

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«МЭИ»

Институт Информационных и Вычислительных Технологий  
Кафедра Вычислительных Технологий

Курсовой проект по дисциплине «Проектирование баз данных»  
**Проектирование приложений трехзвенной архитектуры  
клиент/сервер и работа с ней в среде Windows**

	<b>Выполнил</b>
<b>Студент:</b>	Швец Г. В.
<b>Группа:</b>	А-06М-23
	<b>Проверил</b>
<b>Преподаватель:</b>	Бородин Г. А.
<b>Оценка:</b>	
<b>Дата:</b>	
<b>Подпись:</b>	

Москва, 2024

## СОДЕРЖАНИЕ

1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ .....	4
1.1 Общие сведения .....	4
1.2 Назначение и цели создания системы.....	6
1.3 Характеристики объекта автоматизации .....	6
1.4 Требования к системе.....	6
1.5 Состав и содержание работ по созданию (развитию) системы .....	7
1.6 Порядок контроля и приёма системы .....	8
1.7 Требования к документированию .....	8
1.8 Источники разработки .....	9
2. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ .....	9
2.1 Технология JDBC.....	9
2.2 Технология Java Servlet.....	11
2.3 Выбор средств реализации приложения .....	13
3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ.....	14
3.1 Назначение базы данных .....	14
3.2 Структура проектируемой базы данных .....	14
3.3 Разработка таблиц, их полей и ключей .....	15
3.4 Разработка связей таблиц, ссылочная целостность .....	18
3.5 Хранимая процедура .....	19
3.6 Хранимая функция .....	20
3.7 Представление.....	21
3.8 Оценка размера базы данных .....	22
4. СОЗДАНИЕ СЕРВЕРА ПРИЛОЖЕНИЙ .....	22
4.1 Разворачивание сервера приложений с помощью Jetty.....	22
4.2 Реализация сервлетов приложения .....	25
4.3 Транзакции .....	28
5. СОЗДАНИЕ ПРИЛОЖЕНИЯ КЛИЕНТА .....	29
5.1 Редактирование и загрузка изображения в БД .....	31
5.2 Хранимые процедуры и функции .....	31
5.3 Графический отчет в Excel .....	33
5.4 Табличный отчет.....	34
6. КОМПЛЕКС ПОСТАВКИ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ .....	34

СПИСОК ЛИТЕРАТУРЫ.....	35
ПРИЛОЖЕНИЕ .....	37

# **1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

## **1.1 Общие сведения**

### **1.1.1 Наименование системы:**

Система, основанная на приложении-клиенте, сервере приложений и сервере СУБД трёхзвенной архитектуры клиент/сервер для обеспечения доступа к данным.

### **1.1.2 Задание:**

1. Разработать реляционную базу данных, состоящую из 2-3 связанных таблиц (1-To-Many) и 1-2 ссылочных таблиц. Включить поля типа Memo и OLE-поля (BLOB поля).
2. Создать табличный и графический отчёты, последний получаемый, например, с использованием OLE-связи с Excel.
3. Организовать ввод и редактирование OLE-полей с помощью OLE-технологии, например, с использованием в качестве OLE-сервера PaintBrush.
4. Разработать многостраничную экранную форму, позволяющую редактировать данные в таблицах базы данных и OLE-связи.
5. Работу с базой данных организовать с помощью триггеров, запросов, хранимых процедур и функций, курсоров, представлений, размещаемых в базе данных.

### **1.1.3 Средства проектирования и разработки:**

1. Сервер базы данных – MS SQL Server 2008 R2.
2. Среда разработки – Eclipse IDE for Java EE (Java).
3. Создание отчётов – Jasper Reports.
4. Дополнительные пакеты – MS Office.
5. Технологии реализации трёхзвенных приложений – Servlet.
6. Доступ к базе данных с использованием: JDBC.

#### **1.1.4 Наименование предприятий (объединений) разработчика и заказчика (пользователя) системы и их реквизиты:**

Система создаётся согласно учебному плану НИУ «МЭИ» в рамках курса «Проектирование баз данных» в весеннем семестре 2023/2024 уч. года студентом группы А-06м-23 Швецом Г. В.

#### **1.1.5 Перечень документов, на основании которых создаётся система, кем и когда утверждены эти документы:**

Техническое задание (ТЗ) утверждается профессором кафедры В «НИУ «МЭИ» д.т.н. Бородиным Г.А.

#### **1.1.6 Плановые сроки начала и окончания работ по созданию системы:**

Срок начала работы: 18.02.2024. Срок окончания работы: июнь 2024.

#### **1.1.7 Сведения об источнике и порядке финансирования:**

Финансирование отсутствует.

#### **1.1.8 Порядок оформления и предъявления заказчику результатов работ по созданию системы, по изготовлению и наладке отдельных средств и программно-технических комплексов системы:**

1. На 2-3 учебных неделях необходимо утвердить техническое задание на выполнение курсового проекта с учётом указанных типовых требований к проекту.

2. За неделю до защиты курсового проекта разработчик может сдать пояснительную записку на проверку консультанту для проверки выполнения требований к содержанию пояснительной записки к типовому проекту и устранений грубых ошибок.

3. На защите курсового проекта необходимо уметь:

- Установить клиентское приложение, сервер-приложений и сервер СУБД на различные компьютеры.

- Произвести установку, регистрацию и настройку вспомогательных файлов, утилит, библиотек, сервера СУБД и т. п.
- Продемонстрировать параллельную работу не менее двух экземпляров приложений-клиентов с разных рабочих мест.
- Аргументировано отстаивать выбранные решения.

## **1.2 Назначение и цели создания системы**

### **1.2.1 Назначение системы:**

Разработка приложений для доступа к базе данных

### **1.2.2 Цели создания системы:**

Целью курсового проекта является освоение навыков разработки клиентов, серверов приложений, серверов СУБД, трёхзвенных приложений, а также получение навыков взаимодействия приложений с использованием технологии OLE.

## **1.3 Характеристики объекта автоматизации**

### **1.3.1 Краткие сведения об объекте автоматизации:**

Объектом автоматизации являются процессы обеспечения доступа, редактирования и сохранения данных, хранящихся в базе данных.

## **1.4 Требования к системе**

### **1.4.1 Требования к численности и квалификации пользователей:**

1. По количеству ЭВМ, имеющих доступ в разработанной системе.
2. Базовые навыки работы в ОС Windows.

### **1.4.2 Технические требования и требования к ПО:**

1. Для сервера баз данных: Microsoft SQL Server 2008 R2. Права администратора на время первоначальной установки и настройки.
2. Для сервера приложений: JDK 8, Microsoft .NET Framework 4.5 или выше, Windows Management Framework 5.1 или выше, Windows PowerShell 5.1 или выше. Права администратора на время первоначальной установки и настройки.
3. Для клиентских компьютеров: JDK 8, Microsoft .NET Framework 4.5 или выше, Windows Management Framework 5.1 или выше, Windows PowerShell 5.1 или выше. Права администратора на время первоначальной установки и настройки.

### **1.4.3 Требования к функциям, выполняемым системой:**

1. Подключение двух приложений-клиентов с двух компьютеров.
2. Редактирование одной записи двумя приложениями-клиентами.
3. Каскадное удаление записей из подчинённой таблицы при удалении записи из главной таблицы.
4. Добавление записи в подчинённую таблицу, не имеющей соответствующей записи в главной.
5. Демонстрация сохранения и визуализации OLE-объектов.
6. Демонстрация табличного и графического (в Excel) отчётов.
7. Демонстрация работы представления.
8. Демонстрация работы хранимой процедуры.
9. Демонстрация работы хранимой функции.
10. Демонстрация работы пользователя с ограниченным доступом.
11. Очистка компьютеров от установленной программы.

## **1.5 Состав и содержание работ по созданию (развитию) системы**

### **1.5.1 Стадии и этапы работы:**

1. Сдача технического задания.
2. Выполнение работы.
3. Подготовка и сдача пояснительной записки.
4. Защита работы.

#### **1.5.2 Сроки выполнения:**

Февраль – июнь 2024 года.

#### **1.5.3 Исполнитель работы:**

Студент кафедры ВТ группы А-06м-23 Швец Г. В.

#### **1.5.4 Вид и порядок проведения экспертизы документации:**

Экспертиза проводится профессором кафедры ВТ НИУ «МЭИ» д.т.н. Бородиным Г. А.

### **1.6 Порядок контроля и приёма системы**

#### **1.6.1 Виды, состав, объём и методы испытаний системы и её составных частей:**

Определяются преподавателем.

#### **1.6.2 Сроки проведения:**

Июнь 2024 года.

### **1.7 Требования к документированию**

#### **1.7.1 Перечень подлежащих разработке компонентов и видов документов:**

2. Техническое задание.
3. Пояснительная записка.



## **1.8 Источники разработки**

Техническое задание разработано на основе ГОСТ 34.602-2020 «Техническое задание на создание автоматизированной системы».

## **2. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ**

Для разработки приложения с трёхзвенной архитектурой, включающее приложение-клиент, сервер приложений и систему управления базами данных (СУБД), необходимо обеспечить надежное и эффективное взаимодействие между этими компонентами. На основе технического задания, включающего создание реляционной базы данных, создание отчетов и управление данными через экранные формы, были выбраны соответствующие технологии для реализации поставленных задач.

В данной работе основное внимание будет уделено двум ключевым технологиям, используемым в разработке: JDBC и Java Servlet.

### **2.1 Технология JDBC**

Одним из ключевых элементов, обеспечивающих взаимодействие, является использование платформенно независимого промышленного стандарта JDBC (Java Database Connectivity) [5].

JDBC – это стандартный API (Application Programming Interface), разработанный для взаимодействия Java-приложений с различными СУБД. Он позволяет осуществлять соединение с базой данных, выполнять запросы и управлять результатами запросов на уровне приложения. JDBC входит в состав Java SE (Standard Edition) и реализован в виде пакета ‘java.sql’ [4].

Основные преимущества использования JDBC в проекте включают:

- Платформенная независимость: JDBC представляет единый интерфейс

для работы с различными СУБД, что позволяет разработчику использовать одну и ту же программу для доступа к разным базам данных без необходимости вносить изменения в код.

- Поддержка SQL-запросов: JDBC обеспечивает выполнение стандартных SQL-запросов, включая SELECT, INSERT, UPDATE и DELETE, что позволяет эффективно управлять данными в базе.
- Управление транзакциями: JDBC позволяет управление транзакциями, позволяя выполнить несколько операций с базой данных как одно целое, что важно для сохранения целостности данных.
- Обработка результатов запросов: JDBC предоставляет возможности для удобной обработки результатов запросов через интерфейсы 'ResultSet', позволяя извлекать данные из базы и использовать их в приложении.
- Расширяемость и поддержка драйверов: JDBC поддерживает широкий спектр драйверов, что позволяет подключаться к любым популярным СУБД, таким как MySQL, PostgreSQL, Oracle, MS SQL Server и другим.

Принцип работы JDBC заключается в использовании драйвера, который выступает посредником между Java-приложением и конкретной СУБД. Процесс взаимодействия с базой данных с использованием JDBC включает следующие основные шаги:

1. Загрузка драйвера: загрузка соответствующего драйвера для СУБД с помощью вызова 'Class.forName()'.
2. Установка соединения: установка соединения с базой данных с использованием класса 'DriverManager' и метода 'getConnection()', который принимает URL базы данных, имя пользователя и пароль.
3. Создание и выполнение запросов: создание объекта 'Statement' или 'PreparedStatement' для выполнения SQL-запросов.
4. Обработка результатов: обработка результатов запросов с использованием объекта 'ResultSet'.
5. Закрытие ресурсов: закрытие всех отдельных ресурсов, таких как 'ResultSet', 'Statement' и 'Connection', чтобы освободить ресурсы и

предотвратить утечки памяти.

Таким образом, использование JDBC в проекте позволяет обеспечить гибкость и надежность взаимодействия приложения с базой данных, что является важным аспектом при разработке современных информационных систем с трёхзвенной архитектурой.

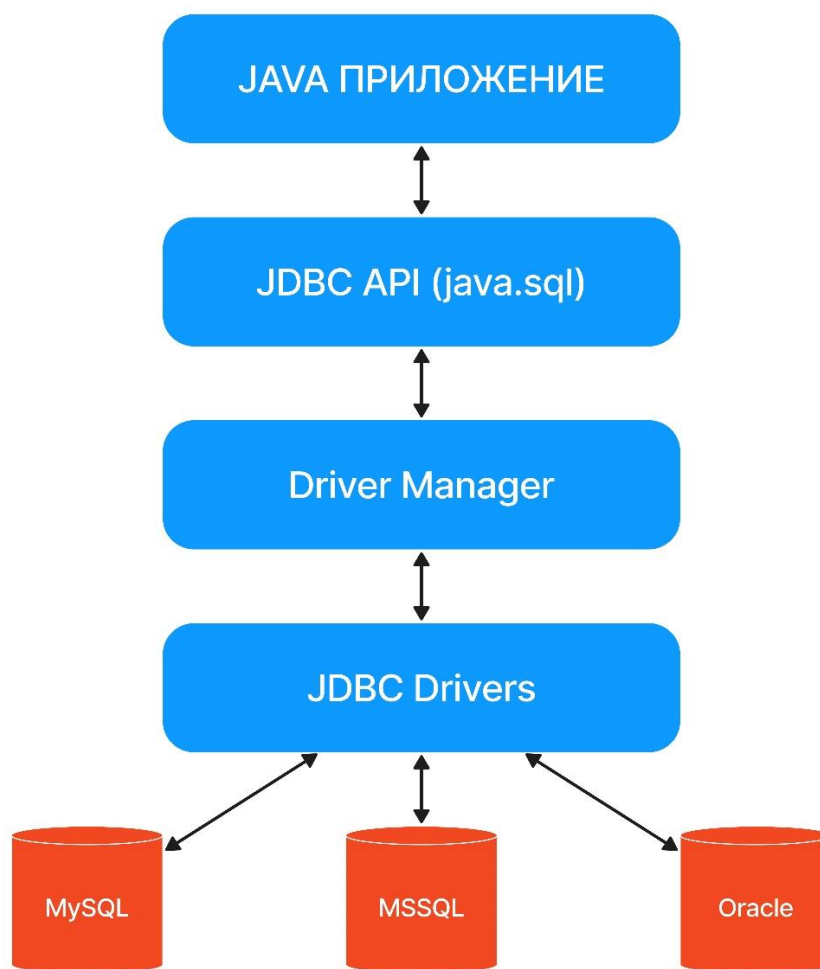


Рисунок 1 - Архитектура использования технологии JDBC

## 2.2 Технология Java Servlet

### 2.2.1 Технология сервлетов:

Java Servlet – это серверные компоненты, которые обрабатывают запросы и генерируют ответы в веб-приложениях, работающих на сервере. Servlets

являются частью платформы Java EE и реализуют спецификацию сервлетов, что позволяет создавать динамические веб-приложения с использованием языка программирования Java.

Основные преимущества использования сервлетов включают:

- **Эффективность и производительность:** сервлеты работают в рамках одного процесса сервера, что обеспечивает высокую производительность и низкие накладные расходы.
- **Мощные возможности:** сервлеты могут использовать весь потенциал языка Java, включая библиотеки и фреймворки, что позволяет создавать сложные и высокопроизводительные веб-приложения.
- **Простота и удобство:** создание и развертывание сервлетов относительно просты, что позволяет быстро разрабатывать и внедрять веб-приложения.

Процесс работы с сервлетами включает следующие шаги:

1. **Создание сервлета:** написание класса сервлета, который расширяет 'HttpServlet' и переопределяет методы 'doGet', 'doPost' или другие методы для обработки HTTP-запросов.
2. **Настройка и развертывание:** настройка сервлета в файле конфигурации (web.xml) или с использованием аннотаций и развертывание веб-приложения на сервере приложений.
3. **Обработка запросов:** сервлеты обрабатывают входящие HTTP-запросы, выполняют бизнес-логику и генерируют HTTP-ответы, которые отправляются обратно клиенту.

Использование технологии Java Servlet в проекте позволяет создать эффективные и безопасные веб-приложения, обеспечивая динамическое взаимодействие между клиентами и сервером приложений.

### **2.2.2 Контейнер сервлетов Jetty:**

Jetty – это легковесный контейнер сервлетов, который позволяет запускать веб-приложения на основе сервлетов. Jetty поддерживает спецификации Java Servlet и JavaServer Pages (JSP), обеспечивая быстрый и надежный серверный

движок для разработки и тестирования веб-приложений [6].

Основные преимущества использования Jetty:

- Легковесность: Jetty имеет небольшой размер и не требует значительных ресурсов, что делает его идеальным для разработки и тестирования.
- Легкость интеграции: Jetty легко интегрируется с различными инструментами и фреймворками, что упрощает процесс разработки.
- Гибкость и расширяемость: Jetty поддерживает различные конфигурации и расширения, позволяя адаптировать его под конкретные нужды проекта.
- Поддержка современного веба: Jetty поддерживает Servlet, WebSockets, HTTP/2 и другие современные веб-технологии, обеспечивая актуальность и производительность приложений.

В нашем проекте Jetty используется для запуска сервлетов и управления их жизненным циклом.

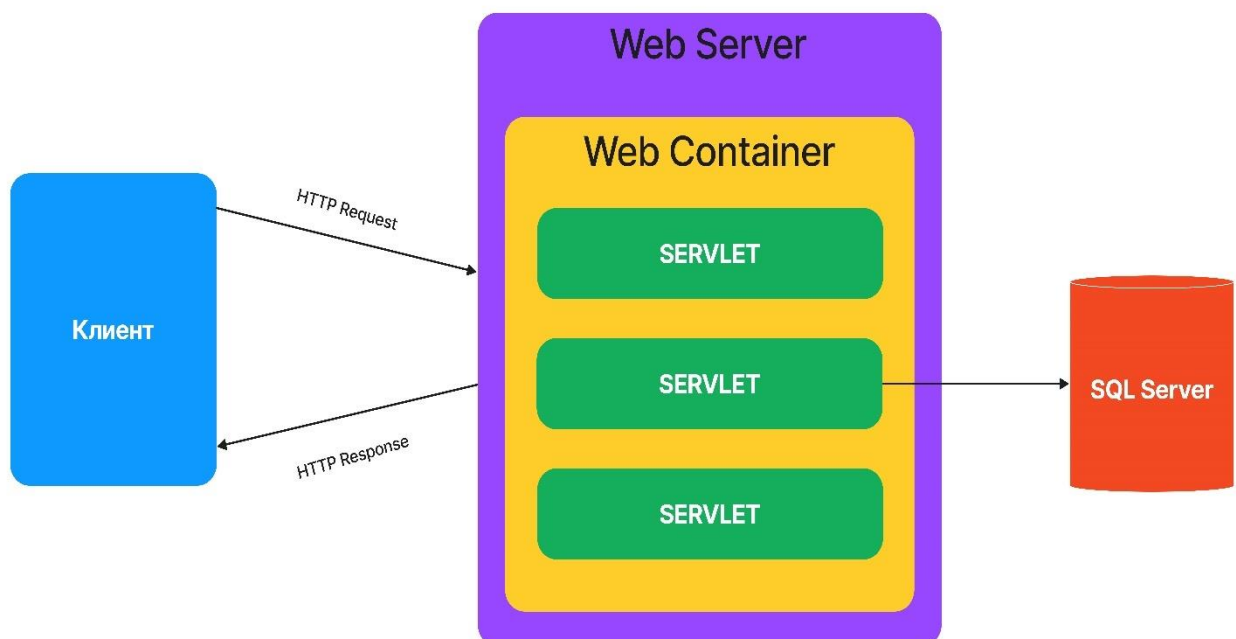


Рисунок 2 - Общая архитектура технологии Java Servlet

## 2.3 Выбор средств реализации приложения

В ходе разработки приложения использовалась среда разработки Eclipse IDE для Java, а также инструмент для управления зависимости и автоматизации

сборки проекта Apache Maven.

В качестве сервера приложений и контейнера сервлетов использовался Jetty, как описано выше.

Для тестирования веб-интерфейса приложения использовался веб-браузер Google Chrome.

Сервер БД – компьютер с установленным Microsoft SQL Server 2008 R2.

### **3. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

#### **3.1 Назначение базы данных**

Проектируемая база данных служит для хранения и управления информацией о национальных музеях и выставках, которые в них проходят. А также она содержит информацию о произведениях изобразительного искусства, которые представлены на соответствующих выставках. Поскольку одним из пожеланий заказчика было придерживаться русской культуры, то речь пойдет именно о национальных музеях.

