Curso de MDE

Roteiro de Execução

Eclipse Modeling Tools - Indigo Service Release 1 - 20110916-0149

Mar/2012: Testado no Ubuntu 11.10 - Oneiric - Daniel Lucrédio

Modeling components	**Version**
Eclipse Modeling Tools	1.4.1.20110909-1818
Graphical Modeling Framework SDK	2.4.0.v20110607-2359-777B-95pDdE_mTYphvVQ8LWuGUjo
Java Emitter Templates (JET) SDK	1.1.1.v201101271243-42186A5NsKEQPHRQeFaJS4A35
Xpand SDK	1.1.1.v201108020519
Xtext SDK	2.0.1.v201108020636

Exemplo 1
Objetivo: Demonstrar como usar xText para criar uma linguagem textual, e criar um interpretador simples baseado em EMF

1. Criar novo projeto xText
   1.1. Project name = “br.org.cursomde.exemplo1”
   1.2. Language name= “br.org.cursomde.Math”
   1.3. File extension = “math”
   1.4. Generator Configuration: Use Experimental 2.0 Features
2. O arquivo Math.xtext irá aparecer
3. Definir a seguinte gramática para uma linguagem matemática simples

```
grammar br.org.cursomde.Math with org.eclipse.xtext.common.Terminals

generate math "http://www.org.br/cursomde/Math"

Problema : 'Problema' name=ID
           (expressoes += Expressao)*
           ;

Expressao :
            Armazena |
            Operacao;

Armazena :
           'Armazene' termo=INT;

Operacao :
           sinal=SinalMatematico termo=INT;

enum SinalMatematico :
           SOMA = '+' |
           SUBTRACAO = '-' |
           MULTIPLICACAO = '*' |
           DIVISAO = '/';
```

4. Observar que no Xtext que sintaxe abstrata e concreta são juntas
5. Gerar o plug-in da linguagem
   5.1. Clicar com botão direito sobre o arquivo:
      src/br.org.cursomde/GenerateMath.mwe2
   5.2. Escolher Run As -> MWE2 Workflow
   5.3. Neste momento, pode ser solicitado o download do ANTLR 3 parser generator, no console. Digitar “y” e continuar.
6. Observar que foi gerado o metamodelo OO correspondente:
      src-gen/br.org.cursomde/Math.ecore
   6.1. Para visualizar melhor o metamodelo, clicar com botão direito sobre o arquivo .ecore e escolher “Initialize Ecore Diagram File...”
7. Fazer deploy do plug-in
   7.1. Clicar com botão direito no projeto e escolher “Export”
   7.2. Mandar exportar como “Deployable plug-ins and fragments” os projetos br.org.cursmomde.exemplo1 e br.org.cursomde.exemplo1.ui
      7.2.1. Ao exportar, escolher a opção arquivos .jar
      7.2.2. Escolha uma pasta qualquer para exportar os arquivos
   7.3. Copiar a pasta gerada dentro da pasta do eclipse e reiniciar o ambiente
8. Criar um novo projeto xPand (em alguns momentos, o Eclipse vai perguntar se deseja adicionar natureza xText ao projeto. Escolher sempre “não”)
   8.1. Project name = “br.org.cursomde.exemplo1.generator”
   8.2. Abrir o arquivo META-INF/MANIFEST.MF, e adicionar dependência para os seguintes plugins:

-Para isso clicar na aba "dependencies", botão add e escrever o nome dos plugins.
br.org.cursomde.exemplo1
org.eclipse.emf.mwe2.launch

8.3. Criar a pasta src/model

8.4. Criar um arquivo teste.math, para testar o editor:

```
Problema p1
Armazene 20
+ 10
* 5
/ 3
```

9. Fazer um interpretador

9.1. Ainda no projeto Xpand, Criar uma nova classe denominada Executa

```
package br.org.cursomde.generator;

import java.util.List;
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowComponent;
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowContext;
import br.org.cursomde.math.Armazena;
import br.org.cursomde.math.Expressao;
import br.org.cursomde.math.Operacao;
import br.org.cursomde.math.Problema;

public class Executa implements IWorkflowComponent {
	@Override
	public void invoke(IWorkflowContext ctx) {
		@SuppressWarnings("unchecked")
		List<Problema> model = (List<Problema>) ctx.get("model");
		for (Problema problema : model) {
			float resultado = 0;
			for (Expressao expressao : problema.getExpressoes()) {
				int valor = expressao.getTermo();
				if (expressao instanceof Armazena) {
					System.out.println("  Armazena " + valor);
					resultado = valor;
				} else {
					Operacao operacao = (Operacao) expressao;
					switch (operacao.getSinal()) {
					case SOMA:
						System.out.println("  Soma " + valor);
						resultado = resultado + valor;
						break;
					case SUBTRACAO:
						System.out.println("  Subtrai " + valor);
						resultado = resultado - valor;
						break;
					case DIVISAO:
						System.out.println("  Divide " + valor);
						resultado = resultado / valor;
						break;
					case MULTIPLICACAO:
						System.out.println("  Multiplica " + valor);
						resultado = resultado * valor;
						break;
					}
				}
			}
```

```
                System.out.println("=" + resultado);
            }
        }
        @Override
        public void postInvoke() { }
        @Override
        public void preInvoke() { }
}
```

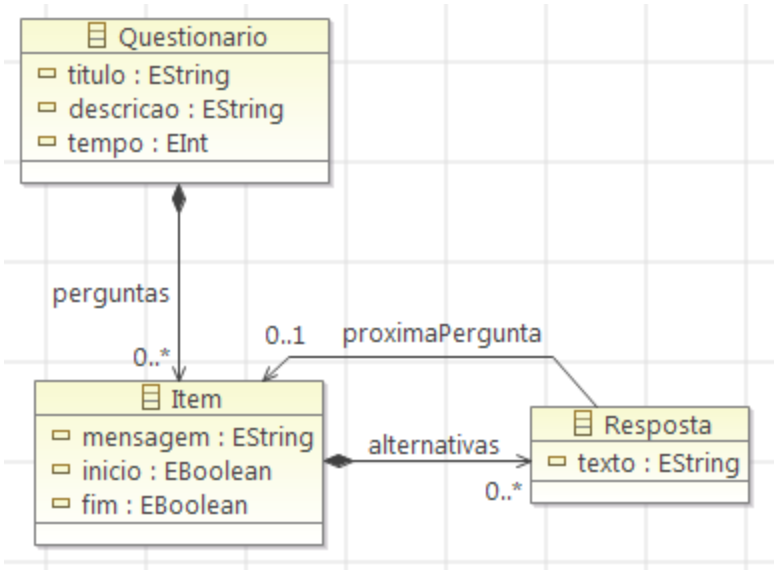
9.2. Criar um novo arquivo de workflow:
src/br.org.cursomde.generator/InterpretaMath.mwe2

```
module br.org.cursomde.generator.InterpretaMath
import org.eclipse.emf.mwe.utils.*
var modelPath = "src/model"
Workflow {
        component = org.eclipse.xtext.mwe.Reader {
            // lookup all resources on the classpath
            // useJavaClassPath = true

            // or define search scope explicitly
            path = modelPath

            // this class will be generated by the xtext generator
            register = br.org.cursomde.MathStandaloneSetup {}
            load = {
                slot = "model"
                type = "Problema"
            }
        }
        component = br.org.cursomde.generator.Executa {} //nome do pacote + Executa
}
```

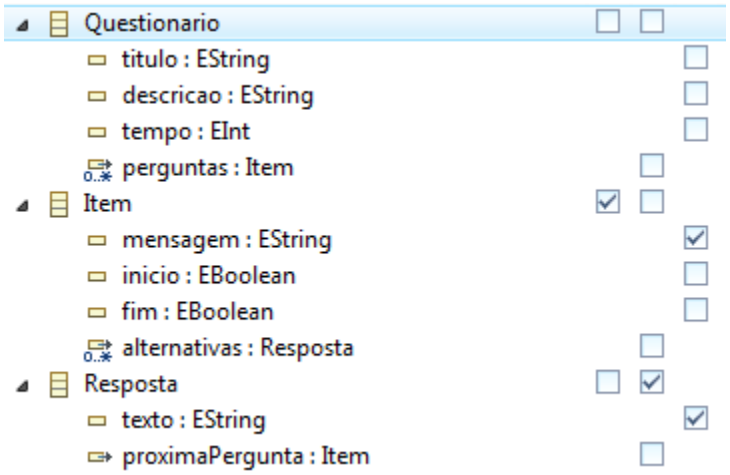
9.3. Rodar o exemplo
9.3.1. Clicar com botão direito sobre InterpretaMath.mwe2
9.3.2. Escolher “Run As” -> “MWE2 Workflow”

Exemplo 2
Objetivo: Demonstrar como usar GMF para criar uma linguagem visual, e criar um interpretador simples baseado em EMF

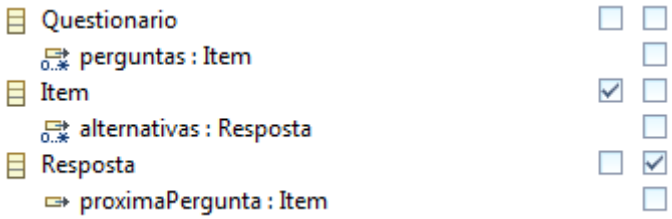
1. Criar novo projeto GMF
   1.1. name = "br.org.cursomde.exemplo2"
   1.2. Escolher "Next" e mandar mostrar a dashboard
   1.3. Mover a pasta model para dentro da pasta src
2. Criar um Domain Model (usando o dashboard), dentro da pasta model, chamado questionario.ecore
3. Inicializar diagrama ecore para editar mais facilmente
   3.1. Clicar com botão direito sobre questionario.ecore e escolher "Initialize Ecore Diagram File..." (Não utilizar "Initialize ecore_diagram diagram file")
4. Criar o metamodelo
   4.1. Se não estiver aparecendo, mandar mostrar a "properties view"
   4.2. Clicar em qualquer lugar do diagrama, e setar as propriedades:
      4.2.1. Name: questionario
      4.2.2. Ns Prefix: questionarioNsPrefix
      4.2.3. Ns URI: questionarioNsURI
   4.3. Criar o seguinte metamodelo



