|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ  НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»  Факультет бизнес-информатики, отделение программной инженерии  Кафедра управления разработкой программного обеспечения | | | | | |
| СОГЛАСОВАНО  Старший преподаватель каф. УРПО отделения Программной инженерии Национального исследовательского университета «Высшая школа экономики»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Л.В. Дворянский  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. | |  | УТВЕРЖДАЮ  Заведующий отделением Программной инженерии Национального исследовательского университета «Высшая школа экономики»  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С.М. Авдошин  «\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. | | |
| **«СИСТЕМА КОДОГЕНЕРАЦИИ ИЗ МОДЕЛИ ВЛОЖЕННЫХ СЕТЕЙ ПЕТРИ В СИСТЕМЫ КОМПОНЕНТ НА БАЗЕ ENTERPRISE JAVABEANS»**  **«CODE GENERATION SYSTEM FROM NESTED PETRI NETS TO SYSTEMS OF ENTERPRISE JAVABEANS COMPONENTS»**  Текст программы | | | | | |
| Подп. и дата |  | **ЛИСТ УТВЕРЖДЕНИЯ**  **RU.17701729. 501620-01 12 01-1-ЛУ**  **Листов 35** | | | | | |
| Инв. № дубл. |  |  | | | | | |
| Взам. инв. № |  |  |  | | | Исполнитель  студент группы 372 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_/Николенко А.П. /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_ 2013 г. | |
| Подп. и дата |  |  | | | | |  |
| Инв. № подл. |  | **2013** | | | | | |

|  |  |  |
| --- | --- | --- |
|  |  | УТВЕРЖДЕН  RU.17701729. 501620-01 12 01-1-ЛУ  «Система кодогенерации из модели вложенных сетей Петри в системы компонент на базе Enterprise JavaBeans»  «Code Generation System from Nested Petri Nets to Systems of Enterprise JavaBeans Components»  Текст программы  RU.17701729. 501620-01 12 01-1    Листов 35 |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. № подл. |  |

2013

Оглавление

[1. Основная программа 1](#_Toc367937772)

[Лист регистрации изменений. 2](#_Toc367937773)

# Основная программа

## 1.1. EJBGenerator.rb

require 'nokogiri'

require 'fileutils'

**def** **generate(**filepath**)**

filePath **=** filepath

fileDir **=** filePath**.**slice**(**0**..**filePath**.**rindex**(**'\\'**))**

f **=** File**.**open**(**filePath**)**

doc **=** Nokogiri**::**XML**(**f**)**

f**.**close

doc **=** doc**.**xpath**(**'npnets:PetriNetNestedMarked'**)**

systemNet **=** doc**.**xpath**(**'child::net'**)[**0**]**

elementNets **=** systemNet**.**xpath**(**'//typeElementNet'**)**

package **=** systemNet**.**xpath**(**'netSystem'**)[**0**][**'name'**].**gsub**(**' '**,** '\_'**)**

genPath **=** fileDir **+** "output\\java\\**#{**package**}**"

**if** **(!**File**.**directory?**(**genPath**))**

FileUtils**.**mkdir\_p genPath

**end**

systemPlaces **=** systemNet**.**xpath**(**'netSystem/places'**)**

**(**0**..**systemPlaces**.**count**-**1**).**each **do** **|**i**|**

placeName **=** systemPlaces**[**i**][**'name'**]**

arcsIds **=** systemPlaces**[**i**][**'outArcs'**].**delete**(**'#'**).**split

placeType **=** systemPlaces**[**i**][**'type'**].**delete**(**'#'**)**

transitionsNames **=** **[]**

**(**0**..**arcsIds**.**count**-**1**).**each **do** **|**i**|**

transitionsId **=** systemNet**.**xpath**(**"netSystem/arcsPT[@id='**#{**arcsIds**[**i**]}**']/@outTransition"**).**to\_s**.**delete**(**'#'**)**

transitionsNames**.**push**(**systemNet**.**xpath**(**"netSystem/transitions[@id='**#{**transitionsId**}**']/@name"**).**to\_s**)**

**end**

transitionsBlock **=** ''

notifyBlock **=** ''

**(**0**..**transitionsNames**.**count**-**1**).**each **do** **|**i**|**

transitionsBlock **<<** "@EJB\n**#{**transitionsNames**[**i**]}**Bean **#{**transitionsNames**[**i**]}**;\n"

notifyBlock **<<** "**#{**transitionsNames**[**i**]}**.reciveNotification(name, true);\n"

**end**

variablesBlock **=** ""

createBlock **=**""

**if** **(**systemNet**.**xpath**(**"typeAtomic[@id='**#{**placeType**}**']"**).**count **==** 1**)**

variablesBlock **=** "String type = \"black\";

Integer idCounter = 0;"

createBlock**=**"public void createToken() {

idCounter++;

list.add(type + \"\_\" + name + \"\_\" + idCounter);

notifyTransitions();

}"

**end**

initBlock **=** ""

markings **=** doc**.**xpath**(**"marking/map[@place='#**#{**systemPlaces**[**i**][**'id'**]}**']"**)**

**(**0**..**markings**.**count**-**1**).**each **do** **|**i**|**

markingType **=** markings**[**i**].**xpath**(**"marking/@type"**)[**0**].**to\_s**.**delete**(**'#'**)**

**if** **(**systemNet**.**xpath**(**"typeAtomic[@id='**#{**markingType**}**']"**).**count **==** 1**)**

weight **=** markings**[**i**].**xpath**(**"marking/weight/@weight"**)[**0**].**to\_s**.**to\_i

**(**0**..**weight**-**1**).**each **do** **|**j**|**

initBlock **<<** "createToken();\n"

**end**

**else**

token **=** markings**[**i**].**xpath**(**"marking/weight/@token"**)[**0**].**to\_s**.**delete**(**'#'**)**

weight **=** markings**[**i**].**xpath**(**"marking/weight/@weight"**)[**0**].**to\_s**.**to\_i

marking **=** doc**.**xpath**(**"//tokenNets[@id=**#{**token**}**]/@value"**)[**0**].**to\_s**.**delete**(**'#'**)**

net **=** doc**.**xpath**(**"//tokenNets[@id=**#{**token**}**]/ancestor::typeElementNet/@name"**)[**0**]**

**if** **(!**variablesBlock**.**include?**(**"**#{**net**}**ManagerBean"**))**

variablesBlock **<<** "@EJB\n**#{**net**}**ManagerBean **#{**net**}**Manager;\n"

**end**

**(**0**..**weight**-**1**).**each **do** **|**j**|**

initBlock **<<** "addToken(**#{**net**}**Manager.createToken(\"**#{**marking**}**\"));\n"

**end**

**end**

**if** **(**i **==** markings**.**count **-** 1**)**

initBlock **<<** "notifyTransitions();\n"

**end**

**end**

placeBean **=** "package **#{**package**}**;

import java.util.ArrayList;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import java.util.Timer;

import java.util.TimerTask;

import javax.ejb.Singleton;

import javax.ejb.Startup;

@Startup

@Singleton

public class **#{**placeName**}**Bean {

**#{**transitionsBlock**}**

List<String> list = new ArrayList<String>();

String name = \"**#{**placeName**}**\";

**#{**variablesBlock**}**

Timer timer = new Timer();

String blocked = \"\";

@PostConstruct

void init() {

timer.schedule(new TimerTask() {

@Override

public void run() {

**#{**initBlock**}**

}

}, 1000);

}

public void notifyTransitions() {

if(!list.isEmpty())

{

**#{**notifyBlock**}**

} else

{

**#{**notifyBlock**.**gsub**(**'true'**,** 'false'**)}**

}

}

public Boolean blockPosition(String name) {

if((blocked.equals(\"\") || blocked.equals(name)) && (!list.isEmpty()))

{

blocked = name;

return true;

}

return false;

}

public void unblockPosition(String name) {

if(blocked.equals(name) || blocked.equals(\"\"))

