Introdução ao Uso de Serviços Web Usando REST

Prof. Julio Cesar Nardi.

Aplicação introdutória usando Spring Boot, Spring MVC, Spring Data JPA e o banco de dados H2, além do Spring Tool Suite 4 e o Maven visando apresentar, em linhas gerais, o uso de serviços web usando REST.

Tal aplicação, chamada MyLib, objetiva gerenciar títulos de livros de uma biblioteca pessoal. Teremos a classe *Titulo* (com os atributos *nome* e *isbn*) e a classe *Autor* (com os atributos *nome* e *cpf*).

Criando um projeto

No Spring Tool Suite 4, escolha *File* → *New* → *Spring Starter Project*.

Após isso, aparecerá a tela de configuração da aplicação com os respectivos campos, conforme apresentado na Figura 1.

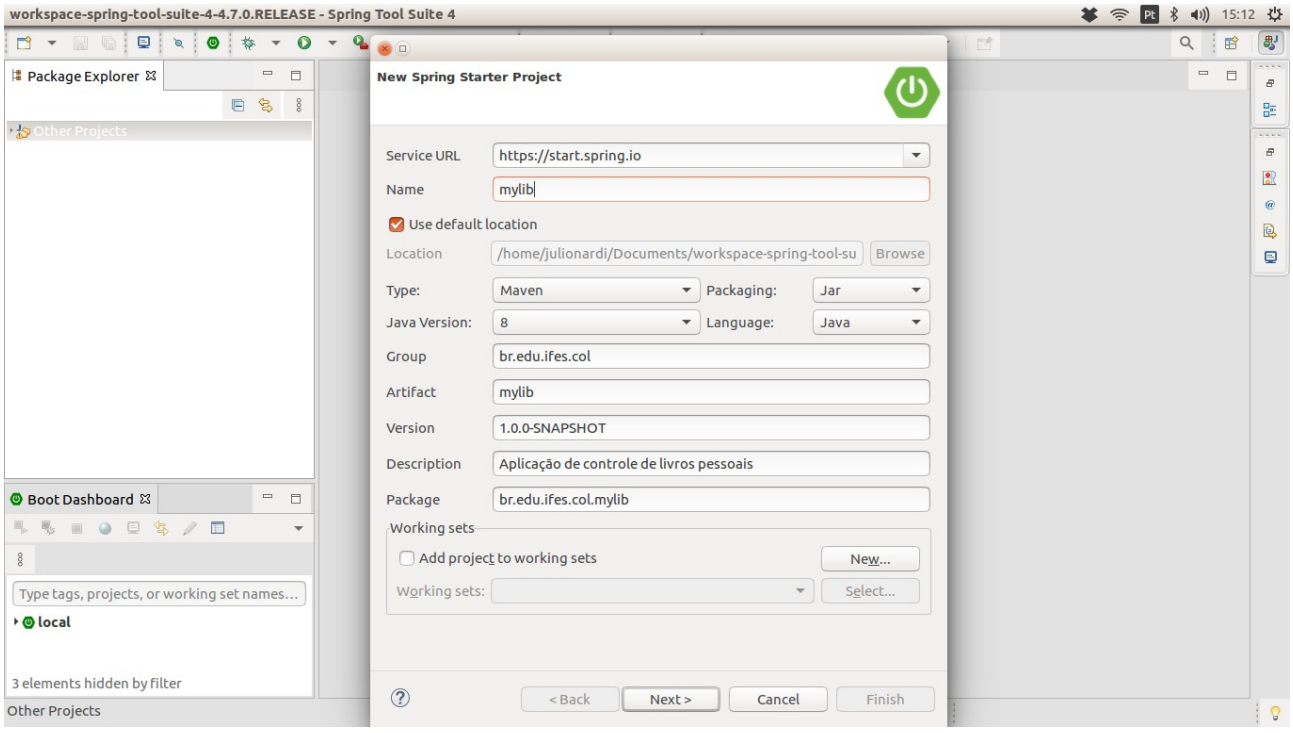

Figura 1: Tela de configuração da aplicação.