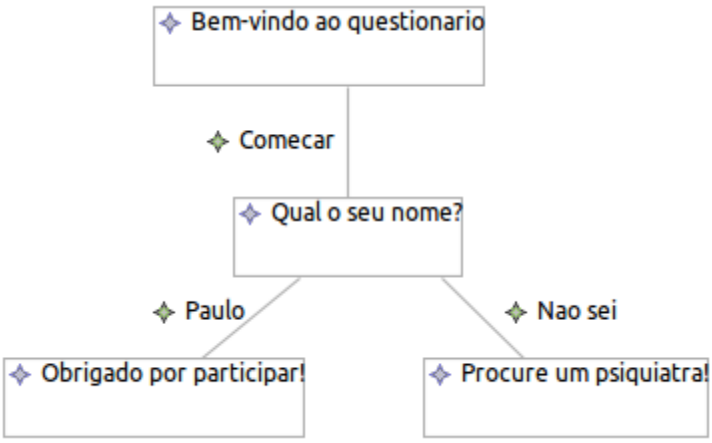
5. Derivar o Domain Gen Model (usando o dashboard) e gerar o código EMF
   5.1. Abrir o genmodel gerado
5.2. Clicar sobre o elemento raiz (Questionario) com o botão direito e escolher "Generate All"
6. Derivar o Graphical Def Model (usando o dashboard)
   6.1. Escolher "Questionario" como Diagram Element
   6.2. Na tela de definição gráfica, escolher as seguintes opções:



7. Derivar o Tooling Def Model (usando o dashboard)
   7.1. Escolher “Questionario” como Diagram Element
   7.2. Na tela de definição das ferramentas, escolher as seguintes opções:



8. Combinar o Graphical e Tooling Def Models no Mapping Model (usando o dashboard)
   8.1. Escolher “Questionario” como elemento do canvas mapping
   8.2. Seguir até o fim
   8.3. Após o gmfmap ser criado, expandir o “Link Mapping” e clicar sobre o nó filho “Feature Label Mapping false”
   8.4. Na propriedade “Diagram Label” escolher “Diagram Label RespostaTexto”
9. Transformar o Mapping Model em Diagram Editor Gen Model (usando o dashboard)
   9.1. Aguardar, pois não é exibida nenhuma janela nesse último passo
10. Mandar gerar o editor de diagramas
   10.1. Clicar sobre “Generate diagram editor” no dashboard
11. Fazer deploy
   11.1. Clicar com botão direito no projeto e escolher “Export...”
   11.2. Mandar exportar como “Deployable plug-ins and fragments” todos os projetos br.org.cursomde.exemplo2.*
      11.2.1. Usar a opção de exportar para arquivos .jar
      11.2.2. No campo Qualifier replacement, inserir “noqualifier”
   11.3. Copiar a pasta gerada dentro da pasta do eclipse e reiniciar o ambiente
12. Testar o editor criado
   12.1. Criar um novo Xpand Project
      12.2.1. nome = “br.org.cursomde.exemplo2.generator”
   12.2. Criar um novo arquivo do diagrama
      12.2.1. Clicar com botão direito no pacote src, escolher “New” -> “Example...” -> “Questionario diagram”
      12.2.2. Testar o editor, criando um questionario
         12.2.2.1. Clicar em qualquer lugar sobre o diagrama, e editar as propriedades básicas do questionário
            Descricao = “Questionário de teste”
            Tempo = 10
            Titulo = “Testando a linguagem para questionários”
         12.2.2.2. Criar o seguinte questionário



12.2.2.3. Marcar o início e o fim do questionário, utilizando a janela de propriedades
13. Embelezar o diagrama
13.1. Trocar os ícones
13.1.1. Sair do eclipse, remover os arquivos .jar do plug-in e iniciar novamente
13.1.2. Editar os ícones da pasta icons/full/obj16 do projeto br.org.cursomde.exemplo2.edit
13.1.3. Exportar os plug-ins novamente e testar
13.2. Colocar seta na resposta
13.2.1. No gmfgraph, adicionar um filho “Polyline Decoration” ao nó “Polyline Connection RespostaFigure” (Filho de “Figure Descriptor Resposta Figure”)
13.2.2. Clicar sobre o elemento recém-criado e colocar no campo “Name” o valor “Seta”
13.2.3. Clicar sobre o elemento “Polyline Connection RespostaFigure” e escolher para a propriedade “Target decoration” o valor “Polyline Decoration Seta”
13.2.4. Refazer os passos “Transform” e “Generate diagram editor” do dashboard
13.2.5. Exportar os plug-ins novamente e testar
14. Fazer um interpretador
14.1. Criar um novo arquivo de workflow (src/workflow/interpretaQuestionario.mwe2) no projeto br.org.cursomde.generator:

```
module workflow.interpretaQuestionario

var model = "br.org.cursomde.exemplo2.generator/src/default.questionario"

Workflow {
    bean = org.eclipse.emf.mwe.utils.StandaloneSetup {
        platformUri = ".."
        registerGeneratedEPackage = "questionario.QuestionarioPackage"
    }

    component = org.eclipse.emf.mwe.utils.Reader {
        uri = "platform:/resource/${model}"
        modelSlot = "model"
    }
    component = questionario.Executa {}
}
```

14.2. Configurar as dependências do projeto
14.2.1. Abrir o arquivo META-INF/MANIFEST.MF
14.2.2. Na aba de dependências, adicionar os seguintes plug-ins como required:
- br.org.cursomde.exemplo2
- org.eclipse.emf.mwe2.launch
14.3. Criar nova classe (questionario.Executa) no pacote src

```
package questionario;
import java.util.Scanner;
import org.eclipse.emf.ecore.EObject;
import org.eclipse.emf.mwe.core.WorkflowContext;
import org.eclipse.emf.mwe.core.issues.Issues;
import org.eclipse.emf.mwe.core.lib.AbstractWorkflowComponent;
import org.eclipse.emf.mwe.core.monitor.ProgressMonitor;

public class Executa extends AbstractWorkflowComponent {
      @Override
      public void checkConfiguration(Issues issues) { }

    protected void invokeInternal(WorkflowContext ctx, ProgressMonitor monitor, Issues issues) {
```

```
        EObject model = (EObject) ctx.get("model");
        Questionario questionario = (Questionario)model;
        Item itemAtual = null;
        for(Item i: questionario.getPerguntas()) {
            if(i.isInicio()) { itemAtual = i;  break;  }
        }
        if(itemAtual == null) {
            issues.addError("Não existe mensagem definida como inicial!");
            return;
        }
        System.out.println(itemAtual.getMensagem());
        Scanner console = new Scanner(System.in);
        while(!itemAtual.isFim()) {
            for(int i=1;i<=itemAtual.getAlternativas().size();i++) {
                Resposta r = itemAtual.getAlternativas().get(i-1);
                System.out.println(i+")  "+r.getTexto());
            }
            System.out.println("Digite sua resposta");
            int opcao = console.nextInt();
            itemAtual = itemAtual.getAlternativas().get(opcao-1).getProximaPergunta();
            System.out.println(itemAtual.getMensagem());
        }
        System.out.println("Fim do questionário");
    }
}
```

15. Rodar o arquivo interpretaQuestionario.mwe2

Exemplo 3

Objetivo: Demonstrar como fazer uma transformação declarativa utilizando xTend2

1. Copiar projeto br.org.cursomde.exemplo1.generator
   1.1. Novo nome = “br.org.cursomde.exemplo3”
2. Criar uma nova classe xTend br.org.cursomde.generator.MathTransformations

```
package br.org.cursomde.generator

import java.util.List
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowComponent
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowContext
import br.org.cursomde.math.*

class MathTransformations implements IWorkflowComponent {
	override void invoke(IWorkflowContext ctx) {
		val model = ctx.get("model") as List<Problema>;
		model.transforma()
	}
	override void postInvoke() { }
	override void preInvoke() { }

	def transforma(List<Problema> probs) {
		for(Problema p:probs)
			p.transformaProblema()
	}

	def transformaProblema(Problema problema) {
		for(exp: problema.expressoes) {
			switch(exp) {
				Armazena: exp.incrementaArmazena()
				Operacao: exp.trocaSomaPorSubtracao()
			}
		}
		problema.adicionaExpressao()
	}

	def incrementaArmazena(Armazena armazena) {
		armazena.setTermo(armazena.termo + 10)
	}

	def trocaSomaPorSubtracao(Operacao operacao) {
		if(operacao.sinal == SinalMatematico::SOMA) {
			operacao.setSinal(SinalMatematico::SUBTRACAO)
		}
	}

	def adicionaExpressao(Problema problema) {
		val op = MathFactory::eINSTANCE.createOperacao()
		op.setSinal(SinalMatematico::SOMA)
		op.setTermo(1234)
		problema.expressoes.add(op)
	}
}
```

   2.1. Abrir arquivo META-INF/MANIFEST.MF e adicionar a dependência para os seguintes plugins:
	org.eclipse.xtext.xbase.lib

      org.eclipse.xtext.xtend2.lib
   2.2. Clicar com botão direito no projeto, em seguida Configure → Remove Xtext/Xpand/Xtend nature (remover todos)
   2.3. Clicar novamente no mesmo menu e selecionar: Add Xtext nature
3. Adicionar o seguinte trecho ao InterpretaMath.mwe2 (antes do Executa)
      component = MathTransformations {}
4. Rodar o workflow

Exemplo 4
Objetivo: Demonstrar como fazer uma transformação procedural usando EMF

