign to it that activates by clicking on the title bar icon.



Window element context menu

The options are as follows:

* Restore/Maximize: restores a window to its original dimensions before maximizes or maximizes the window
* Minimize: minimizes the window to the footer bar
* Refresh/Reload: triggers a reload for the displayed page
* Send to Desktop: creates a new desktop icon with the tile and pictogram of the window that triggers the displayed page
* Close: closes the window and removes it from the footer bar

### Widgets

#### Login Widget

The login widget triggers the login mode switching from the enhanced mode to the normal mode. The enhanced mode is ok for a small number or users but for a large number of users it works slower due to the number of user images that must be loaded. The normal mode is a clean user and password form which loads the user picture when a valid user is filled in.



Enhanced login form

By clicking on the login widget button () found on the top right corner of the form the user will be redirected to the normal login web page.

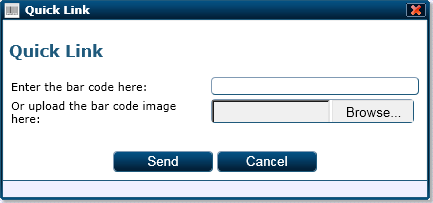


Normal login form

The default login form can be set using the “LoginMode” constant from the database. This constant can be set by using the “System Parameters” form found in the menu tree structure.

#### Quick Link Widget

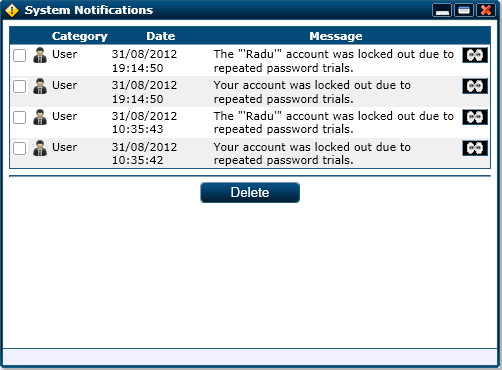
This widget () triggers a form that is designed to open an object form from the object’s barcode. The barcode can be sent as text or bar code image. The web page itself will be described later on but at this moment we will provide the image for that web page.



Quick link web page adjacent to the quick link widget

#### Notification Widget

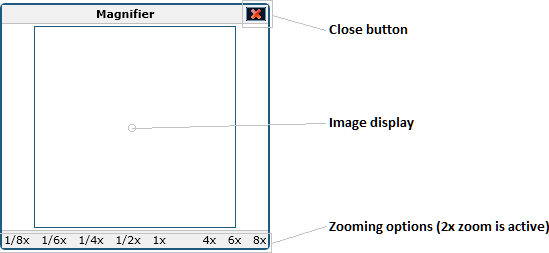
This widget () triggers a window that displays the system notification for the current logged on user. The web page for this widget will be detailed later on but we will include an image here as well.



Notifications web page adjacent to the notification widgets

#### Zoom Widget

The zoom widget is integrated into the platform and can be triggered by clicking in a registered image or by clicking the () button found on the right upper corner of the interface. The registered picture is displayed in the following form and manipulated as shown by the comments on the figure bellow.



The zoom widget adjacent form

The widget window offers the possibility to zoom the picture at the following levels: 1/8x, 1/6x, 1/4x, 1/2x, 1x, 2x, 4x, 6x and 8x. By default a register image is opened zoomed at 2x.

#### Language Widget

The language widget () is responsible for switching the interfaces language. At this point the platform comes in two different languages (English and Romanian) but languages can be added to the platform by using an application adjacent to the platform.

By clicking the language widget button a small bar will be opened which displays the available languages. When selecting a language the whole interface will be refreshed and the language option will be saved for the logged on user. The default language can be set using the constant named “Language”.



Language options given by the language widget

#### About Widget

The about widget () displays a small fixed border window containing the platform’s characteristics. You can see a sample for this window below for the 2.5.55 version of the platform.



About window displayed by the about widget

#### Impersonation Widget

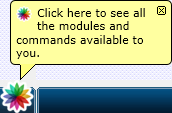
The impersonation widget () is displayed when the current logged on user is impersonated. By triggering this widget the impersonation stops and the original user’s session is restored.

### Taskbar Section

The taskbar section takes all the footer section regarding space. The elements displayed here are similar to the ones displayed on a Windows taskbar and we cover them briefly so in this chapter we will detail them a bit further.

#### Start Button

The start button is found on the lower left corner of the interface and by triggering it the taskbar is displayed. Triggering the start button when the taskbar is active will hide the taskbar.



The start button

The start button will not be displayed on any of the login web pages (enhanced or normal).

#### Time and Date Display

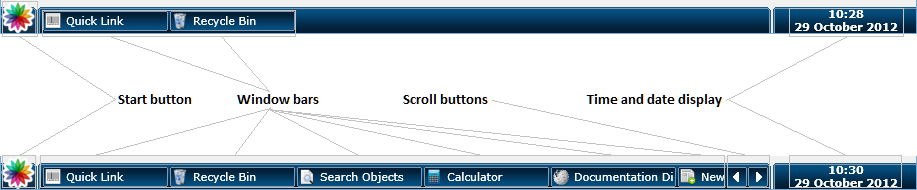
This part of the taskbar is just a display only thing. Clicking on the date or time won’t trigger anything and no windows will be displayed.



Time and date display

#### Window Adjacent Buttons Zone

We will re-describe this part here as shown in the figure bellow. We have the window adjacent buttons displayed from the left to right. If the given space on the taskbar is full the scroll buttons will be displayed as shown in the figure bellow.



Task bar displays

### Java Scripts

All the general scripts used by the platform are stored in the “Scripts” folder and are grouped by categories.

#### Controls Folder

##### CheckBox

This script file is adjacent to the check box control object and helps operate the check box controls displayed on a web page.

###### Methods

**checkboxToggle**

Purpose: triggers the object’s display change and switches its value between 0 and 1.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The check box object’s HTML ID. |

##### ComboBox

This script file is adjacent to the combo box control object and helps operate the check box controls displayed on a web page.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| comboboxActiveObject | String | The HTML ID for the active combo box object. |

###### Methods

**comboboxActivate**

Purpose: activates a combo box object and registers it with the interface.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The combo box object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the combo box object was activated. |

**comboboxDeactivate**

Purpose: deactivates the active combo box object, if any.

Returns: NOTHING

Parameters: NONE

**comboboxSelect**

Purpose: marks a row as selected and saves the corresponding value for it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The combo box object’s HTML ID. |
| Row | Int | The selected row’s number. |

##### Controls

This script is responsible for marking the fields with errors and registers them with the OverDiv class in order to display the error when the mouse cursor is over the control.

###### Methods

**controlsMarkControlWithError**

Purpose: tags the control as containing a bad value by making its border red and registers it to the OverDiv class.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | The control’s HTML ID. |
| Message | String | The error message. |

##### DatePicker

This script is used to operate the date picker controls from a web page.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| datepickerActiveID | String | The HTML ID for the active date picker control. |
| datepickerMonths | Array | The names for the calendar months. |
| datepickerMonthDays | Array | The number of days in each corresponding calendar month. |

NOTE: for February the datepickerMonthDays the array will contain the value of 28 and it will be adjusted to 29 by the datepickerPopulate method when it is called.

###### Methods

**datepickerClose**

Purpose: deactivates the active date picker object, if any.

Returns: NOTHING

Parameters: NONE

**datepickerClear**

Purpose: clears the value from a date picker object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The control’s HTML ID. |

**datepickerShiftMonth**

Purpose: moves the calendar display for the next or previous month.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The control’s HTML ID. |
| Step | Int | Specifies the way in which the calendar moves. The values used for this parameter are -1 and 1. |

**datepickerShiftYear**

Purpose: moves the calendar display for the next or previous year.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The control’s HTML ID. |
| Step | Int | Specifies the way in which the calendar moves. The values used for this parameter are -1 and 1. |

**datepickerDeactivate**

Purpose: deactivates the active date picker control.

Returns: NOTHING

Parameters: NONE

**datepickerActivate**

Purpose: activates a date piker object and populates it with the corresponding dates for the specified month.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The date picker object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the date picker object was activated. |

**datepickerPopulate**

Purpose: populates a date picker control calendar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The date picker object’s HTML ID. |
| Month | Int | The month of the year. |
| Year | Int | The year. |

**datepickerSetDate**

Purpose: sets the selected date as the value for the corresponding field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The date picker object’s HTML ID. |
| Row | Int | The row for the clicked calendar cell. |
| Col | Int | The column for the clicked calendar cell. |

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Controls\_DatePicker.js | Script |
| Scripts/Resources/RO-RO/Controls\_DatePicker.js | Script |

##### FileUpload

This script is adjacent to the FileUpload class and helps operate the file upload controls from the web pages.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| fileuploadActiveID | String | The HTML ID for the active file upload control. |

###### Methods

**fileuploadActivate**

Purpose: activates the triggered file upload control.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The file upload object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the file upload object was activated. |

**fileuploadDeactivate**

Purpose: deactivates the active file upload control and sets the provided file as the control’s value.

Returns: NOTHING

Parameters: NONE

**fileuploadClose**

Purpose: deactivates the active file upload control.

Returns: NOTHING

Parameters: NONE

##### IconSelector

This script is responsible for operating and animating icon selector controls.

###### Methods

**iconselectorOnClick**

Purpose: marks the selected icon and saves the corresponding value to the adjacent field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The icon selector object’s HTML ID. |
| ID | Int | The image ID from the icon selector list. |

**iconselectorScroll**

Purpose: moves the image array to left or right with the specified number of pixels.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The icon selector object’s HTML ID. |
| Step | Int | The pixels which trigger the move. The used values for this parameters are at the moment -100 and 100. |

##### ImageUpload

This script is adjacent to the ImageUpload class and helps operate the image upload controls from the web pages.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| imageuploadActiveID | String | The HTML ID for the active image upload control. |

###### Methods

**imageuploadActivate**

Purpose: activates the triggered image upload control.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The image upload object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the image upload object was activated. |

**imageuploadDeactivate**

Purpose: deactivates the active image upload control and sets the provided image as the control’s value.

Returns: NOTHING

Parameters: NONE

**imageuploadClose**

Purpose: deactivates the active image upload control.

Returns: NOTHING

Parameters: NONE

##### Integer

This script is adjacent to the Integer class and limits the user to selecting an integer value within a given range.

###### Methods

**integerChange**

Purpose: changes the value for the control and saves that value to the adjacent field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | The integer object’s HTML ID. |
| Unit | Int | The value that triggers the slide, negative or positive. |

**integerParse**

Purpose: parses the integer value from the field and validates it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | The integer object’s HTML ID. |

##### MultiCheck

This script is adjacent to the multi check control and provides the necessary methods to accommodate its behaviour on web pages.

###### Methods

**multicheckToggle**

Purpose: triggers the toggle in the multi check element’s state and saves the value in the adjacent control field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The multi check object’s HTML ID. |
| OptionID | Int | The option’s number within the multi check control. |

##### Password

This script is adjacent to the password controls and completes the behaviour of password controls on web pages.

###### Methods

**passwordRegister**

Purpose: registers a textbox control as a password control.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | String | The password object’s HTML ID. |

**passwordOnKeyPress**

Purpose: triggers a password check when the password changes.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | String | The password object’s HTML ID. |

**passwordCheck**

Purpose: checks the strength of the password and updates the password’s field display.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | String | The password object’s HTML ID. |

##### RadioGroup

This script is adjacent to the radio group control and provides the necessary methods to accommodate its behaviour on web pages.

###### Methods

**radiogroupToggle**

Purpose: triggers the toggle in the radio group element’s state and saves the value in the adjacent control field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The radio group object’s HTML ID. |
| OptionID | Int | The option’s number within the radio group control. |

##### Rating

This script is adjacent to the rating controls and helps their behaviour on web pages.

###### Methods

**ratingOnClick**

Purpose: changes the control’s display state according to the sent value.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | The rating object’s HTML ID. |
| Value | Int | The value that must be represented by the rating control. |

**ratingOnMouseOver**

Purpose: animates the control when the mouse pointer is over the control’s space.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | The rating object’s HTML ID. |
| Value | Int | The value that must be represented by the rating control. |

**ratingOnMouseOut**

Purpose: stops animating the control when the mouse pointer leaves the control area.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| HTMLObjectID | String | The rating object’s HTML ID. |

##### Slider

This script is adjacent to the slider control and helps with their behaviour on web pages.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| sliderActiveObjectID | String | The HTML ID for the active slider control. |

###### Methods

**sliderSlide**

Purpose: changes the value for the slider control and modifies the control’s display accordingly.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The slider object’s HTML ID. |

**sliderSetValue**

Purpose: sets a value for the control and displays it in the adjacent field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The slider object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the slider object was activated. |

**sliderActivate**

Purpose: activates the triggered slider control.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The slider object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the slider object was activated. |

**sliderDeactivate**

Purpose: deactivates the currently active slider control.

Returns: NOTHING

Parameters: NONE

**sliderClear**

Purpose: clears the value from the adjacent control field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The slider object’s HTML ID. |

**sliderClose**

Purpose: deactivates the currently active slider control.

Returns: NOTHING

Parameters: NONE

##### TimePicker

This script is adjacent to the time picker controls and helps with their behaviour on web pages.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| timepickerActiveID | String | The HTML ID for the active time picker control. |

###### Methods

**timepickerClose**

Purpose: deactivates the active time picker object.

Returns: NOTHING

Parameters: NONE

**timepickerClear**

Purpose: clears the value from the time picker’s adjacent field.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The time picker object’s HTML ID. |

**timepickerShiftMinute**

Purpose: shifts the minute for the time picker backward or forward.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The time picker object’s HTML ID. |
| Step | Int | The minute shift backward or forward. The value for this parameter is either 1 or -1. |

**timepickerShiftHour**

Purpose: shifts the hour for the time picker backward or forward.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The time picker object’s HTML ID. |
| Step | Int | The hour shift backward or forward. The value for this parameter is either 1 or -1. |

**timepickerDeactivate**

Purpose: deactivates the active time picker object.

Returns: NOTHING

Parameters: NONE

**timepickerActivate**

Purpose: activates the triggered time picker control.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The time picker object’s HTML ID. |
| e | Event | The event object that encapsulates the data from when the time picker object was activated. |

#### Core Folder

##### Animation

This script is used by other objects in order to animate some of the elements of the interface. The best example for this is the animation that happens when a window is maximized.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| animationDivider | Int | In order to increase the size of an object, the difference between the actual size and final size is divided with this number and the object chances its size. |
| animationInterval | Int | The interval in milliseconds for the next step of the animation. |
| animationSureStep | Int | When the difference of size between the actual size and final size of the object is smaller that this number the object will snap to the final size. |
| animationMovingObjects | String | Retains the HTML IDs of the currently animated objects for some animations. |

###### Methods

**animationMove**

Purpose: triggers a moving animation for an object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The time picker object’s HTML ID. |
| PosX | Int | The destination’s horizontal position in pixels. |
| PosY | Int | The destination’s vertical position in pixels. |

**animationMoveStep**

Purpose: moves the given object closer to the destination point.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The time picker object’s HTML ID. |
| PosX | Int | The destination’s horizontal position in pixels. |
| PosY | Int | The destination’s vertical position in pixels. |

##### Compat

The role for this script is to mediate cross browser incompatibilities. In order to avoid inline decisions regarding the browser and version we decided to have function calls for each non-standard element.

###### Methods

**compatDocumentHeight**

Purpose: retrieves the current height of the web page’s container.

Returns: Int (representing the container’s height in pixels).

Parameters: NONE

**compatDocumentWidth**

Purpose: retrieves the current width of the web page’s container.

Returns: Int (representing the container’s width in pixels).

Parameters: NONE

**compatMouseButton**

Purpose: retrieves the code for the mouse button pressed.

Returns: Int (representing the pressed button code)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**compatOffsetX**

Purpose: retrieves the mouse pointer’s horizontal coordinate in pixels relative to the upper left corner of the HTML element.

Returns: Int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**compatOffsetY**

Purpose: retrieves the mouse pointer’s vertical coordinate in pixels relative to the upper left corner of the HTML element.

Returns: Int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**compatGetPageX**

Purpose: retrieves the mouse pointer’s horizontal position relative to the document upper left corner.

Returns: Int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**compatGetPageY**

Purpose: retrieves the mouse pointer’s vertical position relative to the document upper left corner.

Returns: Int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**compatGetClientX**

Purpose: retrieves the mouse pointer’s horizontal position relative to the document upper left corner.

Returns: Int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**compatGetClientY**

Purpose: retrieves the mouse pointer’s vertical position relative to the document upper left corner.

Returns: Int

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

##### Error

This script is used to display error boxes on a web page.

###### Methods

**errorDisplayError**

Purpose: displays a modal box containing all the specified error messages in it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Errors | String | The error messages separated by the “|” character. |

##### Mouse

This script mediates all the drag events for the interface.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| mousePosX | Int | Keeps the last horizontal mouse position before the current event. |
| mousePosY | Int | Keeps the last vertical mouse position before the current event. |
| mouseActiveEvent | String | Keeps the active mouse even (such as drag or resize). |
| mouseActiveObject | Int | Keeps the affected window’s ID. |
| mouseActiveObjectOffsetX | Int | Keeps the original horizontal offset of the mouse pointer inside the active object. |
| mouseActiveObjectOffsetY | Int | Keeps the original vertical offset of the mouse pointer inside the active object. |

###### Methods

**mouseOnMouseMove**

Purpose: triggers the dispatch event for the current mouse event when the mouse cursor is moved.

Returns: bool (specifying that the event must not propagate any further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**mouseDispatchEvent**

Purpose: dispatches the current marked event.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**mouseOnMouseUp**

Purpose: marks the end of the current registered event.

Returns: NOTHING

Parameters: NONE

**mouseInitialize**

Purpose: generates the dummy div element used as canvas for drag and resize events.

Returns: NOTHING

Parameters: NONE

**mouseRegisterWindow**

Purpose: register’s a window with the specified event and dispatches any mouse move event to that window.

Returns: bool (specifying that the event must not propagate any further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |
| Event | String | The event’s name. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**mouseUnregisterWindow**

Purpose: unregisters the current window and closes the current active event.

Returns: NOTHING

Parameters: NONE

**mouseRegisterIcon**

Purpose: register’s an icon with the specified event and dispatches any mouse move event to that icon.

Returns: bool (specifying that the event must not propagate any further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The icon ID composing the icon’s HTML ID. |
| Event | String | The event’s name. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**mouseUnregisterIcon**

Purpose: unregisters the current icon and closes the current active event.

Returns: NOTHING

Parameters: NONE

**mouseDragWindow**

Purpose: treats the drag event for the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseDragIcon**

Purpose: treats the drag event for the given icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The icon ID composing the icon’s HTML ID. |

**mouseResizeWindowN**

Purpose: treats the resize event for the top margin of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowE**

Purpose: treats the resize event for the right margin of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowS**

Purpose: treats the resize event for the bottom margin of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowW**

Purpose: treats the resize event for the left margin of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowNE**

Purpose: treats the resize event for the right upper corner of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowSE**

Purpose: treats the resize event for the right lower corner of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowNW**

Purpose: treats the resize event for the left upper corner of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

**mouseResizeWindowSW**

Purpose: treats the resize event for the left lower corner of the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The window ID composing the window’s HTML ID. |

##### Utility

This script is used by the web pages displayed in windows.

###### Methods

**utilityStringToBase16String**

Purpose: converts the given string into a hex encoded string based on characters ASCII codes.

Returns: string (representing the given string’s encoding

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The string that must be encoded. |

**utilityActivate**

Purpose: activates and brings to front the window that called the method.

Returns: NOTHING

Parameters: NONE

**utilitySetLoadingStatus**

Purpose: sets the status to “Loading” for the window that called the method.

Returns: NOTHING

Parameters: NONE

**utilitySetStatus**

Purpose: sets the window status that called the method to the given HTML string.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| StatusHTML | String | The new status for the window. |

**utilityMakeFixedBorder**

Purpose: modifies the window that called the method have fixed borders.

Returns: NOTHING

Parameters: NONE

**utilityBalloon**

Purpose: displays a balloon tool tip for the taskbar button associated with the window that called the method.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The tool tip text. |
| Icon | String | The relative HTTP path for the icon displayed in the tool tip balloon. |

**utilitySetIcon**

Purpose: sets the given icon as representing icon for the window that called the method.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconPath | String | The relative HTTP path for the icon. |

**utilityMove**

Purpose: moves the window that called the method to the specified coordinates.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| PosX | Int | The horizontal position in pixels. |
| PosY | Int | The vertical position in pixels. |

**utilityResize**

Purpose: resizes the window that called the method to the specified dimensions.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Width | Int | The new width in pixels. |
| Height | Int | The new height in pixels. |

**utilityClose**

Purpose: triggers the window that called the method to close itself.

Returns: NOTHING

Parameters: NONE

#### Objects Folder

##### Balloon

This is a stand-alone java script designed to be used in order to display balloon tool tips for the task bar buttons associated to the windows displayed on the content section.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| balloonCounter | Int | The counter that retains the last ID given to a balloon object. |

###### Methods

**balloonDisplayForWindow**

Purpose: displays a new balloon tool tip for the given window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID for the window. |
| Text | String | The text that must be displayed for the balloon. |
| Icon | String | The HTTP relative path for the pictogram that must be displayed inside the balloon. |

**balloonClose**

Purpose: closes the specified balloon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| BalloonID | Int | The ID of the balloon. |

**balloonFade**

Purpose: triggers the specified balloon to fade out until 100% transparency at which point it is closed.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| BalloonID | Int | The ID of the balloon. |

**balloonFreeze**

Purpose: causes the given balloon to stop fading.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| BalloonID | Int | The ID of the balloon. |

**balloonUnfreeze**

Purpose: restarts the fading effect for the given balloon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| BalloonID | Int | The ID of the balloon. |

**baloonDisplay**

Purpose: displays a balloon object at the giving coordinates.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| PosX | Int | The horizontal position in pixels for the balloon. |
| PosY | Int | The vertical position in pixels for the balloon. |
| Text | String | The text that must be displayed for the balloon. |
| Icon | String | The HTTP relative path for the pictogram that must be displayed inside the balloon. |

##### ContextMenu

This is a stand-alone java script used to display context menus for registered objects.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| contextmenuCount | Int | The counter that retains the last ID given to a context menu object. |
| contextmenuActiveID | Int | The ID for the active context menu. |

###### Methods

**contextmenuInitialize**

Purpose: initializes the context menu elements that aid in interacting with context menu objects.

Returns: NOTHING

Parameters: NONE

**contextmenuSetZIndex**

Purpose: sets the z-index for a context menu object in order to display it above other objects.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| ZIndex | Int | The z-index that needs to be set. |

**contextmenuReset**

Purpose: resets the context menu objects by hiding any active context menu.

Returns: NOTHING

Parameters: NONE

**contextmenuHide**

Purpose: hides the specified context menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |

**contextmenuShow**

Purpose: displays the specified context menu according to the mouse event specified.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**contextmenuNewContextMenu**

Purpose: creates a new context menu object.

Returns: int (representing the new context menu object ID)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Width | Int | The width in pixels for the context menu object. |

**contextmenuAddMenuParam**

Purpose: adds a new parameter for the specified menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| ParamName | String | The name of the context menu parameter. |

**contextmenuSetParam**

Purpose: sets the value for the specified context menu parameter.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| ParamName | String | The name of the context menu parameter. |
| ParamValue | String | The parameter’s value. |

**contextmenuCommandOnClick**

Purpose: triggers the corresponding command from the given context menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| CommandID | Int | The ID for the context menu’s command. |

**contextmenuCommandOnMouseOver**

Purpose: activates the command when the mouse pointer enters its space.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| CommandID | Int | The ID for the context menu’s command. |

**contextmenuCommandOnMouseOut**

Purpose: deactivates the command when the mouse pointer leaves its space.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| CommandID | Int | The ID for the context menu’s command. |

**contextmenuAddCommand**

Purpose: adds a new command to the specified context menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |
| Title | String | The title for the command. |
| Icon | String | The relative HTTP path of the pictogram associated with the command. |
| OnClickEvent | String | A java scrip sequence that must be run when the command is triggered. |

**contextmenuAddSpace**

Purpose: adds a command separator for the specified context menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ContextMenuID | Int | The ID for the context menu object. |

##### Icon

This is a stand-alone java script needed for the icon elements interaction and display and it is specific to this window interface.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| iconCount | Int | The counter that retains the last ID given to an icon object. |
| iconActiveID | Int | The ID for the active/selected icon object. |
| iconList | String | The list of icon IDs displayed on the interface separated by the “|” character. |
| iconContextMenuID | Int | The context menu ID for the context menu object associated with the icons displayed. |

###### Methods

**iconInitialize**

Purpose: initializes the icon adjacent HTML objects used in background by this module.

Returns: NOTHING

Parameters: NONE

**iconShowContextMenu**

Purpose: displayes the context menu associated to the given icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**iconUpdateIcon**

Purpose: opens the web page that can alter the specified icon in an independed window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |
| DBIconID | Int | The primary key value for the associated entry in the DesktopIcon table. |

**iconChangeIcon**

Purpose: alters an icon displayed on the contect section on the interface.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |
| Title | String | The title for the icon. |
| Icon | String | The relative HTTP path of the pictogram associated with the icon. |
| Link | String | The HTTP path for the web page that must be opened when the icon is triggered. |

**iconUnregisterIcon**

Purpose: triggers the specified icon’s removal from the content section of the interface.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |

**iconMove**

Purpose: moves the specified icon on the content section to the specified position.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |
| PosX | Int | The horizontal position in pixels. |
| PosY | Int | The vertical position in pixels. |

**iconCheckPositions**

Purpose: in case of a browser window resize this method checks the positions of all the icons displayed in the content section and repositions them if they fall out of the display area.

Returns: NOTHING

Parameters: NONE

**iconOnDrag**

Purpose: marks the specified icon as being dragged and registeres it with the mouse script.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**iconOnRightClick**

Purpose: opens the context menu associated with the icon.

Returns: bool (the value returned is false in order to stop the right click event to be propagated further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**iconNewIcon**

Purpose: creates a new icon and places it on the content section.

Returns: int (representing the new icon’s ID)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title for the icon. |
| Icon | String | The HTTP relative path for the pictogram associated with the icon. |
| Link | String | The HTTP path for the web page that must be opened when the icon is triggered. |
| PosX | Int | The horizontal position in pixels. |
| PosY | Int | The vertical position in pixels. |

**iconDeactivateActiveIcon**

Purpose: deactivates the currently active icon.

Returns: NOTHING

Parameters: NONE

**iconOnDblClick**

Purpose: triggers the given icon to open a new window with the destined web link.

Returns: bool (marking the event as threated in order to stop its propagation)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |

**iconRemoveIcon**

Purpose: removes the specified icon from the content section.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |

**iconOnClick**

Purpose: activates the given icon and deaectivates the current active icon (if any).

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The ID for the icon object. |

##### Menu

This script is adjacent to the menu class and helps the menu items animate on mouse actions.

###### Methods

**menuOnMouseOver**

Purpose: highlights the specified menu option when the mouse pointer is over it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | Int | The ID for the menu option object. |

**menuOnMouseOut**

Purpose: dehighlight the specified menu option when the mouse pointer leaves its area.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| id | Int | The ID for the menu option object. |

##### MenuBar

This script is adjacent to the menu bar class and is used to animate the menu bars displayed on web pages.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| menubarActiveID | Int | The ID for the active menu bar sub menu. |
| menubarActiveState | String | The state for the active menu bar. |

###### Methods

**menubarInitialize**

Purpose: initializes the window’s surface in order to close the opened submenues when the mouse pointer is on its surface.

Returns: NOTHING

Parameters: NONE

**menubarInactivateMenu**

Purpose: deactivates the active menu (if any).

Returns: NOTHING

Parameters: NONE

**menubarMenuOnMouseOver**

Purpose: activates the specified menu from the menu bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The ID for the menu object. |

**menubarMenuOnClick**

Purpose: details the commands for the specified menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The ID for the menu object. |

**menubarCommandOnMouseOver**

Purpose: highlights the specified command when the mouse pointer enters its space.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The ID for the menu object. |
| CommandID | Int | The ID for the command object. |

**menubarCommandOnMouseOut**

Purpose: unhighlights the specified menu command when the mouse pointer leaves its space.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The ID for the menu object. |
| CommandID | Int | The ID for the command object. |

**menubarCommandOnClick**

Purpose: triggers the specified java script sequence for the menu command.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| MenuID | Int | The ID for the menu object. |
| CommandID | Int | The ID for the command object. |

##### ModalBox

This is a stand-alone script used to display modal boxes on windows and on the content section.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| modalboxList | String | The list of modal boxe IDs displayed separated by the “|” character. |
| modalboxCount | Int | The first available ID for the next modal box to be display. |

###### Methods

**modalboxCheckBoxes**

Purpose: checks the modal boxes positions in case of a web page resize.

Returns: NOTHING

Parameters: NONE

**modalboxRepositionBox**

Purpose: repositions the specified modal box in the center of the web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | Int | The ID for the modal box object. |

**modalboxNewBox**

Purpose: creates a new modal box object and displays it on the center of the web page.

Returns: int (representing the ID of the new modal box object)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title of the modal box. |
| Icon | String | The relative HTTP path for the pictogram associated with the modal box. |
| Message | String | The message displayed by the modal box. |
| Width | Int | The width in pixels of the modal box object. |
| Height | Int | The height in pixels of the modal box object. |
| Buttons | String | A HTML script describing the buttons that must be displayed by the modal box object. |

**modalboxClose**

Purpose: removes a modal box object from the web page display.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ModalBoxID | Int | The ID for the modal box object. |

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Objects\_ModalBox.js | Script |
| Scripts/Resources/RO-RO/Objects\_ModalBox.js | Script |

##### Notification

This script is used to display new system notifications on the content section and animate them.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| notificationCount | Int | The first available ID for the next notification panel to be displayed. |
| notificationTop | Int | The top restriction in pixels for the notification panels. |
| notificationList | String | The list of IDs for all the active notification panels separated by the “|” character. |

###### Methods

**notificationNew**

Purpose: generates a new notification panel and displays it on the content section.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The category of the notification. |
| Icon | String | The relative HTTP path of the pictogram associated with the notification category. |
| Text | String | The text for the notification. |

**notificationOnMouseOut**

Purpose: restarts the specified notification’s fade process.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| NotificationID | Int | The ID of the notification panel. |

**notificationOnMouseOver**

Purpose: stops the specified notification panel fading process and restores it to full opacity.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| NotificationID | Int | The ID of the notification panel. |

**notificationFade**

Purpose: fades the given notification panel until it is completely transparent and then removes it from the content section.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| NotificationID | Int | The ID of the notification panel. |

##### OverDiv

This script is adjacent to the over div class and it it used to display tool tip messages for the registered objects.