{

blocked = \"\";

notifyTransitions();

}

}

public String getFirstToken() {

return list.get(0);

}

public List<String> getTokens() {

return list;

}

public void removeToken(String id) {

list.remove(list.indexOf(id));

}

public void addToken(String id) {

list.add(id);

if(blocked.equals(\"\"))

notifyTransitions();

}

**#{**createBlock**}**

}

"

File**.**open**(**"**#{**genPath**}**\\**#{**placeName**}**Bean.java"**,** 'w'**)** **do** **|**f**|**

f**.**puts placeBean

**end**

**end**

systemTransitions **=** systemNet**.**xpath**(**'netSystem/transitions'**)**

**(**0**..**systemTransitions**.**count**-**1**).**each **do** **|**i**|**

transName **=** systemTransitions**[**i**][**'name'**]**

arcsPTIds **=** systemTransitions**[**i**][**'inArcs'**].**delete**(**'#'**).**split

arcsTPIds **=** systemTransitions**[**i**][**'outArcs'**].**delete**(**'#'**).**split

synch **=** **{}**

**if** **(!**systemTransitions**[**i**][**'synchronization'**].**nil?**)**

synchId **=** systemTransitions**[**i**][**'synchronization'**].**delete**(**'#'**).**split

synchName **=** elementNets**.**xpath**(**"net/transitions[@synchronization='#**#{**synchId**[**0**]}**']/@name"**).**to\_s

synchType **=** elementNets**.**xpath**(**"net/transitions[@synchronization='#**#{**synchId**[**0**]}**']/ancestor::typeElementNet/@id"**).**to\_s

synch**.**store**(**synchName**,** synchType**)**

**end**

inputVariables **=** **{}**

inputTypes **=** **{}**

**(**0**..**arcsPTIds**.**count**-**1**).**each **do** **|**i**|**

placesId **=** systemNet**.**xpath**(**"netSystem/arcsPT[@id='**#{**arcsPTIds**[**i**]}**']/@inPlace"**).**to\_s**.**delete**(**'#'**)**

variableId **=** systemNet**.**xpath**(**"netSystem/arcsPT[@id='**#{**arcsPTIds**[**i**]}**']/inscription/monoms/@variable"**).**to\_s**.**delete**(**'#'**)**

inputName **=** systemNet**.**xpath**(**"netSystem/places[@id='**#{**placesId**}**']/@name"**).**to\_s

inputVariable **=** systemNet**.**xpath**(**"netSystem/transitions/variables[@id='**#{**variableId**}**']/@name"**).**to\_s

inputType **=** systemNet**.**xpath**(**"netSystem/places[@id='**#{**placesId**}**']/@type"**).**to\_s

inputVariables**.**store**(**inputName**,** inputVariable**)**

inputTypes**.**store**(**inputName**,** inputType**)**

**end**

outputs **=** **{}**

constants **={}**

**(**0**..**arcsTPIds**.**count**-**1**).**each **do** **|**i**|**

placesId **=** systemNet**.**xpath**(**"netSystem/arcsTP[@id='**#{**arcsTPIds**[**i**]}**']/@outPlace"**).**to\_s**.**delete**(**'#'**)**

variableId **=** systemNet**.**xpath**(**"netSystem/arcsTP[@id='**#{**arcsTPIds**[**i**]}**']/inscription/monoms/@variable"**).**to\_s**.**delete**(**'#'**)**

outputName **=** systemNet**.**xpath**(**"netSystem/places[@id='**#{**placesId**}**']/@name"**).**to\_s

outputVariable **=** systemNet**.**xpath**(**"netSystem/transitions/variables[@id='**#{**variableId**}**']/@name"**).**to\_s

constantValue **=** systemNet**.**xpath**(**"netSystem/arcsTP[@id='**#{**arcsTPIds**[**i**]}**']/inscription/monomConstants/@value"**).**to\_s**.**delete**(**'#'**)**

constantMarking **=** doc**.**xpath**(**"//tokenNets[@id='**#{**constantValue**}**']/@value"**).**to\_s**.**delete**(**'#'**)**

outputs**.**store**(**outputName**,** outputVariable**)**

**if** **(**constantMarking!**=**''**)**

constants**.**store**(**outputName**,** constantMarking**)**

**end**

**end**

puts constants

variablesBlock **=** ''

variables **=** inputVariables**.**merge**(**outputs**).**merge**(**synch**)**

**(**0**..**variables**.**count**-**1**).**each **do** **|**i**|**

variablesBlock **<<** "@EJB\n**#{**variables**.**keys**[**i**]}**Bean **#{**variables**.**keys**[**i**]}**;\n"

**end**

inputsBlock **=** ''

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

inputsBlock **<<** "inputs.put(\"**#{**inputVariables**.**keys**[**i**]}**\", false);\n"

**end**

blockBlock **=** "blocked = blocked"

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

blockBlock **<<** " && **#{**inputVariables**.**keys**[**i**]}**.blockPosition(name)"

**end**

blockBlock **<<** ";"

unblockBlock **=** ""

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

unblockBlock **<<** "**#{**inputVariables**.**keys**[**i**]}**.unblockPosition(name);\n"

**end**

triggerBlock **=** ''

**if** **(**synch**.**count **==** 0**)**

triggerBlock **<<** "String token;\n"

blackInput **=** inputVariables**.**select **{** **|**k**,** v**|** systemNet**.**xpath**(**"typeAtomic[@id='**#{**inputTypes**[**k**].**delete**(**'#'**)}**']"**).**count **==** 1 **}**

**(**0**..**blackInput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = **#{**blackInput**.**keys**[**i**]}**.getFirstToken();\n**#{**blackInput**.**keys**[**i**]}**.removeToken(token);\n"

**end**

blackOutput**=** outputs**.**select **{** **|**k**,** v**|** blackInput**.**has\_value?**(**v**)** **}**

**(**0**..**blackOutput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "**#{**blackOutput**.**keys**[**i**]}**.createToken();\n"

**end**

**(**0**..**constants**.**count**-**1**).**each **do** **|**i**|**

netManager **=** doc**.**xpath**(**"//elementNetMarkeds[@id='**#{**constants**.**values**[**i**]}**']/ancestor::typeElementNet/@name"**).**to\_s

**if** **(!**variablesBlock**.**include?**(**"**#{**netManager**}**ManagerBean"**))**

variablesBlock **<<** "@EJB\n**#{**netManager**}**ManagerBean **#{**netManager**}**Manager;\n"

**end**

triggerBlock **<<** "token = **#{**netManager**}**Manager.createToken(\"**#{**constants**.**values**[**i**]}**\");\n"

triggerBlock **<<** "**#{**constants**.**keys**[**i**]}**.addToken(token);\n"

**end**

nonblackInput **=** inputVariables**.**select **{** **|**k**,** v**|** **!**blackInput**.**has\_key?**(**k**)** **}**

**(**0**..**nonblackInput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = **#{**nonblackInput**.**keys**[**i**]}**.getFirstToken();\n**#{**nonblackInput**.**keys**[**i**]}**.removeToken(token);\n"

nonblackOutput **=** outputs**.**select **{** **|**k**,** v**|** v **==** nonblackInput**.**values**[**i**]** **}**

**if** **(**nonblackOutput**.**count **==** 0**)**

netManagerType **=** systemNet**.**xpath**(**"netSystem/places[@name='**#{**nonblackOutput**.**keys**[**0**]}**']/@type"**).**to\_s**.**delete**(**'#'**)**

netManager **=** doc**.**xpath**(**"typeElementNet[@id='**#{**netManagerType**}**']/@name"**)**

**if** **(!**variablesBlock**.**include?**(**"**#{**netManager**[**0**]}**ManagerBean"**))**

variablesBlock **<<** "@EJB\n**#{**netManager**[**0**]}**ManagerBean **#{**netManager**[**0**]}**Manager;\n"

