Міністерство освіти і науки України

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ПОЛІТЕХНІЧНИЙ УНІВЕРСИТЕТ

Кафедра радіотехнічних пристроїв

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА

з дисципліни: «Сучасні технології розробки програмного забезпечення»

Розробка веб-додатку на основі Spring Boot

|  |  |  |
| --- | --- | --- |
| Напрям підготовки 6.170102 –  «Системи технічного захисту інформації»  Курс 4  Семестр 7  Навчальний рік 2017 / 2018 |  | Розробив ст. групи РБ-141  \_\_\_\_\_\_\_\_\_\_\_\_ Казімір О.А.  Керівники роботи ст. викл.  \_\_\_\_\_\_\_\_\_\_\_\_ І.А. Белека  \_\_\_\_\_\_\_\_\_\_\_\_ Н.І. Кушніренко  Кількість балів \_\_\_\_\_\_\_\_\_\_\_\_\_\_  “\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_ 2017 р. |

Одеса 2017

ЗАВДАННЯ

1. Розробити власний веб-додаток із застосуванням Spring Boot.

2. Створити веб-сторінку з кнопкою «Отримати список об’єктів».

3. Звернутися асинхронно за допомогою технології AJAX до API функції http://radar.lafox.net/api/getInvaders методом GET і отримати список відомих об’єктів.

4. Вивести отримані об’єкти на веб-сторінку.

5. Додати кнопку «Отримати поточну карту».

6. Звернутися методом GET асинхронно (AJAX) до API функції http://radar.lafox.net/api/getMap і отримати поточний знімок карти и унікальний "id" карти.

7. Показати карту на веб-сторінці.

8. Додати кнопку «Знайти на карті об’єкти».

9. Написати сервіс (як частина Spring Boot додатку) для пошуку об’єктів.

10. Відобразити таблицю знайдених об’єктів NAME, X,Y на веб-сторінці.

11. Відіслати список знайдених об’єктів в JSON форматі на API http://radar.lafox.net/api/checkMyResult методом POST.

ЗМІСТ

[ВСТУП 4](#_Toc500706817)

[1 ТЕОРЕТИЧНІ ВІДОМОСТІ 5](#_Toc500706818)

[1.1 Apache Maven 5](#_Toc500706819)

[1.2 XML 5](#_Toc500706820)

[1.3 JSON 7](#_Toc500706821)

[1.4 REST 8](#_Toc500706822)

[1.5 AJAX 11](#_Toc500706823)

[1.6 Spring Framework 12](#_Toc500706824)

[1.7 Spring Boot 13](#_Toc500706825)

[1.8 MVC 14](#_Toc500706826)

[2 ПРАКТИЧНА ЧАСТИНА 15](#_Toc500706827)

[ВИСНОВКИ 16](#_Toc500706828)

[ЛІТЕРАТУРА 17](#_Toc500706829)

[ДОДАТКИ 18](#_Toc500706830)

# ВСТУП

На сьогоднішній день вважається кращою програма, яка потребує менших затрат на розробку та супроводження, а не найкоротша чи найшвидша. Найбільш перспективний напрям організаційного розвитку є напрямок розподілу праці, спрощення кожної функції або операції, нормування праці, поетапний контроль результатів, високий рівень стандартизації. Така організація праці у виробництві становить технологію виробничого процесу. Немає універсальної технології розробки програм, але більшість запропонованих технологій мають спільні риси, адже об’єкт праці – програмний продукт відрізняється продуктовою однаковістю, одним рівнем розвитку обчислювальної техніки та самої галузі програмування

Розробка програми на основі наперед підготовлених компонентів (технологія збірного програмування) є засобом переведення програмування на індустріальну основу, зменшення складності програмування реальних задач, продовження терміну використання окремих передових рішень, що реалізовані в модулях програми застосовуються і для процедурно орієнтованих програм.

Метою розрахунково-графічної роботи є ознайомлення з Spring Core, вивчення Spring MVC і стандартної архітектури веб-додатків, а також, розробка веб-додатку на основі Spring Boot.

