# Курсовая работа

## по предмету «программирование на языке Java»

### на тему «Автоматизированное рабочее место сотрудника банка»

## студента 7 курса группы К14

## Мохова Михаила

Одесса 2019

### Введение

Автоматизированные системы обработки информации – это системы или комплексы систем, предназначенные для автоматизации обработки определенного вида информации. В данной работе рассматривается автоматизация банковской системы для сотрудников этого банка. Для своевременной и качественной переработки все возрастающих объемов поступающей в банки информации требуется применение все более совершенных технических и программных средств. Банковская система – это не только банки, но и кредитные учреждения, а также специализированные организации, не осуществляющие банковских операций, но обеспечивающих деятельность банков и кредитных учреждений (расчетно-кассовые центры и клиринговые центры, кредитные магазины, фирмы по аудиту банков и т. д. Для более эффективной работы банковских систем были созданы автоматизированные банковские системы. Целями автоматизированного рабочего места сотрудника банка являются:

1. Улучшение качества обслуживания клиентов;
2. Повышение квалификации персонала;
3. Сокращение времени ни операции со счетами при обслуживании клиентов;
4. Централизация всех банковских операций для повышения отказоустойчивости и надежности системы в целом;

Углубление процесса автоматизации функционирования банковских и прочих финансовых структур сопровождается совершенствованием технологии банковских операций и повышением уровней их управляемости. Современные информационные технологии позволяют координировать деятельность подразделений банков, расширить межбанковские связи, комплексно решать проблемы анализа банковской деятельности. Автоматизация информационных и других технологий банка содействуют улучшению качества обслуживания путем создания автоматизированных рабочих мест для специалистов всех уровней. В автоматизации банковских технологий находят место как простые программные продукты, позволяющие заполнять только несколько выходных форм для отчетности, так и достаточно интеллектуальные комплексы, решающие задачи управления банком. Предметом исследования в данной работе является отчетность сотрудника банка в целях упрощения процедуры регистрации новых клиентов банка, а так же счетов этих клиентов. В данной работе было разработано и развернуто приложение по принципу автоматизированного рабочего места для сотрудников банка, которые обслуживают клиентов.

Анализ специфики предметной области

**Анализ возможностей языка java**

Java - язык программирования, разработанный компанией Sun Microsystems. Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) независимо от компьютерной архитектуры. Дата официального выпуска - 23 мая 1995 года. Сегодня технология Java предоставляет средства для превращения статических Web-страниц в интерактивные динамические документы и для создания распределенных не зависящих от платформы приложений.

Программы на Java транслируются в байт-код, выполняемый виртуальной машиной Java (JVM) - программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинство подобного способа выполнения программ - в полной независимости байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности благодаря тому, что исполнение программы полностью контролируется виртуальной машиной. Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером) вызывают немедленное прерывание.

Часто к недостаткам концепции виртуальной машины относят то, что исполнение байт-кода виртуальной машиной может снижать производительность программ и алгоритмов, реализованных на языке Java. В последнее время был внесен ряд усовершенствований, которые несколько увеличили скорость выполнения программ на Java:

* применение технологии трансляции байт-кода в машинный код непосредственно во время работы программы (JIT-технология) с возможностью сохранения версий класса в машинном коде,
* широкое использование платформенно-ориентированного кода (native-код) в стандартных библиотеках,
* аппаратные средства, обеспечивающие ускоренную обработку байт-кода (например, технология Jazelle, поддерживаемая некоторыми процессорами фирмы ARM).

Основные возможности:

* автоматическое управление памятью;
* расширенные возможности обработки исключительных ситуаций;
* богатый набор средств фильтрации ввода/вывода;
* набор стандартных коллекций, таких как массив <#"justify">§на уровне отдельных SQL-запросов - на основе JDBC, SQLJ;
* на уровне концепции объектов, обладающих способностью к хранению в базе данных - на основе Java Data Objects и Java Persistence API;
* поддержка шаблонов (начиная с версии 1.5);
* параллельное выполнение программ.