###### Methods

**overdivRegisterObject**

Purpose: registers the specified HTML object with the over div object to display the specified tool tip for it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ObjectID | String | The HTML ID for the object that must be registered. |
| Description | String | The tool tip text in HTML form. |

**overdivOnMouseOver**

Purpose: displays the destined tool tip for the given HTML object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |
| Description | String | The tool tip text in HTML form. |

**overdivOnMouseOut**

Purpose: hides the object’s tool tip when the mouse pointer leaves its space.

Returns: NOTHING

Parameters: NONE

**overdivOnMouseMove**

Purpose: moves the tool tip window to follow the mouse pointer if the mouse pointer is still on the registered HTML object’s surface.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

##### Page

This script is adjacent to the page class and it is used to mediate some backgroung tasks and mechanisms of interaction with the user.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| pageContextMenuID | Int | Holds the ID for the context menu associated with the content section. |
| pageWindowCallBack | String | Holds a java script code sequence that must be runned when a displayed object is deactivated. |

###### Methods

**pageSetWindowTitle**

Purpose: sets the window title according to the HTML title from the displayed page.

Returns: NOTHING

Parameters: NONE

**pageWindowInitialize**

Purpose: initializes the objects needed for a window web page interaction.

Returns: NOTHING

Parameters: NONE

**pageObjectActivate**

Purpose: marks the specified HTML object as being active and specifies the callback code for it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CallBack | String | The code sequence that is triggered when the active object must be deactivated. |

**pageObjectDeactivate**

Purpose: runs the callback code specified by the pageObjectActivate method.

Returns: NOTHING

Parameters: NONE

**pageInitialize**

Purpose: initializes the objects needed for the main web page.

Returns: NOTHING

Parameters: NONE

**pageOnRightClick**

Purpose: displays the context menu for the main web page.

Returns: bool (returns the false value just to stop the right click event from propagating)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**pageShowContextMenu**

Purpose: displays the context menu for the main web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**pagePersonalize**

Purpose: opens a window that displayes the web page needed to personalize the display.

Returns: NOTHING

Parameters: NONE

**pageAddNewIcon**

Purpose: opens a window that displayes the web page needed to add a new icon on the content section.

Returns: NOTHING

Parameters: NONE

**pageOpenUserData**

Purpose: opens a window that displayes the web page needed to modify the current logged on user data.

Returns: NOTHING

Parameters: NONE

**pageHideTaskBar**

Purpose: closes the taskbar.

Returns: NOTHING

Parameters: NONE

**pageTaskbarDeactivate7**

Purpose: closes the Windows 7 style taskbar.

Returns: NOTHING

Parameters: NONE

**pageTaskbarDeactivate8**

Purpose: closes the Windows 8 style taskbar.

Returns: NOTHING

Parameters: NONE

**pageTaskbarOnClick7**

Purpose: opens the task bar if it is set for the Windows 7 display style.

Returns: NOTHING

Parameters: NONE

**pageTaskbarOnClick8**

Purpose: opens the task bar if it is set for the Windows 8 display style.

Returns: NOTHING

Parameters: NONE

**pageTime**

Purpose: updates the time and date displayed on the left lower corner.

Returns: NOTHING

Parameters: NONE

**pageOnResize**

Purpose: checks the page dimension and objects displayed when the display container is resized.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Objects\_Page.js | Script |
| Scripts/Resources/RO-RO/Objects\_Page.js | Script |

##### SidePanel

This script is adjacent to the side panel class and its purpose is to help with the toggling of panels.

###### Methods

**sidepanelOnClick**

Purpose: activates the specified panel of the specified tab panel object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| TabID | Int | The ID of the side panel object. |
| PanelID | Int | The ID of the contained panel object. |

##### TabPanel

This script is adjacent to the tab panel class and its purpose is to help with the toggling of panels.

###### Methods

**tabpanelOnClick**

Purpose: activates the specified panel of the specified tab panel object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| TabID | Int | The ID of the side panel object. |
| PanelID | Int | The ID of the contained panel object. |

##### TaskBar

This scriptis used to animate the task bar buttons associated with thw windows displayed on the content section.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| taskbarContextMenuID | Int | Holds the ID for the context menu associated with the windows’ adjacent buttons. |
| taskbarProportion | Int | The shrinkage percentage for the preview windows displayed. |

###### Methods

**taskbarReduceSize**

Purpose: processes the preview HTML code for the specified HTML element and children elements.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | HTML element | The HTML element that must be processed. |

**taskbarOnMouseOver**

Purpose: processes the preview form and displays in using the OverDiv object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**taskbarOnMouseOut**

Purpose: hides the OverDiv object displayed for a task bar button.

Returns: NOTHING

Parameters: NONE

**taskbarOnMouseMove**

Purpose: triggers the move of a displayed OverDiv object.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**taskbarInitialize**

Purpose: initializes the HTML element needed by the task bar section.

Returns: NOTHING

Parameters: NONE

**taskbarShowContextMenu**

Purpose: displays the context menu associated with the buttons displayed on the task bar.

Returns: Boolean (returns “false” to stop the event from propagating further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**taskbarScroll**

Purpose: scrolles the buttons displayed to left or right according to the specified step value.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Step | Int | The step value in pixels for the button list move animation. Negative values means scroll left and positive values means scroll right. |

**taskbarUnregisterWindow**

Purpose: removes the adjacent button for the specified window from the task bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |

**taskbarChangeButtonTitle**

Purpose: changes the task bar titled displayed on the button adjacent to the specified window with the new given title.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |
| Title | String | The new title for the window’s adjacent button. |

**taskbarRegisterWindow**

Purpose: creates a new task bar button for the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |

**taskbarCheckSize**

Purpose: resizes the taskbar space and recalibrates it according to the browser’s display surface.

Returns: NOTHING

Parameters: NONE

**taskbarToggleWindow**

Purpose: minimizes the specified window if it is active or activates it if it is inactive or minimized.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Objects\_TaskBar.js | Script |
| Scripts/Resources/RO-RO/Objects\_TaskBar.js | Script |

##### TreeMenu

This script is needed to animate the taskbar and taskbar menu for both Window 7 and Windows 8 styles.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| treemenuContextMenuID | Int | Holds the ID for the context menu associated with the taskbar. |

###### Methods

**treemenuInitialize**

Purpose: initializes the needed HTML objects needed for the taskbar.

Returns: NOTHING

Parameters: NONE

**treemenuSendToDesktop**

Purpose: creates a desktop icon from the specified menu element.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command. |

**treemenuContextClick7**

Purpose: displays the context menu associated with the Windows 7 style taskbar.

Returns: Boolean (returns “false” in order to stop the event from propaganing further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**treemenuContextClick8**

Purpose: displays the context menu associated with the Windows 8 style taskbar.

Returns: Boolean (returns “false” in order to stop the event from propaganing further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**treemenuOnMouseOver7**

Purpose: highlights the specified menu item for the Windows 7 taskbar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command or menu. |

**treemenuOnMouseOver8**

Purpose: highlights the specified menu item for the Windows 8 taskbar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command or menu. |

**treemenuOnMouseOut7**

Purpose: dehighlights the specified menu item for the Windows 7 taskbar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command or menu. |

**treemenuOnMouseOut8**

Purpose: dehighlights the specified menu item for the Windows 8 taskbar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command or menu. |

**treemenuMenuOnClick**

Purpose: collapses or expands the specified menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu. |

**treemenuCommandOnClick**

Purpose: opens a new window on the content section with the title and icon of the specified menu item and loads the web page associated with the specified menu item.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu. |

**treemenuReset**

Purpose: collapses all the tree menu elements for a Window 7 style taskbar.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Objects\_TreeMenu.js | Script |
| Scripts/Resources/RO-RO/Objects\_TreeMenu.js | Script |

##### Window

This script is a stand-alone script that is used to create window elements on the content section.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| windowCount | Int | Retains the first available window element ID. |
| windowActiveID | Int | The ID of the current active window element. |
| windowList | String | Holds all the displayed windows’ IDs separated by the “|” character. |
| windowContextID | Int | Holds the ID of the context menu object used by all the window elements. |
| windowLastzIndex | Int | Holds the last z-index given to a window element. The starter point for this is 100. |

###### Methods

**windowSetStatus**

Purpose: sets the status for the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| StatusHTML | String | The status that must be displayed in the window’s status bar. |

**windowCheckDuplicates**

Purpose: checks to see if there is a window on the content section with the given link loaded.

Returns: Int (representing the ID of the window (if any) else it will return the value “-1”)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Link | String | The http link. |

**windowExists**

Purpose: checks if there is a window with the specified ID.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowSendToDesktop**

Purpose: creates a new icon that will open a window with an identical link, icon and title with the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowInitialize**

Purpose: initializes the needed HTML object that cove the window element’s functionalities.

Returns: NOTHING

Parameters: NONE

**windowShowContextMenu**

Purpose: displays the context menu for the specified window

Returns: Boolean (returns false to stop the event from propagating further)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**windowSetContent**

Purpose: changes web page loaded by the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| Link | String | The http path for the new web page that must be loaded by the window. |

**windowDeactivate**

Purpose: deactivates the specified window if it is the active window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowActivate**

Purpose: activates the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowMinimize**

Purpose: minimizes the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowMaximize**

Purpose: maximizes the specified window if it’s not or restores it to the original size if it is maximized.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowClose**

Purpose: removes the specified window from the content section and from the task bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowUnregister**

Purpose: removes the specified window from the windows list kept and from the task bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowRegister**

Purpose: adds the specified window to the windows list kept and to the task bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowCheckPosition**

Purpose: verifies the positions and dimensions for all the windows displayed when the braowser is resized in order to keep the windows accessible to the user.

Returns: NOTHING

Parameters: NONE

**windowNewWindow**

Purpose: creates a new window on the content section.

Returns: Int (representing the new window’s ID)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The window’s title. |
| Icon | String | The relative http path for the pictogram associated with the window. |
| Link | String | The http link for the web page that will be loaded by the window. |
| PosX | Int | The horizontal position in pixels of the upper left corner of the window. |
| PosY | Int | The vertical position in pixels of the upper left corner of the window. |
| Width | Int | The width of the window in pixels. |
| Height | Int | The height of the window in pixels. |

**windowSetIcon**

Purpose: changes the pictogram for the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| IconPath | String | The relative http path of the new pictogram. |

**windowAddWindow**

Purpose: adds a window on the content section that is already saved in the database.

Returns: Int (representing the new window’s ID)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The window’s title. |
| Icon | String | The relative http path for the pictogram associated with the window. |
| Link | String | The http link for the web page that will be loaded by the window. |
| PosX | Int | The horizontal position in pixels of the upper left corner of the window. |
| PosY | Int | The vertical position in pixels of the upper left corner of the window. |
| Width | Int | The width of the window in pixels. |
| Height | Int | The height of the window in pixels. |
| ID | Int | The window primary key value from the DesktopWindow table. |

**windowIsJustOpened**

Purpose: checks in the specified window is newly opened or if it loaded more than one page.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowCancelJustOpen**

Purpose: marks the specified window as opened and finished loaded the first web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowChangeTitle**

Purpose: changes the title of the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| Title | String | The new title for the window. |

**windowContainerOnLoad**

Purpose: called when the window finishes loading a web page in order to register the changes with the database layer.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowReload**

Purpose: triggers the window to reload the displayed web page.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowOnResize**

Purpose: registers the specified window for a resize event.

Returns: Boolean (in order to propagate the event further or stop it)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| Direction | String | The direction for the resize event. Possible values are: N, NE, E, SE, S, SV, V and NV. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**windowOnDrag**

Purpose: registers the specified window for a drag event.

Returns: Boolean (in order to propagate the event further or stop it)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| e | Event | The event object that encapsulates the data from when the inquiry was lunched. |

**windowMove**

Purpose: moves the specified window to the specified position.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| PosX | Int | The new horizontal position in pixels of the upper left corner of the window. |
| PosY | Int | The new vertical position in pixels of the upper left corner of the window. |

**windowResize**

Purpose: resizes the specified window to the specified dimensions.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| Width | Int | The new width in pixels. |
| Height | Int | The new height in pixels. |

**windowIsActive**

Purpose: checks if the specified window is the active window.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowIsMinimized**

Purpose: checks if the specified window is minimized.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowIsMaximized**

Purpose: checks if the specified window is maximized.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Images/Objects/Window/btClose.png | Image |
| Images/Objects/Window/btMaximize1.png | Image |
| Images/Objects/Window/btMaximize2.png | Image |
| Images/Objects/Window/btMinimize.png | Image |
| Images/Objects/Window/Component\_1\_1.png | Image |
| Images/Objects/Window/Component\_1\_2.png | Image |
| Images/Objects/Window/Component\_1\_3.png | Image |
| Images/Objects/Window/Component\_2\_1.png | Image |
| Images/Objects/Window/Component\_2\_2.png | Image |
| Images/Objects/Window/Component\_3\_1.png | Image |
| Images/Objects/Window/Component\_3\_2.png | Image |
| Images/Objects/Window/Component\_3\_3.png | Image |
| Images/Objects/Window/Loading.gif | Image |
| Scripts/Resources/EN-UK/Objects\_Window.js | Script |
| Scripts/Resources/RO-RO/Objects\_Window.js | Script |

#### Resources Folder

This folder holds resource scripts used by controls, objects, web pages and widgets in order to cover the language issues. The resources scripts are loaded on web pages as needed and the variables contained in them are used in java scripts to display texts in the interface’s language.

The subfolders contained here are named according to the language codes for the languages supported by the interface. For a script file found in the EN-UK folder there should be files named identical in all the other language folders.

NOTE: If the interface is in a language for which a resource script is missing it will include the EN-UK script in its place.

At this point the interface has resource scripts for English (EN-UK) and Romanian (RO-RO). A new language can be added by using the adjacent translator application or manualy (adding resource files, resource scripts and database language tags).

#### Tools Folder

##### Mapping

This script is used by the Mapping class from the Amaryllis.Tools namespace in order to provide functionality for the object listings destined for mapping operations.

###### Methods

**mappingSelect**

Purpose: marks the selected row/object for the mapping operation.

Returns: NOTING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier for the listing table. |
| Item | Int | The selected row’s number in the listing. |
| RowsPerItem | Int | The number of rows that represent a single item. |

##### Object

This script is used by the Object class from the Amaryllis.Tool namespace in order to provide functionality for the object listings.

###### Methods

**objectRestore**

Purpose: triggers the restore of the specified object according to the latest log entry with all the ascending objects. Descending objects are restored if specified.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ID | Int | The primary key value for the oject. |
| WithChildren | Int | Specifies if the object will be restored with all its children object. |

**objectRemoveMapping**

Purpose: removes the mapping filter for the specified listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectSetMapping**

Purpose: sets a mapping filter for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectToggleSelectPage**

Purpose: triggers a page change according to the page selected from the drop-down box.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectTogglePage**

Purpose: triggers a page change according to the predefined page boxes displayed at the bottom of the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |
| Page | Int | The page number. |

**objectRemoveFilter**

Purpose: removes the filters for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectSetFilter**

Purpose: sets the filters for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectSetSorting**

Purpose: sets the sorting criteria for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |
| Value | String | The sort criteria composed from the column name and order (ASC or DESC). |

**objectToggleMapping**

Purpose: toggles the mapping header for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectToggleSort**

Purpose: toggles the sorting header for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

**objectToggleFilter**

Purpose: toggles the filtering header for the specified object listing.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Identifier | String | The identifier of the object listing. |

##### Report

This script is used by the Report class from the Amaryllis.Tool namespace in order to provide functionality for the report display.

###### Methods

**reportToggleSelectPage**

Purpose: changes the report page displayed by the report view according to the drop-down box.

Returns: NOTHING

Parameters: NONE

**reportTogglePage**

Purpose: changes the report page displayed by the report view according to the specified parameter.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Page | Int | The page number. |

**reportRefresh**

Purpose: triggers a report display refresh.

Returns: NOTHING

Parameters: NONE

#### WebServices Folder

The scripts contained in this folder are used to call web services. All of the java scripts here are used by the default.aspx web page to send windows and icons data to the database layer and also retrieve users notifications from the database.

##### Icon

This script is responsible for sending icon data from the database layer in case new icons appear on the desktop or old icons are moved.

###### Methods

**wsiconRegisterIcon**

Purpose: saves a new icon’s information in the database.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The HTML ID for the new icon. |
| Title | String | The title for the icon. |
| Icon | String | The http relative path for the pictogram associated with the icon. |
| Link | String | The http path for the web page associated with the icon. |
| PosX | Int | The horizontal position in pixels for the icon. |
| PosY | Int | The vertical position in pixels for the icon. |

**wsiconRegisterIcon\_CallBack**

Purpose: call-back for the wsiconRegisterIcon to set the database ID for the icon.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Result | String | The web service returned result representing the HTML ID and DB ID separated by the “|” character. |

**wsiconChangeIcon**

Purpose: updates the specified icon data in the database.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The HTML ID for the new icon. |
| Title | String | The title for the icon. |
| Icon | String | The http relative path for the pictogram associated with the icon. |
| Link | String | The http path for the web page associated with the icon. |
| PosX | Int | The horizontal position in pixels for the icon. |
| PosY | Int | The vertical position in pixels for the icon. |

**wsiconUnregisterIcon**

Purpose: deletes the specified icon from the database.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| IconID | Int | The HTML ID for the icon. |

##### Notification

This script is used to retrieve the latest notifications for the currently logged on user.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| wsnotificationLastID | Int | The primary key value for latest notification retrieved. |

###### Methods

**wsnotificationCheck**

Purpose: triggers the notification check on the server via a web service.

Returns: NOTHING

Parameters: NONE

**wsnotificationCheck\_CallBack**

Purpose: the call back method for the wsnotificationCheck method.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Result | String | The new notifications encoded in an XML form. |

##### Window

This script is responsible for sending window data from the database layer in case new windows appear on the desktop or old windows are moved.

###### Methods

**wswindowNewWindow**

Purpose: saves a new window’s information in the database.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The HTML ID for the new window. |
| Title | String | The title for the window. |
| Icon | String | The http relative path for the pictogram associated with the window. |
| Link | String | The http path for the web page associated with the window. |
| PosX | Int | The horizontal position in pixels for the window. |
| PosY | Int | The vertical position in pixels for the window. |
| Width | Int | The width in pixels for the window. |
| Height | Int | The height in pisels for the window. |

**wswindowNewWindow\_CallBack**

Purpose: call-back for the wswindowNewWindow to set the database ID for the window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Result | String | The web service returned result representing the HTML ID and DB ID separated by the “|” character. |

**wswindowChangeWindow**

Purpose: updates the specified window data in the database.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The HTML ID for the new window. |
| Title | String | The title for the window. |
| Icon | String | The http relative path for the pictogram associated with the window. |
| Link | String | The http path for the web page associated with the window. |
| PosX | Int | The horizontal position in pixels for the window. |
| PosY | Int | The vertical position in pixels for the window. |
| Width | Int | The width in pixels for the window. |
| Height | Int | The height in pisels for the window. |

**wswindowUnregisterWindow**

Purpose: deletes the specified window from the database.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The HTML ID for the window. |

#### Widgets

The scripts from this folder are used by the widgets displayed in the upper right corner of the interface.

##### About

This script is used by the about widget button.

###### Methods

**aboutToggle**

Purpose: opens an about form or activates it if it is already opened.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Widgets\_About.js | Script |
| Scripts/Resources/RO-RO/Widgets\_About.js | Script |

##### BarCoder

This script is used by the quick link widget button.

###### Methods

**barcoderToggle**

Purpose: opens a quick link form or activates it if it is already opened.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Widgets\_About.js | Script |
| Scripts/Resources/RO-RO/Widgets\_About.js | Script |

##### Harddrive

This script is used by the harddrive widget button.

###### Methods

**harddriveToggle**

Purpose: opens a harddrive form or activates it if it is already opened.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Widgets\_Harddrive.js | Script |
| Scripts/Resources/RO-RO/Widgets\_Harddrive.js | Script |

##### Language

This script is used by the language widget button.

###### Methods

**languageSetLanguage**

Purpose: triggers an interface language change according to the specified language code.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Language | String | The language code in which the interface will be translated. |

**languageToggle**

Purpose: toggles the language selecting window.

Returns: NOTHING

Parameters: NONE

**languageShow**

Purpose: shows the language selecting window.

Returns: NOTHING

Parameters: NONE

**languageHide**

Purpose: hides the language selecting window.

Returns: NOTHING

Parameters: NONE

**languageCheckPosition**

Purpose: checks the position of the language select window in case of a browser resize.

Returns: NOTHING

Parameters: NONE

##### Notification

This script is used by the notification widget button.

###### Methods

**notificationToggle**

Purpose: opens a system notification form or activates it if it is already opened.

Returns: NOTHING

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Resources/EN-UK/Widgets\_Notification.js | Script |
| Scripts/Resources/RO-RO/Widgets\_Notification.js | Script |

##### TabScroller

This script is used by the object displayed in the welcome form in order to animate the sliding panels.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| tabscrollDivider | Int | Used to animate the movement, A tab will move with this fraction relatice to the distance between where it is and where it has to get. |
| tabscrollerSureStep | Int | When the distance between the object’s location and the location where it has to be is less then this variable, the object snaps to the destination location. |

###### Methods

**tabscrollerSwitchItem**

Purpose: counts down and triggers the switch animation between sliding panels.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Object | String | The tab scroller HTML ID. |

**tabscrollerActivateDIV**

Purpose: animates the specified panel activation.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Object | String | The tab scroller HTML ID. |
| Item | Int | The identifier for the panel that must be activated. |

**tabscrollerOnClick**

Purpose: resets the count down until the next change and activates the specified panel.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Object | String | The tab scroller HTML ID. |
| Item | Int | The identifier for the panel that must be activated. |

**tabscrollerAnimate**

Purpose: animates the change between the two specified items.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Object | String | The tab scroller HTML ID. |
| Item1 | Int | The identifier for the panel that is expanded. |
| Item2 | Int | The identifier for the panel that is collapsed. |

##### Zoom

This script is used by the zooming window and by the zoom widget.

###### Methods

**zoomChangeLevel**

Purpose: adjusts the magnification level to the specified one.

Returns: NOTHING

Parameteres:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ZoomLevel | String | The zoom level. |

**zoomSetImage**

Purpose: sets the image that will be manipulated by the magnification window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ImagePath | String | The relative http path for the image file. |
| Width | Int | The original picture’s width in pixels. |
| Height | Int | The original picture’s height in pixels. |

**zoomToggle**

Purpose: toggles the zoom window.

Returns: NOTHING

Parameters: NONE

**zoomShow**

Purpose: shows the zoom window.

Returns: NOTHING

Parameters: NONE

**zoomHide**

Purpose: hides the zoom window.

Returns: NOTHING

Parameters: NONE

**zoomCheckPosition**

Purpose: checks the position and repositions the zoom window in case of a browser resie.

Returns: NOTHING

Parameters: NONE

##### ZoomWindow

This script is used in window displayed pages in order to send an image to the zoom widget.

###### Methods

**zoomMagnifyImage**

Purpose: registers the specified image with the zoom widget and activates the widget.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ImagePath | String | The relative http path for the image file. |
| Width | Int | The original picture’s width in pixels. |
| Height | Int | The original picture’s height in pixels. |

### Styles

In the styles folder we have only two css stylesheets used by the platform and two images used in the style sheets. The files as used as described bellow:

* Desktop.css: the stylesheet used by the main web pages such as Default.aspx, Login.aspx or Login2.aspx
* inputButtonBG.png: the image used as buttons background
* PageBG.png: the backgroung image for the main web pages
* Window.css: the stylesheet used by the web pages displayed in windows on the content section

### Tools

Some features for the platform need additional 3rd party software so we grouped the 3rd party software files in the “Tools” folder. At this point we have the following 3rd party folder:

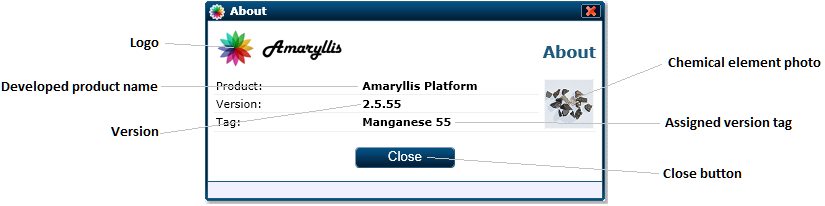
* iTextSharp: used to process HTML content into PDF files. Not the most perfect tool but it is free. The platform can be altered to work with another HTML to PDF library with minimum impact.
* jwPlayer: a free flash player used to play video content. This is used by the documentation module for the moment.
* MIMETeX: a free application that converts a LaTeX math formula into an image. This is used by the documentation module and it works from a command line.

### Web Pages

In this chapter we will pass through all the web pages used by the window interface and explain their functionalities and fields displayed.

#### About

This is a simple web page that describes the platform’s version and it is triggered by the about widget.



The About web page

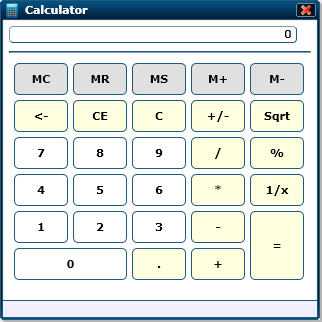
As you can see in the figure above, the only working component for it is the close button displayed at the bottom of the web page. Also this web page will make the window in which it is displayed a fixed border window.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/About/Page\_About.resx | Resources |
| App\_GlobalResources/Pages/About/Page\_About.RO-RO.resx | Resources |
| Images/Pages/About/Element.png | Image |

#### AccessoryCalc

Let us just say that this page was developed just for fun and was left on the platform just to give a more Windows feel to the platform. After developing this web page and played with it a bit in order to test some functionalities we left the page in the Accessory menu of the taskbar. In some cases it helps having a pocket calculator on the platform.



The AccessoryCalc web page

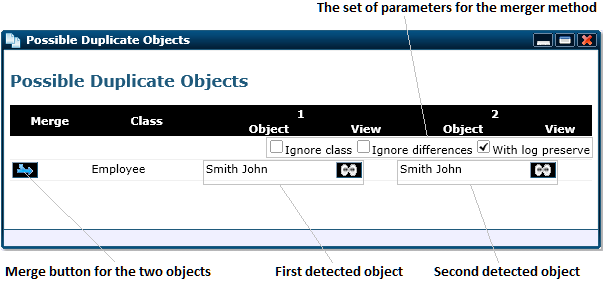
As you can see in the figure above, this is just a regular calculator to help the user compute some arithmetical expressions.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AccessoryCalc/Page\_AccessoryCalc.resx | Resources |
| App\_GlobalResources/Pages/AccessoryCalc/Page\_AccessoryCalc.RO-RO.resx | Resources |

#### AdminDuplicateObjects

This is an administrative page design in order for the user to identify possible duplicated objects and unify them. The objects are displayed in pairs and are identified by considering objects of the same class that have identical values for their attributes.



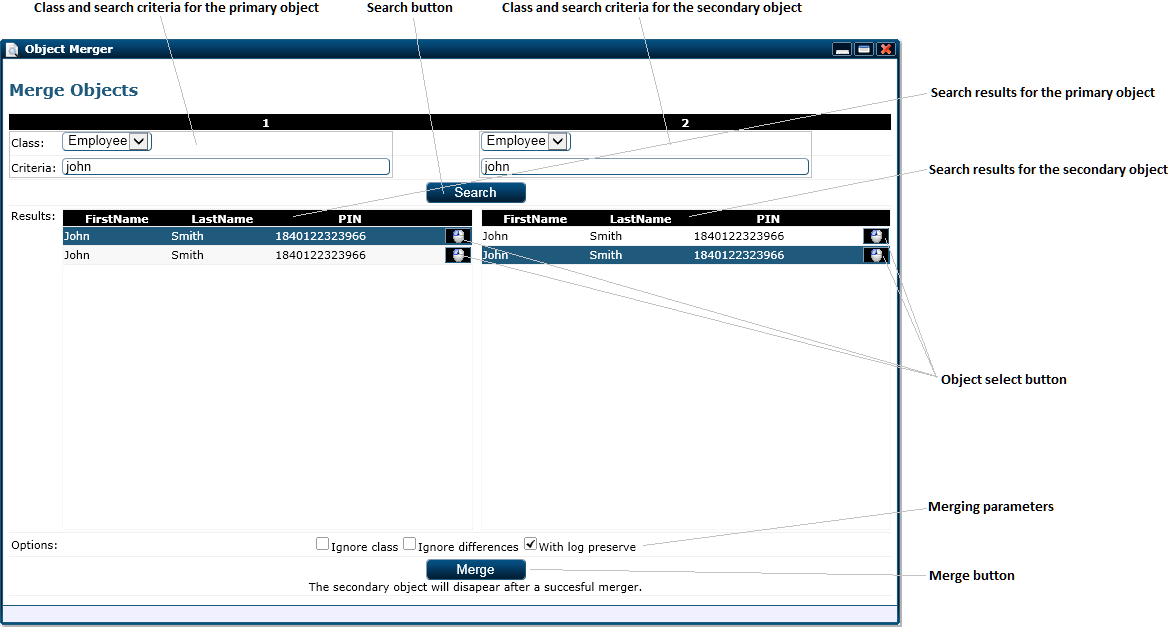
The possible duplicate object web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AdminDuplicateObjects/Page\_AdminDuplicateObjects.resx | Resources |
| App\_GlobalResources/Pages/AdminDuplicateObjects/Page\_AdminDuplicateObjects.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_AdminMergeObjects.js | Script |
| Scripts/Resources/RO-RO/Page\_AdminMergeObjects.js | Script |

#### AdminMergeObjects

This page is responsible for merging two existing platform objects together. The merged objects may be of a different class or different regarding attributes values but if there is a need for the unification then it is possible.



The merge objects web page

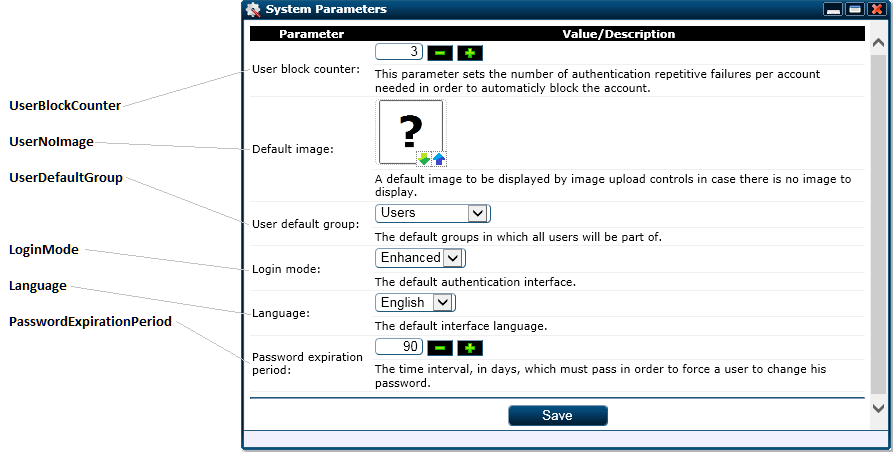
The user is able to search for a primary object for the merger and a secondary object for the merger. After selecting the two objects, the user may select the option for the merger and click on the merge button. The merger options are very important especially when merging two objects of a different class. As a relevant example, you cannot preserve the log if the objects are of a different class.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AdminMergeObjects/Page\_AdminMergeObjects.resx | Resources |
| App\_GlobalResources/Pages/AdminMergeObjects/Page\_AdminMergeObjects.RO-RO.resx | Resources |

#### AdminSettings

This page is for administrative use only and it is destined to change the platform’s constants. All the available constants/parameters for the platform are displayed on this page as shown in the figure below.