# 1 ТЕОРЕТИЧНІ ВІДОМОСТІ

## 1.1 Apache Maven

Apache Maven — це засіб автоматизації роботи з програмними проектами, який спочатку використовувався для Java проектів. Використовується для управління (management) та складання (build) програм. Створений 2002 року Джейсоном ван Зилом. За принципами роботи кардинально відрізняється від Apache Ant, та має простіший вигляд щодо build-налаштувань, яке надається в форматі XML.

Ключовою особливістю Maven є його мережева готовність (network-ready).

Maven забезпечує підтримку побудови не просто перебираючи файли з цього репозиторію, але й завантажуючи назад артефакти у кінці побудови. Локальний кеш звантажених артефактів діє як первісний засіб синхронізації виходу проектів на локальній системі.

Maven базується на плаґін-архітектурі, що дозволяє зробити використання будь-якої програми контрольованим через стандартний вхід. Теоретично, це могло б дозволити будь-кому писати плаґіни для інтерфейсу з інструментами для побудови (компілятори, тестери тощо) для будь-якої мови. В дійсності, підтримка і використання для мов відмінних від Java були мінімальною.

## 1.2 XML

Розширювана мова розмітки (англ. Extensible Markup Language, скорочено XML) — запропонований консорціумом World Wide Web (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет. Є спрощеною підмножиною мови розмітки SGML. XML-документ складається із текстових знаків, і придатний до читання людиною.

Стандарт XML (англ. Recommendation, перше видання від 10 лютого 1998, останнє, четверте видання 29 вересня 2006) визначає набір базових лексичних та синтаксичних правил для побудови мови описання інформації шляхом застосування простих тегів. Цей формат достатньо гнучкий для того, аби бути придатним для застосування в різних галузях. Іншими словами, запропонований стандарт визначає метамову, на основі якої шляхом запровадження обмежень на структуру та зміст документів визначаються специфічні, предметно-орієнтовані мови розмітки даних. Прикладами мов, заснованих на XML, є: XSLT, XAML, XUL, RSS, MathML, GraphML, XHTML, SVG, а також XML Schema.

XML-документи повинні бути коректними та валідними. Перевіряється синтаксичним аналізатором.

XML-документи мають як фізичну, так і логічну структуру.

1. Фізична структура

* Сутності (англ. Entity). Головною сутністю є зміст документа. Інші можливі сутності вказуються за допомогою

Посилання на сутності (&назва; в самому документі, та, наприклад %назва; у визначені його типу) можуть слугувати в ролі позначень спеціальних символів, посилань на спеціальні символи (вказуючи коди символів &#десяткове;, або &#xшістнадцяткове;) або окремих документів чи фрагментів тексту.

* XML-декларація, в ній вказується версія XML, кодування та інша допоміжна інформація.
* Декларація типу документа може застосовуватись для того, аби додавати нові типи сутностей та визначати логічну структуру документа.

1. Логічна структура

XML-документ має ієрархічну логічну структуру, і може представлятись у вигляді дерева. Вузлами цього дерева можуть бути:

* елементи, фізична структура яких складається із:
* коректної пари відкриваючого та закриваючого тегів (<Назва-тега>) та (</Назва-тега>), або
* тега порожнього елемента (<Назва-тега />),
* Атрибути, що мають вигляд пар ключ-значення (назва атрибута=»значення атрибута») і знаходяться або у відкриваючому, або у порожньому тезі (подібно до метаданих),
* Вказівки щодо обробки документа (англ. Processing Instruction) (<?Обробник параметр ?>)
* Коментарі (<!–Текст коментаря 🡪)
* Текст, або у вигляді простого тексту, або фрагментів CDATA (<![CDATA[ довільний текст]]>).

XML-документ повинен мати лише один кореневий елемент. Решта елементів є піделементами цього кореневого елемента.

