Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.41
DOI: https://doi.org/10.24867/26BE29Djordjevic

ИМПЛЕМЕНТАЦИЈА ПОДРШКЕ ЗА ВИШЕСТРУКО ИНСТАНЦИРАЊЕ АКТИВНОСТИ У ПРОЦЕСНОМ ОКРУЖЕЊУ NEW WAVE

IMPLEMENTING SUPPORT FOR MULTIINSTANCE ACTIVITIES IN NEW WAVE PROCESS ENGINE

Никола Ђорђевић, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Кратак садржај – *У раду је представљена имплементација модула који процесном оркужењу New-Wave Engine ломогућава рад са вишеструко инстанцираним задацима (енгл. multiinstance task). Овај модул је написан у програмском језику Фаро.*

Кључне речи: *Пословни процес, Workflow engine, Фаро, NewWave*

Abstract – *This document presents the implementation of a module that will add support for multiinstance tasks to the NewWave Engine process execution environment. This module is written in the programming language Pharo.*

Keywords: *Business process, Workflow engine, Pharo, NewWave*

1. УВОД

Пословни процеси су од кључног значаја за сваку компанију. Пословни процес се може дефинисати као активност или скуп активности којима се постиже пословни циљ компаније[1]. Тржиште које је све конкурентније и динамичније повлачи потребу организација да унапреде своје пословне процесе. Управљање пословним процесима (engl. *Business Process Management - BPM*) је дисциплина која за примарни циљ има да унапреди пословне процесе, да се смањи комплексност, а повећа ефикасност. *Workflow engine* конвертује традиционални мануелни *workflow* од ИТ-вођених задатака до процеса којим управља човек или софтвер и врше рутирање и усмеравање информационих путања, одговорности и контролишу комуникационе канале при сарадњи различитих процеса како би се што ефикасније искористили ресурси. Већина користи БПМН процесни модел за дијаграм који усмерава *workflow*. *Workflow engine* је софтверски сервис који пружа окружење за праводобно извршавање задатака за процесне инстанце [2], [3], у складу са ограничењима и редоследом дефинисаним у процесном моделу.

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Мирослав Зарић, ванр. проф.

Вишеструке активности (активности са више инстанци - енгл. *Multi instance activities*) - поред спецификације задатка који треба да се изврши, као улазни параметар имају и колекцију улазних објеката.

За сваки објекат дате колекције креира се једна инстанца задатка. Уколико се ради о корисничким задацима, сваку појединачну инстанцу типично извршава други корисник. Могу бити секвенцијални или паралелни. **Секвенцијалне** вишеструке активности су оне код којих се инстанце извршавају једна за другом – секвенцијално. Други задатак по реду неће почети са својим извршавањем све док се први не заврши, трећи ће чекати на други, итд. Обележавају се следећим маркером:

Слика 1. Графичка репрезентација маркера за секвенцијалне вишеструке активности

Код паралелних вишеструких активности сви инстанцирани таскови ће почети да се извршавају у истом тренутку – паралелно. Целокупна активност ће бити завршена тек онда када се оконча свака инстанца у оквиру посматране паралелне вишеструке активности. Обележавају се паралелним маркером:

Слика 2. Графичка репрезентација маркера за паралелне вишеструке активности

Фаро (енгл. *Pharo*) је једноставан, објектно оријентисан програмски језик отвореног кода (енгл. *open-source*). *Open-source* значи да је изворни код доступан свима, и за преглед, и за модификацију кода. Његова виртуална машина је у потпуности написана у Фару, што значајно олакшава симулацију, проналажење грешака, анализу и измене из самог Фара.

PharoLauncher је апликација која омогућава преузимање и управљање фаро сликама (енгл. *Pharo image*). Кориснику даје приступ сликама који садрже специфичне библиотеке што значајно олакшава

њихову употребу и елиминише потребу да се ручно инсталирају и конфигуришу [4].

NewWave је окружење отвореног кода, за управљање пословним процесима, написано у програмском језику Фаро. У питању је лако проширив систем, са циљем да омогући једноставну спецификацију и имплементацију пословних процеса. Иако је у фази развоја, извршно окружење (*NewWave engine*) подржава основне обрасце контроле тока, као што су секвенцијални, паралелно гранање и спајање са синхронизацијом, екскулзивна гранања и спајања, различите типове догађаја, итд [5].

WaveEngine је главна компонента *NewWave* радног окружења, и она организује и контролише извршавање радних токова. Садржи један главни *WaveExecutor*, који је одговоран за само извршавање радног тока (енгл. *workflow*) и има задатак да пружи погодно окружење да би се извршио елемент који је дошао на ред. *Flowhandler* је компонента која на основу модела радног тока одређује који елемент је следећи по реду за извршавање.

