Programação em C# - Na Regra de Negócio

> IMPORTANTE: as variáveis e métodos descritos aqui só funcionarão na programação C# na **Regra de Negócio** do formulário. Para a programação C# nas **APIs** clique aqui.

O agilityflow permite a customização em C#, além de já disponibilizar diversas bibliotecas e funções para facilitar sua programação, incluindo acesso a dados, validações, envio de e-mail, notificação, entre outras.

Abaixo mostramos alguns exemplos, se o material abaixo não for suficiente , entre em contato com nossa equipe.

C# - Propriedades

Variável	**Tipo**	
IsNovoFormulario	bool	retorna true, caso seja um novo formulário, e false caso contrário.
IdFormulario	Guid	retorna o ID do formulário

CurrentStatusFluxo	StatusFluxo? (Enum)	Variável disponível apenas para Workflow
		
		Nessa ENUM será retornado o Status do fluxo.
		As opções podem ser:
		
		• **StatusFluxo.Pendente -** Enquanto o fluxo ainda estiver pendente em alguma das etapas.
		• **StatusFluxo.Aprovado -** Depois do fluxo encerrado e completamente aprovado
		• **StatusFluxo.Reprovado -** Depois do fluxo encerrado e completamente reprovado
		
		Exemplo de utilização:
		
		`if(CurrentStatusFluxo == StatusFluxo.Aprovado){`
		`  //restante do código`
		`}`

(deprecated) ~~FormularioCamposPreenchidos~~	**(deprecated)** ~~dynamic (Json)~~	**(deprecated**: utilizar o método FormContext.GetValue descrito mais abaixo)
		retorna um json com os campos do preenchidos no formulário, exemplo:
		{
		"campo1": "xxxxx",
		"campo2": "yyyyy",
		"campo3": "zzzzz",
		}
		
		Para resgatar a informação do campo1, utilize:
		
		`var campo1 = FormularioCamposPreenchidos["campo1"];`

C# - Buscar e preencher valor de um campo

Método	**Retorno**	

FormContext.SetValue(string idCampo, string value)	void	Esse método preencherá um campo com um novo **Valor** (passado por paramêtro)
		
		Para valores que são **Datas**, utilizar o padrão dd/MM/aaaa HH:mm:ss:
		Exemplo: ***31/12/2010 22:55***
		`var data_e_hora = "31/01/2023 22:55";`
		`FormContext.SetValue("data_e_hora",data_e_hora);`
		
		Para valores **Numéricos**, o número deve estar no formato com ponto no separador decimal e sem utilização de virgulas.
		Exemplo para 2 milhões e 50 centavos**:**
		2000000.50
		`var moeda = "2000000.50";`
		`FormContext.SetValue("moeda",moeda);`
		
		Veja exemplo

FormContext.**GetValue**(string idCampo)	string	Esse método retornará o **Valor** de um determinado campo
FormContext.GetText(string idCampo**)**	string	Esse método retornará o **Texto** de um determinado campo, utilizado apenas se o campo for um campo de **lista dinâmica**, exemplo: Combo, Auto complete, Campo de Múltipla escolha, radio etc. Veja exemplo
FormContext.GetInt(string idColuna)	int?	Retorna o valor da coluna no tipo INT (inteiro), caso esse campo no formulário seja um número, se o valor for 10.000,99 Será retornado: 10000 Veja exemplo

FormContext.GetDecimal(string idColuna)	decimal?	Retorna o valor da coluna no tipo Decimal, caso esse campo no formulário seja um número, se o valor for 10.000,99 Será retornado: 10000.99 Veja exemplo
FormContext.GetDateTime(string idColuna)	DateTime?	Retorna o valor da coluna no tipo DateTime para valores que estão no formato de data dd/MM/yyyy ou dd/MM/yyyy hh:mm Veja exemplo

C# - Buscar Rascunho por Id

Método	Retorno	

FormContext.**GetDrafJsontById**(string draftid)
ou
FormContext.**GetDrafJsontById**(Guid?draftid)
	dynamic	var str_preco_de_venda = "";
		var str_quantidade = "";
		var temRascunho = false;
		var rascunho_json = PageContext.GetDraftJsontById(str_draftid);
		if(rascunho_json != null)
		{
		temRascunho = true;
		str_preco_de_venda = rascunho_json.preco_de_venda;
		str_quantidade = rascunho_json.quantidade;
		}