Деякі веб-браузери здатні безпосередньо відображати XML-документи. Це може досягатись шляхом застосування таблиці стилів (англ. Stylesheet). Вказані у таблиці стилів операції можуть призводити до перетворення XML-документа в інший, відмінний від XML формат.

## 1.3 JSON

JavaScript Object Notation (JSON, об’єктний запис JavaScript) — це формат обміну даними. Не зважаючи на те, що JSON не є строгою підмножиною JavaScript, він нагадує його синтаксис. Хоча багато мов програмування підтримують JSON, та він є особливо корисним для використання у програмах, що базуються на JavaScript, таких, як веб-сайти чи розширення браузерів.

JSON може представляти числа, булеві значення, строки, null, масиви (впорядковані послідовності значень) та об’єкти (пари ключ-значення), що включають у себе ці значення, чи інші обїекти та масиви. JSON не підтримує представлення більш складних даних, таких як функції, регулярні вирази, дати та інше. (Об’єкти Date за замовчуванням серіалізуються у строку, що містить дату у форматі ISO, отже інформація не є остаточно втраченою). Якщо JSON вам потрібен для відображення додаткових типів даних, трансформуйте значення у строку, яку ви пізніше будете десеріалізовувати.

Як і XML, JSON може зберігати ієрархічні дані на відміну від більш традиційного формату CSV. Багато інструментів дає можливість трансформіції із одного формату в інший, наприклад JSON to CSV Converter.

## 1.4 REST

REST (скор. Англ. Representational State Transfer, «передача репрезентативного стану») — підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів. Був описаний і популяризований 2000 року Роєм Філдінгом, одним із творців протоколу HTTP. В основі REST закладено принципи функціонування Всесвітньої павутини і, зокрема, можливості HTTP.

Дані повинні передаватися у вигляді невеликої кількості стандартних форматів (наприклад, HTML, XML, JSON). Будь-який REST протокол (HTTP в тому числі) повинен підтримувати кешування, не повинен залежати від мережевого прошарку, не повинен зберігати інформації про стан між парами «запит-відповідь». Стверджується, що такий підхід забезпечує масштабовність системи і дозволяє їй еволюціонувати з новими вимогами.

REST, як і кожен архітектурний стиль відповідає ряду архітектурних обмежень (англ. Architectural constraints). Це гібридний стиль який успадковує обмеження з інших архітектурних стилів.

*Клієнт-сервер*

Перша архітектура від якої він успадковує обмеження — це клієнт-серверна архітектура. Її обмеження вимагає розділення відповідальності між компонентами, які займаються зберіганням та оновленням даних (сервером), і тими компонентами, які займаються відображенням даних на інтерфейсі користувача та реагування на дії з цим інтерфейсом (клієнтом). Таке розділення дозволяє компонентам еволюціонувати незалежно.

*Відсутність стану*

Наступним обмеженням є те, що взаємодії між сервером та клієнтом не мають стану, тобто кожен запит містить всю необхідну інформацію для його обробки, і не покладається на те, що сервер знає щось з попереднього запиту.

Відсутність стану не означає що стану немає. Відсутність стану означає, що сервер не знає про стан клієнта. Коли клієнт, наприклад, запитує головну сторінку сайту, сервер відповідає на запитання і забуває про клієнта. Клієнт може залишити сторінку відкритою протягом кількох років, перш ніж натиснути посилання, і тоді сервер відповість на інший запит. Тим часом сервер може відповідати на запити інших клієнтів, або нічого не робити — для клієнта це не має значення.

Таким чином, наприклад дані про стан сесії (користувача, який аутентифікувався) зберігаються на клієнті, і передаються з кожним запитом. Це покращує масштабність, бо сервер після закінчення обробки запиту може звільнити всі ресурси, задіяні для цієї операції, без жодного ризику втратити цінну інформацію. Також спрощується моніторинг і зневадження, бо для того аби розібратись, що відбувається в певному запиті, досить подивитись лише на той один запит. Збільшується надійність, бо помилка в одному запиті не зачіпає інші.

