**UNIVERSIDADE FEDERAL DE GOIÁS**

ENGENHARIA DE SOFTWARE

**Prática em engenharia de software**

Fábio Nogueira de Lucena

Modelo de serialização de dados em JSON de cada uma das classes do MR

**SUMÁRIO**

[1.Classes](#h.ycm15ynn33tv)

[1.1.DvBoolean](#h.x5gb00nqh8ff)

[1.2.DvIdentifier](#h.6dve6d28eug4)

[1.3.DvParagraph](#h.r4m8rzabbh3l)

[1.4.TermMapping](#h.xjr1bax8q2om)

[1.5.DvCodedText](#h.uouaozeetp80)

[1.6.DvText](#h.sdv2muejc1ra)

[1.7.Archetyped](#h.cxc849l5ct8f)

[1.8.FeederAudit](#h.kl3kz1uuz39n)

[1.9.FeederAuditDetails](#h.sx5qzvm76p2k)

[1.10.Link](#h.mf8mjo7hxu0j)

[1.11.Locatable](#h.l6zxmjckr3y)

[1.12.Pathable](#h.7dx73gveadin)

[1.13.Contribution](#h.et8o3ir63gft)

[1.14.CodePhrase](#h.3p5oak67npa7)

[1.15.TermMapping](#h.4iaqaa3uadpi)

[2.Método toJSON()](#h.65lacfxyqtdh)

[3.Método fromJSON()](#h.5som6xegfzsm)

[4.Observações](#h.38jk69owvrkd)

# 1.Classes

## 1.1.DvBoolean

Parâmetros:

{

“value”: boolean

}

## 1.2.DvIdentifier

Parâmetros:

{

“issuer”: String (required),

“assigner”: String (required),

“id”: String (required),

“type”: String (required)

}

## 1.3.DvParagraph

Parâmetros:

{

“items”: [

chama DvText

]

}

## 1.4.TermMapping

Parâmetros:

{

“target”: {

chama CodePhrase

},

“match”: {

chama Match

},

“purpose”: {

chama DvCodedText

}

}

## 1.5.DvCodedText

Classe pai: [DvText](#h.sdv2muejc1ra)

Parâmetros: @param value (required)

@param mapping

@param formatting

@param hyperlink

@param language

@param charset

@param definingCode not null (required)

@param terminologyService not null (interface)

Fields: private CodePhrase definingCode;

Construtor: super(value, mapping, formatting, hyperlink, language, charset, terminologyService);

{

“value”: String (required),

“mappings”: [

chama TermMapping

],

“formatting”: String,

“hyperlink”: {

chama DvUri

},

“language”: {

chama CodePhrase

},

“charset”: {

chama CodePhrase

}

“definingCode”: {

chama CodePhrase

}

}

## 1.6.DvText

Parâmetros:

{

“value”: String,

“mappings”: [ {

chama TermMapping

} ],

“formatting”: String,

“hyperlink”: {

chama DvUri

},

“language”: {

chama CodePhrase

},

“encoding”: {

chama CodePhrase

}

}

## 1.7.Archetyped

{

“archetypeId”: {

chama ArchetypeID

},

“templateId”: {

chama TemplateID

},

“rmVersion”: String

}

## 1.8.FeederAudit

{

“originatingSystemAudit”: {

chama FeederAuditDetails

},

“originatingSystemItemIds”: [ {

chama DvIdentifier

} ],

“feederSystemAudit”: {

chama FeederAuditDetails

},

“feederSystemItemIds”: [ {

chama DvIdentifier

} ],

“originalContent”: {

chama DvEncapsulated

}

}

## 1.9.FeederAuditDetails

{

“systemId”: String,

“provider”: {

chama PartyIdentified

},

“location”: {

chama PartyIdentified

},

“time”: {

chama DvDateTime

},

“subject”: {

chama PartyProxy

}

“versionId”: String

}

## 1.10.Link

{

“meaning”: {

chama DvText

},

“type”: {

chama DvText

},

“target”: {

chama DvEHRURI

}

}

## 1.11.Locatable

{

“PATH\_SEPARATOR”: String,

“ROOT”: String,

“uid”: {

chama UIDBasedID

},

“originalArchetypeNodeId”: String,

“archetypeNodeId”: String,

“name”: {

chama DvText

},

“archetypeDetails”: {

chama Archetyped

},

“feederAudit”: {

chama FeederAudit

},

“links”: [ {

chama Link

} ]

}

## 1.12.Pathable

{

“parent”: {

chama Pathable

}

}

## 1.13.Contribution

{

“uid”: ,

}

## 1.14.CodePhrase

{

“terminologyId”: {

chama TerminologyID

},

“codeString”: String

}

## 1.15.TermMapping

{

“target”: {

chama CodePhrase

},

“match”: {

chama Match

},

“purpose”: {

chama DvCodedText

}

}

# 2.Método toJSON()

String toJSON(){

for(int i=0; i<totalObjetos(); i++){

if(obtemTipo(idObjeto) == 30){ // obtem tipo de um objeto

buildJson(30)

//...

}

// um caso contemplando cada uma das 150 e muitas classes

}

}

# 3.Método fromJSON()

# 4.Observações

(Thiago) - Deixei pronto tudo em /datatypes/text/ ou seja, todas as classes já tem o mapeamento pro json. Teve só um enum Match que não sabia como colocar no json, ele é usado na classe TermMapping, depois vou pensar direito