5.23. Client Class

Namespace : `Consigno.Architecture`
Inheritance : `Object > SQLObject > Client`

Represents a client's file and history. This class allows direct edits to all aspects of a client, from basic contact information to payments. After making any changes to a client object, call the `Save()` method to commit changes to the database. In the event of a database failure or network disconnection, `SaveLocal()` can be used to prevent data loss.

Examples

The following code example creates a Client, adds an InvoiceItem, and commits the changes to the database.

```
private void CreateTestClient()
{
    // Create the client.
    Client dutch = new Client(firstName: "Dutch", lastName: "Schaefer");

    // Create an invoice item.
    dutch.InvoiceItems.Add(new InvoiceItem(
        date: DateTime.Now,
        description: "Chopper Transportation Services",
        rate: 129.99,
        hours: 2.5,
        employeeID: 87
    ));

    // Commit changes to the database.
    dutch.Save();

    // The database assigned Dutch an ID.  Print it to the console.
    Console.WriteLine($"Created Client {dutch.ID}.");
}
```

The next code example handles printing invoices for the first 100 clients. The application uses an HTML template to customize email messages. The application also attaches a PDF of the invoice to the message. Finally, an internal notification email is sent when a client has more than $10,000 in unpaid debts.

```
private void PrintSomeInvoices(string htmlTemplate, string reportTemplate)
{
    int[] range = Enumerable.Range(1, 100).ToArray();

    foreach (Client client in Client.FromIDs(range))
    {
        // Fill in the templates with relevant missing data, e.g. client names.
        string html = String.Format(htmlTemplate, /* ... */);
        string accountingReport = String.Format(reportTemplate, /* ... */);

        // Print the invoice and email the client if we have an email address.
        string pdf = client.PrintInvoice();
        if (!String.IsNullOrEmpty(client.EmailAddress))
            client.SendEmail("Your April Invoice from TinyCorp", html, pdf);

        // Send an email to accounts payable if the client is very overdue.
        pdf = client.PrintSummary();
        if (client.TotalDelinquentBalance > 10_000)
            Consigno.Util.Mailman.SendEmail("accountspayable@tinycorp.com",
$"Delinquent Client: {client.LastName}", accountingReport, pdf);
    }
}
```

Notes

When you load a client, we recommend that you call `LoadLocal()` to make sure there were no lost or tentative changes to the client the last time the software was used.

If uncommitted changes exist to a client when the application is exiting, the application displays a pop-up prompting the user to either save changes to the database, save changes locally, or discard the changes.

Constructors

`Client(string json)`	Protected constructor. Initializes a client from a JSON element retrieved from the database or from a local record.

`Client(`
`int id,`
`string firstName,`
`string lastName,`
`string company,`
`string email,`
`string defaultPhone,`
`string alternatePhone,`
`string addressLine1,`
`string addressLine2,`
`string addressLine3,`
`BalanceHistory history,`
`bool contractSigned,`
`bool exemptLateFees)`
	Initializes a Client using provided parameters. You must provide at least one of the following: `firstName`, `lastName`, or `company`.
	If an ID is not explicitly provided, the database will assign the ID number automatically when the Client is saved.
	All fields are optional and may be omitted by using named parameters.

Properties

`AddressLine1`	`string`	Gets or sets the first line of the client's address.
`AddressLine2`	`string`	Gets or sets the second line of the client's address.
`AddressLine3`	`string`	Gets or sets the third line of the client's address.
`AlternatePhone`	`string`	Gets or sets the client's alternate phone number.
`BalanceHistory`	`BalanceHistory`	Gets or sets the `BalanceHistory` for the client.