Vamos, agora, selecionar os *frameworks* e bibliotecas necessários à execução do projeto.

A Figura 2 apresenta a tela de configuração correspondente, já com os pacortes necessários ao nosso projeto. Observe que, por meio desta tela, podemos selecionar os pacotes necessários a qualquer aplicação e, assim, o STS gerará, automaticamente, o arquivo *pom.xml*, o qual contem as

ligações de dependência entre pacotes. No contexto deste projeto, vamos utilizar DevTools, Spring Web, Spring Data JPA e H2 Database.

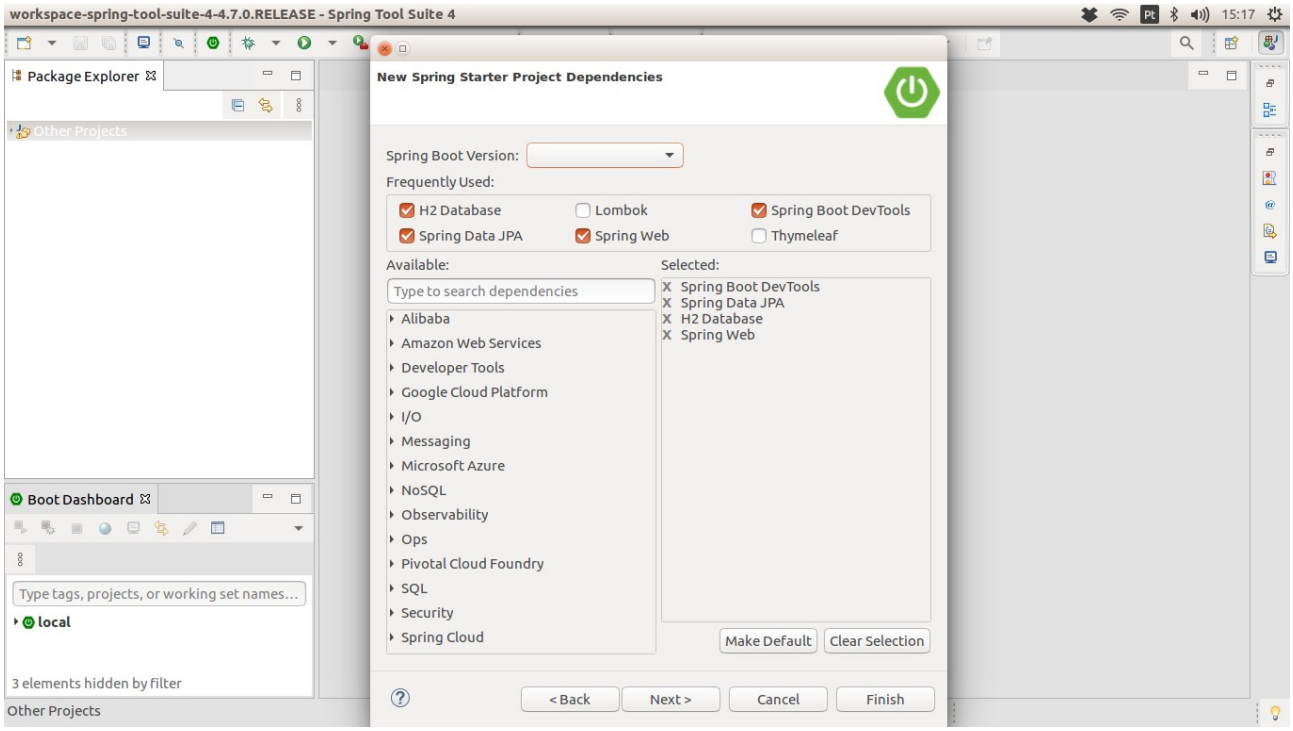

Figura 2: Tela de seleção de pacotes necessários à aplicação.

Ao clicar em *Finish*, o STS criará o projeto já com as configurações e toda a estrutura de pacotes, como pode ser observado na Figura 3.