1. Copiar projeto br.org.cursomde.exemplo2.generator
   1.1. Novo nome = “br.org.cursomde.exemplo4”
2. Criar dentro do pacote src uma classe questionario.Transforma

```
package questionario;

import org.eclipse.emf.ecore.EObject;
import org.eclipse.emf.mwe.core.WorkflowContext;
import org.eclipse.emf.mwe.core.issues.Issues;
import org.eclipse.emf.mwe.core.lib.AbstractWorkflowComponent;
import org.eclipse.emf.mwe.core.monitor.ProgressMonitor;

public class Transforma extends AbstractWorkflowComponent {
	@Override
	public void checkConfiguration(Issues issues) { }

	@Override
	protected void invokeInternal(WorkflowContext ctx, ProgressMonitor monitor,
			Issues issues) {
		EObject model = (EObject) ctx.get("model");
		Questionario questionario = (Questionario) model;
		Item itemInicial = null;
		for (Item i : questionario.getPerguntas()) {
			if (i.isInicio()) {
				itemInicial = i;
				break;
			}
		}
		if (itemInicial == null) {
			issues.addError("Não existe mensagem definida como inicial!");
			return;
		}
		for (Item i : questionario.getPerguntas()) {
			if (!i.isInicio() && !i.isFim()) {
				Resposta r = QuestionarioFactory.eINSTANCE.createResposta();
				r.setTexto("Voltar ao início");
				r.setProximaPergunta(itemInicial);
				i.getAlternativas().add(r);
			}
		}
	}
}
```

3. Adicionar a chamada para a nova classe criada, no final arquivo interpretaQuestionario.mwe2, antes do componente Executa, e modificar o começo do arquivo para apontar para o projeto atual

```
var model = "br.org.cursomde.exemplo4/src/default.questionario"

Workflow {
    ...
    component = questionario.Transforma {}
    component = questionario.Executa {}
}
```

4. Rodar o workflow

Exemplo 5

Objetivo: Demonstrar como gerar código utilizando Xtend2

1. Copiar projeto br.org.cursomde.exemplo3
   1.1. Novo nome = “br.org.cursomde.exemplo5”
2. Criar classe Xtend src/br.org.cursomde.generator.GeraCodigoMath

```
package br.org.cursomde.generator

import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IGenerator
import org.eclipse.xtext.generator.IFileSystemAccess
import static extension org.eclipse.xtext.xtend2.lib.ResourceExtensions.*
import br.org.cursomde.math.*

class GeraCodigoMath implements IGenerator {
	override void doGenerate(Resource resource, IFileSystemAccess fsa) {
		for(e:resource.allContentsIterable.filter(typeof(Problema))) {
			fsa.generateFile(e.name+".java",e.compile)
		}
	}

	def compile(Problema p) '''
	public class «p.name» {
		public static void main(String args[]) {
			double resultado = 0;
			«FOR expressao:p.expressoes»
				«expressao.compile»
			«ENDFOR»
			System.out.println("O resultado é "+resultado);
		}
	}
	'''

	def dispatch compile(Armazena a) '''
		resultado = «a.termo»;
	'''

	def dispatch compile(Operacao o) '''
		resultado = resultado «o.sinal» «o.termo»;
	'''
}
```

3. Criar um workflow para chamar o gerador
3.1. Criar uma classe para fazer a amarração do gerador recém-criado, br.org.cursomde.generator.GeraCodigoRuntimeModule

```
package br.org.cursomde.generator;

import org.eclipse.xtext.generator.IGenerator;
import br.org.cursomde.MathRuntimeModule;

public class GeraCodigoRuntimeModule extends MathRuntimeModule {

	@Override
	public Class<? extends IGenerator> bindIGenerator() {
	  return GeraCodigoMath.class;
	}
```

```
}
```

3.2. Criar uma classe para fazer o registro do injetor acima, br.org.cursomde.generator.GeraCodigoStandaloneSetup

```
package br.org.cursomde.generator;

import br.org.cursomde.MathStandaloneSetup;
import com.google.inject.Guice;
import com.google.inject.Injector;

public class GeraCodigoStandaloneSetup extends MathStandaloneSetup {
	@Override
	public Injector createInjectorAndDoEMFRegistration() {
		return Guice.createInjector(new GeraCodigoRuntimeModule());
	}
}
```

3.3. Criar o arquivo de workflow GeradorMath.mwe2

```
module br.org.cursomde.generator.GeradorMath

import org.eclipse.emf.mwe.utils.*

var targetDir = "src-gen"
var modelPath = "src/model"

Workflow {

	component = org.eclipse.xtext.mwe.Reader {
		// lookup all resources on the classpath
		// useJavaClassPath = true

		// or define search scope explicitly
		path = modelPath

		// this class will be generated by the xtext generator
		register = br.org.cursomde.MathStandaloneSetup {}
		loadResource = {
			slot = "model"
		}
	}

	component = org.eclipse.xtext.generator.GeneratorComponent {
		register = br.org.cursomde.generator.GeraCodigoStandaloneSetup {}
		slot = 'model'
		outlet = {
			path = targetDir
		}
	}
}
```

4. Rodar o workflow

4.1. Observar que foi gerado um arquivo src-gen/<problema>.java para cada modelo na pasta de entrada

Exemplo 6

Objetivo: Demonstrar como gerar código utilizando JET

1. Criar um novo JET Transformation Project
   1.1. nome = “br.org.cursomde.exemplo6”
   1.2. Deixar as outras opções como default
2. Configurar o projeto para ler modelos EMF
   2.1. Abrir o arquivo plugin.xml no raiz do projeto
   2.2. Clicar na aba Extensions
   2.3. Expandir o elemento org.eclipse.jet.transform na árvore All Extensions
   2.4. Clicar no elemento (transform)
   2.5. No campo modelLoader, preencher org.eclipse.jet.emf
3. Criar templates para gerar um HTML para o questionário
   3.1. arquivo templates/main.jet

```
<%@taglib prefix="ws" id="org.eclipse.jet.workspaceTags" %>

<c:setVariable var="org.eclipse.jet.taglib.control.iterateSetsContext"
select="true()"/>

<ws:project name="mpes.cesar.edu.mc3.exemplo6.generated">
   <ws:folder path="html-gen">
      <ws:file path="questionario.html" template="templates/questionario.html.jet" />
   </ws:folder>
</ws:project>
```

   3.2. arquivo templates/questionario.html.jet

```
<html>
<head>
<title><c:get select="/Questionario/@titulo" /></title>
</head>
<body>
<h1><c:get select="/Questionario/@titulo" /></h1>
<br>
Descrição: <c:get select="/Questionario/@descricao" />
<br>
<br>
O tempo estimado para este questionário é de <c:get select="/Questionario/@tempo" />
minutos
<hr>
<br>
<% int contadorPergunta = 1; %>
<c:iterate select="/Questionario/perguntas[@fim!='true']" var="pergunta">
<c:get select="$pergunta/@mensagem" /> <br>
<c:iterate select="$pergunta/alternativas" var="resposta" delimiter=" ">
<input type="radio" name="rbPergunta<%=contadorPergunta%>" value=""><c:get
select="$resposta/@texto" />
</c:iterate>
<% contadorPergunta++; %>
<br><br>
</c:iterate>
Possíveis conclusões:<br>
<c:iterate select="/Questionario/perguntas[@fim='true']" var="pergunta">
<input type="radio" name="rbConclusao" value=""><c:get select="$pergunta/@mensagem" />
<br>
</c:iterate>
</body>
</html>
```

4. Rodar a transformação
   4.1. Clicar no menu “Run” -> “Run Configurations...”
   4.2. Clicar com botão direito sobre “JET Transformation” e escolher “New”
   4.3. No campo Transformation Input, escolher
        - br.org.cursomde.exemplo4/src/default.questionario
   4.4. No campo Transformation ID, escolher
        - br.org.cursomde.exemplo6
   4.5. Clicar em “Apply” e em seguida em “Run”
   4.6. Observar que foi criado um novo projeto mpes.cesar.edu.mc3.exemplo6.generated, com o conteúdo do questionário sendo gerado no arquivo html-gen/questionario.html
5. Mostrar como combinar com a transformação
   5.1. Copiar projeto br.org.cursomde.exemplo4
      5.1.1. Novo nome = “br.org.cursomde.exemplo6.transforma”
   5.2. Modificar o arquivo de workflow
      5.2.1. Modificar a propriedade “model” para apontar para o modelo deste projeto
      5.2.2. Incluir uma propriedade “transformedModel” para apontar para o modelo transformado
      5.2.3. Comentar o componente “questionario.Executa”
      5.2.4. Incluir um componente de escrita do modelo transformado

```
module workflow.interpretaQuestionario

var model = "br.org.cursomde.exemplo6.transforma/src/default.questionario"
var transformedModel = "br.org.cursomde.exemplo6.transforma/src/
transformed.questionario"

Workflow {
    bean = org.eclipse.emf.mwe.utils.StandaloneSetup {
        platformUri = ".."
        registerGeneratedEPackage = "questionario.QuestionarioPackage"
    }

    component = org.eclipse.emf.mwe.utils.Reader {
        uri = "platform:/resource/${model}"
        modelSlot = "model"
    }
    component = questionario.Transforma {}
    //    component = questionario.Executa {}
    component = org.eclipse.emf.mwe.utils.Writer {
        modelSlot = "model"
        uri = "platform:/resource/${transformedModel}"
    }
}
```

   5.3. Rodar o workflow
      5.3.1. Será criado o arquivo transformed.questionario
      5.3.2. Clicar com botão direito sobre o arquivo criado e escolher “Initialize questionario_diagram diagram file” e verificar que o novo modelo inclui as transformações
   5.4. Rodar o gerador JET novamente, mas com o modelo transformado
      5.4.1. Clicar em “Run” -> “Run Configurations...”
      5.4.2. Clicar na configuração JET Transformation criada anteriormente
5.4.3. Modificar o campo “Transformation Input” para: br.org.cursomde.exemplo6.transforma/src/transformed.questionario
      5.4.4. Clicar em “Apply” e “Run”
      5.4.5. Verificar que o novo arquivo HTML gerado agora inclui as transformações

Estudo de caso

Objetivo: Demonstrar a aplicação dos conceitos do desenvolvimento orientado a modelos na Engenharia Web

1. Modelagem estrutural
   1.1. Criar novo projeto Xtext
      1.1.1. Name = “br.org.cursomde.estudodecaso.estrutura”
      1.1.2. Language name = “br.org.cursomde.estudodecaso.Estrutura”
      1.1.3. Extensions = “estrutura”
      1.1.4. Generator Configuration: Use Experimental 2.0 Features
   1.2. Criar a seguinte gramática

```
grammar br.org.cursomde.estudodecaso.Estrutura with org.eclipse.xtext.common.Terminals

generate estrutura "http://www.org.br/cursomde/estudodecaso/Estrutura"

Model :
    'Model' name=ID '{'
    (imports+=Import)*
    (elements+=Type)*
    '}'
    ;

Import :
    'import' importURI=STRING;

Type:
    SimpleType | Entity;

SimpleType:
    'type' name=ID;

Entity :
    'entity' name=ID ('extends' extends=[Entity])? '{'
        properties+=Property*
    '}';

Property:
    'property' name=ID ':' type=[Type] (many?='[]')?;
```