The admin settings web page

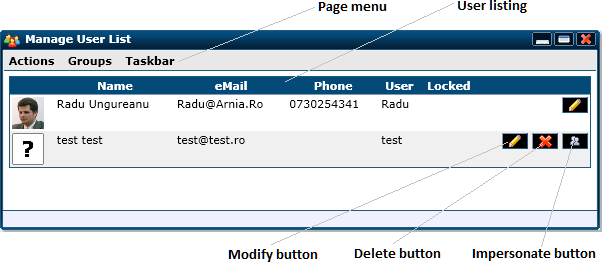
For each constant there is a special section and the value is controlled through the appropriate control. The only constant missing from this page is the “ConstantDummyFile” because there is no point in setting it to something else.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AdminSettings/Page\_AdminSettings.resx | Resources |
| App\_GlobalResources/Pages/AdminSettings/Page\_AdminSettings.RO-RO.resx | Resources |

#### AdminUsers

This is an administrative page design to maintain the user list for the platform. As you can see in the image bellow, you can see all the platform’s users and the options for each user are limited.



The users’ administration list page

The options are limited to modifying, deleting and impersonating a user. As you can see for the current logged on user the options are limited to only modifying because impersonating yourself or delete your own user don’t make sense.

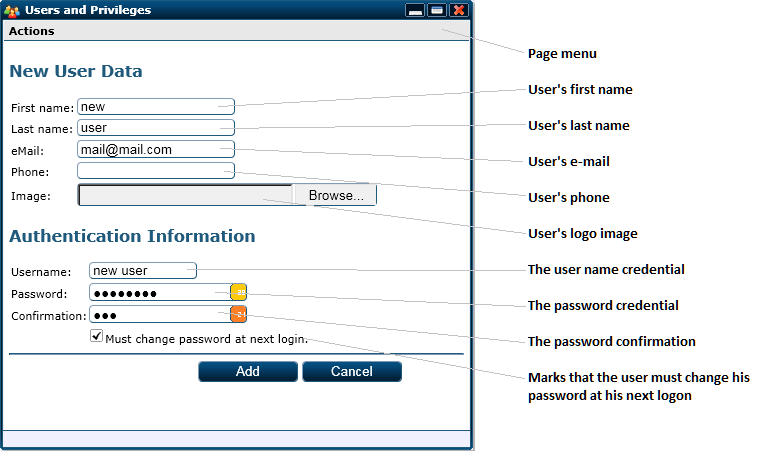
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AdminUsers/Page\_AdminUsers.resx | Resources |
| App\_GlobalResources/Pages/AdminUsers/Page\_AdminUsers.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/AdminUsers.js | Script |
| Scripts/Resources/RO-RO/AdminUsers.js | Script |

#### AdminUsers\_Insert

The main functionality for this web page is to add a new user for the platform. Keep in mind that this web page is also available only for the platform’s administrators.

A new user is tagged by default with the “must change password at next logon” option because his password is set by the administrator. Evidently the option specified can be deactivated and the user will be forced to change his password when the expiration date is reached.



The admin user insert web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AdminUsers\_Insert/Page\_AdminUsers\_Insert.resx | Resources |
| App\_GlobalResources/Pages/AdminUsers\_Insert/Page\_AdminUsers\_Insert.RO-RO.resx | Resources |

#### AdminUsers\_Update

For an existing user we must be able to change his data, so in order to complement the AdminUser\_Insert.aspx, administrators need this page. This page is structured in two panels: one to change the user’s data and one to alter the user’s security group affiliation.

By default a new user is affiliated with the group called “Users” and it can’t be removed from it but the user can be affiliated with other security groups in order to gain access to developed features.



User data update page design for administrators

For the first tab displayed we have the specific user data group into two sections: the user’s personal data and the authentication data. Al this data is saved as provided, except the password. If no new password is specified, then the password for the user won’t be altered.

The second tab is designed to assign the user with security groups. To add a user to a specific group you have to select the group from the drop-down box and click on the “Add” button.

The “Save” button on this web page will save only the user data and the user data is reloaded if a user in added to another user group or removed from one. By logic, the “Cancel” button will close the form so we won’t disguss this further. The “Lock” button will block the user account and prevent the user from loggin on. If the user account is blocked, the “Lock” button will be transformed into the “Unlock” button.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| UserID | Int | The primary key value for the user. |

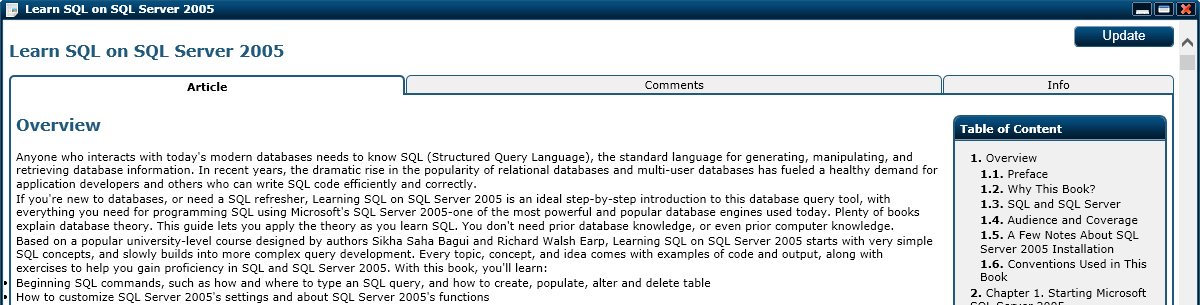
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/AdminUsers\_Update/Page\_AdminUsers\_Update.resx | Resources |
| App\_GlobalResources/Pages/AdminUsers\_Update/Page\_AdminUsers\_Update.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_AdminUsers\_Update.js | Script |
| Scripts/Resources/RO-RO/Page\_AdminUsers\_Update.js | Script |

#### Article

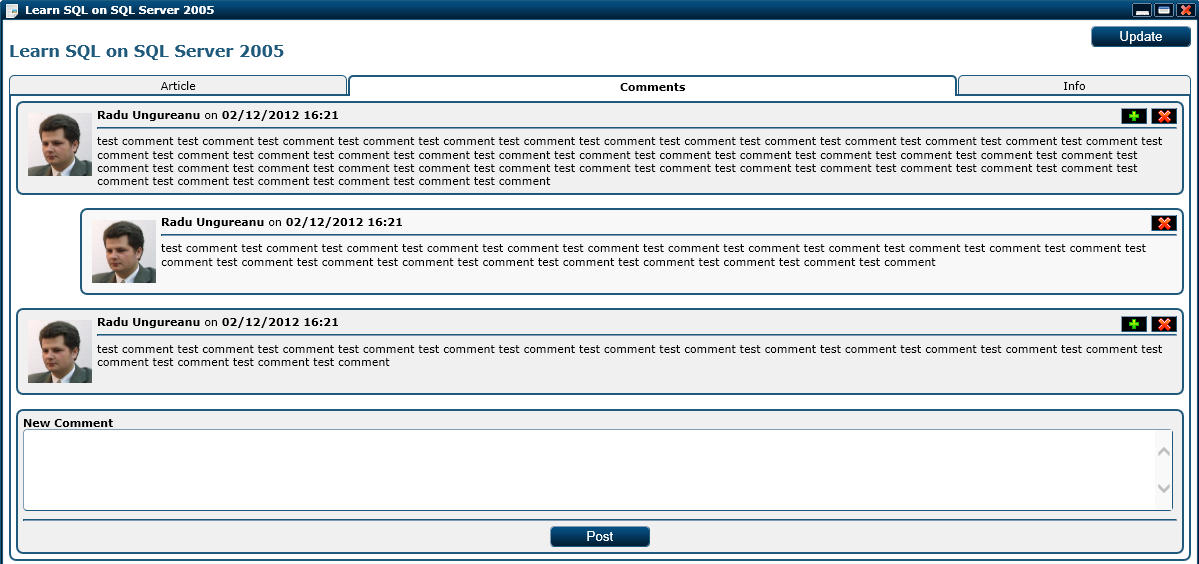
This page is used to display documentation articles to the user and has the functionalities adjacent to the article. The information displayed is structured in three tabs and the first tab displays the articles itself.

For the first tab the web page displays the article’s content and the table of content for the article, these are the main items for an article, thus they take the first main tab.



First tab of the Article web page

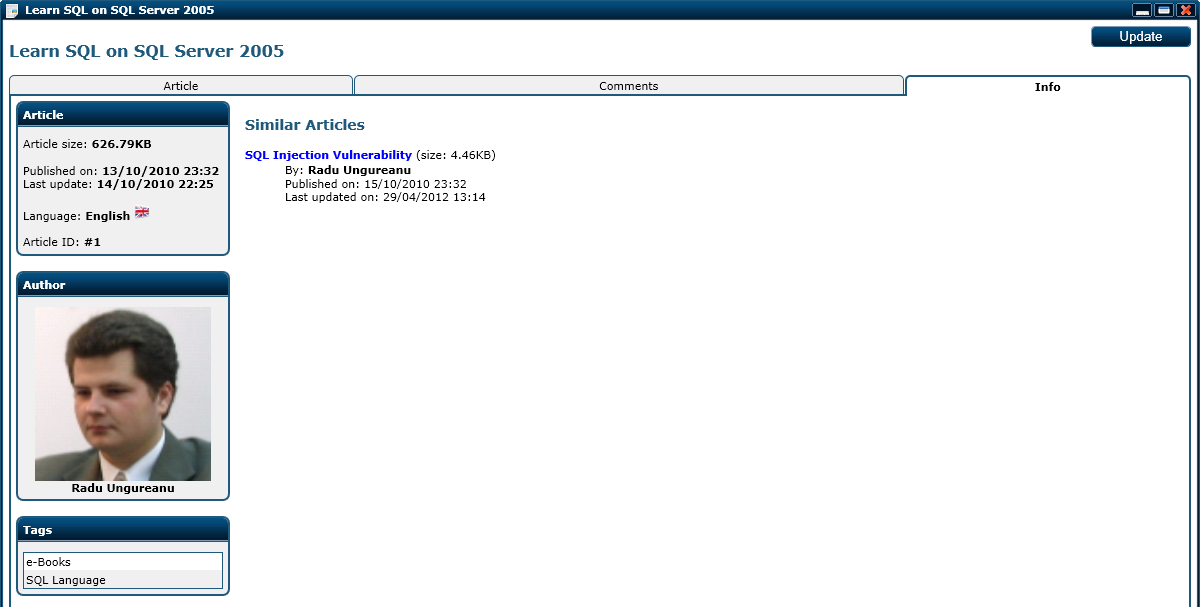
The second tab, named “Comments”, displays the comments left by users regarding the article and gives the user the posibility to add a new comment or a sub comment for another comment.



“Comments” tab of the Article web page

The third and last tab displays the full author’s information and the full article information on the left side and on the right side it displays a list of similar articles in order for the user to be able to navigate to other articles.

NOTE: Similar articles ar determined based on the sets of common tags with the displayed article.



“Info” tab of the Article web page

In all of the three figures shown above, you can see the “Update” button in the upper right corner. This button triggers a web page change for the web page used to update/change an article and it is for administrators and for the article’s author.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ArticleID | Int | The primary key value for the article. |
| ActivePanelID | Int (optional) | The ID for the active panel to activate it when the page is loaded. |

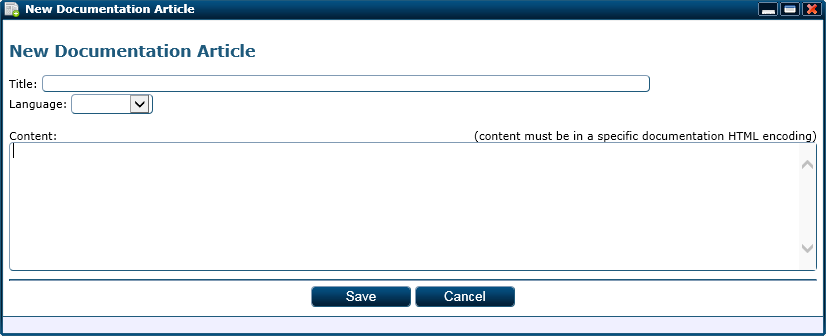
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Article/Page\_Article.resx | Resources |
| App\_GlobalResources/Pages/Article/Page\_Article.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Article.js | Script |
| Scripts/Resources/RO-RO/Page\_Article.js | Script |

#### Article\_Insert

This page is used to insert a new article in the database. The article is defined as this point by the following attributes collected by the form shown bellow:

* Title: the article’s title
* Language: the language in which the article was written
* Content: the article’s content in specific HTML form



The article insert web page

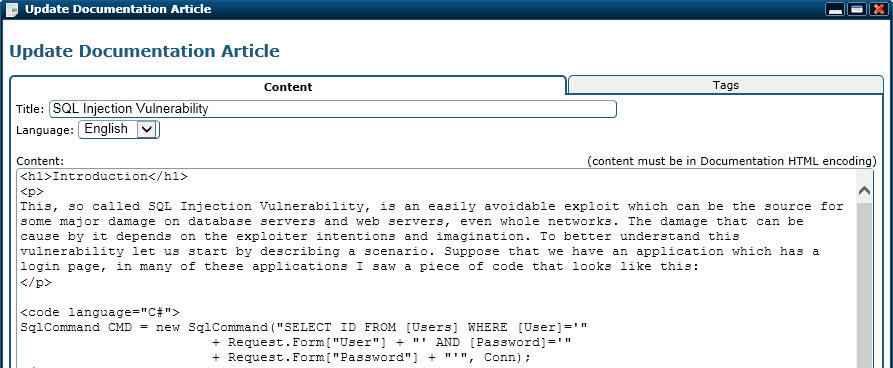
After saving the new article, the user is redirected to the article modify web page.

##### Adjacent Files and Folders

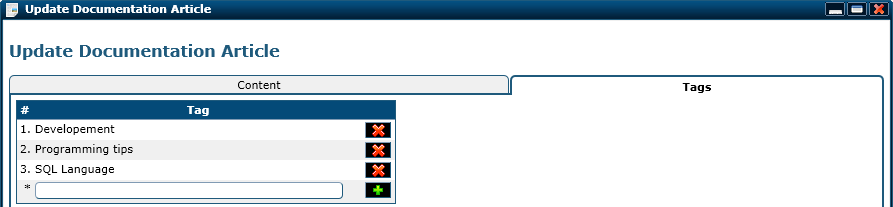
|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Article\_Insert/Page\_Article\_Insert.resx | Resources |
| App\_GlobalResources/Pages/Article\_Insert/Page\_Article\_Insert.RO-RO.resx | Resources |

#### Article\_Update

The article update web page is similar to the article insert page but it has an extra tab needed to update the list of tags linked to the article. Bellow you can see the first tab containing the three attributes shared with the article insert page and the image for the second panel containing the tag list.



The content tab of the article modify web page



The tag list tab of the article modify web page

The tag list management form is very much explanatory. A user can add a new tag or delete an existing tag.

The option buttons for this page displayed at the bottom are the following:

* Save: updates the article’s information according to the new information and resets the article’s content cache and navigation cache
* Delete: deletes the article from the database
* Preview: opens the web page that displays the article in order to get a better view of the article
* Close: closes the article modify form

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ArticleID | Int | The primary key value for the article. |

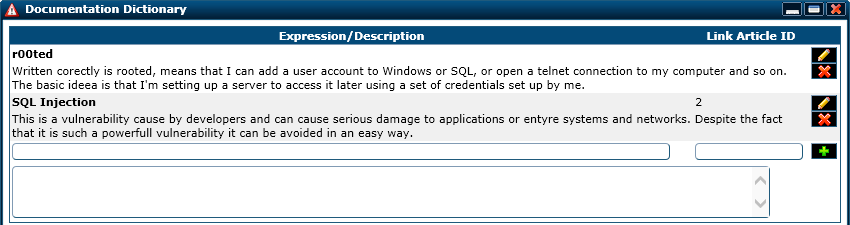
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Article\_Update/Page\_Article\_Update.resx | Resources |
| App\_GlobalResources/Pages/Article\_Update/Page\_Article\_Update.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Article\_Update.js | Script |
| Scripts/Resources/RO-RO/Page\_Article\_Update.js | Script |

#### ArticleLink

This page was design for administrative purposes to manage the expressions that might appear in articles with definitions and link article (if any). An expression is characterized by the following:

* Expression: the expression itself
* Definition: the full definition for the expression
* Article: the primary key value for the article that details the expression



The article link web page

The button functionalities for this page are as follows:

* () triggers the delete for the corresponding exception
* () copies the corresponding expresion’s data to the form bellow in order to be altered
* () saves the data for an expression marked for update or adds the new given exception

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ArticleLink/Page\_ArticleLink.resx | Resources |
| App\_GlobalResources/Pages/ArticleLink/Page\_ArticleLink.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_ArticleLink.js | Script |
| Scripts/Resources/RO-RO/Page\_ArticleLink.js | Script |

#### ArticleSearch

Considering that the documentation database may grow in size this article search page is a must in order to help the user retrieve the desired articles fast.



The article search page with results displayed

As you can see in the above picture, by specifying the search criteria “sql s”, five articles are displayed and ordered by relevance, also they are grouped by author. By clicking on the article’s title link triggers the opening of the article page with the triggered article displayed.

##### Parameters

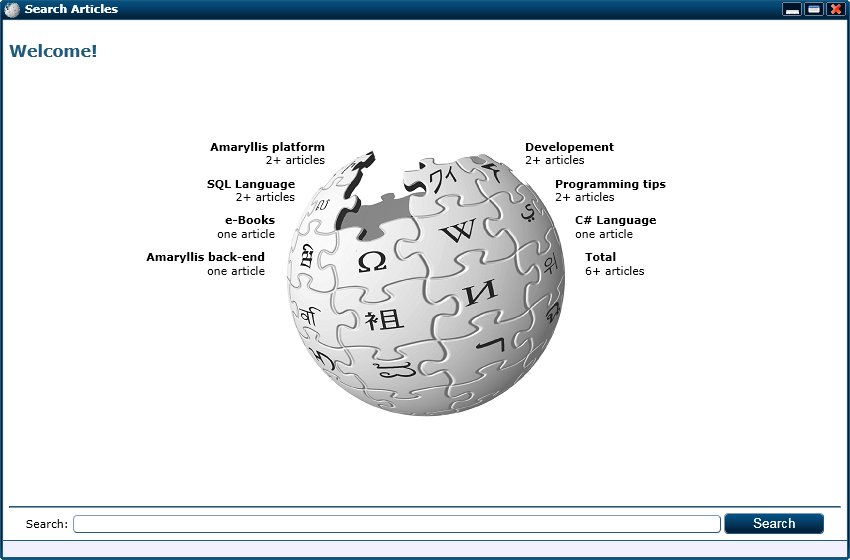
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| Words | String | The search criteria encoded in a base 64 string. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ArticleSearch/Page\_ArticleSearch.resx | Resources |
| App\_GlobalResources/Pages/ArticleSearch/Page\_ArticleSearch.RO-RO.resx | Resources |

#### ArticleStart

This is the start page for the documentation module. Its main feature is to display the most used tags regarding articles number. As you can see in the figure bellow, at the bottom of the page we have a search bar that will redirect the user to the article search page.



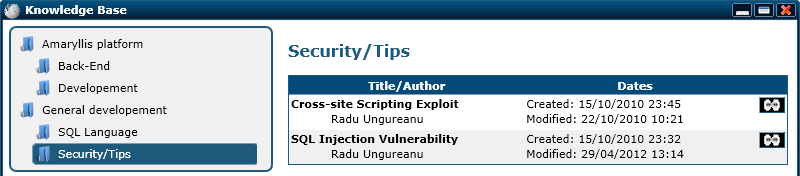
The article start web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ArticleStart/Page\_ArticleStart.resx | Resources |
| App\_GlobalResources/Pages/ArticleStart/Page\_ArticleStart.RO-RO.resx | Resources |

#### ArticleTree

This page displays the documentation folder hierarchy on the left side and the list of articles from the selected folder hierarchy on the right side.



The article tree web page with the list of articles displayed for the “Security/Tips” node.

By triggering the () icon will open an article window with the corresponding icon displayed. The list of articles displayed is generated by a subordered page shown in an iFrame object.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ArticleTree/Page\_ArticleTree.resx | Resources |
| App\_GlobalResources/Pages/ArticleTree/Page\_ArticleTree.RO-RO.resx | Resources |
| ArticleTreeFrame.aspx | Web page |

#### ArticleTreeFrame

This is a web page used by the ArticleTree page to display the list of articles for the selected documentation hierarchy folder. The functionality for this page is part of the ArticleTree wep page functionality.

##### Parameters

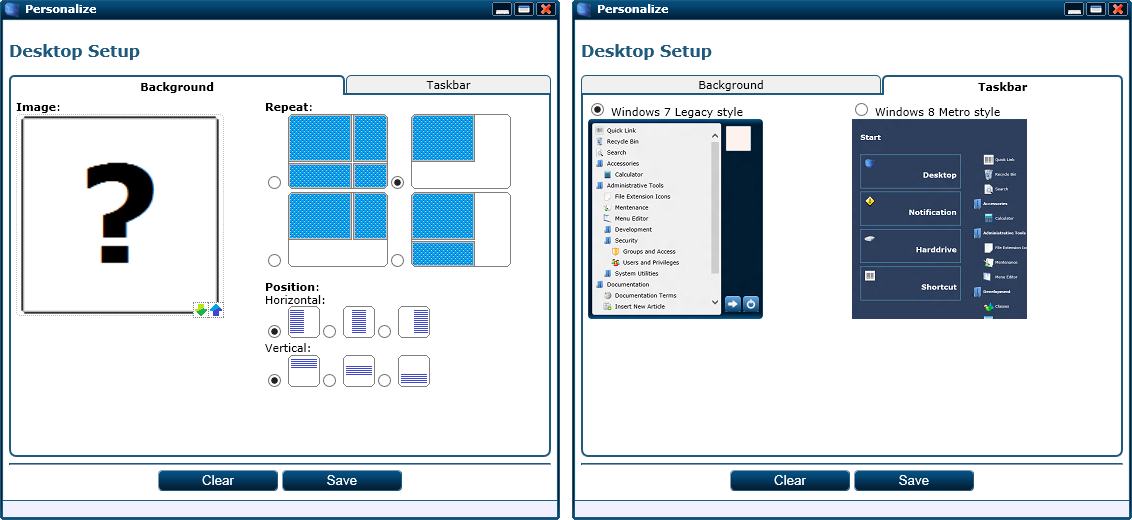
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| TreeID | Int | The primary key value for the tree. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ArticleTreeFrame/Page\_ArticleTreeFrame.resx | Resources |
| App\_GlobalResources/Pages/ArticleTreeFrame/Page\_ArticleTreeFrame.RO-RO.resx | Resources |

#### Background

This web page is actualy used to collect the logged on user;s preferences regarding the interface, meaning that it collects the preferences for the backgroung image and the taskbar style. The informations are collected separately on two tabs as you can see in the image below.



The background web page, first and second panels

First tab is design to collect the information regarding the background image and the way it is displayed and the second tab lets the user choose between a Windows 7 style taskbar and a Windows 8 style taskbar.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Background/Page\_Background.resx | Resources |
| App\_GlobalResources/Pages/Background/Page\_Background.RO-RO.resx | Resources |
| Images/Pages/Background/Horizontal-Center.png | Image |
| Images/Pages/Background/Horizontal-Left.png | Image |
| Images/Pages/Background/Horizontal-Right.png | Image |
| Images/Pages/Background/Repeat-No.png | Image |
| Images/Pages/Background/Repeat-X.png | Image |
| Images/Pages/Background/Repeat-Y.png | Image |
| Images/Pages/Background/Repeat.png | Image |
| Images/Pages/Background/Taskbar7.png | Image |
| Images/Pages/Background/Taskbar8.png | Image |
| Images/Pages/Background/Vertical-Bottom.png | Image |
| Images/Pages/Background/Vertical-Middle.png | Image |
| Images/Pages/Background/Vertical-Top.png | Image |

#### BarCode\_Download

This web page is used to trigger a download for the specified bar code image. It does not contain any visual elements and it has no functionality, other web pages can redirect to this page in order to trigger the browser download.

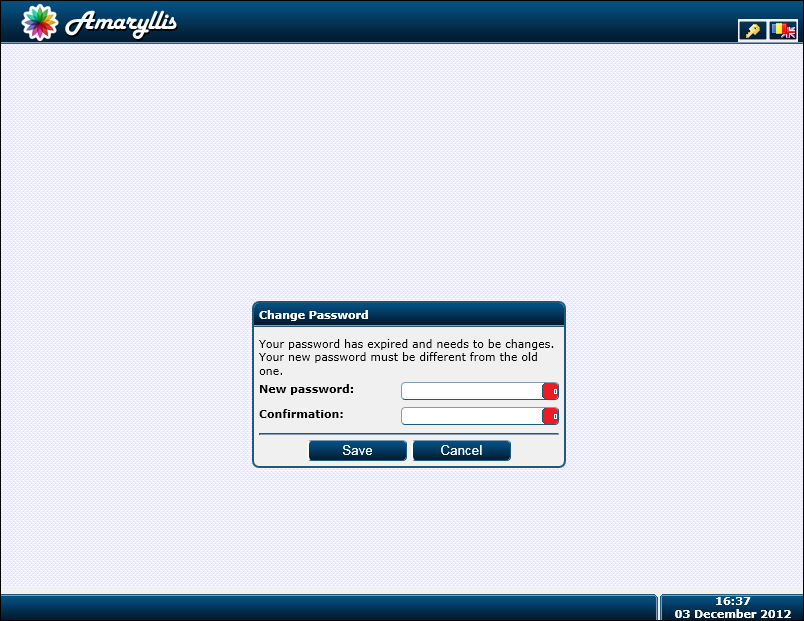
##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| BarCode | String | The bar code value for the bar code image that is downloaded. |

#### ChangePassword

This is an intermediate page between the login page and default page. If the user’s password expired, the user is redirected to this page after loging in order to change his password.

The platform can be set to force users to change their password at a timed interval between 1 and 90 days or when a new user is created he can be marked to change his password at the next logon.



The ChangePassword web page

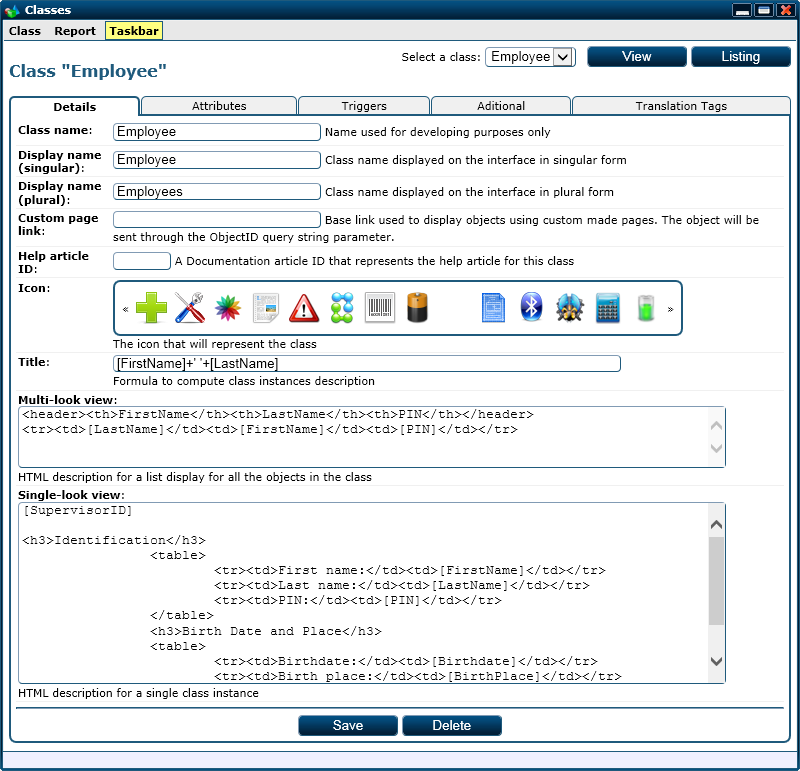
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ChangePassword/Page\_ChangePassword.resx | Resources |
| App\_GlobalResources/Pages/ChangePassword/Page\_ChangePassword.RO-RO.resx | Resources |

#### Class

This page is made to administer and alter classes. The fields are pretty musch self explanatory but we will go over them just in case. The information regarding a class is structured in five tabs:

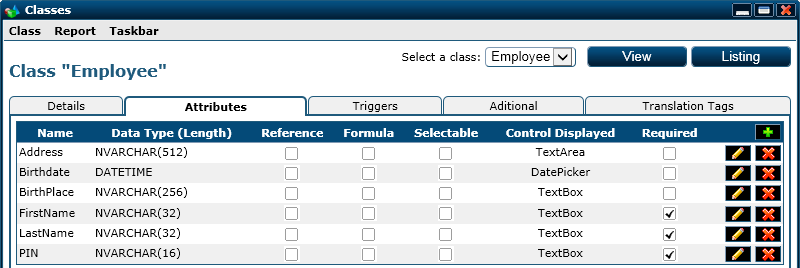
* Details: general information regarding the class
* Attributes: the list of attributes for the class
* Triggers: the scripts for the triggers used by the class
* Aditional: the class hierarchy section, mapping details and security data
* Translation tags: the language tags used by the class



The first tab for the class web page

The fields for this tab are as follows:

* Class name: the internal name for the class used for developing
* Display name (singular): the name of the class in singular form used for display. This is also considered to be the name of a language tag in order to display it on different languages. In the absence of a language tag, the value entered in this field will be displayed.
* Display name (plural): the name of the class in plural form used for display. This is also considered to be the name of a language tag in order to display it on different languages. In the absence of a language tag, the value entered in this field will be displayed.
* Custom page link: the custom link used to display an object of the class. This is given if the standard display form does not suit the developer needs and there is another form developed.
* The primary key value of the documentation article that describe the form for this class to the final user.
* Icon: the pictogram associated with the class.
* Title: a specific SQL expression used to process an object’s title. This expression may contain language tags and the result of the expression must be an NVARCHAR value.
* Multi-look view: a specific HTML sequence which describes how a listing of object for the class must look like. The description may contain language tags in order to make the listing language friendly.
* Single-look view: a specific HTML sequence that describes the form used to collect an object data. This description may also contain language tags in order to make it language friendly.