Одним из интереснейших аспектов Java программирования являются Java-апплеты. Апплет - прикладная программа на Java в форме байт-кода. Java-апплеты выполняются в веб-браузере с использованием виртуальной Java машины (JVM), или в Sun'sAppletViewer, автономном инструменте для тестирования апплетов. Java-апплеты были внедрены в первой версии языка Java в 1995. Java-апплеты обычно пишутся на языке программирования Java, но могут быть написаны и на других языках, которые компилируются в байт-код Java, таких, как Jython.

Апплеты используются для предоставления интерактивных возможностей веб-приложений, которые не могут быть предоставлены HTML. Так как байт-код Java платформо-независим, то Java-апплеты могут выполняться с помощью плагинов браузерами многих платформ, включая Microsoft Windows, UNIX, Apple Mac OS и GNU/Linux.

***JSP***

JSP (JavaServer Pages) — технология, позволяющая веб-разработчикам динамически генерировать HTML, XML и другие веб-страницы. Является составной частью единой технологии создания бизнес-приложений J2EE1. Технология позволяет внедрять Java-код, а также EL (expression language) в статичное содержимое страницы. Также могут использоваться библиотеки JSP тегов для внедрения их в JSP-cтраницы. Страницы компилируются JSP-компилятором в сервлеты, представляющие собой Java-классы, которые выполняются на сервере. Сервлеты также могут быть написаны разработчиком, не используя JSP-страницы. Эти технологии могут дополнять друг друга.

JSP — одна из высокопроизводительных технологий, так как весь код страницы транслируется в java-код сервлета с помощью компилятора JSP страниц Jasper, и затем компилируется в байт-код виртуальной машины java (JVM). Сервлет-контейнеры (Tomcat), способные исполнять JSP страницы, написаны на платформонезависимом языке Java, который может работать под различными операционными системами и платформами. Сервлет-контейнеры могут работать как полноценные самостоятельные веб-серверы, работать поставщиком страниц для другого веб-сервера или интегрироваться в J2EE сервер приложений. Web-контейнер обеспечивает обмен данными между сервлетом и клиентами, берет на себя выполнение таких функций, как создание программной среды для функционирующего сервлета, идентификацию и авторизацию клиентов, организацию сессии для каждого из них.

***Thymeleaf***

Thymeleaf - это современный серверный шаблонизатор Java для веб-приложений и автономных сред.

Основная цель Thymeleaf - привнести элегантные естественные шаблоны рабочий процесс разработки - HTML, который может корректно отображаться в браузерах, а также работать как статические прототипы, обеспечивая более тесное сотрудничество в группах разработчиков.

Благодаря модулям для Spring Framework, множеству интеграций c инструментами и возможности подключения собственных функций, Thymeleaf идеально подходит для современной веб-разработки на HTML5 в связке с JVM.

Thymeleaf представляет из себя библиотеку, написанную на языке java и поддерживает работу со множеством XML- подобных форматов, таких как:

* HTML (HTML5, XHTML 1.0/1.1, HTML 4);
* XML
* TEXT (сплошной текст)
* JAVASCRIPT (файлы .js)
* CSS (файлы .css)

***Spring***

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET.

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.1.2.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет бо́льшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

***Spring boot***

Из-за громоздкой конфигурации зависимостей настройка Spring для корпоративных приложений превратилась в весьма утомительное и подверженное ошибкам занятие. Особенно это относится к приложениям, которые используют также несколько сторонних библиотек.

Каждый раз, создавая очередное корпоративное Java-приложение на основе Spring, вам необходимо повторять одни и те же рутинные шаги по его настройке:

* В зависимости от типа создаваемого приложения (Spring MVC, Spring JDBC, Spring ORM и т.д.) импортировать необходимые Spring-модули
* Импортировать библиотеку web-контейнеров (в случае web-приложений)
* Импортировать необходимые сторонние библиотеки (например, Hibernate, Jackson), при этом вы должны искать версии, совместимые с указанной версией Spring
* Конфигурировать компоненты DAO, такие, как: источники данных, управление транзакциями и т.д.
* Конфигурировать компоненты web-слоя, такие, как: диспетчер ресурсов, view resolver
* Определить класс, который загрузит все необходимые конфигурации

Spring Boot — это полезный проект, целью которого является упрощение создания приложений на основе Spring. Он позволяет наиболее простым способом создать web-приложение, требуя от разработчиков минимум усилий по его настройке и написанию кода. Он обладает большим функционалом, но его наиболее значимыми особенностями являются: управление зависимостями, автоматическая конфигурация и встроенные контейнеры сервлетов

Чтобы ускорить процесс управления зависимостями, Spring Boot неявно упаковывает необходимые сторонние зависимости для каждого типа приложения на основе Spring и предоставляет их разработчику посредством так называемых starter-пакетов (spring-boot-starter-web, spring-boot-starter-data-jpa и т. д.)