2. ИМПЛЕМЕНТАЦИЈА СИСТЕМА

2.1. Бекенд апликације

Првобитно је потребно креирати класе за вишеструке задатке:

Узимајући у обзир да су задаци у суштини активности, одлучено је да ће класа под називом *NWMultiInstanceTask* наслеђивати *NWBaseActivity* класу. Ова класа садржи две варијабле на нивоу инстанци:

- task
- users (Колекција *Username*-ова)

Наредни корак је био да се специфицира шта и на који начин треба да се изврши када *FlowHandler* наиђе на *NWMultiInstanceUserTask*. Ово је урађено у класи – *NWMultiInstanceUserTaskBehavior*. Примарна метода у овој класи је *performExecution*. Она поседује два обавезна улазна параметра:

- *elemToExecute* – Ова променљива је инстанца објекта *NWMultiInstanceUserTask* и у себи садржи неопходне податке

(*task, users*) за креирање инстанци паралелних таскова.

- *waveExecutor* – Садржи инстанцу *WaveExecutor*-а који је задужен за извршавање ове активности.

На слици 3. је приказан код и начин имплементације ове методе.

Као што се може видети, на самом почетку је неопходно креирати чвор за паралелно спајање, инстанцирањем класе *NWParallelJoin*. Овај чвор (елемент) има задатак да спречи наставак извршавања пословног процеса, све док се не заврше сви таскови који имају *outGoingEdge* према њему.

```
performExecution: elemToExecute executor: waveExecutor
   | pluginData userIds join endMulti |
   pluginData := waveExecutor processHandler engine pluginData.


   join := NWParallelJoin new.
   join description: 'MultiInstance Join'.

   userIds := elemToExecute users.
   userIds do: [ :each |
      | tempUserTask |
      tempUserTask := (self generateDefaultUserTask:
      each pluginData: pluginData).
      tempUserTask addOutgoingEdge: join.
      waveExecutor processHandler addSubExecutor: (NWSequence source:
      nil target: tempUserTask) ].

   endMulti := NWEndMultiInstanceEvent new.
   endMulti description: 'End MultiInstance Task'.

   join addOutgoingEdge: endMulti.

   waveExecutor processHandler engineAnnouncer
   when: EndMultiInstanceAnnouncer
      do: [ :ann |
         waveExecutor flowHandler node:
         elemToExecute outgoingFlows first targetRef.
         waveExecutor tryToExecuteNext: elemToExecute.
         ].
```

Слика 3. Метода *performExecution* класе *NWMultiInstanceUserTaskBehavior*

Затим се преузима колекција корисничких имена (енгл. *username*). За сваког корисника из дате колекције креирамо један предефинисани UserTask користећи методу *generateDefaultUserTask*, којој прослеђујемо *username* корисника као и променљиву *pluginData* која садржи податке о свим корисницима у систему. Ова метода је приказана на слици 4.

```
generateDefaultUserTask: userId pluginData: pluginData
   | user task do adresa |
   user := (pluginData at: #NWUserManagement) findUserByName:
   userId asString.
   Transcript show: user userId.

   adresa := Adresa example3.
   do := NWDataObject new.
   do valuedDataObject:
   (NWClassDescriptionGenerator dataObject: adresa) createClassDescription.
   do name: ('Adress data object example ', userId asString).

   task := NWUserTask new.
   task id: userId.
   task description: 'Entering the address'.
   task name: 'Address task'.
   task value: 'Task1 value'.
   "t1 group: (management findGroup: '1')."
   task user: user.
   task addDataOutputAssociation: do.
   "wave waveAnnouncer announce: JoinEventAnnouncer new."
      ^ task
```

Слика 4. Метода *generateDefaultUserTask* класе *NWMultiInstanceUserTaskBehavior*

Када се метода покрене, користећи *pluginData* приступамо *NWUserManagement*. Прослеђивањем корисничког имена у *findUserByName* методу, довлачимо све податке о датом кориснику. У наредном кораку се креира *NWDataObject* који садржи информације о форми и подацима који се од корисника очекују да унесе у изгенерисану форму.

Даље је потребно инстанцирати *NWUserTask* и за њега унети тражене податке као што су нпр.:

- id - Идентификациони број таска
- description - Кратак опис таска
- name – Назив таска
- value – Вредност таска
- *user* – Корисник од којег се очекује да изврши задатак
- *dataObject* – Подаци о форми коју корисник попуњава када извршава задатак

Наредни корак у *performExecution* методи је да се сваки нови *UserTask* повеже са претходно креираним паралелним спајањем. У овом тренутку је завршено све што је неопходно за један UserTask и остаје само да се за њега дефинише подпроцес и додели објекат типа *NWSubExecutor* који ће бити задужен за његово извршавање.