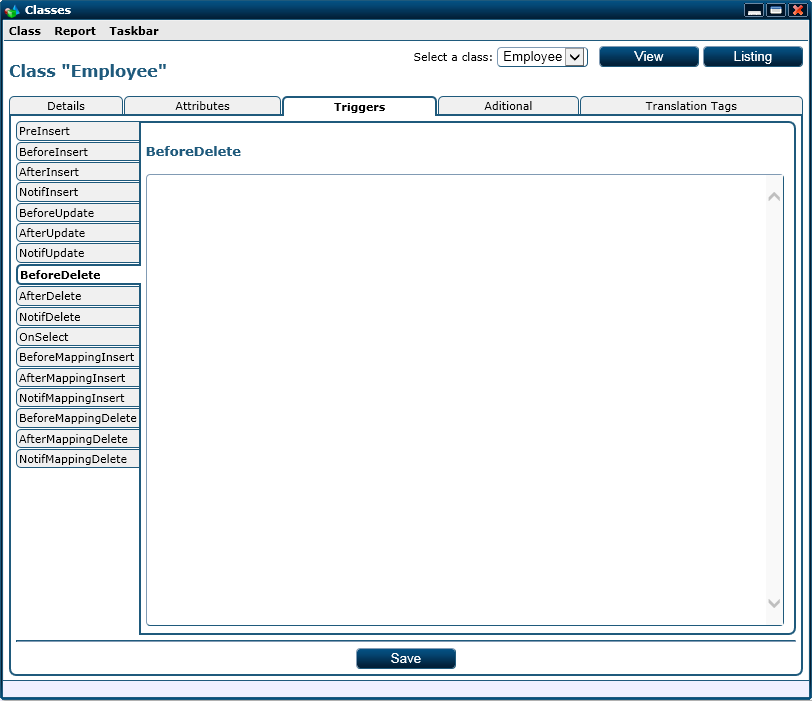
The second tab of the class web page

The second tab displays the list of attributes for the class and a limited set of characteristics for them. The columns displayed are as follows:

* Name: the attribute’s name
* Data type (Length): the attribute’s data type and length (if any)
* Reference: specifies that the attribute contains a primary key value of another object (meaning that is a foreign key)
* Forumla: specifies that the attribute is computed based on a formula
* Selectable: specifies that the value for the attribute is chosen from a set of values (drop-down box, multi-check box, …)
* Control displayed: the name of the control through which the attribute’s value will be collected
* Required: marks the attribute’s value as being required

The buttons displayed for the second tab are as follows:

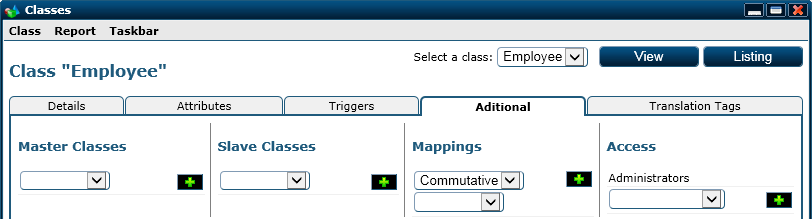
* () is used to trigger a new attribute insert
* () is used to trigger a modification for the corresponding attribute
* () is used to trigger the delete of the corresponding attribute



The third tab of the class web page

The third tab for this web page is destined to alter the event scripts for the class. For each event we have a tab in the side tab panel object displayed. The scripts for the events are specific SQL queries that may contain language tags. The language tags must only be used for the error messages returned or for the notifications returned and they are translated in the proper language for the specific user.

NOTE: the specific SQL query syntax will de discussed in anoter chapter.

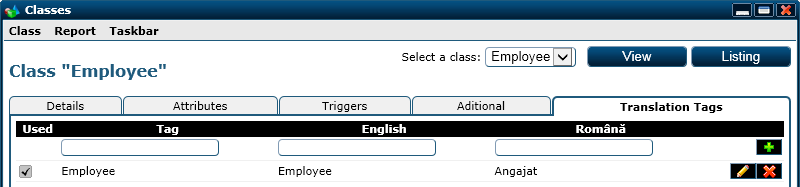


The forth tab of the class web page

The forth tab for this web page is devided into 4 sections as you can see in the image above:

* Master classes: the list of classes that contain possible parent objects for the objects of the displayed class.
* Slave classes: the list of classes of objects that can have objects from the displayed class as parents.
* Mappings: the list of mapping relations that involve the displayed class.
* Access: the list of security groups that have access to the objects of the displayed class. By default, the “Administrators” group has access to all the classes and its access can’t be revoke.

For all of these 4 sections we have an add sections at the end in order to add new list items.



The fifth tab of the class web page

The fifth and final tab of this web page displayes the list of language tags associated with the class. The columns displayed are as follows:

* Used: specifies if the corresponding language tag is used by the class. This helps the developer clean unused language tags or remainds him to use them.
* Tag: the name of the tag.
* The next columns represent the tag’s values for specific languages.

The buttons shown can be used as follows:

* () saves the tag specified and if a tag with the same name exists, it updates it.
* () copies the tag’s information to the row associated with the () button.
* () triggers the delete for the associated tag.

As you can see in the upper right corner, the form displays the following combo used to select a class to be loaded by the class web page or open the listing form for the objects of the displayed class.



The class selecteor

##### Parameters

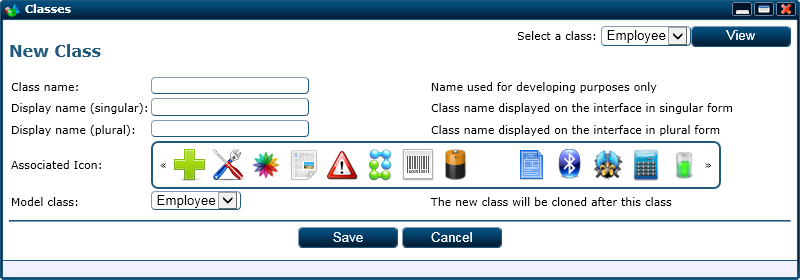
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int (optional) | The primary key value for the class. If this parameters is not specified, the first class in the listing will be displayed. |
| SelectedTab | Int (optional) | The ID for the active panel to activate it when the page is loaded. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Class/Page\_Class.resx | Resources |
| App\_GlobalResources/Pages/Class/Page\_Class.RO-RO.resx | Resources |
| Images/Pages/Class/Unused.png | Image |
| Images/Pages/Class/Used.png | Image |
| Scripts/Resources/EN-UK/Page\_Class.js | Script |
| Scripts/Resources/RO-RO/Page\_Class.js | Script |

#### Class\_Insert

This web page is used to add a new class into the system. In order to create a new class, fewer items are required and the rest can be filled using the Class web page.



The Class\_Insert web page

The fields for this tab are as follows:

* Class name: the internal name for the class used for developing
* Display name (singular): the name of the class in singular form used for display. This is also considered to be the name of a language tag in order to display it on different languages. In the absence of a language tag, the value entered in this field will be displayed.
* Display name (plural): the name of the class in plural form used for display. This is also considered to be the name of a language tag in order to display it on different languages. In the absence of a language tag, the value entered in this field will be displayed.
* Associated Icon: the pictogram associated with the class.
* Model class: the class that is used as a model for the new class. By specifying this, the new class will be a copy of the specified class (attributes, descriptions, event, everything).

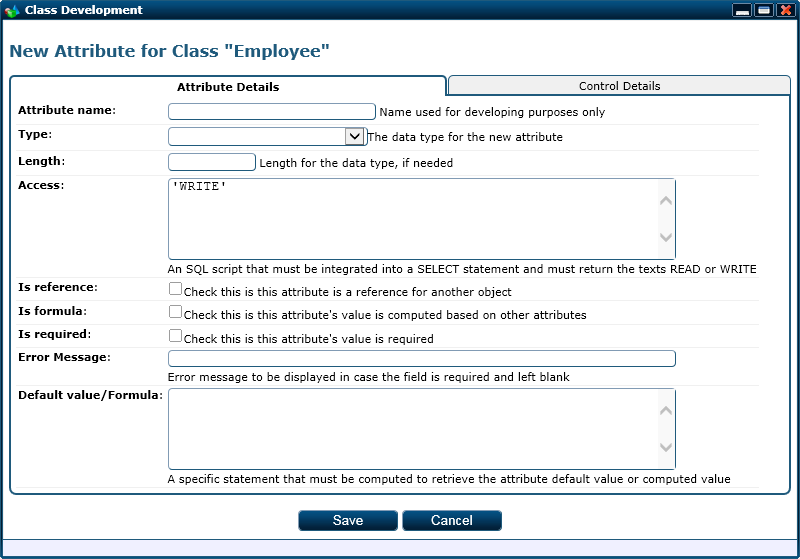
In the upper right corner we can see a similar structure as on the Class web page. The only thing missing is the “Listing” button which has no relevance on this web page.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Class\_Insert/Page\_Class\_Insert.resx | Resources |
| App\_GlobalResources/Pages/Class\_Insert/Page\_Class\_Insert.RO-RO.resx | Resources |

#### Class\_InsertAttribute

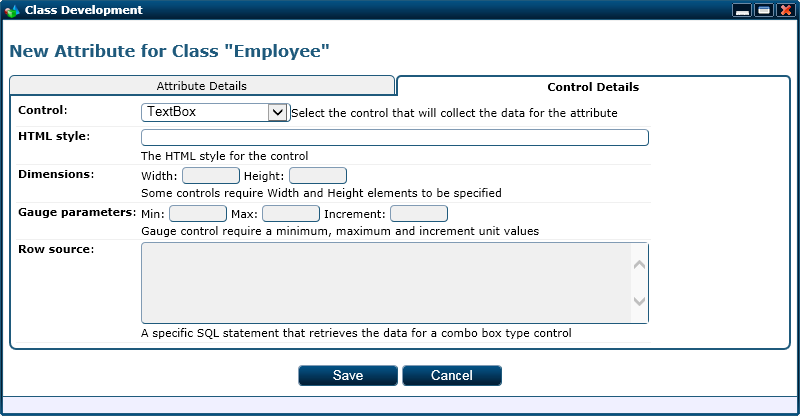
This page is used to add a new attribute to a class and also specify the new attributes characteristics. The information regarding a new attribute is split into the two sections seen in the images bellow. The first section refers to the attributes characteristics needed for the database schema and data processing and the second section refers to the characteristics that influence the interface impact of the attribute.



The first section of the class attribute insert web page

For this first section we have the following fields:

* Attribute name: represents the internal name of the attribute and must be unique inside the class. This name must be different from the following: ID, ParentID, Title, LogID, UserName and Date. The six names are reserved for internal use and an error will be thrown if the attribute name coincides with any one of them.
* Type: the data type for the attribute’s value which is restricted to the given list.
* Length: for some data type this field is required (such as NVARCHAR). The possible values for this field are integers between 1 and 4000 or the text “MAX”.
* Access: this must be a specific SQL expression that can be integrated as a column in a SELECT statement. The values that this expression must return are “WRITE” to mark that the attribute’s value is editable by the user and “READ” to mark that the attribute’s value is read only.
* Is reference: marks the attribute as being a reference to another object. If this field is checked then the data type of the attribute must be INT.
* Is formula: marks the attribute as being computed. If this field is checked then the default value/formula field will contain the formula by which the attribute’s value will be computed and the access field must contain an expression that returns always the “READ” value (for this, the simplest expression is “ ’READ’ ”).
* Is required: marks the attribute as being mandatory and the user must fill in a value for it. This field is incompatible with the “is required” field because if it is computed it will be read only.
* Error message: this will represent the error message that will be displayed to the user if the attribute is marked as being required and the user lefts it blank. Keep in mind that the error message may containg language tags that will be replaced according to the interface’s language.
* Default value/formula: a specific SQL expression that can be integrated as a column in a SELECT statement. The value returned by this expression will be the default value for the attribute or the computed value for the attribute if the “is formula” field is checked. As a default value field, the expression is computed before displaying the class form to the user. As a formula field, the expression is computed post a save operation and it can contain other attribute tags.



The second section of the class attribute insert web page

The second section for this web page contains the information regarding the control through which the attribute’s value will be collected. The fields displayed are the following:

* Control: the control name for the control through which the attribute’s value will be collected.
* HTML style: the HTML style for the control. Some controls can be altered through this field in order to change their apparence. Example: for a text box field the size may be increased by filling in “width: 300px;” in this field.
* Width: some controls do not support HTML styles and their width can’t be altered that way so the width must be specified this way. Example: the image upload control requires the width of the displayed picture to be specified through this field.
* Height: similar to the width field but is used for height.
* Min: represents the minimum value for the attribute’s value. Example: the gauge control selects a floating value inside a range; this field specifies the lower bound of the range.
* Max: similar to the min field but used for a maximum field value.
* Unit: represents the rounding step for the collected value. Example: for a gauge control the chosen float number will be rounded to the specified incremental values.
* Row source: a specific SQL query that returns a table containing the columns and values used by the control. Example: for a select control the query specified here returns the options that the control will offer.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int | The primary key value for the parent class. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Class\_InsertAttribute/Page\_Class\_InsertAttribute.resx | Resources |
| App\_GlobalResources/Pages/Class\_InsertAttribute/Page\_Class\_InsertAttribute.RO-RO.resx | Resources |

#### Class\_UpdateAttribute

This page is similar in functionality and fields with the Class\_InsertAttribute web page. The only two differences between the mentioned web page and this one are the following:

* the update page will load the specified attribute’s characteristics
* The title displayed by the web page will look as seen in the image bellow



The title model displayed by the class update attribute web page

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| AttributeID | Int | The primary key value for the class’s attribute. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Class\_UpdateAttribute/Page\_Class\_UpdateAttribute.resx | Resources |
| App\_GlobalResources/Pages/Class\_UpdateAttribute/Page\_Class\_UpdateAttribute.RO-RO.resx | Resources |

#### Controls\_FileUpdate

This page is used by the file upload control and it is displayed inside a div element when the control is triggered. As you can see in the image bellow, this web page contains a single upload file field and two buttons (one to trigger the upload and one to deactivate the control).



The file upload control’s adjacent web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Controls\_FileUpdate/Page\_Controls\_FileUpdate.resx | Resources |
| App\_GlobalResources/Pages/Controls\_FileUpdate/Page\_Controls\_FileUpdate.RO-RO.resx | Resources |

#### Controls\_FileUpdate

This page is used by the image upload control and it is displayed inside a div element when the control is triggered. As you can see in the image bellow, this web page contains a single upload file field and two buttons (one to trigger the upload and one to deactivate the control).



The image upload control’s adjacent web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Controls\_ImageUpdate/Page\_Controls\_ImageUpdate.resx | Resources |
| App\_GlobalResources/Pages/Controls\_ImageUpdate/Page\_Controls\_ImageUpdate.RO-RO.resx | Resources |

#### Default

This is the main/master web page for the entire platform and it is displayed only to authenticated users. Only on this web page the platform displayes the task bar, icons, windows and also the full widget bar.



The default web page

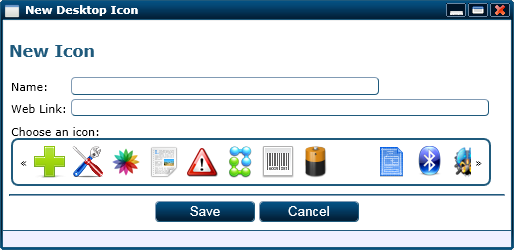
Window and icon interaction are mediated through this web page and the taskbar can be used to open web pages or create icons on the content section.

* + - * 1. Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Default/Page\_Default.resx | Resources |
| App\_GlobalResources/Pages/Default/Page\_Default.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Default.js | Script |
| Scripts/Resources/RO-RO/Page\_Default.js | Script |

#### DesktopIcon\_Insert

This web page is used to add a new icon to the content section. This web page is fairly simple to use but we will go over the fields.



The desktop icon insert web page

The fields for this web page are as follows:

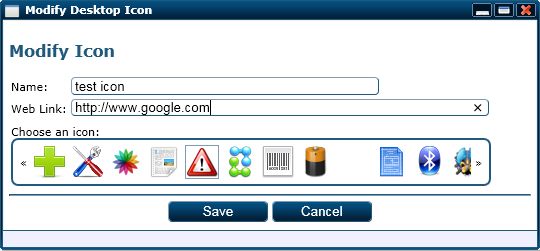
* Name: the title for the icon.
* Web link: the http path for the web page that must be opened in a new window when the icon is triggered.
* Choose an icon: the pictogram associated with the icon.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/DesktopIcon\_Insert/Page\_DesktopIcon\_Insert.resx | Resources |
| App\_GlobalResources/Pages/DesktopIcon\_Insert/Page\_DesktopIcon\_Insert.RO-RO.resx | Resources |

#### DesktopIcon\_Update

The functionality for this web page is similar to the DesktopIcon\_Insert web page. The display differs a bit due to the title and the fact that the page will load the specified icons properties.



The desktop icon update web page

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| IconID | Int | The primary key value for the icon. |
| DesktopIconID | Int | The HTML ID of the icon. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/DesktopIcon\_Update/Page\_DesktopIcon\_Update.resx | Resources |
| App\_GlobalResources/Pages/DesktopIcon\_Update/Page\_DesktopIcon\_Update.RO-RO.resx | Resources |

#### Download

This page is used to trigger file downloads and it is limited to the “Files” and “Files/Temp” folders. It is similar to the BarCode\_Download web page in terms of code so we won’t discuss it any further.

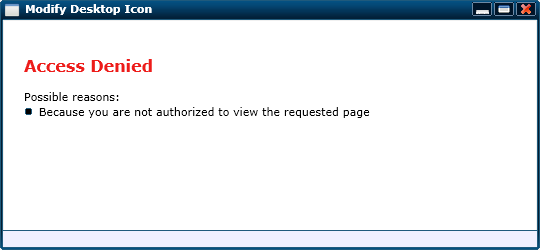
##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FilePath | String | The primary key value for the file or the temporary GUID for a temporary file. |

#### Error

This web page is used to display platform errors to the user. The set of errors displayed is limited to the generic errors encountered when operating the platform and it doesn’t cover exceptions regarding the written code (such as a badly formed SQL string or an untreated C# exception).

The page itself doesn’t offer options to the user, it just displayes the error message as you can see in the image bellow.



The error web page for an access denied error

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| Code | String | The internal code for the encountered error. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Error/Page\_Error.resx | Resources |
| App\_GlobalResources/Pages/Error/Page\_Error.RO-RO.resx | Resources |

#### ExportDownload

This web page is used to trigger downloads for the files that contain exported data (such as report results). It is a purely server side web page and it is not displayed to the user so we won’t go into much detail here.

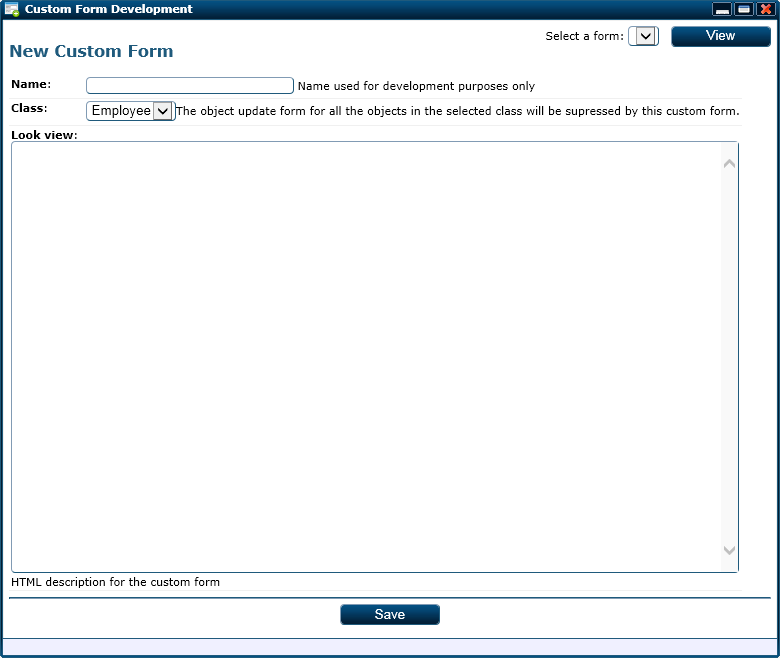
NOTE: Keep in mind that the files downloaded through this web page will be deleted from the server after the download is complete.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| File | String | The temporary file’s name. |

#### Form\_Insert

This web page is used to add a new custom form on the platform. For some classes the basic object view may not be enough so a particularized form view is needed. This being said, the need for this module was natural and developers are able to compose custom forms using this page.



The custom form insert web page

The fields for this web page are as follows:

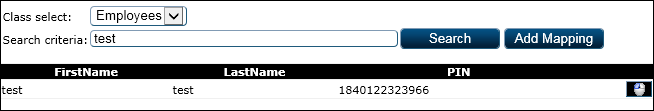
* Name: the internal name for the custom form used internaly.
* Class: the class for which the custom form is destined for.
* Look view: a specific HTML description that describes how the custom form must look like.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Form\_Insert/Page\_Form\_Insert.resx | Resources |
| App\_GlobalResources/Pages/Form\_Insert/Page\_Form\_Insert.RO-RO.resx | Resources |

#### FormAddMapping

This web page is a utility page for the FormView web page and it is used to map the main object for the displayed form with other objects.



The custom form add mapping adjacent web page

This web page is displayed on the FormView web page inside an iframe and it will trigger a refresh on the parent web page if an “add mapping” operation is triggered.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int | The primary key value for the class. |

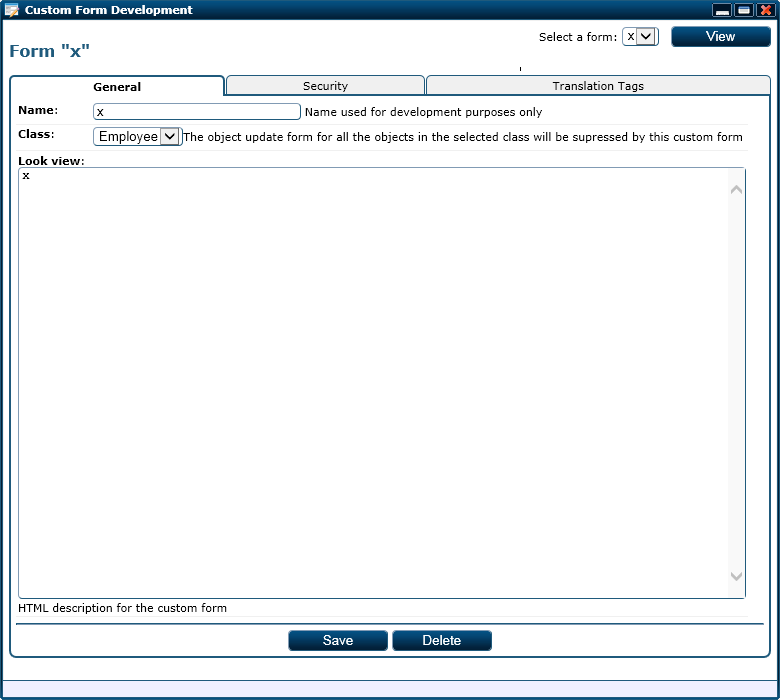
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/FormAddMapping/Page\_FormAddMapping.resx | Resources |
| App\_GlobalResources/Pages/FormAddMapping/Page\_FormAddMapping.RO-RO.resx | Resources |

#### Forms

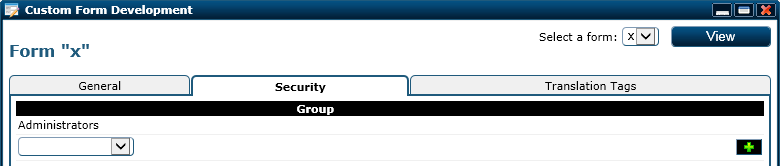
This web page is used to alter existing custom forms. The functionality and look is similar with the Form\_Insert web page except the fact that we have two extra sections dedicated to the translation tags and security.

As said, the first section contains the displayed custom form data.



The first section of the custom forms web page

The second section is dedicated for the security part and determines which user groups have access to the displayed custom form.



The second section of the custom forms web page

Group access can be granted or revoked and by default the “Administrators” group has access on all the custom forms and its access can’t be revoked.



The third section of the custom forms web page

The third section manages the language tags used by the displayed custom form.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FormID | Int | The primary key value for the custom form. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Forms/Page\_Forms.resx | Resources |
| App\_GlobalResources/Pages/Forms/Page\_Forms.RO-RO.resx | Resources |
| Images/Pages/Forms/Unused.png | Image |
| Images/Pages/Forms/Used.png | Image |
| Scripts/Resources/EN-UK/Page\_Forms.js | Script |
| Scripts/Resources/RO-RO/Page\_Forms.js | Script |

#### FormView

This web page displays a specified custom form for a root object of the corresponding class. Also, this web page provides the needed functionality for the custom form.

A preview for this form is pointless because its apparence is based on the custom form displayed.

##### Parameters

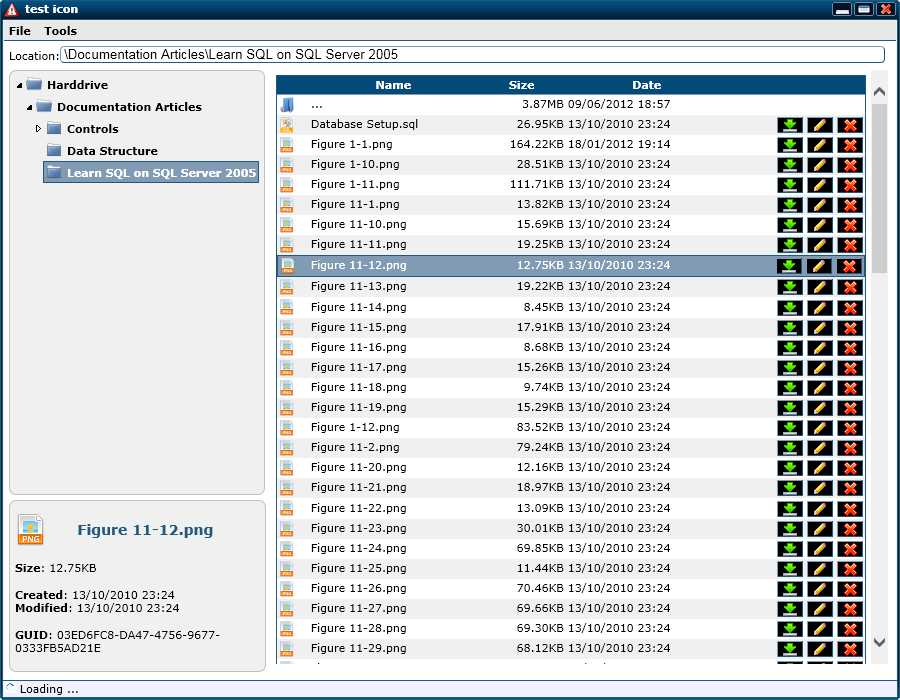
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FormID | Int | The primary key value for the custom form. |
| ObjectID | Int | The primary key value for the root object. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/FormView/Page\_FormView.resx | Resources |
| App\_GlobalResources/Pages/FormView/Page\_FormView.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_FormView.js | Script |
| Scripts/Resources/RO-RO/Page\_FormView.js | Script |
| FormAddMapping.aspx | Web page |

#### Harddrive

This is the virtual drive main page that displayes the folder hierarchy for the logged on user as well as single file information and folder sub-items listings.



The harddrive web page

As you can see in the above image, on the left side we have a gray column which displays the hierarchical folder structure for the user and a box of information regarding the last selected file or folder. On the right side we have the items for the “Learn SQL on SQL Server 2005” folder listed with the proper options:

* () triggers the alter page for the given item.
* () triggers the delete operation for the given item.
* () opens the item if it is a folder.
* () triggers the download of the item if it is a file.

NOTE: On the top of the page the virtual path for the displayed folder is shown in a read only mode.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FolderID | Int (optional) | The primary key value for the folder that must be activated by default. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Harddrive/Page\_Harddrive.resx | Resources |
| App\_GlobalResources/Pages/Harddrive/Page\_Harddrive.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Harddrive.js | Script |
| Scripts/Resources/RO-RO/Page\_Harddrive.js | Script |
| HarddriveDownload.aspx | Web page |
| HarddriveFile\_Insert.aspx | Web page |
| HarddriveFile\_Update.aspx | Web page |
| HarddriveFolder\_Insert.aspx | Web page |
| HarddriveFolder\_Update.aspx | Web page |
| HarddrivePage.aspx | Web page |
| HarddriveRefresh.aspx | Web page |
| HarddriveSearch.aspx | Web page |

#### HarddriveDownload

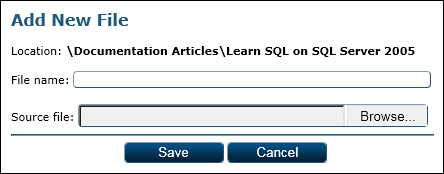
This web page is used to trigger file downloads and it is limited to the files that are loaded in the virtual harddrive module.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FolderID | Int (optional) | The primary key value for the folder that must be activated by default. |

#### HarddriveFile\_Insert

This web page is design to give the user the posibility to add a new file to his virtual drive structure in the folder sent via a parameter.



The add a new file web page

This web page displayes the following fields:

* Location: information only field displayed in order for the user to know in which folder the new file will be placed.
* File name: the file’s name inside the virtual drive module.
* Source file: the sources file from the user’s harddrive that must be uploaded.

##### Parameters

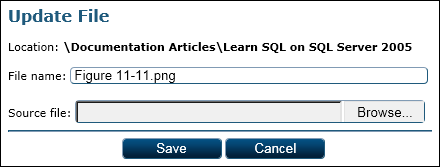
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FolderID | Int (optional) | The primary key value for the parent folder. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddriveFile\_Insert/Page\_HarddriveFile\_Insert.resx | Resources |
| App\_GlobalResources/Pages/HarddriveFile\_Insert/Page\_HarddriveFile\_Insert.RO-RO.resx | Resources |

#### HarddriveFile\_Update

This web page is design to give the user the posibility to update an existing file from his virtual drive structure.