C# - Executar uma consulta SQL (Query SQL)

Método	Retorno	
await FormContext.**GetDataTableAsync**(string sql, params DbParameter[] parameters)	DataTable	Permite retornar informações dos bancos de dados através de uma query. Veja exemplo

C# - Inserir (salvar/criar), alterar, deletar e aprovar outros formulários

Método	**Retorno**	
await FormContext.SaveEntityAsync (Guid estruturaformularioid_ASerCriadoOuAtualizado, DataDictionary campos_e_valores)	Guid	Salva (Insere e Atualiza) as informações de um formulário no bancos de dados. Para atualizar um registro, no parâmetro **campos_e_valores**, apenas adicione um parâmetro chamado "**id**" com o id do elemento que você deseja atualizar. Obs: Para cadastrar valores numéricos, o números devem estar no formato com ponto no separador decimal: 999999.99 sem utilização de virgulas. Veja exemplo
await FormContext.ApproveEntityAsync (Guid estruturaformularioid_ASerAprovado, DataDictionary campos_e_valores)	Guid	Avança uma etapa em um formulário com Workflow. As regras de usuário aprovador, permissão de aprovação e responsabilidade por uma etapa devem ser validadas antes desse método ser executado. Obs: Para cadastrar valores numéricos, o números devem estar no formato com ponto no separador decimal: 999999.99 sem utilização de virgulas. Veja exemplo
await FormContext.DeleteEntityAsync(Guid formularioid)	void	Deletar um um registro

C# - Apresentar mensagem para o usuário

Método	**Retorno**	
FormContext.WarningMessage (string message)	void	Apresenta um aviso para o usuário veja detalhes aqui

C# - Buscar informações do usuário logado

Método	Retorno	
FormContext. GetUsuarioLogadoNome();	string	Nome Completo do usuário logado
FormContext. GetUsuarioLogadoPrimeiroNome();	string	Primeiro nome do usuário logado
FormContext. GetUsuarioLogadoPrimeiroESegundoNome();	string	Primeiro e Segundo nome do usuário logado
FormContext. GetUsuarioLogadoEmail();	string	E-mail do usuário logado
FormContext. GetUsuarioLogadoId();	Guid	Id do usuário logado

C# - Buscar variáveis de ambiente

Método	Retorno	
FormContext.GetEnvironmentVariable(string nomeVariavel) ou **FormContext.GetEnvironmentVariable_Text**(string nomeVariavel) **	string	Recupera o Valor de uma variável de ambiente. ** Recupera a Descrição (texto), no caso de variáveis do Tipo "Query com Id + Descrição":

C# - Formatação Numérica

> ***Se você está utilizando o método GetValue(..) para buscar o valor de um campo numérico, esse método já retornará o número formatado e não será necessário nenhuma formatação adicional.***
>
> ***Se você está utilizando o método GetDecimal(..) ele não virá formatado pois ele retorna o tipo DECIMAL do C#, sendo assim, se você precisa do dado formatado você pode utilizar o GetText(..)***

string **FormContext.FormatDecimalToString**(object numDecimal, string mascaraTipo = "dec2")	formata object 999999999.55 para o formato da mascara de acordo com o idioma. - se for ingles, será 999,999,999.55 - se for portugues, será 999.999.999,55
string **FormContext.FormatNumberToString_EnglishFormat**(object num, [optional]int? qtdCasasDecimais)	converte decimal para string, porém sempre a string vem no formato Inglês. coloca a a mesma qtd de casas decimais, caso o parâmetro qtdCasasDecimais estivar em branco

C# - Gerar log

Método	Retorno	
FormContext.Log(string log, string logTipo)	void	Cria um log na tabela de log geral, o log pode ser consultado entrando na Área de Customização e clicando na opção "Log Geral >> consultar Log"

C# - Métodos Úteis

Método	Retorno	
FormContext.GetUrlBase()		Retorna a URL base da sua aplicação exemplo: https://suaempresa.agilityflow.io/ *A url sempre virá com uma barra no final **/**

Método principal para utilização da Programação em C# na opção "Regra de Negócio"

A única regra é que o método se chame Execute, não tenha parâmetros de entrada e o retorno seja void.

