



# Cento Universitário UNA

Sistemas de Informação

Recuperação de Informação

Práticas de Laboratório

Wesley Dias Maciel

2019/01



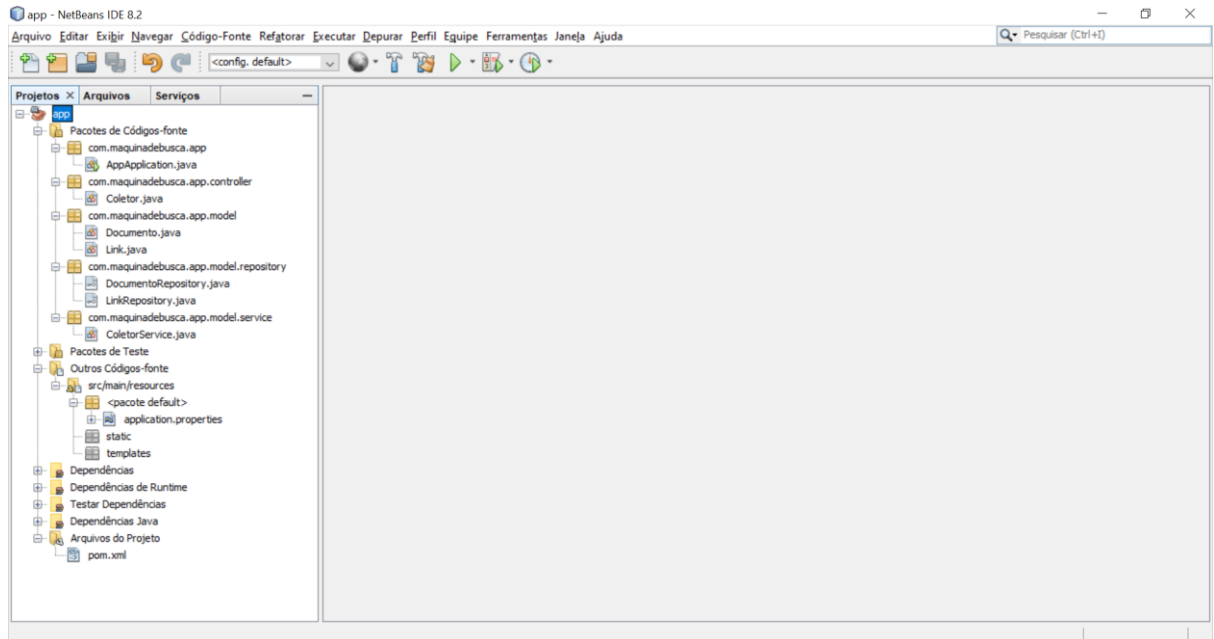
Centro Universitário UNA  
Sistemas de Informação  
Recuperação de Informação  
Prática de Laboratório  
Wesley Dias Maciel  
2019/01

# Spring Boot, Hibernate e JPA

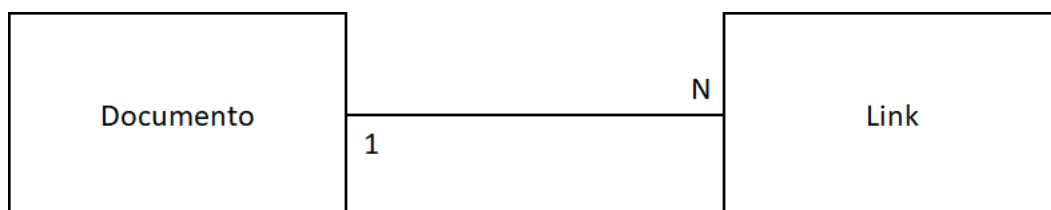


## Prática 04

1) Observe abaixo a estrutura atual do projeto.



Crie um relacionamento de “1 para N” entre as entidades **Documento** e **Link**, como apresentado abaixo.



Remova através do comando DROP todas as tabelas previamente existentes no banco de dados “**maquinadebusca**” do MySQL.