The update an existing file web page

This web page displayes the following fields:

* Location: information only field displayed in order for the user to know in which folder the file is located.
* File name: the file’s name inside the virtual drive module.
* Source file: the sources file from the user’s harddrive that must be uploaded.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FileID | Int | The primary key value for the file. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddriveFile\_Update/Page\_HarddriveFile\_Update.resx | Resources |
| App\_GlobalResources/Pages/HarddriveFile\_Update/Page\_HarddriveFile\_Update.RO-RO.resx | Resources |

#### HarddriveFolder\_Insert

This web page is design to give the user the posibility to add a new folder to his virtual drive structure subordered to the specified folder sent via a parameter.



The add a new folder web page

This web page displayes the following fields:

* Location: information only field displayed in order for the user to know in which folder the new file will be placed.
* Folder name: the folder’s name inside the virtual drive module.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FolderID | Int (optional) | The primary key value for the parent folder. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddriveFolder\_Insert/Page\_HarddriveFolder\_Insert.resx | Resources |
| App\_GlobalResources/Pages/HarddriveFolder\_Insert/Page\_HarddriveFolder\_Insert.RO-RO.resx | Resources |

#### HarddriveFolder\_Update

This web page is design to give the user the posibility to update an existing folder from his virtual drive structure.



The update an existing folder web page

This web page displayes the following fields:

* Location: information only field displayed in order for the user to know in which folder the file is located.
* Folder name: the folder’s name inside the virtual drive module.

##### Parameters

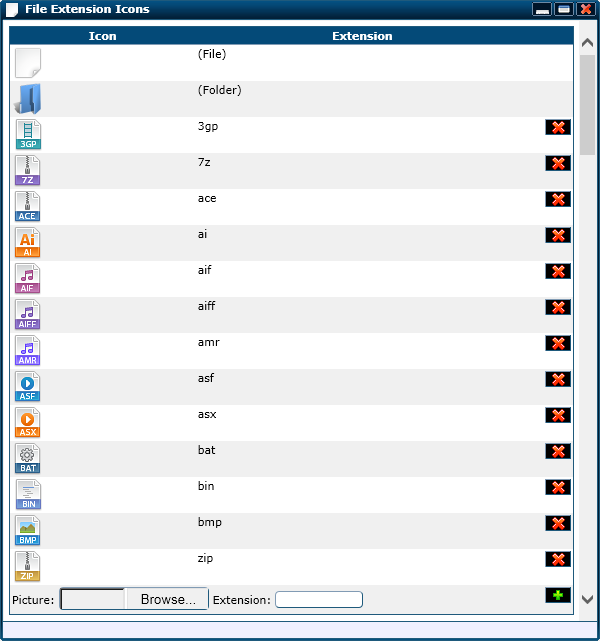
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FolderID | Int | The primary key value for the folder. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddriveFolder\_Update/Page\_HarddriveFolder\_Update.resx | Resources |
| App\_GlobalResources/Pages/HarddriveFolder\_Update/Page\_HarddriveFolder\_Update.RO-RO.resx | Resources |

#### HarddriveIcons

This web page is used to update and maintain the list of pictograms associated with file icons. The pictograms are used by the harddrive module when a file listing is displayed.



The file extension icons web page

There are two extensions on this web page that can’t be deleted but they can be updated. Those two are the generic file pictograme (“(File)”) and the folder pictograme (“(Folder)”).

At the bottom of the listing the following fields are displayed:

* Picture: the file source from the harddrive that represents the given extension pictogram.
* Extension: the extension itself.

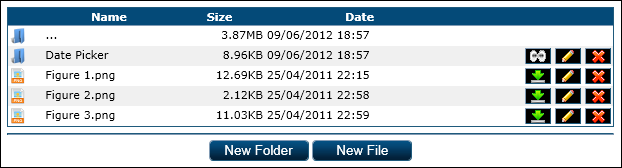
If the specified extension exists than the associated pictograme will be updated else it will be created.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddriveIcon/Page\_HarddriveIcon.resx | Resources |
| App\_GlobalResources/Pages/HarddriveIcon/Page\_HarddriveIcon.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_HarddriveIcons.js | Script |
| Scripts/Resources/RO-RO/Page\_HarddriveIcons.js | Script |

#### HarddrivePage

This web page is used by the Harddrive web page in order to display the list of subordered items (files and folders) for a specified folder.



The harddrive page web page

The functionality for this web page was discussed earlier for the Harddrive web page so we won’t go into this subject any further.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| FolderID | Int | The primary key value for the folder. |

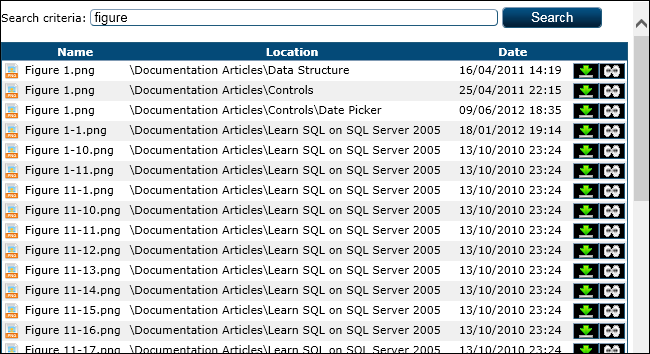
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddrivePage/Page\_HarddrivePage.resx | Resources |
| App\_GlobalResources/Pages/HarddrivePage/Page\_HarddrivePage.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_HarddrivePage.js | Script |
| Scripts/Resources/RO-RO/Page\_HarddrivePage.js | Script |

#### HarddriveSearch

This web page is used by the harddrive module in order to give the user the posibility to search through his files and folders. The found files and folders are displayed in a list with the following adjacent options:

* () triggers a file download for the associated file.
* () opend the folder that contain the associated file or the listed found folder.



The harddrive page web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/HarddriveSearch/Page\_HarddriveSearch.resx | Resources |
| App\_GlobalResources/Pages/HarddriveSearch/Page\_HarddriveSearch.RO-RO.resx | Resources |

#### Login

As suggested, this is the main login web page for the platform and it represents the enhanced login mode design to look like a Windows Vista/7 login page but is has some Linus influences. The user can click on a shown user picture or enter the user’s name in the displayed box and the web page will scroll to the corresponding user image. After that, the user will be asked for his password in order to login.

NOTE: the scrolling user images are an influence from Windows Vista/7 and the two stage login is taken from the common Linux login interface.



The first stage of the login web page



The second stage of the login web page

##### Parameters

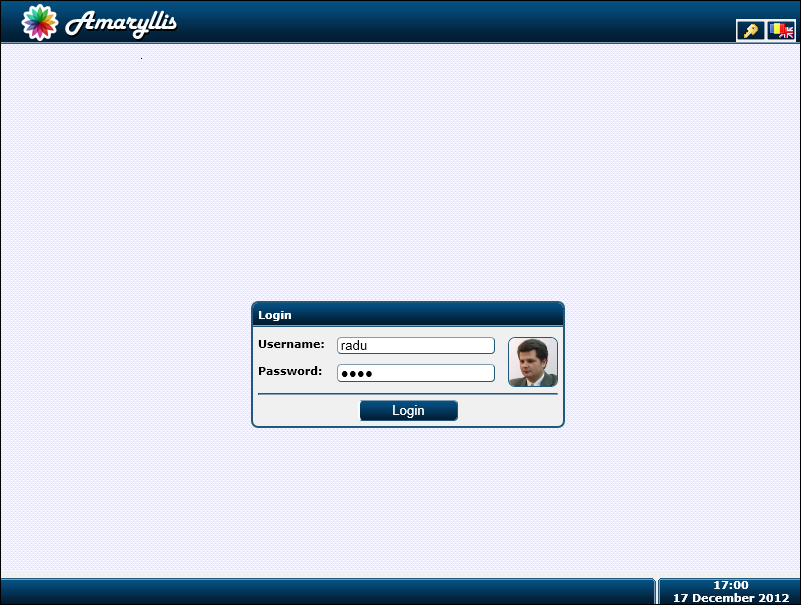
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| DefaultLogin | String (optional) | If the default login mode differs from the Enhanced mode, the login web page will redirect to the Login2 web page unless this parameter is specified. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Login/Page\_Login.resx | Resources |
| App\_GlobalResources/Pages/Login/Page\_Login.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Login.js | Script |
| Scripts/Resources/RO-RO/Page\_Login.js | Script |

#### Login2

This web page is similar to the Login web page but it is a straight forward one stage login similar to the Windows 2003/2008 Serves login with one difference, after the user name is filled in the user’s image is updated.



The Login2 web page

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| DefaultLogin | String (optional) | If the default login mode differs from the Normal mode, the login web page will redirect to the Login web page unless this parameter is specified. |

##### Adjacent Files and Folders

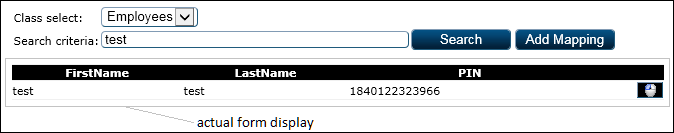
|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Login/Page\_Login2.resx | Resources |
| App\_GlobalResources/Pages/Login/Page\_Login2.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Login2.js | Script |
| Scripts/Resources/RO-RO/Page\_Login2.js | Script |

#### Logout

This is a non-display web page and it is used to terminate a user session by setting the ID and Impersonation session variables to the “null” value.

#### MappingObject

This web page is used by the Object\_Update web page in order to map the displayed object with another object from the database according to the defined mapping relations. This web page displays a list of objects for the specified class that correspond to the given search criteria.



The mapping object web page with some elements from the Object\_Update master web page

The only functionality here is given by the () select button that sends the primary key value of the selected object to the parent web page in order to establish the mapping.

##### Parameters

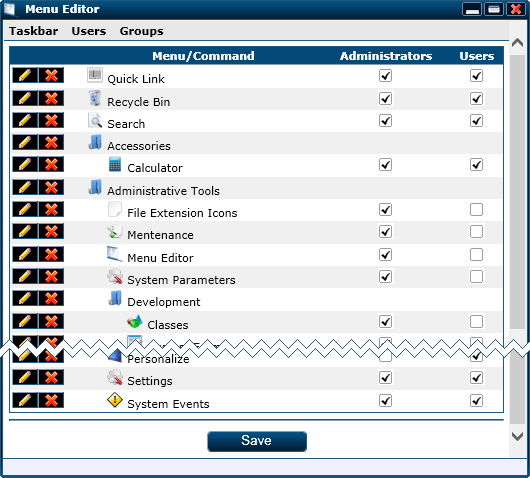
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int | The primary key value for the class. |
| SearchCriteria | String | The search criteria encoded in base 16. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/MappingObject/Page\_Login2.resx | Resources |
| App\_GlobalResources/Pages/MappingObject/Page\_Login2.RO-RO.resx | Resources |

#### Menu

This web page offers a spread sheet of taskbar menu commands and user groups in order to grant and revoke access to user groups over taskbar commands.



A Menu web page cut-out (due to the size)

As you can see in the image above, the taskbar menu tree structure is displayed and for each command or menu we have the following options:

* () used to trigger an update operation for the associated menu or command.
* () used to trigger a delete operation for the associated menu or command.

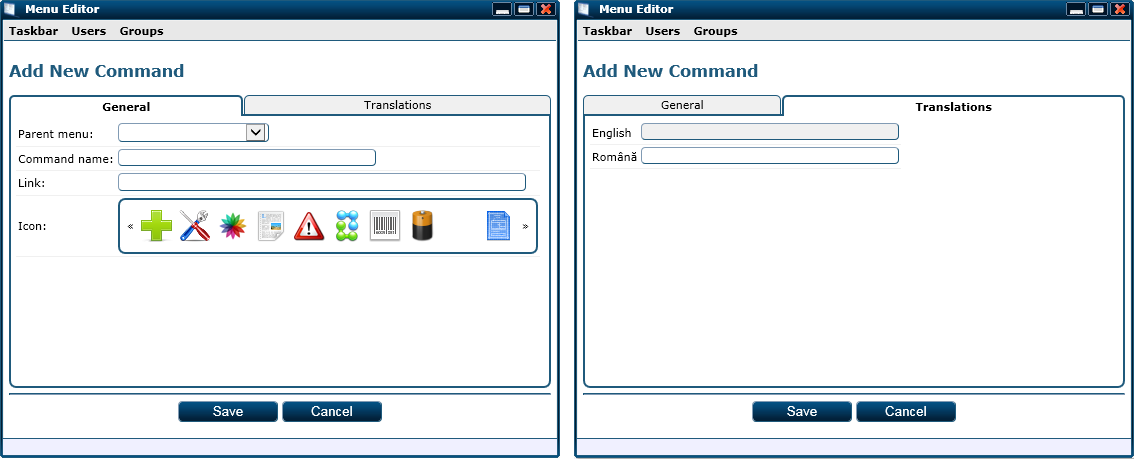
Also, the security groups defined on the platform are added to this table as columns and the intersection between a command’s row and a security group column will be a check box that shows if the security group is allowed to see the command in the taskbar menu or not.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Menu/Page\_Menu.resx | Resources |
| App\_GlobalResources/Pages/Menu/Page\_Menu.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Menu.js | Script |
| Scripts/Resources/RO-RO/Page\_Menu.js | Script |

#### Menu\_InsertCommand

This web page is design to facilitate adding new commands to the taskbar tree menu structure. As you can see in the image bellow, the information regarding a menu command is found on the first section of this page and the adjacent language tags are placed on the second section of the web page.



The menu command insert web page

The fields displayed on the first section are as follows:

* Parent menu: the menu from the taskbar tree structure in which the command will be placed.
* Command name: the command’s title.
* Link: the http path for the web page that must be opened when the command is triggered.
* Icon: the pictogram associated with the command.

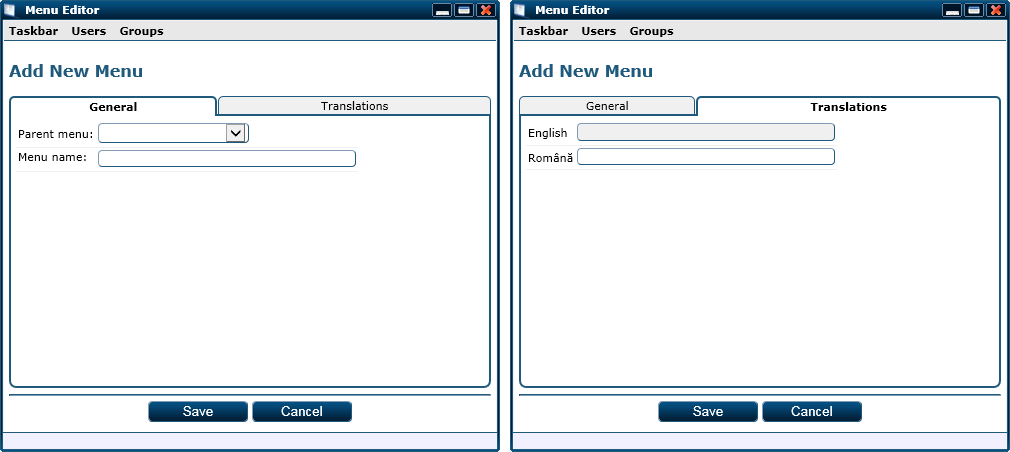
On the second section we have a language tag box displayed for each language supported by the platform. Automaticly, the language tag for English is read-only because it will be the same with the command’s title.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Menu\_InsertCommand/Page\_Menu\_InsertCommand.resx | Resources |
| App\_GlobalResources/Pages/Menu\_InsertCommand/Page\_Menu\_InsertCommand.RO-RO.resx | Resources |

#### Menu\_InsertMenu

This web page is design to facilitate adding new menu to the taskbar tree menu structure. As you can see in the image bellow, the information regarding a menu is found on the first section of this page and the adjacent language tags are placed on the second section of the web page.



The menu insert web page

The fields displayed on the first section are as follows:

* Parent menu: the menu from the taskbar tree structure in which the menu will be placed.
* Menu name: the menu’s title.

On the second section we have a language tag box displayed for each language supported by the platform. Automaticly, the language tag for English is read-only because it will be the same with the menu’s title.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Menu\_InsertMenu/Page\_Menu\_InsertMenu.resx | Resources |
| App\_GlobalResources/Pages/Menu\_InsertMenu/Page\_Menu\_InsertMenu.RO-RO.resx | Resources |

#### Menu\_UpdateCommand

This web page helps the user modify an existing taskbar menu command. Its look and functionality are the same as the Menu\_InsertCommand web page so we won’t go into details here.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| CommandID | Int | The primary key value for the command. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Menu\_UpdateCommand/Page\_Menu\_UpdateCommand.resx | Resources |
| App\_GlobalResources/Pages/Menu\_UpdateCommand/Page\_Menu\_UpdateCommand.RO-RO.resx | Resources |

#### Menu\_UpdateMenu

This web page helps the user modify an existing taskbar menu. Its look and functionality are the same as the Menu\_InsertMenu web page so we won’t go into details here.

##### Parameters

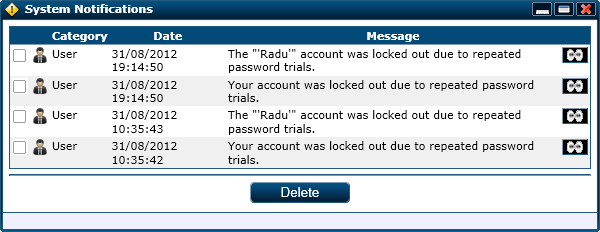
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| MenuID | Int | The primary key value for the menu. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Menu\_UpdateMenu/Page\_Menu\_UpdateMenu.resx | Resources |
| App\_GlobalResources/Pages/Menu\_UpdateMenu/Page\_Menu\_UpdateMenu.RO-RO.resx | Resources |

#### Notification

This web page displayes the system notifications destined for the current logged on user. The notification widget uses this page as well but it is also available as a taskbar option. The user can see this way all of his notifications and delete them selectively.



The notifications web page

As functionalities, the user has the following:

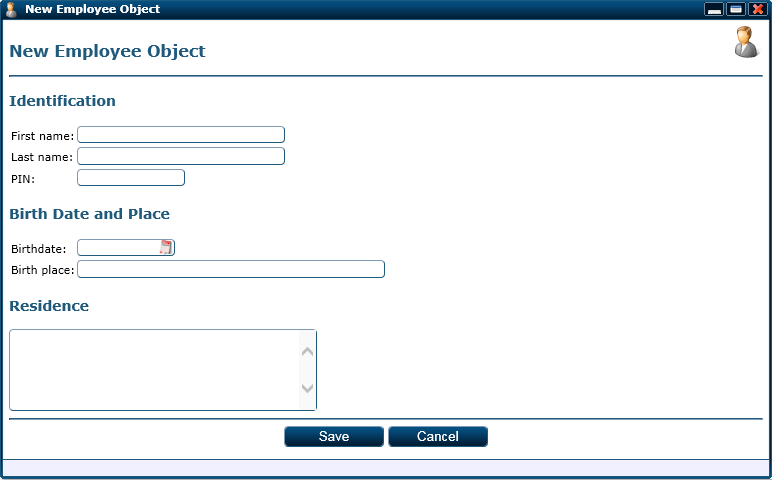
* Select some notifications and bulk delete them by clicking on the bottom “Delete” button
* Open a web page that displays the affected platform item (if any) by clicking on a notification associated () button.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Notification/Page\_Notification.resx | Resources |
| App\_GlobalResources/Pages/Notification/Page\_Notification.RO-RO.resx | Resources |

#### Object\_Insert

This web page is used to insert any object for any class in the database. The aparence for this web page changes according to the single-look description provided for the object’s class so we are not able to provide a full image for the page, however we will go into some detail to explain some of the functions of this web page and give an image based on an example class.



The object insert web page displayed for the Employee class

As said, this page displayes the data collecting form for any kind of object. The options available to the user on this web page are as follows:

* Save: triggers a save operation for the filled in object data.
* Cancel: closes the form without saving the filled in object data.
* Help: opens a new window that displayes the article from the documentation module associated with the object’s class.

WARNING: If the new object requires a parent object and if a parent object primary key value is not passed through the ParentID parameter, an additional combo box will be placed on this page forcing the user to manualy select a parent object. In case that the class has attributes that have default values depending on the parent object or are computed based on the parent object keep in mind that those formulas won’t be computed properly and may cause abnormal data in your product.

##### Parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int | The primary key value for the object’s class. |
| ParentID | Int | The primary key value for the object’s parent object. |

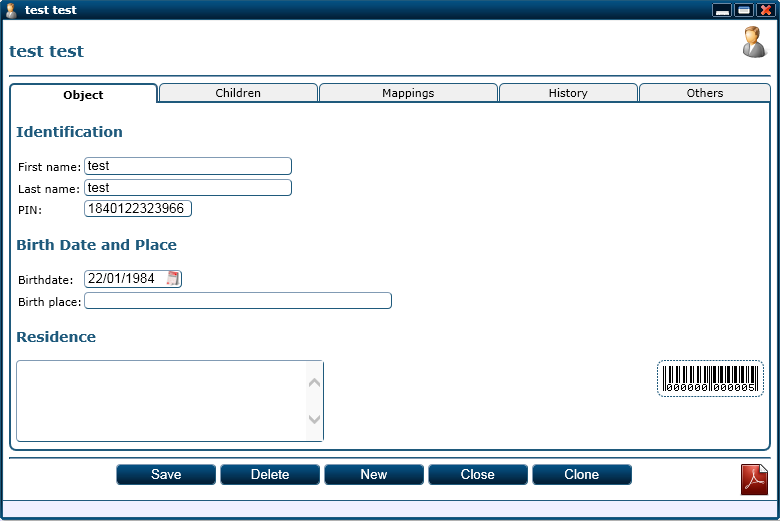
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Object\_Insert/Page\_Object\_Insert.resx | Resources |
| App\_GlobalResources/Pages/Object\_Insert/Page\_Object\_Insert.RO-RO.resx | Resources |

#### Object\_Update

This web page is used to update an object of any class so, similar, its display is based on the single-look description given for the class but we also have some additional sections based on the mappings defined for the class and other stuff. We will illustrate this web page based on an existing object of the Employee sample class.

NOTE: In order to illustrate this page we altered the Employee class in order to have all the sections shown. This being said, some relations do not make sence.



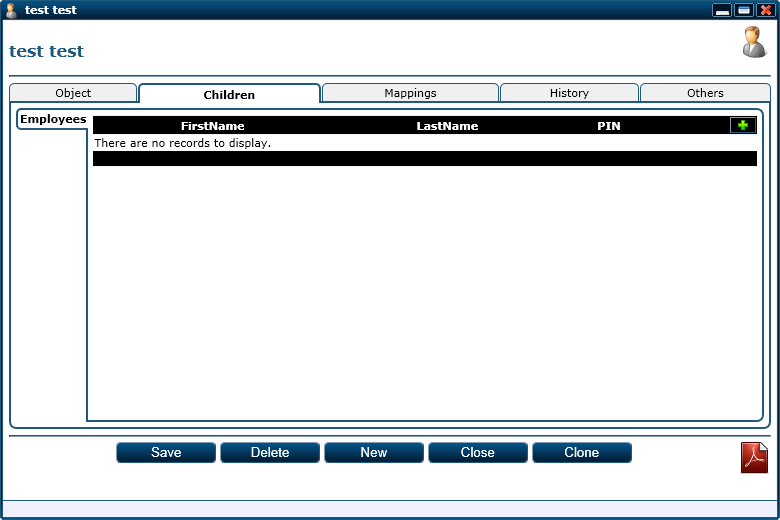
The first section for the object update web page displayed for a object of the Employee class

As you can observe, the first section for this web page is similar to the Object\_Insert web page displayed for the same class of the object displayed here. A small difference here is that for an existing object, the platform will display the barcode associated with the object on the right side of the form. On top of this, the options offered are the following:

* Save: triggers the saving operation for the displayed object’s data.
* Delete: triggers a delete operation for the displayed object.
* New: triggers the insert of a new object of the same class as the one displayed with the same parent object as the one displayed.
* Close: closes the displayed window.
* Clone: duplicates the displayed object and opens the update form for the cloned object.
* Help: opens a new window that displayes the article from the documentation module associated with the object’s class.

A PDF icon is displayed on the lower right corner and by clicking on it, the displayed object form will be exported into a PDF document and the user will be prompted with the document download.

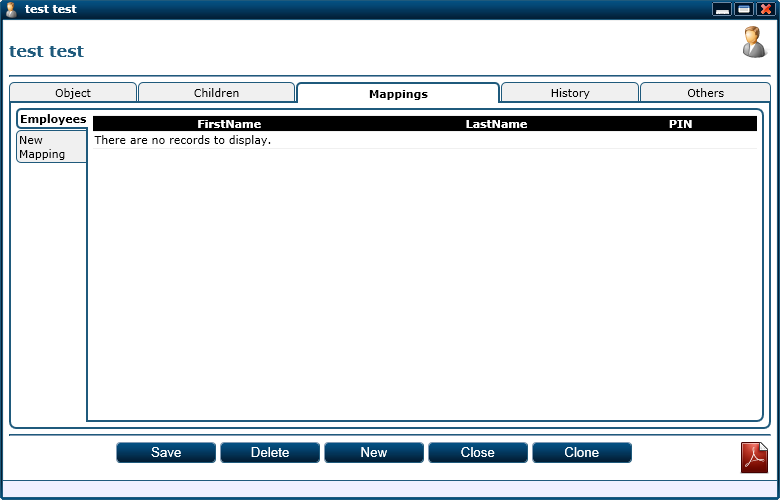
The second section for this web page consists in the subordered objects. For illustrating sample we made the Employee class its own slave class so we can suborder employee objects to other employee objects.



The second section of the object update page that illustrates the Employee-Employee master-slave relation

For each class-to-class relation where the class of the display object is the master class, a side panel will be displayed on this section named after the slave class in the relation. On each panel we will find the listing of objects of the associated class subordered to the displayed object. A new subordered object of the class associated with the panel may be added by clicking on the () button found on the upper right corner of the object listing. For each listed object a () button will be displayed. This button will open a new window that will display the Object\_update web page for the associated button object.

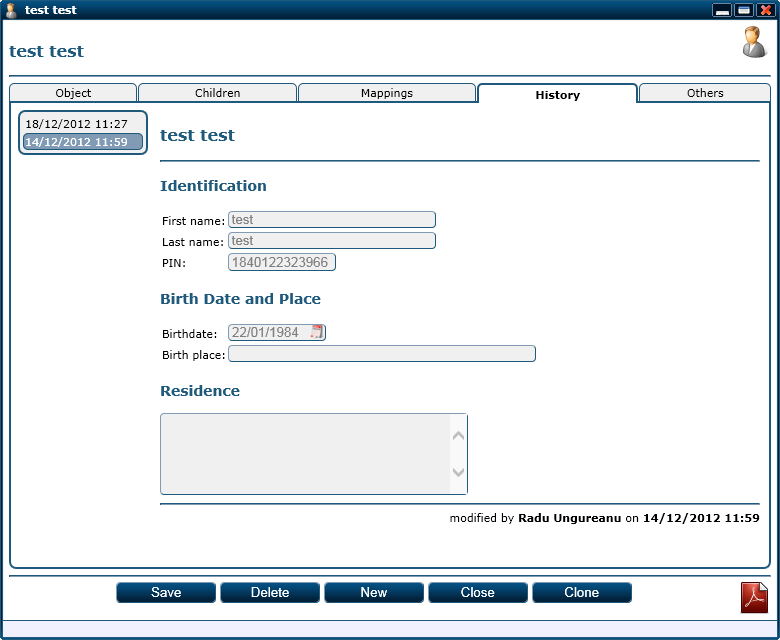
NOTE: This section may be missing if there are no master-slave relations defined that have the displayed object’s class as master.



The third section of the object update page that illustrates the Employee-Employee mapping relation

For the third section we have a side panel for each mapping relation that has the displayed object’s class as the source class named after the target class and an extra panel that helps to add a new mapping for the displayed object. Each class panel will display a list of the map objects and for each object there will be an associated () button to open a window and see the details regarding the object. For the “New Mapping” panel you can find the details in the chapter discussing the MappingObject form.

NOTE: If there are no mapping relations defined that have the displayed object’s class as source, this section will not be displayed.



The forth section of the object update page that illustrates the displayed object’s history

In the forth section we have a left side menu that shows all the moments when the displayed object was altered. By selecting one of these altered moments, we get the displayed object’s look at that point in time on the right side. At the bottom of the log display we get the name of the user that modified the object and the moment in time.

The last section, named “Others”, is used to display additional information for the displayed object. At the moment the two informations displayed are:

* Change parent: if the displayed object supports a parent object, this information is displayed to give the user the posibility to reassign the object to another master object.
* Breadcrumbs: this displayes the chain of parent objects for the displayed object until the displayed object.



The fifth section of the object update page

##### Parameters

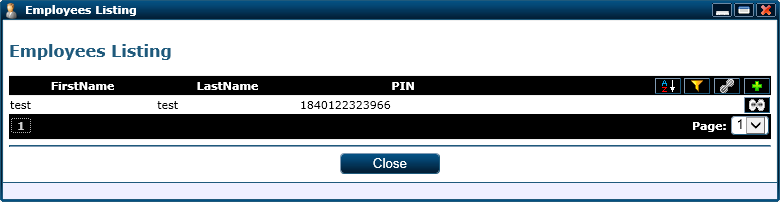
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ObjectID | Int | The primary key value for the displayed object. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Object\_Update/Page\_Object\_Update.resx | Resources |
| App\_GlobalResources/Pages/Object\_Update/Page\_Object\_Update.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Object\_Update.js | Script |
| Scripts/Resources/RO-RO/Page\_Object\_Update.js | Script |
| ObjectLog.aspx | Web page |

#### ObjectList

This web page is used to list objects of the same class. The functionality and display are dependent of the multi-look description provided for the class.



The object listing web page

The posible options available on this web page are related to the object table displayed and they are as follows:

* () toggles the sorting header and it is displayed only if the sorting header is specified in the class multi-look description.
* () toggles the filtering header and it is displayed only if the filtering header is specified in the class multi-look description.
* () toggles the mapping header and it is displayed only if there are mapping relations defined on the listing class.
* () triggers an add operation for an object of the listing’s class.
* () opens the associate object into a new window.

The listing has a pagination made by default. For the above sample it shows a single page but the pagination part won’t be displayed if there is a single page. The pagination is in two sections: quick links buttons relative to the current displayed page and a drop down for selecting any page available.

