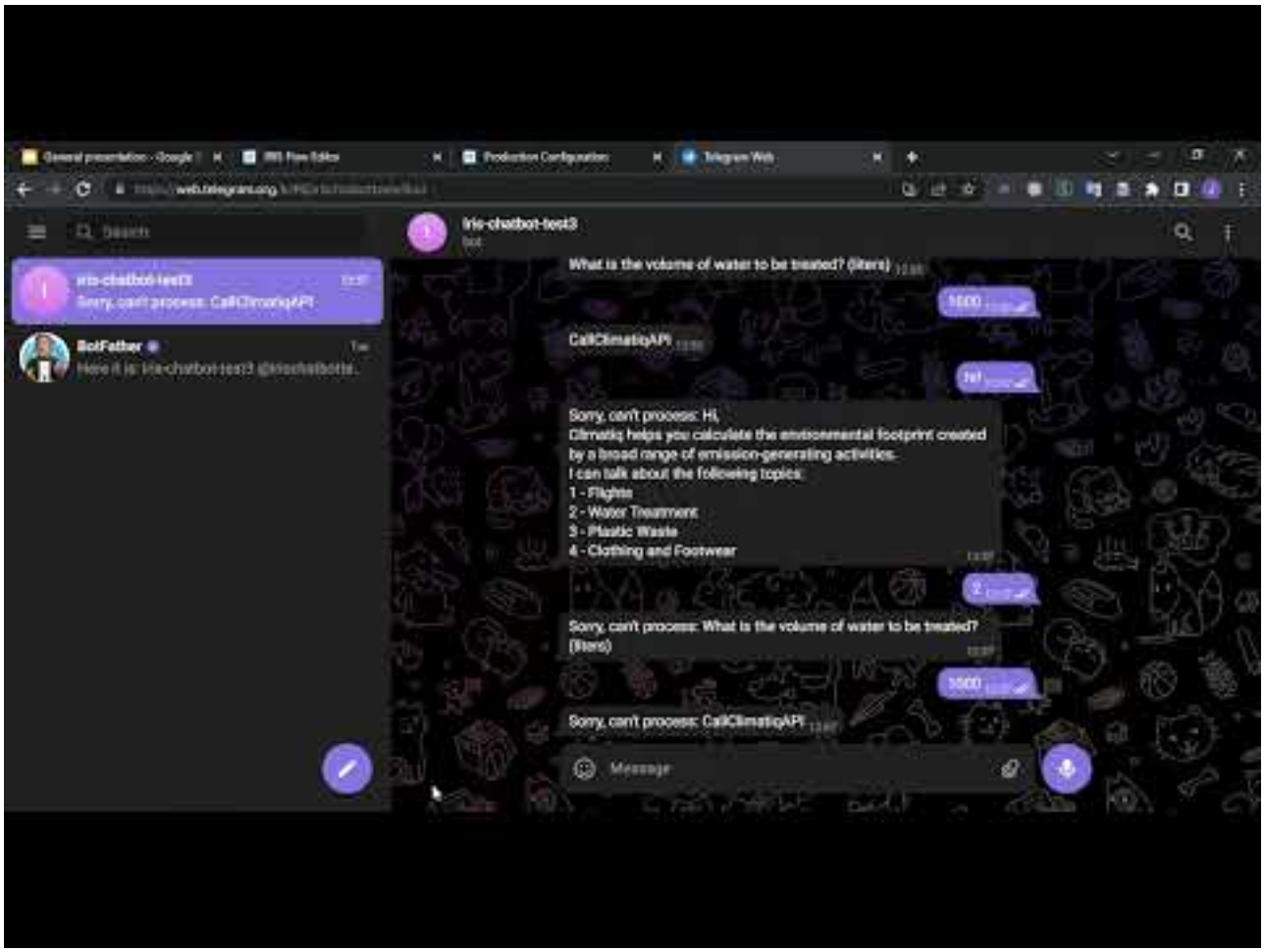
Artigo
José Pereira · Set. 16, 2022 10min de leitura

IRIS Flow - Atualizações para o Concurso de Interoperabilidade: Construindo Soluções Sustentáveis

TL;DR: você pode experimentar o chatbot criado com o IRIS Flow aqui; se você quiser entender como ele foi criado, leia este artigo.

