Зборник радова Факултета техничких наука, Нови Сад

UDK: 004.9
DOI: https://doi.org/10.24867/25BE11Hrnjak

SECURADSL – НАМЕНСКИ ЈЕЗИК ЗА ПОДРШКУ БРЗОГ УСПОСТАВЉАЊА БЕЗБЕДНОСНИХ АСПЕКАТА У РАДНОМ ОКВИРУ *SPRING*

SECURADSL - A DOMAIN-SPECIFIC LANGUGAGE FOR RAPID CONFIGURATION OF SECURITY ASPECTS IN THE SPRING FRAMEWORK

Јелена Хрњак, *Факултет техничких наука, Нови Сад*

Област – ЕЛЕКТРОТЕХНИЧКО И РАЧУНАРСКО ИНЖЕЊЕРСТВО

Кратак садржај – *У овом раду описан је наменски језик securaDSL за моделовање Spring апликација са безбедносном конфигурацијом подржаном за три безбедносна механизма: основну аутентификацију, JWT и OAuth2.0. Поред наменског језика, развијени су генератори извршивог кода на основу модела креираног помоћу језика securaDSL. Употреба овог наменског језика и генератора олакшава, убрзава и унапређује квалитет рада доменских експерата у пољу безбедносних конфигурација софтвера. За развој мета-модела коришћено је окружење Eclipse Modeling Framework које за мета-моделовање користи језик Ecore. Додатна ограничења описана су помоћу наменског језика Object Constraint Language. За развој текстуалне синтаксе коришћен је радни оквир Xtext, а за развој генератора програмски језик Java и језик Xtend.*

Кључне речи: *наменски језици, развој софтвера вођен моделима, моделовање веб апликација, безбедносна конфигурација*

Abstract – *In this paper, we present securaDSL, a domain-specific language designed for modelling Spring applications with security configuration supporting three security mechanisms: Basic Authentication, JWT and OAuth2.0. Additionally, we present multiple generators that produce executable code based on models created using the securaDSL. The utilization of this domain-specific language and generators simplifies, accelerates, and enhances the quality of work for domain experts in the field of software security configurations. For the development of the meta-model, we used Eclipse Modeling Framework environment, utilizing the Ecore language for meta-modeling. To create the textual syntax, we used the Xtext framework, while we implemented generators using the Java programming language and Xtend language.*

Keywords: *Domain-Specific Languages, Model-Driven Software Development, Web Applications modeling, Security Configuration*

1. УВОД

Корисници веб апликација често нису свесни колико личних података оне прикупљају, обрађују и складиште. Неовлашћен приступ тим подацима могао би да доведе до злоупотребе и нарушавања приватности корисника. Стога, обезбеђивање високог нивоа безбедносни представља важан део развоја веб апликација.

Безбедност у веб апликацијама представља скуп мера и механизама који су примењени како би се корисници, систем и подаци заштитили од различитих видова напада, злоупотреба и крађа. Имплементација жељеног нивоа аутентификације и ауторизације, као два основна концепта у области безбедности, представља основу ефикасне заштите.

Аутентификација представља процес потврђивања идентитета корисника или ентитета који приступа систему. Ауторизација се односни на контролу приступа одређеним ресурсима или функционалностима система.

Java је објектно-оријентисан програмски језик, а платформска независност, једноставност и објектна оријентисаност су особине које чине овај језик једним од најпопуларнијих [1]. Радни оквир *Spring* чини развој серверског дела веб апликација у програмском језику *Java* бржим, једноставнијим и сигурнијим што га чини најпопуларнијим радним оквиром за ову намену [2].

Међутим, обезбеђивање одговарајуће заштите за апликације у радном оквиру *Spring* представља сложен и временски захтеван процес, те је самим тим подложан грешкама. С обзиром да се безбедносни аспекти изнова конфигуришу при почетној имплементацији сваке апликације, овакав посао постаје и репетативан.

Са циљем уклањања наведених недостатака, тежи се оптимизацији и аутоматизацији развоја безбедних веб апликација. Једно од могућих решења представља аутоматско генерисање почетне *Spring* веб апликације са конфигурисаним безбедносним аспектима које се врши на основу параметара које корисник унесе, а који дефинишу основне карактеристике апликације и њених елемената.

