



Имплементација *Spring MVC* веб апликација уз ослонац на инжењерство вођено моделима

Implementation of Spring MVC Web Applications Based on Model-Driven Engineering

Александар Радивојевић, Факултет техничких наука, Нови Сад

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – У овом раду представљен је генератор програмског кода који генерише Java и HTML датотеке за *Spring MVC* веб апликације. Генератор као улаз користи UML модел класног дијаграма серијализован у XMI формату. Над моделом су примењени стереотипи UML профила *SpringMVCProfile* у алату *Eclipse Papyrus*. Примена генератора демонстрирана је у домену библиотечног информационог система БИСИС, на моделу циркулације.

Кључне речи: генератори кода, инжењерство вођено моделима, *Spring MVC*

Abstract – This work presents a code generator that produces Java and HTML source files for *Spring MVC* web applications. The generator takes as input a UML class diagram model serialized in the XMI format. The stereotypes of the *SpringMVCProfile* UML profile were applied to the model in the *Eclipse Papyrus* tool. The application of the generator is demonstrated in the domain of the *BISIS* library information system, on the circulation model.

Keywords: code generators, model-driven engineering, *Spring MVC*

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је била др Гордана Милосављевић, ред. проф.

1. УВОД

Сложеност савремених софтверских система непрекидно расте, а са њом и количина програмског кода који је потребно одржавати, синхронизовати и развијати кроз више технолошких слојева. Традиционални развој, заснован на ручном писању кода, не прати ову динамику – додавање или измена компоненте захтева усклађивање над више нивоа апликације, од базе података преко серверске до клијентске стране. Модерни алати за развој софтвера, као што је *IntelliJ IDEA*, нуде аутоматизовано рефакторисање (енг. *refactoring*), међутим када су у питању вишеслојне веб апликације, оно је ограничено

искључиво на серверску страну (енг. *backend*). Једно од решења је софтверско инжењерство вођено моделима (енг. *Model-Driven Software Engineering, MDSE*), или краће инжењерство вођено моделима (енг. *Model-Driven Engineering, MDE*). *MDE* представља приступ у којем модели не служе само као документација, већ као примарни елементи из којих се могу аутоматски или полуаутоматски генерисати софтверске компоненте [1]. У овом раду принципе *MDE* примењујемо кроз језик за моделовање UML (*Unified Modeling Language*) [2] у алату *Eclipse Papyrus* [3]. Користећи класни дијаграм (енг. *class diagram*) моделујемо ентитете и њихове везе, а дијаграм се серијализује у XMI формату (*XML Metadata Interchange*). Овакав формат је погодан за програмску обраду и парсирање (енг. *parse*) након којег се модел трансформише у Java доменске објекте погодне за генерисање кода. Генератор применом *Apache FreeMarker* [4] шаблона (енг. *templates*) над Java објектима производи изворни код апликације. Као циљна платформа изабран је *Spring MVC – framework* за развој веб апликација у Java екосистему, чија слојевита архитектура природно одговара структури коју је могуће извести из UML модела. Практична примена генератора кода демонстрирана је у домену библиотечног информационог система БИСИС [5], на моделу циркулације.

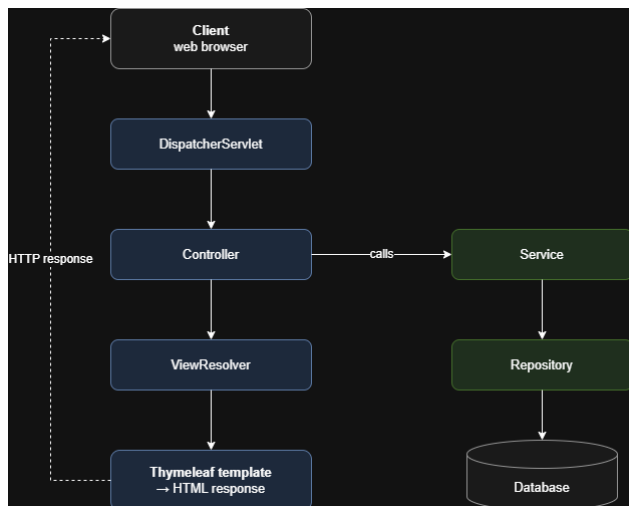
2. АРХИТЕКТУРА SPRING MVC ВЕБ АПЛИКАЦИЈА

Spring MVC је веб *framework*, део *Spring* платформе, који користи *MVC* (енг. *Model-View-Controller*), архитектурни образац који раздваја апликациону логику на три међусобно повезане компоненте:

- **Model** – представља податке и пословну логику апликације. У контексту *Spring MVC*-а, модел обухвата објекте који се прослеђују презентационом слоју ради приказа.
- **View** – одговоран је за визуелни приказ података кориснику.
- **Controller** – прихвата *HTTP* захтеве од клијента, делегира обраду сервисном слоју и враћа одговарајући *view* са припремљеним подацима.