Altere a entidade **Documento** conforme apresentado abaixo.

```
package com.maquinadebusca.app.model;
```

```
import com.fasterxml.jackson.annotation.JsonIdentityInfo;
```



```
import com.fasterxml.jackson.annotation.ObjectIdGenerators;
import java.io.Serializable;
import java.util.HashSet;
import java.util.Objects;
import java.util.Set;
import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.persistence.OneToOne;
import javax.validation.constraints.NotBlank;

@Entity
@JsonIdentityInfo (
    generator = ObjectIdGenerators.PropertyGenerator.class,
    property = "id"
)
public class Documento implements Serializable {

    static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue (strategy = GenerationType.AUTO)
    private Long id;

    @NotBlank
    private String url;

    @Lob
    @NotBlank
    private String texto;

    @Lob
    @NotBlank
    private String visao;

    @OneToOne (
        mappedBy = "documento", // Nome do atributo na classe Link.
        cascade = CascadeType.ALL,
        fetch = FetchType.LAZY,
```



```
        orphanRemoval = true
    )
    private Set<Link> links;

    public Documento () {
        links = new HashSet ();
    }

    public Documento (String url, String texto, String visao) {
        this.url = url;
        this.texto = texto;
        this.visao = visao;
        this.links = new HashSet ();
    }

    public Long getId () {
        return id;
    }

    public void setId (Long id) {
        this.id = id;
    }

    public String getUrl () {
        return url;
    }

    public void setUrl (String url) {
        this.url = url;
    }

    public String getTexto () {
        return texto;
    }

    public void setTexto (String texto) {
        this.texto = texto;
    }

    public String getVisao () {
        return visao;
    }
}
```



```
public void setVisao (String visao) {
    this.visao = visao;
}

public Set<Link> getLinks () {
    return links;
}

public void setLinks (Set<Link> links) {
    this.links = links;
}

public void addLink (Link link) {
    link.setDocumento (this);
    this.links.add (link);
}

public void removeLink (Link link) {
    link.setDocumento (null);
    links.remove (link);
}

@Override
public int hashCode () {
    int hash = 5;
    hash = 59 * hash + Objects.hashCode (this.id);
    hash = 59 * hash + Objects.hashCode (this.url);
    return hash;
}

@Override
public boolean equals (Object obj) {
    if (this == obj) {
        return true;
    }
    if (obj == null) {
        return false;
    }
    if (getClass () != obj.getClass ()) {
        return false;
    }
    final Documento other = (Documento) obj;
    if (!Objects.equals (this.url, other.url)) {
```



```
        return false;
    }
    if (!Objects.equals(this.id, other.id)) {
        return false;
    }
    return true;
}
}
```

Altere a entidade **Link** conforme apresentado abaixo.

```
package com.maquinadebusca.app.model;

import com.fasterxml.jackson.annotation.JsonIdentityInfo;
import com.fasterxml.jackson.annotation.ObjectIdGenerators;
import java.io.Serializable;
import java.time.LocalDateTime;
import java.util.Objects;
import javax.persistence.Basic;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.validation.constraints.NotBlank;

@Entity
@JsonIdentityInfo (
    generator = ObjectIdGenerators.PropertyGenerator.class,
    property = "id"
)
public class Link implements Serializable {

    static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue (strategy = GenerationType.AUTO)
    private Long id;

    @NotBlank
    private String url;
```



@Basic

```
private LocalDateTime ultimaColeta;
```

```
@ManyToOne (fetch = FetchType.EAGER)
```

```
@JoinColumn (name = "documento_id")
```

```
private Documento documento;
```

```
public Link () {  
}
```

```
public Link (String url, Documento documento) {  
    this.url = url;  
    this.ultimaColeta = null;  
    this.documento = documento;  
}
```

```
public Long getId () {  
    return id;  
}
```

```
public void setId (Long id) {  
    this.id = id;  
}
```

```
public String getUrl () {  
    return url;  
}
```

```
public void setUrl (String url) {  
    this.url = url;  
}
```

```
public LocalDateTime getUltimaColeta () {  
    return ultimaColeta;  
}
```

```
public void setUltimaColeta (LocalDateTime ultimaColeta) {  
    this.ultimaColeta = ultimaColeta;  
}
```

```
public Documento getDocumento () {  
    return documento;  
}
```





```
}  
  
public void setDocumento (Documento documento) {  
    this.documento = documento;  
}  
  
@Override  
public int hashCode () {  
    int hash = 5;  
    hash = 71 * hash + Objects.hashCode (this.id);  
    hash = 71 * hash + Objects.hashCode (this.url);  
    return hash;  
}  
  
@Override  
public boolean equals (Object obj) {  
    if (this == obj) {  
        return true;  
    }  
    if (obj == null) {  
        return false;  
    }  
    if (getClass () != obj.getClass ()) {  
        return false;  
    }  
    final Link other = (Link) obj;  
    if (!Objects.equals (this.url, other.url)) {  
        return false;  
    }  
    if (!Objects.equals (this.id, other.id)) {  
        return false;  
    }  
    return true;  
}  
  
}
```