За постизање овог циља креиран је наменски језик *Secura Domain-Specific Language* (*securaDSL*) за моделовање *Spring* веб апликација уз генераторе који на основу модела генеришу извршиви код написан у програмском језику *Java* и радног оквира *Spring*. Користећи језик *securaDSL*, експерти у пољу

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Владимир Димитриески, доцент.

безбедносних конфигурација могу брзо и једноставно да дефинишу параметре апликација и конфигуришу различите безбедносне механизме помоћу синтаксе која им је лако читљива.

Имплементација сигурних веб апликација на овај начин постаје једноставнија и ефикаснија, а уједно се смањује могућност грешака у процесу развоја.

2. ТЕОРИЈСКЕ ОСНОВЕ МОДЕЛИМА ВОЂЕНОГ РАЗВОЈА СОФТВЕРА И НАМЕНСКИХ ЈЕЗИКА

Развој софтвера вођен моделима (енгл. *Model-Driven Software Development*) је методологија у којој модел представља централну тачку у процесу развоја софтвера. Сваки модел се креира помоћу неког језика за моделовање. Језици за моделовање обухватају апстрактну и конкретну синтаксу, као и семантику. Апстрактна синтакса дефинише структуру језика и начин на који се различити концепти могу комбиновати без обзира на репрезентацију [3]. Конкретна синтакса, која може бити текстуална или графичка, описује специфичну репрезентацију језика за моделовање [3]. Семантика језика за моделовање описује значења и ограничења дефинисаних концепата помоћу којих се креирају модели, те самим тим и модели имају јасно дефинисану семантику.
Наменски језици, познати и као језици специфични за домен (енгл. *Domain-Specific Languages*), представљају језике за моделовање који су уско специјализовани и пројектовани за специфичан домен, контекст или компанију. Језик *securaDSL* представља наменски језик јер је пројектован и прилагођен домену безбедносних конфигурација *Spring* веб апликација.
Трансформације модела су битан корак при развоју софтвера вођеним моделом и подразумевају аутоматско генерисање циљног модела или текста на основу изворног модела. У оквиру овог рада имплементиране су трансформације модела у текст, који представља извршиви код веб апликације.

2.1 Опис технологија коришћених за развој наменског језика *securaDSL* и генератора

При развоју наменског језика *securaDSL,* први корак представља креирање апстрактне синтаксе која је приказана помоћу мета-модела. За креирање мета-модела коришћен је радни оквир за моделовање и генерисање кода *Eclipse Modeling Framework (EMF)* који као језик за мета-моделовање користи језик *Ecore.*
Додатна ограничења описана су помоћу наменског језика *Object Constraint Language* (*OCL*), односно имплементације овог језика под називом *Eclipse OCL.* Након тога, на основу апстрактне креирана је конкретна синтакса помоћу радног оквира *Xtext.*
За развој генератора коришћен је програмски језик *Java* и језик *Xtend*.

Xtend доприноси детекцији типова података, побољшању синтаксе ламбда израза за лаку претрагу и издвајање података из модела и омогућава употребу шаблона помоћу којих се може описати изглед генерисаног кода.

3. ПРЕГЛЕД БЕЗБЕДНОСНИХ МЕХАНИЗАМА У *SPRING* АПЛИКАЦИЈАМА

Код који се генерише написан је у програмском језику *Java* коришћењем радног оквира *Spring.* Овај радни оквир је организован у модуле који нуде функционалности за подршку различитих аспеката развоја апликација.
Подржана су три релациона система за управљање базама података: *PostgreSQL, MySQL* и *Oracle Database.* Сва три система користе и проширују језик *SQL* за рад са подацима.
Радни оквир *Spring* пружа могућност безбедносне конфигурације веб апликација помоћу библиотеке *Spring Security*. Конфигурација се врши у складу са различитим безбедносним механизмима и на различитим нивоима, а конфигурацију је потребно прилагодити специфичним потребама система. Подржана су три безбедносна механизма: основна аутентификација, стандард *JSON* веб токен и стандард *Open Authorization.*

3.1 Основна аутентификација

Основна аутентификација (енгл. *Basic Authentication*) представља метод у ком се корисник идентификује помоћу корисничког имена и лозинке. При сваком захтеву се у заглављу захтева налазе идентификациони параметри корисника и на основу тога се потврђује идентитет и право приступа ресурсу или функционалности система. Овај метод се једноставно имплементира и користи, те је погодан за једноставније системе. Лозинке преносе у заглављу захтева, што их чини подложним нападима и може угрозити сигурност апликације, па се самим тим препоручује коришћење додатних безбедносних механизама.