**end**

triggerBlock **<<** "**#{**netManager**}**Manager.removeToken(token);\n"

**elsif** **(**nonblackOutput**.**count **==** 1**)**

triggerBlock **<<** "**#{**nonblackOutput**.**keys**[**0**]}**.addToken(token);\n"

**elsif** **(**nonblackOutput**.**count **>** 1**)**

netManagerType **=** systemNet**.**xpath**(**"netSystem/places[@name='**#{**nonblackOutput**.**keys**[**0**]}**']/@type"**).**to\_s**.**delete**(**'#'**)**

netManager **=** doc**.**xpath**(**"typeElementNet[@id='**#{**netManagerType**}**']/@name"**)**

**if** **(!**variablesBlock**.**include?**(**"**#{**netManager**[**0**]}**ManagerBean"**))**

variablesBlock **<<** "@EJB\n**#{**netManager**[**0**]}**ManagerBean **#{**netManager**[**0**]}**Manager;\n"

**end**

triggerBlock **<<** "**#{**nonblackOutput**.**keys**[**0**]}**.addToken(token);\n"

**(**1**..**nonblackOutput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = **#{**netManager**}**Manager.cloneToken(token);\n"

triggerBlock **<<** "**#{**nonblackOutput**.**keys**[**i**]}**.addToken(token);\n"

**end**

**end**

**end**

triggerBlock **=** "**#{**triggerBlock**}**doExtraStuff();\nunblockInputs();\n"

**else**

triggerBlock **<<** 'boolean fullSynch = true;

List<String> synch = new ArrayList<String>();

List<String> keys = new ArrayList<String>();'

synchedInputs **=** inputVariables**.**select **{** **|**k**,** v**|** synch**.**has\_value?**(**inputTypes**[**k**].**delete**(**'#'**))** **}**

synchedElements **=** **[]**

**(**0**..**synchedInputs**.**count**-**1**).**each **do** **|**i**|**

element **=** synch**.**keys**[**synch**.**values**.**index**(**inputTypes**[**synchedInputs**.**keys**[**i**]].**delete**(**'#'**))]**

**if** **(!**synchedElements**.**include?**(**element**))**

synchedElements**.**push**(**element**)**

**end**

triggerBlock **<<** "keys = **#{**synchedInputs**.**keys**[**i**]}**.getTokens();

fullSynch = false;

for(int i = 0; i<keys.size(); i++)

{

if(**#{**element**}**.synchronize(keys.get(i), name))

{

synch.add(keys.get(i));

fullSynch = true;

break;

}

}

if(!fullSynch)

{

synch.add(\"\");

}"

**end**

triggerBlock **<<** "fullSynch = true;

for(int i = 0; i<synch.size(); i++)

if(synch.get(i).equals(\"\"))

fullSynch = false;

if(fullSynch)

{

String token;"

**(**0**..**synchedElements**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "**#{**synchedElements**[**i**]}**.triggerTransition(synch.get(**#{**i**}**));\n"

**end**

blackInput **=** inputVariables**.**select **{** **|**k**,** v**|** systemNet**.**xpath**(**"typeAtomic[@id='**#{**inputTypes**[**k**].**delete**(**'#'**)}**']"**).**count **==** 1 **}**

**(**0**..**blackInput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = **#{**blackInput**.**keys**[**i**]}**.getFirstToken();\n**#{**blackInput**.**keys**[**i**]}**.removeToken(token);\n"

**end**

blackOutput**=** outputs**.**select **{** **|**k**,** v**|** blackInput**.**has\_value?**(**v**)** **}**

**(**0**..**blackOutput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "**#{**blackOutput**.**keys**[**i**]}**.createToken();\n"

**end**

nonblackInput **=** inputVariables**.**select **{** **|**k**,** v**|** **!**blackInput**.**has\_key?**(**k**)** **}**

**(**0**..**nonblackInput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = synch.get(**#{**i**}**);\n**#{**nonblackInput**.**keys**[**i**]}**.removeToken(token);\n"

nonblackOutput **=** outputs**.**select **{** **|**k**,** v**|** v **==** nonblackInput**.**values**[**i**]** **}**

**if** **(**nonblackOutput**.**count **==** 0**)**

netManagerType **=** systemNet**.**xpath**(**"netSystem/places[@name='**#{**nonblackInput**.**keys**[**0**]}**']/@type"**).**to\_s**.**delete**(**'#'**)**

netManager **=** doc**.**xpath**(**"//typeElementNet[@id='**#{**netManagerType**}**']/@name"**)**

**if** **(!**variablesBlock**.**include?**(**"**#{**netManager**}**ManagerBean"**))**

variablesBlock **<<** "@EJB\n**#{**netManager**}**ManagerBean **#{**netManager**}**Manager;\n"

**end**

triggerBlock **<<** "**#{**netManager**}**Manager.removeToken(token);\n"

**elsif** **(**nonblackOutput**.**count **==** 1**)**

triggerBlock **<<** "**#{**nonblackOutput**.**keys**[**0**]}**.addToken(token);\n"

**elsif** **(**nonblackOutput**.**count **>** 1**)**

netManagerType **=** systemNet**.**xpath**(**"netSystem/places[@name='**#{**nonblackOutput**.**keys**[**0**]}**']/@type"**).**to\_s**.**delete**(**'#'**)**

netManager **=** doc**.**xpath**(**"typeElementNet[@id='**#{**netManagerType**}**']/@name"**)**

**if** **(!**variablesBlock**.**include?**(**"**#{**netManager**[**0**]}**ManagerBean"**))**

variablesBlock **<<** "@EJB\n**#{**netManager**[**0**]}**ManagerBean **#{**netManager**[**0**]}**Manager;\n"

**end**

triggerBlock **<<** "**#{**nonblackOutput**.**keys**[**0**]}**.addToken(token);\n"

**(**1**..**nonblackOutput**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = **#{**netManager**}**Manager.cloneToken(token);\n"

triggerBlock **<<** "**#{**nonblackOutput**.**keys**[**i**]}**.addToken(token);\n"

**end**

**end**

**end**

triggerBlock **<<** 'doExtraStuff();

unblockInputs();

} else {'

**(**0**..**synchedElements**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "**#{**synchedElements**[**i**]}**.removeSynchronization(synch.get(**#{**i**}**), name);\n"

**end**

triggerBlock **<<** 'if (!waiting4Check) {

Timer timer = timerService.createTimer(duration, null);

waiting4Check = true;

}

unblockInputs();

}'

**end**

importBlock **=** ''

extraBlock **=** ''

actionBlock **=** ''

**if** **(**File**.**exists?**(**fileDir**+**transName**+**'.iea'**))**

part **=** '%import%'

File**.**open**(**fileDir**+**transName**+**'.iea'**,** "r"**)** **do** **|**infile**|**

**while** **(**line **=** infile**.**gets**)**

**if** **(**line**[**0**]** **==** '%'**)**

part **=** line**.**chop

**else**

**if** **(**part **==** '%import%'**)**

importBlock **<<** line

**elsif** **(**part **==** '%extra%'**)**

extraBlock **<<** line

**elsif** **(**part **==** '%action%'**)**

actionBlock **<<** line

**end**

**end**

**end**

**end**

**end**

**if** **(**extraBlock **==** ''**)**

extraBlock **=** "return true;\n"

**end**

transBean **=** "package **#{**package**}**;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.annotation.PostConstruct;

import javax.annotation.Resource;

import javax.ejb.EJB;

import javax.ejb.Singleton;

import javax.ejb.Startup;

import javax.ejb.Timeout;

import javax.ejb.Timer;

import javax.ejb.TimerService;

**#{**importBlock**}**

@Startup

@Singleton

public class **#{**transName**}**Bean {

**#{**variablesBlock**}**

@Resource

TimerService timerService;

long duration = 5;

Map<String, Boolean> inputs = new HashMap<String, Boolean>();

