# **Projeto Final**

#### Assunto da aula

Esta aula te mostrará como realizar a leitura e escrita de arquivos, como fazer requisições para páginas web e como coletar informações específicas de um site.

Depois será realizado o desenvolvimento de um projeto especial, finalizando o conteúdo deste curso.

#### Conteúdo

* Leitura e Escrita de Arquivos
* Chamadas Web
* Web Scraping
* Projeto Final

#### Leitura e Escrita de Arquivos

O ruby torna sua vida mais fácil oferecendo uma forma simples para **leitura e escrita** de **arquivos**.

Através de poucas instruções, é possível **ler**o conteúdo, ou então, **escrever**algumas informações dentro de um **arquivo externo**. Isso é feito por meio de uma classe chamada **File,** a qual conta com diversas opções para **manipulação de arquivos**.

### Leitura de arquivo

Para aprender a ler arquivos, faça uma lista de compras em um arquivo de texto e depois leia cada item da lista usando um programa Ruby.

1- Crie um arquivo chamado **list.txt** com o seguinte texto

|  |  |
| --- | --- |
| 1  2  3 | morango  brocolis  palmito |

2- Em um arquivo com o nome **read.rb** adicione o código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | puts '--Lista de Compras--'    file = File.open('list.txt')    file.each do |line|   puts line  end |

Você utilizou o método **open** da classe **File**para acessar o arquivo **list.txt.** Depois o método **each** para percorrer e exibir o conteúdo do arquivo.

### Escrita de Arquivos

Adicione mais itens à lista de compras e aprenda como **inserir conteúdo** dentro de um arquivo.

1- Crie um arquivo chamado **write.rb**com o código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | File.open('shopping-list.txt', 'a') do |line|   line.puts('arroz')   line.puts('feijão')   line.print('azeite')   line.print(' de ')   line.print('oliva')  end |

Para adicionar conteúdo sem sobrescrever o que já existe , foi necessário passar o argumento **‘a’.**Este argumento significa **append,**ou seja,  acrescentar conteúdo.

Perceba que o método **puts**e **print** tem o mesmo significado do que em outros  exemplos deste curso. Os dois inserem valores ao arquivo, com a diferença de que apenas o**puts** insere uma **nova linha** após a string.

2- Por curiosidade, caso queira saber o tamanho do arquivo gerado abra o **irb** e faça

|  |  |
| --- | --- |
| 1 | File.open('shopping-list.txt').size |

O método **size** retorna o tamanho do arquivo em **bytes.**

3- Para substituir o conteúdo desse arquivo utilize o argumento **‘w’**

|  |  |
| --- | --- |
| 1  2  3 | File.open('shopping-list.txt', 'w') do |line|    line.puts('batata')  end |

Após executar este programa, a lista de compras contará apenas com o item batata.

#### Chamadas Web

Talvez sua aplicação precise recuperar informações, enviar formulários, ou então, enviar documentos para websites.

Para isso o ruby conta com uma biblioteca chamada **Net::HTTP** que é capaz de realizar chamadas web.

### Requisições http

1- Crie um arquivo chamado **web\_get.rb** com o código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | require 'net/http'    example = Net::HTTP.get('example.com', '/index.html')    File.open('example.html', 'w') do |line|   line.puts(example)  end |

**O que aconteceu?**

* No início, é preciso  **adicionar** a biblioteca **Net:HTTP**
* Depois é feito uma requisição **HTTP** do tipo **GET**para o domínio **example.com**, com o caminho **/index.html**.

É a mesma coisa que entrar no browser e procurar pelo endereço [**http://example.com/index.html**](http://example.com/index.html)

* Você salvou a **resposta** dentro de uma variável para depois escrevê-la dentro de um arquivo.
* Quando você utilizou  **File.open** para um **arquivo inexistente**, o **ruby detectou** isso e **criou o arquivo** antes de escrever as informações nele.

### Requisições https

Para fazer as próximas requisições  você **utilizará o site** [https://reqres.in](https://reqres.in/)

Ele está preparado para **receber e responder** suas **requisições**, o que facilitará os testes de requisições **https**.