#### **3.2 Структура проектируемой базы данных**

Согласно назначению базы данных она должна включать в себя следующие 4 таблицы:

- museums
- exhibitions
- art\_objects
- art\_exhibitions

На рисунке 3 представлена модель разработанной базы данных:

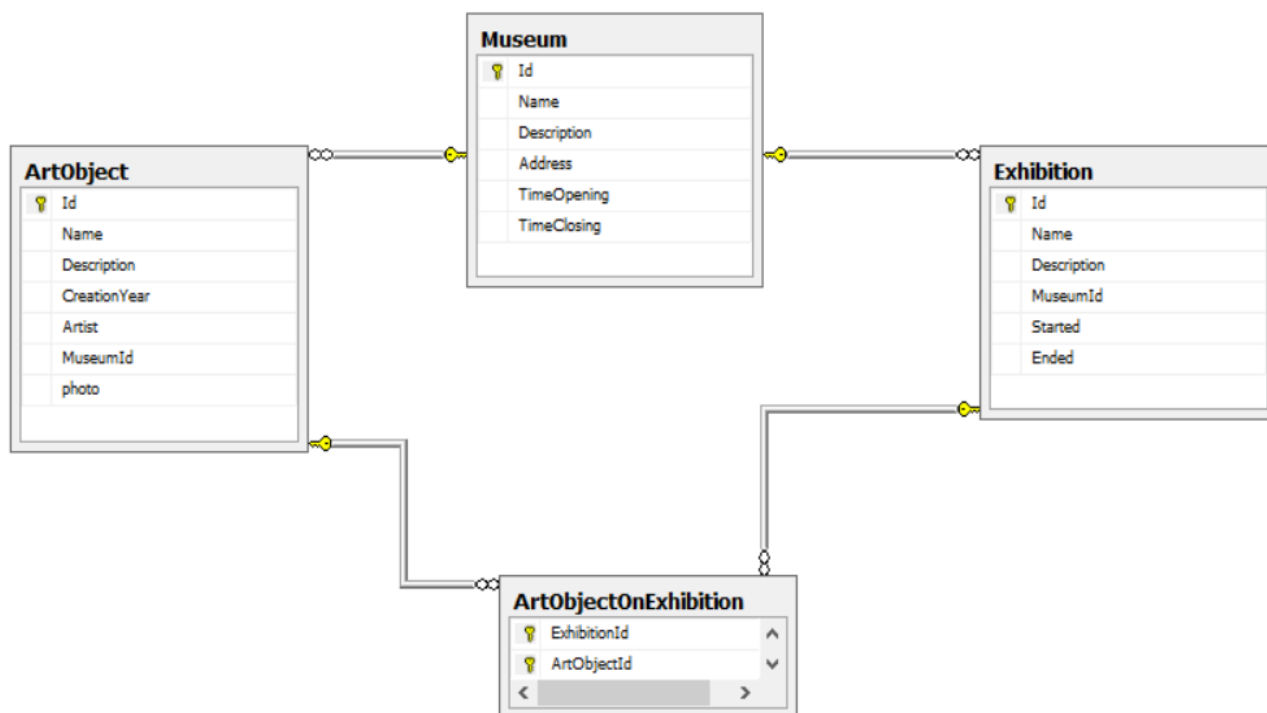


Рисунок 3 - Модель смоделированной базы данных

### 3.3 Разработка таблиц, их полей и ключей

Сведения о спроектированных таблицах представлены в таблице 1.

Таблица 1 – Описание таблиц

Название таблицы	Назначение
Museum	Содержит информацию о музее: идентификатор музея, название, описание, адрес, время открытия и закрытия, количество выставок в музее
Exhibition	Содержит информацию о выставках: идентификатор выставки, название, описание, дата начала и конца выставки, Идентификатор музея, к которому относится выставка
ArtObject	Содержит информацию произведений искусства: идентификатор произведения, название, автор, год создания, описание и изображение произведения

Продолжение таблицы 1

Название таблицы	Назначение
ArtObjectOnExhibition	Содержит информацию о произведениях искусства, которые выставлены на выставках: идентификатор выставки, идентификатор произведения искусства

Таблица 2 – Поля таблицы Museum

Имя столбца	NULL OPTION	Тип данных	Is PK	Is FK	Параметры
Id	NOT NULL	INT	Yes	No	IDENTITY(1,1)
Name	NOT NULL	VARCHAR	No	No	—
Description	NULL	TEXT	No	No	—
Address	NULL	VARCHAR	No	No	—
TimeOpening	NULL	TIME	No	No	—
TimeClosing	NULL	TIME	No	No	—

Таблица 3 – Поля таблицы Exhibition

Имя столбца	NULL OPTION	Тип данных	Is PK	Is FK	Параметры
Id	NOT NULL	INT	Yes	No	IDENTITY(1,1)
Name	NOT NULL	VARCHAR	No	No	—
Description	NULL	TEXT	No	No	—
MuseumId	NOT NULL	INT	No	Yes	—
Started	NULL	DATE	No	No	—
Ended	NULL	DATE	No	No	—

Таблица 4 – Поля таблицы ArtObject

Имя столбца	NULL OPTION	Тип данных	Is PK	Is FK	Параметры
Id	NOT NULL	INT	Yes	No	IDENTITY(1,1)
Name	NULL	VARCHAR	No	No	—
Description	NULL	TEXT	No	No	—
CreationYear	NULL	SMALLINT	No	No	—
Artist	NULL	VARCHAR	No	No	—
MuseumId	NULL	INT	No	Yes	—
photo	NULL	VARBINARY	No	No	—

Таблица 5 – Поля таблицы ArtObjectOnExhibition

Имя столбца	NULL OPTION	Тип данных	Is PK	Is FK	Параметры
ExhibitionId	NOT NULL	INT	Yes	Yes	—
ArtObjectId	NOT NULL	INT	Yes	Yes	—



## Обоснование выбора типов данных и ограничений:

- Поля типа 'int' используются по большей части в качестве идентификаторов, так как они обеспечивают уникальность и авто генерацию значений за счет свойства IDENTITY.
- Поля типа 'varchar (255)' используются для строковых данных, таких как названия музеев, адресов, названий выставок. Ограничение в виде 255 символов вполне достаточно для полного описания.
- Поля типа 'text' используются для хранения большого объема текстовых данных, таких как описания. Обязательное наличие полей данного типа прописано в техническом задании по требованию заказчика.
- Поля типа 'varbinary' используются для хранения изображений. В данном случае это изображения произведений искусства. Обязательное наличие полей данного типа прописано в техническом задании по требованию заказчика.
- Поля типа 'date' и 'time' используются для хранения дат и времени.

```
CREATE TABLE [dbo].[Museum](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](300) NOT NULL,
    [Description] [text] NULL,
    [Address] [varchar](300) NULL,
    [TimeOpening] [time](7) NULL,
    [TimeClosing] [time](7) NULL,

CREATE TABLE [dbo].[ArtObject](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](128) NULL,
    [Description] [text] NULL,
    [CreationYear] [smallint] NULL,
    [Artist] [varchar](200) NULL,
    [MuseumId] [int] NULL,
    [photo] [varbinary](max) NULL);

CREATE TABLE [dbo].[Exhibition](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Name] [varchar](161) NOT NULL,
    [Description] [text] NULL,
    [MuseumId] [int] NOT NULL,
    [Started] [date] NULL,
    [Ended] [date] NULL);

CREATE TABLE [dbo].[ArtObjectOnExhibition](
    [ExhibitionId] [int] NOT NULL,
    [ArtObjectId] [int] NOT NULL);
```

Рисунок 4 - SQL-скрипт создания таблиц

### **3.4 Разработка связей таблиц, ссылочная целостность**

#### **3.4.1 Типы связей между таблицами**

В спроектированной базе данных используются два вида связей. Первая связь – связь «Один-ко-многим» (1-to-Many). В первом случае это таблицы Museum и ArtObject, а во втором - Museum и Exhibition. Если более детально рассматривать первый случай, то он подразумевает под собой, что один музей может содержать несколько произведений искусства, но каждое произведение искусства связано только с одним музеем. Во втором случае связь означает, что один музей может проводить несколько выставок, но каждая выставка проводится только в одном музее.

Второй вид связи, используемый в спроектированной базе данных, это связь «Многие ко многим» (Many-to-Many). К данному виду относятся таблицы ArtObject и Exhibition. Трактовать данный вид связи можно следующим образом: одно произведение искусства может участвовать в нескольких выставках, и одна выставка может включать несколько произведений искусства. Эта связь реализуется через промежуточную таблицу ArtObjectOnExhibition, которая содержит внешние ключи к таблицам ArtObject и Exhibition.

#### **3.4.2 Ссылочная целостность**

Для обеспечения ссылочной целостности в базе данных были установлены внешние ключи, которые обеспечивают правильность данных и предотвращают возникновение несогласованных записей. Ниже приведены подробные сведения о внешних ключах:

##### **1. Внешние ключи в таблице ArtObject**

MuseumId: ссылается на Id в таблице Museum. Обеспечивает, что каждое произведение искусства связано с существующим музеем. При удалении музея, все связанные с ним произведения искусства также удаляются (ON DELETE CASCADE).

##### **2. Внешние ключи в таблице Exhibition**

MuseumId: ссылается на Id в таблице Museum. Обеспечивает, что каждая выставка проводится в существующем музее. При удалении музея, все связанные с ним выставки также удаляются (ON DELETE CASCADE).

### 3. Внешние ключи в таблице ArtObjectOnExhibition

ExhibitionId: ссылается на Id в таблице Exhibition. Обеспечивает, что запись в таблице ArtObjectOnExhibition относится к существующей выставке. При удалении выставки, все связанные записи в ArtObjectOnExhibition также удаляются (ON DELETE CASCADE).

ArtObjectId: ссылается на Id в таблице ArtObject. Обеспечивает, что запись в таблице ArtObjectOnExhibition относится к существующему произведению искусства.

Таким образом, описанные связи и ограничительные условия обеспечивают целостность данных, гарантируя, что все записи связаны корректно и отсутствуют несогласованные данные.

## 3.5 Хранимая процедура

Всего было разработано две хранимые процедуры. Суть работы первой процедуры заключается в привязке какого-либо выбранного произведения искусства к определенной выставке, которая проходит в музее. То бишь происходит некая ассоциация. Работа второй состоит в разрыве вышеописанной связи произведения искусства с выставкой. SQL - код данных процедур представлен на рисунке 5 и рисунке 6.

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[UnbindArtObjectFromExhibition]
    @ArtObjectId INT,
    @ExhibitionId INT
AS
BEGIN
    DELETE FROM ArtObjectOnExhibition
    WHERE ArtObjectId = @ArtObjectId AND ExhibitionId = @ExhibitionId;
END;
```

Рисунок 5 - Процедура отвязки произведения искусства от выставки

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE PROCEDURE [dbo].[AttachArtObjectFromExhibition]
    @ArtObjectId INT,
    @ExhibitionId INT
AS
BEGIN
    INSERT INTO ArtObjectOnExhibition (ExhibitionId, ArtObjectId)
    VALUES (@ExhibitionId, @ArtObjectId);
END;

```

Рисунок 6 - Процедура связывания произведения искусства с выставкой

### 3.6 Хранимая функция

Всего в текущем проекте реализовано 2 хранимые процедуры. Первая функция ведет подсчет количества выставок, зарегистрированных за определенным произведением. Вторая хранимая функция, в свою очередь, ведет сбор информации о том, на каких именно выставках было представлено определенное произведение искусства. SQL – коды данных функций представлены на рисунке 7 и 8.

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetExhibitionsCountByArtObjectId]
    (@ArtObjectId INT)
RETURNS INT
AS
BEGIN
    DECLARE @ExhibitionsCount INT;

    SELECT @ExhibitionsCount = COUNT(*)
    FROM ArtObjectOnExhibition
    WHERE ArtObjectId = @ArtObjectId;

    RETURN @ExhibitionsCount;
END;
GO

```

Рисунок 7 – Хранимая функция подсчета количества выставок для произведения искусства

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE FUNCTION [dbo].[GetExhibitionsByArtObjectId]
    (@ArtObjectId INT)
RETURNS TABLE
AS
RETURN
(
    SELECT e.*
    FROM Exhibition e
    INNER JOIN ArtObjectOnExhibition aoe ON e.Id = aoe.ExhibitionId
    WHERE aoe.ArtObjectId = @ArtObjectId
);
GO

```

Рисунок 8 – Хранимая функция сбора информации о выставках произведения искусства

### 3.7 Представление

Реализованное представление содержит в себе информацию о музеях, которая является главной, а также информация о количестве проводимых выставок, подсчет которой проводится с помощью хранимой функции, описанной в пункте выше. SQL – код описанного выше представления и результата его выполнения в MS SQL Server Manager представлен на рисунке 9 и 10.

```

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE view [dbo].[MuseumWithExhibitionCount] as
SELECT museum.*, (SELECT COUNT(*) FROM exhibition WHERE exhibition.museumid = museum.id) as ExhibitionCount FROM museum
GO

```

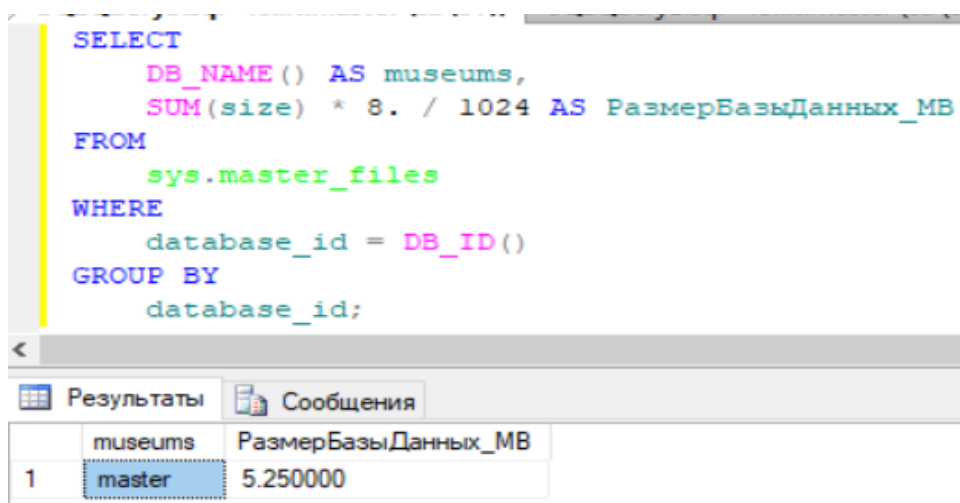
Рисунок 9 - Реализованное представление

Результаты		Сообщения					
Id	Name	Description	Address	TimeOpening	TimeClosing	ExhibitionCount	
1	16	Третьяковская галерея	Историческое здание Третьяков...	Лаврушинский переулок, д. 10, Москв...	10:00:00.0000000	20:00:00.0000000	3
2	23	Государственный музей изобразит...	ГМИИ им. А.С. Пушкина – музейн...	Россия, Москва, Волхонка, 12	10:00:00.0000000	19:00:00.0000000	1
3	33	abc	abc abc	abcbabcb	01:00:00.0000000	23:00:00.0000000	0
4	34	Название музея	Описание музея	Адрес музея	01:00:00.0000000	23:00:00.0000000	1

Рисунок 10 - Результат вызова представления.

### 3.8 Оценка размера базы данных

На данный момент времени база данных насчитывает 4 записи о музеях, 5 записей о выставках, 9 записей с произведениями искусства и 10 записей с информацией о выставленных произведениях искусства на соответствующих выставках. Текущий размер базы данных, включая представление, хранимые функции и процедуры составляет 5.25 МБ, что продемонстрировано на рисунке 11.



The screenshot shows a SQL query window with the following text:

```
SELECT
    DB_NAME() AS museums,
    SUM(size) * 8. / 1024 AS РазмерБазыДанных_MB
FROM
    sys.master_files
WHERE
    database_id = DB_ID()
GROUP BY
    database_id;
```

Below the query window, there are two tabs: "Результаты" (Results) and "Сообщения" (Messages). The "Результаты" tab is active, showing a table with the following data:

	museums	РазмерБазыДанных_MB
1	master	5.250000

Рисунок 11 - Расчет текущего размера базы данных

## 4. СОЗДАНИЕ СЕРВЕРА ПРИЛОЖЕНИЙ

### 4.1 Разворачивание сервера приложений с помощью Jetty

Для нашего проекта, основанного на трёхзвенной архитектуре, был выбран сервер приложений Jetty. Jetty – это легковесный и высокопроизводительный контейнер сервлетов, который поддерживает спецификации Java Servlet и JavaServer Pages (JSP). Его преимущества мы уже рассматривали.

При создании Maven проекта в Eclipse IDE прежде всего надо добавить зависимости и плагины в файл `pom.xml`, которые позволят использовать Jetty,

используя при этом центральный репозиторий Maven [2]. Ниже приведены рисунки конфигурации Maven для интеграции Jetty с разрабатываемым проектом.

```
<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-servlet</artifactId>
  <version>9.4.44.v20210927</version>
</dependency>

<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-server</artifactId>
  <version>9.4.44.v20210927</version>
</dependency>

<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-webapp</artifactId>
  <version>9.4.44.v20210927</version>
</dependency>
```

Рисунок 12 – Зависимости Jetty в pom.xml

```
<build>
  <sourceDirectory>src</sourceDirectory>
  <plugins>

    <plugin>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.5.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>

    <plugin>
      <artifactId>maven-war-plugin</artifactId>
      <version>3.4.0</version>
      <configuration>
        <warSourceDirectory>WebContent</warSourceDirectory>
        <failOnMissingWebXml>>false</failOnMissingWebXml>
      </configuration>
    </plugin>

    <plugin>
      <groupId>org.eclipse.jetty</groupId>
      <artifactId>jetty-maven-plugin</artifactId>
      <version>9.4.54.v20240208</version>
    </plugin>

  </plugins>
</build>
</project>
```

Рисунок 13 – Плагины Maven и Jetty в pom.xml

Запуск сервера приложений Jetty происходит в два этапа: 1) компиляция и сборка проекта через команду в консоли «mvnw clean install»; 2) Запуск сервера Jetty через команду в консоли «mvnw jetty: run-war»;

Этот запуск приведет к тому, что Jetty поднимет сервер и развернет наше веб-приложение. Оно будет доступно по адресу «http://localhost:8080». При желании, стандартный порт 8080 можно изменить в конфигурации Jetty на любой другой, но в данной работе будет использоваться именно он, поскольку он свободный и, как было экспериментально установлено, ни с кем не конфликтует. Соответственно, при закрытии приложения, данный порт будет освобожден и может использоваться в других целях.

## 1. Конфигурации сервера

Для конфигурации сервера любого веб-приложения на Java важной частью является web.xml файл. Он используется для определения параметров сервера, таких как пути к сервлетам, параметры подключения к базам данных и обработка ошибок. Конкретно в нашем проекте данный файл содержит параметры для подключения к базе данных и настройки для обработки исключений. Содержимое файла представлено на рисунке 14.

```
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
id="WebApp_ID" version="3.1">
<display-name>Museum Web Application</display-name>

<context-param>
    <param-name>jdbcURL</param-name>
    <!-- <param-value>jdbc:sqlserver://10.4.130.105:1433;database=B2;trustServerCertificate=true;</param-value> -->
    <param-value>jdbc:sqlserver://localhost:1433;database=museums;trustServerCertificate=true;</param-value>
</context-param>

<context-param>
    <param-name>jdbcUsername</param-name>
    <param-value>sa</param-value>
</context-param>

<context-param>
    <param-name>jdbcPassword</param-name>
    <param-value>123</param-value>
</context-param>

<error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/Error.jsp</location>
</error-page>
</web-app>
```

Рисунок 14 – содержимое web.xml файла



## 4.2 Реализация сервлетов приложения

Для того, чтобы наше приложение могло обрабатывать HTTP запросы, которые присылает клиент на сервер, необходимо реализовать соответствующие сервлеты, которые эти запросы будут принимать, обрабатывать и возвращать HTTP ответ обратно клиенту. Для работы с сервлетами, опять же, необходимо подключить несколько зависимостей в pom.xml файле. Данные зависимости представлены на рисунке 15.