String name = \"**#{**transName**}**\";

boolean inWork = false;

boolean waiting4Check = false;

@PostConstruct

void init() {

**#{**inputsBlock**}**

}

public void reciveNotification(String name, Boolean status) {

inputs.put(name, status);

checkInputs();

}

@Timeout

public void timeout() {

waiting4Check = false;

checkInputs();

}

private void checkInputs()

{

if(inWork)

{

if (!waiting4Check) {

Timer timer = timerService.createTimer(duration, null);

waiting4Check = true;

}

return;

}

inWork = true;

boolean inputsFull = true;

List<Boolean> list = new ArrayList<Boolean>(inputs.values());

for(int i = 0; i < inputs.size(); i++)

{

inputsFull = inputsFull && list.get(i);

}

if(inputsFull)

{

triggerTransition();

}

inWork = false;

}

private void triggerTransition() {

if (blockInputs() && checkExtra()) {

**#{**triggerBlock**}**

} else {

if (!waiting4Check) {

Timer timer = timerService.createTimer(duration, null);

waiting4Check = true;

}

}

}

private boolean blockInputs() {

boolean blocked = true;

**#{**blockBlock**}**

if (!blocked) {

unblockInputs();

}

return blocked;

}

private void unblockInputs()

{

**#{**unblockBlock**}**

}

private boolean checkExtra() {

**#{**extraBlock**}**

}

private void doExtraStuff() {

**#{**actionBlock**}**

}

}"

File**.**open**(**"**#{**genPath**}**\\**#{**transName**}**Bean.java"**,** 'w'**)** **do** **|**f**|**

f**.**puts transBean

**end**

**end**

**(**0**..**elementNets**.**count**-**1**).**each **do** **|**en**|**

elementPlaces **=** elementNets**[**en**].**xpath**(**'net/places'**)**

**(**0**..**elementPlaces**.**count**-**1**).**each **do** **|**j**|**

placeName **=** elementPlaces**[**j**][**'name'**]**

arcsIds **=** elementPlaces**[**j**][**'outArcs'**].**delete**(**'#'**).**split

placeType **=** elementPlaces**[**j**][**'type'**].**delete**(**'#'**)**

transitionsNames **=** **[]**

**(**0**..**arcsIds**.**count**-**1**).**each **do** **|**j**|**

transitionsId **=** elementNets**[**en**].**xpath**(**"net/arcsPT[@id='**#{**arcsIds**[**j**]}**']/@outTransition"**).**to\_s**.**delete**(**'#'**)**

transitionsNames**.**push**(**elementNets**[**en**].**xpath**(**"net/transitions[@id='**#{**transitionsId**}**']/@name"**).**to\_s**)**

**end**

transitionsBlock **=** ''

notifyBlock **=** ''

**(**0**..**transitionsNames**.**count**-**1**).**each **do** **|**i**|**

transitionsBlock **<<** "@EJB\n**#{**transitionsNames**[**i**]}**Bean **#{**transitionsNames**[**i**]}**;\n"

notifyBlock **<<** "**#{**transitionsNames**[**i**]}**.reciveNotification(keys.get(i), name, true);\n"

**end**

placeBean **=** "package **#{**package**}**;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.ejb.Singleton;

import javax.ejb.Startup;

@Startup

@Singleton

public class **#{**placeName**}**Bean {

**#{**transitionsBlock**}**

List<String> keys = new ArrayList<String>();

Map<String, List<String>> list = new HashMap<String, List<String>>();

String type = \"black\";

String name = \"**#{**placeName**}**\";

Integer idCounter = 0;

Map<String, String> blocked = new HashMap<String, String>();

@PostConstruct

void init() {

}

public void notifyTransitions() {

List<String> tmpList;

for (int i = 0; i < keys.size(); i++) {

tmpList = list.get(keys.get(i));

if (!tmpList.isEmpty()) {

**#{**notifyBlock**}**

} else {

**#{**notifyBlock**.**gsub**(**'true'**,** 'false'**)}**

}

}

}

public Boolean blockPosition(String key, String name) {

if((blocked.get(key).equals(\"\") || blocked.get(key).equals(name)) && (!list.get(key).isEmpty()))

{

blocked.put(key, name);

return true;

}

return false;

}

public void unblockPosition(String key, String name) {

if(blocked.get(key).equals(name))

{

blocked.put(key, \"\");

notifyTransitions();

}

}

public String getFirstToken(String key) {

return list.get(key).get(0);

}

public List<String> getTokens(String key) {

return list.get(key);

}

public void removeToken(String key, String id) {

list.get(key).remove(list.get(key).indexOf(id));

}

public void addToken(String key, String id) {

list.get(key).add(id);

if(blocked.get(key).equals(\"\"))

notifyTransitions();

}

public void createToken(String key) {

idCounter++;

list.get(key).add(type + \"\_\" + name + \"\_\" + idCounter);

notifyTransitions();

}

public void createNetToken(String key)

{

keys.add(key);

list.put(key, new ArrayList<String>());

blocked.put(key, \"\");

}

public void destroyNetToken(String key)

{

keys.remove(key);

list.remove(key);

blocked.remove(key);

}

}

"

File**.**open**(**"**#{**genPath**}**\\**#{**placeName**}**Bean.java"**,** 'w'**)** **do** **|**f**|**

f**.**puts placeBean

**end**

**end**

elementTransitions **=** elementNets**[**en**].**xpath**(**'net/transitions'**)**

**(**0**..**elementTransitions**.**count**-**1**).**each **do** **|**i**|**

transName **=** elementTransitions**[**i**][**'name'**]**

arcsPTIds **=** elementTransitions**[**i**][**'inArcs'**].**delete**(**'#'**).**split

arcsTPIds **=** elementTransitions**[**i**][**'outArcs'**].**delete**(**'#'**).**split

synch **=** **false**

**if** **(!**elementTransitions**[**i**][**'synchronization'**].**nil?**)**

synch **=** **true**

**end**

inputVariables **=** **{}**

**(**0**..**arcsPTIds**.**count**-**1**).**each **do** **|**i**|**

placesId **=** elementNets**[**en**].**xpath**(**"net/arcsPT[@id='**#{**arcsPTIds**[**i**]}**']/@inPlace"**).**to\_s**.**delete**(**'#'**)**

variableId **=** elementNets**[**en**].**xpath**(**"net/arcsPT[@id='**#{**arcsPTIds**[**i**]}**']/inscription/monoms/@variable"**).**to\_s**.**delete**(**'#'**)**

inputName **=** elementNets**[**en**].**xpath**(**"net/places[@id='**#{**placesId**}**']/@name"**).**to\_s

inputVariable **=** elementNets**[**en**].**xpath**(**"net/transitions/variables[@id='**#{**variableId**}**']/@name"**).**to\_s

inputVariables**.**store**(**inputName**,** inputVariable**)**

**end**

outputs **=** **{}**

**(**0**..**arcsTPIds**.**count**-**1**).**each **do** **|**i**|**

placesId **=** elementNets**[**en**].**xpath**(**"net/arcsTP[@id='**#{**arcsTPIds**[**i**]}**']/@outPlace"**).**to\_s**.**delete**(**'#'**)**

variableId **=** elementNets**[**en**].**xpath**(**"net/arcsTP[@id='**#{**arcsTPIds**[**i**]}**']/inscription/monoms/@variable"**).**to\_s**.**delete**(**'#'**)**

outputName **=** elementNets**[**en**].**xpath**(**"net/places[@id='**#{**placesId**}**']/@name"**).**to\_s

outputVariable **=** elementNets**[**en**].**xpath**(**"net/transitions/variables[@id='**#{**variableId**}**']/@name"**).**to\_s

outputs**.**store**(**outputName**,** outputVariable**)**

**end**

variablesBlock **=** ''

variables **=** inputVariables**.**merge**(**outputs**)**

**(**0**..**variables**.**count**-**1**).**each **do** **|**i**|**

variablesBlock **<<** "@EJB\n**#{**variables**.**keys**[**i**]}**Bean **#{**variables**.**keys**[**i**]}**;\n"