3.2 Стандард *JSON* веб токен

JSON веб токен (енгл. *JSON web token, JWT*) представља формат за представу токена за аутентификацију. Састоји се од три дела: заглавље (енгл. *header*), главног дела (енгл. *payload*) и потписа (енгл. *signature*). Токен се генерише при свакој успешној аутентификацији и додељује се пријављеном кориснику, при чему садржи све неопходне податке о њему. При сваком захтеву се проверава валидност *JWT* токена на основу информација из њега и одређује се да ли је кориснику дозвољен приступ ресурсу или функционалности система.

3.3 Стандард *Open Authorization*

Стандард *Open Authorization* (*OAuth*) представља стандард за доделу права приступа који омогућава корисницима да доделе овлашћења апликацијама за приступ њиховим подацима који се налазе у другим апликацијама. Уместо уношења идентификационих параметара, сервер за доделу права приступа генерише токен који се користи за приступ ресурсима апликације. Овај стандард користи велики број компанија као што су *Google* и *Facebook,* како би омогућиле корисницима да поделе податке са својих корисничких налога са другим апликацијама. Последња верзија овог механизма је *OAuth2.0.*

4. ПРЕГЛЕД ПОСТОЈЕЋЕГ СТАЊА У ОБЛАСТИ

Савремене методе развоја софтвера теже брзом, безбедном и ефикасном развоју и смањењу потребе за ручно писаним кодом, што је довело до повећаног броја решења за генерисање различитих видова и нивоа софтвера.

Spring Initializer [4] је алат за брзо креирање основне структуре *Spring* апликација. Погодан је за једноставне пројекте, али је неопходна ручна конфигурација безбедносних аспеката, базе података и имплементација осталих слојева апликације.

У радовима [5,6] описани су алати који омогућавају и олакшавају процес спецификације и конфигурације микросервисне архитектуре, а уз то и генерисање извршивог кода за овако конфигурисане апликације. Ова решења нису погодна за генерисање монолитних апликација, иако су се показала као ефикасна за развој микросервисне архитектуре.

Анализа постојећих решења за моделовање апликација довела је до закључка да ниједно од решења не испуњава све захтеве у потпуности. Главни проблем који је уочен је недостатак могућности за конфигурисање безбедносних аспеката апликације.

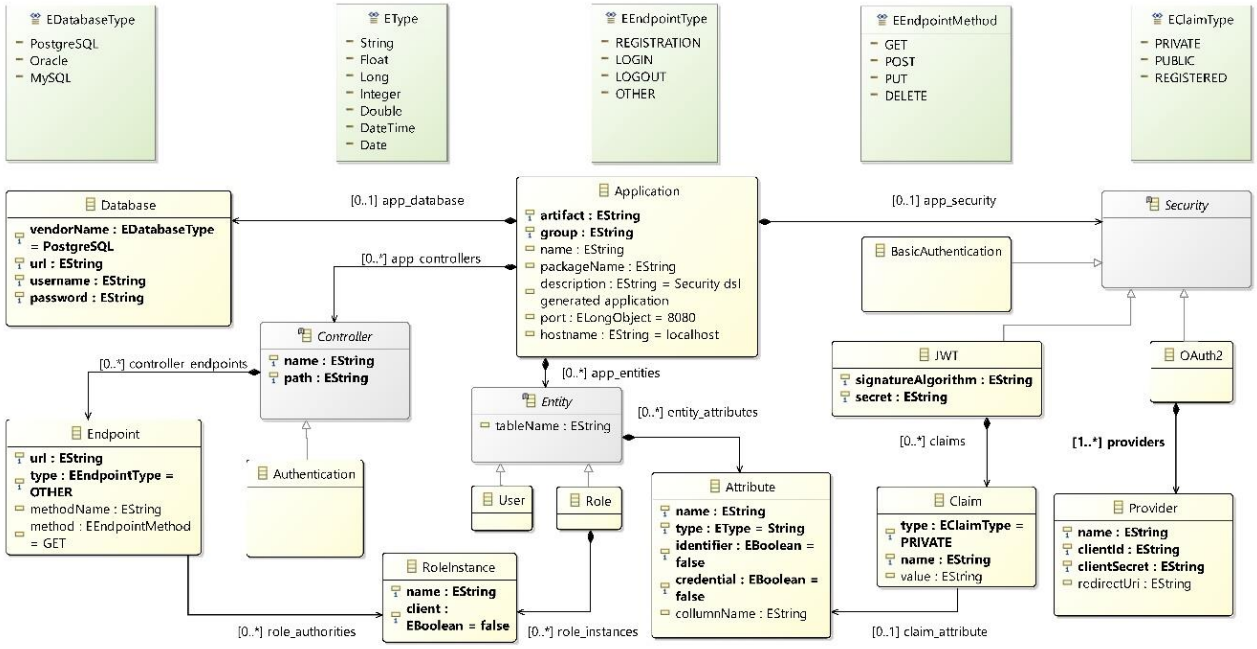
5. НАМЕНСКИ ЈЕЗИК ЗА ПОДРШКУ БРЗОГ УСПОСТАВЉАЊА КОНФИГУРАЦИЈЕ БЕЗБЕДНОСНИХ АСПЕКАТА У РАДНОМ ОКВИРУ *SPRING*