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>

<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>

<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>javax.servlet.jsp-api</artifactId>
  <version>2.3.1</version>
  <scope>provided</scope>
</dependency>
```

Рисунок 15 – Зависимости для сервлетов в pom.xml

В качестве примера рассмотрим реализованный сервлет MuseumServlet.java, который обрабатывает запросы, связанные с управлением данных о музеях. В нём определены методы для обработки различных действий, таких как создание, обновление, удаление и отображение музеев. Для каждого сервлета в нашем приложении есть общий скелет – методы init(), doPost() и doGet(). Данные методы предназначены для обработки вышеупомянутых запросов. Например, doPost() перенаправляет все запросы POST-запросы к методу doGet() для унифицированной обработки запросов. Метод doGet() является основным методом обработки запросов, который определяет действие, исходя из

параметров запроса, и вызывает соответствующий метод для обработки этого действия. Примеры метода doGet() и метода вставки информации в главную таблицу представлены на рисунках 16 и 17.

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String action = Utils.getAction(request);

    try {
        switch (action) {
            case "/new":
                showNewForm(request, response);
                break;
            case "/insert":
                insertMuseum(request, response);
                break;
            case "/delete":
                deleteMuseum(request, response);
                break;
            case "/edit":
                showEditForm(request, response);
                break;
            case "/update":
                updateMuseum(request, response);
                break;
            case "/list":
            default:
                listMuseum(request, response);
                break;
        }
    } catch (SQLException ex) {
        throw new ServletException(ex);
    }
}
```

Рисунок 16 – метод doGet() для MuseumServlet.java

```
private void insertMuseum(HttpServletRequest request, HttpServletResponse response)
    throws SQLException, IOException {
    Museum newMuseum = convertMuseum(request);
    museumDAO.insertMuseum(newMuseum);
    response.sendRedirect("list");
}
```

Рисунок 17 – метод insertMuseum() для MuseumServlet.java

Метод «convertMuseum» используется для преобразования параметров запроса в объект «Museum».

Остальные сервлеты в приложении реализованы аналогичным образом. Каждый сервлет отвечает за управление конкретным аспектом данных, такими как выставки или произведения искусства. Все сервлеты содержат методы, описанные выше, и специализированные методы для обработки

соответствующих действий. Таким образом, сервлеты в нашем приложении обеспечивают динамическое взаимодействие между клиентом и сервером, обрабатывают запросы, выполняют бизнес-логику и возвращают результаты клиенту. Это позволяет создавать гибкие и масштабируемые веб-приложения на основе трехзвенной архитектуры.

Немаловажным является также реализация моделей и DAO (Data Access Object). Модели, в свою очередь, представляют собой некий Java-класс, который содержит свойства и методы для работы с данными и передачи данных между слоями приложения. Если продолжать пример с музеями, то это будут поля главной таблицы, а также геттеры и сеттеры. Часть кода модели представлена на рисунке 18.

```
public class Museum {
    private Integer Id;
    private String Name;
    private String Description;
    private String Address;
    private Time TimeOpening;
    private Time TimeClosing;
    private Integer exhibitionCount;

    public Museum() {
    }

    public Museum(Integer Id, String Name, String Description, String Address, Time TimeOpening, Time TimeClosing) {
        this.Id = Id;
        this.Name = Name;
        this.Description = Description;
        this.Address = Address;
        this.TimeOpening = TimeOpening;
        this.TimeClosing = TimeClosing;
    }

    public int getId() {
        return this.Id;
    }

    public void setId(int value) {
        this.Id = value;
    }

    public String getName() {
        return this.Name;
    }

    public void setName(String value) {
        this.Name = value;
    }
}
```

Рисунок 18 – модель музеев

DAO классы отвечают за взаимодействие с базой данных. Они содержат методы для выполнения CRUD (Create, Read, Update, Delete) операций с данными, их логгирование для отслеживания действий с базой данных,

обработка исключений и управление транзакциями. Пример операции вставки DAO класса представлен на рисунке 19.

```
public boolean insertMuseum(Museum museum) throws SQLException {
    String sql = "INSERT INTO Museum (Name, Description, Address, TimeOpening, TimeClosing) VALUES (?, ?, ?, ?, ?)";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowInserted = false;

    logger.info("Inserting museum: " + museum.getName());

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        statement = connection.prepareStatement(sql);
        statement.setString(1, museum.getName());
        statement.setString(2, museum.getDescription());
        statement.setString(3, museum.getAddress());
        statement.setTime(4, museum.getTimeOpening());
        statement.setTime(5, museum.getTimeClosing());

        rowInserted = statement.executeUpdate() > 0;
        connection.commit(); // Подтверждение транзакции

        logger.info("Museum inserted: " + museum.getName());
    } catch (SQLException e) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции в случае ошибки
        }
        logger.log(Level.SEVERE, "Error inserting museum", e);
        e.printStackTrace();
    } finally {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true); // Восстановление автокоммита
            connection.close();
        }
    }

    return rowInserted;
}
```

Рисунок 19 – операция вставки в главную таблицу

Таким образом, модели и DAO классы обеспечивают структуру и методы для работы с данными в приложении, позволяя осуществлять основные операции с базой данных.

### 4.3 Транзакции

По умолчанию в JDBC для управления транзакциями, уровень изоляции транзакции по умолчанию определен как «TRANSACTION\_READ\_COMMITTED». Это означает, что внутри одной

транзакции можно видеть только те изменения, которые были подтверждены другими транзакциями. Неподтвержденные изменения (т.е., те, которые были выполнены в других транзакциях, но не были зафиксированы) не будут видны. Следовательно, данный уровень обеспечивает защиту от «грязного» чтения.

Механизм транзакций, реализованный в приложении, служит для отката транзакции в случае невыполнения операции или вызова исключения. В случае возникновения ошибки или исключения транзакция откатывается, что позволяет сохранить целостность данных.

В отличие от некоторых других технологий, в сервлетах нет встроенной поддержки распределенных транзакций, которые могут включать более одного действия, выполняемого в разных методах или сессиях. Это связано с тем, что HTTP-запросы являются stateless (без состояния), и каждое действие в приложении обычно выполняется в пределах одного HTTP-запроса.

Таким образом, для выполнения каждого действия используется своя собственная транзакция. Например, при выполнении операций вставки, обновления или удаления в базе данных, транзакция открывается, выполняется соответствующее действие, и затем либо фиксируется (commit), либо откатывается (rollback) в случае ошибки.

## **5. СОЗДАНИЕ ПРИЛОЖЕНИЯ КЛИЕНТА**

Клиентская часть приложения представляет собой веб-интерфейс, через который пользователи взаимодействуют с сервером. Архитектура клиентской части основана на модели «клиент-сервер», где веб-браузер выступает в роли клиента, а серверная часть отвечает за обработку запросов и предоставление данных.

Для реализации клиентской части приложения были использованы следующие технологии и инструменты:

- HTML – разметка веб-страниц.
- CSS – стилизация веб-страниц для обеспечения визуальной привлекательности и удобства использования.

- JavaScript – обработка событий на стороне клиента и динамическое обновление содержимого веб-страниц.
- JSP (JavaServer Pages) – динамическое создание HTML-кода на стороне сервера с использованием Java[7]
- Servlets – обработка HTTP-запросов и генерация ответов на стороне сервера.

Часть HTML-кода и пример реализации клиентской части главной страницы приложения представлены на рисунках 20 и 21.

```

<? MuseumList.jsp >
C: > MuseumApp > MuseumApp > WebContent > WEB-INF > museums > <? MuseumList.jsp > ?
1 <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
2 <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
3 <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
4
5 <html>
6 <head>
7
8 <style>
9     display: flex;
10    flex-direction: column;
11    align-items: center;
12
13 }
14 </style>
15 </head>
16 <body>
17
18 <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
19     <a class="navbar-brand" href="#">Музейное приложение</a>
20     <div class="collapse navbar-collapse">
21         <ul class="navbar-nav ml-auto">
22             <li class="nav-item">
23                 <a class="nav-link" href="/museums/new">Добавить музей</a>
24             </li>
25             <li class="nav-item">
26                 <a class="nav-link" href="/exhibitions">Список всех выставок</a>
27             </li>
28             <li class="nav-item">
29                 <a class="nav-link" href="/art-objects/">Список всех произведений искусства</a>
30             </li>
31             <li class="nav-item">
32                 <a class="nav-link" href="/ExportToExcelServlet">Экспортировать в Excel</a>
33             </li>
34         </ul>
35     </div>
36 </nav>
37
38 <div class="container">
39     <h1 class="text-center">Управление музеями</h1>
40
41
42
43
44
45
46
47
48
49
50
51
52

```

Рисунок 20 – часть HTML-кода главной страницы приложения

ID	Название	Описание	Адрес	Время открытия	Время закрытия	Всего выставок	Действия
16	Третьяковская галерея	Историческое здание Третьяковской галереи по адресу Лаврушинский переулок, д. 10, Москва 119017 Россия является особняком второй половины XVIII века. Семья Третьяковых приобрела этот дом в 1851 году. Начав в 1856 собирать картины русских художников, Павел Третьяков размещал их в жилых комнатах. По мере роста коллекции для ее экспонирования потребовались новые помещения, и с 1872 года к дому начали делать пристройки. При жизни Третьякова такие работы проводились пять раз. В 1902–1904 годах внешняя часть здания со стороны	Лаврушинский переулок, д. 10, Москва 119017 Россия	10:00	20:00	3	<div> Выставки </div> <div> Произведения искусства </div> <div> Изменить </div> <div> Удалить </div>

Рисунок 21 – главная страница приложения

Кнопки действий над записями на всех страницах расположены правее определенных записей. Кнопки для перехода между страницами или добавление записей расположены на черной полосе страниц, находящихся сверху. Туда же помещена кнопка экспорта отчёта в Excel.

## 5.1 Редактирование и загрузка изображения в БД

Одной из ключевых функций приложения является возможность загрузки из файловой системы компьютера и редактирования изображений в базе данных. Это достигается посредством сервлетов и формы на стороне клиента, которая позволяет пользователю загружать файлы изображений. В серверной части используется библиотека Apache Commons FileUpload для обработки загруженных файлов и сохранения их в формате BLOB в базе данных. Пример загруженного и отредактированного изображения представлен на рисунке 22.

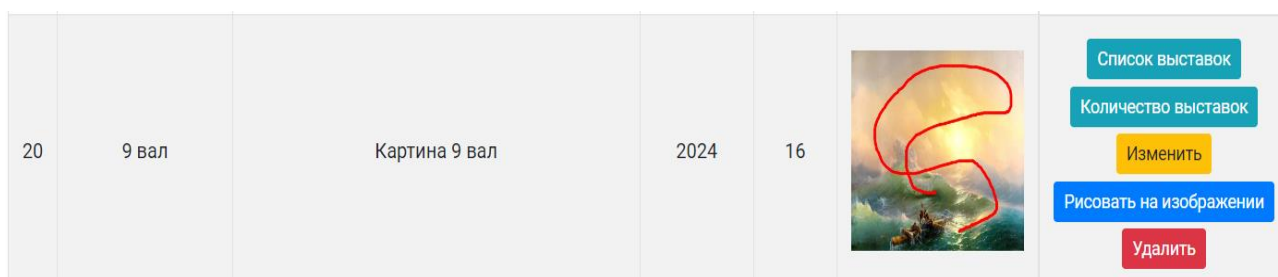


Рисунок 22 – загруженное и отредактированное изображение

## 5.2 Хранимые процедуры и функции

Приложение использует хранимые процедуры и функции для выполнения сложных операций с данными. Например, для привязки произведений искусства к выставкам и отвязки их используются хранимые процедуры `AttachArtObjectFromExhibition` и `UnbindArtObjectFromExhibition`. Для получения данных о количестве выставок, на которых было представлено произведение искусства, используется хранимая функция `GetExhibitionsCountByArtObjectId`.

Результаты выполнения хранимой функции представлен на рисунке 23, а возможности хранимых процедур по привязке и отвязке произведений искусств к выставкам на рисунках 24 и 25.

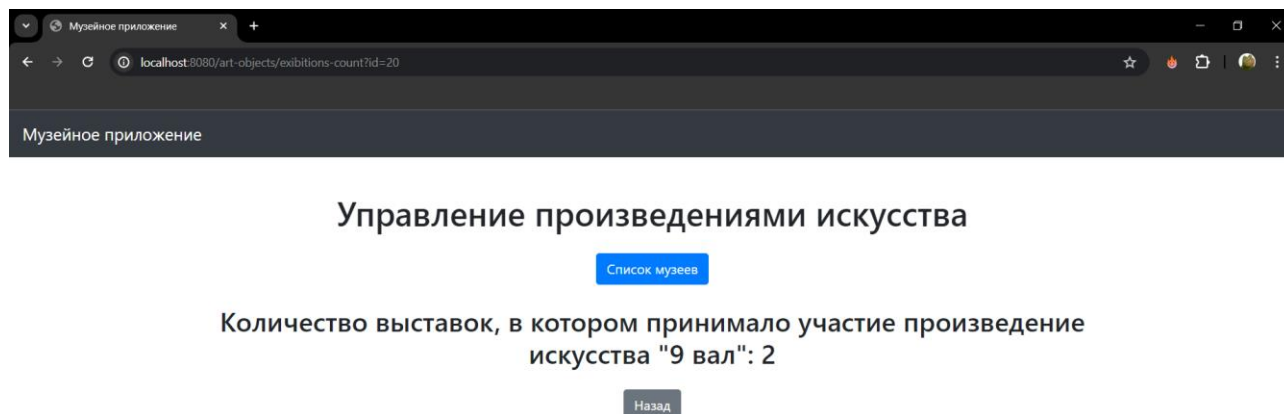


Рисунок 23 – результат выполнения хранимой функции.

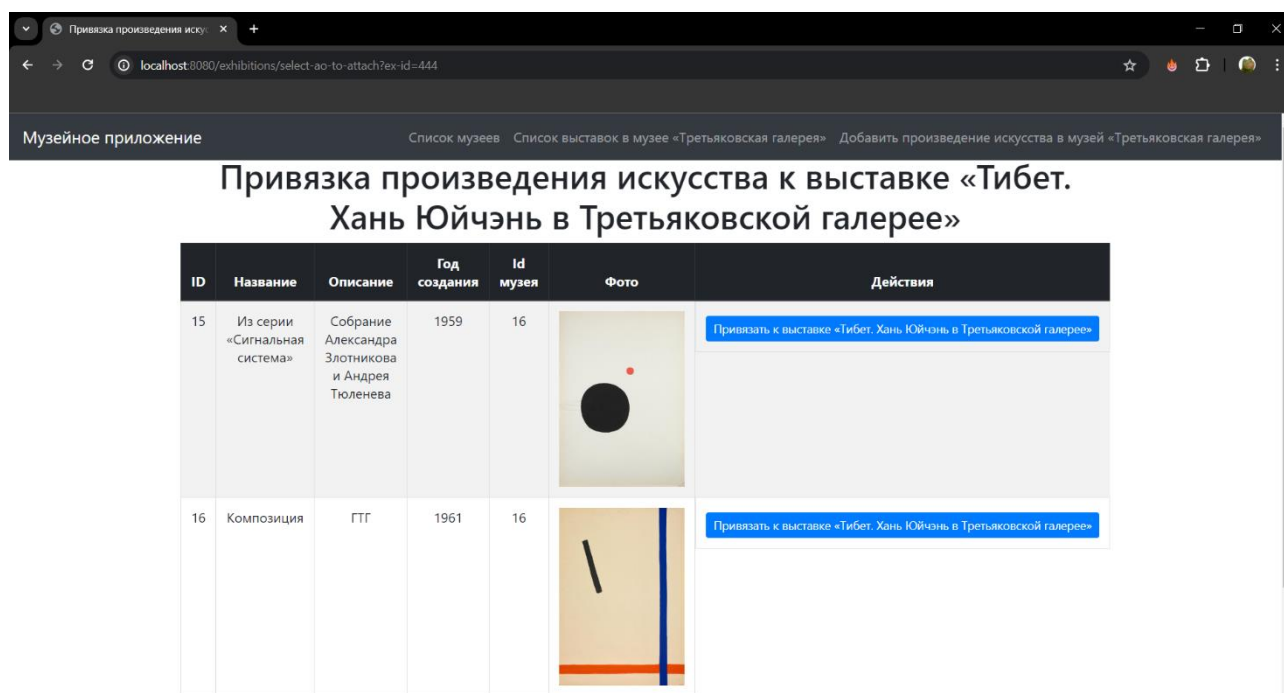


Рисунок 24 – возможность привязки картины к выставке



Музейное приложение

Список музеев   Список выставок в музее «Третьяковская галерея»   Привязка произведения искусства к выставке «Тибет. Хань Юйчань в Третьяковской галерее»

### Список произведений искусства на выставке




ID	Название	Описание	Год создания	Id музея	Фото	Операции
12	На пути паломничества	Ханьданьский художественный музей Хань Юйчаня, Ханьдань	2009	16		Убрать с выставки
13	Простирая	Ханьданьский художественный музей Хань Юйчаня, Ханьдань	2014	16		Убрать с выставки
14	Фэн-ма флаг	Ханьданьский художественный музей Хань Юйчаня, Ханьдань	2023	16		Убрать с выставки

Рисунок 25 – возможность отвязать картину от выставки

### 5.3 Графический отчет в Excel

Для создания графического отчета в Excel используется библиотека Apache POI[8]. Отчет генерируется на основе данных из базы и сохраняется в формате .xlsx. Чтобы его сгенерировать необходимо нажать «Экспорт в Excel» на главной странице приложения. Пример сгенерированного отчета на рисунке 26.

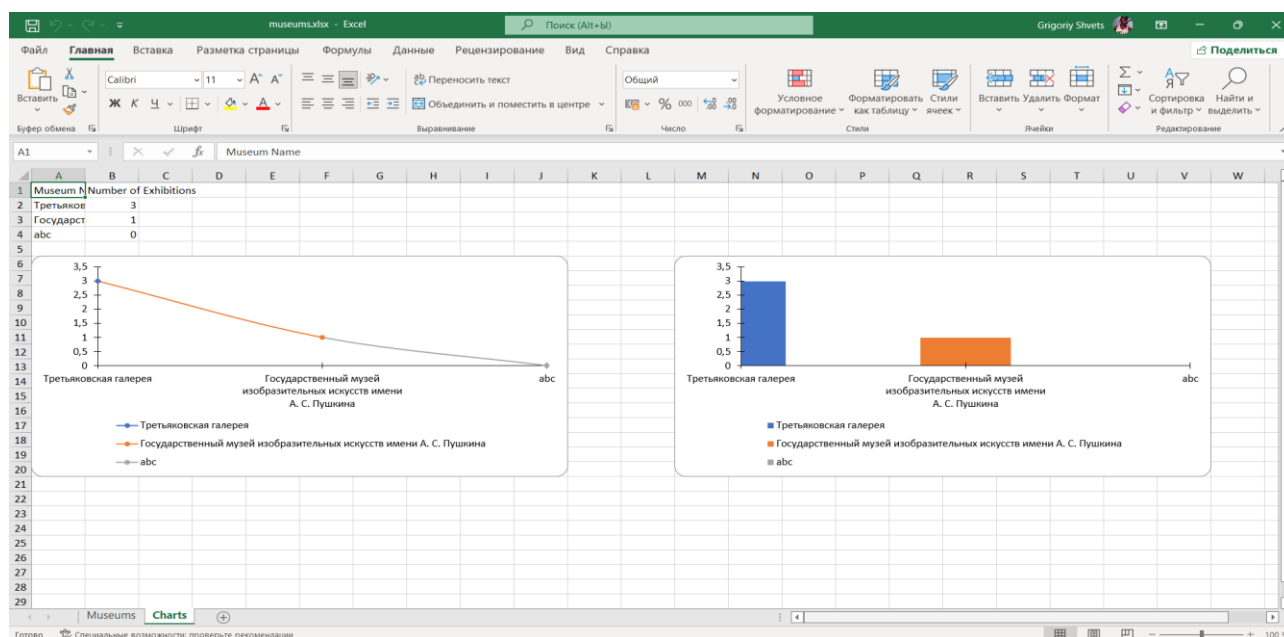


Рисунок 26 – пример сгенерированного отчета Excel

## 5.4 Табличный отчет

В разработанном приложении также присутствует возможность формирования табличного отчета при помощи JasperReports [11], зависимости которого необходимо указать в pom.xml файле [3]. Для формирования отчета необходимо так же реализовать соответствующий сервлет и .jgxml файл разметки данного отчета. Пример отчета на рисунке 27.