Након што петља прође кроз све кориснике, инстанцира таскове и изгенерише подпроцесе потребно је креирати помоћни евент *NWEndMultiInstanceEvent*. Непосредно после, се чвор спајања повезује на почетак овог евента. Он има једноставан задатак, треба да позове *EndMultiInstanceAnnouncer* који ће обавестити вишеструки задатак да су сви корисници успешно завршили своје задатке и да процес може да настави са даљим извршавањем. Вратимо се на *NWMultiInstanceUserTask* и његову *performExecution* методу.

```
waveExecutor processHandler engineAnnouncer
when: EndMultiInstanceAnnouncer
   do: [ :ann |
      waveExecutor flowHandler node:
      elemToExecute outgoingFlows first targetRef.
      waveExecutor tryToExecuteNext: elemToExecute.
      ].
```

Слика 5. Део класе *NWMultiInstanceUserTaskBehavior* где се чека на *announcement*

У овом исечку кода је приказан *listener* који чека на такозвани *announcment* од стране *EndMultiInstanceAnnouncer*-а. Када добије обавештење да је примио очекивани *announcment,* извршиће део кода који се налази испод у *do* блоку. С обзиром на то да је вишеструки задатак у принципу завршен, обавештавамо *FlowHandler*да су подпроцеси завршени и да је на ред извршавања дошао излазни ток од посматраног мулти-инстанц таска. Позива се затим *tryToExecuteNext,* и ту је целокупна активност званично завршена и прелази се на следећу.

2.2. Кориснички интерфејс - фронтенд апликација

Након што улоговани корисник изабере да жели да креира нови вишеструки задатак, изгенерисаће се једноставан *UI*(кориснички интерфејс) који ће му то омогућити. Са сервера је неопходно добавити податке о свим корисницима и ово је могуће учинити користећи методу *getAllUsers* класе *FUGetUserServer*. Податке прихватамо са серверу у форми *JSON* стринга. Наведени код са слике нам омогућава да JSON објекте ишчитамо у форми тока (енгл. *stream).* Добијени ток се потом мапира на низ објеката типа *NWUser*.Метода враћа податке у облику уређене колекције (енгл. ordered collection).

```
getAllUsers
   | entity obj |
   entity := (ZnEasy get:
   'http://localhost:8081/allusers/') entity string.
   entity := (NeoJSONReader fromString: entity).
   obj:= (NeoJSONReader
      on: entity readStream)
      mapInstVarsFor: NWUser;
      for: #ArrayOfNWUsers
         customDo: [
         :mapping | mapping listOfElementSchema: NWUser ];
      nextAs: #ArrayOfNWUsers.

   "obj := (NeoJSONReader fromString: entity)."
   ^ obj asOrderedCollection.
```

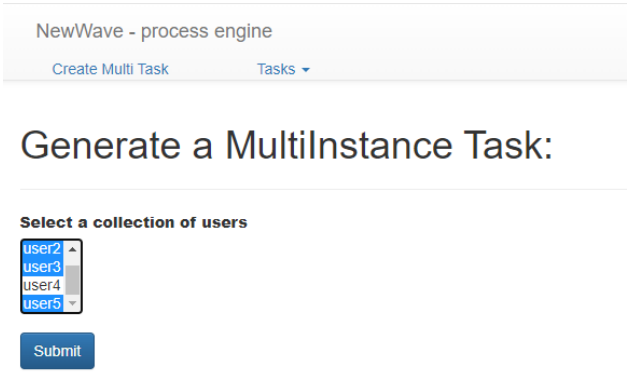
Слика 6. Метода getAllUsers класе *FUGetUserServer*

Након што се добаве неопходни подаци, рендерује се једноставна форма са MultiSelect компонентом. У њој се налази листа имена корисника којима је могуће доделити вишеструки задатак.
Истовремено је могуће изабрати више корисника. Након што што се корисници изаберу, кликне се на дугме *Submit* и на тај начин активира *callback.* Позива се метода *commitMultiInstanceTask* klase *FUCommitMultiInstanceTask,* где се као улазни параметар прослеђују корисничка имена селектованих корисника.

Захтев са стране корисничког интерфејса апликације стиже на сервер, где се активира метода addNewMultiInstanceTask. Улазни подаци се враћају назад у форми колекције. Првобитно се креира NWStartEvent који означава почетак рада процеса. Затим инстанцирамо NWMultiInstanceUserTask и додељујемо му колекцију корисника који су пристигли са корисничког интерфејса апликације.