1- Faça uma requisição **https**utilizando o método use\_ssl

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | require 'net/http'    https = Net::HTTP.new('reqres.in', 443)  # para fazer chamadas https  https.use\_ssl = true    response = https.get("/api/users")  # status code  puts response.code  # status message  puts response.message  # body (json)  puts  response.body |

**O que aconteceu?**

* Você **construiu** um objeto **Net::HTTP** iniciando com os valores de  **domínio** e **porta**
* Você informou que é uma requisição que utiliza o certificado ssl com o método **use\_ssl = true**
* Fez um get para o caminho **api/users**
* Por fim exibiu o código da resposta, o status e o corpo da resposta

2- Agora, crie um arquivo chamado **web\_post.rb** com o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | require 'net/http'    req = Net::HTTP::Post.new("/api/users")  # para fazer chamadas https  req.set\_form\_data({ name:'Mario', job:'Encanador' })    response = Net::HTTP.start('reqres.in', use\_ssl: true) do |http|   http.request(req)  end    puts response.code  puts response.body |

**O que aconteceu?**

* Vocẽ inicializou um objeto **Net::HTTP::Post.new()** passando como argumento o caminho da url
* Depois informou quais são os  parâmetros a serem enviados com o método **set\_form\_data**
* Por fim utilizou o método **start** para criar uma conexão com o servidor e dentro de um bloco fez a requisição **POST**

#### Web Scraping

O que você faria se precisasse percorrer diversas páginas de um site coletando um tipo específico de informação? Ou recuperar dados que não são fornecidos por uma API?

O Web Scraping existe para evitar essas tarefas manuais, ou então, trabalhosas. Uma técnica que automatiza o processo de extração de dados de uma página web.

Uma das bibliotecas mais utilizadas no mundo ruby para fazer isso é a **nokogiri**

1- **Instale** a gem **nokogiri** com o comando

|  |  |
| --- | --- |
| 1 | gem install nokogiri |

2- Crie um projeto chamado **web\_scraping.rb**com o seguinte código

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | require 'nokogiri'  require 'net/http'    https = Net::HTTP.new('onebitcode.com', 443)  # para fazer chamadas https  https.use\_ssl = true    response = https.get("/")    doc = Nokogiri::HTML(response.body)    h1 = doc.at('h1')    puts h1.content |

**O que aconteceu?**

* No começo nenhuma novidade. Você realizou uma **requisição** para o site do **onebitcode**
* Depois utilizou **Nokogiri::HTML** para **parsear** o documento **HTML**
* Por fim, você fez uso do método **at**para buscar a tag **h1**e imprimir o seu conteúdo.

2-  O ultimo post do onebitcode encontra-se dentro das tags **<h3> <a href=’/ultimo-post’>Título Ultimo Post</a></h3>**

Para buscar este post faça

|  |  |
| --- | --- |
| 1  2  3 | last\_post = doc.at('h3 a')  puts last\_post.content  puts last\_post['href'] |

3- Você também pode procurar por **todos os posts** que estão na página inicial do onebitcode com o **método search**

|  |  |
| --- | --- |
| 1  2  3  4  5 | doc.search('h3 a').each do |a|    puts a.content    puts a['href']    puts ''  end |

#### Projeto Final

Crie um software que permita que o usuário entre com um texto, o idioma em que o texto está escrito e também o idioma para qual ele deseja traduzir o texto e receba como resposta na tela o texto traduzido.

**Critérios:**

* Você deve utilizar orientação a objetos no seu software
* O resultado de cada tradução (junto com a frase original) deve ser guardado em um arquivo que possua no nome a data e horário da tradução (exp: 10-10-18\_10:30.txt)

**[Recursos]**

* Para aprender mais sobre APIs, Json e etc acompanhe nosso curso gratuito: [Criando uma API completo com Ruby On Rails](https://onebitcode.com/course/criando-uma-api-completa-com-rails/)
* Para realizar as chamadas Web utilize a Gem Rest Client (documentação): <https://github.com/rest-client/rest-client>
* Para realizar a tradução utilize a API do Yandex.
  + Crie sua conta e pegue seu Token de autenticação nesta URL: <https://translate.yandex.com/developers/keys>
  + Veja como realizar a chamada nessa URL: <https://tech.yandex.com/translate/doc/dg/reference/translate-docpage/>