Другими словами, Spring Boot собирает все общие зависимости и определяет их в одном месте, что позволяет разработчикам просто использовать их, вместо того, чтобы изобретать колесо каждый раз, когда они создают новое приложение

*Starter*-пакеты представляют собой набор удобных дескрипторов зависимостей, которые можно включить в свое приложение. Это позволит получить универсальное решение для всех, связанных со Spring технологий, избавляя программиста от лишнего поиска примеров кода и загрузки из них требуемых дескрипторов зависимостей (пример таких дескрипторов и стартовых пакетов будет показан ниже)

Например, если вы хотите начать использовать Spring Data JPA для доступа к базе данных, просто включите в свой проект зависимость *spring-boot-starter-data-jpa* и все будет готово (вам не придется искать совместимые драйверы баз данных и библиотеки Hibernate)

***Spring MVC***

Фреймворк Spring Web model-view-controller (MVC) или по нашему модель-представление-контроллер построен вокруг DispatcherServlet, который распределяет запросы по обработчикам. В нём настраивается мэппинг запросов, локали, временные зоны и многое другое. Обработчик по умолчанию строится на аннотациях *@Controller* и *@RequestMapping*, которые предоставляют широкий набор гибких методов для обработки запросов. После версии Spring 3.0. механизм *@Controller* так же позволяет создавать RESTful веб сайты и приложения, используя аннотацию *@PathVariable* и другие возможности.

В Spring Web MVC вы можете использовать любой объект в качестве команды или объекта с обратной связью; вам нет необходимости реализовывать какой-либо специальный интерфейс фреймворка или базовый класс. Связывание данных в Spring является очень гибким: например, оно рассматривает несоответствие типов как ошибки валидации и поэтому это может быть обработано в приложении, а не в качестве системных ошибок. Таким образом, вам не нужно дублировать свойства бизнес-объектов, в качестве простых нетипизированных строк для ваших объектов форм. Поэтому можно легко обрабатывать неправильные подтверждения (сабмиты) или правильно конвертировать их в строки. Вместо этого, желательно связывать такие объекты напрямую с объектами бизнес логики.

#### *DispatcherServlet*

Spring MVC построен вокруг центрального сервлета, который распределяет запросы по контроллерам, а также предоставляет другие широкие возможности при разработке веб приложений. На самом деле DispatcherServlet — полностью интегрированный сервлет в Spring IoC контейнер и таким образом получает доступ ко всем возможностям Spring.

***Hibernate***

На сегодняшний день, очень сложно найти разработчика, который бы при программировании приложений, не работал с базами данных. Потому, для упрощения взаимодействия с СУБД начали писаться различные фреймворки. JDBC (Java DataBase Connectivity) является платформенно-независимым промышленным стандартом взаимо действия Java-приложений с различными СУБД, реализованным в виде пакета java.sql, входящего в состав Java SE. После этого стали появляться ORM-решения, которые упрощали взаимодействия с базой данных и объектами Java.

ORM (Object-relational mapping) позволяет заносить, получать, изымать данные из базы данных. ORM-решением для языка Java, является технология Hibernate, которая помогает сэкономить значительную часть времени при построении запросов и извлечении данных. Однако, существует множество споров на тему, что лучше использовать при разработке проектов: JDBC или Hibernate. Данное сравнение нельзя назвать верным, по скольку Hibernate - это оболочка для JDBC, которая обладает огромным набором функций и возможностей. Hibernate имеет целый ряд преимуществ, которые являются решающими при выборе данной технологии:

* устраняет множество спагетти кода (повторяющегося), который постоянно преследует разработчика при работе с JDBC. Скрывает от разработчика множество кода, необходимого для управления ресурсами и позволяет сосредоточиться на бизнес логике.
* поддерживает XML так же как и JPA аннотации, что позволяет сделать реализацию кода независимой.
* предоставляет собственный мощный язык запросов (HQL), который похож на SQL. Стоит отметить, что HQL полностью объектно-ориентирован и понимает такие принципы, как наследование, полиморфизм и ассоциации (связи).
* Hibernate — широко распространенный open source проект. Благодаря этому доступны тысячи открытых статей, примеров, а так же документации по использованию фреймворка.
* Hibernate поддерживает ленивую инициализацию используя proxy объекты и выполняет запросы к базе данных только по необходимости.
* поддерживает разные уровни cache, а, следовательно, может повысить производительность.
* важно, что Hibernate может использовать чистый SQL, а, значит, поддерживает возможность оптимизации запросов и работы с любым сторонним поставщиком БД и его возможностями.
* лёгкость разработки: разработчик может не знать специфики базы данных, с которой работает.

***Maven***

Maven - это инструмент для сборки Java проекта: компиляции, создания jar-файлов, создания дистрибутива программы, генерации документации. Простые проекты можно собрать в командной строке. Если собирать большие проекты с командной строки, то команда для сборки будет очень длинной, поэтому её иногда записывают в bat/sh скрипт. Но такие скрипты зависят от платформы. Для того чтобы избавиться от этой зависимости и упростить написание скрипта используют инструменты для сборки проекта.

Для платформы Java существуют два основных инструмента для сборки: Ant и Maven.

Основные преимущества Maven:

* Независимость от OS. Сборка проекта происходит в любой операционной системе. Файл проекта один и тот же.
* Управление зависимостями. Редко какие проекты пишутся без использования сторонних библиотек(зависимостей). Эти сторонние библиотеки зачастую тоже в свою очередь используют библиотеки разных версий. Мавен позволяет управлять такими сложными зависимостями. Что позволяет разрешать конфликты версий и в случае необходимости легко переходить на новые версии библиотек.
* Возможна сборка из командной строки. Такое часто необходимо для автоматической сборки проекта на сервере (Continuous Integration).
* Хорошая интеграция со средами разработки. Основные среды разработки на java легко открывают проекты которые собираются c помощью maven. При этом зачастую проект настраивать не нужно - он сразу готов к дальнейшей разработке.   
  Как следствие - если с проектом работают в разных средах разработки, то maven удобный способ хранения настроек. Настроечный файл среды разработки и для сборки один и тот же - меньше дублирования данных и соответственно ошибок.
* Декларативное описание проекта.

***Описание существующих и преимуществ выбранной СУБД***

Для данной работы была выбрана реляционная база данных MySQL, так как она хорошо подходит для хранения специфической информации при работе с банковскими системами.

MySQL - это самая распространенная полноценная серверная СУБД. MySQL очень функциональная, свободно распространяемая СУБД, которая успешно работает с различными сайтами и веб приложениями. Обучиться использованию этой СУБД довольно просто, так как на просторах интернета вы легко найдете большее количество информации.

Несмотря на то, что в ней не реализован весь SQL функционал, MySQL предлагает довольно много инструментов для разработки приложений. Так как это серверная СУБД, приложения для доступа к данным, в отличии от SQLite работают со службами MySQL.

Преимущества:

* Простота в работе - установить MySQL довольно просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД
* Богатый функционал - MySQL поддерживает большинство функционала SQL.
* Безопасность - большое количество функций обеспечивающих безопасность, которые поддерживается по умолчанию
* Масштабируемость - MySQL легко работает с большими объемами данных и легко масштабируется
* Скорость - упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки:

* Известные ограничения - по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях.
* Проблемы с надежностью - из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по надежности.
* Медленная разработка - Хотя MySQL технически открытое ПО, существуют жалобы на процесс разработки. Стоит заметить, что существуют другие довольно успешные СУБД созданные на базе MySQL, например MariaDB.

Выводы:

В данной главе были рассмотрены средства и технологии, использованные при разработке информационной системы, проведен сравнительный анализ, выявлены достоинства и недостатки по сравнению с аналогами.

Проектирование информационной системы «Автоматизированное рабочее место сотрудника банка»