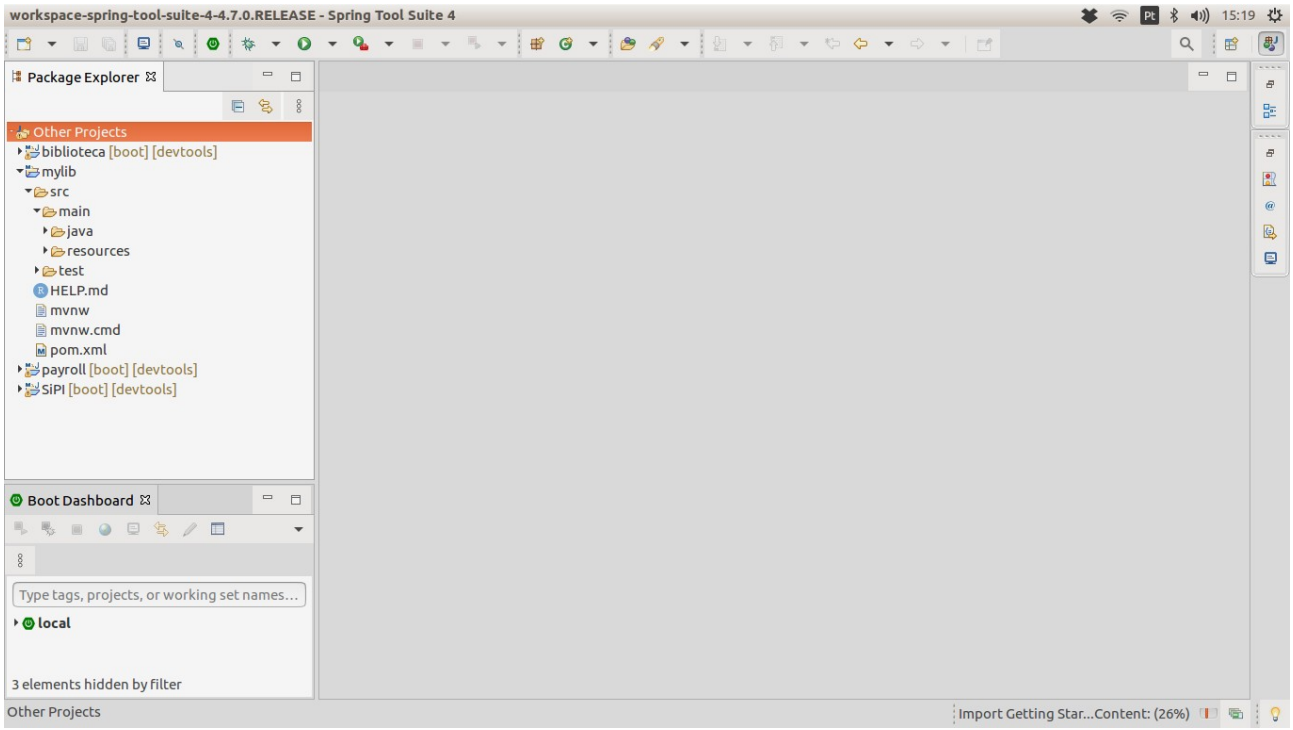

Figura 3: Tela inicial do projeto já criado.

Em *src/main/java*, dentro do pacote *br.edu.ifes.col.mylib* você encontra a classe *MylibApplication.java*, a qual é o ponto de início da nossa aplicação. Basta executar, portanto, o

método *main* desta classe. Observe que a classe *MylibApplication.java* está anotada com *@SpringBootApplication*.

Testando o projeto criado

Para testar a configuração do projeto e verificar se a aplicação inicia sem erros, *clique com o direito na classe MylibApplication* e selecione *Run As → 3 Spring Boot App*. Verifique no console se houve alguma mensagem de erro ou se o Spring iniciou corretamente, conforme mostra a Figura 4.