`Company`	`string`	Gets or sets the client’s company name.
`DefaultPhone`	`string`	Gets or sets the client’s default phone number.
`EmailAddress`	`string`	Gets or sets the client’s email address.
`FirstName`	`string`	Gets or sets the client’s given name.
`HasSignedContract`	`bool`	Gets or sets the value indicating whether or not the client has a signed and completed contract with the business.
`InvoiceItems`	`List<InvoiceItem>`	Gets or sets all of the client’s currently recorded charges.
`InvoiceItemsToPost`	`List<InvoiceItem>`	Gets or sets all of the client’s currently recorded charges for the current billing cycle.
`IsExemptedFromLateFees`	`bool`	Gets or sets the value indicating whether or not late fees apply to the client.
`LastName`	`string`	Gets or sets the client’s family name.
`Payments`	`List<Payment>`	Gets or sets the client’s payments.
`Rates`	`Rate[]`	Gets or sets the default employee Rates for the client, sorted by EmployeeID.
`TotalDelinquentBalance`	`double`	Readonly. Returns the sum of previous unpaid invoice totals subject to late fees.
`TotalInvoiceItems`	`double`	Readonly. Returns the sum of the InvoiceItem `Fees` property.
`TotalInvoiceItemsToPost`	`double`	Readonly. Returns the sum of the posted InvoiceItem `Fees` property.
`TotalPastPayments`	`double`	Readonly. Returns the sum of all past Payments.
`TotalPaymentsToPost`	`double`	Readonly. Returns the sum of the payments for the current billing cycle.
`UpcomingInvoiceTotal`	`double`	Readonly. Returns the sum of charges for the current billing cycle, payments made, and any applicable late fees.
`UpcomingLateFees`	`double`	Readonly. Returns the sum of the late fees based on unpaid past invoice totals.

Instance Methods

`Age()`	Processes the client for the end of the billing cycle. Marks related InvoiceItems and payments as posted, applies late fees, and updates the database accordingly. By default, InvoiceItems older than 90 days are permanently deleted, and their totals only reflected in the balance history. This time period is

	configurable by the administrator in Settings > Invoice Item Data Retention.
`CloseDetailsWindow()`	Closes the ClientDetails window for the client. If there is no window open, a warning is added to the system logs.
`Delete()`	Deletes the client from the database. Changes are committed immediately, so this change is final.
`DeleteLocal()`	Deletes any local record for the client if one exists.
`LoadLocal(bool purge)`	Updates the client from locally-saved changes. If `purge` is set to true, the local record is deleted after being retrieved. Returns true if a local record existed and false if no record could be found.
`OpenDetailsWindow()`	Launches a ClientDetails window for the current client. If the client object has not yet been committed to the database, an `UncommittedDataException` is thrown.
`PrintSummary(Document doc)`	Creates a print job for a monthly report on the client and sends it to the default printer. If no `Document` is provided, the default report creates a line item for invoice totals by month and running totals for invoices and payments. Returns a string containing the path of a generated PDF document for the summary report.
`PrintInvoice(Document doc)`	Creates a print job for the current invoice and sends it to the default printer. If no `Document` is provided, the default invoice template is used. Returns a string containing the path of a generated PDF document for the current invoice.
`Save()`	Commits all changes made to the client to the database.
`SaveLocal()`	Saves a JSON representation of the client to Consigno's temp directory.

`SendEmail(`
`    string subject,`
`    string body,`
`    string attachmentPath)`
	Sends an email to the client with specified subject, body, and attachment (if any).

`ToString()`	For debugging purposes only. Returns a string representation of the client. Example: `"87 Schaefer, Dutch"`

Static Methods

`FromID(int id)`	Returns a single `Client` object containing the specified client. If the ID does not exist, returns null and raises the IDNotFound event.
`FromIDs(params int[] ids)`	Returns an array of `Client` objects containing all clients with provided IDs. If a specified ID is not found, the IDNotFound event is raised.
`GetAll()`	Returns a `List<Client>` of all existing clients, sorted by ID.
`GetNextID(bool reserve)`	Gets the next available ID from the database. If `reserve` is set to true, the

	database will hold the ID for the current client until `Save()` is called or the application exits.

Event Listeners

`OnAgingFailure(EventArgs)`	Raises an event when the `Age()` function fails for any reason.
`OnDBCommitFailure(EventArgs)`	Raises an event when `Save()` fails and the database transaction is not committed.
`OnDBConnectFailure(EventArgs)`	Raises an event when the database is unreachable for any reason.
`OnEmailFailed(EventArgs)`	Raises an event when an email is not sent due to a bad configuration or an email address fails the RFC5322-compliant regex. The event is not raised if the format of the client's email address is valid, but the address does not exist.
`OnEmailSent(EventArgs)`	Raises an event when an email is successfully sent to the client.
`OnIDNotFound(EventArgs)`	Raises an event when the database does not contain a client with the specified ID.