Апликације засноване на *Spring MVC* типично се организују у слојеве, од којих сваки има јасно дефинисану одговорност: *Entity* (*JPA* ентитети који се пресликавају на табеле у бази), *Repository* (приступ подацима кроз *Spring Data JPA*), *Service* (пословна логика), *Controller* (обрада *HTTP* захтева) и *DTO/Mapper* (раздвајање презентационог од ентитетског слоја).

Пример тока једног *HTTP* захтева у *Spring MVC* приказан је на слици 1.



Слика 1. Ток обраде *HTTP* захтева кроз слојеве *Spring MVC* апликације

3. ДЕФИНИСАЊЕ ПРОФИЛА

Профил *SpringMVCProfile* (Слика 2) дефинисан је у алату *Eclipse Papyrus*. Профил проширује стандардни *UML* увођењем стереотипа прилагођених домену

Spring MVC веб апликација, где сваки стереотип садржи означене вредности које прецизирају понашање генерисаног кода.

SpringMVCProfile састоји се од стереотипа:

MVCElement – проширује метакласу *Element*, садржи атрибут *label* за назив елемента

MVCProperty – проширује метакласу *Property*, наслеђује *MVCElement*, садржи атрибуте за опис својства ентитета (*nullable*, *minLength*, *maxLength*, *toString*, *unique*).

MVCView – проширује метакласу *Class*, наслеђује *MVCElement*, апстрактни стереотип за *view*

MVCForm – наслеђује *MVCView*, служи за означавање форме на *View*-у, садржи атрибут *readonly* – да ли је могуће мењати елемент.

MVCList – наслеђује *MVCView*, служи за означавање приказа листе на *view*-у, садржи атрибуте: *pageSize*, *deleteEnabled*, *createEnabled*, *editEnabled*.

SortBy – наслеђује *MVCProperty* и служи за означавање елемента по ком се сортира листа, садржи атрибут *sortDirection* који дефинише смер сортирања, одређен еnumerацијом *SortDirection*

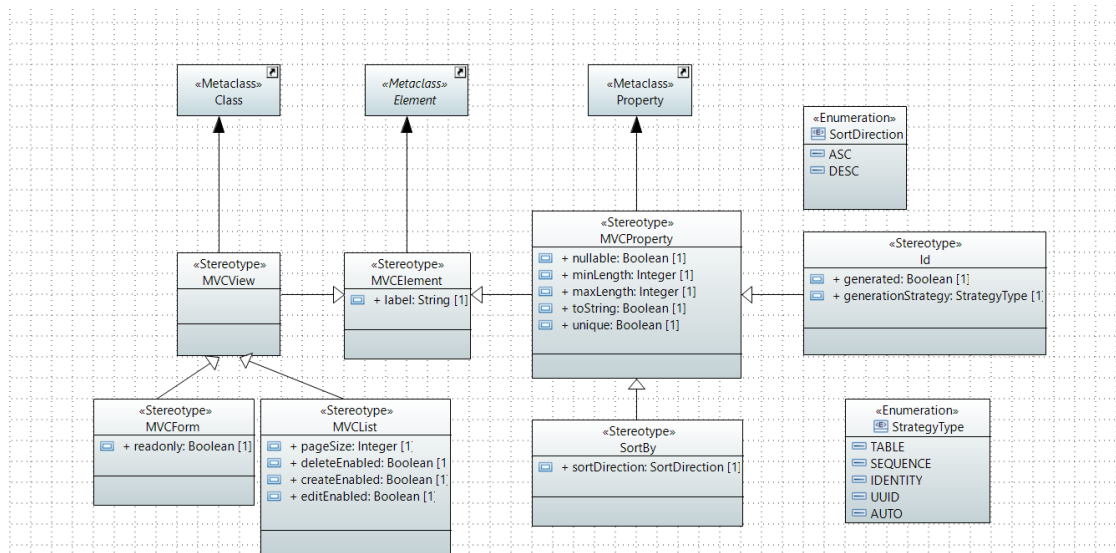
Id – наслеђује *MVCProperty* и служи за означавање примарног кључа, садржи атрибут *generated* – да ли је аутоматски генерисан и *generationStrategy* који дефинише тип стратегије генерисања, одређен еnumerацијом *StrategyType*.

Профил садржи и две еnumerације:

SortDirection – одређује смер сортирања листе. Могуће вредности *ASC* и *DESC*

StrategyType – одређује стратегију генерисања примарног кључа у бази података.