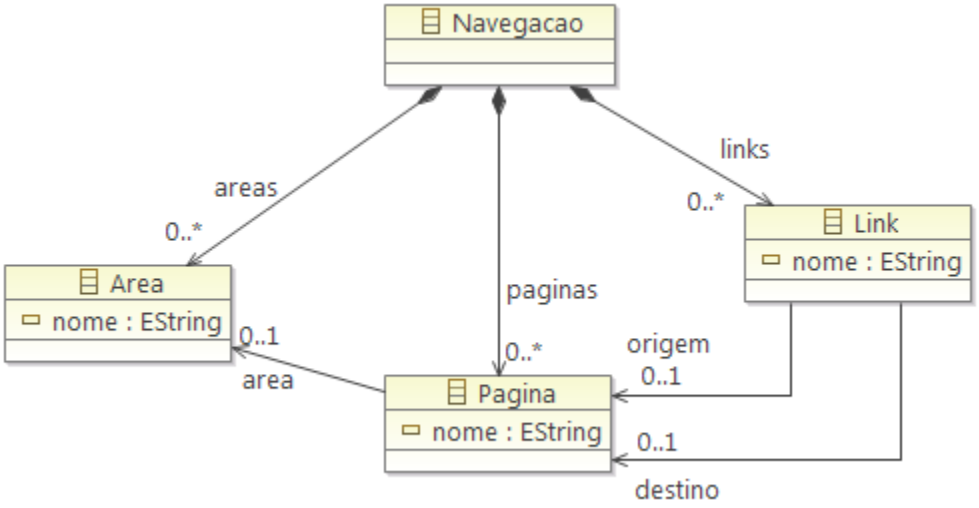
   1.3. Mandar gerar o plug-in para a linguagem
      1.3.1. Rodar o workflow GenerateEstrutura.mwe2
   1.4. Exportar como Deployable plug-ins and fragments os projetos:
        - br.org.cursomde.estudodecaso.estrutura
        - br.org.cursomde.estudodecaso.estrutura.ui
   1.5. Sair do eclipse, copiar os arquivos exportados para a pasta do eclipse e reiniciar o ambiente
   1.6. Para testar, criar um arquivo Example.estrutura em qualquer pasta

```
Model MeuSite {
      type String
      type int

      entity Pessoa {
            property nome : String
            property idade : int
            property dependentes : Pessoa[]
            property enderecoResidencial : Endereco
      }
```

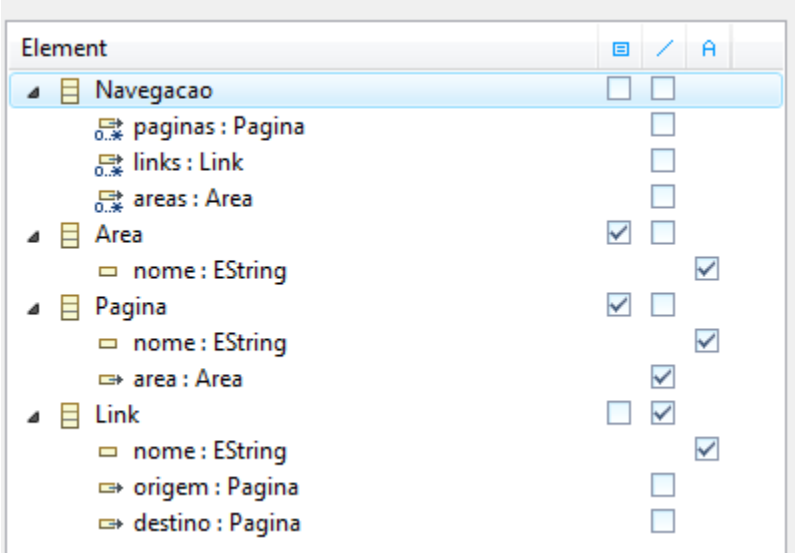
```
	entity Endereco {
		property rua : String
		property numero : int
	}
}
```

---

2. Modelagem da navegação
   2.1. Criar novo projeto GMF
      2.1.1. Nome = “br.org.cursomde.estudodecaso.navegacao”
      2.1.2. Mandar mostrar o dashboard
      2.1.3. Mover a pasta model para dentro da pasta src
   2.2. Criar um Domain Model (usando o dashboard), dentro da pasta model, chamado navegacao.ecore
   2.3. Inicializar diagrama ecore para editar mais facilmente
      2.3.1. Clicar com botão direito sobre navegacao.ecore e escolher “Initialize Ecore Diagram File...” (Não utilizar “Initialize ecore_diagram diagram file”)
   2.4. Criar o metamodelo
      2.4.1. Se não estiver aparecendo, mandar mostrar a “properties view”
      2.4.2. Clicar em qualquer lugar do diagrama, e setar as propriedades:
         2.4.2.1. Name: navegacao
         2.4.2.2. Ns Prefix: navegacaoNsPrefix
         2.4.2.3. Ns URI: navegacaoNsURI
      2.4.3. Criar o seguinte metamodelo:



2.5. Mandar derivar o Domain Gen Model (usando o dashboard)
      2.5.1. Deixar todas as opções default
      2.5.2. Após a criação, o arquivo gerado será aberto
      2.5.3. Clicar com botão direito sobre o nó raiz “Navegacao” e escolher “Generate All”
   2.6. Mandar derivar o Graphical Def Model (usando o dashboard)
      2.6.1. Escolher “Navegacao” como “Diagram Element”
      2.6.2. Na tela de definição gráfica, escolher a seguinte configuração



2.6.3. Ajustar o arquivo gmfgraph gerado:

2.6.3.1. Apagar o nó “Rectangle AreaFigure”

2.6.3.2. Clicar com botão direito sobre Figure Descriptor AreaFigure e escolher -> “New Child” -> “Rounded Rectangle”

2.6.3.3. Clicar sobre “Rounded Rectangle” para visualizar suas propriedades e digitar “AreaFigure” no campo “Name”

2.6.3.4. Mudar o campo “Line Width” para 2

2.6.3.5. Clicar com botão direito sobre o nó “Rounded Rectangle AreaFigure” e escolher -> “New Child” -> “Flow Layout”

2.6.3.6. Clicar com botão direito sobre o nó “Rounded Rectangle AreaFigure” e escolher -> “New Child” -> “Label”

2.6.3.7. Clicar sobre o recém-criado elemento “Label” e setar os seguintes campos:

Name = AreaNomeFigure

Text = <...>

2.6.3.8. Clicar sobre o elemento “Child Access getFigureNull” e setar o campo “Figure” para “Label AreaNomeFigure” (O campo Accessor irá atualizar automaticamente)

2.6.3.9. Clicar sobre o elemento “Polyline Connection PaginaAreaFigure” e mudar o campo Line Kind para LINE_DOT

2.6.3.10. Clicar com botão direito sobre o elemento “Polyline Connection LinkFigure” e escolher -> “New Child” -> “Polyline Decoration”

2.6.3.11. Clicar sobre o elemento recém-criado “Polyline Decoration” e colocar no campo Name o valor “SetaLink”

2.6.3.12. Clicar sobre o elemento “Polyline Connection LinkFigure” e colocar no campo Target Decoration o valor “Polyline Decoration SetaLink”

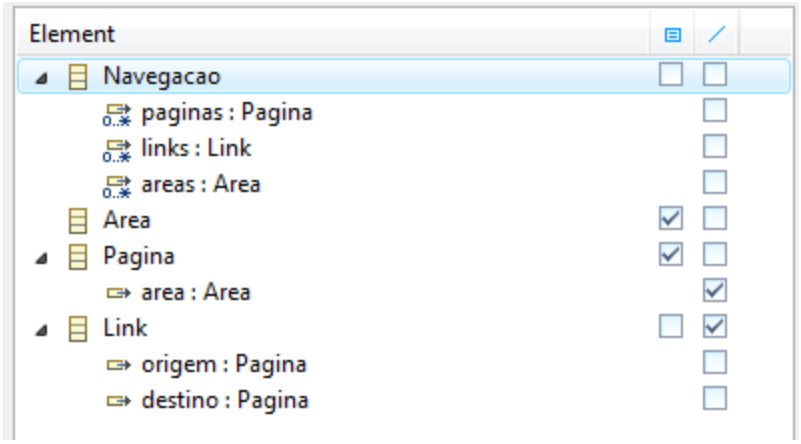
2.6.3.13. Clicar com botão direito sobre o elemento “Polyline Connection LinkFigure” e escolher -> “New Child” -> “Foreground Color Constant Color”