Se preferir, você pode seguir o tutorial deste artigo neste vídeo:



Introdução

Algum tempo atrás, Henrique, Henry e eu começamos o projeto IRIS-Megazord para juntar vários de nossos projetos e apresentar o IRIS Flow - uma ferramenta para criação de fluxos de automação suportados pelo framework do IRIS Interoperability.

Seguindo o tema atual do concurso - Soluções Sustentáveis, decidimos incrementar os recursos de automação no IRIS Flow, adicionando um novo adaptador para uso da API Climatiq. Essa API é um serviço beta que permite fazer estimativas de quanto emissões de CO2 várias atividades humanas podem gerar.

Assim, com este novo recurso e utilizando os anteriores, que nos permitem utilizar recursos como comunicação

Telegram e execução de código ObjectScript, criamos um chatbot simples para tirar dúvidas sobre emissões de CO2.

Usando o EnsLib.HTTP.OutboundAdapter para acessar a API do Climatiq

Para usar a API do Climatiq, nosso adaptador estende o EnsLib .HTTP.OutboundAdapter. Este adaptador ajuda você a fazer solicitações HTTP para servidores externos, como um serviço de API REST por exemplo - a API REST Climatiq no nosso caso.

Ao herdar a classe EnsLib.HTTP.OutboundAdapter, precisamos apenas configurar informações básicas como URL da API, porta para HTTPS e um cabeçalho com a API KEY. Depois disso, você pode usar o método SendRequest para realizar o POST na API.

Você pode ver como usamos o EnsLib.HTTP.OutboundAdapter para desenvolver nosso adaptador personalizado nas classes [ClimatiqOutboundAdapter.cls](https://github.com/jrpereirajr/iris-megazord/blob/master/src/dc/irisflow/interoplib/climatiq/ClimatiqOutboundAdapter.cls) e [dc.irisflow.interoplib.climatiq.ApiBeta3](https://github.com/jrpereirajr/iris-megazord/blob/master/src/dc/irisflow/interoplib/climatiq/ApiBeta3.cls).

Experimentando o exemplo de chatbot simples do Climatiq online

Se você está curioso sobre nosso aplicativo de exemplo, implantamos uma demonstração online para mostrá-lo em execução. Você pode acessá-lo aqui.

Para usá-lo, basta enviar qualquer mensagem e aguardar as instruções.

Como configurar o exemplo do bot Climatiq em seu próprio ambiente

Se você quiser experimentar o exemplo do chatbot em seu próprio contêiner docker, primeiro você precisa criar uma conta nas APIs do Telegram e Climatiq. Você pode fazer isso seguindo as instruções descritas aqui para Telegram e [aqui para Climatiq] (https://www.climatiq.io/docs/guides/getting-api-key).

Depois de obter as chaves de API para cada serviço, armazene-as em credenciais de interoperabilidade IRIS. Para a chave da API do Telegram, crie uma credencial com o ID telegram-api-key; para o Climatiq, crie uma credencial identificada por climatiq-api-key. Esses IDs são os esperados pela produção pré-configurada que você vai importar.

Agora, você pode importar a produção executando as seguintes etapas:

1) Instale o projeto usando o docker conforme descrito aqui
2) Abra um terminal IRIS:

```
docker exec -it iris-megazord_iris_1 bash
irissession iris
```

3) Execute este comando do ObjectScript no namespace USER que criará a produção:

```
Do ##class(dc.irisflow.demo.ClimatiqAPIExample02).Create()
```

4) Abra e inicie a produção User.ClimatiqAPIExample
5) Abra o bot do Telegram criado anteriormente e comece a enviar mensagens para ele

Criando o exemplo passo a passo no editor do IRIS Flow

Nesta seção, vamos criar o chatbot da API Climatiq passo a passo. Então, você pode conferir como é fácil usar o IRIS Flow! :)

Primeiro você tem que escolher um nome para o seu fluxo. Neste exemplo vamos escolher User.ClimatiqAPIExample.