**end**

inputsBlock **=** ''

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

inputsBlock **<<** "map.put(\"**#{**inputVariables**.**keys**[**i**]}**\", false);\n"

**end**

blockBlock **=** "blocked = blocked"

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

blockBlock **<<** " && **#{**inputVariables**.**keys**[**i**]}**.blockPosition(key, name)"

**end**

blockBlock **<<** ";"

unblockBlock **=** ""

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

unblockBlock **<<** "**#{**inputVariables**.**keys**[**i**]}**.unblockPosition(key, name);\n"

**end**

triggerBlock **=** ""

triggerBlock **<<** "String token;\n"

**(**0**..**inputVariables**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "token = **#{**inputVariables**.**keys**[**i**]}**.getFirstToken(key);\n**#{**inputVariables**.**keys**[**i**]}**.removeToken(key, token);\n"

**end**

**(**0**..**outputs**.**count**-**1**).**each **do** **|**i**|**

triggerBlock **<<** "**#{**outputs**.**keys**[**i**]}**.createToken(key);\n"

**end**

importBlock **=** ''

extraBlock **=** ''

actionBlock **=** ''

**if** **(**File**.**exists?**(**fileDir**+**transName**+**'.iea'**))**

part **=** '%import%'

File**.**open**(**fileDir**+**transName**+**'.iea'**,** "r"**)** **do** **|**infile**|**

**while** **(**line **=** infile**.**gets**)**

**if** **(**line**[**0**]** **==** '%'**)**

part **=** line**.**chop

**else**

**if** **(**part **==** '%import%'**)**

importBlock **<<** line

**elsif** **(**part **==** '%extra%'**)**

extraBlock **<<** line

**elsif** **(**part **==** '%action%'**)**

actionBlock **<<** line

**end**

**end**

**end**

**end**

**end**

**if** **(**extraBlock **==** ''**)**

extraBlock **=** "return true;\n"

**end**

**if** **(!**synch**)**

transBean **=** "

package **#{**package**}**;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.annotation.PostConstruct;

import javax.annotation.Resource;

import javax.ejb.EJB;

import javax.ejb.Singleton;

import javax.ejb.Startup;

import javax.ejb.Timeout;

import javax.ejb.Timer;

import javax.ejb.TimerService;

**#{**importBlock**}**

@Startup

@Singleton

public class **#{**transName**}**Bean {

**#{**variablesBlock**}**

@Resource

TimerService timerService;

long duration = 50;

Map<String, Map<String, Boolean>> inputs = new HashMap<String, Map<String, Boolean>>();

String name = \"**#{**transName**}**\";

boolean inWork = false;

List<String> waiting4Check = new ArrayList<String>() ;

@PostConstruct

void init() {

}

public void reciveNotification(String key, String name, Boolean status) {

inputs.get(key).put(name, status);

checkInputs(key);

}

@Timeout

public void timeout() {

if (!waiting4Check.isEmpty()) {

String waitingKey = waiting4Check.get(0);

waiting4Check.remove(waitingKey);

checkInputs(waitingKey);

}

}

private void checkInputs(String key)

{

if(inWork)

{

if(!waiting4Check.contains(key))

{

Timer timer = timerService.createTimer(duration, null);

waiting4Check.add(key);

}

return;

}

inWork = true;

boolean inputsFull = true;

List<Boolean> list = new ArrayList<Boolean>(inputs.get(key).values());

for (int i = 0; i < inputs.get(key).size(); i++) {

inputsFull = inputsFull && list.get(i);

}

if (inputsFull) {

triggerTransition(key);

} else {

unblockInputs(key);

}

inWork = false;

//checkInputs();

}

private void triggerTransition(final String key)

{

if(!blockInputs(key))

{

Timer timer = timerService.createTimer(duration, null);

waiting4Check.add(key);

} else if(checkExtra(key))

{

**#{**triggerBlock**}**

doExtraStuff(key);

unblockInputs(key);

}

}

private boolean blockInputs(String key) {

boolean blocked = true;

**#{**blockBlock**}**

if (!blocked) {

unblockInputs(key);

}

return blocked;

}

private void unblockInputs(String key) {

**#{**unblockBlock**}**

}

private boolean checkExtra(String key) {

**#{**extraBlock**}**

}

private void doExtraStuff(String key) {

**#{**actionBlock**}**

}

public void createNetToken(String key)

{

Map<String, Boolean> map = new HashMap<String, Boolean>();

**#{**inputsBlock**}**

inputs.put(key, map);;

}

public void destroyNetToken(String key)

{

inputs.remove(key);

waiting4Check.remove(key);

}

}"

**else**

transBean **=** "package **#{**package**}**;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.ejb.Singleton;

import javax.ejb.Startup;

**#{**importBlock**}**

@Startup

@Singleton

public class **#{**transName**}**Bean {

**#{**variablesBlock**}**

Map<String, Map<String, Boolean>> inputs = new HashMap<String, Map<String, Boolean>>();

String name = \"**#{**transName**}**\";

Map<String, String> synch = new HashMap<String, String>();

@PostConstruct

void init() {

}

public void reciveNotification(String key, String name, Boolean status) {

inputs.get(key).put(name, status);

//checkInputs(key);

}

public boolean synchronize(String key, String name) {

if (synch.get(key).equals(\"\") || synch.get(key).equals(name)) {

boolean inputsFull = true;

List<Boolean> list = new ArrayList<Boolean>(inputs.get(key).values());

for (int i = 0; i < inputs.get(key).size(); i++) {

inputsFull = inputsFull && list.get(i);

}

if (inputsFull && blockInputs(key) && checkExtra(key)) {

synch.put(key, name);

return true;

} else {

unblockInputs(key);

}

return false;

} else

{

return false;

}

}

public void removeSynchronization(String key, String name) {

if(!key.equals(\"\") && synch.get(key).equals(name))

{

synch.put(key, \"\");

unblockInputs(key);

}

}

public void triggerTransition(String key) {

**#{**triggerBlock**}**

doExtraStuff(key);

unblockInputs(key);

synch.put(key, \"\");

}

private boolean blockInputs(String key) {

boolean blocked = true;

**#{**blockBlock**}**

if (!blocked) {

unblockInputs(key);

}

return blocked;

}

private void unblockInputs(String key) {

**#{**unblockBlock**}**

}

private boolean checkExtra(String key) {

**#{**extraBlock**}**

}

private void doExtraStuff(String key) {

**#{**actionBlock**}**

}

public void createNetToken(String key)

{

Map<String, Boolean> map = new HashMap<String, Boolean>();

**#{**inputsBlock**}**

inputs.put(key, map);

synch.put(key, \"\");

}

public void destroyNetToken(String key)

{

inputs.remove(key);

synch.remove(key);

}

}"

**end**

File**.**open**(**"**#{**genPath**}**\\**#{**transName**}**Bean.java"**,** 'w'**)** **do** **|**f**|**

f**.**puts transBean

**end**

**end**

manager **=** elementNets**[**en**];**

elementNodes **=** manager**.**xpath**(**'net/places/@name'**).**to\_a **+** manager**.**xpath**(**'net/transitions/@name'**).**to\_a

variablesBlock **=** ""

createBlock **=** ""

destroyBlock **=** ""

**(**0**..**elementNodes**.**count **-** 1**).**each **do** **|**i**|**

variablesBlock **<<** "@EJB\n**#{**elementNodes**[**i**]}**Bean **#{**elementNodes**[**i**]}**;\n"

createBlock **<<** "**#{**elementNodes**[**i**]}**.createNetToken(token);\n"

destroyBlock **<<** "**#{**elementNodes**[**i**]}**.destroyNetToken(id);\n"