### Museum Report

ID	Name	Description	Address	Time Opening	Time Closing	Exhibitions
16	Третьяковская галерея	Историческое здание Третьяковской галереи по адресу Лаврушинский	Лаврушинский переулок, д. 10, Москва 119017 Россия	10:00:00	20:00:00	3
23	Государственный музей изобразительных	ГМИИ им. А.С. Пушкина – музейный комплекс, обладающий одним из	Россия, Москва, Волхонка, 12	10:00:00	19:00:00	1
33	abc	abc abc	abcbcabcb	01:00:00	23:00:00	0

Рисунок 27 – табличный отчет JasperReports

## 6. КОМПЛЕКС ПОСТАВКИ РАЗРАБОТАННОГО ПРИЛОЖЕНИЯ

Разработанное приложение состоит из следующих частей:

- Сервер БД (Microsoft SQL Server 2008 R2).

- Сервер приложений (Jetty и .war файл).
- Приложение клиент (Современный браузер).

### **Требования к серверу БД:**

На сервере должна быть установлена СУБД Microsoft SQL Server 2008 R2.

### **Требования к компьютерам:**

Для успешной работы разработанного приложения необходимо учитывать минимальные требования к аппаратному и программному обеспечению как для сервера приложений, так и для клиентских компьютеров.

На компьютерах должна быть установлена платформа Microsoft .NET Framework 4.5 или выше, Windows Management Framework 5.1 или выше, должна быть обновлена оболочка Windows PowerShell до версии 3.0 или выше (предпочтительнее 5.1), JDK 8, современный браузер (Google Chrome). Также компьютеры должны находиться в одной доменной сети и иметь доступ друг к другу.

## **СПИСОК ЛИТЕРАТУРЫ**

1. Java Servlet API Documentation // Официальная документация Oracle [Электронный ресурс]. – URL: <https://docs.oracle.com/javaee/7/api/javax/servlet/package-summary.html> (дата обращения: 05.04.2024)

2. Maven: The Complete Reference // Официальная документация Apache Maven [Электронный ресурс]. – URL: <https://maven.apache.org/guides/> (дата обращения: 06.04.2024)

3. JasperReports Library // Официальная документация JasperReports [Электронный ресурс]. – URL: <https://community.jaspersoft.com/documentation> (дата обращения: 01.06.2024)

4. Microsoft SQL Server JDBC Driver Documentation // Официальная документация Microsoft [Электронный ресурс]. – URL:

<https://docs.microsoft.com/en-us/sql/connect/jdbc/> (дата обращения: 10.05.2024)

5. The Java™ Tutorials: JDBC Basics // Официальная документация Oracle [Электронный ресурс]. – URL: <https://docs.oracle.com/javase/tutorial/jdbc/> (дата обращения: 10.05.2024)

6. Jetty Documentation // Официальная документация Eclipse Jetty [Электронный ресурс]. – URL: <https://www.eclipse.org/jetty/documentation/current/> (дата обращения: 10.05.2024)

7. JavaServer Pages (JSP) // Официальная документация Oracle [Электронный ресурс]. – URL: <https://docs.oracle.com/javaee/7/tutorial/jsf-basic001.htm> (дата обращения: 11.05.2024)

8. POI - the Java API for Microsoft Documents // Официальная документация Apache POI [Электронный ресурс]. – URL: <https://poi.apache.org/documentation/> (дата обращения: 02.06.2024)

9. Java SE Development Kit 8 Documentation // Официальная документация Oracle [Электронный ресурс]. – URL: <https://docs.oracle.com/javase/8/docs/> (дата обращения: 01.05.2024)

10. Effective Java, Third Edition // Joshua Bloch – Addison-Wesley Professional, 2017. – 416 p.

11. Отчеты для баз данных: учеб. пособие / Г.А. Бородин, И.Н. Андреева. – М.: Издательство МЭИ, 2018. – 64 с.

## ПРИЛОЖЕНИЕ

### **AbstractDAO.Java**

```
package org.servlet.museum_management.DAO;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

public abstract class AbstractDAO {
    private String jdbcURL;
    private String jdbcUsername;
    private String jdbcPassword;

    private static final Logger LOGGER =
        Logger.getLogger(AbstractDAO.class.getName());

    public AbstractDAO(String jdbcURL, String jdbcUsername, String jdbcPassword)
    {
        this.jdbcURL = jdbcURL;
        this.jdbcUsername = jdbcUsername;
        this.jdbcPassword = jdbcPassword;
    }

    protected Connection getConnection() throws SQLException {
        try {
            Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
            LOGGER.log(Level.INFO, "==== Connection established successfully!
=====");
        } catch (ClassNotFoundException e) {
            LOGGER.log(Level.SEVERE, "==== Failed to establish connection!
=====", e);
            throw new SQLException(e);
        }
        return DriverManager.getConnection(jdbcURL, jdbcUsername, jdbcPassword);
    }
}
```

### **ArtObjectDAO.Java**

```
package org.servlet.museum_management.DAO;
```

```

import org.servlet.museum_management.model.ArtObject;
import org.servlet.museum_management.model.Exhibition;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ArtObjectDAO extends AbstractDAO {

    public ArtObjectDAO(String jdbcURL, String jdbcUsername, String
        jdbcPassword) {
        super(jdbcURL, jdbcUsername, jdbcPassword);
    }

    private static final Logger logger =
        Logger.getLogger(ArtObjectDAO.class.getName());

    public boolean insertArtObject(ArtObject artObject) throws SQLException {
        String sql = "INSERT INTO ArtObject (Name, Description, CreationYear,
            artist, MuseumId, photo) VALUES (?, ?, ?, ?, ?, ?)";
        Connection connection = null;
        PreparedStatement statement = null;
        boolean rowInserted = false;

        logger.info("Inserting art object: " + artObject.getName());

        try {
            connection = getConnection();
            connection.setAutoCommit(false); // Начало транзакции

            statement = connection.prepareStatement(sql);
            statement.setString(1, artObject.getName());
            statement.setString(2, artObject.getDescription());
            statement.setInt(3, artObject.getCreationYear());
            statement.setString(4, artObject.getArtist());
            statement.setInt(5, artObject.getMuseumId());
            statement.setBytes(6, artObject.getPhoto());

            rowInserted = statement.executeUpdate() > 0;
            connection.commit(); // Подтверждение транзакции

            logger.info("Art object inserted: " + artObject.getName());
        }
    }
}

```

```

    } catch (SQLException ex) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции при ошибке
        }
        logger.log(Level.SEVERE, "Error inserting art object", ex);
        throw ex;
    } finally {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true);
            connection.close();
        }
    }

    return rowInserted;
}

public List<ArtObject> listAllArtObjects(Integer museumId) throws
SQLException {
    List<ArtObject> artObjects = new ArrayList<>();
    String sql = "SELECT * FROM ArtObject WHERE museumId = ?";

    logger.info("Listing all art objects for museumId: " + museumId);

    try (Connection connection = getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, museumId);
        ResultSet resultSet = statement.executeQuery();

        while (resultSet.next()) {
            ArtObject artObject = convertArtObject(resultSet);
            artObjects.add(artObject);
        }

        resultSet.close();
    } catch (SQLException ex) {
        logger.log(Level.SEVERE, "Error listing all art objects for museumId: " +
museumId, ex);
        throw ex;
    }

    return artObjects;
}

```

```

public List<ArtObject> listAllArtObjectsNotAttachedToExhibition(Integer
museumId, int exhibitionId) throws SQLException {
    List<ArtObject> artObjects = new ArrayList<>();
    String sql = "SELECT ao.* " +
        "FROM ArtObject ao " +
        "LEFT JOIN ArtObjectOnExhibition aoe " +
        " ON ao.Id = aoe.ArtObjectId AND aoe.ExhibitionId = ? " +
        "WHERE ao.MuseumId = ? AND aoe.ArtObjectId IS NULL; ";

    logger.info("Listing all art objects not attached to exhibitionId: " + exhibitionId +
" for museumId: " + museumId);

    try (Connection connection = getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, exhibitionId);
        statement.setInt(2, museumId);
        ResultSet resultSet = statement.executeQuery();

        while (resultSet.next()) {
            ArtObject artObject = convertArtObject(resultSet);
            artObjects.add(artObject);
        }

        resultSet.close();
    } catch (SQLException ex) {
        logger.log(Level.SEVERE, "Error listing all art objects not attached to
exhibitionId: " + exhibitionId + " for museumId: " + museumId, ex);
        throw ex;
    }

    return artObjects;
}

public List<ArtObject> listAllArtObjects() throws SQLException {
    List<ArtObject> artObjects = new ArrayList<>();
    String sql = "SELECT * FROM ArtObject";

    logger.info("Listing all art objects");

    try (Connection connection = getConnection();
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet resultSet = statement.executeQuery()) {

        while (resultSet.next()) {

```



```

        ArtObject artObject = convertArtObject(resultSet);
        artObjects.add(artObject);
    }
} catch (SQLException ex) {
    logger.log(Level.SEVERE, "Error listing all art objects", ex);
    throw ex;
}

return artObjects;
}

public boolean deleteArtObject(ArtObject artObject) throws SQLException {
    String sql1 = "DELETE FROM ArtObjectOnExhibition WHERE ArtObjectId = ?";
    String sql2 = "DELETE FROM ArtObject WHERE id = ?";
    Connection connection = null;
    PreparedStatement statement1 = null;
    PreparedStatement statement2 = null;
    boolean rowDeleted = false;

    logger.info("Deleting art object: " + artObject.getName());

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        statement1 = connection.prepareStatement(sql1);
        statement1.setInt(1, artObject.getId());
        statement1.executeUpdate();

        statement2 = connection.prepareStatement(sql2);
        statement2.setInt(1, artObject.getId());
        rowDeleted = statement2.executeUpdate() > 0;

        connection.commit(); // Подтверждение транзакции

        logger.info("Art object deleted: " + artObject.getName());
    } catch (SQLException ex) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции при ошибке
        }
        logger.log(Level.SEVERE, "Error deleting art object", ex);
        throw ex;
    } finally {
        if (statement1 != null) {

```

```

        statement1.close();
    }
    if (statement2 != null) {
        statement2.close();
    }
    if (connection != null) {
        connection.setAutoCommit(true);
        connection.close();
    }
}

return rowDeleted;
}

public boolean updateArtObject(ArtObject artObject, boolean updatePhoto)
throws SQLException {
    String sql = "UPDATE ArtObject SET Name = ?, Description = ?, CreationYear
= ?, artist = ?, MuseumId = ?" +
        (updatePhoto ? ", photo = ? " : "") +
        "WHERE id = ?";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowUpdated = false;

    logger.info("Updating art object: " + artObject.getName());

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        statement = connection.prepareStatement(sql);
        statement.setString(1, artObject.getName());
        statement.setString(2, artObject.getDescription());
        statement.setInt(3, artObject.getCreationYear());
        statement.setString(4, artObject.getArtist());
        statement.setInt(5, artObject.getMuseumId());
        if (updatePhoto) {
            statement.setBytes(6, artObject.getPhoto());
            statement.setInt(7, artObject.getId());
        } else {
            statement.setInt(6, artObject.getId());
        }
    }

    rowUpdated = statement.executeUpdate() > 0;
    connection.commit(); // Подтверждение транзакции
}

```

```

        logger.info("Art object updated: " + artObject.getName());
    } catch (SQLException ex) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции при ошибке
        }
        logger.log(Level.SEVERE, "Error updating art object", ex);
        throw ex;
    } finally {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true);
            connection.close();
        }
    }
}

return rowUpdated;
}

public ArtObject getArtObject(int id) throws SQLException {
    ArtObject artObject = null;
    String sql = "SELECT * FROM ArtObject WHERE id = ?";

    logger.info("Getting art object with id: " + id);

    try (Connection connection = getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, id);
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            artObject = convertArtObject(resultSet);
        }

        resultSet.close();
    } catch (SQLException ex) {
        logger.log(Level.SEVERE, "Error getting art object with id: " + id, ex);
        throw ex;
    }
    return artObject;
}

public static ArtObject convertArtObject(ResultSet resultSet) throws

```

```

SQLException {
    int id = resultSet.getInt("id");
    String name = resultSet.getString("name");
    String description = resultSet.getString("description");
    Integer creationYear = resultSet.getInt("creationYear");
    String artist = resultSet.getString("artist");
    Integer museumId = resultSet.getInt("museumId");
    byte[] photo = resultSet.getBytes("photo");

    logger.info("Converting ResultSet to ArtObject with id: " + id);
    return new ArtObject(id, name, description, creationYear, artist, museumId,
photo);
}

public List<Exhibition> getExhibitionsByArtObjectId(int artObjectId) throws
SQLException {
    List<Exhibition> listExhibition = new ArrayList<>();
    String sql = "select * from GetExhibitionsByArtObjectId(?)";

    logger.info("Getting exhibitions for ArtObject with id: " + artObjectId);

    try (Connection connection = getConnection();
        CallableStatement stmt = connection.prepareCall(sql)) {
        stmt.setInt(1, artObjectId);

        ResultSet rs = stmt.executeQuery();
        while (rs.next()) {
            Exhibition exhibition = ExhibitionDAO.convertExhibition(rs);
            listExhibition.add(exhibition);
        }
        logger.info("Found " + listExhibition.size() + " exhibitions for ArtObject with
id: " + artObjectId);
    } catch (SQLException e) {
        logger.severe("Error getting exhibitions for ArtObject with id: " + artObjectId
+ " - " + e.getMessage());
        throw e;
    }

    return listExhibition;
}

public int getExhibitionsCountByArtObjectId(int artObjectId) throws
SQLException {
    String sql = "{ ? = CALL GetExhibitionsCountByArtObjectId(?) }";

```

```

logger.info("Getting exhibition count for ArtObject with id: " + artObjectId);

try (Connection connection = getConnection();
    CallableStatement stmt = connection.prepareCall(sql)) {
    stmt.setInt(2, artObjectId);
    stmt.registerOutParameter(1, java.sql.Types.INTEGER);

    stmt.execute();
    return stmt.getInt(1);
} catch (SQLException e) {
    logger.severe("Error getting exhibition count for ArtObject with id: " +
artObjectId + " - " + e.getMessage());
    throw e;
}
}

public void saveDrawing(int id, byte[] drawing) throws SQLException {
    String sql = "UPDATE ArtObject SET photo = ? WHERE Id = ?";
    try (Connection connection = getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setBytes(1, drawing);
        statement.setInt(2, id);
        statement.executeUpdate();
    } catch (SQLException ex) {
        logger.log(Level.SEVERE, "Error saving drawing", ex);
        throw ex;
    }
}
}

```

## **ExhibitionDAO.Java**

```

package org.servlet.museum_management.DAO;

import org.servlet.museum_management.model.ArtObject;
import org.servlet.museum_management.model.Exhibition;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

public class ExhibitionDAO extends AbstractDAO {

```

```

public ExhibitionDAO(String jdbcURL, String jdbcUsername, String
    jdbcPassword) {
    super(jdbcURL, jdbcUsername, jdbcPassword);
}

private static final Logger logger =
    Logger.getLogger(ExhibitionDAO.class.getName()); // Определяем поле логера

public boolean insertExhibition(Exhibition exhibition) throws SQLException {
    String sql = "INSERT INTO Exhibition (Name, Description, Started, Ended,
    MuseumId) VALUES (?, ?, ?, ?, ?)";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowInserted = false;

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        // Проверка существования музея
        if (!museumExists(exhibition.getMuseumId())) {
            throw new SQLException("Museum with ID " + exhibition.getMuseumId()
+ " does not exist.");
        }

        statement = connection.prepareStatement(sql);
        statement.setString(1, exhibition.getName());
        statement.setString(2, exhibition.getDescription());
        statement.setDate(3, exhibition.getStarted());
        statement.setDate(4, exhibition.getEnded());
        statement.setInt(5, exhibition.getMuseumId());

        rowInserted = statement.executeUpdate() > 0;
        connection.commit(); // Подтверждение транзакции

        logger.info("Exhibition inserted: " + exhibition.getName()); // Лог
транзакции
    } catch (SQLException e) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции в случае ошибки
        }
        logger.log(Level.SEVERE, "Error inserting exhibition", e); // Логи ошибок
транзакции
        throw e; // Переброс исключения для обработки в сервлете
    } finally {

```

```

        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true); // Восстановление автокоммита
            connection.close();
        }
    }

    return rowInserted;
}

private boolean museumExists(int museumId) throws SQLException {
    String sql = "SELECT COUNT(*) FROM Museum WHERE id = ?";
    try (Connection connection = getConnection(); PreparedStatement statement =
        connection.prepareStatement(sql)) {
        statement.setInt(1, museumId);
        ResultSet resultSet = statement.executeQuery();
        if (resultSet.next()) {
            return resultSet.getInt(1) > 0;
        }
        return false;
    }
}

public List<Exhibition> listAllExhibitions(Integer museumId) throws
    SQLException {
    List<Exhibition> listExhibition = new ArrayList<>();
    String sql = "SELECT * FROM exhibition WHERE museumId = ?";
    logger.info("Listing all exhibitions for museumId: " + museumId);

    try (Connection connection = getConnection()) {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, museumId);
        ResultSet resultSet = statement.executeQuery();

        while (resultSet.next()) {
            Exhibition exhibition = convertExhibition(resultSet);
            listExhibition.add(exhibition);
        }

        resultSet.close();
        statement.close();
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "Error listing exhibitions for museumId: " +

```

```

museumId, e);
    e.printStackTrace();
}

return listExhibition;
}

public List<Exhibition> listAllExhibitions() throws SQLException {
    List<Exhibition> listExhibition = new ArrayList<>();
    String sql = "SELECT * FROM exhibition";

    try (Connection connection = getConnection()) {
        ResultSet resultSet = connection.createStatement().executeQuery(sql);

        while (resultSet.next()) {
            Exhibition exhibition = convertExhibition(resultSet);
            listExhibition.add(exhibition);
        }

        resultSet.close();
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "Error listing all exhibitions", e);
        e.printStackTrace();
    }

    return listExhibition;
}

public boolean deleteExhibition(Exhibition exhibition) throws SQLException {
    String sql = "DELETE FROM exhibition where id = ?";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowDeleted = false;

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        statement = connection.prepareStatement(sql);
        statement.setInt(1, exhibition.getId());

        rowDeleted = statement.executeUpdate() > 0;
        connection.commit(); // Подтверждение транзакции

        logger.info("Exhibition deleted: " + exhibition.getId());
    }

```



```

    } catch (SQLException e) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции в случае ошибки
        }
        logger.log(Level.SEVERE, "Error deleting exhibition: " + exhibition.getId(),
e);
        e.printStackTrace();
    } finally {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true); // Восстановление автокоммита
            connection.close();
        }
    }

    return rowDeleted;
}

```

```

public boolean updateExhibition(Exhibition exhibition) throws SQLException {
    String sql = "UPDATE exhibition SET name = ?, description = ?, started = ?,
ended = ?, museumId = ? WHERE id = ?";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowUpdated = false;

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        statement = connection.prepareStatement(sql);
        statement.setString(1, exhibition.getName());
        statement.setString(2, exhibition.getDescription());
        statement.setDate(3, exhibition.getStarted());
        statement.setDate(4, exhibition.getEnded());
        statement.setInt(5, exhibition.getMuseumId());
        statement.setInt(6, exhibition.getId());

        rowUpdated = statement.executeUpdate() > 0;
        connection.commit(); // Подтверждение транзакции

        logger.info("Exhibition updated: " + exhibition.getName());
    } catch (SQLException e) {
        if (connection != null) {

```

```

        connection.rollback(); // Откат транзакции в случае ошибки
    }
    logger.log(Level.SEVERE, "Error updating exhibition: " +
exhibition.getName(), e);
    e.printStackTrace();
} finally {
    if (statement != null) {
        statement.close();
    }
    if (connection != null) {
        connection.setAutoCommit(true); // Восстановление автокоммита
        connection.close();
    }
}

return rowUpdated;
}

```

```

public Exhibition getExhibition(int id) throws SQLException {
    Exhibition exhibition = null;
    String sql = "SELECT * FROM exhibition WHERE id = ?";
    logger.info("Getting exhibition with id: " + id);

    try (Connection connection = getConnection()) {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, id);
        ResultSet resultSet = statement.executeQuery();

        if (resultSet.next()) {
            exhibition = convertExhibition(resultSet);
        }

        resultSet.close();
        statement.close();
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "Error getting exhibition with id: " + id, e);
        e.printStackTrace();
    }

    return exhibition;
}

```

```

public static Exhibition convertExhibition(ResultSet resultSet) throws
SQLException {
    int id = resultSet.getInt("id");