Como estamos planejando usar a API do Telegram como nosso serviço de mensagens instantâneas, vamos começar adicionando apenas dois nós ao nosso fluxo: um para ouvir um bot do Telegram e outro para enviar mensagens para o mesmo bot. Nomeie esses nós como FromTelegram e SendToTelegram, respectivamente.

O bot do Telegram é identificado por sua chave de API armazenada em uma credencial com um ID definido no campo Credentials nos nós do Telegram. Observe que nossos nós usam uma credencial chamada telegram-api-key, portanto, você deve definir nessa credencial sua chave de API do Telegram para que as coisas funcionem.

Agora, clique no botão Gerar. Se tudo der certo, você poderá ver uma produção da IRIS Interoperability chamada User.ClimatiqAPIExample02. Abra-a e você deverá ver algo assim:

Inicie a produção e comece a enviar mensagens para o seu bot do Telegram. Você verá que nossa produção envia de volta o que você digitou.

Esse comportamento se deve ao fato de que nosso fluxo apenas recebe a entrada obtida do Telegram pelo nó FromTelegram e a envia para o nó SendToTelegram, sem nenhum tratamento. Você pode verificar isso pelo IRIS Interoperability Message Viewer:

Ok, agora vamos adicionar mais complexidade ao nosso fluxo. Como pretendemos criar um chatbot, precisaremos processar alguma interação com os usuários. Para esta demonstração, desenvolvemos uma simples árvore de decisão para fornecer um conjunto básico de recursos aos nossos usuários.

Essa árvore de decisão é tratada por uma classe de utilitário que possui métodos de classe para usar tal árvore de decisão e, fornecer uma interface a ser usada dentro de nós do tipo ObjectScriptOperation em nosso fluxo.

Assim, vamos adicionar um nó ObjectScriptOperation, chamá-lo de ChatbotDecisionTree, excluir o link anterior entre FromTelegram e SendToTelegram e vinculá-los conforme a imagem abaixo:

Defina as propriedades Expression e ContextExpression com o Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).ChatbotDecisionTree(input, context) e Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils). ChatbotDecisionTreeContext(input, context), respectivamente. Estes são métodos utilitários que formatam a entrada e a enviam para a árvore de decisão.

Salve essa alteração no fluxo, reinicie a produção e envie outra mensagem para seu bot do Telegram. Agora, você deve ver nossa árvore de decisão do chatbot apresentando um conjunto de opções para o usuário:

Você pode notar que nosso chatbot termina a conversa com um token CallClimatiqAPI. Isso porque ele foi projetado para funcionar com a API Climatiq. Esse token significa que uma mensagem válida está pronta para ser enviada à API do Climatiq.

Como temos nossa árvore de decisão tratando de interações com usuários e coletando informações para serem enviadas para a API do Climatiq, vamos adicionar um nó que seja capaz de conversar com tal API: o nó ClimatiqOperation. Adicione-o ao fluxo, nomeie-o como CallClimatiqAPI e refaça os links como abaixo:

Reinicie a produção, repita o diálogo anterior para o chatbot. Você verá que agora estamos recebendo mensagens de erro como Sorry, can't process: blahblah.

Esta mensagem é gerada pelo nó CallClimatiqAPI porque este espera alguma mensagem JSON seguindo o esquema da API Climatiq. Então, vamos resolver isso.

Aqui, é importante observar que temos dois tipos de mensagens para tratar: uma para interações do usuário e outra para chamadas de API. O primeiro tipo de mensagem deve ser enviado diretamente aos usuários, enquanto o segundo deve ser enviado primeiro para a API do Climatiq. As mensagens de chamada de API são identificadas pelo token CallClimatiqAPI.