**end**

markupsBlock **=** ""

markups **=** manager**.**xpath**(**'elementNetMarkeds'**)**

**(**0**..**markups**.**count **-** 1**).**each **do** **|**i**|**

markupsBlock **<<** "if(markup.equals(\"**#{**markups**[**i**][**'id'**]}**\"))

{\n"

placeIds **=** markups**.**xpath**(**'marking/map/@place'**)**

**(**0**..**placeIds**.**count **-** 1**).**each **do** **|**j**|**

placeName **=** manager**.**xpath**(**"net/places[@id='**#{**placeIds**[**j**].**to\_s**.**delete**(**'#'**)}**']/@name"**).**to\_s

markupsBlock **<<** "**#{**placeName**}**.createToken(token);\n"

**end**

markupsBlock **<<** "}"

**end**

managerBean **=** "package **#{**package**}**;

import java.util.ArrayList;

import java.util.List;

import javax.annotation.PostConstruct;

import javax.ejb.EJB;

import javax.ejb.Singleton;

import javax.ejb.Startup;

@Startup

@Singleton

public class **#{**manager**[**'name'**]}**ManagerBean {

**#{**variablesBlock**}**

List<String> list = new ArrayList<String>();

String type = \"net\";

String name = \"**#{**manager**[**'name'**]}**\";

Integer idCounter = 0;

Integer activeNets;

@PostConstruct

void init() {

activeNets = 0;

}

public String createToken(String markup) {

idCounter++;

String token = (type + \"\_\" + name + \"\_\" + idCounter);

**#{**createBlock**}**

**#{**markupsBlock**}**

activeNets++;

return token;

}

public void removeToken(String id) {

activeNets--;

**#{**destroyBlock**}**

}

public Integer getActiveNets()

{

return activeNets;

}

}

"

File**.**open**(**"**#{**genPath**}**\\**#{**manager**[**'name'**]}**ManagerBean.java"**,** 'w'**)** **do** **|**f**|**

f**.**puts managerBean

**end**

**end**

**end**

**def** **compile(**jdkPath**,** glassfishPath**,** filepath**)**

filePath **=** filepath

fileDir **=** filePath**.**slice**(**0**..**filePath**.**rindex**(**'\\'**))**

f **=** File**.**open**(**filePath**)**

doc **=** Nokogiri**::**XML**(**f**)**

f**.**close

doc **=** doc**.**xpath**(**'npnets:PetriNetNestedMarked'**)**

systemNet **=** doc**.**xpath**(**'child::net'**)[**0**]**

package **=** systemNet**.**xpath**(**'netSystem'**)[**0**][**'name'**].**gsub**(**' '**,** '\_'**)**

genPath **=** fileDir **+** "output\\classes"

**if** **(!**File**.**directory?**(**genPath**))**

FileUtils**.**mkdir\_p genPath

**end**

Dir**.**chdir**(**"**#{**fileDir**}**output/java"**)** **do**

system "\"**#{**jdkPath**}**\\bin\\javac\" -cp \"**#{**glassfishPath**}**\\glassfish\\lib\\javaee.jar\" -d \"**#{**genPath**}**\" **#{**package**}**/\*.java"

**end**

**end**

**def** **pack(**jdkPath**,** filepath**)**

filePath **=** filepath

fileDir **=** filePath**.**slice**(**0**..**filePath**.**rindex**(**'\\'**))**

f **=** File**.**open**(**filePath**)**

doc **=** Nokogiri**::**XML**(**f**)**

f**.**close

doc **=** doc**.**xpath**(**'npnets:PetriNetNestedMarked'**)**

systemNet **=** doc**.**xpath**(**'child::net'**)[**0**]**

package **=** systemNet**.**xpath**(**'netSystem'**)[**0**][**'name'**].**gsub**(**' '**,** '\_'**)**

genPath **=** fileDir **+** "output"

**if** **(!**File**.**directory?**(**genPath**))**

FileUtils**.**mkdir\_p genPath

**end**

Dir**.**chdir**(**"**#{**fileDir**}**output/classes"**)** **do**

system "\"**#{**jdkPath**}**\\bin\\jar\" -cvf \"**#{**genPath**}**\\**#{**package**}**.jar\" **#{**package**}**"

**end**

**end**

**def** **deploy(**glassfishPath**,** filepath**)**

filePath **=** filepath

fileDir **=** filePath**.**slice**(**0**..**filePath**.**rindex**(**'\\'**))**

f **=** File**.**open**(**filePath**)**

doc **=** Nokogiri**::**XML**(**f**)**

f**.**close

doc **=** doc**.**xpath**(**'npnets:PetriNetNestedMarked'**)**

systemNet **=** doc**.**xpath**(**'child::net'**)[**0**]**

package **=** systemNet**.**xpath**(**'netSystem'**)[**0**][**'name'**].**gsub**(**' '**,** '\_'**)**

jarFile **=** fileDir **+** "output\\**#{**package**}**.jar"

FileUtils**.**cp**(**jarFile**,** glassfishPath **+** '\glassfish\domains\domain1\autodeploy'**)**

**end**

**if** **(ARGV[**0**]** **==** 'generate'**)**

generate**(ARGV[**1**].**to\_s**)**

**elsif** **(ARGV[**0**]** **==** 'compile'**)**

compile**(ARGV[**1**].**to\_s**,** **ARGV[**2**].**to\_s**,** **ARGV[**3**].**to\_s**)**

**elsif** **(ARGV[**0**]** **==** 'pack'**)**

pack**(ARGV[**1**].**to\_s**,** **ARGV[**2**].**to\_s**)**

**elsif** **(ARGV[**0**]** **==** 'deploy'**)**

deploy**(ARGV[**1**].**to\_s**,** **ARGV[**2**].**to\_s**)**

**end**

## 1.2. EJBGeneratorGUIApp.java

/\*

\* EJBGeneratorGUIApp.java

\*/

package ejbgeneratorgui**;**

**import** org**.**jdesktop**.**application**.**Application**;**

**import** org**.**jdesktop**.**application**.**SingleFrameApplication**;**

/\*\*

\* The main class of the application.

\*/

public class EJBGeneratorGUIApp **extends** SingleFrameApplication **{**

/\*\*

\* At startup create and show the main frame of the application.

\*/

@Override protected void startup**()** **{**

show**(new** EJBGeneratorGUIView**(this));**

**}**

/\*\*

\* This method is to initialize the specified window by injecting resources.

\* Windows shown in our application come fully initialized from the GUI

\* builder, so this additional configuration is not needed.

\*/

@Override protected void configureWindow**(**java**.**awt**.**Window root**)** **{**

**}**

/\*\*

\* A convenient static getter for the application instance.

\* **@return** the instance of EJBGeneratorGUIApp

\*/

public static EJBGeneratorGUIApp getApplication**()** **{**

**return** Application**.**getInstance**(**EJBGeneratorGUIApp**.**class**);**

**}**

/\*\*

\* Main method launching the application.

\*/

public static void main**(**String**[]** args**)** **{**

launch**(**EJBGeneratorGUIApp**.**class**,** args**);**

**}**

**}**

## 1.3. EJBGeneratorGUIView.java

/\*

\* EJBGeneratorGUIView.java

\*/

package ejbgeneratorgui**;**

**import** java**.**io**.**IOException**;**

**import** java**.**util**.**logging**.**Level**;**

**import** java**.**util**.**logging**.**Logger**;**

**import** org**.**jdesktop**.**application**.**SingleFrameApplication**;**

**import** org**.**jdesktop**.**application**.**FrameView**;**

**import** java**.**io**.**BufferedReader**;**

**import** java**.**io**.**InputStreamReader**;**

**import** javax**.**swing**.**JFileChooser**;**

**import** javax**.**swing**.**JFrame**;**

/\*\*

\* The application's main frame.