```

```

String name = resultSet.getString("name");
String description = resultSet.getString("description");
Date started = resultSet.getDate("started");
Date ended = resultSet.getDate("ended");
Integer museumId = resultSet.getInt("museumId");

return new Exhibition(id, name, description, started, ended, museumId);
}

public List<ArtObject> listExhibitedArtObjects(int exhibitionId) throws
SQLException {
    List<ArtObject> artObjects = new ArrayList<>();
    String sql = "SELECT ao.* FROM ArtObject ao " +
        "INNER JOIN ArtObjectOnExhibition aoe ON ao.Id = aoe.ArtObjectId " +
        "WHERE aoe.ExhibitionId = ?";
    logger.info("Listing art objects for exhibitionId: " + exhibitionId);

    try (Connection connection = getConnection()) {
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, exhibitionId);
        ResultSet resultSet = statement.executeQuery();

        while (resultSet.next()) {
            ArtObject artObject = ArtObjectDAO.convertArtObject(resultSet);
            artObjects.add(artObject);
        }

        resultSet.close();
        statement.close();
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "Error listing art objects for exhibitionId: " +
exhibitionId, e);
        e.printStackTrace();
    }

    return artObjects;
}

public void unbindArtObjectFromExhibition(int artObjectId, int exhibitionId)
throws SQLException {
    String sql = "{ call UnbindArtObjectFromExhibition(?, ?) }";
    Connection connection = null;
    logger.info("Unbinding art object with id: " + artObjectId + " from exhibitionId:
" + exhibitionId);

```

```

try {
    connection = getConnection();
    connection.setAutoCommit(false); // Начало транзакции

    try (CallableStatement stmt = connection.prepareCall(sql)) {
        stmt.setInt(1, artObjectId);
        stmt.setInt(2, exhibitionId);

        stmt.execute();
        connection.commit(); // Подтверждение транзакции

        logger.info("Art object with id: " + artObjectId + " unbound from
exhibitionId: " + exhibitionId);
    }
} catch (SQLException e) {
    if (connection != null) {
        connection.rollback(); // Откат транзакции в случае ошибки
    }
    logger.log(Level.SEVERE, "Error unbinding art object from exhibition", e);
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.setAutoCommit(true); // Восстановление автокоммита
        connection.close();
    }
}
}

public void attachArtObjectFromExhibition(int artObjectId, int exhibitionId)
throws SQLException {
    String sql = "{ call AttachArtObjectFromExhibition(?, ?) }";
    Connection connection = null;
    logger.info("Attaching art object with id: " + artObjectId + " to exhibitionId: " +
exhibitionId);

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        try (CallableStatement stmt = connection.prepareCall(sql)) {
            stmt.setInt(1, artObjectId);
            stmt.setInt(2, exhibitionId);

            stmt.execute();
            connection.commit(); // Подтверждение транзакции

```

```

        logger.info("Art object with id: " + artObjectId + " attached to
exhibitionId: " + exhibitionId);
    }
} catch (SQLException e) {
    if (connection != null) {
        connection.rollback(); // Откат транзакции в случае ошибки
    }
    logger.log(Level.SEVERE, "Error attaching art object to exhibition", e);
    e.printStackTrace();
} finally {
    if (connection != null) {
        connection.setAutoCommit(true); // Восстановление автокоммита
        connection.close();
    }
}
}
}

```

```

public boolean exists(int museumId) throws SQLException {
    String sql = "SELECT COUNT(*) FROM Museum WHERE id = ?";
    try (Connection connection = getConnection();
        PreparedStatement statement = connection.prepareStatement(sql)) {
        statement.setInt(1, museumId);
        ResultSet resultSet = statement.executeQuery();
        if (resultSet.next()) {
            return resultSet.getInt(1) > 0;
        }
    }
    return false;
}
}

```

## **MuseumDAO.Java**

```

package org.servlet.museum_management.DAO;

import org.servlet.museum_management.model.Museum;

import java.sql.*;
import java.util.ArrayList;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

```

```

public class MuseumDAO extends AbstractDAO {

    public MuseumDAO(String jdbcURL, String jdbcUsername, String jdbcPassword)
    {
        super(jdbcURL, jdbcUsername, jdbcPassword);
    }

    private static final Logger logger =
        Logger.getLogger(MuseumDAO.class.getName());

    private static Museum convertMuseum(ResultSet resultSet) throws SQLException
    {
        int id = resultSet.getInt("id");
        String name = resultSet.getString("name");
        String description = resultSet.getString("description");
        String address = resultSet.getString("address");
        Time TimeOpening = resultSet.getTime("TimeOpening");
        Time TimeClosing = resultSet.getTime("TimeClosing");

        Museum museum = new Museum(id, name, description, address, TimeOpening,
        TimeClosing);
        return museum;
    }

    public boolean insertMuseum(Museum museum) throws SQLException {
        String sql = "INSERT INTO Museum (Name, Description, Address,
        TimeOpening, TimeClosing) VALUES (?, ?, ?, ?, ?)";
        Connection connection = null;
        PreparedStatement statement = null;
        boolean rowInserted = false;

        logger.info("Inserting museum: " + museum.getName());

        try {
            connection = getConnection();
            connection.setAutoCommit(false); // Начало транзакции

            statement = connection.prepareStatement(sql);
            statement.setString(1, museum.getName());
            statement.setString(2, museum.getDescription());
            statement.setString(3, museum.getAddress());
            statement.setTime(4, museum.getTimeOpening());
            statement.setTime(5, museum.getTimeClosing());

            rowInserted = statement.executeUpdate() > 0;
        }
    }
}

```

```

        connection.commit(); // Подтверждение транзакции

        logger.info("Museum inserted: " + museum.getName());
    } catch (SQLException e) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции в случае ошибки
        }
        logger.log(Level.SEVERE, "Error inserting museum", e);
        e.printStackTrace();
    } finally {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true); // Восстановление автокоммита
            connection.close();
        }
    }

    return rowInserted;
}

public List<Museum> listAllMuseums() throws SQLException {
    List<Museum> listMuseum = new ArrayList<>();
    String sql = "SELECT * FROM MuseumWithExhibitionCount";

    logger.info("Listing all museums");

    try (Connection connection = getConnection()) {
        try (Statement statement = connection.createStatement()) {
            try (ResultSet resultSet = statement.executeQuery(sql)) {
                while (resultSet.next()) {
                    Museum museum = convertMuseum(resultSet);
                    museum.setExhibitionCount(resultSet.getInt("ExhibitionCount"));
                    listMuseum.add(museum);
                }
            }
        }
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "Error listing all museums", e);
        throw e;
    }

    return listMuseum;
}

```

```

public boolean deleteMuseum(Museum museum) throws SQLException {
    String sql = "DELETE FROM museum where id = ?";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowDeleted = false;

    logger.info("Deleting museum: " + museum.getName());

    try {
        connection = getConnection();
        connection.setAutoCommit(false); // Начало транзакции

        statement = connection.prepareStatement(sql);
        statement.setInt(1, museum.getId());

        rowDeleted = statement.executeUpdate() > 0;
        connection.commit(); // Подтверждение транзакции

        logger.info("Museum deleted: " + museum.getName());
    } catch (SQLException e) {
        if (connection != null) {
            connection.rollback(); // Откат транзакции в случае ошибки
        }
        logger.log(Level.SEVERE, "Error deleting museum", e);
        e.printStackTrace();
    } finally {
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.setAutoCommit(true); // Восстановление автокоммита
            connection.close();
        }
    }

    return rowDeleted;
}

```

```

public boolean updateMuseum(Museum museum) throws SQLException {
    String sql = "UPDATE museum SET name = ?, description = ?, address = ?,
    TimeOpening = ?, TimeClosing = ? WHERE id = ?";
    Connection connection = null;
    PreparedStatement statement = null;
    boolean rowUpdated = false;

```



```

logger.info("Updating museum: " + museum.getName());

try {
    connection = getConnection();
    connection.setAutoCommit(false); // Начало транзакции

    statement = connection.prepareStatement(sql);
    statement.setString(1, museum.getName());
    statement.setString(2, museum.getDescription());
    statement.setString(3, museum.getAddress());
    statement.setTime(4, museum.getTimeOpening());
    statement.setTime(5, museum.getTimeClosing());
    statement.setInt(6, museum.getId());

    rowUpdated = statement.executeUpdate() > 0;
    connection.commit(); // Подтверждение транзакции

    logger.info("Museum updated: " + museum.getName());
} catch (SQLException e) {
    if (connection != null) {
        connection.rollback(); // Откат транзакции в случае ошибки
    }
    logger.log(Level.SEVERE, "Error updating museum", e);
    e.printStackTrace();
} finally {
    if (statement != null) {
        statement.close();
    }
    if (connection != null) {
        connection.setAutoCommit(true); // Восстановление автокоммита
        connection.close();
    }
}

return rowUpdated;
}

public Museum getMuseum(int id) throws SQLException {
    Museum museum = null;
    String sql = "SELECT * FROM museum WHERE id = ?";

    logger.info("Getting museum with id: " + id);

    try (Connection connection = getConnection()) {

```

```

        PreparedStatement statement = connection.prepareStatement(sql);
        statement.setInt(1, id);
        ResultSet resultSet = statement.executeQuery();
        if (resultSet.next()) {
            museum = convertMuseum(resultSet);
        }
        resultSet.close();
        statement.close();
    } catch (SQLException e) {
        logger.log(Level.SEVERE, "Error getting museum with id: " + id, e);
        throw e;
    }

    return museum;
}
}

```

## **ArtObject.Java**

```

package org.servlet.museum_management.model;

import java.nio.charset.StandardCharsets;
import java.util.Base64;

public class ArtObject {

    private Integer id;
    private String name;
    private String description;
    private Integer creationYear;
    private String artist;
    private Integer museumId;
    private byte[] photo;

    public ArtObject() {
    }

    public ArtObject(Integer id, String name, String description, Integer creationYear,
        String artist, Integer museumId, byte[] photo) {
        this.id = id;
        this.name = name;
        this.description = description;
        this.creationYear = creationYear;
    }
}

```

```

    this.artist = artist;
    this.museumId = museumId;
    this.photo = photo;
}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public Integer getCreationYear() {
    return creationYear;
}

public void setCreationYear(Integer creationYear) {
    this.creationYear = creationYear;
}

public String getArtist() {
    return artist;
}

```

```

public void setArtist(String artist) {
    this.artist = artist;
}

public Integer getMuseumId() {
    return museumId;
}

public void setMuseumId(Integer museumId) {
    this.museumId = museumId;
}

public byte[] getPhoto() {
    return photo;
}

public String getPhotoBase64() {
    if (photo == null) {
        return null;
    } else {
        byte[] encodeBase64 = Base64.getEncoder().encode(photo);
        return new String(encodeBase64, StandardCharsets.UTF_8);
    }
}

public void setPhoto(byte[] photo) {
    this.photo = photo;
}
}

```

## **Exhibition.Java**

```

package org.servlet.museum_management.model;

import java.sql.Date;

public class Exhibition {

    private Integer id;
    private String name;
    private String description;

```

```

private Date started;
private Date ended;
private Integer museumId;

public Exhibition(Integer id, String name, String description, Date started, Date
ended, Integer museumId) {
    this.id = id;
    this.name = name;
    this.description = description;
    this.started = started;
    this.ended = ended;
    this.museumId = museumId;
}

public Exhibition() {

}

public Integer getId() {
    return id;
}

public void setId(Integer id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

```

```

public Date getStarted() {
    return started;
}

public void setStarted(Date started) {
    this.started = started;
}

public Date getEnded() {
    return ended;
}

public void setEnded(Date ended) {
    this.ended = ended;
}

public Integer getMuseumId() {
    return museumId;
}

public void setMuseumId(Integer museumId) {
    this.museumId = museumId;
}
}

```

## **Museum.Java**

```

package org.servlet.museum_management.model;

import java.sql.Time;

public class Museum {
    private Integer Id;
    private String Name;
    private String Description;
    private String Address;
    private Time TimeOpening;
    private Time TimeClosing;
    private Integer exhibitionCount;

    public Museum() {
    }

    public Museum(Integer Id, String Name, String Description, String Address, Time

```

```

TimeOpening, Time TimeClosing) {
    this.Id = Id;
    this.Name = Name;
    this.Description = Description;
    this.Address = Address;
    this.TimeOpening = TimeOpening;
    this.TimeClosing = TimeClosing;
}

public int getId() {
    return this.Id;
}

public void setId(int value) {
    this.Id = value;
}

public String getName() {
    return this.Name;
}

public void setName(String value) {
    this.Name = value;
}

public String getDescription() {
    return this.Description;
}

public void setDescription(String value) {
    this.Description = value;
}

public String getAddress() {
    return this.Address;
}

public void setAddress(String value) {
    this.Address = value;
}

public Time getTimeOpening() {
    return this.TimeOpening;
}

```

```

    public void setTimeOpening(Time value) {
        this.TimeOpening = value;
    }

    public Time getTimeClosing() {
        return this.TimeClosing;
    }

    public void setTimeClosing(Time value) {
        this.TimeClosing = value;
    }

    public Integer getExhibitionCount() {
        return exhibitionCount;
    }

    public void setExhibitionCount(Integer exhibitionCount) {
        this.exhibitionCount = exhibitionCount;
    }
}

```

### **ArtObjectServlet.Java**

```

package org.servlet.museum_management.Servlet;

import org.servlet.museum_management.DAO.ArtObjectDAO;
import org.servlet.museum_management.DAO.MuseumDAO;
import org.servlet.museum_management.model.ArtObject;
import org.servlet.museum_management.model.Exhibition;
import org.servlet.museum_management.model.Museum;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.MultipartConfig;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.sql.SQLException;
import java.util.List;

```



```

@MultipartConfig
@WebServlet("/art-objects/*")
public class ArtObjectServlet extends HttpServlet {
    private ArtObjectDAO artObjectDAO;
    private MuseumDAO museumDAO;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("jdbcURL");
        String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
        String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

        artObjectDAO = new ArtObjectDAO(jdbcURL, jdbcUsername, jdbcPassword);
        museumDAO = new MuseumDAO(jdbcURL, jdbcUsername, jdbcPassword);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        doGet(request, response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("UTF-8");
        String action = Utils.getAction(request);

        try {
            if (request.getParameter("museum-id") != null) {
                Integer museumId = Integer.valueOf(request.getParameter("museum-id"));
                Museum museum = museumDAO.getMuseum(museumId);
                request.setAttribute("museum", museum);
            }

            switch (action) {
                case "/new":
                    showNewForm(request, response);
                    break;
                case "/insert":
                    insertArtObject(request, response);
                    break;
                case "/delete":
                    deleteArtObject(request, response);
                    break;
                case "/edit":
                    showEditForm(request, response);
            }
        }
    }
}

```

```

        break;
    case "/update":
        updateArtObject(request, response);
        break;
    case "/exhibitions-taked-place":
        listExhibitionsWhereAOTakedPlace(request, response);
        break;
    case "/exhibitions-count":
        showExhibitonCount(request, response);
        break;
    case "/list":
    default:
        listArtObjects(request, response);
        break;
    }
} catch (SQLException ex) {
    throw new ServletException(ex);
}
}

private void listArtObjects(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    Museum museum = (Museum) request.getAttribute("museum");
    List<ArtObject> artObjects;
    if (museum != null) {
        artObjects = artObjectDAO.listAllArtObjects(museum.getId());
    } else {
        artObjects = artObjectDAO.listAllArtObjects();
    }
    request.setAttribute("artObjects", artObjects);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/art-
objects/ArtObjectList.jsp");
    dispatcher.forward(request, response);
}

private void showNewForm(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/art-
objects/ArtObjectForm.jsp");
    dispatcher.forward(request, response);
}

private void showEditForm(HttpServletRequest request, HttpServletResponse

```

```

response)
    throws SQLException, ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    ArtObject existingArtObject = artObjectDAO.getArtObject(id);
    request.setAttribute("artObject", existingArtObject);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/art-
objects/ArtObjectForm.jsp");
    dispatcher.forward(request, response);
}

private void insertArtObject(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    ArtObject newArtObject = createArtObjectObj(request, null);
    artObjectDAO.insertArtObject(newArtObject);
    response.sendRedirect("list");
}

private void updateArtObject(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    request.setCharacterEncoding("UTF-8");
    Integer id = Integer.parseInt(request.getParameter("id"));
    ArtObject updatedArtObject = createArtObjectObj(request, id);
    artObjectDAO.updateArtObject(updatedArtObject,
updatedArtObject.getPhoto().length > 0);
    response.sendRedirect("list");
}

private void deleteArtObject(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    ArtObject artObject = new ArtObject();
    artObject.setId(id);
    artObjectDAO.deleteArtObject(artObject);
    response.sendRedirect("list");
}

private static ArtObject createArtObjectObj(HttpServletRequest request, Integer
id) throws ServletException, IOException {
    String name = request.getParameter("name");
    String description = request.getParameter("description");
    Integer creationYear = Integer.valueOf(request.getParameter("creationYear"));
    String artist = request.getParameter("artist");

```

```

Integer museumId = Integer.valueOf(request.getParameter("museumId"));
byte[] photo = readAllBytes(request.getPart("photo").getInputStream());

ArtObject artObject = new ArtObject();
artObject.setId(id);
artObject.setName(name);
artObject.setDescription(description);
artObject.setCreationYear(creationYear);
artObject.setArtist(artist);
artObject.setMuseumId(museumId);
artObject.setPhoto(photo);
return artObject;
}

private static byte[] readAllBytes(InputStream inputStream) throws IOException {
    final int bufLen = 4 * 0x400; // 4KB
    byte[] buf = new byte[bufLen];
    int readLen;
    IOException exception = null;

    try {
        try (ByteArrayOutputStream outputStream = new ByteArrayOutputStream())
        {
            while ((readLen = inputStream.read(buf, 0, bufLen)) != -1)
                outputStream.write(buf, 0, readLen);

            return outputStream.toByteArray();
        }
    } catch (IOException e) {
        exception = e;
        throw e;
    } finally {
        if (exception == null) inputStream.close();
        else try {
            inputStream.close();
        } catch (IOException e) {
            exception.addSuppressed(e);
        }
    }
}

private void listExhibitionsWhereAOTakedPlace(HttpServletRequest request,
        HttpServletResponse response)
        throws SQLException, IOException, ServletException {
    int id = Integer.parseInt(request.getParameter("id"));

```

```

        ArtObject existingArtObject = artObjectDAO.getArtObject(id);
        request.setAttribute("artObject", existingArtObject);
        List<Exhibition> listExhibition;
        listExhibition = artObjectDAO.getExhibitionsByArtObjectId(id);
        request.setAttribute("listExhibitionTakedPlace", listExhibition);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/art-
objects/OnExhibitionList.jsp");
        dispatcher.forward(request, response);
    }

    private void showExhibitonCount(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException, SQLException {
        int id = Integer.parseInt(request.getParameter("id"));
        ArtObject existingArtObject = artObjectDAO.getArtObject(id);
        request.setAttribute("artObject", existingArtObject);
        int exhibitionsCount = artObjectDAO.getExhibitionsCountByArtObjectId(id);
        request.setAttribute("exhibitionsCount", exhibitionsCount);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/art-
objects/ExhibitionCount.jsp");
        dispatcher.forward(request, response);
    }
}

```

## **ExhibitionReportServlet.Java**

```

package org.servlet.museum_management.Servlet;

import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import org.servlet.museum_management.DAO.ExhibitionDAO;
import org.servlet.museum_management.model.Exhibition;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

```

```

@WebServlet("/ExhibitionReportServlet")
public class ExhibitionReportServlet extends HttpServlet {
    private ExhibitionDAO exhibitionDAO;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("jdbcURL");
        String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
        String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

        exhibitionDAO = new ExhibitionDAO(jdbcURL, jdbcUsername,
        jdbcPassword);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        try {
            // Установите путь к директории со шрифтами
            String fontDir = getServletContext().getRealPath("/WEB-INF/classes/fonts");
            System.setProperty("jasperreports.fonts.dir", fontDir);

            List<Exhibition> exhibitions = exhibitionDAO.listAllExhibitions();
            JRBeanCollectionDataSource dataSource = new
            JRBeanCollectionDataSource(exhibitions);

            String reportPath = getServletContext().getRealPath("/WEB-
            INF/reports/ExhibitionReport.jrxml");
            JasperReport jasperReport =
            JasperCompileManager.compileReport(reportPath);

            Map<String, Object> parameters = new HashMap<>();
            parameters.put("ReportTitle", "Exhibition Report");
            parameters.put("CurrentDate", new Date()); // Передача текущей даты
            String logoPath = getServletContext().getRealPath("/WEB-
            INF/reports/logo.png");
            parameters.put("LogoPath", logoPath); // Передача пути к логотипу

            JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport,
            parameters, dataSource);

            response.setContentType("application/pdf");
            response.setHeader("Content-Disposition", "attachment;
            filename=ExhibitionReport.pdf");

            JasperExportManager.exportReportToPdfStream(jasperPrint,