2.6.3.14. Clicar sobre o elemento recém-criado e escolher o valor “blue”

2.7. Mandar derivar o Tooling Def Model (usando o dashboard)

2.7.1. Escolher “Navegacao” como “Diagram Element”

2.7.2. Na tela de definição de ferramentas, escolher a seguinte configuração

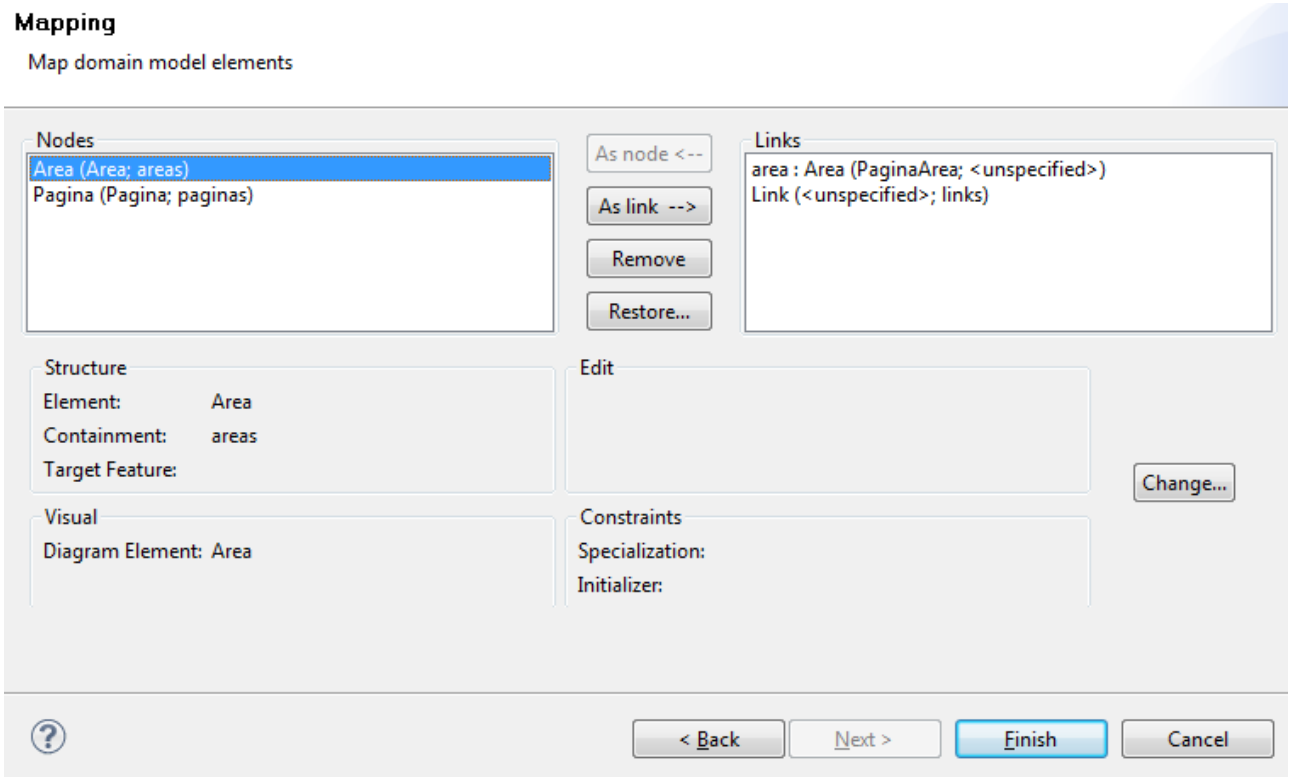


2.8. Mandar combinar Graphical e Tooling Def Models no Mapping Model (usando o dashboard)

2.8.1. Escolher “Navegação” como a classe do “Canvas Mapping”

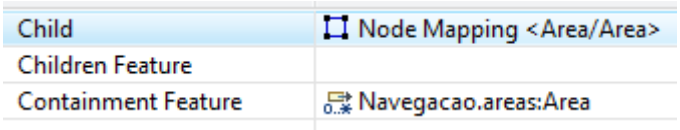
2.8.2. Deixar nas opções padrão a seleção de “Palette” e “Canvas”

2.8.3. Na tela de mapeamento, escolher a seguinte configuração e clicar em “Finish”



2.8.4. Clicando sobre os elementos do mapping, verificar/modificar as propriedades para que estejam conforme a seguir:

- Top Node Reference <areas:Area/Area>



- Node Mapping <Area/Area>

Domain meta information	
Element	Area
Misc	
Related Diagrams	
Visual representation	
Appearance Style	
Context Menu	
Diagram Node	Node Area (AreaFigure)
Tool	Creation Tool Area

- Feature Label Mapping false (Area/Area)

Domain meta information	
Features to display	Area.nome:EString
Features to edit	
Misc	
Diagram Label	Diagram Label AreaNome
Read Only	false
Visual representation	
Edit Method	MESSAGE_FORMAT
Editor Pattern	
Edit Pattern	
View Method	MESSAGE_FORMAT
View Pattern	

- Top Node Reference <paginas:Pagina/Pagina>

Child	Node Mapping <Pagina/Pagina>
Children Feature	
Containment Feature	Navegacao.paginas:Pagina

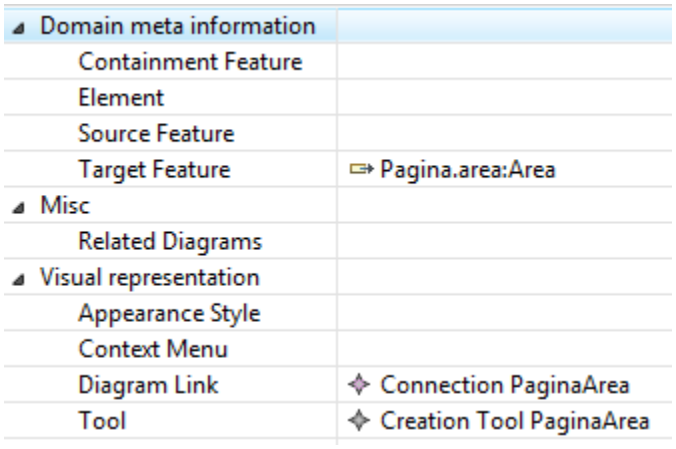
- Node Mapping <Pagina/Pagina>

Domain meta information	
Element	Pagina
Misc	
Related Diagrams	
Visual representation	
Appearance Style	
Context Menu	
Diagram Node	Node Pagina (PaginaFigure)
Tool	Creation Tool Pagina

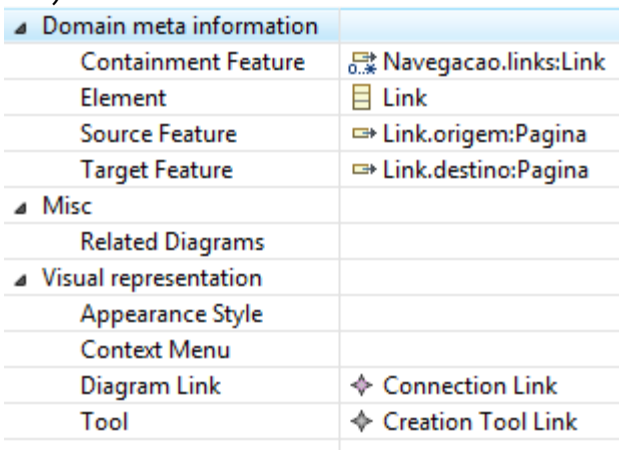
- Feature Label Mapping false (Pagina/Pagina)

Domain meta information	
Features to display	Pagina.nome:EString
Features to edit	
Misc	
Diagram Label	Diagram Label PaginaNome
Read Only	false
Visual representation	
Edit Method	MESSAGE_FORMAT
Editor Pattern	
Edit Pattern	
View Method	MESSAGE_FORMAT
View Pattern	

- Link Mapping <{Pagina.area:Area}/PaginaArea>



- Link Mapping <Link{}/> (após alterar os campos, clique com botão direito e escolha “Refresh”. O nome deste nó irá mudar para: “Link{Link.origem:Pagina->Link.destino:Pagina}/Link”)



2.8.5. Clicar com botão direito sobre este último nó e escolher “New Child” -> “Feature Label Mapping”

2.8.6. Clicar sobre o elemento recém-criado e setar os seguintes campos:

Features to display = Link.nome:Estring

Diagram Label = Diagram Label LinkNome

2.9. Transformar o Mapping Model no Diagram Editor Gen Model (usando o dashboard)

2.9.1. Aguardar, pois não é exibida nenhuma janela nesta ação

2.10. Gerar o Diagram Editor (usando o dashboard)

2.11. Exportar os projetos como plug-in

2.11.1. Clicar com botão direito sobre o projeto e escolher “Export...”

2.11.2. Exportar como Deployable plug-ins and fragments os projetos “br.org.cursomde.estudodecaso.navegacao.*”

2.11.2.1. Usar a opção de exportar para arquivos .jar

2.11.2.2. No campo Qualifier replacement, inserir “noqualifier”

2.11.3. Copiar a pasta gerada dentro da pasta do eclipse e reiniciar o ambiente

2.12. Para testar, clicar em qualquer pasta e escolher “New” -> ”Example” -> ”Navegacao Diagram” (criar qualquer conteúdo, depois iremos criar um conteúdo de exemplo)