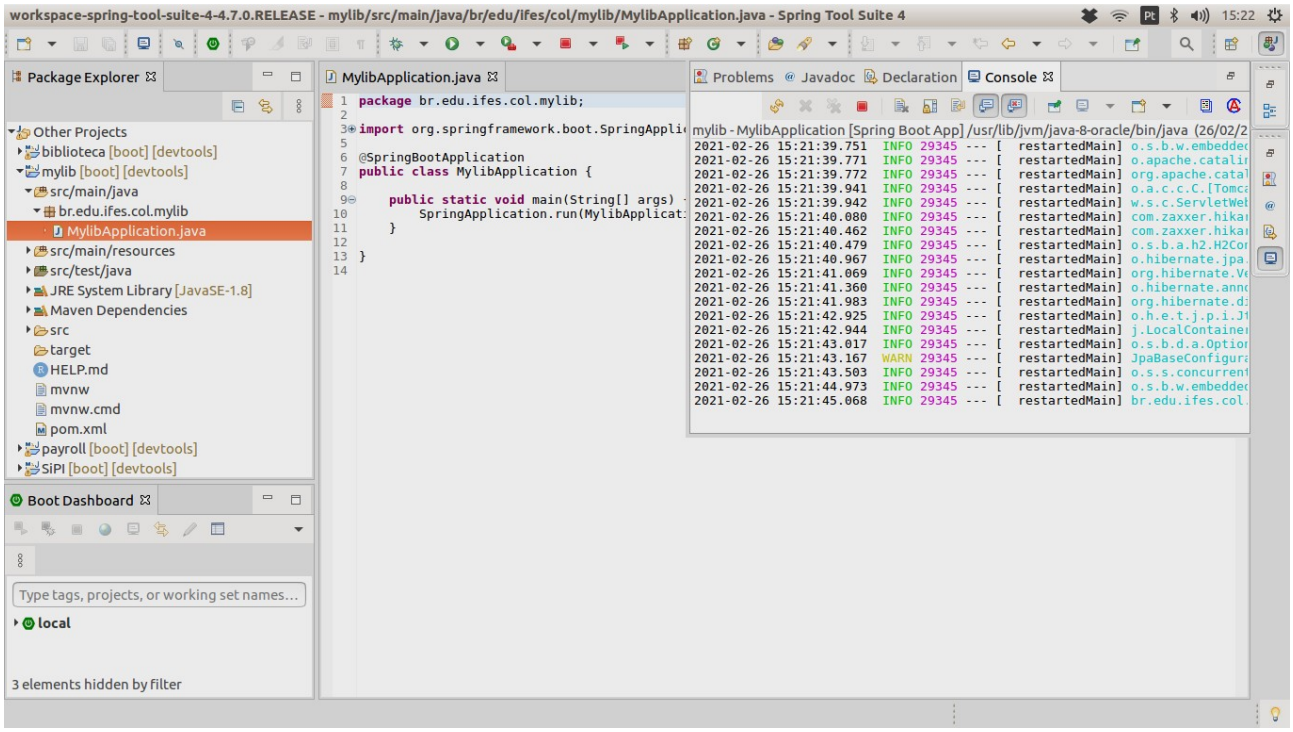

Figura 4: Tela do console onde são apresentadas mensagens de sucesso ou erro.

Criando a classe de domínio "Ator"

Dentro de *src/main/java*, crie o pacote *br.edu.ifes.col.mylib.domain*. Neste pacote colocaremos as classes de domínio do problema. Crie, agora, a classe Ator, conforme o código-fonte abaixo. Crie, também, a interface *AtorRepository*. Para facilitar, vamos coloque tal interface no mesmo pacote em que foi criada a classe *Ator* (embora isso não seja obrigatório).

```java
package br.edu.ifes.col.mylib.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Ator {

	@Id
	@GeneratedValue
	private Long id;

	private String nome;

	private String cpf;


	public Ator(){

	}

	// Getters and Setters seguem a partir de aqui

}
```

```java
package br.edu.ifes.col.mylib.domain;

import org.springframework.data.jpa.repository.JpaRepository;

public interface AtorRepository extends JpaRepository<Ator, Long>{

}
```

Criando o primeiro serviço web

Dentro de *src/main/java*, crie o pacote *br.edu.ifes.col.mylib.resources*. Neste pacote colocaremos os serviços web REST que criaremos ao longo deste tutorial.