\*/

public class EJBGeneratorGUIView **extends** FrameView **{**

JFileChooser fc **=** **new** JFileChooser**();**

public EJBGeneratorGUIView**(**SingleFrameApplication app**)** **{**

**super(**app**);**

getFrame**().**setResizable**(false);**

//getFrame().pack();

initComponents**();**

**}**

/\*\* This method is called from within the constructor to

\* initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is

\* always regenerated by the Form Editor.

\*/

@SuppressWarnings**(**"unchecked"**)**

// <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents

private void initComponents**()** **{**

mainPanel **=** **new** javax**.**swing**.**JPanel**();**

jLabel1 **=** **new** javax**.**swing**.**JLabel**();**

inputTextField **=** **new** javax**.**swing**.**JTextField**();**

openButton **=** **new** javax**.**swing**.**JButton**();**

generateButton **=** **new** javax**.**swing**.**JButton**();**

jdkTextField **=** **new** javax**.**swing**.**JTextField**();**

jLabel2 **=** **new** javax**.**swing**.**JLabel**();**

jdkButton **=** **new** javax**.**swing**.**JButton**();**

jLabel3 **=** **new** javax**.**swing**.**JLabel**();**

glassFishTextField **=** **new** javax**.**swing**.**JTextField**();**

glassFishButton **=** **new** javax**.**swing**.**JButton**();**

compileButton **=** **new** javax**.**swing**.**JButton**();**

packButton **=** **new** javax**.**swing**.**JButton**();**

deployButton **=** **new** javax**.**swing**.**JButton**();**

jLabel4 **=** **new** javax**.**swing**.**JLabel**();**

satusLabel **=** **new** javax**.**swing**.**JLabel**();**

mainPanel**.**setName**(**"mainPanel"**);** // NOI18N

org**.**jdesktop**.**application**.**ResourceMap resourceMap **=** org**.**jdesktop**.**application**.**Application**.**getInstance**(**ejbgeneratorgui**.**EJBGeneratorGUIApp**.**class**).**getContext**().**getResourceMap**(**EJBGeneratorGUIView**.**class**);**

jLabel1**.**setText**(**resourceMap**.**getString**(**"jLabel1.text"**));** // NOI18N

jLabel1**.**setName**(**"jLabel1"**);** // NOI18N

inputTextField**.**setText**(**resourceMap**.**getString**(**"inputTextField.text"**));** // NOI18N

inputTextField**.**setName**(**"inputTextField"**);** // NOI18N

openButton**.**setText**(**resourceMap**.**getString**(**"openButton.text"**));** // NOI18N

openButton**.**setName**(**"openButton"**);** // NOI18N

openButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

openButtonMouseClicked**(**evt**);**

**}**

**});**

generateButton**.**setText**(**resourceMap**.**getString**(**"generateButton.text"**));** // NOI18N

generateButton**.**setName**(**"generateButton"**);** // NOI18N

generateButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

generateButtonMouseClicked**(**evt**);**

**}**

**});**

jdkTextField**.**setText**(**resourceMap**.**getString**(**"jdkTextField.text"**));** // NOI18N

jdkTextField**.**setName**(**"jdkTextField"**);** // NOI18N

jLabel2**.**setText**(**resourceMap**.**getString**(**"jLabel2.text"**));** // NOI18N

jLabel2**.**setName**(**"jLabel2"**);** // NOI18N

jdkButton**.**setText**(**resourceMap**.**getString**(**"jdkButton.text"**));** // NOI18N

jdkButton**.**setName**(**"jdkButton"**);** // NOI18N

jdkButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

jdkButtonMouseClicked**(**evt**);**

**}**

**});**

jLabel3**.**setText**(**resourceMap**.**getString**(**"jLabel3.text"**));** // NOI18N

jLabel3**.**setName**(**"jLabel3"**);** // NOI18N

glassFishTextField**.**setText**(**resourceMap**.**getString**(**"glassFishTextField.text"**));** // NOI18N

glassFishTextField**.**setName**(**"glassFishTextField"**);** // NOI18N

glassFishButton**.**setText**(**resourceMap**.**getString**(**"glassFishButton.text"**));** // NOI18N

glassFishButton**.**setName**(**"glassFishButton"**);** // NOI18N

glassFishButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

glassFishButtonMouseClicked**(**evt**);**

**}**

**});**

compileButton**.**setText**(**resourceMap**.**getString**(**"compileButton.text"**));** // NOI18N

compileButton**.**setName**(**"compileButton"**);** // NOI18N

compileButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

compileButtonMouseClicked**(**evt**);**

**}**

**});**

packButton**.**setText**(**resourceMap**.**getString**(**"packButton.text"**));** // NOI18N

packButton**.**setName**(**"packButton"**);** // NOI18N

packButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

packButtonMouseClicked**(**evt**);**

**}**

**});**

deployButton**.**setText**(**resourceMap**.**getString**(**"deployButton.text"**));** // NOI18N

deployButton**.**setName**(**"deployButton"**);** // NOI18N

deployButton**.**addMouseListener**(new** java**.**awt**.**event**.**MouseAdapter**()** **{**

public void mouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**

deployButtonMouseClicked**(**evt**);**

**}**

**});**

jLabel4**.**setText**(**resourceMap**.**getString**(**"jLabel4.text"**));** // NOI18N

jLabel4**.**setName**(**"jLabel4"**);** // NOI18N

satusLabel**.**setText**(**resourceMap**.**getString**(**"satusLabel.text"**));** // NOI18N

satusLabel**.**setToolTipText**(**resourceMap**.**getString**(**"satusLabel.toolTipText"**));** // NOI18N

satusLabel**.**setName**(**"satusLabel"**);** // NOI18N

javax**.**swing**.**GroupLayout mainPanelLayout **=** **new** javax**.**swing**.**GroupLayout**(**mainPanel**);**

mainPanel**.**setLayout**(**mainPanelLayout**);**

mainPanelLayout**.**setHorizontalGroup**(**

mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**mainPanelLayout**.**createSequentialGroup**()**

**.**addContainerGap**()**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**)**

**.**addComponent**(**jLabel4**)**

**.**addComponent**(**jLabel2**)**

**.**addComponent**(**jLabel1**)**

**.**addComponent**(**jLabel3**))**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**mainPanelLayout**.**createSequentialGroup**()**

**.**addGap**(**9**,** 9**,** 9**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** **false)**

**.**addComponent**(**inputTextField**,** javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 290**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**jdkTextField**,** javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addComponent**(**glassFishTextField**,** javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** 340**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**))**

**.**addGroup**(**mainPanelLayout**.**createSequentialGroup**()**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**,** **false)**

**.**addComponent**(**generateButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**packButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 129**,** Short**.**MAX\_VALUE**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**,** 77**,** Short**.**MAX\_VALUE**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** **false)**

**.**addComponent**(**deployButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**compileButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 138**,** Short**.**MAX\_VALUE**))))**

**.**addGap**(**10**,** 10**,** 10**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**,** **false)**

**.**addComponent**(**jdkButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**glassFishButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** Short**.**MAX\_VALUE**)**

**.**addComponent**(**openButton**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** 123**,** Short**.**MAX\_VALUE**)))**

**.**addGroup**(**mainPanelLayout**.**createSequentialGroup**()**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addComponent**(**satusLabel**)))**

**.**addContainerGap**())**

**);**

mainPanelLayout**.**setVerticalGroup**(**

mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**LEADING**)**

**.**addGroup**(**mainPanelLayout**.**createSequentialGroup**()**

**.**addContainerGap**()**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**TRAILING**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**jdkTextField**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**jdkButton**))**

**.**addComponent**(**jLabel2**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**UNRELATED**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**jLabel3**)**

**.**addComponent**(**glassFishTextField**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**glassFishButton**))**

**.**addGap**(**12**,** 12**,** 12**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**jLabel1**)**

**.**addComponent**(**inputTextField**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**,** javax**.**swing**.**GroupLayout**.**DEFAULT\_SIZE**,** javax**.**swing**.**GroupLayout**.**PREFERRED\_SIZE**)**