3. Modelagem da apresentação

3.1. Criar novo projeto Xtext

3.1.1. Name = “br.org.cursomde.estudodecaso.apresentacao”

3.1.2. Language name = “br.org.cursomde.estudodecaso.Apresentacao”

3.1.3. DSL-file extension = “apresentacao”

3.1.4. Generator Configuration: Use Experimental 2.0 Features

3.2. Criar a seguinte gramática

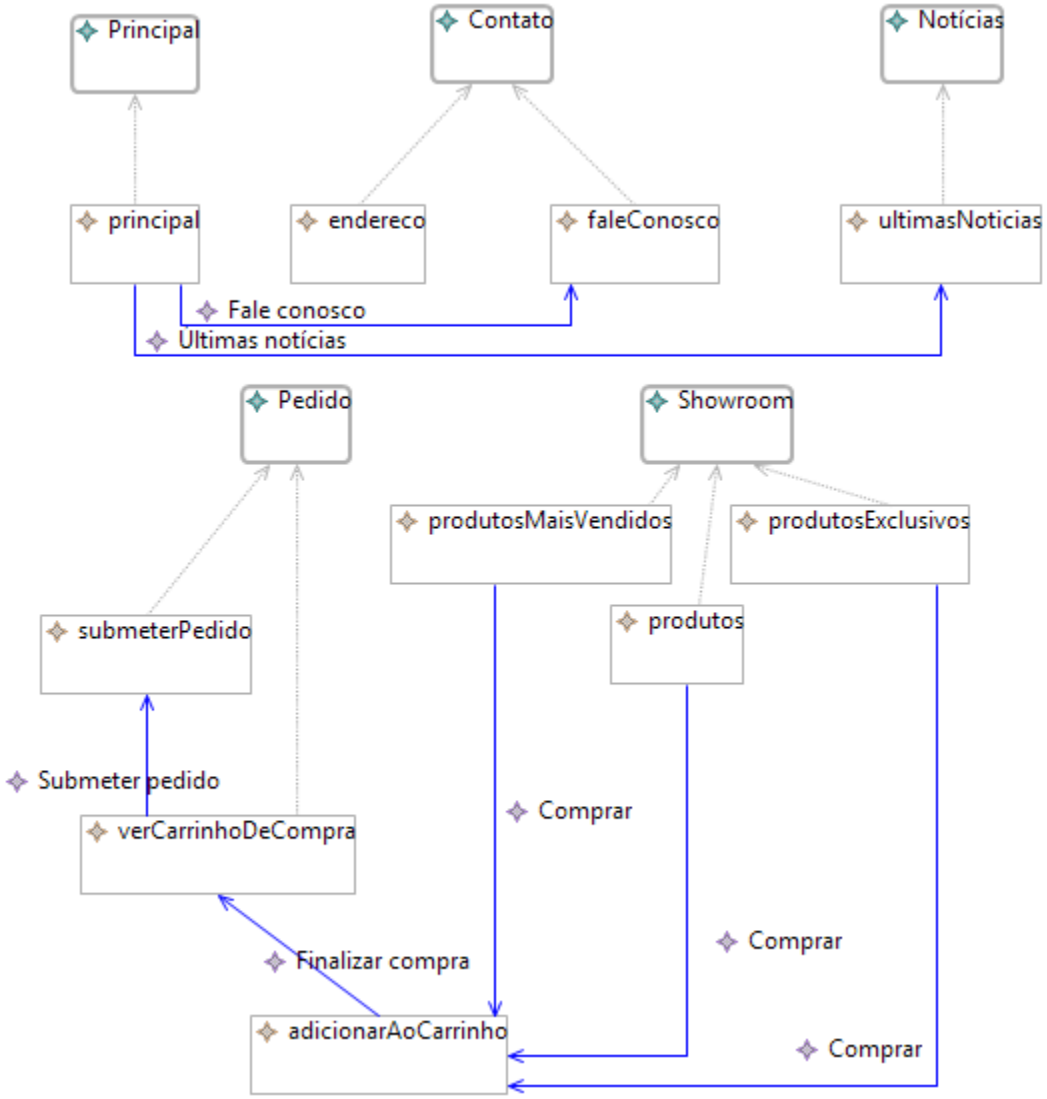
```
grammar br.org.cursomde.estudodecaso.Apresentacao with
org.eclipse.xtext.common.Terminals
```

```
generate apresentacao "http://www.org.br/cursomde/estudodecaso/Apresentacao"

Apresentacao : 'apresentacao' name=ID '{'
    (conteudo+=Conteudo)*
    (paginas+=Pagina)*
    '}'
    ;
Conteudo :
    'conteudo' name=ID '{'
    ('paragrafo' texto+=STRING)*
    '}'
    ;
Pagina :
    'pagina' name=ID '{'
    'titulo' titulo=STRING
    'ref_conteudo' (conteudoPagina += [Conteudo])+
    '}'
    ;
```

```
   3.3. Rodar o workflow “GenerateApresentacao.mwe2”
   3.4. Exportar como Deployable plug-ins and fragments os projetos
          - br.org.cursomde.estudodecaso.apresentacao
          - br.org.cursomde.estudodecaso.apresentacao.ui
   3.5. Copiar a pasta gerada dentro do eclipse e reiniciar o ambiente
   3.6. Para testar, crie um arquivo Example.apresentacao em qualquer pasta (não
precisa criar um modelo completo, no próximo item iremos criar um exemplo completo)
4. Geração de código
   4.1. Criar novo projeto Xpand
      4.1.1. Name = “br.org.cursomde.estudodecaso.gerador”
   4.2. Criar nova pasta “model” no pacote “src”
   4.3. Criar arquivo model/meusite.estrutura
```

```
Model MeuSite {
      type String
      type boolean
      type int
      type double

      entity Pessoa {
            property nome : String
            property email : String
            property receberInformacoes : boolean
      }
      entity Noticia {
            property titulo : String
            property texto : String
      }
      entity Produto {
            property nome : String
            property descricao : String
            property preco : double
      }
      entity Pedido {
            property pessoa : Pessoa
            property produtos : Produto []
      }
}
```

4.4. Criar arquivo meusite.navegacao (“New”->”Example...”->”Navegacao diagram”)



4.5. Criar arquivo meusite.apresentacao

4.5.1. Criar conteúdo e páginas de acordo com o diagrama acima

4.5.2. Manter os nomes das páginas

4.5.3. Exemplo:

```
apresentacao MeuSite {

	conteudo bemVindo {
	    paragrafo "Bem-vindo ao site de compras"
	    paragrafo "Este site foi completamente gerado a partir de modelos"
	}

	conteudo noticias {
	    paragrafo "Rio de Janeiro é a sede dos Jogos Olímpicos de 2016 - A cidade
brasileira conquistou o direito de sediar as competições das Olimpíadas de 2016.
Dirigentes, políticos e atletas se emocionaram ao comemorar a conquista."
	    paragrafo "Inquérito pelo acidente da TAM termina sem apontar culpados - A
PF entregou o inquérito pelo acidente da TAM, ocorrido em São Paulo em 2007. O
relatório não indiciou ninguém pela tragédia, o que irritou representantes das
famílias das vítimas."
	    paragrafo "Deputados brasileiros se reúnem na embaixada em Honduras - O
presidente interino, Roberto Micheletti, suspendeu o ultimato ao governo brasileiro
para definir o status de presidente deposto, Manuel Zelaya. Os dois lados deram
sinais que começam a ceder."
	    paragrafo "Governo brasileiro vai liberar recursos para vítimas de
```

```
catástrofes na Ásia e Oceania - O governo brasileiro ofereceu ajuda às vítimas dos terremotos, tsunamis e tufões na Ásia e na Oceania. O cenário é de desolação nas ilhas arrasadas pelos tsunamis."
          paragrafo "Rio 2016: bateria do Salgueiro é destaque da festa em Copa - Um palco de 70 metros foi montado, no posto 2 da praia, para os shows de comemoração. O anúncio da cidade que vai sediar as olimpíadas 2016 deve acontecer no início da tarde."
          paragrafo "Polícia Federal apura fraude que provocou cancelamento do Enem - No Rio de Janeiro, estudantes protestaram na porta do prédio do Ministério da Educação. O novo exame ainda não tem data marcada, mas deve acontecer em novembro."
          paragrafo "Adiado julgamento de Marcos Camacho em São Paulo - Marcos Camacho, acusado da morte do juiz Antônio José Dias, não foi ao fórum e o advogado dele alegou que não teve tempo de analisar a defesa."
          paragrafo "Relatório da autópsia revela que Michael Jackson estava saudável - O resultado está no relatório obtido pela Agência Associated Press. A autópsia mostrou que o corpo do popstar tinha marcas de perfurações e cicatrizes, mas a saúde estava normal."
      }

      conteudo faleconosco {
          paragrafo "Estamos abertos a críticas e sugestões"
          paragrafo "Entre em contato pelo e-mail: faleconosco@site.com"
          paragrafo "Estaremos respondendo em até 24 horas"
      }

      conteudo endereco {
          paragrafo "O site de compras tem sede em São Carlos - SP"
          paragrafo "Rua central, número 1234, Centro"
          paragrafo "CEP: 13560-100"
          paragrafo "Telefone: (16) 7263-2382 / Fax: (16) 7263-2382"
      }

      conteudo produtos {
          paragrafo "Computador Celeron Positivo"
          paragrafo "Multifuncional HP"
          paragrafo "Telefone celular Sony Ericsson"
          paragrafo "Geladeira Frost-Free Brastemp"
          paragrafo "Livro Dan Brown - O Código Da Vinci"
          paragrafo "Esteira ergométrica"
      }

      conteudo adicionarAoCarrinho {
          paragrafo "O produto foi adicionado ao carrinho"
      }

      conteudo verCarrinhoDeCompra {
          paragrafo "Este é o seu carrinho de compra"
          paragrafo "Você tem atualmente 3 produtos no seu carrinho"
          paragrafo "Telefone celular Sony Ericsson"
          paragrafo "Geladeira Frost-Free Brastemp"
          paragrafo "Livro Dan Brown - O Código Da Vinci"
      }

      conteudo submeterPedido {
          paragrafo "Use este formulário para submeter seu pedido"
          paragrafo "Verifique os dados antes de submeter"
          paragrafo "Total das compras: R$ 100,00"
      }
```

```
    pagina principal {
        titulo "Página principal"
        ref_conteudo bemVindo endereco
    }

    pagina ultimasNoticias {
        titulo "Últimas notícias"
        ref_conteudo noticias
    }

    pagina faleConosco {
        titulo "Entre em contato"
        ref_conteudo faleconosco endereco
    }

    pagina endereco {
        titulo "Endereço"
        ref_conteudo endereco
    }

    pagina produtos {
        titulo "Lista de produtos"
        ref_conteudo produtos
    }

    pagina produtosMaisVendidos {
        titulo "Produtos mais vendidos"
        ref_conteudo produtos
    }

    pagina produtosExclusivos {
        titulo "Produtos exclusivos"
        ref_conteudo produtos
    }

    pagina adicionarAoCarrinho {
        titulo "Produto adicionado"
        ref_conteudo adicionarAoCarrinho
    }

    pagina verCarrinhoDeCompra {
        titulo "Seu Carrinho"
        ref_conteudo verCarrinhoDeCompra
    }

    pagina submeterPedido {
        titulo "Submeter Pedido"
        ref_conteudo submeterPedido
    }
}
```

4.6. Adicionar as dependências do projeto

4.6.1. Abrir arquivo META-INF/MANIFEST.MF, e na aba Dependencies, adicionar os seguintes plug-ins como Required:

- org.eclipse.emf.mwe2.launch
- br.org.cursomde.estudodecaso.apresentacao
- br.org.cursomde.estudodecaso.estrutura
- br.org.cursomde.estudodecaso.navegacao
- org.eclipse.xtext.xbase.lib