Então, vamos adicionar dois novos nós ObjectScriptOperation para implementar essa lógica em nosso fluxo.

Para o primeiro nó, nomeie-o como HandleClimatiqApiCalls e defina suas propriedades Expression e ContextExpression com Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).HandleClimatiqApiCalls(input, context) e Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).HandleClimatiqApiCallsContext(input, context), respectivamente.

Para o segundo nó, faça como o anterior, mas desta vez usando os valores HandleUserInteration, Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).HandleUserInteration(input, context) e Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).HandleUserInterationContext(input, context).

Por fim, refaça os vínculos de todos os nós como abaixo:

Reinicie a produção, repita o diálogo anterior para o chatbot. Agora, você verá a interação normal do chatbot com o usuário e uma resposta JSON da API Climatiq em vez do token CallClimatiqAPI quando a conversa terminar:

Vamos verificar o rastreamento de mensagens para mensagens de interação de usuários e interações de API:

Observe que em ambos os casos, o nó ChatbotDecisionTree envia a mesma mensagem para os nós HandleClimatiqApiCalls e HandlerUserInteraction. Mas no primeiro rastreamento, apenas o nó HandlerUserInteraction continua enviando mensagens no fluxo. Por outro lado, no segundo trace, apenas o nó HandleClimatiqApiCalls continua enviando mensagens.

Isso ocorre porque o código invocado por esses nós testa o tipo da mensagem - interação do usuário ou chamada de API e, cancela o fluxo com base nesse teste. Abaixo, um trecho da lógica para lidar com o tipo de mensagens:

```
ClassMethod IsClimatiqApiCall(pInput As %String) As %Boolean
{
    Return $FIND(pInput, ##class(dc.irisflow.demo.ChatbotDecisionTree).#CallClimatiqA
PI) > 0
}

ClassMethod HandleUserInteration(input As %String, context As %DynamicObject) As %Str
ing
{
    Set response = input
    If (..IsClimatiqApiCall(input)) {
        Return ##class(dc.irisflow.components.misc.ObjectScriptOperation).#CancelSend
Request
    }
    Return response
}

ClassMethod HandleClimatiqApiCalls(input As %String, context As %DynamicObject) As %S
tring
{
    Set response = ""
    If ('..IsClimatiqApiCall(input)) {
```

```
        Return ##class(dc.irisflow.components.misc.ObjectScriptOperation).#CancelSend
Request
    }
    Return response
}
```

Assim, a última operação que precisamos fazer para finalizar nosso chatbot é tratar a resposta JSON da API. Para este propósito, adicione um novo nó ObjectScriptOperation, nomeado como FormatResponse, defina sua Expression e ContextExpression como Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).FormatResponse(input, context) e Return ##class(dc.irisflow.demo.ClimatiqAPIExample02Utils).FormatResponseContext(input, context), e refaça os vínculos como na imagem abaixo:

E pela última vez neste artigo, reinicie a produção, repita o diálogo anterior para o chatbot. Agora, você poderá ver uma caixa de diálogo completa com uma resposta tratada:

Conclusão

Neste artigo apresentamos o novo nó para o IRIS Flow, o nó ClimatiqOperation. Este nó também usa um novo adaptador personalizado para usar a API REST do Climatiq.

Usando este novo nó e os anteriores no IRIS Flow, conseguimos criar um chatbot simples no Telegram, que pode informar aos usuários quantas emissões de CO2 suas atividades podem gerar.

Esperamos que este artigo possa inspirá-lo a criar ótimos aplicativos e estar ciente de sua pegada de CO2 também! :)

#InterSystems IRIS

URL de origem:https://pt.community.intersystems.com/post/iris-flow-atualiza%C3%A7%C3%B5es-para-o-concurso-de-interoperabilidade-construindo-solu%C3%A7%C3%B5es-sustent%C3%A1veis