Exemplo:

```
public void Execute(){



  //aqui você pode programar



}
```

```
  //aqui você pode programar outros métodos
```

Exemplos

Recuperando dados do banco de dados através de uma query SQL

```
//parametros da query
var paramsQuery = new List<NpgsqlParameter>();
paramsQuery.Add(new NpgsqlParameter("@param1", "xxx"));
paramsQuery.Add(new NpgsqlParameter("@param2", "yyy"));
paramsQuery.Add(new NpgsqlParameter("@param3", "zzz"));

//query usando (nolock) nas tabelas para evitar problema com a transação que está ativa
var sql = "select column1,column2,column3 from table (nolock) where column1 = @param1 or  column2 =
@param2 or  column3 = @param3 ";

//executar no banco de dados
var dt = await FormContext.GetDataTableAsync(sql, paramsQuery.ToArray());

//percorrendo as linhas de retorno da tabela
foreach (DataRow dr in dt.Rows)
{
  if (dr["column1"] != DBNull.Value)
  FormContext.WarningMessage(dr["column1"].ToString());
}
```

Modificando os campos do próprio formulário que está sendo salvo

```
//concatenando o texto "abcdef" ao campo de texto chamado "texto_simples"
var texto_simples = FormContext.GetValueField("texto_simples");
texto_simples += " abcdef ";
FormContext.SetValue("texto_simples",texto_simples);

//colocando uma data fixa em um campo de data chamado "data_e_hora"
var data_e_hora = "31/01/2023 22:55";
FormContext.SetValue("data_e_hora",data_e_hora);



//colocando um valor decimal fixo em um campo de data chamado "moeda"
var moeda = "2000000.50";
```

```
FormContext.SetValue("moeda",moeda);



//somando 200 dias a um campo de data chamado "data_e_hora_2"
var dt_data_e_hora = FormContext.GetDateTime("data_e_hora_2");
if(dt_data_e_hora != null){
  var nova_data = dt_data_e_hora.Value.AddDays(200);
  FormContext.SetValue("data_e_hora_2", nova_data);
}

//somando 900.12 a um campo de moeda chamado "moeda2"
var num_moeda = FormContext.GetDecimal("moeda2");
if(num_moeda != null){
  var novo_valor = num_moeda.Value + 9000000.12m;

  FormContext.SetValue("moeda2",novo_valor);
}
```

Salvando/Criar um outro formulário através do método SaveEntityAsync

* Para cadastrar valores numéricos, o números devem estar no formato com ponto no separador decimal: 999999.99 sem utilização de virgulas.

```
//guarda em uma variável o ID da  E S T R U T U R A  do formulário de pessoa
var idEstruturaFormulario_PESSOA = Guid.Parse("0b56b66a-4f6f-4ded-ad04-016d7c0724e1");

var values = new DataDictionary();
values.Add("nome", FormContext.GetValue("nome"));
values.Add("email", FormContext.GetValue("email"));



//salva a informação no banco de dados
await FormContext.SaveEntityAsync(idEstruturaFormulario_PESSOA, values);
```

Atualizar um outro formulário através do método SaveEntityAsync

* Para cadastrar valores numéricos, o números devem estar no formato com ponto no separador decimal: 999999.99 sem utilização de virgulas.

```
//guarda em uma variável o ID da  E S T R U T U R A  do formulário de pessoa
var idEstruturaFormulario_PESSOA = Guid.Parse("0b56b66a-4f6f-4ded-ad04-016d7c0724e1");

var values = new DataDictionary();
values.Add("id", "33f6bdf8-26d9-42b9-94ae-ad44c62725b5"); //insere aqui o ID do registro que você deseja atualizar
values.Add("nome", FormContext.GetValue("nome"));
values.Add("email", FormContext.GetValue("email"));



//salva a informação no banco de dados
await FormContext.SaveEntityAsync(idEstruturaFormulario_PESSOA, values);
```

> Caso você queira atualizar informações do próprio formulário que está sendo salvo, utilize a função SetValue, descrita mais acima nos métodos da Classe FormContext