##### Parameters

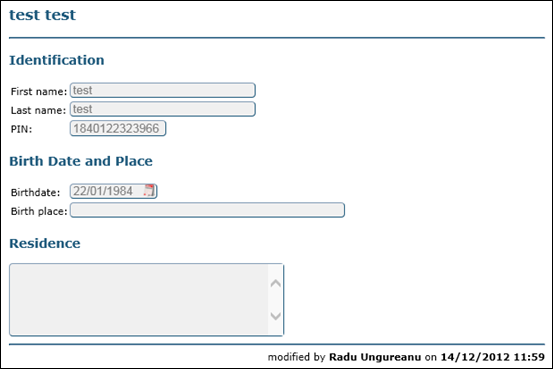
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int | The primary key value for the class. |
| ParentID | Int (optional) | The parent object for the objects listed. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ObjectList/Page\_ObjectList.resx | Resources |
| App\_GlobalResources/Pages/ObjectList/Page\_ObjectList.RO-RO.resx | Resources |

#### ObjectLog

This web page is used by the Object\_Update web page to display an object’s log instance. This web page is displayed according to the single-look description provided for the displayed object’s class.



The object log web page display for an Employee class object

##### Parameters

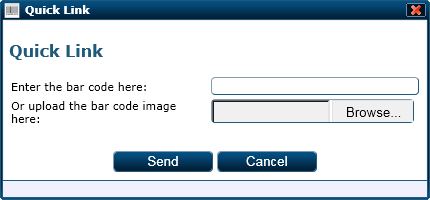
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| LogID | Int | The primary key value for the object’s log isntance. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ObjectLog/Page\_ObjectLog.resx | Resources |
| App\_GlobalResources/Pages/ObjectLog/Page\_ObjectLog.RO-RO.resx | Resources |

#### QuickLink

This web page is used by the shortcut widget in order to quickly open object pages. The functionality for this web page is rather simple; the fields displayed are as follows:



The quick link web page

* Enter the bar code here: the object’s bar code can be entered here by keyboard or scanner and the object for the corresponding bar code will be opened when the web page is posted back.
* Or upload the bar code image here: the bar code image can be uploaded and the platform will read the object code from it.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/QuickLink/Page\_QuickLink.resx | Resources |
| App\_GlobalResources/Pages/QuickLink/Page\_QuickLink.RO-RO.resx | Resources |

#### Recycle

This web page is destined to give the users the posibility to undelete objects. Deleted objects leave their logs behind so they can be restored to their last known instance.



The recyle bin web page

For the above image we had two employees objects deleted and their data coincide so ignore the duplicate data. The deleted objects are grouped by class in panels and listed with the available option for the restoration. The options available for an object restoration are the following:

* () simply restores the object if it doesn’t have a parent object.
* () restores the object along with its parent object chain.
* () restores the object along with its subordered object if the object doesn’t have a parent object.
* () restores the object along with its subordered objects and its parent object chain.

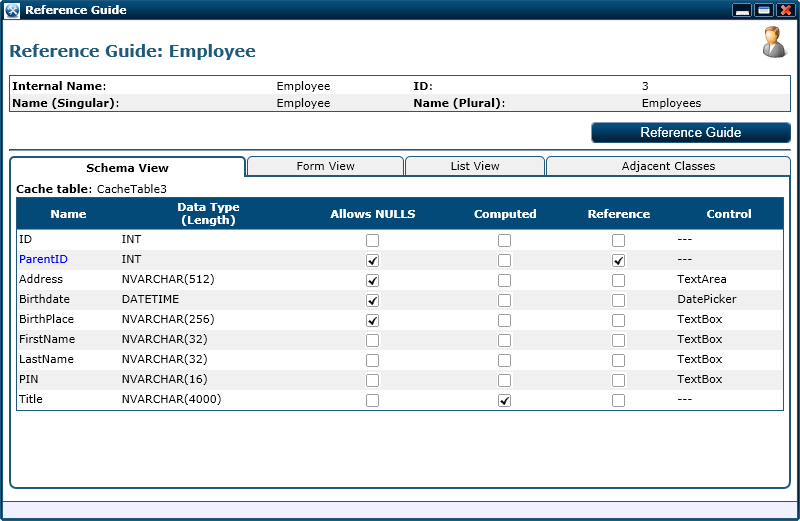
##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Recycle/Page\_Recycle.resx | Resources |
| App\_GlobalResources/Pages/Recycle/Page\_Recycle.RO-RO.resx | Resources |

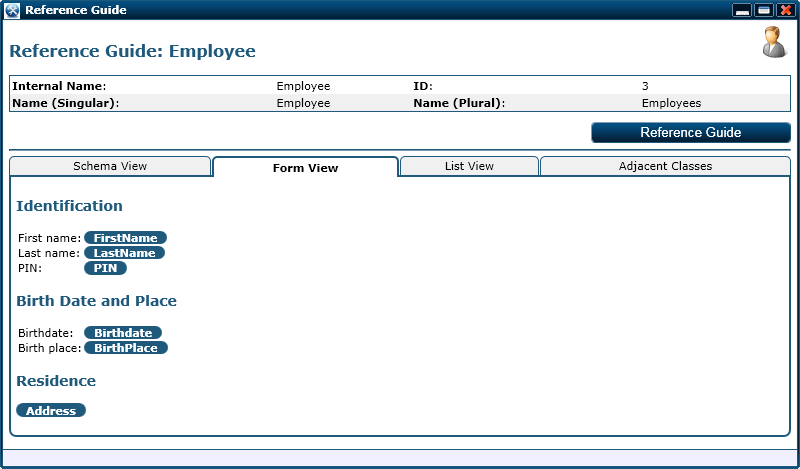
#### RefGuide\_Class

This web page displayes the information regarding a class and its main use is to provide essential information for specific query builds. The information is devided into four sections:

* Schema view: shows the attributes for the given class and foreign key attributes activate context menues to help the user navigate to the refered classes.
* Form view: displays the form for object data collecting marking the places for the data collecting fields.
* List view: displays the way an object listing row will look like marking the places for the attributes’ values.
* Adjacent classes: a list of classes that are related with the displayed class by a master-slave relation or a mapping relation.



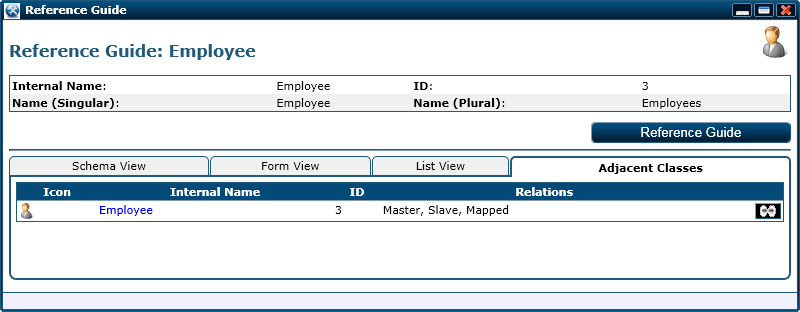
The first section of the reference guide class web page



The second section of the reference guide class web page



The third section of the reference guide class web page



The forth section of the reference guide class web page

##### Parameters

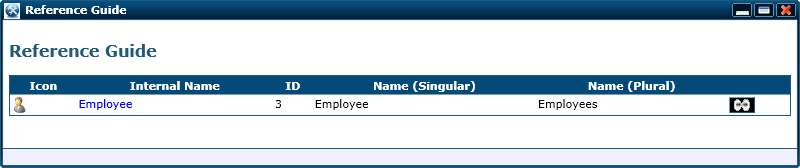
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ClassID | Int | The primary key value for the class. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/RefGuide\_Class/Page\_RefGuide\_Class.resx | Resources |
| App\_GlobalResources/Pages/RefGuide\_Class/Page\_RefGuide\_Class.RO-RO.resx | Resources |

#### RefGuide\_ClassList

This is a simple web page that displays the list of classes registered with minimal information needed in order to identify the class. As you can see in the image bellow, for each class, a () button is displayed on the right in order to open the window that details the class.



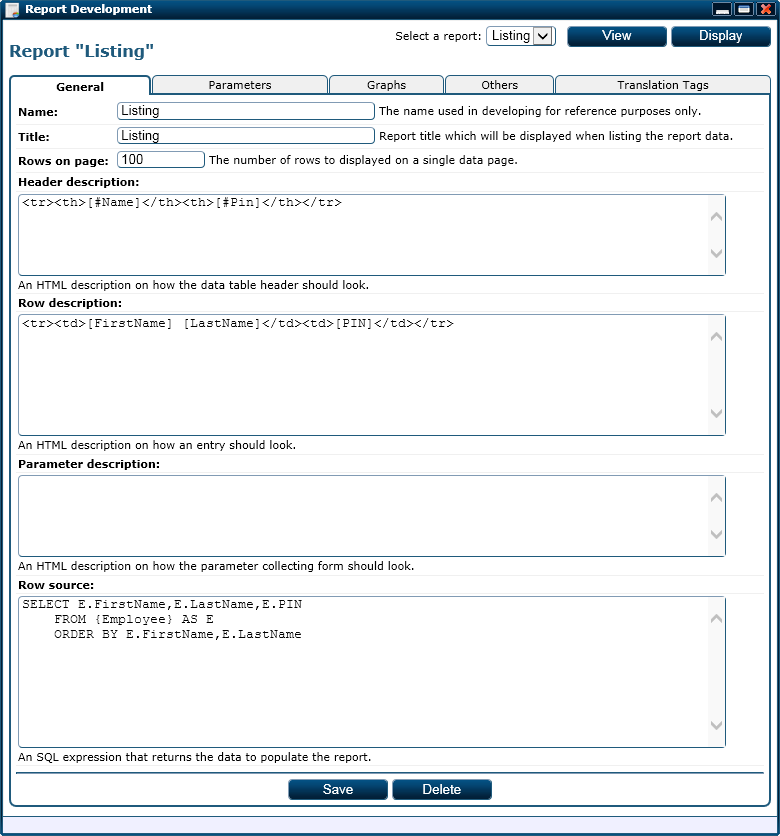
The reference guide class list web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/RefGuide\_ClassList/Page\_RefGuide\_ClassList.resx | Resources |
| App\_GlobalResources/Pages/RefGuide\_ClassList/Page\_RefGuide\_ClassList.RO-RO.resx | Resources |

#### Report

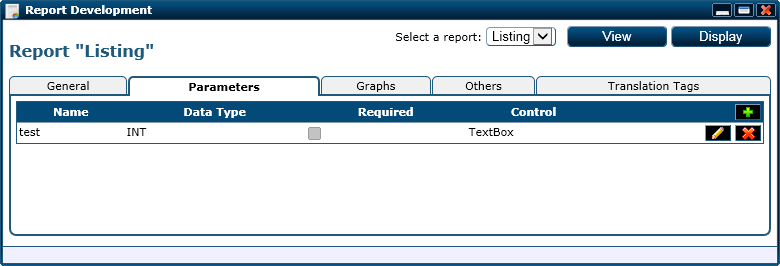
This web page is used to alter an existing report development wise. From the select box found in the upper right corner we can select a new report to modify. The information for a report is structured in five sections detailed bellow.



The first section of the report web page

The fields displayed on the first section are the following:

* Name: the report’s name used in development.
* Title: a specific string that may contain parameter tokens and language tokens representing the report’s title.
* Rows on page: the number of result entries per page.
* Header description: a specific HTML string representing the report result header.
* Row description: a specific HTML string representing a report result entry.
* Parameter description: a specific HTML string representing the parapeter form.
* Row source: a specific SQL query that returns the data entries for the report.



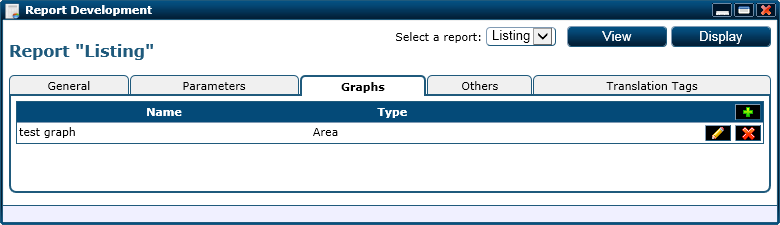
The second section of the report web page

The second section is a list of parameters for the report’s query. The columns for the displayed parameters are as follows:

* Name: the parameter’s name.
* Data Type: the data type for the parameter and the data length (if it apllyes).
* Required: marks the parameter as being required.
* Control: the control through which the parameter value will be collected.

The options available here are the following:

* () triggers a new parameter add.
* () triggers a modify operation for the associated parameter.
* () triggers the deletion of the associated parameter.



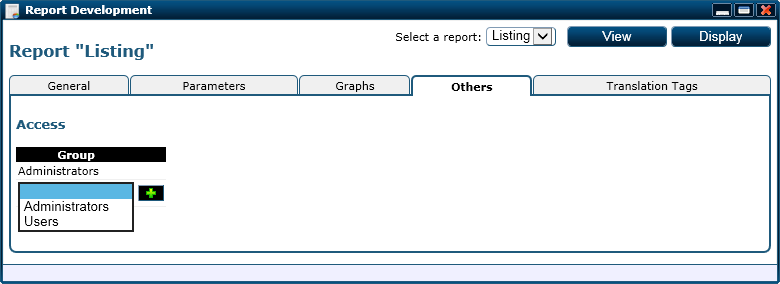
The third section of the report web page

This third section is dedidated to the charts/graphs attached to the report. The columns displayed for the chart list are:

* Name: representing the name of the graph for internal and user display list.
* Type: the graph/chart type.

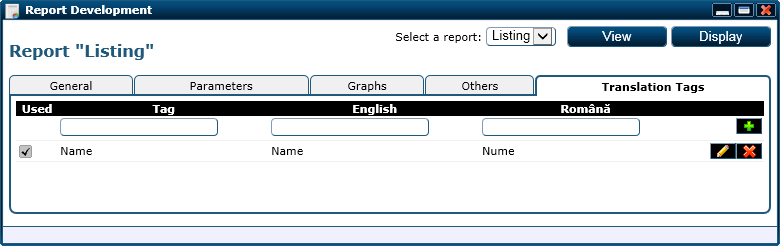
The options available in this section are:

* () triggers a new chart add operation.
* () triggers a modify operation for the associated chart.
* () triggers the deletion of the associated chart.



The forth section of the report web page

The forth section is momentarly just for the report and charts access. Any group can be added to this list to grant access to the report and subordered charts and by default the “Administrators” group has access and its access can’t be revoked.



The fifth section of the report web page

The fifth section is destined to the report and charts adjacent language tags. This section is similar to the class language tag section and custom form language tag section so we won’t detail it any further.

##### Parameters

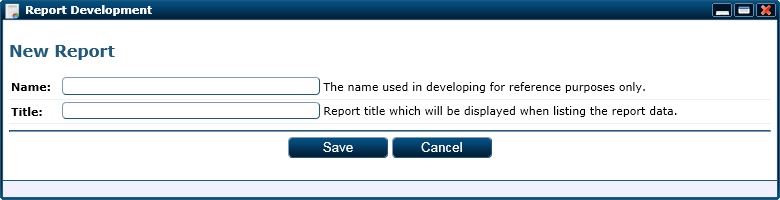
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ReportID | Int (optional) | The primary key value for the report. If it is missing, the first available report will be displayed. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Report/Page\_Report.resx | Resources |
| App\_GlobalResources/Pages/Report/Page\_Report.RO-RO.resx | Resources |
| Images/Pages/Report/Used.png | Image |
| Images/Pages/Report/Unused.png | Image |
| Scripts/Resources/EN-UK/Pae\_Report.js | Script |
| Scripts/Resources/RO-RO/Pae\_Report.js | Script |

#### Report\_Insert

This web page is used to add a new report to the report collection with a minimum of required information. The rest of the information regarding the report can be added after the insert operation.



The report insert web page

The fields displayed on this web page are:

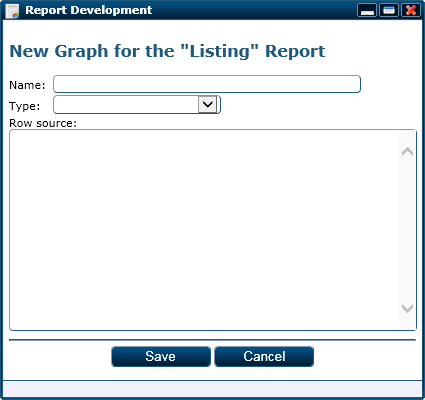
* Name: the name of the report used internaly for development.
* Title: a minimal title for the report which will be displayed to the user.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Report\_Insert/Page\_Report\_Insert.resx | Resources |
| App\_GlobalResources/Pages/Report\_Insert/Page\_Report\_Insert.RO-RO.resx | Resources |

#### Report\_InsertGraph

This web page is used to insert a new graph/chart and suborder it to a report.



The report graph insert web page

The fields displayed and required are:

* Name: the name for the chart used internaly and for display.
* Type: the graphs type. This was detailed in the “Graph Objects” chapter.

##### Parameters

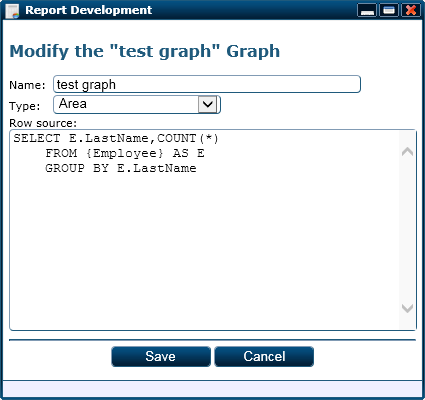
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ReportID | Int | The primary key value for the parent report. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Report\_InsertGraph/Page\_Report\_InsertGraph.resx | Resources |
| App\_GlobalResources/Pages/Report\_InsertGraph/Page\_Report\_InsertGraph.RO-RO.resx | Resources |

#### Report\_UpdateGraph

This web page is used to update and existint report graph and it is similar with the Report\_InsertGraph web page sa we won’t detail it further.



The report update graph web page

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| GraphID | Int | The primary key value for the parent report’s graph. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Report\_UpdateGraph/Page\_Report\_UpdateGraph.resx | Resources |
| App\_GlobalResources/Pages/Report\_UpdateGraph/Page\_Report\_UpdateGraph.RO-RO.resx | Resources |

#### ReportGraphView

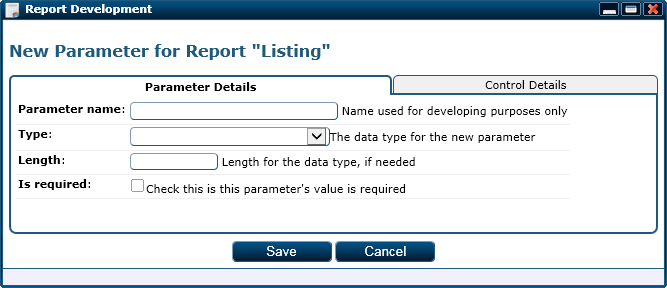
This web page displayes a graph/chart associated with a report. Due to the fact that its display is based on the graph type we won’t be able to give a display for this web page but you may check out the “Graph Objects” chapter to get a better idea on what to expect from this web page.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| GraphID | Int | The primary key value for the graph. |

#### ReportParameter\_Insert

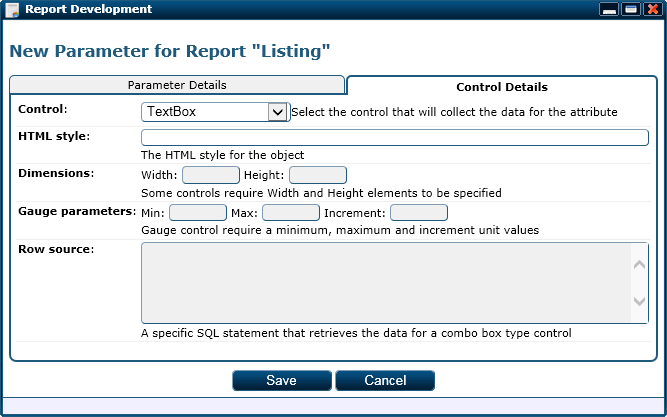
This web page is used to insert a new parameter for a report. The information regarding the parameter is split into two sections just like the information for a new class attribute.



The first section of the report parameter insert web page

The fields displayed for the first section displays the following fields:

* Parameter name: the name of the parameter used internaly.
* Type: the parameter’s value data type.
* Length: some data types require a length to be specified. The length may be an integer between 1 and 4000 or the string “MAX”.
* Is required: marks the parameter as being required for the data processing.



The second section of the report parameter insert web page

The fields displayed on this section are related to the control through which the parameter’s value will be collected and they are the following:

* Control: the control name for the control through which the parameter’s value will be collected.
* HTML style: the HTML style for the control. Some controls can be altered through this field in order to change their apparence. Example: for a text box field the size may be increased by filling in “width: 300px;” in this field.
* Width: some controls do not support HTML styles and their width can’t be altered that way so the width must be specified this way. Example: the image upload control requires the width of the displayed picture to be specified through this field.
* Height: similar to the width field but is used for height.
* Min: represents the minimum value for the attribute’s value. Example: the gauge control selects a floating value inside a range; this field specifies the lower bound of the range.
* Max: similar to the min field but used for a maximum field value.
* Unit: represents the rounding step for the collected value. Example: for a gauge control the chosen float number will be rounded to the specified incremental values.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ReportID | Int | The primary key value for the parent report. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ReportParameter\_Insert/Page\_ReportParameter\_Insert.resx | Resources |
| App\_GlobalResources/Pages/ReportParameter\_Insert/Page\_ReportParameter\_Insert.RO-RO.resx | Resources |

#### ReportParameter\_Update

This web page is used to alter a report parameter and it has similar functionality and field with the ReportParameter\_Insert web page so we won’t go into details.

##### Parameter

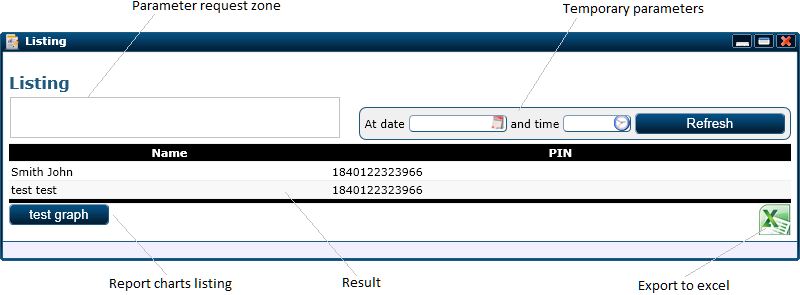
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ParamID | Int | The primary key value for the report parameter. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ReportParameter\_Update/Page\_ReportParameter\_Update.resx | Resources |
| App\_GlobalResources/Pages/ReportParameter\_Update/Page\_ReportParameter\_Update.RO-RO.resx | Resources |

#### ReportView

This web page is is used to display the data for a report. The overall look of this web page depends on the report displayed but to illustrate the common parts we will provide a display for a sample report.



The report view web page shown for a simple report

Any parameters required for the report will be shown in the marked area found on the upper left corner of the window. The “Refresh” button is grouped together with the temporary parameters. By filling in the temporary parameters the result is reprocess to reflet the situation at that given moment in time.

The list of subordered graphs is displayed at the bottom on the result table (bottom left). In the bottom right corner you can see the excel export button which will trigger a CSV file generation with the result and a download for the generated file.

##### Parameters

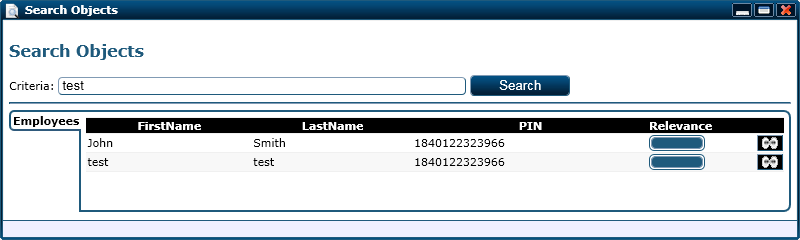
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| ReportID | Int | The primary key value for the report. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/ReportView/Page\_ReportView.resx | Resources |
| App\_GlobalResources/Pages/ReportView/Page\_ReportView.RO-RO.resx | Resources |

#### Search

This web page is used to display objects that correspond to a given search criteria. The objects are listed in side panels grouped according to their corresponding classes and sorted by relevance as you can see in the image bellow.



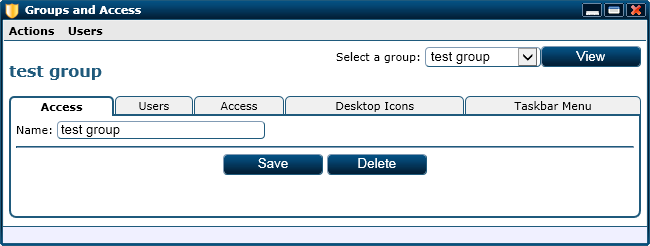
The search web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Search/Page\_Search.resx | Resources |
| App\_GlobalResources/Pages/Search/Page\_Search.RO-RO.resx | Resources |

#### Security

This page is the main point in mamaging security groups. The information regarding security groups is split into the five sections illustrated bellow.



The first section of the security web page

The only information displayed here is the security group name.

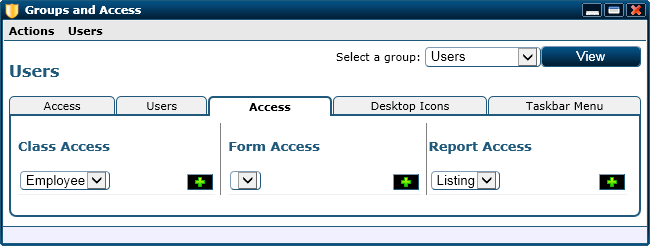
NOTE: The two options (Save and Delete) are displayed only when the displayed group is not the “Administrators” group or “Users” group.



The second section of the security web page

The second section represents the list of users that are part of the displayed security group. A user can be removed from the security group displayed by clicking on the () button and a user can be added using the mini-form found at the bottom on the listing.

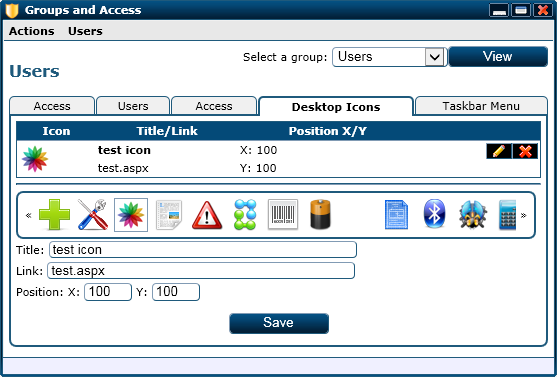
NOTE: the user with the primary key value of “1” can’t be deleted or removed from any security group.



The third section of the security web page

The third section is dedicated to the security group access over the platform information. The platform information can be accessed through class objects (Class access left panel), custom forms (Form access center panel) and reports (Report access right section). Access on a section can be granted by selecting the object and clicking the () button. To remove access to an item we can use the () button.

NOTE: The “Administrators” security group has access to all items on the platform and his access can’t be revoked.

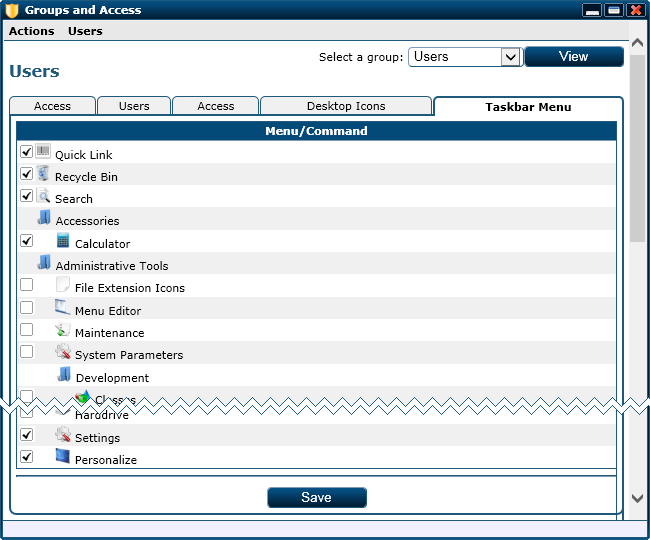


The forth section of the security web page

The forth section for this web page determines the default desktop icons for the displayed security group. If a new user is added to this security group, that user’s desktop will be populated with the default icons. A new icon can be added using the form at the bottom of the listing and using the same form an existing icon can be altered but we have to trigger the () button first. An existing icon can also be removed by triggering the () button.

The fields for defining a desktop item are the following:

* Title: the icon’s title.
* Link: the http path that will be loaded by a new window.
* Position X: the position in pixels from the left margin.
* Position Y: the position in pixels from the top margin.



The fifth section of the security web page

The fifth section is design to grant access on taskbar menu command for the taskbar menu. As you can see, this page displays the taskbar menu tree structure and on the left side we have check boxes for each tree menu command.

##### Parameters

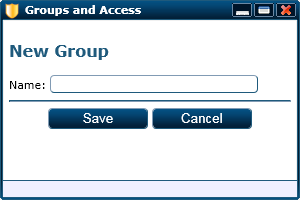
|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| GroupID | Int | The primary key value for the security group. |
| SelectedTab | Int (optional) | The ID of tab that must be activated after the web page load. |

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Security/Page\_Security.resx | Resources |
| App\_GlobalResources/Pages/Security/Page\_Security.RO-RO.resx | Resources |
| Scripts/Resources/EN-UK/Page\_Security.js | Script |
| Scripts/Resources/RO-RO/Page\_Security.js | Script |

#### Security\_InsertGroup

This web page is used to create a new security group. Users, access rights, default desktop icons and menu access will be added after the creation step.



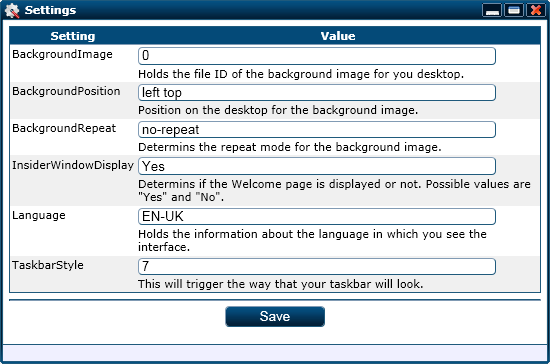
The security group insert web page

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Security\_Insert/Page\_Security\_Insert.resx | Resources |
| App\_GlobalResources/Pages/Security\_Insert/Page\_Security\_Insert.RO-RO.resx | Resources |

#### Settings

This web page is used to help users set the platform’s parameters regarding their own environment to the wanted values. All of the settings displayed on this web page can be set individualy from other web pages.



The settings web page

Each setting is explained on the form so we won’t go over the details.