Мінусом цього обмеження є те, що знижується продуктивність через те, що в кожен запит тепер доводиться додавати дані сесії з клієнта. Також збереження стану на різних клієнтах важче підтримувати, бо реалізації клієнтів можуть відрізнятись, тоді як середовище сервера повністю під контролем розробника.

*Кешування*

Додатковим обмеженням стилю REST є те, що системи, написані в цьому стилі, повинні підтримувати кешування, тобто дані, які передаються між сервером повинні містити інформацію про те, чи можна їх кешувати, і якщо можна, то як довго. Це дозволяє збільшувати продуктивність уникаючи зайвих запитів, але також зменшує надійність системи, через те, що дані в кеші можуть бути застарілі.

*Однорідний інтерфейс*

Всі компоненти в архітектурі REST підтримують однорідний інтерфейс. Це зменшує зв’язність між компонентами і сервісами які вони надають і дозволяє нескладно змінювати компоненти при потребі.

*Шари абстракції*

Наступним обмеженням для REST є поділ на шари абстракції. Кожен компонент потрапляє в якийсь шар, і спілкується лише з компонентами в шарі під ним або в шарі над ним. Обмежнення знання системи одним шаром зменшує складність компонентів.

*Запитування коду*

Останнім архітектурним обмеженням в REST є те що клієнти повинні дозволяти розширювати свою функціональність дозволяючи завантаження додаткового коду (code on demand) в формі аплетів чи скриптів. Це спрощує клієнти, дозволяючи не реалізовувати всі необхідні функції попередньо. Щоправда це необов’язкове обмеження, і якщо воно не дає переваг для конкретного застосування, то його не обов’язково реалізовувати. Наприклад, дозвіл завантаження стороннього коду може бути не бажаним з точки зору безпеки.

## 1.5 AJAX

AJAX (Asynchronous JavaScript And XML) — підхід до побудови користувацьких інтерфейсів веб-застосунків, за яких веб-сторінка, не перезавантажуючись, у фоновому режимі надсилає запити на сервер і сама звідти довантажує потрібні користувачу дані. AJAX — один з компонентів концепції DHTML.

AJAX-підхід до розробки, який призначений для користувачів інтерфейсів, комбінує кілька основних методів і прийомів:

* використання DHTML для динамічної зміни змісту сторінки.
* використання XMLHttpRequest для звернення до сервера «на льоту», не перезавантажуючи всю сторінку повністю
* альтернативний метод — динамічне підвантаження коду JavaScript в тег <SCRIPT> з використанням DOM, що здійснюється із використанням формату JSON)
* динамічне створення дочірніх фреймів

Використання цих підходів дозволяє створювати набагато зручніші веб-інтерфейси користувача на тих сторінках сайтів, де необхідна активна взаємодія з користувачем. AJAX — асинхронний, тому користувач може переглядати далі контент сайту, поки сервер все ще обробляє запит. Браузер не перезавантажує web-сторінку і дані посилаються на сервер без візуального підтвердження (крім випадків, коли ми самі захочемо показати процес з’єднання з сервером). Використання AJAX стало популярним після того, як компанія Google почала активно використовувати його при створенні своїх сайтів, таких як Gmail, Google Maps і Google Suggest. Створення цих сайтів підтвердило ефективність використання даного підходу.

## 1.6 Spring Framework

Spring Framework — це програмний каркас (фреймворк) з відкритим кодом та контейнера з підтримкою інверсії управління для платформи Java.

Основні особливості Spring Framework можуть бути використані будь-яким додатком Java, але є розширення для створення веб-додатків на платформі Java EE. Незважаючи на це, Spring Framework не нав’язує якоїсь конкретної моделі програмування.

Spring Framework складається з кількох модулів, які надають широкий спектр послуг.