Недостаци описани у претходном поглављу, доводе до потребе за развојем наменског језика који би омогућио брзо успостављање конфигурације безбедносних аспеката. Да би наменски језик омогућио брзу и ефикасну конфигурацију у радном оквиру *Spring,* неопходно је да подржи моделовање свих неопходних концепата за иницијализацију апликације. Основни концепти се могу сврстати у целине које се односе на метаподатке апликације, параметре базе података, слој за моделовање података складиштених у бази података, обраду захтева корисника и безбедносну конфигурацију. Наменски језик *securaDSL* подржава конфигурацију три безбедносна механизма: основну аутентификацију, стандард *JWT* и *OAuth2.0.* Овим се обезбеђује флексибилност и могућност одабира оптималног безбедносног механизма у зависности од потреба и захтева система.

5.1 Апстрактна синтакса

Апстрактна синтакса омогућава опис структуре наменског језика *securaDSL* и представљена је помоћу мета-модела (Слика 4.1). Коренски концепт апстрактне синтаксе је апликација (*Application*) и садржи податке неопходне за њену иницијализацију. Могуће је подесити параметре за повезивање са базом података *(Database)*, а поред тога, могу се дефинисати ентитети (*Entity*) који одговарају табелама у бази података. Описани су ентитети који се односе на кориснике (*User*) и улоге (*Role*). При дефинисању ентитета, неопходно је навести обележја (*Attribute*) за сваки ентитет. За концепт који се односи на обраду захтева корисника (*Controller*) је могуће додати методе (*Endpoint*) које се односе на различите функционалности апликације. Навођењем инстанци улога (*RoleInstance*) које се налазе у систему и повезивањем са одређеним методама омогућена је контрола приступа. Контролер за аутентификацију (*Authentication*) може да садржи методе за регистрацију, пријаву на систем и одјаву са система.

Посебан део апстрактне синтаксе односи се на сигурносни слој (*Security*), где су подржана три безбедносна механизма: основна аутентификација (*BasicAuthentication*), стандард *JWT* (*JWT*) и *OAuth2.0 (OAuth2).* У зависности од жељеног механизма могу се дефинисати додатни параметри описани адекватним концептима.



Слика 4.1 – Апстрактна синтакса наменског језика *securaDSL*

Сваки од наведених концепата може се додати или изоставити у зависности од потреба корисника.

5.2 Конкретна синтакса

Коришћењем радног оквира *Xtext*, на основу мета-модела генерисана је почетна верзија текстуалне конкретне синтаксе. Ова граматика прилагођена је домену како би била лако читљива и интуитивна за развојне тимове којима је *securaDSL* намењен. Почетна верзија граматике подсећа на уобичајене синтаксе за дефинисање разних конфигурационих фајлова, те таква граматика не захтева велике измене узимајући у обзир да циљну групу чине експерти у пољу безбедносне конфигурације. Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *OAuth2.0* приказан је на Листингу 5.1.

```
application:
        artifact: "securaDSL"
        group: "uns.ftn"
        port: 8080
        hostname: "localhost"

database:
  vendor: PostgreSQL
  url: "localhost:5432/securaDSL"
  username: "securaDSL"
  password: "securaDSL"

security:
        OAuth2.0:
          providers: [
            {
              name: "google",
              clientId: "x",
              clientSecret: "x"
            },
            {
              name: "github",
              clientId: "x",
              clientSecret: "x"
            }
          ]
```