Agora, crie uma nova classe chamada *AtorResource* por meio do menu *File → New → Class*. Abaixo você pode ver o código-fonte dessa classe.

Note que após criar a classe, você deve anotá-la com @Service e @RestController, a fim de que o Spring a reconheça como um serviço REST.

Observe também que:

1. a linha @RequestMapping(produces = MediaType.***APPLICATION_JSON_VALUE***) define que os retornos do métodos de toda a classe serão em formato JSON.

2. a linha @GetMapping (value="/mylib/resouces/atores") define o endereço, o padrão de acesso ao serviço e o método HTTP correspondente (nesse caso, usamos GET).

```java
package br.edu.ifes.col.mylib.resouces;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.stereotype.Service;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import br.edu.ifes.col.mylib.domain.Ator;
import br.edu.ifes.col.mylib.domain.AtorRepository;

@Service
@RestController
@RequestMapping(produces = MediaType.APPLICATION_JSON_VALUE)
public class AtorResouce {

	@Autowired
	private AtorRepository repositorioAtores;

	@GetMapping (value="/mylib/resouces/atores")
	public List<Ator> obterAtores(){

		return repositorioAtores.findAll();

	}
}
```

Testando o serviço web criado

Para nossos testes, vamos usar o banco de dados H2, o qual trabalha em memória RAM. Lembre-se que o módulo de acesso/uso do H2 foi incluído no projeto, quando da sua criação.

Localize o arquivo *application.properties* na árvore de diretórios do seu projeto e insira as seguintes linhas para configuração de acesso ao seu banco de dados:

```
# H2
spring.h2.console.enabled=true
spring.h2.console.path=/h2

# Datasource
spring.datasource.url=jdbc:h2:mem:mylibbd
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.driver-class-name=org.h2.Driver
spring.jpa.hibernate.ddl-auto=update
```

Com isso, você tem acesso ao banco de dados por meio do *console* disponível via *browser*, através do caminho **http://localhost:8080/h2**. A Figura 5 apresenta a tela inicial do *console* para acesso ao banco de dados.