Aprovar um outro formulário através do método ApproveEntityAsync

* Para cadastrar valores numéricos, o números devem estar no formato com ponto no separador decimal: 999999.99 sem utilização de virgulas.

```
//guarda em uma variável o ID da  E S T R U T U R A  do formulário de pessoa
var idEstruturaFormulario_PESSOA = Guid.Parse("0b56b66a-4f6f-4ded-ad04-016d7c0724e1");

var values = new DataDictionary();
values.Add("nome", FormContext.GetValue("nome"));
values.Add("email", FormContext.GetValue("email"));



//salva a informação no banco de dados
await FormContext.ApproveEntityAsync(idEstruturaFormulario_PESSOA, values);
```

Salvando um outro formulário com Tabela Associativa através do método SaveEntityAsync

* Para cadastrar valores numéricos, o números devem estar no formato com ponto no separador decimal: 999999.99 sem utilização de virgulas.

```
//guarda em uma variável o ID da  E S T R U T U R A  do formulário de pessoa
var idEstruturaFormulario_PESSOA = Guid.Parse("0b56b66a-4f6f-4ded-ad04-016d7c0724e1");

var values = new DataDictionary();
values.Add("nome", json["nome"].ToString());
values.Add("email", json["email"].ToString());



    //no caso do campo do formulário ser uma tabela associativa (tabela relacional N:N)
    var idCampoTabelaAssociativa = "70b6f362-2587-4fd2-a2fb-148bd0caf437";

    //itens da tabela associativa que serão adicionados
    var idItem1 = "51cf02f1-3787-4dca-8a2c-e219a5ce1298";
    var idItem2 = "f999f103-c775-4245-92d3-034cb3ded5e4";
    var idItem3 = "XXXXf103-c775-4245-92d3-034cb3ded5e4";
    var idItem4 = "YYYYf103-c775-4245-92d3-034cb3ded5e4";
    var idsJuntos_ConcatenadosComVirgula = idItem1 + "," + idItem2 + "," + idItem3 + "," + idItem4 + ",";

    //adiciona
    values.Add(idCampoTabelaAssociativa, idItem1); //adiciona idItem1
    values.Add(idCampoTabelaAssociativa, idItem2); //adiciona idItem2
    values.Add(idCampoTabelaAssociativa, idItem3); //adiciona idItem3
    values.Add(idCampoTabelaAssociativa, idItem4); //adiciona idItem4
    values.Add("tabela_associativa_added-"+idCampoTabelaAssociativa,idsJuntos_ConcatenadosComVirgula);//
itens concatenados com virgula
    values.Add("tabela_associativa_removed-"+idCampoTabelaAssociativa,"");//em branco
    values.Add("tabela_associativa_colunas_adicionais-"+idCampoTabelaAssociativa,"");//em branco
    //----------------------------------------



//salva a informação no banco de dados
await FormContext.SaveEntityAsync(idEstruturaFormulario_PESSOA, values);
```

Para recuperar o Texto de um campo que seja uma Lista Dinâmica (Combo, Auto completar, Múltipla seleção, Radio):

Suponhamos que você tenha um formulário de Pedido e nesse formulário você tenha um campo chamado 'Produto'.
Esse campo é do tipo 'Pesquisa com Auto Completar' e listará os 'Produtos' por 'Nome'.

Quando o usuário salvar o formulário, no Json de envio não retornará o Nome do Produto, apenas o Id do produto selecionado (Esse id estará no formato de um GUID).
Para recuperar o Nome do produto, você precisará executar o método **GetText** da seguinte forma:

```
var nomeProduto = FormContext.GetText("produto");
```

Recuperar os valores das "Variáveis de Ambiente"

Para saber mais sobre variáveis de ambiente entre em Variáveis de ambiente

```
var valorDaVariavel = FormContext.GetEnvironmentVariable("var_nomeDaVariavel");
```

Recuperar a Descrição (texto), no caso de variáveis do Tipo "Query com Id + Descrição":