1. Контейнер Інверсії управління: Конфігурація компонентів додатків і управління життєвим циклом об’єктів Java, здійснюється головним чином через інверсію управління.
2. Аспектно-орієнтоване програмування: дозволяє реалізувати наскрізні процедури.
3. Доступ до даних: робота з реляційною системою управління базами даних на платформі Java з використанням JDBC і об’єктно-реляційні відображення та інструментів з NoSQL баз даних.
4. Управління транзакціями: об’єднує кілька API, управління транзакціями та координує операції для Java-об’єктів.
5. Модель-Вигляд-Управління (Model-View-Controller): програмний каркас на основі HTTP сервлета, що забезпечує створення веб-додатків і веб-служб RESTful.
6. Аутентифікація і авторизація: налаштовувані процеси безпеки, які підтримують цілий ряд стандартів, протоколів, інструментів і практик за допомогою підпроекту Spring Security (колишня система безпеки АсеrІ для Spring).
7. Віддалене управління: конфігураційній вплив і управління Java-об’єктами для місцевої (локальної) або віддаленої конфігурації через JMX.
8. Тестування: підтримка класів для написання юніт-тестів та інтеграційних тестів.

## 1.7 Spring Boot

Spring Boot – це Java фреймворк, що є частиною Java-EE фреймоворку Spring Framework. Фактично, Spring Boot являє собою плагін для систем автоматичного збирання проектів Maven та Gradle. Він надає можливості для комфортної розробки та тестування Spring додатків. Окрім того, Spring Boot дозволяє упаковувати додаток в окремий jar-файл із вбудованим повноцінним контейнером Tomcat. Основною перевагою використання Spring Boot є зведення до мінімуму конфігураційних xml файлів, які зазвичай використовуються при розробці Spring додатку.

Можливості:

* створення повноцінних Spring додатків;
* вбудований Tomcat або Jetty (не потрібна установка WAR файлів);
* забезпечує початкові POMs для спрощення Maven конфігурації;
* автоматична конфігурірація Spring коли це можливо;
* забезпечує такими можливостями, як метрики, моніторинг станами і розширена конфігурація;
* абсолютно без генерації коду і без написання XML конфігурація;

## 1.8 MVC

Модель–вигляд–контролер (або Модель–представлення–контролер, англ. Model-view-controller, MVC) — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов’язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

У мові програмування Java концепція MVC підтримується на рівні стандартних класів-бібліотек. В результаті використання парадигми MVC програміст отримує в своє розпорядження могутню структуру об’єктів-компонентів, функції яких чітко розмежовані, що гарантує надійність і розширюваність системи, що розробляється.

# 2 ПРАКТИЧНА ЧАСТИНА

1) Створюємо головну сторінку додатка index.html (Додаток 1).

2) Підключаємо контролер DomoController та (Додаток 2).

3) До html сторінці підключаємо JavaScript javascript.js (Додаток 3) для виконання 2 завдань:

a) отримання загарбників (function getInvaders);

b) отримання карти (function getMap);

# ВИСНОВКИ

Мова програмування Java розроблялася як платформа для програмування різноманітних пристроїв. Проте стрімкий розвиток інтернет-технологій змусив творців java працювати і в цьому напрямку. Початково ставка робилася на аплети, проте через певні проблеми ранніх віртуальних машин джави із швидкодією та безпекою, аплети так і не змогли скласти конкуренцію іншим дещо подібним технологіям як то flash.