- org.eclipse.xtext.xtend2.lib

4.6.2. Clicar com botão direito no projeto, em seguida Configure → Remove Xtext/Xpand/Xtend nature (remover todos)

4.6.3. Clicar novamente no mesmo menu e selecionar: Add Xtext nature

4.7. Criar gerador da estrutura (beans)

4.7.1. Criar um componente para configurar o gerador. Classe br.org.cursomde.estudodecaso.gerador.MyGeneratorComponent

```
package br.org.cursomde.estudodecaso.gerador;

import static com.google.common.collect.Lists.newArrayList;
import static com.google.common.collect.Maps.newHashMap;

import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowComponent;
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowContext;
import org.eclipse.xtext.generator.IFileSystemAccess;
import org.eclipse.xtext.generator.JavaIoFileSystemAccess;

public class MyGeneratorComponent implements IWorkflowComponent {
	private List<String> slotNames = newArrayList();
	private Map<String,String> outlets = newHashMap();
	public void addSlot(String slot) {
		this.slotNames.add(slot);
	}
	public void preInvoke() {
		if (slotNames.isEmpty())
			throw new IllegalStateException("no 'slot' has been configured.");
		if (outlets.isEmpty())
			throw new IllegalStateException("no 'outlet' has been configured.");
		for (Entry<String, String> outlet : outlets.entrySet()) {
			if (outlet.getKey()==null)
				throw new IllegalStateException("One of the outlets was
configured without a name");
			if (outlet.getValue()==null)
				throw new IllegalStateException("The path of
outlet '"+outlet.getKey()+"' was null.");
		}
	}

	public static class Outlet {
		private String outletName = IFileSystemAccess.DEFAULT_OUTPUT;
		private String path;
		public void setOutletName(String outputName) {
			this.outletName = outputName;
		}
		public void setPath(String path) {
			this.path = path;
		}
		public String getOutletName() {
			return outletName;
		}
		public String getPath() {
			return path;
		}
	}
```

```
	public void addOutlet(Outlet out) {
		outlets.put(out.outletName,out.path);
	}

	public void invoke(IWorkflowContext ctx) {
		EstudoDeCasoGerador gerador =
Guice.createInjector().getInstance(EstudoDeCasoGerador.class);
		IFileSystemAccess fileSystemAccess = getConfiguredFileSystemAccess();
		gerador.gerarCodigo(ctx, fileSystemAccess);
	}

	protected IFileSystemAccess getConfiguredFileSystemAccess() {
		final JavaIoFileSystemAccess configuredFileSystemAccess =
Guice.createInjector().getInstance(JavaIoFileSystemAccess.class);
		for (Entry<String, String> outs : outlets.entrySet()) {
			configuredFileSystemAccess.setOutputPath(outs.getKey(),
outs.getValue());
		}
		return configuredFileSystemAccess;
	}
	public void postInvoke() { }
}
```

4.7.2. Criar componente de workflow. Arquivo Gera.mwe2, dentro da pasta br.org.cursomde.estudodecaso.gerador

```
module br.org.cursomde.estudodecaso.gerador.Gera

import org.eclipse.emf.mwe.utils.*

var codeTargetDir = "src-gen"
var webTargetDir = "html-gen"
var modelPath = "src/model"
var modeloNavegacao = "br.org.cursomde.estudodecaso.gerador/src/model/
meusite.navegacao"

Workflow {
	bean = org.eclipse.emf.mwe.utils.StandaloneSetup {
		platformUri = ".."
		registerGeneratedEPackage = "navegacao.NavegacaoPackage"
	}
	component = org.eclipse.xtext.mwe.Reader {
		path = modelPath
		register = br.org.cursomde.estudodecaso.EstruturaStandaloneSetup {}
		register = br.org.cursomde.estudodecaso.ApresentacaoStandaloneSetup {}
		load = {
			slot = "modeloEstruturaLido"
			type = "Model"
		}
		load = {
			slot = "modeloApresentacaoLido"
			type = "Apresentacao"
		}
	}
	component = org.eclipse.emf.mwe.utils.Reader {
		uri = "platform:/resource/${modeloNavegacao}"
		modelSlot = "modeloNavegacaoLido"
	}
```

```
	component = br.org.cursomde.estudodecaso.gerador.MyGeneratorComponent {
		slot = "modeloEstruturaLido"
		slot = "modeloApresentacaoLido"
		slot = "modeloNavegacaoLido"
		outlet = {
			outletName = "codigo"
			path = codeTargetDir
		}
		outlet = {
			outletName = "web"
			path = webTargetDir
		}
	}
}
```

4.7.3. Criar classe Xtend br.org.cursomde.estudodecaso.gerador.EstudoDeCasoExtensions

```
package br.org.cursomde.estudodecaso.gerador
import br.org.cursomde.estudodecaso.estrutura.*
class EstudoDeCasoExtensions {
	def model(Type t) {
		t.eContainer
	}
	def entities(Model m) {
		m.elements.filter(typeof(Entity))
	}
	def getter(Property p) {
		"get" + p.name.toFirstUpper()
	}
	def setter(Property p) {
		"set" + p.name.toFirstUpper()
	}
}
```

4.7.4. Criar classe Xtend br.org.cursomde.estudodecaso.gerador.EstudoDeCasoGerador

```
package br.org.cursomde.estudodecaso.gerador

import br.org.cursomde.estudodecaso.estrutura.*
import br.org.cursomde.estudodecaso.apresentacao.*
import navegacao.*
import org.eclipse.xtext.generator.IGenerator
import java.util.List
import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IFileSystemAccess
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowContext
import com.google.inject.Inject

class EstudoDeCasoGerador {
	@Inject extension EstudoDeCasoExtensions extensions

	def gerarCodigo(IWorkflowContext ctx, IFileSystemAccess fsa) {
		val List<Model> modeloEstruturaLido = ctx.get("modeloEstruturaLido") as
List<Model>;
		val List<Apresentacao> modeloApresentacaoLido =
ctx.get("modeloApresentacaoLido") as List<Apresentacao>;
		val Navegacao modeloNavegacaoLido = ctx.get("modeloNavegacaoLido") as
Navegacao;
```

```
            main(modeloEstruturaLido.get(0), modeloApresentacaoLido.get(0),
modeloNavegacaoLido, fsa)
      }

      def main(Model m, Apresentacao a, Navegacao n, IFileSystemAccess fsa) {
            for(e:m.entities) {
                  fsa.generateFile("beans/"+e.name+".java", "codigo", e.compile)
            }
      }

      def compile(Entity e) '''
            package beans;
            public class «e.name» {
                  «FOR p:e.properties»
                        «IF !p.many»
                              private «p.type.name» «p.name»;
                              public void «p.setter()»(«p.type.name» «p.name») {
this.«p.name» = «p.name»; }
                              public «p.type.name» «p.getter()»() { return «p.name»; }
                        «ELSE»
                              private java.util.Collection<«p.type.name»> «p.name»;
                              public
void «p.setter()»(java.util.Collection<«p.type.name»> «p.name») { this.«p.name»
= «p.name»; }
                              public java.util.Collection<«p.type.name»> «p.getter()»()
{ return «p.name»; }
                        «ENDIF»
                  «ENDFOR»
            }
      '''
}
```

4.7.5. Rodar o workflow e observar que gerou código para as entidades

4.8. Criar gerador das páginas

4.8.1. Adicionar as seguintes linhas ao arquivo EstudoDeCasoExtensions.xtend

```
package br.org.cursomde.estudodecaso.gerador

import br.org.cursomde.estudodecaso.estrutura.*
import navegacao.*

class EstudoDeCasoExtensions {
      def model(Type t) {
            t.eContainer
      }

      def entities(Model m) {
            m.elements.filter(typeof(Entity))
      }

      def getter(Property p) {
            "get" + p.name.toFirstUpper()
      }

      def setter(Property p) {
            "set" + p.name.toFirstUpper()
      }
```

```
	def area(br.org.cursomde.estudodecaso.apresentacao.Pagina pagina, Navegacao
navegacao) {
		val a = navegacao.paginas.findFirst(e|e.nome==pagina.name).area;
		if(a != null) { a.nome; }
		else { “sem área" }
	}
}
```

4.8.2. Adicionar as seguintes linhas ao arquivo EstudoDeCasoGerador.xtend

```
package br.org.cursomde.estudodecaso.gerador

import br.org.cursomde.estudodecaso.estrutura.*
import br.org.cursomde.estudodecaso.apresentacao.*
import navegacao.*
import org.eclipse.xtext.generator.IGenerator
import java.util.List
import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IFileSystemAccess
import org.eclipse.emf.mwe2.runtime.workflow.IWorkflowContext
import com.google.inject.Inject

class EstudoDeCasoGerador {

	@Inject extension EstudoDeCasoExtensions eext

	def gerarCodigo(IWorkflowContext ctx, IFileSystemAccess fsa) {
		val List<Model> modeloEstruturaLido = ctx.get("modeloEstruturaLido") as
List<Model>;
		val List<Apresentacao> modeloApresentacaoLido =
ctx.get("modeloApresentacaoLido") as List<Apresentacao>;
		val Navegacao modeloNavegacaoLido = ctx.get("modeloNavegacaoLido") as
Navegacao;

		main(modeloEstruturaLido.get(0), modeloApresentacaoLido.get(0),
modeloNavegacaoLido, fsa)
	}

	def main(Model m, Apresentacao a, Navegacao n, IFileSystemAccess fsa) {
		for(e:m.entities) {
			fsa.generateFile("beans/"+e.name+".java", "codigo", e.compile)
		}
		for(p:a.paginas) {
			fsa.generateFile("paginas/"+p.name+".html", "web", p.compile(n,a))
		}
	}

	def compile(Entity e) '''
		package beans;
		public class «e.name» {
			«FOR p:e.properties»
				«IF !p.many»
					private «p.type.name» «p.name»;
					public void «p.setter()»(«p.type.name» «p.name») {
this.«p.name» = «p.name»; }
					public «p.type.name» «p.getter()»() { return «p.name»; }
				«ELSE»
					private java.util.Collection<«p.type.name»> «p.name»;
					public
```

```
void «p.setter()»(java.util.Collection<«p.type.name»> «p.name») { this.«p.name»
= «p.name»; }
                              public java.util.Collection<«p.type.name»> «p.getter()»()
{ return «p.name»; }
                        «ENDIF»
                  «ENDFOR»
            }
      '''

      def compile(br.org.cursomde.estudodecaso.apresentacao.Pagina p, Navegacao n,
Apresentacao a) '''
            <html>
                  <head>
                        <title>«p.titulo»</title>
                  </head>
                  <body>
                        <h1>«p.titulo»</h1>
                        <br/>
                        Você está em: «p.area(n)» > «p.titulo»
                        <hr>
                        <table><tr>
                              <td bgcolor="#CCCCCC">
                                    «menu(n, p.name, a)»
                              </td>
                              <td>
                              <br>
                              «FOR conteudo:p.conteudoPagina»
                                    «FOR paragrafo: conteudo.texto»
                                          <p>«paragrafo»</p>
                                    «ENDFOR»
                              «ENDFOR»
                              <br>
                        </td></tr></table>
                        <hr>
                        Links: «links(n, p.name)»
                  </body>
            </html>
      '''


      def menu(Navegacao n, String nome, Apresentacao a) '''
            «FOR area:n.areas»
                  «area.nome»<br>
                  «FOR pagina:n.paginas.filter(e|e.area==area)»
                          
                        «IF pagina.nome!=nome»<a href="«pagina.nome».html">«ENDIF»
                        «a.paginas.findFirst(e|e.name==pagina.nome).titulo»
                        «IF pagina.nome!=nome»</a>«ENDIF»
                        <br/>
                  «ENDFOR»
            «ENDFOR»
      '''

      def links(Navegacao n, String nome) '''
            «FOR link:n.links.filter(e|e.origem.nome==nome) SEPARATOR "|"»
                  <a href="«link.destino.nome».html">«link.nome»</a>
            «ENDFOR»
      '''
}
```