##### Adjacent Files and Folders

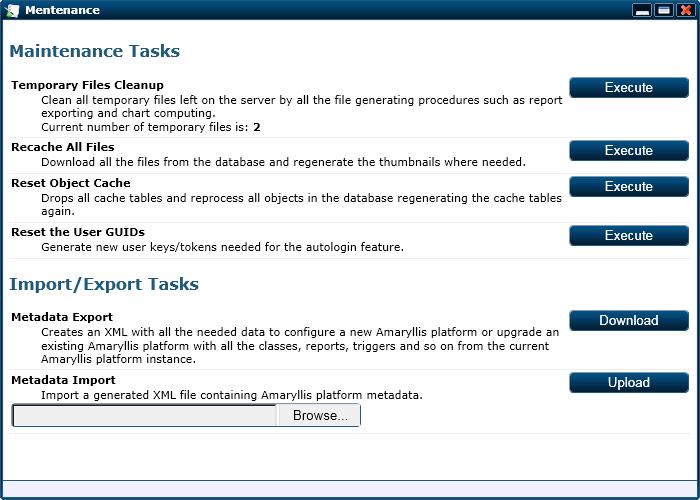
|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Settings/Page\_Settings.resx | Resources |
| App\_GlobalResources/Pages/Settings/Page\_Settings.RO-RO.resx | Resources |

#### StopImpersonation

This web page is triggered to sotp a user impersonation. It has no display and everytime we redirect to it it will check if an inpersonation operation is in progress. If so, it will stop the impersonation operation and either way it will redirect to the default web page.

#### Task

This web page is made for administrative use and it helps the user run a few administrative tasks as well as importing and exporting the platform’s metadata.



The task web page

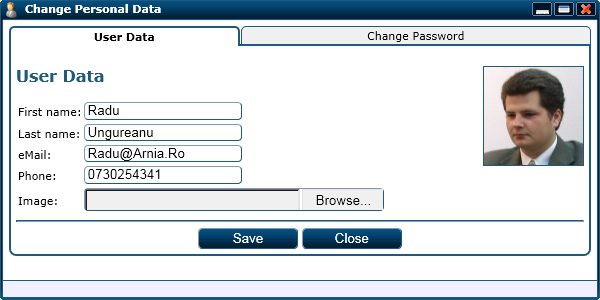
The tasks are explained on the web page and the as well as the metadata import/export operations so we won’t detail them here.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/Task/Page\_Task.resx | Resources |
| App\_GlobalResources/Pages/Task/Page\_Task.RO-RO.resx | Resources |

#### UserChangeData

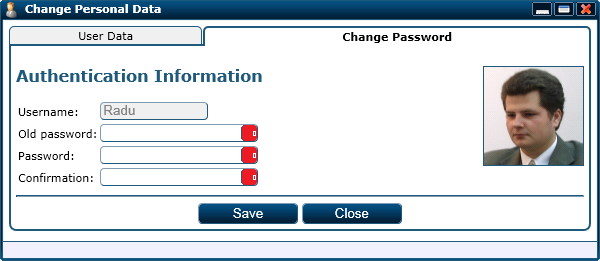
This is a simple web page designed for all the users to help them change their data and password. The information for this web page is split into two sections illustrated bellow.



The first section of the user change data web page

The fields displayed for the first section of this web page are the following:

* First name: the user’s first name.
* Last name: the user’s last name.
* eMail: the user’s e-mail address or addresses separated by the “;” character. This is used for e-mail alerts and notifications.
* Phone: the user’s cell phone number. This is used for SMS alerts and notifications if the platform is integrated with a Web2SMS service.
* Image: the user’s representative image.



The second section of the user change data web page

For the second section, the web page displays the following fields:

* Username: readonly field shown just for information purposes.
* Old password: the old user password, the password won’t be changed if this field is not filled in.
* Password: the new password.
* Confirmation: the new password retyped to assure that there are no mistakes in the previous field.

##### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Pages/UserChangeData/Page\_UserChangeData.resx | Resources |
| App\_GlobalResources/Pages/UserChangeData/Page\_UserChangeData.RO-RO.resx | Resources |

#### UtilityCloseWindow

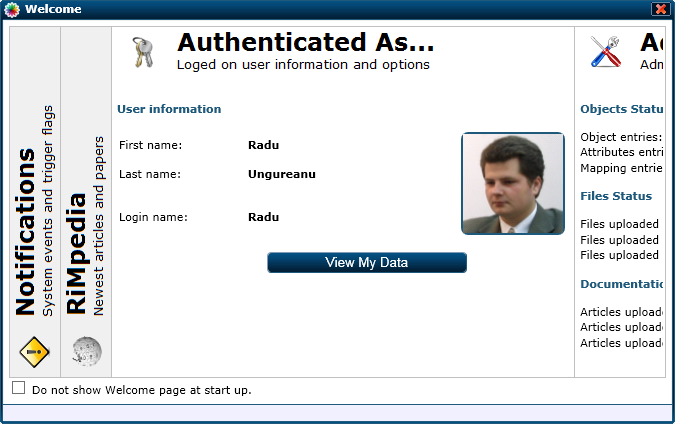
This web page is used only to close windows after a successful server side operation. If after a server side operation the window displaying the web page must be closed, we can simply redirect from the server’s side to this web page and it will close the window for us.

##### Parameters

|  |  |  |
| --- | --- | --- |
| Parameter | Type | Description |
| WithMainRefresh | Int (optional) | Triggers the refresh of the main display page along with the window close. |

#### Welcome

This web page is a welcoming page for the user and provides data regarding the latest activity on the platform.



The welcome web page

The tabs scroll periodicaly from one side to another and pauses on each of them. The user can click any tab in order to expand it and the scroll will pause more on it before resumimng the scrolling.

NOTE: Some tabs may be missing due to a user‘s inssuficient priviledges.

## Plain Interface

The plain interface is orientated designed for users that are not used to a more Windows like web interface.The design for this is simpler and is resembles more with a traditional web site.

### Programming Classes

#### Amaryllis.Objects Namespace

##### Page Class

This class is responsible for the web pages and window pages design. All of the methods for this class can be called similar to html tags just to keep a clear structure of the web pages. Bellow you can see the way to call the methods for a main web page and a window web page.

|  |  |
| --- | --- |
| <%Amaryllis.Objects.Page P = new Amaryllis.Objects.Page(); %>  <%P.BeginPage(); %>  <%P.EndPage(); %> | <%Amaryllis.Objects.Page P = new Amaryllis.Objects.Page(); %>  <%P.BeginWindow(); %>  <%P.EndWindow(); %> |
| Main page | Window page |

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |
| UserID | Int | The primary key value for the currently logged in user. |

###### Methods

**Page**

Purpose: (constructor) initializes the Response and UserID properties.

**BeginPage**

Purpose: writes the HTML code that begins the enclosing main web page.

Parameters: NONE

**EndPage**

Purpose: writes the HTML code that closes the enclosing main web page.

Parameters: NONE

**BeginWindow**

Purpose: writes the HTML code that begins the enclosing window web page.

Parameters: NONE

**EndWindow**

Purpose: writes the HTML code that closes the enclosing window web page.

Parameters: NONE

**DisplayErrors**

Purpose: writes the code that displays the encountered errors on the web page.

Parameters: NONE

**ThrowError**

Purpose: adds a new set of errors on the existing error stack.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Errors | String | The error messages separated by the “|” character. |

**PlacePageScripts**

Purpose: writes the HTML code in order to include the needed java scripts on the main web page.

Parameters: NONE

**PlaceWindowScripts**

Purpose: writes the HTML code in order to include the needed java scripts on the window web page.

Parameters: NONE

**PlaceResourceScript**

Purpose: writes the HTML code that includes the given resource java script on the web page for the currently interface language.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| FileName | String | The name of resource java script file that must be included on the web page. |

**PlaceResourcesPageScript**

Purpose: triggers the platform to include the proper resources java script file on the current web page.

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| App\_GlobalResources/Objects/Page/Objects\_Page.resx | Resource |
| App\_GlobalResources/Objects/Page/Objects\_Page.RO-RO.resx | Resource |
| Images/Objects/Page/Background.png | Image |
| Images/Objects/Page/Bullet.png | Image |
| Images/Objects/Page/FooterBar.png | Image |
| Images/Objects/Page/HeaderBar.png | Image |
| Scripts/Objects/Page.js | Script |

##### TreeMenu Class

This class is responsible for the tree structure displayed on the left section. The methods for this class are used internally so we will only concentrate on the properties and methods without showing an example on how to aggregate them.

###### Properties

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Response | System.Web.HttpResponse | The response object for the current http context. |
| Level | Int | Holds the hierarchy level for the tree menu. |
| Count | Int | Holds the option count and contributes to the created objects HTML IDs. |
| IgnoredCommands | String[] | An array of commands that don’t have to be displayed because the adjacent web pages are not part of this interface. |
| IgnoredMenus | String[] | An array of menus that don’t have to be displayed because their submenus and commands are not a part of this interface. |

###### Methods

**IgnoreCommand**

Purpose: determines if the specified command must be avoided.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title of the command that must be checked. |

**IgnoreMenu**

Purpose: determines if the specified menu must be avoided.

Returns: Boolean

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The title of the menu that must be checked. |

**TreeMenu**

Purpose: (constructor) initializes the properties.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Version | String (optional) | Specifies the tree menu version. |

**Reset**

Purpose: resets the Level property to zero.

Parameters: NONE

**BeginMenu**

Purpose: starts a new encapsulating menu object with the specified title.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The menu’s title. |

**AddCommand**

Purpose: adds a new command to the menu.

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The command’s title. |
| Icon | String (optional) | The relative HTTP path for the pictogram associated with the command. |
| Link | String | The HTTP path for the web page that must be opened when the command is triggered. |

**EndMenu**

Purpose: closes the current encapsulating menu.

Parameters: NONE

**DisplayMenuForUser**

Purpose: generates the HTML code for the menu adjacent to the current logged on user.

Parameters: NONE

###### Adjacent Files and Folders

|  |  |
| --- | --- |
| File and Path | Type |
| Scripts/Objects/TreeMenu.js | Script |

### Interface Elements

#### Header Bar

As describe above, we have the following items on the title bar.

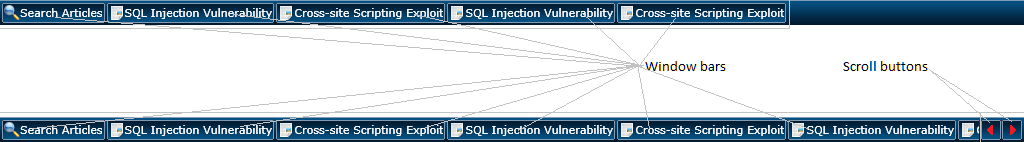


Header bar and elements

We have the logo in the left side of the header bar and on the right corner we have the widget buttons. All of these items have to fit on a 43 pixels height space.

#### Footer Bar

For this interface the footer bar acts similar to the taskbar from Windows. It displays the time and date on the right side and a start button on the left side. The center part is, as shown below, used to display the window bars for all the opened windows. If the window bars do not fit on the given space, scrolling buttons will be displayed.



Footer bar and elements

By clicking on a window bar will trigger all the window bars after the one clicked to be removed from the task bar and their adjacent windows to be closed.

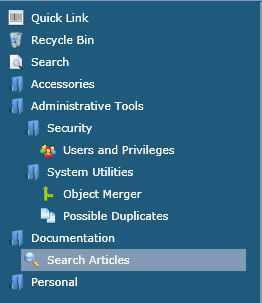
#### Content Section

The content section is a blank zone over which windows will be displayed on top of each other covering the whole content section. The order on which the windows are aopened determine the top window (the last opened is on top) and the user doesn’t have the posibility to switch between them.

By chosing an earlier opened window causes all the windows opened after that one to close and if a menu command is triggered, all the currently loaded windows will be closed.

#### Menu

The main menu for this interface is placed on the left side and it has a tree like structure as you can see in the image bellow. Intuitivly, a menu can be opened and closed by clicking on it and a command can be triggered by clicking on it.



Left side tree menu

#### Window Element

The only element for the content section is the window element which loads web pages. A window element consists in an iFrame object which loads a web page. The iFrame object covers the whole content section so the only functionality provided is given by the loaded web page.

### Widgets

#### Quick Link Widget

This is identical to the quick link widget from the Window interface so we won’t detail it again.

#### Notification Widget

This is identical to the notification widget from the Window interface so we won’t detail it again.

#### Zoom Widget

This is identical to the zoom widget from the Window interface so we won’t detail it again.

#### Language Widget

This is identical to the language widget from the Window interface so we won’t detail it again.

#### About Widget

This is identical to the about widget from the Window interface so we won’t detail it again.

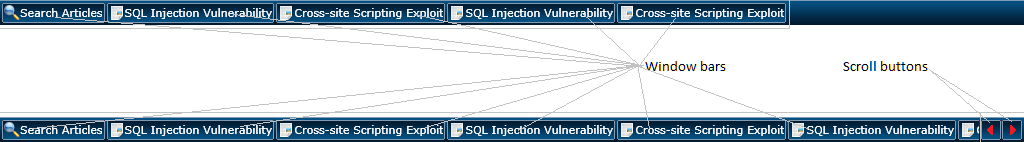
#### Impersonation Widget

This is identical to the impersonation widget from the Window interface so we won’t detail it again.

### Taskbar Section

#### Window Adjacent Buttons Zone

We will re-describe this part here as shown in the figure bellow. We have the window adjacent buttons displayed from the left to right. If the given space on the taskbar is full the scroll buttons will be displayed as shown in the figure bellow.



Task bar displays

### Java Scripts

All the general scripts used by the platform are stored in the “Scripts” folder and are grouped by categories.

#### Controls Folder

All of the scripts in this folder are identical with the scripts found in the same folder of the Window interface but they are duplicated here in order provide the needed functionality.

#### Core Folder

##### Animation

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Compat

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Error

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Utility

###### Methods

**utilityStringToBase16String**

Purpose: converts the given string into a hex encoded string based on characters ASCII codes.

Returns: string (representing the given string’s encoding

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Text | String | The string that must be encoded. |

#### Objects Folder

##### Menu

This script is adjacent to the menu class and helps the menu items animate on mouse actions.

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### MenuBar

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### ModalBox

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### OverDiv

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Page

This script is adjacent to the page class and it is used to mediate some backgroung tasks and mechanisms of interaction with the user.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| pageWindowCallBack | String | Holds a java script code sequence that must be runned when a displayed object is deactivated. |

###### Methods

**pageObjectActivate**

Purpose: marks the specified HTML object as being active and specifies the callback code for it.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| CallBack | String | The code sequence that is triggered when the active object must be deactivated. |

**pageObjectDeactivate**

Purpose: runs the callback code specified by the pageObjectActivate method.

Returns: NOTHING

Parameters: NONE

**pageInitialize**

Purpose: initializes the objects needed for the main web page.

Returns: NOTHING

Parameters: NONE

**pageOnResize**

Purpose: checks the page dimension and objects displayed when the display container is resized.

Returns: NOTHING

Parameters: NONE

##### SidePanel

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### TabPanel

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### TaskBar

This scriptis used to animate the task bar buttons associated with thw windows displayed on the content section.

###### Methods

**taskbarInitialize**

Purpose: initializes the HTML element needed by the task bar section.

Returns: NOTHING

Parameters: NONE

**taskbarScroll**

Purpose: scrolles the buttons displayed to left or right according to the specified step value.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Step | Int | The step value in pixels for the button list move animation. Negative values means scroll left and positive values means scroll right. |

**taskbarUnregisterWindow**

Purpose: removes the adjacent button for the specified window from the task bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |

**taskbarRegisterWindow**

Purpose: creates a new task bar button for the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the adjacent window. |

**taskbarCheckSize**

Purpose: resizes the taskbar space and recalibrates it according to the browser’s display surface.

Returns: NOTHING

Parameters: NONE

##### TreeMenu

This script is needed to animate the taskbar and taskbar menu for both Window 7 and Windows 8 styles.

###### Methods

**treemenuOnMouseOver**

Purpose: highlights the specified menu item.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command or menu. |

**treemenuOnMouseOut**

Purpose: dehighlights the specified menu item.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu command or menu. |

**treemenuMenuOnClick**

Purpose: collapses or expands the specified menu.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu. |

**treemenuCommandOnClick**

Purpose: opens a new window on the content section with the title and icon of the specified menu item and loads the web page associated with the specified menu item.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| ElementID | Int | The ID of the tree menu. |

##### Window

This script is a stand-alone script that is used to create window elements on the content section.

###### Variables

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| windowCount | Int | Retains the first available window element ID. |

###### Methods

**windowClearAll**

Purpose: closes all the opened windows.

Returns: NOTHING

Parameters: NONE

**windowSetContent**

Purpose: changes web page loaded by the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |
| Link | String | The http path for the new web page that must be loaded by the window. |

**windowActivate**

Purpose: activates the specified window.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowClose**

Purpose: removes the specified window from the content section and from the task bar.

Returns: NOTHING

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| WindowID | Int | The ID of the window element. |

**windowCheckPosition**

Purpose: verifies the positions and dimensions for all the windows displayed when the braowser is resized in order to keep the windows accessible to the user.

Returns: NOTHING

Parameters: NONE

**windowNewWindow**

Purpose: creates a new window on the content section.

Returns: Int (representing the new window’s ID)

Parameters:

|  |  |  |
| --- | --- | --- |
| Name | Type | Description |
| Title | String | The window’s title. |
| Icon | String | The relative http path for the pictogram associated with the window. |
| Link | String | The http link for the web page that will be loaded by the window. |
| PosX | Int | Left for script compatibility with the Window interface. |
| PosY | Int | Left for script compatibility with the Window interface. |
| Width | Int | Left for script compatibility with the Window interface. |
| Height | Int | Left for script compatibility with the Window interface. |

#### Resources Folder

This folder holds resource scripts used by controls, objects, web pages and widgets in order to cover the language issues. The resources scripts are loaded on web pages as needed and the variables contained in them are used in java scripts to display texts in the interface’s language.

The subfolders contained here are named according to the language codes for the languages supported by the interface. For a script file found in the EN-UK folder there should be files named identical in all the other language folders.

NOTE: If the interface is in a language for which a resource script is missing it will include the EN-UK script in its place.

At this point the interface has resource scripts for English (EN-UK) and Romanian (RO-RO). A new language can be added by using the adjacent translator application or manualy (adding resource files, resource scripts and database language tags).

#### Tools Folder

The scripts found in this folder are identical with the scripts found in the same folder from the Window interface so we won’t detail them again.

#### Widgets

The scripts from this folder are used by the widgets displayed in the upper right corner of the interface.

##### About

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### BarCoder

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Language

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Notification

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### Zoom

This script is identical with the similar script found in the window interface so we won’t detail it again.

##### ZoomWindow

This script is identical with the similar script found in the window interface so we won’t detail it again.

### Styles

In the styles folder we have only two css stylesheets used by the platform and two images used in the style sheets. The files as used as described bellow:

* Desktop.css: the stylesheet used by the main web pages such as Default.aspx, Login.aspx or Login2.aspx
* inputButtonBG.png: the image used as buttons background
* PageBG.png: the backgroung image for the main web pages
* Window.css: the stylesheet used by the web pages displayed in windows on the content section

### Tools

Some features for the platform need additional 3rd party software so we grouped the 3rd party software files in the “Tools” folder. At this point we have the following 3rd party folder:

* iTextSharp: used to process HTML content into PDF files. Not the most perfect tool but it is free. The platform can be altered to work with another HTML to PDF library with minimum impact.
* jwPlayer: a free flash player used to play video content. This is used by the documentation module for the moment.
* MIMETeX: a free application that converts a LaTeX math formula into an image. This is used by the documentation module and it works from a command line.

### Web Pages

The web pages for this interface have the same functionalities and behaviours as the corresponding web pages from the Window interface however, the collection of web pages from the Plain interface is smaller than the web page collection from the Window interface. This is because the development functionalities were taken out from the plain interface because they may alter in the future and it will be hard to keep track of the changes. The rest of the pages were doubled because the regular users must be able to operate on both interfaces.

We won’t detail the pages again because there is no point but you can check Annex 4 to see the collection of web pages found on this interface.

# Special Syntaxes

In this chapter we will cover the specifics regarding HTML, SQL and XML syntaxes that the platform supports.Keep in mind that the code written using these syntaxes is converted into regular code by replacing tokens and language tags so they have to be written in such a way that after translation they don’t become ambiguous.

Also keep in mind that syntaxes differ according to the contex in which they are used and restrictions may aply over the variety of tokens that can be used.

# Specific XML Syntax

Considering the variety of attributes that can be available for a class or the variety of parameters that can be available for a report, we excapsulate multiple values into an XML structure in order to keep the number of parameters constant for stored procedures and business logic procedure.

To illustrate this structure we will consider our example of the Employee class that has the following attributes: FirstName, LastName, PIN, Birthdate, BirthPlace and Address. In order to insert a new object for this class or update an existing one the information regarding the employee object will be ancapsulated into the following XML string:

<Object>

<Attribute Name="FirstName" Value="John" />

<Attribute Name="LastName" Value="Doe" />

<Attribute Name="PIN" Value="1840122323966" />

<Attribute Name="Birthdate" Value="22/01/1984" />

<Attribute Name="BirthPlace" Value="unknown" />

<Attribute Name="Address" Value="somewhere" />

</Object>

Note that the XML structure has a root node called “Object”. The root node is required by XML standards so we had to have one. Also, as you can see, all attributes have an associated “Attribute” node containing their name and value for them.

The encapsulation is identical for report parameters. If we have a report with two parameters called “FilterFirstName” and “FilterPIN” the encapsulation will look like this:

<Object>

<Attribute Name="FilterFirstName" Value="John" />

<Attribute Name="FilterPIN" Value="1840122323966" />

</Object>

This incapsulation can be made programaticaly by using the Amaryllis.Tools.Packer class from the BusinessLayer. We can use the “Clear” method to clear all attribute and then add them one by one using the “Append” method. In order to retrieve the encoded attributes/parameters we can call the “GetPack” method which will return a string containing the XML encapsulation for the appended attributes and values.

NOTE: this XML encapsulation is used internaly and the developers and users do not have to work with it unless they are doing debugging on a stored procedure or a data import.

## Specific HTML Syntax

In order to describe object data forms, custom form and create documentation articles we point to the standard xHTML language version 1.0 which we enhanced depending on the context used to provide additional functionality.

### Class and Report Development Syntax

The specific HTML language used for the class and report development is enhanced to cover language tags, object display identifiers and attributes/parameters tokens.

#### Language Tag Tokens

##### Single-look and Multi-look

Considering the language tags, they are characterized by the name of the language tag and its values for each of the languages supported by the platform. In order to include a language tag token in an HTML description we must include the language tag’s name between the “[#” and “]” sequences. Bellow he have a small sample of an HTML sequence that has a few language tags embedded.

<table>

<tr>

<td>[#Name]:</td>

<td>[Name]</td>

</tr>

<tr>

<td>[#ShortName]:</td>

<td>[ShortName]</td>

</tr>

<tr>

<td>[#Supervisor]:</td>

<td>[SupervisorID]</td>

</tr>

</table>

In the above example, “[#Name]”, “[#ShortName]” and “[#Supervisor]” are identified as language tags and replaced with the proper strings depending on the interface language.

NOTE: These kinds of tags can also be used in mandatory attributes error messages and class triggers the same way.

For our sample, the above HTML description is translated into the following (assuming the interface is set to English):

<table>

<tr>

<td>Name:</td>

<td>[Name]</td>

</tr>

<tr>

<td>Short name:</td>

<td>[ShortName]</td>

</tr>

<tr>

<td>Supervisor:</td>

<td>[SupervisorID]</td>

</tr>

</table>

As you can see this is a simple string replacement and it is safe if you forget to insert a language tag. If a language tag is missing, then the embedded language tag will remain as is on the translated HTML description.

##### Error Message

For an error message written for a required class attribute we can specify multiple language tags combined with regular text if needed. For example if we have a required field error message that looks like this:

You forgot to enter the [#ShortName].

It will be translated into:

You forgot to enter the Short name.

This makes sense if the interface language is English, but if the interface is set to another language, the error message won’t make any sense. Knowing this, we can combine multiple language tags to form a sintactical and semantical corrent error message or we can provide a single language tag for the whole error message.

#### Class Attributes Tags

These tags are similar with the language tags but they do not have the “#” character marking. The additional “#” marker is excluded in order to help the platform make the difference between language tags and attribute tags.

To conclude, these tags consist in the name of the attribute enclosed in square brackets.

In HTML these tags can be used in the single look and multi look descriptions. And they will be replaced by the HTML code that displays the control through the attribute value will be collected for the single look and by the attribute’s value for the multi-look. For a simple illustration we will consider the example given in the previous chapter:

<table>

<tr>

<td>Name:</td>

<td>[Name]</td>

</tr>

<tr>

<td>Short name:</td>

<td>[ShortName]</td>

</tr>

<tr>

<td>Supervisor:</td>

<td>[SupervisorID]</td>

</tr>

</table>

In this example we have the following attribute tokens:

* Name: collected through a text box control.
* ShortName: collected through a text box control.
* SupervisorID: collected through a select control.

By replacing the attribute tags we get the following HTML description:

<table>

<tr>

<td>Name:</td>

<td>

<input name="Field\_\_Name" class="inputText" id="Field\_\_Name" style="width: 300px;" type="text" value="" />

<input id="Field\_\_Name\_ControlType" type="hidden" value="TextBox" />

</td>

</tr>

<tr>

<td>Short name:</td>

<td>

<input name="Field\_\_ShortName" class="inputText" id="Field\_\_ShortName" style="width: 150px;" type="text" value="" />

<input id="Field\_\_ShortName\_ControlType" type="hidden" value="TextBox" />

</td>

</tr>

<tr>

<td>Supervisor:</td>

<td>

<select name="Field\_\_SupervisorID" class="inputText" id="Field\_\_SupervisorID">

<option value=""></option>

<option value="626">Person 1</option>

<option value="707">Person 2</option>

...

<option value="600">Person n</option>

</select>

<input id="Field\_\_SupervisorID\_ControlType" type="hidden" value="Select" />

</td>

</tr>

</table>

Considering the multi-look description from the same class, the following multi-look description:

<header>

<th>[#ShortName]</th>

<th>[#Name]</th>

<th>[#Supervisor]</th>

</header>

<tr>

<td>[ShortName]</td>

<td>[Name]</td>

<td>[SupervisorID]</td>

</tr>

Will be translated into:

<table width="100%" class="listing" border="0" cellspacing="0" cellpadding="0">

<tr>

<th>Short name</th>

<th>Name</th>

<th>Supervisor</th>

<th style="text-align: right;">

<img class="imageButton" onclick="window.parent.windowNewWindow('','Images/Icons/UsersAndPriviledges.png','Object\_Insert.aspx?ClassID=1', 200, 200, 200, 200);" alt="Add new Department" src="Images/Buttons/Insert.png" />

</th>

</tr>

<tr>

<td class="odd">Short name 1</td>

<td class="odd">Name 1</td>

<td class="odd">Supervisor 1</td>

<td class="odd" valign="top" style="text-align: right;">

<img class="imageButton" onclick="window.parent.windowNewWindow('Name 1','Images/Icons/UsersAndPriviledges.png','Object\_Update.aspx?ObjectID=2741', 200, 200, 200, 200);" alt="View" src="Images/Buttons/View.png" />

</td>

</tr>

<tr>

<td class="odd">Short name 2</td>

<td class="odd">Name 2</td>

<td class="odd">Supervisor 2</td>

<td class="odd" valign="top" style="text-align: right;">

<img class="imageButton" onclick="window.parent.windowNewWindow('Name 2','Images/Icons/UsersAndPriviledges.png','Object\_Update.aspx?ObjectID=2744', 200, 200, 200, 200);" alt="View" src="Images/Buttons/View.png" />

</td>

</tr>

...

<tr>

<td class="odd">Short name n</td>

<td class="odd">Name n</td>

<td class="odd">Supervisor n</td>

<td class="odd" valign="top" style="text-align: right;">

<img class="imageButton" onclick="window.parent.windowNewWindow('Name n','Images/Icons/UsersAndPriviledges.png','Object\_Update.aspx?ObjectID=2783', 200, 200, 200, 200);" alt="View" src="Images/Buttons/View.png" />

</td>

</tr>

</table>

NOTE: the “header” tag is converted into a regular “tr” tag but this will be discussed later on. Also, the table row described will be multiplicated for each object of the given class.

##### Composed Tags

**Just the Value (Single-look)**

Given the fact that we may need to include just the value of an attribute on the objects form, we ca prefix the attribute tag with the “[value]” string. This will trigger the platform to replace the composed tag “[value][Attribute\_Name]” with just the value of an attribute.

**Image Fields (Multi-look)**

For an attribute that represents an image we can make some improvements on the multi-look description by prefixing the attribute tag with either “[image16]”, “[image32]” or “[image48]” to display the actual image. If an image attribute’s tag not prefixed by an image prefix, the tag will just be replaced with the primary key value of the image file.

[image16] displays the image as a 16x16 thumbnail.

[image32] displays the image as a 32x32 thumbnail.

[image48] displays the image as a 48x48 thumbnail.

For this purpose, image files are cached along with a 48x48 thumbnail adjacent image.

**File Fields (Multi-look)**

Similar to image fields, if we prefix the attribute’s tag with the “[file]” prefix, then the attribute tag will be replaced by a () download button.

**Yes/No Fields (Multi-look)**

For a checkbox field type, if we prefixate the attribute’s tag with the “[check]” prefix, the attribute tag will be replace by a () checked or () unchecked image depending on the attribute’s value.

#### Class Tags

Considering the way in which the information is stored, it is hard for a developer to gather multiple informations from multiple classes and keep up with security restriction. In order to facilitate the data access, we introduced the class tags which are available only for SQL development.

A class tag is given by the internal name of a class enclosed in braces. For example, if we want to select the primary key value and title for all the objects of a certain class we can write an SQL syntax that looks like the following:

SELECT C.ID,C.Title

FROM {Class\_Name} AS C

where the “Class\_Name” text represents the internal name of the class.

A class tag written like in the above example is replaced by the corresponding cache table for the class or a subquery that retrieves only the objects over which the users has access.

WARNING: Due to the way in which the class tags are replaced it is imperative that we assign aliases to class tags.

There are several class tag forms available and they are as follows:

* {ClassName}: the regular class tag that is replaced by the cache table associated with the class or a sub query that retrieves only the unrestricted objects for the user.
* {ClassName\_Log}: the regular class tag which is replaced by the cache table log associated with the class or a subquery that retieves the logs for the objects on which the user has access.
* {ClassName#}: a special tag replaced by the cache table associated with the class.
* {ClassName\_Log#}: a special tag replaced by the cache table log associated with the class.

The “ClassName” in the above descriptions represents the internal name of the class.

#### Special Tags

**Add New Tag**

When describing a single-look form for a class we may want to give the user the posibility to jump on an insert form for another class if, let’s say, a select box is missing a value and its options are populated by the objects of another class. For this purpose we have at our disposal the artificial “addnew” HTML tag which has the following syntax:

<addnew class="Class\_Name" />

This tag will be translated into the following lne of code:

<img alt="Add new" src="Images/Buttons/Insert.png" class="imageButton" onclick="window.parent.windowNewWindow('', '', 'Object\_Insert.aspx?ClassID=Class\_ID', 100, 100, 300, 300);" />

Where “Class\_Name” is the internal class name and “Class\_ID” is the primay key value for the class.