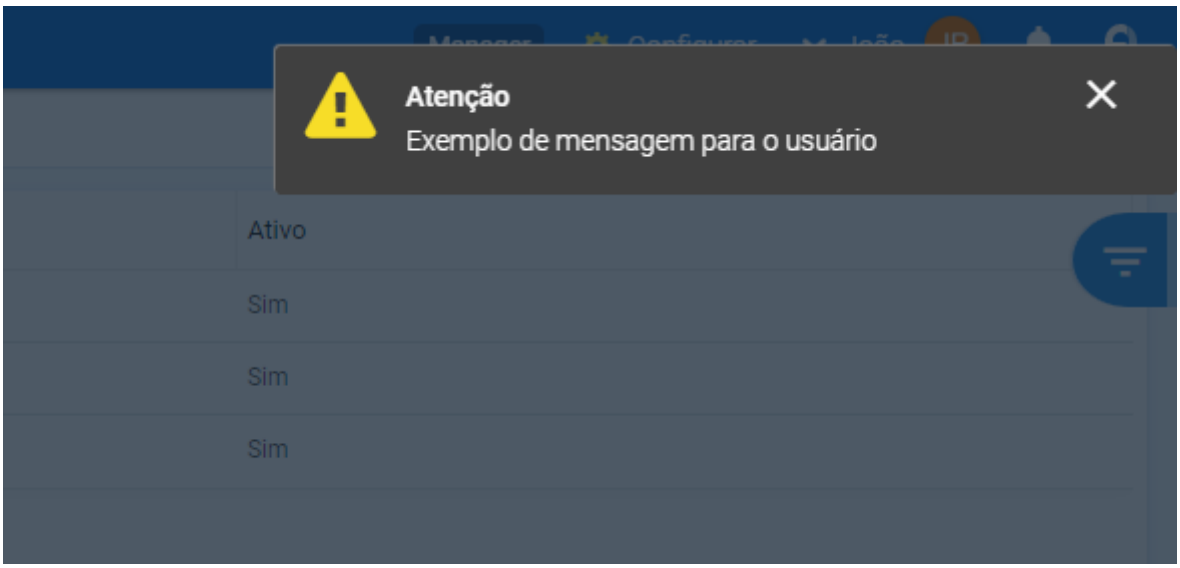
```
var textDaVariavel = FormContext.GetEnvironmentVariable_Text("var_nomeDaVariavel");
```

Enviar Mensagem para o usuário: A execução do código abaixo apresentará uma mensagem para o usuário

```
FormContext.WarningMessage("Exemplo de mensagem para o usuário");
```

> O código C# se encerrará no exato momento em que for chamado o método para apresentação de mensagem.

Retorno da mensagem



Envio de e-mail

Responsável pelo envio de e-mail

Método	**Retorno**	
EmailSender.SendToGrupoUsuario(Guid grupoUsuarioId, string subject, string htmlContent)	void	Enviar um e-mail para um determinado Grupo de Usuario
EmailSender.SendEmail(string toEmail, string subject, string htmlContent)	void	Enviar um e-mail
EmailSender.SendEmail(string toEmail, string subject, string htmlContent, string ccEmail**)**	void	Enviar um e-mail, com copia (CC)

Enviar e-mail para um Grupo de Usuário

Antes de iniciar, crie um "Grupo de Usuário" no menu "Segurança e Acesso". Depois de criado, associe os usuários que receberão e-mail ao novo grupo de usuário que você acabou de criar.

Pegue o ID desse novo Grupo de usuário. Você vai precisar dele para enviar o e-mail.

```
var grupoUsuarioId = Guid.Parse("af7c9275-0e23-4b64-a433-f238bb457005"); //substitua o ID "af7c9275-0e23-4b64-a433-f238bb457005" pelo Id do grupo de usuario que você deseja enviar o e-mail
var assunto = "Novo E-mail para um Grupo de usuario ";
var html = "Olá Grupo, <BR><BR> Teste para envio de e-mail para um Grupo de Usuário no AgilityFlow!";

EmailSender.SendToGrupoUsuario(grupoUsuarioId, assunto, html);
```

Enviar e-mail

```
var htmlContent = "<strong>Olá,</strong> teste envio de e-mail.";
var emailTo = "john@email.com";
var subject = "Subject do E-mail";

EmailSender.SendEmail(emailTo, subject, htmlContent);
```

Revision #81
Created 19 March 2019 20:03:14 by agilityflow
Updated 5 March 2025 12:33:44 by agilityflow