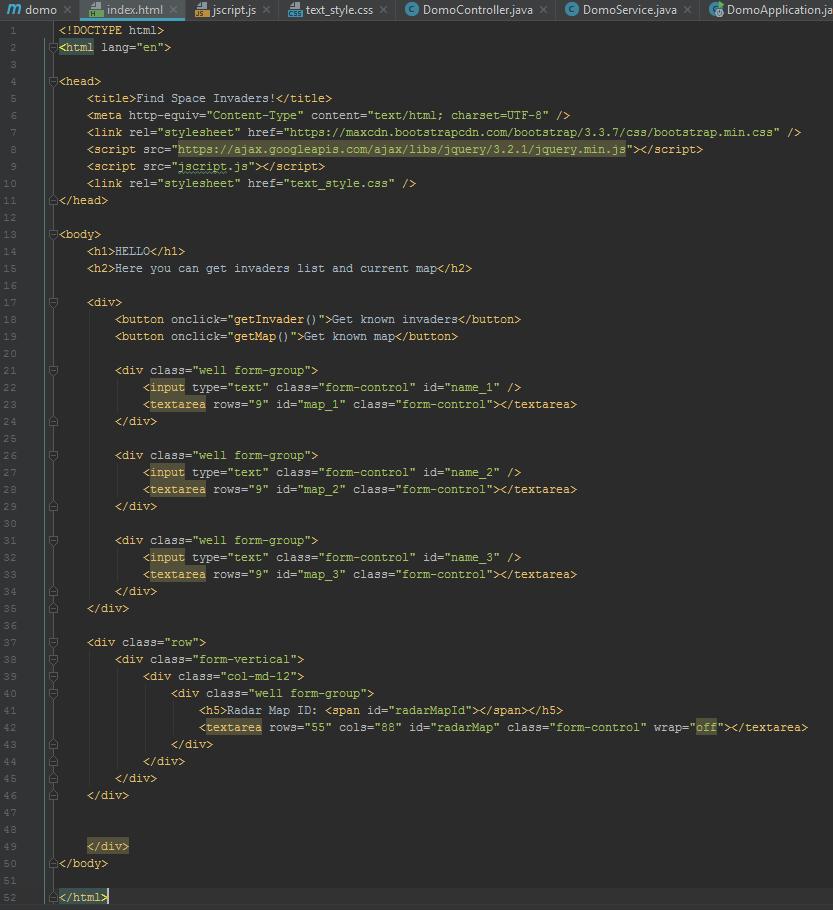
В ході даної розрахунково-графічної роботи, я ознайомився з Spring Core; вивив Spring MVC і стандартну архітектуру веб-додатків; розробив навчальний веб-додаток відповідно до шаблону MVС.

# ЛІТЕРАТУРА

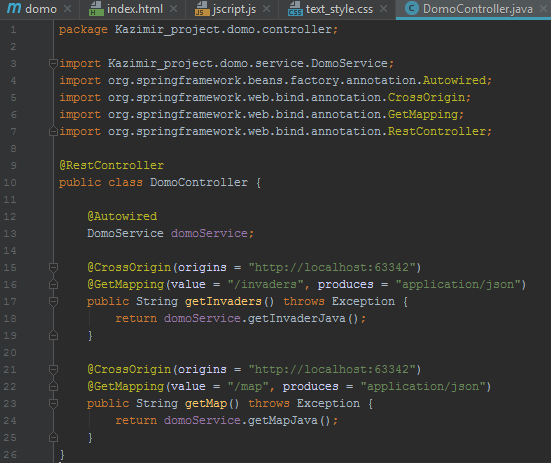
1. <https://maven.apache.org/docs/3.5.0/release-notes.html>
2. Apache Foundation. Building a Project with Maven. Apache Maven Documentation.
3. Fielding, Roy (2000). Architectural Styles and the Design of Network-based Software Architectures (Ph.D.) (англійською). University of California, Irvine[en]. Архів оригіналу за 2017-01-07. Процитовано 2009-02-20.
4. Richardson, Leonard; Amundsen, Mike; Ruby, Sam (2013). RESTful Web APIs (вид. First edition). O'Reilly. ISBN 978-1-4493-5806-8.
5. Кларенс Хо, Роб Харроп Spring 3 для профессионалов. — М. : «Вильямс», 2012. — 880 с. — ISBN 978-5-8459-1803-1.
6. Крейг Уоллс Spring в действии. — Третє. — М. : «Manning», 2014. — 624 с. — ISBN 9781617291203.
7. MFC Microsoft Foundation Classes - Бібліотека базових класів Microsoft. Бібліотека класів мови C++, призначена для розробки застосунків для Windows.

# ДОДАТКИ

Додаток 1



Додаток 2



Додаток 3