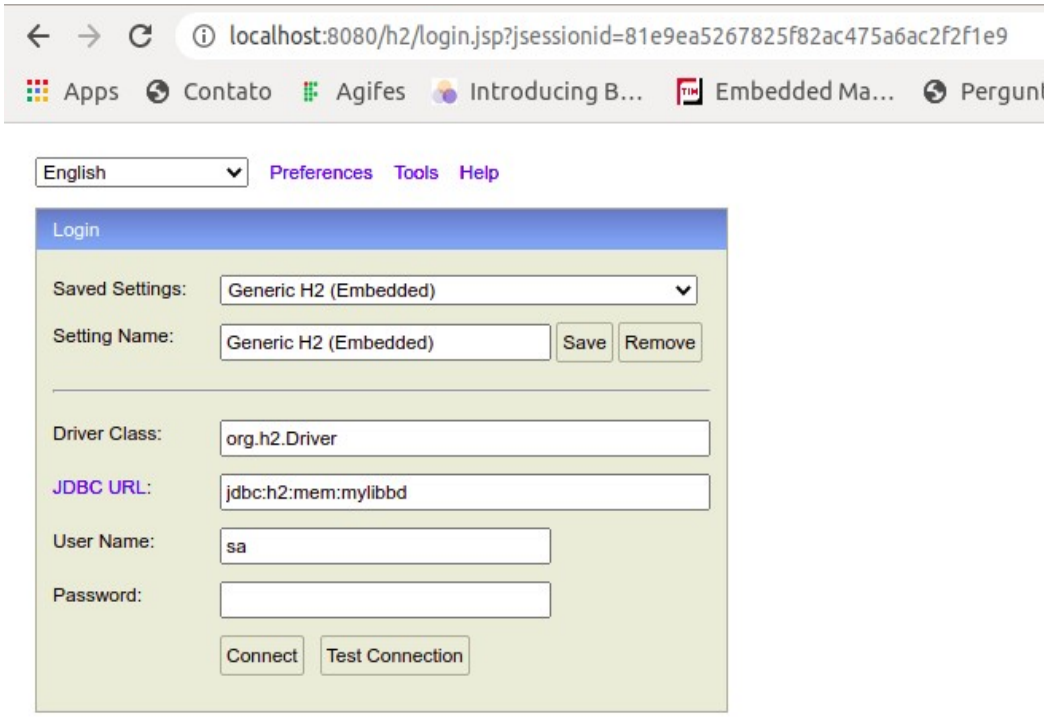

Figura 5: Tela inicial do console para acesso ao banco de dados.

Uma vez acessado o console e logado devidamente, você terá acesso à tela de administração do seu banco de dados, conforme apresenta a Figura 6.

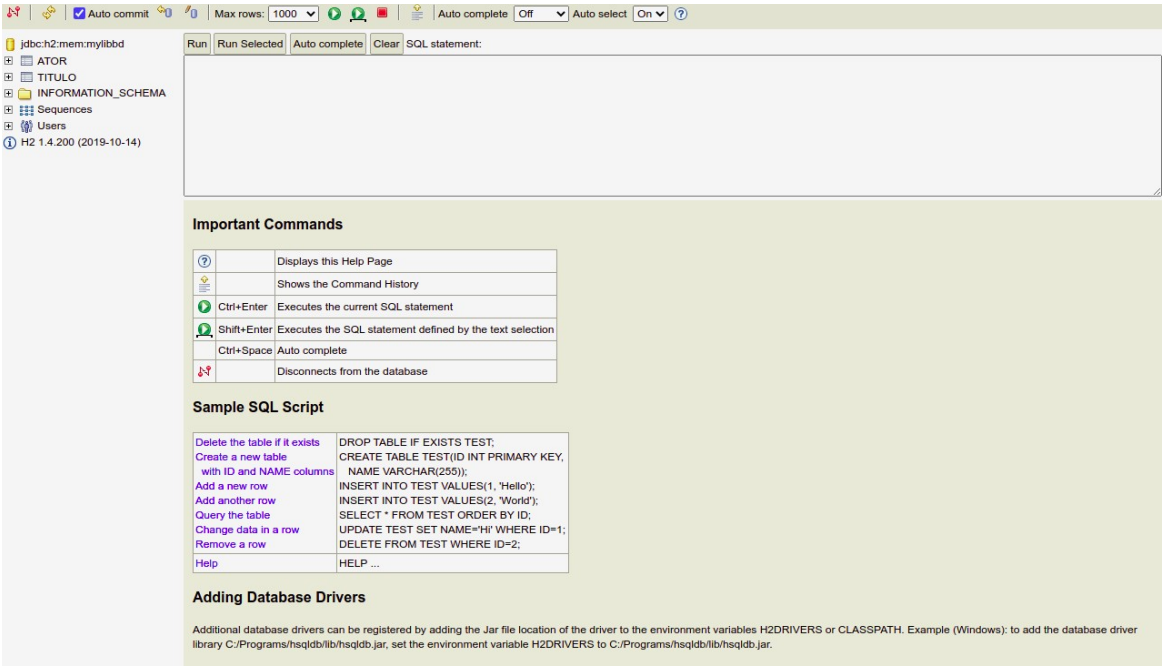

Figura 6: Tela inicial de administração do banco de dados.

No campo de texto, você pode inserir consultas SQL para criação e manipulação de dados. Assim, insira os seguintes comandos SQL e os execute para popularmos o nosso banco de dados.

```
insert into ator (id, nome, cpf) values (1, 'Ator 1', '111');
insert into ator (id, nome, cpf) values (2, 'Ator 2', '222');
insert into ator (id, nome, cpf) values (3, 'Ator 3', '333');
```

Em seguida, acesse o browser e acesse a seguinte URL http://localhost:8080/mylib/resouces/atores. Caso tudo tenha sido realizado corretamente, o resultado é aquele apresentado pela Figura 7.

Figura 7: Tela com o resultado da execução do serviço implementado.