```

```

        response.getOutputStream());
    } catch (SQLException | JRException e) {
        throw new ServletException(e);
    }
}
}

```

## **ExhibitionServlet.Java**

```

package org.servlet.museum_management.Servlet;

import org.servlet.museum_management.DAO.ArtObjectDAO;
import org.servlet.museum_management.DAO.ExhibitionDAO;
import org.servlet.museum_management.DAO.MuseumDAO;
import org.servlet.museum_management.model.ArtObject;
import org.servlet.museum_management.model.Exhibition;
import org.servlet.museum_management.model.Museum;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.Date;
import java.sql.SQLException;
import java.util.List;

@WebServlet("/exhibitions/*")
public class ExhibitionServlet extends HttpServlet {
    private ExhibitionDAO exhibitionDAO;
    private MuseumDAO museumDAO;
    private ArtObjectDAO artObjectDAO;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("jdbcURL");
        String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
        String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

        exhibitionDAO = new ExhibitionDAO(jdbcURL, jdbcUsername,
        jdbcPassword);
        museumDAO = new MuseumDAO(jdbcURL, jdbcUsername, jdbcPassword);
        artObjectDAO = new ArtObjectDAO(jdbcURL, jdbcUsername, jdbcPassword);
    }
}

```

```

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    doGet(request, response);
}

```

```

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String action = Utils.getAction(request);

```

```

    try {
        if (request.getParameter("museum-id") != null) {
            Integer museumId = Integer.valueOf(request.getParameter("museum-id"));
            Museum museum = museumDAO.getMuseum(museumId);
            request.setAttribute("museum", museum);
        }

```

```

        switch (action) {
            case "/new":
                showNewForm(request, response);
                break;
            case "/insert":
                insertExhibition(request, response);
                break;
            case "/delete":
                deleteExhibition(request, response);
                break;
            case "/edit":
                showEditForm(request, response);
                break;
            case "/update":
                updateExhibition(request, response);
                break;
            case "/exhibited-ao":
                listExhibitedArtObjects(request, response);
                break;
            case "/remove-ao-from-ex":
                removeArtObjectFromExhibition(request, response);
                break;
            case "/select-ao-to-attach":
                listArtObjectsCanBeAttachedToExhibition(request, response);
                break;
            case "/attach-to-exhibition":

```



```

        attachArtObjectFromExhibition(request, response);
        break;
    case "/list":
    default:
        listExhibition(request, response);
        break;
    }
} catch (SQLException ex) {
    request.setAttribute("javax.servlet.error.exception", ex);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/error.jsp");
    dispatcher.forward(request, response);
}
}

```

```

private void listExhibition(HttpServletRequest request, HttpServletResponse response)

```

```

    throws SQLException, IOException, ServletException {
    Museum museum = (Museum) request.getAttribute("museum");
    List<Exhibition> listExhibition;
    if (museum != null) {
        listExhibition = exhibitionDAO.listAllExhibitions(museum.getId());
    } else {
        listExhibition = exhibitionDAO.listAllExhibitions();
    }
    request.setAttribute("listExhibition", listExhibition);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/exhibitions/ExhibitionList.jsp");
    dispatcher.forward(request, response);
}

```

```

private void showNewForm(HttpServletRequest request, HttpServletResponse response)

```

```

    throws ServletException, IOException {
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/exhibitions/ExhibitionForm.jsp");
    dispatcher.forward(request, response);
}

```

```

private void showEditForm(HttpServletRequest request, HttpServletResponse response)

```

```

    throws SQLException, ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    Exhibition existingExhibition = exhibitionDAO.getExhibition(id);
    request.setAttribute("museum",

```

```

museumDAO.getMuseum(existingExhibition.getMuseumId()));
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/exhibitions/ExhibitionForm.jsp");
    request.setAttribute("exhibition", existingExhibition);
    dispatcher.forward(request, response);
}

private void insertExhibition(HttpServletRequest request, HttpServletResponse response)
    throws SQLException, IOException, ServletException {
    try {
        Exhibition newExhibition = createExhibitionObject(request, null);
        exhibitionDAO.insertExhibition(newExhibition);
        response.sendRedirect("list?museum-id=" + newExhibition.getMuseumId());
    } catch (SQLException e) {
        request.setAttribute("javax.servlet.error.exception", e);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/error.jsp");
        dispatcher.forward(request, response);
    }
}

private void updateExhibition(HttpServletRequest request, HttpServletResponse response)
    throws SQLException, IOException, ServletException {
    try {
        request.setCharacterEncoding("UTF-8");
        Integer id = Integer.parseInt(request.getParameter("id"));
        Exhibition updatedExhibition = createExhibitionObject(request, id);
        exhibitionDAO.updateExhibition(updatedExhibition);
        response.sendRedirect("list?museum-id=" + updatedExhibition.getMuseumId());
    } catch (SQLException e) {
        request.setAttribute("javax.servlet.error.exception", e);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/error.jsp");
        dispatcher.forward(request, response);
    }
}

private void deleteExhibition(HttpServletRequest request, HttpServletResponse response)
    throws SQLException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));

```

```

    Exhibition exhibition = new Exhibition();
    exhibition.setId(id);
    exhibitionDAO.deleteExhibition(exhibition);
    response.sendRedirect("list");
}

private static Exhibition createExhibitionObject(HttpServletRequest request,
Integer id) {
    String name = request.getParameter("name");
    String description = request.getParameter("description");
    Date started = null;
    if (!request.getParameter("started").equals("")) {
        started = Date.valueOf(request.getParameter("started"));
    }
    Date ended = null;
    if (!request.getParameter("ended").equals("")) {
        ended = Date.valueOf(request.getParameter("ended"));
    }
    Integer museumId = Integer.valueOf(request.getParameter("museum-id"));

    return new Exhibition(id, name, description, started, ended, museumId);
}

private void listExhibitedArtObjects(HttpServletRequest request,
    HttpServletResponse response)
    throws SQLException, IOException, ServletException {
    int exhibitionId = Integer.parseInt(request.getParameter("id"));
    Exhibition exhibition = exhibitionDAO.getExhibition(exhibitionId);
    request.setAttribute("exhibition", exhibition);
    Museum museum = museumDAO.getMuseum(exhibition.getMuseumId());
    request.setAttribute("museum", museum);

    List<ArtObject> exhibitedArtObjects =
    exhibitionDAO.listExhibitedArtObjects(exhibitionId);
    request.setAttribute("exhibitedArtObjects", exhibitedArtObjects);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-INF/exhibitions/ExhibitedArtObjects.jsp");
    dispatcher.forward(request, response);
}

private void removeArtObjectFromExhibition(HttpServletRequest request,
    HttpServletResponse response)
    throws IOException, SQLException {
    int exhibitionId = Integer.parseInt(request.getParameter("ex-id"));
    int artObjectId = Integer.parseInt(request.getParameter("ao-id"));

```

```

        exhibitionDAO.unbindArtObjectFromExhibition(artObjectId, exhibitionId);
        response.sendRedirect(request.getHeader("referer"));
    }

    private void listArtObjectsCanBeAttachedToExhibition(HttpServletRequest request,
        HttpServletResponse response)
        throws SQLException, IOException, ServletException {
        int exhibitionId = Integer.parseInt(request.getParameter("ex-id"));
        Exhibition exhibition = exhibitionDAO.getExhibition(exhibitionId);
        request.setAttribute("exhibition", exhibition);

        Museum museum = museumDAO.getMuseum(exhibition.getMuseumId());
        request.setAttribute("museum", museum);

        List<ArtObject> artObjects;
        artObjects =
        artObjectDAO.listAllArtObjectsNotAttachedToExhibition(exhibition.getMuseum
            Id(), exhibition.getId());
        request.setAttribute("artObjects", artObjects);
        RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-
            INF/exhibitions/AttachArtObjectToExhibition.jsp");
        dispatcher.forward(request, response);
    }

    private void attachArtObjectFromExhibition(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, SQLException {
        int exhibitionId = Integer.parseInt(request.getParameter("ex-id"));
        int artObjectId = Integer.parseInt(request.getParameter("ao-id"));
        exhibitionDAO.attachArtObjectFromExhibition(artObjectId, exhibitionId);
        response.sendRedirect("/exhibitions/exhibited-ao?id=" + exhibitionId);
    }
}

```

## ExportToExcelServlet.Java

```

package org.servlet.museum_management.Servlet;

import org.apache.poi.ss.usermodel.*;
import org.apache.poi.xssf.usermodel.*;
import org.apache.poi.ss.util.CellRangeAddress;
import org.apache.poi.xddf.usermodel.chart.*;

import org.servlet.museum_management.DAO.MuseumDAO;
import org.servlet.museum_management.model.Museum;

```

```

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.util.List;

@WebServlet("/ExportToExcelServlet")
public class ExportToExcelServlet extends HttpServlet {
    private MuseumDAO museumDAO;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("jdbcURL");
        String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
        String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

        museumDAO = new MuseumDAO(jdbcURL, jdbcUsername, jdbcPassword);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("application/vnd.openxmlformats-officedocument.spreadsheetml.sheet");
        response.setHeader("Content-Disposition", "attachment; filename=museums.xlsx");

        try (Workbook workbook = new XSSFWorkbook()) {
            Sheet sheet = workbook.createSheet("Museums");
            createHeaderRow(sheet);

            List<Museum> listMuseums = museumDAO.listAllMuseums();
            int rowCount = 1;

            for (Museum museum : listMuseums) {
                Row row = sheet.createRow(rowCount++);
                writeMuseum(museum, row);
            }

            // Create a sheet for charts
            XSSFSheet chartSheet = (XSSFSheet) workbook.createSheet("Charts");
            createChart(chartSheet, listMuseums);

            workbook.write(response.getOutputStream());

```

```

    } catch (Exception e) {
        throw new ServletException("Exception in Excel Export", e);
    }
}

```

```

private void createHeaderRow(Sheet sheet) {
    Row headerRow = sheet.createRow(0);
    CellStyle headerStyle = sheet.getWorkbook().createCellStyle();
    Font font = sheet.getWorkbook().createFont();
    font.setBold(true);
    headerStyle.setFont(font);

    Cell cell = headerRow.createCell(0);
    cell.setCellValue("ID");
    cell.setCellStyle(headerStyle);

    cell = headerRow.createCell(1);
    cell.setCellValue("Name");
    cell.setCellStyle(headerStyle);

    cell = headerRow.createCell(2);
    cell.setCellValue("Description");
    cell.setCellStyle(headerStyle);

    cell = headerRow.createCell(3);
    cell.setCellValue("Address");
    cell.setCellStyle(headerStyle);

    cell = headerRow.createCell(4);
    cell.setCellValue("Opening Time");
    cell.setCellStyle(headerStyle);

    cell = headerRow.createCell(5);
    cell.setCellValue("Closing Time");
    cell.setCellStyle(headerStyle);

    cell = headerRow.createCell(6);
    cell.setCellValue("Number of Exhibitions");
    cell.setCellStyle(headerStyle);
}

```

```

private void writeMuseum(Museum museum, Row row) {
    Cell cell = row.createCell(0);
    cell.setCellValue(museum.getId());
}

```

```

        cell = row.createCell(1);
        cell.setCellValue(museum.getName());

        cell = row.createCell(2);
        cell.setCellValue(museum.getDescription());

        cell = row.createCell(3);
        cell.setCellValue(museum.getAddress());

        cell = row.createCell(4);
        cell.setCellValue(museum.getTimeOpening());

        cell = row.createCell(5);
        cell.setCellValue(museum.getTimeClosing());

        cell = row.createCell(6);
        cell.setCellValue(museum.getExhibitionCount());
    }

    private void createChart(XSSFSheet chartSheet, List<Museum> listMuseums) {
        System.out.println("Creating chart with data size: " + listMuseums.size());

        // Заголовки
        Row headerRow = chartSheet.createRow(0);
        headerRow.createCell(0).setCellValue("Museum Name");
        headerRow.createCell(1).setCellValue("Number of Exhibitions");

        int rowNum = 1;
        for (Museum museum : listMuseums) {
            Row row = chartSheet.createRow(rowNum++);
            row.createCell(0).setCellValue(museum.getName());
            row.createCell(1).setCellValue(museum.getExhibitionCount());
        }

        try {
            XSSFDrawing drawing = chartSheet.createDrawingPatriarch();

            // Создаем линейный график
            XSSFClientAnchor lineAnchor = drawing.createAnchor(0, 0, 0, 0, 5, 10,
20);
            XSSFChart lineChart = drawing.createChart(lineAnchor);
            XDDFChartLegend lineLegend = lineChart.getOrAddLegend();
            lineLegend.setPosition(LegendPosition.BOTTOM);

            XDDFCategoryAxis lineBottomAxis =

```

```

lineChart.createCategoryAxis(AxisPosition.BOTTOM);
    XDDFValueAxis lineLeftAxis =
lineChart.createValueAxis(AxisPosition.LEFT);
    lineLeftAxis.setCrosses(AxisCrosses.AUTO_ZERO);

    XDDFDataSource<String> xs =
XDDFDataSourcesFactory.fromStringCellRange(chartSheet, new
CellRangeAddress(1, listMuseums.size(), 0, 0));
    XDDFNumericalDataSource<Double> ys =
XDDFDataSourcesFactory.fromNumericCellRange(chartSheet, new
CellRangeAddress(1, listMuseums.size(), 1, 1));

    XDDFLineChartData lineData = (XDDFLineChartData)
lineChart.createData(ChartTypes.LINE, lineBottomAxis, lineLeftAxis);
    XDDFLineChartData.Series lineSeries = (XDDFLineChartData.Series)
lineData.addSeries(xs, ys);
    lineSeries.setTitle("Number of Exhibitions", null);
    lineSeries.setMarkerStyle(MarkerStyle.CIRCLE);

    lineChart.plot(lineData);

    // Создаем гистограмму
    XSSFClientAnchor barAnchor = drawing.createAnchor(0, 0, 0, 0, 12, 5, 22,
20); // Задаем другое положение
    XSSFChart barChart = drawing.createChart(barAnchor);
    XDDFChartLegend barLegend = barChart.getOrAddLegend();
    barLegend.setPosition(LegendPosition.BOTTOM);

    XDDFCategoryAxis barBottomAxis =
barChart.createCategoryAxis(AxisPosition.BOTTOM);
    XDDFValueAxis barLeftAxis =
barChart.createValueAxis(AxisPosition.LEFT);
    barLeftAxis.setCrosses(AxisCrosses.AUTO_ZERO);

    XDDFBarChartData barData = (XDDFBarChartData)
barChart.createData(ChartTypes.BAR, barBottomAxis, barLeftAxis);
    XDDFBarChartData.Series barSeries = (XDDFBarChartData.Series)
barData.addSeries(xs, ys);
    barSeries.setTitle("Number of Exhibitions", null);
    barData.setBarDirection(BarDirection.COL);

    barChart.plot(barData);

} catch (Exception e) {
    e.printStackTrace();
}

```



```

    }
}

}

```

## **MuseumServlet.Java**

```

package org.servlet.museum_management.Servlet;

import org.servlet.museum_management.DAO.MuseumDAO;
import org.servlet.museum_management.model.Museum;

import java.io.IOException;
import java.sql.SQLException;
import java.sql.Time;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.List;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.time.format.DateTimeParseException;

@WebServlet("/museums/*")
public class MuseumServlet extends HttpServlet {
    private MuseumDAO museumDAO;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("jdbcURL");
        String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
        String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

        museumDAO = new MuseumDAO(jdbcURL, jdbcUsername, jdbcPassword);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        request.setCharacterEncoding("UTF-8");
    }
}

```

```

doGet(request, response);
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    request.setCharacterEncoding("UTF-8");
    String action = Utils.getAction(request);

    try {
        switch (action) {
            case "/new":
                showNewForm(request, response);
                break;
            case "/insert":
                insertMuseum(request, response);
                break;
            case "/delete":
                deleteMuseum(request, response);
                break;
            case "/edit":
                showEditForm(request, response);
                break;
            case "/update":
                updateMuseum(request, response);
                break;
            case "/list":
            default:
                listMuseum(request, response);
                break;
        }
    } catch (SQLException ex) {
        throw new ServletException(ex);
    }
}

private void listMuseum(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException, ServletException {
    List<Museum> listMuseum = museumDAO.listAllMuseums();
    request.setAttribute("listMuseum", listMuseum);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-
INF/museums/MuseumList.jsp");
    dispatcher.forward(request, response);
}

```

```

private void showNewForm(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-
INF/museums/MuseumForm.jsp");
    dispatcher.forward(request, response);
// request.getRequestDispatcher("MuseumForm.jsp").include(request, response);
}

private void showEditForm(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    Museum existingMuseum = museumDAO.getMuseum(id);
    RequestDispatcher dispatcher = request.getRequestDispatcher("/WEB-
INF/museums/MuseumForm.jsp");
    request.setAttribute("museum", existingMuseum);
    dispatcher.forward(request, response);
}

private void insertMuseum(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException {
    Museum newMuseum = convertMuseum(request);
    museumDAO.insertMuseum(newMuseum);
    response.sendRedirect("list");
}

private void updateMuseum(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException {
    Museum museum = convertMuseum(request);
    museumDAO.updateMuseum(museum);
    response.sendRedirect("list");
}

private void deleteMuseum(HttpServletRequest request, HttpServletResponse
response)
    throws SQLException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));

    Museum museum = new Museum();
    museum.setId(id);
    museumDAO.deleteMuseum(museum);
    response.sendRedirect("list");
}

```

```

    }

    private static Museum convertMuseum(HttpServletRequest request) {
        Integer id = null;
        if (request.getParameter("id")!=null) {
            id = Integer.parseInt(request.getParameter("id"));
        }
        String name = request.getParameter("name");
        String description = request.getParameter("description");
        String address = request.getParameter("address");
        DateTimeFormatter timeFormatter =
        DateTimeFormatter.ofPattern("HH:mm[:ss]");
        Time TimeOpening =
        Time.valueOf(LocalTime.parse(request.getParameter("time_opened"),
        timeFormatter));
        Time TimeClosing =
        Time.valueOf(LocalTime.parse(request.getParameter("time_closed"),
        timeFormatter));

        Museum museum = new Museum(id, name, description, address, TimeOpening,
        TimeClosing);
        return museum;
    }
}

```

## PictureDrawServlet.Java

```

package org.servlet.museum_management.Servlet;

import java.io.IOException;
import java.util.Base64;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.servlet.museum_management.DAO.ArtObjectDAO;
import org.servlet.museum_management.model.ArtObject;
import java.sql.SQLException;

@WebServlet("/art-objects/draw")
public class PictureDrawServlet extends HttpServlet {
    private ArtObjectDAO artObjectDAO;

```

```

public void init() {
    String jdbcURL = getServletContext().getInitParameter("jdbcURL");
    String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
    String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

    artObjectDAO = new ArtObjectDAO(jdbcURL, jdbcUsername, jdbcPassword);
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    ArtObject artObject = null;
    try {
        artObject = artObjectDAO.getArtObject(id);
    } catch (SQLException e) {
        throw new ServletException("Error retrieving art object", e);
    }
    request.setAttribute("artObject", artObject);
    request.getRequestDispatcher("/WEB-INF/art-objects/ArtObjectDraw.jsp").forward(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    String drawingDataURL = request.getParameter("drawing");

    if (drawingDataURL == null || !drawingDataURL.startsWith("data:image/")) {
        throw new ServletException("Invalid image data");
    }

    byte[] drawingData =
        Base64.getDecoder().decode(drawingDataURL.split(",")[1]);

    try {
        artObjectDAO.saveDrawing(id, drawingData);
    } catch (SQLException e) {
        throw new ServletException("Error saving drawing", e);
    }

    response.sendRedirect("list");
}
}