Могуће вредности: *TABLE*, *SEQUENCE*, *IDENTITY*, *UUID* и *AUTO*.



Слика 2. *UML* Профил *SpringMVCProfile*

4. АРХИТЕКТУРА ГЕНЕРАТОРА ПРОГРАМСКОГ КОДА

Генератор програмског кода имплементиран је у језику *Java 17* и ослања се на *SAX* парсер за читање модела и *Apache FreeMarker* за генерисање изворног кода на

основу шаблона. Организован је у пет *Java* пакета и директоријум са *FreeMarker* шаблонима (Слика 3).

Јава пакети су:

com.codegenerator.parser – садржи класе задужене за читање *XMI* датотеке и генерисање излазних компоненти: *Reader*, *UmlModelResolver*, *FileGenerator*, *Loader*, *XmiToJavaTypeMapper*, као и помоћне класе

XmiConstants, *StereotypeName* и енумерацију *Component*.

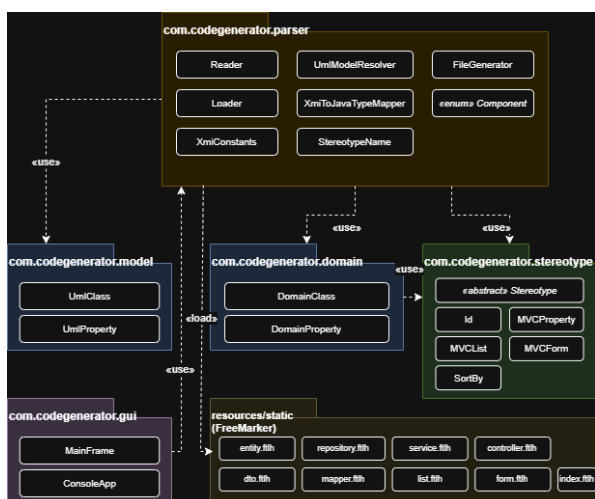
com.codegenerator.model – садржи класе *UmlClass* и *UmlProperty* које представљају UML модел директно попуњен из XMI парсера.

com.codegenerator.domain – садржи класе *DomainClass* и *DomainProperty* које представљају доменски модел прилагођен за потребе *FreeMarker* шаблона.

com.codegenerator.stereotype – садржи класе које моделују стереотипе из *SpringMVCProfile* профила: апстрактна база класа *Stereotype* и њене конкретне имплементације *Id*, *MVCProperty*, *MVCList*, *MVCForm* и *SortBy*.

com.codegenerator.gui – садржи графички (*MainFrame*) и конзолни (*ConsoleApp*) интерфејс за покретање генератора.

FreeMarker шаблони налазе се у директоријуму *src/main/resources/static/* и носе називе по компонентама које генеришу: *entity.fihl*, *repository.fihl*, *service.fihl*, *controller.fihl*, *dto.fihl*, *mapper.fihl*, *list.fihl*, *form.fihl* и *index.fihl*.



Слика 3. Организација пакета пројекта

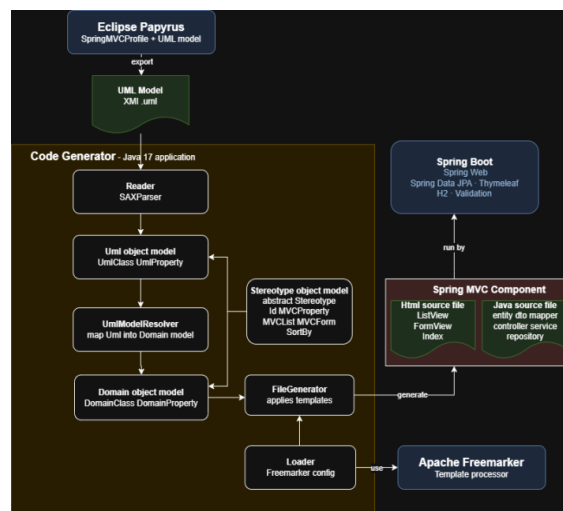
На слици 4 приказана је архитектура генератора програмског кода. Процес генерисања подељен је у више фаза:

Читање и парсирање – Улаз у генератор је формат XMI, серијализација UML модела класног дијаграма са примењеним стереотипима профила *SpringMVCProfile*. SAX парсер (класа *Reader*) секвенцијално парсира XMI датотеку и издваја UML класе, атрибуте и примењене стереотипе, при чему класа *XmiToJavaTypeMapper* мапира UML/Ecore типове на одговарајуће Java типове.

Обрада и мапирање на доменски модел – класа *UmlModelResolver* семантички обрађује парсиране UML елементе кроз три корака: разрешава асоцијације, одређује кардиналност релација и трансформише UML модел (*UmlClass*, *UmlProperty*) у доменски модел (*DomainClass*, *DomainProperty*) и повезује примењене стереотипе са одговарајућим елементима доменског модела.

Примена шаблона и генерисање датотека – Класа *FileGenerator* за сваку класу из доменског модела примењује одговарајуће *FreeMarker* шаблоне по компонентама и генерише датотеке изворног кода.

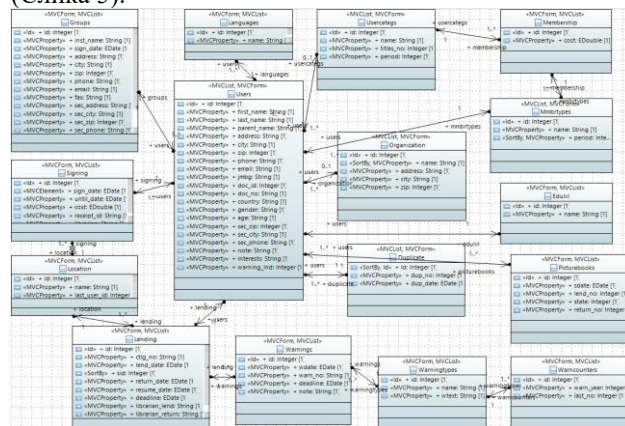
Резултат генерисања је комплетна вишеслојна *Spring MVC* веб апликација која садржи ентитете, репозиторијуме, сервисе, контролере, *DTO* објекте, мапере и *Thymeleaf HTML view*.



Слика 4. Архитектура CodeGenerator-a

5. ПРИМЕНА НА МОДЕЛУ ЦИРКУЛАЦИЈЕ

Примена генератора демонстрирана је за домен библиотечких информационих система (БИСИС), на моделу циркулације. У *Eclipse Papyrus*-у је дефинисан класни дијаграм модела циркулације над којим су примењени стереотипи из профила *SpringMVCProfile* (Слика 5).



Слика 5. Дијаграм класа модела циркулације

Користећи дијалог *MainFrame* или конзолну апликацију *ConsoleApp* са подешеном путањом до локације модела циркулације као улазом и путањом до *Spring Boot* пројекта као излазом, генерише се програмски код. Пример једне генерисане форме приказан је на слици 6.

6. ЗАКЉУЧАК

У оквиру овог рада пројектован је и имплементиран генератор програмског кода који на основу UML класног дијаграма, са примењеним стереотипима из профила *SpringMVCProfile*, аутоматски производи комплетну *Spring MVC* веб апликацију. Примена генератора верификована је на моделу циркулације библиотечког информационог система БИСИС, где је

Слика 6. Генерисана форма *Lending*

из једног модела од 16 класа аутоматски генерисано 129 датотека. Коришћењем генератора убрзава се развој и смањује број грешака у односу на ручно писање кода. Синхронизација између слојева *Spring MVC* веб апликације постиже се новим покретањем генератора, свака промена у моделу одражава се на све генерисане компоненте. Примена моделом вођеног приступа у алату *Eclipse Papyrus* обезбеђује прегледност при пројектовању веб апликација, јер се домен дефинише кроз ентитете и њихове везе на нивоу класног дијаграма. Недостатак приступа произилази из самог алата *Eclipse Papyrus*, који не подржава групну примену стереотипа над елементима класног дијаграма, сваки атрибут и свака класа морају се означити појединачно, што за веће моделе постаје једноличан и временски захтеван процес. Потенцијални правци даљег развоја укључују подршку за генерисање *REST API* контролера и интеграцију генератора у *CI/CD* процес.

7. ЛИТЕРАТУРА

- [1] M. Brambilla, J. Cabot, and M. Wimmer. Model-Driven Software Engineering in Practice, 2nd Edition. Morgan & Claypool Publishers, 2017.
- [2] Object Management Group. OMG Unified Modeling Language (OMG UML), Version 2.5.1. OMG Document Number formal/2017-12-05, 2017.
- [3] Eclipse Papyrus. <https://eclipse.dev/papyrus/>
- [4] The Apache Software Foundation. Apache FreeMarker Manual. <https://freemarker.apache.org/>
- [5] Библиотечки информациони систем (БИСИС). <https://www.bisis.rs/>

Кратка биографија:



Александар Радивојевић рођен је у Лесковцу 1991. год. Дипломирао је на Факултету техничких наука, област Електротехника и рачунарство – Примењене рачунарске науке и информатика 2015. год. Исте године уписао је мастер студије на смеру Инжењеринг информационих система.

Контакт:

radivojevic.aleksandar91@gmail.com