Altere a entidade **ColetorService** conforme apresentado abaixo.

```
package com.maquinadebusca.app.model.service;  
  
import com.maquinadebusca.app.model.Documento;  
import com.maquinadebusca.app.model.Link;
```



```
import java.util.LinkedList;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.maquinadebusca.app.model.repository.DocumentoRepository;
import com.maquinadebusca.app.model.repository.LinkRepository;
import java.time.LocalDateTime;

@Service
public class ColetorService {

    @Autowired
    private DocumentoRepository dr;

    @Autowired
    private LinkRepository lr;

    public List<Documento> executar () {
        List<Documento> documentos = new LinkedList ();
        List<String> sementes = new LinkedList ();

        try {
            sementes.add ("https://www.youtube.com/");
            sementes.add ("https://www.facebook.com/");
            sementes.add ("https://www.twitter.com/");

            for (String url : sementes) {
                documentos.add (this.coletar (url));
            }
        } catch (Exception e) {
            System.out.println ("\n\n Erro ao executar o serviço de coleta! \n\n");
            e.printStackTrace ();
        }
        return documentos;
    }

    public Documento coletar (String urlDocumento) {
        Documento documento = new Documento ();
```



```
try {
    Link link = new Link ();
    Document d = Jsoup.connect (urlDocumento).get ();
    Elements urls = d.select ("a[href]");

    documento.setUrl (urlDocumento);
    documento.setTexto (d.html ());
    documento.setVisao (d.text ());

    link.setUrl (urlDocumento);
    link.setUltimaColeta (LocalDateTime.now ());
    documento.addLink (link);
    int i = 0;
    for (Element url : urls) {
        i++;
        String u = url.attr ("abs:href");
        if (!(u.equals ("")) && (u != null)) {
            link = new Link ();
            link.setUrl (u);
            link.setUltimaColeta (null);
            documento.addLink (link);
        }
    }
    System.out.println ("Número de links coletados: " + i);
    System.out.println ("Tamanho da lista links: " + documento.getLinks ().size ());
    //Salvar o documento no banco de dados.
    documento = dr.save (documento);
} catch (Exception e) {
    System.out.println ("\n\n\n Erro ao coletar a página! \n\n\n");
    e.printStackTrace ();
}
return documento;
}

public List<Documento> getDocumento () {
    Iterable<Documento> documentos = dr.findAll ();
    List<Documento> resposta = new LinkedList ();
    for (Documento documento : documentos) {
        resposta.add (documento);
    }
    return resposta;
}
```



```
public Documento getDocumento (long id) {  
    Documento documento = dr.findByid (id);  
    return documento;  
}
```

```
public List<Link> getLink () {  
    Iterable<Link> links = lr.findAll ();  
    List<Link> resposta = new LinkedList ();  
    for (Link link : links) {  
        resposta.add (link);  
    }  
    return resposta;  
}
```

```
public Link getLink (long id) {  
    Link link = lr.findByid (id);  
    return link;  
}  
}
```

Altere a entidade **Coletor** conforme apresentado abaixo.

```
package com.maquinadebusca.app.controller;  
  
import org.springframework.http.MediaType;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.bind.annotation.RequestMapping;  
import com.maquinadebusca.app.model.service.ColetorService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.PathVariable;  
  
@RestController  
@RequestMapping ("/coletor") // URL: http://localhost:8080/coletor  
public class Coletor {  
  
    @Autowired  
    ColetorService cs;  
  
    // URL: http://localhost:8080/coletor/iniciar
```



```
@GetMapping (value = "/iniciar", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity iniciar () {
    return new ResponseEntity (cs.executar (), HttpStatus.OK);
}

// URL: http://localhost:8080/coletor/documento
@GetMapping (value = "/documento", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity listarDocumento () {
    return new ResponseEntity (cs.getDocumento (), HttpStatus.OK);
}

// Request for: http://localhost:8080/coletor/documento/{id}
@GetMapping (value = "/documento/{id}", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity listarDocumento (@PathVariable (value = "id") long id) {
    return new ResponseEntity (cs.getDocumento (id), HttpStatus.OK);
}

// URL: http://localhost:8080/coletor/link
@GetMapping (value = "/link", produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity listarLink () {
    return new ResponseEntity (cs.getLink (), HttpStatus.OK);
}

// Request for: http://localhost:8080/coletor/link/{id}
@GetMapping (value = "/link/{id}", produces =
MediaType.APPLICATION_JSON_UTF8_VALUE)
public ResponseEntity listarLink (@PathVariable (value = "id") long id) {
    return new ResponseEntity (cs.getLink (id), HttpStatus.OK);
}
}
```

Teste a API da aplicação através do navegador FireFox, verificando, por exemplo:

- <http://localhost:8080/coletor/iniciar>
- <http://localhost:8080/coletor/documento>
- <http://localhost:8080/coletor/documento/1>
- <http://localhost:8080/coletor/link>
- <http://localhost:8080/coletor/link/1>



Observe as alterações ocorridas no banco de dados.

- 2) Altere o projeto, para que ele colete as novas URLs identificadas em cada página. O projeto deve armazenar as páginas e as URLs no banco de dados “maquinadebusca”. Além disso, o projeto também deve armazenar no banco de dados o momento em que as URLs foram coletadas. Crie uma classe “Host” no projeto, para armazenar o host das URL’s armazenadas no banco de dados. Essa classe deve conter um contador para registrar o número de URLs de cada host que foram coletadas e armazenadas no banco de dados. Crie o(s) relacionamento(s) adequados para essa classe e as demais. Configure a classe “Host” adequadamente, para que ela seja armazenada no banco de dados usando JPA e Hibernate. Implemente o processo de identificação e armazenamento de cada host no banco de dados. Altere a API da aplicação, para que ela permita pesquisar os hosts juntamente com o respectivo número de URLs associadas a eles. A resposta da pesquisa deve listar os hosts, o número de URLs e as URLs de cada host. A API também deve permitir pesquisar cada host separadamente.