**ID and ParentID Tags**

In order to display or use the actual object’s primary key value or theparent object’s primary key value we can use int the single-look description the “[ID]” and “[ParentID]” tags which are replaced with the actual values.

**The #ID Tag**

Considering the fact that we are able to display multiple forms on a single web page, in order to avoid multiple fields named identicaly, the “DisplaySingleInstanceForObject” method requires a parameter named “Identifier” which is required when saving an object as well.

In order to display multiple objectcs of the same class on a web page we can provide distinct identifiers for all the forms. In order to use this identifier in java scripts provided in the single-look description we can use the “[#ID]” tag which will be replaced by the “Identifier” parameter’s value.

WARNING: In order to avoid multiple objects with the same HTML ID we must name HTML elements within the single-look description by combining the desired ID with the “[#ID]” tag, something like “<div id=’DivName[#ID]’>”.

### Report Development Syntax

#### Language Tag Tokens

The language tags for the reports are available for the header description, row description and parameter form description. Also, they are similar in use with the class development attribute tags and we won’t detail them further.

#### Report Parameter Tags

These tags are similar with the class attribute tags and they can be used in the report’s title and in the row source query as well as in the subordered graphs’ row sources.

Considering the report title, parameter tags are replaced by the parameter’s value if provided else they are replaced by the “<Unassigned>” text.

#### Class Tags

The class tags are similar to the class tags from the class development chapter and they also apply to SQL queries. This being said, we won’t detail them again.

### Custom Form Development Syntax

Custom forms are developed using HTML and SQL all together but in this chapter we will focus mainly on the HTML syntax. Custom form descriptions are parsed node by node deep first and artificial inserted tags are parsed differently from regular HTML tags.

#### Side Tab Panel Objects

To add a side tab panel to a custom form we can use the tags seen in the bellow sample:

<sidetab>

<sidetabheaders>

<sidetabheader>Header 1</sidetabheader>

<sidetabheader>Header 2</sidetabheader>

</sidetabheaders>

<sidetabpanels>

<sidetabpanel>

panel content

</sidetabpanel>

<sidetabpanel>

panel content

</sidetabpanel>

</sidetabpanels>

</sidetab>

This script will generate a side panel object with two panels.

#### Top Tab Panel Objects

The top tab panel objects are similar to side tab panel objects but they display the headers on top, not on the left side. To add a top tab panel to a custom form we can use a script like the one bellow:

<toptab>

<toptabheaders>

<toptabheader>Header 1</toptabheader>

<toptabheader>Header 2</toptabheader>

</toptabheaders>

<toptabpanels>

<toptabpanel>

panel content

</toptabpanel>

<toptabpanel>

panel content

</toptabpanel>

</toptabpanels>

</toptab>

#### Listing

In order to trigger an object list display we can use the listing tag along with the mandatory attributes for it as seen in the example bellow.

<listing class="ClassName" parentid="0" withsorting="true" withfiltering="true" withmapping="true" withpagination="true" />

The attributes for this tag are as follows:

* class: the name on the class of the objects listed.
* parentid: the primary key value of the parent object for the listed objects. If this attribute is set to “0” then the parent object for the listed objects will be ignored.
* withsorting: triggers the listing to display the sorting header if it is specified in the class’s multi-look description. If this attribute is set to “true” the sorting header will be displayed according to the class’s multi-look description and if this attribute is set to “false” it will suppress the sorting header.
* withfiltering: similar to the withsorting attribute but it refers to the filtering header.
* withmapping: similar to the withsorting attribute but it refers to the mapping header.
* withpagination: if this is set to “true” it will paginate the listing according to the class’s multi-look description and if it is set to “false” it will list all the objects on a single page.

#### Object

The object tag is placed here to help us add full object data collecting forms to the custom form. An object form can be triggered by a line such as the following:

<object class="ClassName" withtitle="true" id="34" parentid="45" />

The attributes for this tag are as follows:

* class: the class name for the object that must be displayed.
* withtitle: if this is set to “true” the object title will be displayed, else if it is set to “false” then the object’s title won’t be displayed.
* id: the primary key value for the object that must be displayed.
* parentid: the primary key value for the parent object for the object that must be displayed.

#### Mappings

In order to display a mapping listing, we can use the mappings tag as seen in the example bellow:

<mappings class="ClassName" objectid="34" />

The attributes for this tag are the following:

* class: the name of the class for the mapped objects displayed.
* objectid: the primary key for the object that is the source object for the mapping relation.

#### Simple Tags

##### Object Associated Tags

As you know, a custom form is displayed for a root object so, considering the fact that the root object can be characterized by primary key value, parent object primary key value and class primary key value, we have the following simple tags that are replaced in the form description with the proper values:

* [ObjectID] is replaced with the primary key value of the object.
* [ParentID] is replaced with the primary key value of the parent object.
* [ClassID] is replaced with the primary key value of the object’s class.

##### Add Mapping Tag

Depending on the how the form must look, if we need a map adding functionality we have the “addmapping” tag at our disposal. This tag can be used in the following three forms:

<addmapping type="image" />

<addmapping type="button" />

<addmapping />

The first example determines the custom form to display a small image button () and the other two display a regular HTML button with the proper text on it.

#### Complex Tags

##### Objects Tag

The “objects” is used to trigger a multiple object data collecting forms display. The syntax for this tag can be understood from the following example:

<objects class="ClassName" display="sidepanel" withtitle="true">

SQL query

</objects>

The attributes for this tag are the following:

* class: the internal name of the class that represent the objects that need to be displayed.
* display: this attribute can be left blank or can be set to “sidepanel” and “toppanel”. If blank, the objects’ form will be displayed one after the other. If it is set to “sidepanel” or “toppanel” the objects’ forms will be displayed one per panel on the corresponding tab panel object.
* withtitle: if it is set to “true” it will display the object title for each object form, else if it is set to “false” it won’t display the objects’ titles.

The “SQL query” represents a specific SQL query that supports class tags and retrieves the list of objects that must be displayed. On top of the class tags, the sql query will support the following additional tags:

* [@ObjectID] is replaced with the primary key value of the object.
* [@ParentID] is replaced with the primary key value of the parent object.
* [@ClassID] is replaced with the primary key value of the object’s class.

The “objects” tag will be replaced by multiple “object” tags and “sidepanel”/”toppanel” tags if needed. All of these tag replacements will be parsed after the replacement to the proper standard HTML form.

##### Matrix Tag

Matrix tags are a bit harder because the SQL query involved must return exactly three columns named “Vertical”, “Horizontal” and “ID”. But for the start let us consider the “matrix” tag usage.

<matix>

SQL query

</matix>

The SQL query for this tag supports the following tags:

* [@ObjectID] is replaced with the primary key value of the object.
* [@ParentID] is replaced with the primary key value of the parent object.
* [@ClassID] is replaced with the primary key value of the object’s class.

Now, considering that the SQL query returns the following result:

|  |  |  |
| --- | --- | --- |
| Vertical | Horizontal | ID |
| V1 | H1 | 1 |
| V1 | H2 | 2 |
| V2 | H1 | 3 |
| V2 | H3 | 4 |
| V3 | H3 | 5 |

The form for the returned objects will be displayed in a table HTML structure that looks as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | H1 | H2 | H3 |
| V1 | 1 | 2 |  |
| V2 | 3 |  | 4 |
| V3 |  |  | 5 |

Thus, the horizontal column will represent the horizontal dimension for the table and the vertical column will represent the vertical dimension so the object forms will fall into place according to their vertical and horizontal columns values.

#### Conclusions

The translation process for this is a two stage process so it will be a bit slow. On the first stage the simple tags are replaced with the appropriate values and then the complex tags are translated into the corresponding tags. On the second stage the HTML description retrieved from the first stage is parsed and translated into simple HTML tags.

### Documentation Syntax

A documentation article is written as an HTML document but we enhanced the syntax for this code as well because some new features were needed. In this chapter we will pass through the added tags and explain their role.

#### Code Tag

Some articles placed on the platform may be technical so we may need to add programming code to them. The point of this tag is to help format code depending on the programming language so that we don’t have to do it with “p” and “span” tags.

A quick example for this tag is the following:

<code language="SQL">

SELECT \*

FROM Table1 AS T1

INNER JOIN Table2 AS T2 ON T1.ID=T2.ParentID

ORDER BY T1.Col1,T1.Col2

</code>

The above specific HTML sequence will be rewritten to generate the following output:



Depending on the language attribute for the code tag, the inner code will be formatted accordingly. At this point the language attribute supports the following values:

* Basic: formats the code according to the Visual Basic standard.
* CSharp: formats the code according to the Visual C# standard.
* HTML: formats the code according to the xHTML standard.
* JavaScript: formats the code according to the java script standard.
* SQL: formats the code according to the SQL standard.
* XML: formats the code according to the XML standard.
* Other: if the language attribute is missing or doesn’t point to one of the above language this will trigger a generic code format.

#### Box and Sidebox Tags

The “box” and “sidebox” tags are used to mark content parts with a border. The only difference between the two is that the “sidebox” tag will compact the content to a 200 pixels box which will be moved on the right side of the display. The usage for the two tags is as illustrated bellow:

|  |  |
| --- | --- |
| box tag | sidebox tag |
| <box>  HTML code ...  </box> | <sidebox>  HTML code ...  </sidebox> |

#### Latex Tag

Some articles may include mathematical formulas. Some formulas can’t be reproduce in HTML at this point or require an additional non-standard exntension for the web browser. In order to make this work without additional setup on the client side, we went for a formula to image conversion solution.

From experience, the easiest way to write a math formula was using LaTeX and with the help of third party software we are able to convert formulas written in LaTeX to images. The usage for this tag is as shown in the following example:

<latex>

LaTeX math formula ...

</latex>

#### The HDD Tags

Content from the virtual harddrive can be placed in article’s content through the following three tags:

* hddimage: inserts the specified file as an image inside the article.
* hddfile: inserts the specified file as a download link inside the article.
* hddvideo: inserts the specified file as a video inside the article.

All the above tags must be written with the “src” attribute. The “src” attribute must contain, as value, a harddrive file GUID. There are three lines bellow that illustrate the usage for these tags.

<hddimage src="FC2A3C1B-E3F6-462F-8613-ADDEE654539A" />

<hddfile src="FC2A3C1B-E3F6-462F-8613-ADDEE654539A" />

<hddvideo src="FC2A3C1B-E3F6-462F-8613-ADDEE654539A" />

#### Bullet Tag

The bullet tag is a simple tag that is replaced with a simple image. This can be used as a simple image or as a bullet point for items or what ever we may need. The usage for this tag is as follows:

<bullet />

## Specific SQL Syntax

### Class Development Syntax

#### Language Tags

For the SQL embedded language tags we have to be more careful. Considering that the language tag tokens are replaced with the corresponding strings, we may end up with an inavid SQL script, for example:

DECLARE @Var VARCHAR(MAX)

SET @Var=[#ShortName]

We end up with the following SQL script:

DECLARE @Var VARCHAR(MAX)

SET @Var=Short name

This is an invalid SQL script. We can correct this by placing the language tag inside quotation signs:

DECLARE @Var VARCHAR(MAX)

SET @Var='[#ShortName]'

#### Class Attribute Tags

The attribute tags can be used in the class’s title description, triggers and attributes computing formulas (default value combined with the “is formula” field). In the case of SQL scripts the attribute tags will be replaced by the attributes’ values enclosed into SQL expresions to convert them to the desired data types due to the fact that the values are stored as strings. For example we have the following transformations:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Data type (Length) | Value | Tag | Formulas |
| AttrBigInt | BIGINT | 12 | [AttrBigInt] | Input:  CONVERT(BIGINT,'1')  Output:  12  Formula:  12 |
| AttrBit | BIT | 1 | [AttrBit] | Input:  CONVERT(BIT,ISNULL(NULLIF('1',''),0))  Output:  CONVERT(INT,'1')  Formula:  ISNULL(NULLIF('1',''),0) |
| AttrDatetime | DATETIME | 22/01/1984 | [AttrDatetime] | Input:  CONVERT(DATETIME,NULLIF('22/01/1984',''),103)  Output:  ISNULL(CONVERT(VARCHAR,'22/01/1984',103),'')  Formula:  CONVERT(VARCHAR,NULLIF('22/01/1984',''),103) |
| AttrFloat | FLOAT | 1.43 | [AttrFloat] | Input:  CONVERT(FLOAT,REPLACE('1.43',',','.'))  Output:  1.43  Formula:  1.43 |
| AttrInt | INT | 13 | [AttrInt] | Input:  CONVERT(INT,'13')  Output:  13  Formula:  13 |
| AttrNVarchar | NVARCHAR(32) | Text text | [AttrNVarchar] | Input:  CONVERT(NVARCHAR(32),N'Text text')  Output:  ‘Text text’  Formula:  CONVERT(NVARCHAR(32),N'Text text') |

NOTE: the VARBINARY data type is avoided because it is hard to illustrate with a given value, but the principal remains the same.

The three replacement forms are used as needed through out the platform and they can be adjusted by updating the corespondin values in the ClassAttributeType table.

#### Special Tags

For the SQL events, title expression, attribute’s rows source, default value expressions and formula expressions we can use the following special tags:

* ID: the tag for the object primary key value is “[@ID]” and it is replaced by the actual value of the object’s primary key.
* ParentID: the tag for the parent object primary key value is “[@ParentID]” and it is replaced by the actual value of the parent object’s primary key.
* UserID: the tag for the user’s primary key value is “[@UserID]” and will be replaced by the primary key value of the logged on user that triggered the batabase operation.

### Report Development

#### Report Parameter Tags

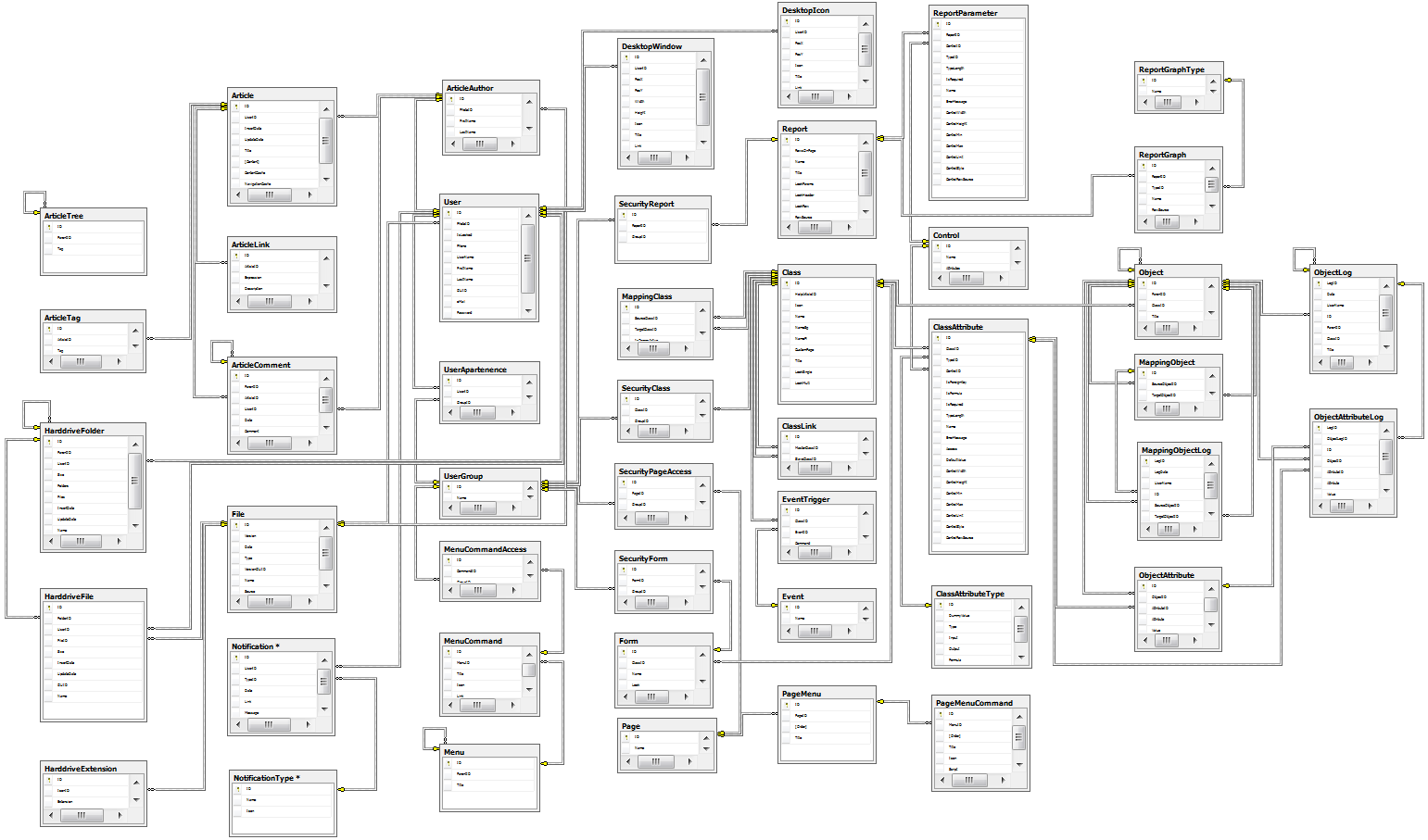
These tags are similar with the class attribute tags and they can be used in the report’s title and in the row source query as well as in the subordered graphs’ row sources.

In SQL queries parameter tags will be replaced in the same way that the class attributes tags are replaced, so we won’t detail them.

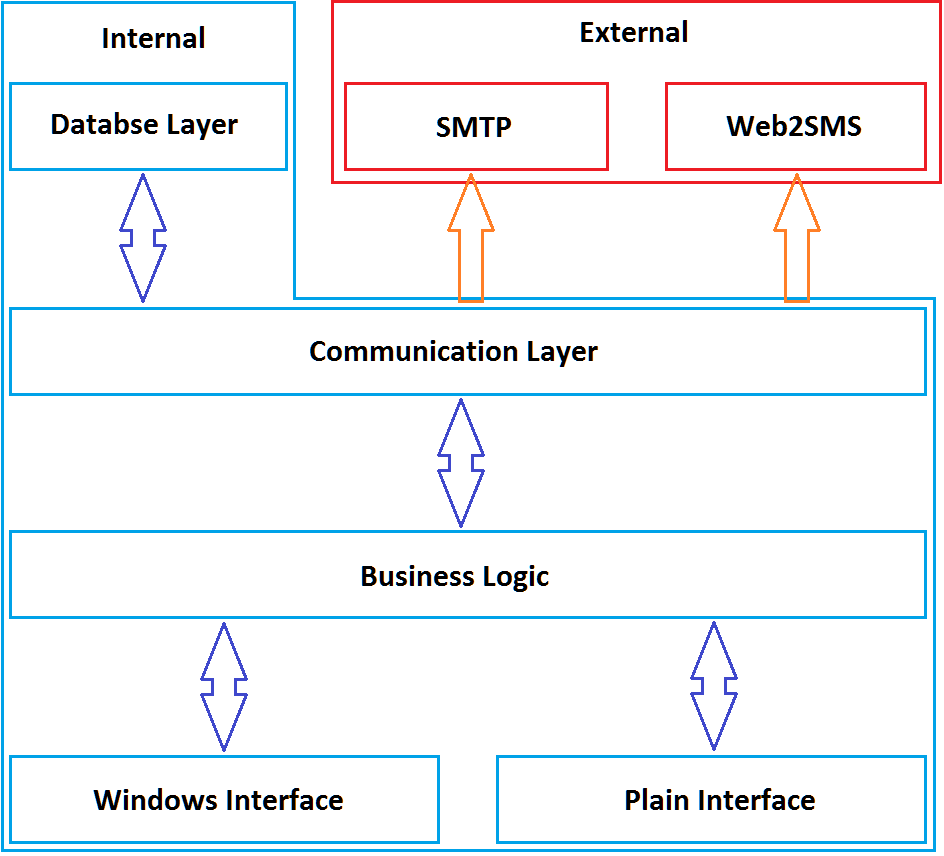
#### Class Tags

The class tags are similar to the class tags from the class development chapter and they also apply to SQL queries. This being said, we won’t detail them again.

# Annex 1 – Global Entity-Relation Diagram



# Annex 2 – Platform Layers and Interactions



# Annex 3 – Default Web Page Access for the Default Security Groups

|  |  |  |
| --- | --- | --- |
| Web Page | Administrators | Users |
| About | Unrestricted | |
| AccessoryCalc |  |  |
| AdminDuplicateObjects |  |  |
| AdminMergeObjects |  |  |
| AdminSettings |  |  |
| AdminUsers |  |  |
| AdminUsers\_Insert |  |  |
| AdminUsers\_Update |  |  |
| Article |  |  |
| Article\_Insert |  |  |
| Article\_Update |  |  |
| ArticleLink |  |  |
| ArticleSearch |  |  |
| ArticleStart |  |  |
| ArticleTree |  |  |
| ArticleTreeFrame |  |  |
| Background |  |  |
| BarCode\_Download |  |  |
| ChangePassword |  |  |
| Class |  |  |
| Class\_Insert |  |  |
| Class\_InsertAttribute |  |  |
| Class\_UpdateAttribute |  |  |
| Controls\_FileUpload |  |  |
| Controls\_ImageUpload |  |  |
| Default |  |  |
| DesktopIcon\_Insert |  |  |
| DesktopIcon\_Update |  |  |
| Download |  |  |
| Error | Unrestricted | |
| ExportDownload |  |  |
| Form\_Insert |  |  |
| FormAddMapping |  |  |
| Forms |  |  |
| FormView |  |  |
| Harddrive |  |  |
| HarddriveDownload |  |  |
| HarddriveFile\_Insert |  |  |
| HarddriveFile\_Update |  |  |
| HarddriveFolder\_Insert |  |  |
| HarddriveFolder\_Update |  |  |
| HarddriveIcons |  |  |
| HarddrivePage |  |  |
| HarddriveRefresh |  |  |
| HarddriveSearch |  |  |
| Login | Unrestricted | |
| Login2 | Unrestricted | |
| Logout | Unrestricted | |
| MappingObject |  |  |
| Menu |  |  |
| Menu\_InserCommand |  |  |
| Menu\_InsertMenu |  |  |
| Menu\_UpdateCommand |  |  |
| Menu\_UpdateMenu |  |  |
| Notification |  |  |
| Object\_Insert |  |  |
| Object\_Update |  |  |
| ObjectList |  |  |
| ObjectLog |  |  |
| QuickLink |  |  |
| Recycle |  |  |
| RefGuide\_Class |  |  |
| RefGuide\_ClassList |  |  |
| Report |  |  |
| Report\_Insert |  |  |
| Report\_InsertGraph |  |  |
| Report\_UpdateGraph |  |  |
| ReportGraphView |  |  |
| ReportParameter\_Insert |  |  |
| ReportParameter\_Update |  |  |
| ReportView |  |  |
| Search |  |  |
| Security |  |  |
| Security\_InsertGroup |  |  |
| Settings |  |  |
| StopImpersonation |  |  |
| Task |  |  |
| UserChangeData |  |  |
| UtilityCloseWindow |  |  |
| Welcome |  |  |

NOTE: Keep in mind that all users are part of the “Users” security group, even administrators, so an administrator has access to all the web pages because he is also part of the “Users” security group.

# Annex 4 – Web Page Availability for the Interfaces

|  |  |  |
| --- | --- | --- |
| Web Page | Plain Interface | Window Interface |
| About |  |  |
| AccessoryCalc |  |  |
| AdminDuplicateObjects |  |  |
| AdminMergeObjects |  |  |
| AdminSettings |  |  |
| AdminUsers |  |  |
| AdminUsers\_Insert |  |  |
| AdminUsers\_Update |  |  |
| Article |  |  |
| Article\_Insert |  |  |
| Article\_Update |  |  |
| ArticleLink |  |  |
| ArticleSearch |  |  |
| ArticleStart |  |  |
| ArticleTree |  |  |
| ArticleTreeFrame |  |  |
| Background |  |  |
| BarCode\_Download |  |  |
| ChangePassword |  |  |
| Class |  |  |
| Class\_Insert |  |  |
| Class\_InsertAttribute |  |  |
| Class\_UpdateAttribute |  |  |
| Controls\_FileUpload |  |  |
| Controls\_ImageUpload |  |  |
| Default |  |  |
| DesktopIcon\_Insert |  |  |
| DesktopIcon\_Update |  |  |
| Download |  |  |
| Error |  |  |
| ExportDownload |  |  |
| Form\_Insert |  |  |
| FormAddMapping |  |  |
| Forms |  |  |
| FormView |  |  |
| Harddrive |  |  |
| HarddriveDownload |  |  |
| HarddriveFile\_Insert |  |  |
| HarddriveFile\_Update |  |  |
| HarddriveFolder\_Insert |  |  |
| HarddriveFolder\_Update |  |  |
| HarddriveIcons |  |  |
| HarddrivePage |  |  |
| HarddriveRefresh |  |  |
| HarddriveSearch |  |  |
| Login |  |  |
| Login2 |  |  |
| Logout |  |  |
| MappingObject |  |  |
| Menu |  |  |
| Menu\_InserCommand |  |  |
| Menu\_InsertMenu |  |  |
| Menu\_UpdateCommand |  |  |
| Menu\_UpdateMenu |  |  |
| Notification |  |  |
| Object\_Insert |  |  |
| Object\_Update |  |  |
| ObjectList |  |  |
| ObjectLog |  |  |
| QuickLink |  |  |
| Recycle |  |  |
| RefGuide\_Class |  |  |
| RefGuide\_ClassList |  |  |
| Report |  |  |
| Report\_Insert |  |  |
| Report\_InsertGraph |  |  |
| Report\_UpdateGraph |  |  |
| ReportGraphView |  |  |
| ReportParameter\_Insert |  |  |
| ReportParameter\_Update |  |  |
| ReportView |  |  |
| Search |  |  |
| Security |  |  |
| Security\_InsertGroup |  |  |
| Settings |  |  |
| StopImpersonation |  |  |
| Task |  |  |
| UserChangeData |  |  |
| UtilityCloseWindow |  |  |
| Welcome |  |  |

# Annex A – Platform Changes Tracker

## Version 2.5.55 – Manganese 55

* [Feature][Minor] Reference guide module for easy documentation regarding the developed classes.
* [Feature] [Minor] Object cloning possibility.
* [Feature][Major] Cache tables are generated with foreign key constraints and foreign key indexes in order to help process reports faster.
* [Feature][Major] Indexes created for the static platform’s tables in order to speed up queries. This was avoided for the time in order to monitor the behavior better.
* [Bug fix] [Minor] The AddSeparator method in the Menu class was ignoring the specified Width for the object.
* [Bug fix] [Minor] Removed the reference guide page from the menu and the article knowledge base page as well on the plain interface.
* [Bug fix] [Minor] Design issue on the report update page. The language tags table had a table header cell on the second table row.
* [Bug fix] [Major] Report was displaying the select control wrong when needing a reference object list.
* [Bug fix][Major] The report edit page was translated the loaded information and loose the embedded tags on save.
* [Bug fix][Major] The merge objects form was passing bad parameters to the mapping object web page.
* [Bug fix][Major] Correct the date picker to determine the day of the week correctly.
* [Bug fix] [Minor] Modal box display was failed when the page was scrolled.
* [Bug fix][Major] The Raise\_Event stored procedure was not translating the language tags in the error messages for the required fields.
* [Bug fix] [Minor] The task bar menu was not closing when the resize event was triggered causing issues when the Windows 8 style taskbar was displayed.
* [Bug fix] [Minor] The recycle bin page was crashing when retrieving object listings for certain classes.
* [Bug fix][Major] Object update and insert procedures altered so that if a formula fails, the formula won’t end up in the ObjectAttribute table to cause errors when computing cache tables.
* [Bug fix] [Major] The title for an object’s log was not processed linguistically.
* [Bug fix][Major] The custom form language tag section was not operating properly when adding a new language tag.
* [Bug fix][Minor] The script resources for the Forms web page were doubled for the form delete operation and they were mixing up the modal box messages for the form delete operation and revoke access operation.
* [Bug fix] [Minor] Design issue fixed for the “History” section of the Object\_Update web page.
* [Bug fix] [Minor] Reference guide module was not displaying table buttons align to the right.
* [Bug fix] [Minor] Security group display for a report was missing the table header.
* [Bug fix][Major] The “Object\_GetSearchResult” stored procedure was returning column names and table aliases as result.
* [Bug fix][Major] The “Setting\_GetSettingsForUser” was returning only the setting that had different values from the default values.
* [Code] [Minor] Altered the balloon java script to correct the objects to the grammatical correct form.

## Version 2.5.54 – Manganese 54

* This is the start version for this document and all the changes from this version forward will be logged in this annex.

# Annex B – Manual Changes Tracker

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Version | Changes | Person |
| September 22nd 2012 | 2.5.55 rev. 00 | Initial manual start for Amaryllis (Manganesse 55). | Radu Ungureanu |
| January 4th 2013 | 2.5.55 rev. 01 | First manual ever made for Amaryllis is finished and it covers the Manganesse 55 distribution with all the fixes and emprouvements done in the period when the manual was written. | Radu Ungureanu |

1. PIN = this stands for Personal Identification Number. [↑](#footnote-ref-1)
2. Primary key = a value that uniquely identifies a row in a table, usually it is an integer incremented automatically. [↑](#footnote-ref-2)
3. Foreign key = a value that represent a primary key from another table or the same table in case of hierarchical relations. [↑](#footnote-ref-3)
4. GUID = stands for Global Unique Identifier. It is a unique sequence of letters and numbers which cannot be generated again on the same machine or another machine. GUID sample: A285E20D-AA93-4404-B3CF-FC4FE3BDF5AB. [↑](#footnote-ref-4)
5. Tokened = the text or expression may contain tokens (token sample: [#token]) that will be replaced by the corresponding text according to the token and the language set for the interface. [↑](#footnote-ref-5)
6. SP = stands for Stored Procedure [↑](#footnote-ref-6)
7. UDF = stands for User Defined Function [↑](#footnote-ref-7)
8. TF = stands for Table Function [↑](#footnote-ref-8)
9. One-to-many = this means that for a single object of one type we can have many linked object of another type. For example, a person has many ID cards in his life time, but an ID card can’t have multiple owners, thus one person has many ID cards. [↑](#footnote-ref-9)
10. Many-to-many = this means that for a single object of one type we can have many linked object of another type but we also means that for a single object of the other type we can have many objects of the first type. For example, a person may possess multiple properties and a property may be possessed by multiple persons. [↑](#footnote-ref-10)