CETEJ35 - Java Web - JAVA_XXX (2024_01)

API Reativa

✓ **Feito:** Ver | **A fazer:** Gastar pelo menos 20 minutos na atividade | **A fazer:** Passar pela atividade até o fim

Fecha: segunda-feira, 2 dez. 2024, 00:00

Nesta aula vamos construir uma nova aplicação que funciona como uma API, recebendo dados usando a arquitetura REST. Para fazer isso, vamos usar uma nova tecnologia do Spring - o Spring WebFlux.

GERENCIADOR DE TAREFAS

Um gerenciador de tarefas (TODO) é um aplicativo que permite controlar as tarefas que devem ser executadas para atingir algum propósito. Nosso gerenciador de tarefas deve permitir criar e apagar tarefas, além de marcar se uma tarefa foi concluída. Além disso, esse gerenciador de tarefas deve funcionar como uma API.

API é o acrônimo de *Aplication Programming Interface*. Uma API funciona como a porta de entrada de uma aplicação. Originalmente, APIs foram criadas para permitir a integração entre interface gráfica e o restante da aplicação. Atualmente, APIs são fundamentais para permitir a integração de aplicações em sistemas orientados a serviços. Frequentemente, APIs são implementadas por frameworks ou bibliotecas. APIs podem ser desenvolvidas usando protocolos, tecnologias e arquiteturas diversas, como RMI, WebSocket, ou REST.

Neste curso, vamos desenvolver o gerenciador de tarefas usando uma API REST. APIs REST fazem requisições usando os métodos do protocolo HTTP para identificar o tipo de operação que será executada. Por exemplo, listar um conjunto de tarefas usa o método `GET`, enquanto a criação de uma tarefa usa o método `POST`.

Uma URL identifica o recurso que sobre o qual a operação será realizada. Por exemplo, podemos usar a URL `/todos` para identificar a tarefa, e podemos usar a URL `/user` para identificar o usuário. Se quisermos identificar um recurso específico, podemos adicionar um caminho à URL. Por exemplo, `/todos/a8sd7f687df` pode identificar uma tarefa em particular. Por fim, é possível enviar um corpo junto à requisição. Normalmente, isso é feito especificando um objeto com o formato `JSON`.

> Quer saber mais sobre APIs REST? Esse **site** tem uma coleção de links úteis sobre o assunto.

Como uma API, o gerenciador de tarefas não tem interface gráfica. Por isso, para realizar as operações vamos usar um cliente HTTP. Um cliente HTTP é um aplicativo que envia solicitações HTTP para uma determinada URL, assim como seu navegador de Internet. Se você usa Linux, já deve ter o cURL instalado - essa é uma ferramenta muito popular no Linux. O Postman também é uma ferramenta bastante conhecida e, possivelmente, a mais simples de usar. Se você tem pouca experiência usando clientes HTTP, eu sugiro que use o Postman. Para usuários Mac, o HTTPie é um excelente cliente HTTP. Contudo, você tem liberdade para usar o cliente que achar melhor.

O gerenciador de tarefas será desenvolvido como uma API reativa. Para isso, vamos usar o Spring Webflux e o MongoDB. O MongoDB é um banco de dados não-SQL (NoSQL) que armazena dados como documentos. No contexto do MongoDB, um documento é como um arquivo em formato `JSON`. Neste projeto, vamos usar o MongoDB porque, no momento em que esse texto está sendo escrito, o Spring Data para JPA ainda não oferece suporte para reatividade. Você não precisa conhecer de MongoDB, mas vai precisar ter ele instalado para executar este o projeto. Para isso, basta fazer o

download direto do site oficial e instalar de acordo com seu sistema operacional. Você também pode usar uma imagem Docker se preferir.

> Se você nunca usou MongoDB, não se preocupe. Toda a interação com o MongoDB será feita por dentro da aplicação usando o Spring Data, que você já conhece.

Precisamos criar um novo projeto usando o Spring Initializr, assim como foi feito anteriormente. Adicione as dependências *Spring Reactive Web* (Webflux) e *Spring Data Reactive MongoDB*. Faça o download do projeto e abra ele na sua IDE preferida. Não esqueça de iniciar o MongoDB, ou o container Docker antes de iniciar o projeto.

> Você também pode clonar o projeto vazio já com as dependências direto do **nosso repositório no Github**, na branch `semana06-10-projeto-vazio`.

[1] ALONSO , G.; CASATI, F.; KUNO, H.; MACHIRAJU, V. Web Services: Concepts, Architectures and Applications. Berlin: Springer, 2004.

Retroceder Avançar

◄ Verificação de aprendizado - Integração

Seguir para...

Atividade II WebConf ►

Contate o suporte do site

Você acessou como RAFAEL ROCHA DA SILVA PROENCA (Sair)
CETEJ35 - Web (2024_01)

Tema
Adaptable
Boost
Clássico
Campus
Apucarana
Campo Mourão
Cornélio Procópio
Curitiba
Dois Vizinhos
Francisco Beltrão
Guarapuava
Londrina
Medianeira
Pato Branco
Ponta Grossa
Reitoria
Santa Helena
Toledo
UTFPR
Ajuda
Chat UTFPR
Calendário Acadêmico
Biblioteca
e-Mail
Nuvem (OwnCloud)
Produção Acadêmica
Secretaria Acadêmica
Sistemas Corporativos

Sistema Eletrônico de Informação - SEI

Suporte ao usuário

Criação de curso

Comunidade

Português - Brasil (pt_br)

Deutsch (de)

English (en)

Português - Brasil (pt_br)

Resumo de retenção de dados

Baixar o aplicativo móvel.

Dê um feedback sobre este software

Universidade Tecnológica Federal do Paraná - UTFPR

Suporte ao usuário