**.**addComponent**(**openButton**))**

**.**addPreferredGap**(**javax**.**swing**.**LayoutStyle**.**ComponentPlacement**.**RELATED**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**jLabel4**)**

**.**addComponent**(**satusLabel**))**

**.**addGap**(**7**,** 7**,** 7**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**generateButton**)**

**.**addComponent**(**compileButton**))**

**.**addGap**(**18**,** 18**,** 18**)**

**.**addGroup**(**mainPanelLayout**.**createParallelGroup**(**javax**.**swing**.**GroupLayout**.**Alignment**.**BASELINE**)**

**.**addComponent**(**packButton**)**

**.**addComponent**(**deployButton**))**

**.**addContainerGap**())**

**);**

openButton**.**getAccessibleContext**().**setAccessibleName**(**resourceMap**.**getString**(**"jButton1.AccessibleContext.accessibleName"**));** // NOI18N

setComponent**(**mainPanel**);**

**}**// </editor-fold>//GEN-END:initComponents

private void openButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_openButtonMouseClicked

fc**.**setFileSelectionMode**(**JFileChooser**.**FILES\_ONLY**);**

int returnVal **=** fc**.**showOpenDialog**(null);**

**if** **(**returnVal **==** JFileChooser**.**APPROVE\_OPTION**)** **{**

inputTextField**.**setText**(**fc**.**getSelectedFile**().**getAbsolutePath**());**

//This is where a real application would open the file.

**}** **else** **{**

**}**

**}**//GEN-LAST:event\_openButtonMouseClicked

private void generateButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_generateButtonMouseClicked

**try** **{**

ProcessBuilder builder **=** **new** ProcessBuilder**(**

"cmd.exe"**,** "/c"**,** "ruby EJBGenerator.rb generate \"" **+** inputTextField**.**getText**()** **+** "\""**);**

builder**.**redirectErrorStream**(true);**

Process p **=** builder**.**start**();**

BufferedReader r **=** **new** BufferedReader**(new** InputStreamReader**(**p**.**getInputStream**()));**

String line**;**

**while** **(true)** **{**

line **=** r**.**readLine**();**

**if** **(**line **==** **null)** **{** **break;** **}**

System**.**out**.**println**(**line**);**

**}**

**}** **catch** **(**IOException ex**)** **{**

Logger**.**getLogger**(**EJBGeneratorGUIView**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

**}**

**}**//GEN-LAST:event\_generateButtonMouseClicked

private void compileButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_compileButtonMouseClicked

**try** **{**

ProcessBuilder builder **=** **new** ProcessBuilder**(**

"cmd.exe"**,** "/c"**,** "ruby EJBGenerator.rb compile \"" **+** jdkTextField**.**getText**()** **+** "\" \"" **+** glassFishTextField**.**getText**()** **+** "\" \"" **+** inputTextField**.**getText**()** **+** "\""**);**

builder**.**redirectErrorStream**(true);**

Process p **=** builder**.**start**();**

BufferedReader r **=** **new** BufferedReader**(new** InputStreamReader**(**p**.**getInputStream**()));**

String line**;**

**while** **(true)** **{**

line **=** r**.**readLine**();**

**if** **(**line **==** **null)** **{** **break;** **}**

System**.**out**.**println**(**line**);**

**}**

**}** **catch** **(**IOException ex**)** **{**

Logger**.**getLogger**(**EJBGeneratorGUIView**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

**}**

**}**//GEN-LAST:event\_compileButtonMouseClicked

private void packButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_packButtonMouseClicked

**try** **{**

ProcessBuilder builder **=** **new** ProcessBuilder**(**

"cmd.exe"**,** "/c"**,** "ruby EJBGenerator.rb pack \"" **+** jdkTextField**.**getText**()** **+** "\" \"" **+** inputTextField**.**getText**()** **+** "\""**);**

builder**.**redirectErrorStream**(true);**

Process p **=** builder**.**start**();**

BufferedReader r **=** **new** BufferedReader**(new** InputStreamReader**(**p**.**getInputStream**()));**

String line**;**

**while** **(true)** **{**

line **=** r**.**readLine**();**

**if** **(**line **==** **null)** **{** **break;** **}**

System**.**out**.**println**(**line**);**

**}**

**}** **catch** **(**IOException ex**)** **{**

Logger**.**getLogger**(**EJBGeneratorGUIView**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

**}**

**}**//GEN-LAST:event\_packButtonMouseClicked

private void deployButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_deployButtonMouseClicked

**try** **{**

ProcessBuilder builder **=** **new** ProcessBuilder**(**

"cmd.exe"**,** "/c"**,** "ruby EJBGenerator.rb deploy \"" **+** glassFishTextField**.**getText**()** **+** "\" \"" **+** inputTextField**.**getText**()** **+** "\""**);**

builder**.**redirectErrorStream**(true);**

Process p **=** builder**.**start**();**

BufferedReader r **=** **new** BufferedReader**(new** InputStreamReader**(**p**.**getInputStream**()));**

String line**;**

**while** **(true)** **{**

line **=** r**.**readLine**();**

**if** **(**line **==** **null)** **{** **break;** **}**

System**.**out**.**println**(**line**);**

**}**

**}** **catch** **(**IOException ex**)** **{**

Logger**.**getLogger**(**EJBGeneratorGUIView**.**class**.**getName**()).**log**(**Level**.**SEVERE**,** **null,** ex**);**

**}**

**}**//GEN-LAST:event\_deployButtonMouseClicked

private void jdkButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_jdkButtonMouseClicked

fc**.**setFileSelectionMode**(**JFileChooser**.**DIRECTORIES\_ONLY**);**

int returnVal **=** fc**.**showOpenDialog**(null);**

**if** **(**returnVal **==** JFileChooser**.**APPROVE\_OPTION**)** **{**

jdkTextField**.**setText**(**fc**.**getSelectedFile**().**getAbsolutePath**());**

//This is where a real application would open the file.

**}** **else** **{**

**}**

**}**//GEN-LAST:event\_jdkButtonMouseClicked

private void glassFishButtonMouseClicked**(**java**.**awt**.**event**.**MouseEvent evt**)** **{**//GEN-FIRST:event\_glassFishButtonMouseClicked

fc**.**setFileSelectionMode**(**JFileChooser**.**DIRECTORIES\_ONLY**);**

int returnVal **=** fc**.**showOpenDialog**(null);**

**if** **(**returnVal **==** JFileChooser**.**APPROVE\_OPTION**)** **{**

glassFishTextField**.**setText**(**fc**.**getSelectedFile**().**getAbsolutePath**());**

//This is where a real application would open the file.

**}** **else** **{**

**}**

**}**//GEN-LAST:event\_glassFishButtonMouseClicked

// Variables declaration - do not modify//GEN-BEGIN:variables

private javax**.**swing**.**JButton compileButton**;**

private javax**.**swing**.**JButton deployButton**;**

private javax**.**swing**.**JButton generateButton**;**

private javax**.**swing**.**JButton glassFishButton**;**

private javax**.**swing**.**JTextField glassFishTextField**;**

private javax**.**swing**.**JTextField inputTextField**;**

private javax**.**swing**.**JLabel jLabel1**;**

private javax**.**swing**.**JLabel jLabel2**;**

private javax**.**swing**.**JLabel jLabel3**;**

private javax**.**swing**.**JLabel jLabel4**;**

private javax**.**swing**.**JButton jdkButton**;**

private javax**.**swing**.**JTextField jdkTextField**;**

private javax**.**swing**.**JPanel mainPanel**;**

private javax**.**swing**.**JButton openButton**;**

private javax**.**swing**.**JButton packButton**;**

private javax**.**swing**.**JLabel satusLabel**;**

// End of variables declaration//GEN-END:variables

**}**

# Лист регистрации изменений.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |