Приложение на инжектирана верига отговорности в университетска информационна система

Мирослав Михайлов, Цветелин Павлов, Венцислав Йорданов

Application of injected chain of responsibility pattern in university information system: *This paper describes the modification and subsequent application of chain of responsibility and dependency injection design patterns in application shell for university information system.*

Keywords: *university information system, chain of responsibility, dependency injection, application shell.*

ВЪВЕДЕНИЕ

Университетската информационна система, разработвана в Русенски университет, притежава приложна обвивка, върху която се изграждат всички клиентски приложения на системата. Основната ѝ задача е да раздели модулите на системата от конкретните библиотеки на потребителския интерфейс, както и да обособи често използваните функционалности в преизползваеми модули. На няколко места в приложната обвивка е използвано съчетание от шаблони за дизайн, разработено специално за целта, наречено инжектирана верига отговорности. То включва елементи от добре познатите шаблони „верига отговорности“, „декоратор“ и „инжекция на зависимости“.

Верига отговорности

“Верига отговорности” се дефинира като шаблон, чрез който се избягва обвързването на изпращача на дадена заявка с получателя ѝ като дава възможност на няколко обекта да обработят заявката (1).

Шаблонът се използва, когато е възможно повече от един обект да обработи дадена заявка и/или изпълнителят не е известен предварително, когато е необходимо обектите-обработчици да бъдат създадени динамично, както и когато изпращачът не се интересува от обекта-обработчик.

Благодарение на шаблона „Верига отговорности“ се осигурява слабо обвързване между обектите при един от посочените по-горе сценарии, както и допълнителна гъвкавост по време на изпълнение (възможно е веригата да бъде модифицирана по време на изпълнение, като се вмъкнат или добавят обекти).

Най-често срещаната употреба е за избягване на големи условни блокове (switch statements). При употреба на условен блок е необходимо модулът, в който се намира условния блок, да има твърди референции към всеки един от класовете, участващи в клоновете на условния блок, а често тези класове идват от модули, които не е задължително да бъдат разпространявани заедно или са взаимно изключващи се.

В контекста на университетската информационна система, шаблонът е използван в приложната обвивка за организиране на верига отговорности от адаптери за елементи на потребителския интерфейс, които могат да бъдат разположени в различни региони на потребителския интерфейс. Решението къде трябва да бъде поставен даден елемент се взима от конкретния обект-обработчик на базата на свойствата на подадения за обработка обект.

Основният недостатък на шаблона е необходимостта всеки конкретен обект-обработчик да знае кои е следващия обект-обработчик по веригата, което налага промяна на класовете на обработчиците (нежелан ефект) като им се даде възможност следващия обработчик да бъде твърдо кодиран в класа (силно обвързване) или да бъде инжектиран от обекта-потребител на веригата отговорности (силно обвързване при обекта-потребител).

Декоратор

„Декоратор“ е шаблон, позволяващ динамично възлагане на отговорности на даден обект. Декораторите са гъвкава алтернатива на създаването на подкласове при разширяване на функционалността (1) (2).

Шаблонът се използва, когато е необходимо прозрачно да се добавят или отнемат отговорности и задължения към даден обект по време на употреба. „Декоратор“ се използва и когато разширяването чрез наследяване е непрактично (например когато води до експлозия от класове).

Употребата на декоратор води до по-голяма гъвкавост в сравнение със статичното наследяване, избягва се претоварването на класовете в йерархията и като цяло обектноориентираният дизайн става съставен от множество малки класове в плитка йерархия, което улеснява модулното тестване и промяната и поддръжката на кода в последствие.

Най-често шаблонът „Декоратор“ се използва, когато е необходимо да бъде разширен клас, който вече е част от съществуваща йерархия и има наследници и/или наследяването му е невъзможно. Тогава, вместо да бъде наследен, класът се „декорира“, като за обектите-потребители на този клас, декораторът е прозрачен (т.е. те го „виждат“ като обект на целевия клас).

В контекста на информационната система, шаблонът декоратор беше използван за да се декорират отделните адаптери на компонентите за потребителски интерфейс с функционалностите, необходими за да участват тези класове във верига отговорности. Така приложният интерфейс на адаптерите остана непроменен.

Инжекция на зависимости

Инжекцията на зависимости е шаблон за дизайн позволяващ разделянето на поведение от резолюция на зависимостите (3). Шаблонът е специфично проявление на принципа на обръщане на контрола (4).

Шаблонът се използва, когато даден клас има необходимост от референции към други класове, за да изпълни задълженията си (за да определи поведението си). При нормални обстоятелства, класът сам трябва да провери дали всички външни обекти са създадени и дали са в правилното състояние, както и да създаде композираните обекти. С помощта на инжекция на зависимости, тази грижа се възлага на външен клас, който има за цел да поддържа списъка с възможни резолюции и да инжектира (да присвои на свойство, поле или аргумент на конструктор или метод) съответните инстанции, когато е необходимо (най-често при конструиране на целевия обект).

В контекста на информационната система инжекция на зависимости се използва по класическия начин[1] както в клиентските приложения, така и на сървърната страна при услугите.

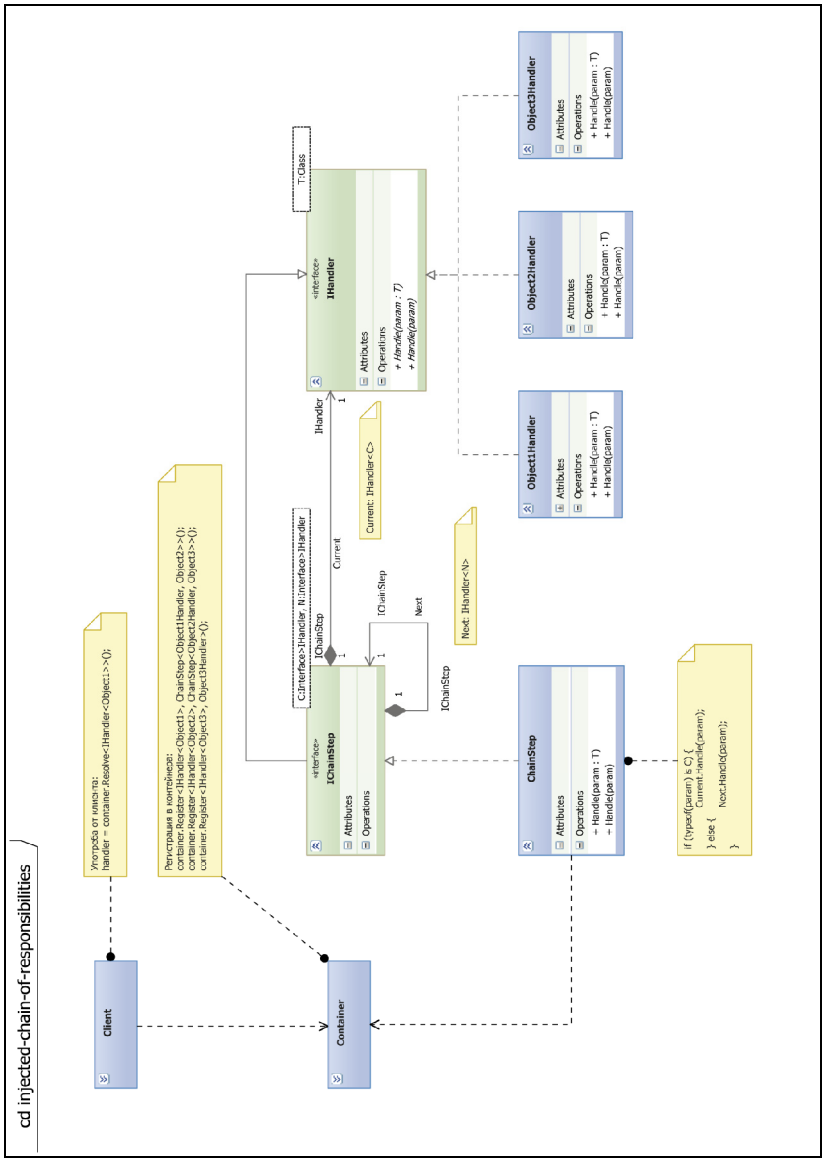
Инжектирана верига отговорности

Инжектирана верига отговорности е съчетание на шаблоните „верига отговорности“, „декоратор“ и „инжекция на зависимости“. Участниците във верига отговорности са декорирани, което позволява инжекция на следващия елемент от веригата отвън, а участниците нямат знание, че участват във верига отговорности.

Примерна диаграма на класовете, участващи в шаблона е показана на фиг.1.

Целевите класове имплементират IHandler интерфейса, който дефинира някаква операция, която трябва да се извърши. Интерфейсът е шаблонен, а параметърът определя коя от конкретните имплементации трябва да се използва. IChainStep наследява IHandler и същевременно композира негова инстанция.

[1] Използва се Unity от Enterprise Library в съчетание с Prism за клиентските приложения.



Фиг. 1 Диаграма на класовете

Конкретната имплементация на IChainStep извършва проверка дали текущата инстанция на обекта от веригата е подходяща за изпълнение и в зависимост от резултата предава управлението или на конкретния целеви клас или на следващия клас от веригата. IChainStep също е шаблонен, а двата класа - текущият и следващият се подават като параметри. Веригата се организира чрез регистрациите в контейнера за инжекция на зависимости, като на всяка една стъпка може да се

регистрира както конкретната имплементация на IHandler, така и ChainStep, като в първия случай веригата ще се прекъсне.

Веригата отговорности се организира от IChainStep, който съдържа референция към следващия изпълнител. Тази инстанция се инжектира при инстанциране на класа, имплементиращ IChainStep. От друга страна, IChainStep е декоратор на IHandler и съдържа референция към конкретния целеви клас. Това позволява прозрачна регистрация в контейнера и взаимозаменяемост на IChainStep изпълнителите и конкретния целеви клас.

В контекста на университетската информационна система, инжектирана верига отговорности се използва в клиентските приложения за резолюция на компоненти от потребителския интерфейс. Клиентските приложения в системата се базират на приложна обвивка, изградена като композитно приложение. Композитните приложения се състоят от множество модули, всеки от които може да регистрира компонент на потребителския интерфейс в определените за целта региони. Регистрацията става чрез контейнер за инжекция на зависимости. При нормална употреба всеки от модулите, регистриращи компоненти на потребителския интерфейс трябва да бъде силно обвързан с библиотеките, използвани за потребителския интерфейс - нежелан ефект, заради който приложенията биха загубили гъвкавост. Затова приложната обвивка на информационната система притежава абстрактен слой от класове, описващи потребителския интерфейс, които нямат пряко графично изражение. Всеки модул, който добавя библиотека за потребителски интерфейс, трябва добави и адаптери за класовете от абстрактния слой. За да се регистрират тези адаптери за резолюция се използва инжектирана верига от отговорности. Така конкретните адаптери „се интересуват“ само от собствените си задължения, а самата верига отговорности може да бъде пренаредена от модулите чрез промяна на регистрациите в контейнера за инжекция на зависимости.

ЗАКЛЮЧЕНИЕ

Употребата на шаблона „инжектирана верига отговорности“ позволява пълно разделяне на логическите модули на информационната система от конкретната технология, използвана за изграждане на потребителския интерфейс на клиентските приложения. Това улеснява миграцията на клиентските приложения към друг вид потребителски интерфейс, поддръжката и последващо развитие на модулите.

Шаблонът е разработен специално за приложната обвивка на университетската информационна система, но има приложение навсякъде, където е необходимо да се избегнат недостатъците на класическия шаблон „верига отговорности“, както и когато е необходимо конкретните класове, участващи във веригата да не знаят по никакъв начин за участието си в нея.

ЛИТЕРАТУРА

[1] Гама, Ерик, и др., и др. *Шаблони за дизайн: Елементи на обектно-ориентирания софтуер за многократно използване.* н.м. : СофтПрес ООД, 2005. 0201633612.

[2] Chain-of-responsibility pattern. *Wikipedia.* [Онлайн] Wikimedia Foundation, Inc., 26 септември 2010 г. [Цитирано на: 30 септември 2010 г.] http://en.wikipedia.org/wiki/Chain-of-responsibility_pattern.

[3] Dependency injection. *Wikipedia.* [Онлайн] Wikimedia Foundation, Inc.,, 19 октомври 2010 г. [Цитирано на: 26 октомври 2010 г.] http://en.wikipedia.org/wiki/Dependency_injection.

[4] Inversion of control. *Wikipedia.* [Онлайн] Wikimedia Foundation, Inc.,, 10 октомври 2010 г. [Цитирано на: 26 октомври 2010 г.] http://en.wikipedia.org/wiki/Inversion_of_control.

За контакти:
доц. д-р Мирослав Михайлов, директор на ЦИКО, Русенски университет „Ангел Кънчев“, тел.: 082 888 782, e-mail: mmihaylov@uni-ruse.bg.
маг. инж. Цветелин Павлов, ЦИКО, Русенски университет „Ангел Кънчев“, тел.: 082 888 328, e-mail: cpavlov@uni-ruse.bg.
маг. инж. Венцислав Йорданов, ЦИКО, Русенски университет „Ангел Кънчев“, тел.: 082 888 328, e-mail: vkiordanov@uni-ruse.bg.

Публикацията е реализирана по проект по ФНИ 2010 РУ-02.

Докладът е рецензиран.

РУСЕНСКИ УНИВЕРСИТЕТ „АНГЕЛ КЪНЧЕВ”
UNIVERSITY OF RUSE „ANGEL KANCHEV“

Д И П Л О М А

**Програмният комитет на
Научната конференция РУ&СУ’10
награждава с КРИСТАЛЕН ПРИЗ
“THE BEST PAPER”
Мирослав Михайлов,
Цветелин Павлов и Венцислав Йорданов
автори на доклада
“Приложение на инжектирана верига
отговорности в университетска
информационна система”**

D I P L O M A

**The Programme Committee of
the Scientific Conference RU&SU'10
Awards the Crystal Prize
"THE BEST PAPER"
to Miroslav Mihaylov, Tsvetelin Pavlov
and Ventsislav Yordanov
authors of the paper
“Application of injected chain of responsibility
pattern in university information system”**

РЕКТОР — доц. д-р Христо Белоев
RECTOR — Prof. D-r Hristo Beloev

01.11.2010