```

## ReportServlet.Java

```

package org.servlet.museum_management.Servlet;

import net.sf.jasperreports.engine.*;
import net.sf.jasperreports.engine.data.JRBeanCollectionDataSource;
import org.servlet.museum_management.DAO.MuseumDAO;
import org.servlet.museum_management.model.Museum;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;
import java.util.Date;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@WebServlet("/ReportServlet")
public class ReportServlet extends HttpServlet {
    private MuseumDAO museumDAO;

    public void init() {
        String jdbcURL = getServletContext().getInitParameter("jdbcURL");
        String jdbcUsername = getServletContext().getInitParameter("jdbcUsername");
        String jdbcPassword = getServletContext().getInitParameter("jdbcPassword");

        museumDAO = new MuseumDAO(jdbcURL, jdbcUsername, jdbcPassword);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        try {
            // Установите путь к директории со шрифтами
            String fontDir = getServletContext().getRealPath("/WEB-INF/classes/fonts");
            System.setProperty("jasperreports.fonts.dir", fontDir);

            List<Museum> museums = museumDAO.listAllMuseums();
            JRBeanCollectionDataSource dataSource = new
            JRBeanCollectionDataSource(museums);

            String reportPath = getServletContext().getRealPath("/WEB-
            INF/reports/MuseumReport.jrxml");

```

```

        JasperReport jasperReport =
JasperCompileManager.compileReport(reportPath);

        Map<String, Object> parameters = new HashMap<>();
        parameters.put("ReportTitle", "Museum Report");
        parameters.put("CurrentDate", new Date()); // Передача текущей даты
        String logoPath = getServletContext().getRealPath("/WEB-
INF/reports/logo.png");
        parameters.put("LogoPath", logoPath); // Передача пути к логотипу

        JasperPrint jasperPrint = JasperFillManager.fillReport(jasperReport,
parameters, dataSource);

        response.setContentType("application/pdf");
        response.setHeader("Content-Disposition", "attachment;
filename=MuseumReport.pdf");

        JasperExportManager.exportReportToPdfStream(jasperPrint,
response.getOutputStream());
    } catch (SQLException | JRException e) {
        throw new ServletException(e);
    }
}
}
}

```

## **Utils.Java**

```

package org.servlet.museum_management.Servlet;

import javax.servlet.http.HttpServletRequest;

public class Utils {
    public static String getAction(HttpServletRequest request) {
//        String action = request.getServletPath();
        String action = request.getPathInfo();
        if (action == null) {
            action = "";
        } else {
            action = action.replaceAll("/$", "");
        }
        return action;
    }
}

```

## ArtObjectDraw.jsp

```
<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"% >
<html>
<head>
    <meta charset="UTF-8">
    <title>Рисование на изображении</title>
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 50px;
        }
        .container {
            text-align: center;
        }
        canvas {
            border: 1px solid black;
            margin-top: 20px;
        }
        .btn {
            margin-top: 20px;
        }
    </style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
    <a class="navbar-brand" href="#">Музейное приложение</a>
</nav>
<div class="container">
    <h1>Рисование на изображении</h1>
    <div style="position: relative;">
        <canvas id="artCanvas" width="350" height="350"></canvas>
    </div>
    <button type="button" class="btn btn-success"
        onclick="saveDrawing()">Сохранить изменения</button>
</div>

<script>
    var canvas = document.getElementById('artCanvas');
    var ctx = canvas.getContext('2d');
    var painting = false;
```



```

var img = new Image();
img.src = 'data:image/jpeg;base64,{artObject.photoBase64}';
img.onload = function() {
    ctx.drawImage(img, 0, 0, canvas.width, canvas.height);
};

function startPosition(e) {
    painting = true;
    draw(e);
}

function endPosition() {
    painting = false;
    ctx.beginPath();
}

function draw(e) {
    if (!painting) return;

    ctx.lineWidth = 5;
    ctx.lineCap = 'round';
    ctx.strokeStyle = 'red';

    var rect = canvas.getBoundingClientRect();
    var x = e.clientX - rect.left;
    var y = e.clientY - rect.top;

    ctx.lineTo(x, y);
    ctx.stroke();
    ctx.beginPath();
    ctx.moveTo(x, y);
}

canvas.addEventListener('mousedown', startPosition);
canvas.addEventListener('mouseup', endPosition);
canvas.addEventListener('mousemove', draw);

function saveDrawing() {
    var drawingDataURL = canvas.toDataURL('image/jpeg');
    var form = document.createElement('form');
    form.method = 'post';
    form.action = '<%= request.getContextPath() %>/art-objects/draw';

    var inputId = document.createElement('input');
    inputId.type = 'hidden';

```

```

        inputId.name = 'id';
        inputId.value = '${artObject.id}';
        form.appendChild(inputId);

        var inputDrawing = document.createElement('input');
        inputDrawing.type = 'hidden';
        inputDrawing.name = 'drawing';
        inputDrawing.value = drawingDataURL;
        form.appendChild(inputDrawing);

        document.body.appendChild(form);
        form.submit();
    }
</script>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
    pt>
</body>
</html>

```

### ArtObjectForm.jsp

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"% >
<html>
<head>
    <title>Форма Произведения Искусства</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9;
            margin: 0;
            padding: 0;
        }
        .container {
            width: 50%;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        }
    </style>

```

```

    border-radius: 8px;
    margin-top: 50px;
}
h1, h3 {
    color: #333;
}
.form-table {
    width: 100%;
    border-collapse: collapse;
}
.form-table th, .form-table td {
    padding: 10px;
    text-align: left;
}
.form-table th {
    width: 30%;
    background-color: #f4f4f9;
}
.form-table input[type="text"],
.form-table input[type="number"],
.form-table input[type="file"],
.form-table textarea {
    width: calc(100% - 20px);
    padding: 8px;
    margin: 5px 0;
    border: 1px solid #ccc;
    border-radius: 4px;
}
.form-table .buttons {
    text-align: center;
    padding-top: 20px;
}
.form-table .buttons input[type="submit"], .form-table .buttons a {
    display: inline-block;
    padding: 10px 20px;
    margin: 10px 5px;
    border: none;
    background-color: #5cb85c;
    color: white;
    text-decoration: none;
    border-radius: 4px;
    cursor: pointer;
}
.form-table .buttons a {
    background-color: #d9534f;

```

```

    }
</style>
</head>
<body>
<div class="container">
  <center>
    <h1>Форма для работы с произведениями искусства</h1>
    <h2>
      <c:if test="\${museum != null}">
        <a href="/art-objects/?museum-id=<c:out value="\${museum.id}"/>">
          Список произведений искусства в музее «<c:out
value="\${museum.name}"/>»
        </a>
      </c:if>
      <c:if test="\${museum == null}">
        <a href="/art-objects/">
          Список произведений искусства
        </a>
      </c:if>
    </h2>
  </center>
  <div>
    <c:if test="\${artObject != null}">
      <form action="update" method="post" enctype='multipart/form-data'>
    </c:if>
    <c:if test="\${artObject == null}">
      <form action="insert" method="post" enctype='multipart/form-data'>
    </c:if>
    <table class="form-table">
      <caption>
        <h3>
          <c:if test="\${artObject != null}">
            Редактирование произведения искусства
          </c:if>
          <c:if test="\${artObject == null}">
            Добавление произведения искусства
          </c:if>
        </h3>
      </caption>
      <c:if test="\${artObject != null}">
        <input type="hidden" name="id" value="<c:out
value="\${artObject.id}"/>" />
      </c:if>
      <tr>
        <th>Название:</th>

```

```

        <td>
            <input type="text" name="name" size="45" value="<c:out
value='${artObject.name}'/>"/>
        </td>
    </tr>
    <tr>
        <th>Описание:</th>
        <td>
            <textarea name="description" rows="5" cols="42"><c:out
value='${artObject.description}'/></textarea>
        </td>
    </tr>
    <tr>
        <th>Год создания:</th>
        <td>
            <input type="number" name="creationYear" value="<c:out
value='${artObject.creationYear}'/>"/>
        </td>
    </tr>
    <tr>
        <th>Художник:</th>
        <td>
            <input type="text" name="artist" value="<c:out
value='${artObject.artist}'/>"/>
        </td>
    </tr>
    <tr>
        <th>Идентификатор музея:</th>
        <td>
            <c:if test="${museum != null}">
                <input type="number" name="museumId" value="<c:out
value='${museum.id}'/>"/>
            </c:if>
            <c:if test="${museum == null}">
                <input type="number" name="museumId" value="<c:out
value='${artObject.museumId}'/>"/>
            </c:if>
        </td>
    </tr>
    <tr>
        <th>Фото:</th>
        <td>
            <input type="file" name="photo"/>
        </td>
    </tr>

```

```

        <tr class="buttons">
            <td colspan="2" class="buttons">
                <input type="submit" value="Сохранить"/>
                <a href="/art-objects/">Отменить</a>
            </td>
        </tr>
    </table>
</form>
</div>
</div>
</body>
</html>

```

## ArtObjectList.jsp

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
    <meta charset="UTF-8">
    <title>Список произведений искусства</title>
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <link rel="stylesheet"
        href="https://fonts.googleapis.com/css2?family=Roboto:wght@400;700&display=swap">
    <style>
        body {
            padding-top: 50px;
            font-family: 'Roboto', sans-serif;
        }
        .navbar {
            margin-bottom: 20px;
        }
        .table th, .table td {
            text-align: center;
            vertical-align: middle;
        }
        .btn {
            margin-top: 5px;
        }
        .actions {
            display: flex;
            flex-direction: column;

```

```

        align-items: center;
    }
    h1, h2 {
        font-weight: 700;
    }
    h3 {
        font-weight: 400;
    }
</style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <a class="navbar-brand" href="#">Музейное приложение</a>
        <div class="collapse navbar-collapse">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/museums/list">Список музеев</a>
                </li>
                <c:choose>
                    <c:when test="{museum != null}">
                        <li class="nav-item">
                            <a class="nav-link" href="/exhibitions/listlist?museum-id=<c:out
value='{museum.id}'/>">
                                Список выставок в музее «<c:out
value="{museum.name}" />»
                            </a>
                        </li>
                    </c:when>
                    <c:otherwise>
                        <li class="nav-item">
                            <a class="nav-link" href="/exhibitions/list">Список выставок</a>
                        </li>
                    </c:otherwise>
                </c:choose>
                <c:choose>
                    <c:when test="{museum != null}">
                        <li class="nav-item">
                            <a class="nav-link" href="/art-objects/new?museum-id=<c:out
value='{museum.id}'/>">
                                Добавить произведение искусства в музей «<c:out
value="{museum.name}" />»
                            </a>
                        </li>
                    </c:when>
                    <c:otherwise>

```

```

        <li class="nav-item">
            <a class="nav-link" href="/art-objects/new">Добавить
произведение искусства</a>
        </li>
    </c:otherwise>
</c:choose>
</ul>
</div>
</nav>

<div class="container">
    <h1 class="text-center">Список произведений искусства</h1>

    <div class="table-responsive">
        <c:choose>
            <c:when test="{empty artObjects}">
                <h3 class="text-center">Произведения искусства в музее "<c:out
value="{museum.id}"/>" не найдены</h3>
            </c:when>
            <c:otherwise>
                <table class="table table-striped table-bordered">
                    <caption><h2 class="text-center">
                        <c:choose>
                            <c:when test="{museum != null}">
                                Список произведений искусства в музее «<c:out
value="{museum.name}"/>»
                            </c:when>
                            <c:otherwise>
                                Список произведений искусства
                            </c:otherwise>
                        </c:choose>
                    </h2></caption>
                    <thead class="thead-dark">
                        <tr>
                            <th>ID</th>
                            <th>Название</th>
                            <th>Описание</th>
                            <th>Год создания</th>
                            <th>Id музея</th>
                            <th>Фото</th>
                            <th>Действия</th>
                        </tr>
                    </thead>
                    <tbody>
                        <c:forEach var="artObject" items="{artObjects}">

```



```

<tr>
  <td><c:out value="\${artObject.id}"/></td>
  <td><c:out value="\${artObject.name}"/></td>
  <td><c:out value="\${artObject.description}"/></td>
  <td><c:out value="\${artObject.creationYear}"/></td>
  <td><c:out value="\${artObject.museumId}"/></td>
  <td>
    <c:if test="\${not empty artObject.photo}">
      
    </c:if>
  </td>
  <td class="actions">
    <a href="/art-objects/exhibitions-taked-place?id=<c:out
value='\${artObject.id}' />" class="btn btn-info btn-sm">Список выставок</a>
    <a href="/art-objects/exibitions-count?id=<c:out
value='\${artObject.id}' />" class="btn btn-info btn-sm">Количество
выставок</a>
    <a href="/art-objects/edit?id=<c:out value='\${artObject.id}'
/>" class="btn btn-warning btn-sm">Изменить</a>
    <a href="/art-objects/draw?id=<c:out
value='\${artObject.id}' />" class="btn btn-primary btn-sm">Рисовать на
изображении</a>
    <a href="/art-objects/delete?id=<c:out
value='\${artObject.id}' />" class="btn btn-danger btn-sm">Удалить</a>
  </td>
</tr>
</c:forEach>
</tbody>
</table>
</c:otherwise>
</c:choose>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
</body>
</html>

```

## ExhibitionCount.jsp

```
<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"% >
<html>
<head>
    <meta charset="UTF-8">
    <title>Музейное приложение</title>
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 50px;
        }
        .navbar {
            margin-bottom: 20px;
        }
        .container {
            text-align: center;
        }
    </style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
    <a class="navbar-brand" href="#">Музейное приложение</a>
</nav>
<div class="container">
    <h1 class="mt-5">Управление произведениями искусства</h1>
    <h2 class="my-4">
        <a href="/museums/list" class="btn btn-primary">Список музеев</a>
    </h2>
    <div class="my-4">
        <h2>Количество выставок, в котором принимало участие произведение
            искусства "<c:out value='${artObject.name}'/>": ${exhibitionsCount}</h2>
    </div>
    <a href="javascript:history.back()" class="btn btn-secondary">Назад</a>
</div>
<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
```

```
</body>
</html>
```

## OnExhibitionList.jsp

```
<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"% >
<html>
<head>
    <meta charset="UTF-8">
    <title>Музейное приложение</title>
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 50px;
        }
        .navbar {
            margin-bottom: 20px;
        }
        .table th, .table td {
            text-align: center;
        }
        .btn {
            margin-top: 5px;
        }
        .actions {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
        .content {
            text-align: center;
        }
    </style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <a class="navbar-brand" href="#">Музейное приложение</a>
        <div class="collapse navbar-collapse">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/museums/list">Список музеев</a>
                </li>
```

```

        </ul>
    </div>
</nav>

<div class="container content">
    <h1 class="text-center">Управление произведениями искусства</h1>
    <h3 class="text-center">Выставки, в которых принимало участие
произведение искусства "<c:out value='${artObject.name}'/>"</h3>
    <c:choose>
        <c:when test="${listExhibitionTakedPlace.size() == 0}">
            <h3 class="text-center">Выставки, в которых принимало участие
произведение искусства "<c:out value='${artObject.name}'/>" не
найжены</h3>
        </c:when>
        <c:otherwise>
            <div class="table-responsive">
                <table class="table table-striped table-bordered">
                    <thead class="thead-dark">
                        <tr>
                            <th>ID</th>
                            <th>Название</th>
                            <th>Описание</th>
                            <th>Начало</th>
                            <th>Конец</th>
                            <th>Id музея</th>
                            <th>Действия</th>
                        </tr>
                    </thead>
                    <tbody>
                        <c:forEach var="exhibition"
items="${listExhibitionTakedPlace}">
                            <tr>
                                <td><c:out value='${exhibition.id}'/></td>
                                <td><c:out value='${exhibition.name}'/></td>
                                <td><c:out value='${exhibition.description}'/></td>
                                <td><c:out value='${exhibition.started}'/></td>
                                <td><c:out value='${exhibition.ended}'/></td>
                                <td><c:out value='${exhibition.museumId}'/></td>
                                <td>
                                    <a href="/exhibitions/remove-ao-from-ex?ex-
id=${exhibition.id}&ao-id=<c:out value='${artObject.id}' />" class="btn btn-
danger btn-sm">
                                        Убрать с выставки
                                    </a>
                                </td>
                            </tr>
                        </c:forEach>
                    </tbody>
                </table>
            </div>
        </c:otherwise>
    </c:choose>

```

```

        </tr>
      </c:forEach>
    </tbody>
  </table>
</div>
</c:otherwise>
</c:choose>
<div class="text-center mt-4">
  <a href="javascript:history.back()" class="btn btn-secondary">Назад</a>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
</body>
</html>

```

### **AttachArtObjectToExhibition.jsp**

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
  <meta charset="UTF-8">
  <title>Привязка произведения искусства к выставке</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  <style>
    body {
      padding-top: 50px;
    }
    .navbar {
      margin-bottom: 20px;
    }
    .table th, .table td {
      text-align: center;
    }
    .btn {
      margin-top: 5px;
    }
  </style>

```

```

    }
    .actions {
        display: flex;
        flex-direction: column;
        align-items: center;
    }
</style>
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
        <a class="navbar-brand" href="#">Музейное приложение</a>
        <div class="collapse navbar-collapse">
            <ul class="navbar-nav ml-auto">
                <li class="nav-item">
                    <a class="nav-link" href="/museums/list">Список музеев</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/exhibitions/listlist?museum-id=<c:out
value='${museum.id}'/>">
                        Список выставок в музее «<c:out value='${museum.name}'/>»
                    </a>
                </li>
                <li class="nav-item">
                    <a class="nav-link" href="/art-objects/new?museum-id=<c:out
value='${museum.id}'/>">
                        Добавить произведение искусства в музей «<c:out
value='${museum.name}'/>»
                    </a>
                </li>
            </ul>
        </div>
    </nav>

    <div class="container">
        <h1 class="text-center">Привязка произведения искусства к выставке
        «<c:out value='${exhibition.name}'/>»</h1>
        <div class="table-responsive">
            <c:choose>
                <c:when test='${empty artObjects}'>
                    <h3 class="text-center">Произведения искусства в музее "<c:out
value='${museum.name}'/>", которые еще не привязаны к выставке «<c:out
value='${exhibition.name}'/>», не найдены</h3>
                </c:when>
                <c:otherwise>
                    <table class="table table-striped table-bordered">

```

<caption><h2 class="text-center">Список произведений искусства  
в музее "<c:out value='\${museum.name}'/>", которые еще не привязаны к  
выставке «<c:out value="\${exhibition.name}" />»</h2></caption>

<thead class="thead-dark">

<tr>

<th>ID</th>

<th>Название</th>

<th>Описание</th>

<th>Год создания</th>

<th>Id музея</th>

<th>Фото</th>

<th>Действия</th>

</tr>

</thead>

<tbody>

<c:forEach var="artObject" items="\${artObjects}">

<tr>

<td><c:out value="\${artObject.id}" /></td>

<td><c:out value="\${artObject.name}" /></td>

<td><c:out value="\${artObject.description}" /></td>

<td><c:out value="\${artObject.creationYear}" /></td>

<td><c:out value="\${artObject.museumId}" /></td>

<td>

<c:if test="\${not empty artObject.photo}">



</c:if>

</td>

<td class="actions">

<a href="/exhibitions/attach-to-exhibition?ex-

id=\${exhibition.id}&ao-id=<c:out value='\${artObject.id}' />" class="btn btn-  
primary btn-sm">

Привязать к выставке «<c:out

value="\${exhibition.name}" />»

</a>

</td>

</tr>

</c:forEach>

</tbody>

</table>

</c:otherwise>

</c:choose>

</div>

</div>

```

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
</body>
</html>

```

## ExhibitedArtObjects.jsp

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
<meta charset="UTF-8">
<title>Список произведений искусства на выставке</title>
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
<style>
body {
padding-top: 50px;
}
.navbar {
margin-bottom: 20px;
}
.table th, .table td {
text-align: center;
}
.btn {
margin-top: 5px;
}
.actions {
display: flex;
flex-direction: column;
align-items: center;
}
</style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
<a class="navbar-brand" href="#">Музейное приложение</a>
<div class="collapse navbar-collapse">

```



```

<ul class="navbar-nav ml-auto">
  <li class="nav-item">
    <a class="nav-link" href="/museums/list">Список музеев</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/exhibitions/listlist?museum-id=<c:out
value='${museum.id}'/>">
      Список выставок в музее «<c:out value='${museum.name}'/>»
    </a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/exhibitions/select-ao-to-attach?ex-
id=${exhibition.id}">
      Привязка произведения искусства к выставке «<c:out
value='${exhibition.name}'/>»
    </a>
  </li>
</ul>
</div>
</nav>

<div class="container">
  <h1 class="text-center">Список произведений искусства на выставке</h1>
  <div class="table-responsive">
    <c:choose>
      <c:when test="${exhibitedArtObjects.size() == 0}">
        <h3 class="text-center">Произведения искусства, которые
принимают участие в выставке "<c:out value='${exhibition.name}'/>" не
найжены</h3>
      </c:when>
      <c:otherwise>
        <table class="table table-striped table-bordered">
          <caption><h2 class="text-center">Произведения искусства,
которые принимают участие в выставке "<c:out
value='${exhibition.name}'/>"</h2></caption>
          <thead class="thead-dark">
            <tr>
              <th>ID</th>
              <th>Название</th>
              <th>Описание</th>
              <th>Год создания</th>
              <th>Id музея</th>
              <th>Фото</th>
              <th>Операции</th>
            </tr>