4.8.3. Rodar o workflow para testar. Observar as páginas geradas
5. Criar transformação M2M
5.1. Adicionar dependência para o plug-in do projeto do questionário
5.1.1. Abrir arquivo META-INF/MANIFEST.MF
5.1.2. Na aba Dependencies, adicionar como plug-in requerido:
- br.org.cursomde.exemplo2
5.2. Copiar, do projeto br.org.cursomde.exemplo4, para a pasta src/model deste projeto (br.org.cursomde.estudodecaso.gerador), o arquivo default.questionario
5.2.1. Clicar com botão direito sobre o arquivo default.questionario e escolher “Initialize questionario_diagram diagram file”
5.3. Criar a classe questionario.GeraParaWeb, que estende AbstractWorkflowComponent

```
package questionario;

import java.util.Arrays;

import navegacao.Area;
import navegacao.Link;
import navegacao.Navegacao;
import navegacao.NavegacaoFactory;

import org.eclipse.emf.ecore.EObject;
import org.eclipse.emf.mwe.core.WorkflowContext;
import org.eclipse.emf.mwe.core.issues.Issues;
import org.eclipse.emf.mwe.core.lib.AbstractWorkflowComponent;
import org.eclipse.emf.mwe.core.monitor.ProgressMonitor;

import br.org.cursomde.estudodecaso.apresentacao.Apresentacao;
import br.org.cursomde.estudodecaso.apresentacao.ApresentacaoFactory;
import br.org.cursomde.estudodecaso.apresentacao.Conteudo;
import br.org.cursomde.estudodecaso.apresentacao.Pagina;
import br.org.cursomde.estudodecaso.estrutura.Entity;
import br.org.cursomde.estudodecaso.estrutura.EstruturaFactory;
import br.org.cursomde.estudodecaso.estrutura.Model;
import br.org.cursomde.estudodecaso.estrutura.Property;
import br.org.cursomde.estudodecaso.estrutura.SimpleType;

public class GeraParaWeb extends AbstractWorkflowComponent {

	@Override
	protected void invokeInternal(WorkflowContext ctx, ProgressMonitor monitor,
			Issues issues) {
		EObject model = (EObject) ctx.get("model");
		Questionario questionario = (Questionario) model;

		Model estrutura = criaModeloEstrutura();
		Navegacao navegacao = criaModeloNavegacao(questionario);
		Apresentacao apresentacao = criaModeloApresentacao(questionario);

		ctx.set("modeloEstruturaLido", Arrays.asList(estrutura));
		ctx.set("modeloNavegacaoLido", navegacao);
		ctx.set("modeloApresentacaoLido", Arrays.asList(apresentacao));
	}

	@Override
	public void checkConfiguration(Issues issues) {
		// TODO Auto-generated method stub
```

```
}

private Model criaModeloEstrutura() {
	Model estrutura = EstruturaFactory.eINSTANCE.createModel();

	SimpleType stringType = EstruturaFactory.eINSTANCE.createSimpleType();
	stringType.setName("String");
	estrutura.getElements().add(stringType);

	Entity perguntaEntity = EstruturaFactory.eINSTANCE.createEntity();
	perguntaEntity.setName("Pergunta");
	estrutura.getElements().add(perguntaEntity);

	Entity respostaEntity = EstruturaFactory.eINSTANCE.createEntity();
	respostaEntity.setName("Resposta");
	estrutura.getElements().add(respostaEntity);

	Property perguntaTexto = EstruturaFactory.eINSTANCE.createProperty();
	perguntaTexto.setName("texto");
	perguntaTexto.setMany(false);
	perguntaTexto.setType(stringType);
	perguntaEntity.getProperties().add(perguntaTexto);

	Property respostaTexto = EstruturaFactory.eINSTANCE.createProperty();
	respostaTexto.setName("texto");
	respostaTexto.setMany(false);
	respostaTexto.setType(stringType);
	respostaEntity.getProperties().add(respostaTexto);

	Property alternativas = EstruturaFactory.eINSTANCE.createProperty();
	alternativas.setName("alternativas");
	alternativas.setMany(true);
	alternativas.setType(respostaEntity);
	perguntaEntity.getProperties().add(alternativas);

	return estrutura;
}

private Navegacao criaModeloNavegacao(Questionario questionario) {
	Navegacao navegacao = NavegacaoFactory.eINSTANCE.createNavegacao();

	Area areaPerguntas = NavegacaoFactory.eINSTANCE.createArea();
	areaPerguntas.setNome("Perguntas");
	navegacao.getAreas().add(areaPerguntas);

	Area areaResultados = NavegacaoFactory.eINSTANCE.createArea();
	areaResultados.setNome("Resultados");
	navegacao.getAreas().add(areaResultados);

	// Cria uma página para cada item
	for (Item item : questionario.getPerguntas()) {
		navegacao.Pagina paginaItem = NavegacaoFactory.eINSTANCE
				.createPagina();
		paginaItem.setNome(transformaMensagemEmNomeDePagina(item
				.getMensagem()));
		if (!item.isFim())
			paginaItem.setArea(areaPerguntas);
		else
			paginaItem.setArea(areaResultados);
```

```
			navegacao.getPaginas().add(paginaItem);
		}

		// Cria um link para cada alternativa
		for (Item item : questionario.getPerguntas()) {
			for (Resposta alternativa : item.getAlternativas()) {
				Link linkAlternativa = NavegacaoFactory.eINSTANCE.createLink();
				linkAlternativa.setNome(alternativa.getTexto());
				linkAlternativa.setOrigem(recuperaPaginaItem(item, navegacao));
				linkAlternativa.setDestino(recuperaPaginaItem(
							alternativa.getProximaPergunta(), navegacao));
				navegacao.getLinks().add(linkAlternativa);
			}
		}

		return navegacao;
	}

	private String transformaMensagemEmNomeDePagina(String mensagem) {
		return mensagem.replaceAll("\\s", "").replaceAll("\\?", "");
	}

	private navegacao.Pagina recuperaPaginaItem(Item item, Navegacao navegacao) {
		for (navegacao.Pagina pagina : navegacao.getPaginas()) {
			if (pagina.getNome().equals(
						transformaMensagemEmNomeDePagina(item.getMensagem()))) {
				return pagina;
			}
		}
		return null;
	}

	private Apresentacao criaModeloApresentacao(Questionario questionario) {
		Apresentacao apresentacao = ApresentacaoFactory.eINSTANCE
				.createApresentacao();

		int itemCount = 1;
		int resultadoCount = 1;
		// Cria uma página para cada item
		for (Item item : questionario.getPerguntas()) {
			Pagina paginaItem = ApresentacaoFactory.eINSTANCE.createPagina();
			paginaItem.setName(transformaMensagemEmNomeDePagina(item
						.getMensagem()));
			if (item.isInicio())
				paginaItem.setTitulo("Início");
			else if (item.isFim())
				paginaItem.setTitulo("Resultado " + (resultadoCount++));
			else
				paginaItem.setTitulo("Item " + (itemCount++));
			apresentacao.getPaginas().add(paginaItem);

			Conteudo conteudoItem = ApresentacaoFactory.eINSTANCE
						.createConteudo();
			conteudoItem.getTexto().add(item.getMensagem());
			paginaItem.getConteudoPagina().add(conteudoItem);
		}

		return apresentacao;
	}
```

```
}
```

5.5. Criar novo arquivo de workflow (GeraCompleto.mwe2)

```
module br.org.cursomde.estudodecaso.gerador.GeraCompleto

import org.eclipse.emf.mwe.utils.*

var codeTargetDir = "src-gen"
var webTargetDir = "html-gen"
var modeloQuestionario = "br.org.cursomde.estudodecaso.gerador/src/model/
default.questionario"

Workflow {
	bean = org.eclipse.emf.mwe.utils.StandaloneSetup {
		platformUri = ".."
		registerGeneratedEPackage = "questionario.QuestionarioPackage"
	}

	component = org.eclipse.emf.mwe.utils.Reader {
		uri = "platform:/resource/${modeloQuestionario}"
		modelSlot = "model"
	}

	component = questionario.GeraParaWeb {}

	component = br.org.cursomde.estudodecaso.gerador.MyGeneratorComponent {
		slot = "modeloEstruturaLido"
		slot = "modeloApresentacaolido"
		slot = "modeloNavegacaoLido"
		outlet = {
			outletName = "codigo"
			path = codeTargetDir
		}
		outlet = {
			outletName = "web"
			path = webTargetDir
		}
	}
}
```

5.6. Rodar o workflow para testar. Observar o código gerado

5.7. Experimentar mudar o questionário e re-gerar para observar as mudanças serem refletidas no código