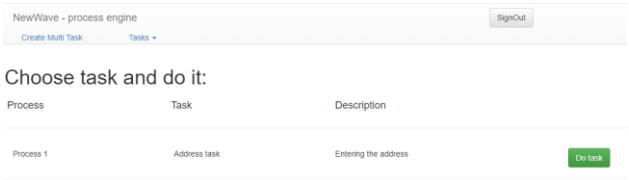
3. ПРИКАЗ РАДА АПЛИКАЦИЈЕ

Након успешног логовања, корисник ће се усмерити ка главној страници са табеларним приказом свих прихваћених, а неизвршених таскова улогованог корисника. У заглављу странице може се уочити дугме 'Create Multi Task', левим кликом на њега прелазимо на страницу за креирање вишеструког задатка.
У *dropdown*-у су приказана корисничка имена пет корисника која су у сврхе демо-а генерисана приликом покретања апликације.



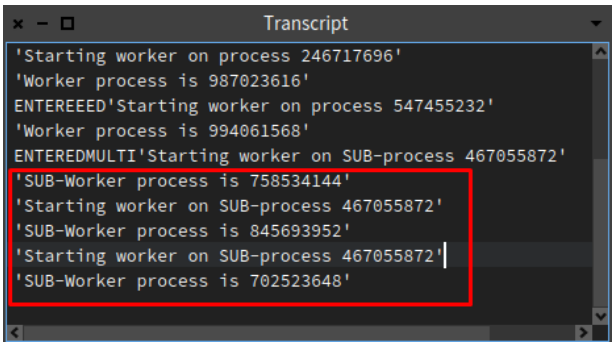
Слика 7. Страница на којој се може креирати нови вишеструког задатка за изабране кориснике.

Једноставном селекцијом једног или више понуђених корисника из листе и кликом на дугме 'Submit' послаћемо захтев серверу да генерише задатак за сваког од изабраних корисника. За овај пример биће селектовани корисници: *user2, user3* і *user5*. Потом, логовањем као user2 може се установити да ли је за њега креиран одговарајући *UserTask* или није. Може се одмах уочити да задатак јесте креиран за овог корисника, понављањем поступка и за остала два, установљено је да су сви кориснички задаци, за сваког изабраног корисника, успешно инстанцирани.



Слика 8. Почетна страница корисника user2 са новокреираним таском

Такође можемо из транскрипта унутар PharoLauncher-а уочити да се заиста јесу креирала три подпроцеса за *UserTask*-ове у оквиру вишеструког задатка, слика 9.



Слика 9. Транскрипт у PharoLauncher-u сапокренутим подпроцесима

4. ЗАКЉУЧАК

Тема рада била је је имплементација модула за подршку креирања и извршавања вишеструких задатака у оквиру процесног окружења NewWave.

У раду је дат кратак преглед основних концепата софтверских система који пружају подршку управљању пословним процесима. Описани су основни ентитети који дефинишу процесе, и појашњен концепт вишеструких задатака. Код је написан у програмском језику Фаро, а за извршно окружење рада је коришћена PharoLauncher апликација.

Даљи правац развоја апликације може бити имплементација и модула који ће омогућити рад са секвенцијалним вишеструким тасковима. Такође би било интересантно и корисно када би се имплементирао графички приказ тока једног процеса. Самим тим би се кориснику омогућило да у реалном времену има јасну слику тока процеса и свих подпроцеса.

5. ЛИТЕРАТУРА

[1]https://www.techtarget.com/searchcio/definition/business-process [Приступљено 12.08.2022.]

[2] *What Is a Workflow Engine* https://www.ibm.com/cloud/blog/workflow-engine [Приступљено 15.09.2022.]

[3] *Workflow Management Coalition Terminology & Glossary*, Workflow Management Coalition, 2 Crown Walk, Winchester, Hampshire SO23 8BB, United Kingdom, Фебруар 1999

[4] Stéphane Ducasse and Gordana Rakic with Sebastijan Kaplar and Quentin Ducasse, *Pharo 9 by Example*, Октобар 2021

[5] Sebastijan Kaplar, Miroslav Zarić, Gordana Milosavljević, *NewWave Workflow Engine*, Cologne, Germany, Август 2019

Кратка биографија:

Никола Ђорђевић је рођен 10.10.1996. у Београду, Република Србија. Основну школу „Петефи Шандор“ завршио је 2011. Године. Након тога је уписао гимназију „Јован Јовановић Змај“ у Новом Саду, природно-математички смер. Гимназију је завршио 2015. године са одличним успехом, и исте године се уписује на Факултет техничких наука у Новом Саду, одсек Рачунарство и Аутоматика. Основне студије је завршио 2019. Године