Листинг 5.1 - Пример модела *Spring* веб апликације са конфигурисаним безбедносним механизмом *OAuth2.0*

6. ГЕНЕРИСАЊЕ *SPRING* ВЕБ АПЛИКАЦИЈА СА БЕЗБЕДНОСНОМ КОНФИГУРАЦИЈОМ

С обзиром да се *Spring* апликација може поделити у целине које су претходно описане и да постоји више подржаних безбедносних механизама, ради прегледности развијено је више генератора. Развијени су следећи генератори: генератор статичких датотека, генератор општих конфигурационих фајлова, генератор слоја који моделује податке из базе података, генератор слоја за обраду захтева корисника и генератори конфигурационих фајлова за основну аутентификацију, стандард *JWT* и стандард *OAuth2.0*.

Генератори формирају излаз на основу података из модела описаног помоћу наменског језика *securaDSL*, чиме се генерише извршиви код за *Spring* веб апликацију са конфигурисаним одабраним безбедносним механизмом. Који генератори ће генерисати излазни код зависи од тога који су концепти дефинисани у моделу. Нпр. уколико је дефинисана основна аутентификација, генератори конфигурационих фајлова за стандарде *JWT* и *OAuth2.0* неће имати излаз.

7. ЗАКЉУЧАК

Наменски језик *securaDSL* описан у овом раду садржи концепте неопходне за моделовање *Spring* апликација, а додатно подржава конфигурацију три безбедносна механизма: основне аутентификације, *JWT* и *OAuth2.0*. Поред наменског језика, описани су генератори који су развијени у циљу генерисања извршивог кода на основу модела креираног помоћу језика *securaDSL*. Коришћење наменског језика за моделовање *Spring* веб апликација са конфигурисаним безбедносним аспектима и пропратних генератора значајно убрзава процес развоја софтвера и умањује могућност грешке која настаје при ручном писању кода. Постојање модела апликације помаже у уочавању делова које захтевају измену, а које би иначе изазвале дораду која би утицала на велики број линија кода и самим тим довела до грешака које се тешко идентификују и отклањају. Структура саме апликације је видљивија и подложнија дискусији унутар развојног тима. Аутоматско генерисање кода повећава ефикасност и доприноси конзистентности, што унапређује квалитет саме апликације.

Остављен је простор за проширење тренутног начина за моделовање и генератора за безбедносни механизам *OAuth 2.0* увођењем регистрације на систем. Могуће је увести подршку за додатне безбедносне механизме, друге програмске језике и радне оквире увођењем нових концепата. Ово би захтевало измену апстрактне и конкретне синтаксе, али и развој нових генератора који би креирали код у различитим програмским језицима. Оваква унапређења би допринела значају и примени наменског језика *securaDSL* и развијених генератора.

ЛИТЕРАТУРА

[1] Sujay, Vailshery, L. (2023). Most used programming langugages worldwide as of 2023. [Online], Приступљено датума: 31.8.2023. https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/

[2] Maple, S., & Binstock A. (2018). JVM Ecosystem Report 2018: About your Platform & Application. [Online], Приступљено датума: 31.8.2023. https://snyk.io/blog/jvm-ecosystem-report-2018-platform-application/

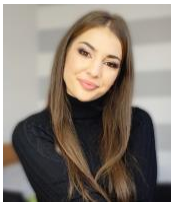
[3] Brambilla, M., Cabot, J., & Wimmer, M. (2012). Model-driven software engineering in practice.

[4] Spring Initializer. [Online], Приступљено датума: 31.8.2023. https://start.spring.io/

[5] Šuljkanović, A., Milosavljević, B., Inđić, V., Dejanović, I. (2022). Developing Microservice-Based Applications Using the Silvera Domain-Specific Language

[6] Terzić, B., Dimitrieski, V., Kordić, S., Milosavljević, G., Luković, I. (2017). MicroBuilder: A Model-Driven Tool for the Specification of REST Microservice Architectures

Кратка биографија:



Јелена Хрњак рођена је 21. августа 1999. године у Бачкој Тополи у Републици Србији. Школске 2018/2019. године уписала се на Факултет техничких наука, смер. Дипломски рад из области Електротехнике и рачунарства одбранила је 2022. године.