```

```

</thead>
<tbody>
  <c:forEach var="artObject" items="${exhibitedArtObjects}">
    <tr>
      <td><c:out value="${artObject.id}"/></td>
      <td><c:out value="${artObject.name}"/></td>
      <td><c:out value="${artObject.description}"/></td>
      <td><c:out value="${artObject.creationYear}"/></td>
      <td><c:out value="${artObject.museumId}"/></td>
      <td>
        <c:if test="${not empty artObject.photo}">
          
        </c:if>
      </td>
      <td class="actions">
        <a href="/exhibitions/remove-ao-from-ex?ex-
id=${exhibition.id}&ao-id=<c:out value='${artObject.id}' />" class="btn btn-
danger btn-sm">Убрать с выставки</a>
      </td>
    </tr>
  </c:forEach>
</tbody>
</table>
</c:otherwise>
</c:choose>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
</body>
</html>

```

## ExhibitionForm.jsp

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"% >
<html>

```

```

<head>
  <title>Музейное приложение</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f4f4f9;
      margin: 0;
      padding: 0;
    }
    .container {
      width: 50%;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
      border-radius: 8px;
      margin-top: 50px;
    }
    h1, h3 {
      color: #333;
    }
    .form-table {
      width: 100%;
      border-collapse: collapse;
    }
    .form-table th, .form-table td {
      padding: 10px;
      text-align: left;
    }
    .form-table th {
      width: 30%;
      background-color: #f4f4f9;
    }
    .form-table input[type="text"],
    .form-table input[type="date"],
    .form-table textarea {
      width: calc(100% - 20px);
      padding: 8px;
      margin: 5px 0;
      border: 1px solid #ccc;
      border-radius: 4px;
    }
    .form-table .buttons {
      text-align: center;
      padding-top: 20px;
    }
  </style>

```

```

    }
    .form-table .buttons input[type="submit"], .form-table .buttons a {
        display: inline-block;
        padding: 10px 20px;
        margin: 10px 5px;
        border: none;
        background-color: #5cb85c;
        color: white;
        text-decoration: none;
        border-radius: 4px;
        cursor: pointer;
    }
    .form-table .buttons a {
        background-color: #d9534f;
    }
</style>
</head>
<body>
<div class="container">
    <center>
        <h1>Управление выставками</h1>
        <h2>
            <c:if test="${museum != null}">
                <a href="/exhibitions?museum-id=<c:out value='${museum.id}'/>">
                    Список выставок в музее «<c:out value='${museum.name}'/>»
                </a>
            </c:if>
            <c:if test="${museum == null}">
                <a href="/exhibitions/">
                    Список выставок
                </a>
            </c:if>
        </h2>
    </center>
    <div align="center">
        <c:if test="${exhibition != null}">
            <form action="update" method="post">
        </c:if>
        <c:if test="${exhibition == null}">
            <form action="insert" method="post">
        </c:if>
        <table class="form-table">
            <caption>
                <h3>
                    <c:if test="${exhibition != null}">

```

```

        Редактирование данных выставки
    </c:if>
    <c:if test="\${exhibition == null}">
        Добавить выставку в музей <c:out
value="\${museum.name}"/>
    </c:if>
    </h3>
</caption>
    <c:if test="\${exhibition != null}">
        <input type="hidden" name="id" value="<c:out
value="\${exhibition.id}' />"/>
    </c:if>
    <tr>
        <th><i>Id музея</i></th>
        <td>
            <input type="text" name="museum-id" value="<c:out
value="\${museum.id}' />"/>
        </td>
    </tr>
    <tr>
        <th>Название:</th>
        <td>
            <input required type="text" name="name" size="45" value="<c:out
value="\${exhibition.name}' />"/>
        </td>
    </tr>
    <tr>
        <th>Описание:</th>
        <td>
            <textarea name="description" rows="5" cols="42"><c:out
value="\${exhibition.description}' /></textarea>
        </td>
    </tr>
    <tr>
        <th>Дата начала:</th>
        <td>
            <input type="date" name="started" value="<c:out
value="\${exhibition.started}' />"/>
        </td>
    </tr>
    <tr>
        <th>Дата окончания:</th>
        <td>
            <input type="date" name="ended" value="<c:out
value="\${exhibition.ended}' />"/>

```

```

        </td>
    </tr>
    <tr class="buttons">
        <td colspan="2" class="buttons">
            <input type="submit" value="Сохранить"/>
            <a href="/exhibitions?museum-id=<c:out
value='${museum.id}'/>">Отменить</a>
        </td>
    </tr>
</table>
</form>
</div>
</div>
</body>
</html>

```

## ExhibitionList.jsp

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"% >
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"% >
<html>
<head>
    <meta charset="UTF-8">
    <title>Музейное приложение</title>
    <link rel="stylesheet"
        href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
    <style>
        body {
            padding-top: 50px;
        }
        .navbar {
            margin-bottom: 20px;
        }
        .table th, .table td {
            text-align: center;
        }
        .btn {
            margin-top: 5px;
        }
        .actions {
            display: flex;
            flex-direction: column;
            align-items: center;
        }
    </style>

```

```

</style>
</head>
<body>
<nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
  <a class="navbar-brand" href="#">Музейное приложение</a>
</nav>

<div class="container">
  <h1 class="text-center">Управление выставками</h1>
  <div class="text-center mb-4">
    <a href="/museums/list" class="btn btn-primary">Список музеев</a>
    <c:if test="${museum != null}">
      <a href="/exhibitions/new?museum-id=<c:out value='${museum.id}'/"
class="btn btn-success">
        Добавить выставку в музей «<c:out value='${museum.name}'/>»
      </a>
    </c:if>
    <c:if test="${museum == null}">
      <a href="/exhibitions/new/" class="btn btn-success">
        Добавить выставку
      </a>
    </c:if>
    <!-- Кнопка для генерации отчета по выставкам -->
    <a href="/ExhibitionReportServlet" class="btn btn-info">Скачать отчет по
выставкам</a>
  </div>

  <div class="table-responsive">
    <c:choose>
      <c:when test="${listExhibition.size() == 0}">
        <h3 class="text-center">Выставки в музее «<c:out
value='${museum.name}'/>» не найдены</h3>
      </c:when>
      <c:otherwise>
        <table class="table table-striped table-bordered">
          <caption><h2 class="text-center">Список выставок
            <c:if test="${museum != null}">
              в музее «<c:out value='${museum.name}'/>»
            </c:if>
          </h2></caption>
          <thead class="thead-dark">
            <tr>
              <th>ID</th>
              <th>Название</th>
              <th>Описание</th>

```

```

        <th>Начало</th>
        <th>Конец</th>
        <th>Id музея</th>
        <th>Действия</th>
    </tr>
</thead>
<tbody>
    <c:forEach var="exhibition" items="${listExhibition}">
        <tr>
            <td><c:out value="${exhibition.id}"/></td>
            <td><c:out value="${exhibition.name}"/></td>
            <td><c:out value="${exhibition.description}"/></td>
            <td><c:out value="${exhibition.started}"/></td>
            <td><c:out value="${exhibition.ended}"/></td>
            <td><c:out value="${exhibition.museumId}"/></td>
            <td class="actions">
                <a href="/exhibitions/exhibited-ao?id=<c:out
value='${exhibition.id}' />" class="btn btn-info btn-sm">Произведения
искусства</a>
                <a href="/exhibitions/select-ao-to-attach?ex-
id=${exhibition.id}" class="btn btn-secondary btn-sm">Привязать
произведение искусства</a>
                <a href="/exhibitions/edit?id=<c:out value='${exhibition.id}'
/>" class="btn btn-warning btn-sm">Изменить</a>
                <a href="/exhibitions/delete?id=<c:out
value='${exhibition.id}' />" class="btn btn-danger btn-sm">Удалить</a>
            </td>
        </tr>
    </c:forEach>
</tbody>
</table>
</c:otherwise>
</c:choose>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
    src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
</body>
</html>

```



## MuseumForm.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<html>
<head>
    <title>Музейное приложение</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #f4f4f9;
            margin: 0;
            padding: 0;
        }
        .container {
            width: 50%;
            margin: 0 auto;
            padding: 20px;
            background-color: #fff;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            border-radius: 8px;
            margin-top: 50px;
        }
        h1, h3 {
            color: #333;
        }
        .form-table {
            width: 100%;
            border-collapse: collapse;
        }
        .form-table th, .form-table td {
            padding: 10px;
            text-align: left;
        }
        .form-table th {
            width: 30%;
            background-color: #f4f4f9;
        }
        .form-table input[type="text"],
        .form-table input[type="time"],
        .form-table textarea {
            width: calc(100% - 20px);
            padding: 8px;
```

```

        margin: 5px 0;
        border: 1px solid #ccc;
        border-radius: 4px;
    }
    .form-table .buttons {
        text-align: center;
        padding-top: 20px;
    }
    .form-table .buttons input[type="submit"], .form-table .buttons a {
        display: inline-block;
        padding: 10px 20px;
        margin: 10px 5px;
        border: none;
        background-color: #5cb85c;
        color: white;
        text-decoration: none;
        border-radius: 4px;
        cursor: pointer;
    }
    .form-table .buttons a {
        background-color: #d9534f;
    }
</style>
</head>
<body>
<div class="container">
    <center>
        <h1>Управление музеями</h1>
    </center>
    <div>
        <c:if test="\${museum != null}">
            <form action="update" method="post">
        </c:if>
        <c:if test="\${museum == null}">
            <form action="insert" method="post">
        </c:if>
        <table class="form-table">
            <caption>
                <h3>
                    <c:if test="\${museum != null}">
                        Редактирование данных музея
                    </c:if>
                    <c:if test="\${museum == null}">
                        Добавить музей
                    </c:if>

```

```

        </h3>
    </caption>
    <c:if test="\${museum != null}">
        <input type="hidden" name="id" value="\<c:out
value='\${museum.id}'/>"/>
    </c:if>
    <tr>
        <th>Название:</th>
        <td>
            <input type="text" name="name" size="45" value="\<c:out
value='\${museum.name}'/>"/>
        </td>
    </tr>
    <tr>
        <th>Описание:</th>
        <td>
            <textarea name="description" rows="5" cols="42">\<c:out
value='\${museum.description}'/></textarea>
        </td>
    </tr>
    <tr>
        <th>Адрес:</th>
        <td>
            <input type="text" name="address" size="45" value="\<c:out
value='\${museum.address}'/>"/>
        </td>
    </tr>
    <tr>
        <th>Время открытия:</th>
        <td>
            <input required step="60" type="time" name="time_opened"
value="\<c:out value='\${museum.timeOpening}'/>"/>
        </td>
    </tr>
    <tr>
        <th>Время закрытия:</th>
        <td>
            <input required step="60" type="time" name="time_closed"
value="\<c:out value='\${museum.timeClosing}'/>"/>
        </td>
    </tr>
    <tr class="buttons">
        <td colspan="2" class="buttons">
            <input type="submit" value="Сохранить"/>
            <a href="/museums/list">Отменить</a>
        </td>
    </tr>

```

```

        </td>
      </tr>
    </table>
  </form>
</div>
</div>
</body>
</html>

```

## MuseumList.jsp

```

<% @ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<% @ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<% @ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Музейное приложение</title>
  <link rel="stylesheet"
    href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css">
  <style>
    body {
      padding-top: 50px;
    }
    .navbar {
      margin-bottom: 20px;
    }
    .table th, .table td {
      text-align: center;
    }
    .btn {
      margin-top: 5px;
    }
    .actions {
      display: flex;
      flex-direction: column;
      align-items: center;
    }
  </style>
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark fixed-top">
    <a class="navbar-brand" href="#">Музейное приложение</a>
    <div class="collapse navbar-collapse">

```

```

<ul class="navbar-nav ml-auto">
  <li class="nav-item">
    <a class="nav-link" href="/museums/new">Добавить музей</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/exhibitions">Список всех выставок</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/art-objects/">Список всех произведений
искусства</a>
  </li>
  <li class="nav-item">
    <a class="nav-link" href="/ExportToExcelServlet">Экспортировать в
Excel</a>
  </li>
</ul>
</div>
</nav>

```

```

<div class="container">
  <h1 class="text-center">Управление музеями</h1>

  <div class="table-responsive">
    <table class="table table-striped table-bordered">
      <caption><h2 class="text-center">Список музеев</h2></caption>
      <thead class="thead-dark">
        <tr>
          <th>ID</th>
          <th>Название</th>
          <th>Описание</th>
          <th>Адрес</th>
          <th>Время открытия</th>
          <th>Время закрытия</th>
          <th>Всего выставок</th>
          <th>Действия</th>
        </tr>
      </thead>
      <tbody>
        <c:forEach var="museum" items="${listMuseum}">
          <tr>
            <td><c:out value="${museum.id}" /></td>
            <td><c:out value="${museum.name}" /></td>
            <td><c:out value="${museum.description}" /></td>
            <td><c:out value="${museum.address}" /></td>
            <td>${fn:substring( museum.timeOpening, 0, 5 )}</td>
          </tr>
        </c:forEach>
      </tbody>
    </table>
  </div>
</div>

```

```

        <td>${fn.substring( museum.timeClosing, 0, 5 )}</td>
        <td><c:out value="${museum.exhibitionCount}" /></td>
        <td class="actions">
            <a href="/exhibitions?museum-id=${museum.id}" class="btn
btn-info btn-sm">Выставки</a>
            <a href="/art-objects?museum-id=${museum.id}" class="btn
btn-info btn-sm">Произведения искусства</a>
            <a href="/museums/edit?id=${museum.id}" class="btn btn-
warning btn-sm">Изменить</a>
            <a href="/museums/delete?id=${museum.id}" class="btn btn-
danger btn-sm">Удалить</a>
        </td>
    </tr>
</c:forEach>
</tbody>
</table>
</div>
<div class="text-center">
    <a href="/ReportServlet" class="btn btn-primary mt-3">Скачать отчет</a>
</div>
</div>

<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.11.0/umd/popper.min.js">
</script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/js/bootstrap.min.js"></scri
pt>
</body>
</html>

```

## ExhibitionReport.jrxml

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "-//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
name="ExhibitionReport" pageWidth="842" pageHeight="595"
columnWidth="802" leftMargin="20" rightMargin="20" topMargin="20"
bottomMargin="20" uuid="c8e5e9a3-e8a6-4f3e-9f8e-5bafdaf0c7a3">
<parameter name="ReportTitle" class="java.lang.String"/>

```

```
<parameter name="CurrentDate" class="java.util.Date"/>
<parameter name="LogoPath" class="java.lang.String"/>
<queryString>
  <![CDATA[]]>
</queryString>
<field name="id" class="java.lang.Integer"/>
<field name="name" class="java.lang.String"/>
<field name="description" class="java.lang.String"/>
<field name="started" class="java.sql.Date"/>
<field name="ended" class="java.sql.Date"/>
<field name="museumId" class="java.lang.Integer"/>
<title>
  <band height="60">
    <image>
      <reportElement x="0" y="0" width="50" height="50"/>
      <imageExpression><![CDATA[${LogoPath}]]></imageExpression>
    </image>
    <textField>
      <reportElement x="60" y="0" width="742" height="50"/>
      <textElement textAlignment="Center">
        <font size="18" isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
      </textElement>
<textFieldExpression><![CDATA[${ReportTitle}]]></textFieldExpression>
    </textField>
  </band>
</title>
<columnHeader>
  <band height="30">
    <rectangle>
      <reportElement x="0" y="0" width="802" height="30"
backcolor="#CCCCCC" mode="Opaque"/>
    </rectangle>
    <staticText>
      <reportElement x="0" y="0" width="100" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
      </textElement>
      <text><![CDATA[ID]]></text>
    </staticText>
  </band>
</columnHeader>
</table>
```

```

    <reportElement x="100" y="0" width="1" height="30"/>
  </line>
  <staticText>
    <reportElement x="101" y="0" width="150" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
      <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
    </textElement>
    <text><![CDATA[Name]]></text>
  </staticText>
  <line>
    <reportElement x="251" y="0" width="1" height="30"/>
  </line>
  <staticText>
    <reportElement x="252" y="0" width="200" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
      <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
    </textElement>
    <text><![CDATA[Description]]></text>
  </staticText>
  <line>
    <reportElement x="452" y="0" width="1" height="30"/>
  </line>
  <staticText>
    <reportElement x="453" y="0" width="100" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
      <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
    </textElement>
    <text><![CDATA[Started]]></text>
  </staticText>
  <line>
    <reportElement x="553" y="0" width="1" height="30"/>
  </line>
  <staticText>
    <reportElement x="554" y="0" width="100" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
      <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
    </textElement>

```



```

        <text><![CDATA[Ended]]></text>
    </staticText>
    <line>
        <reportElement x="654" y="0" width="1" height="30"/>
    </line>
    <staticText>
        <reportElement x="655" y="0" width="147" height="30"/>
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
        </textElement>
        <text><![CDATA[Museum ID]]></text>
    </staticText>
</band>
</columnHeader>
<detail>
    <band height="30">
        <rectangle>
            <reportElement x="0" y="0" width="802" height="30"
backcolor="#F0F0F0" mode="Opaque"/>
        </rectangle>
        <textField>
            <reportElement x="0" y="0" width="100" height="30"/>
            <textElement textAlignment="Center" verticalAlignment="Middle">
                <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
            </textElement>
            <textFieldExpression><![CDATA[${id}]]></textFieldExpression>
        </textField>
        <line>
            <reportElement x="100" y="0" width="1" height="30"/>
        </line>
        <textField>
            <reportElement x="101" y="0" width="150" height="30"/>
            <textElement textAlignment="Center" verticalAlignment="Middle">
                <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
            </textElement>
            <textFieldExpression><![CDATA[${name}]]></textFieldExpression>
        </textField>
        <line>
            <reportElement x="251" y="0" width="1" height="30"/>
        </line>
        <textField>

```

```

    <reportElement x="252" y="0" width="200" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
    </textElement>

<textFieldExpression><![CDATA[$F{description}]]></textFieldExpression>
</textField>
<line>
    <reportElement x="452" y="0" width="1" height="30"/>
</line>
<textField>
    <reportElement x="453" y="0" width="100" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
    </textElement>
    <textFieldExpression><![CDATA[$F{started}]]></textFieldExpression>
</textField>
<line>
    <reportElement x="553" y="0" width="1" height="30"/>
</line>
<textField>
    <reportElement x="554" y="0" width="100" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
    </textElement>
    <textFieldExpression><![CDATA[$F{ended}]]></textFieldExpression>
</textField>
<line>
    <reportElement x="654" y="0" width="1" height="30"/>
</line>
<textField>
    <reportElement x="655" y="0" width="147" height="30"/>
    <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
    </textElement>

<textFieldExpression><![CDATA[$F{museumId}]]></textFieldExpression>
</textField>
</band>
</detail>
<pageFooter>

```

```

    <band height="30">
      <textField>
        <reportElement x="0" y="10" width="802" height="20"/>
        <textElement textAlignment="Right">
          <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
        </textElement>
        <textFieldExpression><![CDATA["Report generated on: " +
$P{CurrentDate}]]></textFieldExpression>
      </textField>
    </band>
  </pageFooter>
</jasperReport>

```

### **MuseumReport.jrxml**

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jasperReport PUBLIC "-//JasperReports//DTD Report Design//EN"
"http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
http://jasperreports.sourceforge.net/xsd/jasperreport.xsd" name="MuseumReport"
pageWidth="842" pageHeight="595" columnWidth="802" leftMargin="20"
rightMargin="20" topMargin="20" bottomMargin="20" uuid="c8e5e9a3-e8a6-
4f3e-9f8e-5bafdaf0c7a3">
  <parameter name="ReportTitle" class="java.lang.String"/>
  <parameter name="CurrentDate" class="java.util.Date"/>
  <parameter name="LogoPath" class="java.lang.String"/>
  <queryString>
    <![CDATA[]]>
  </queryString>
  <field name="id" class="java.lang.Integer"/>
  <field name="name" class="java.lang.String"/>
  <field name="description" class="java.lang.String"/>
  <field name="address" class="java.lang.String"/>
  <field name="timeOpening" class="java.sql.Time"/>
  <field name="timeClosing" class="java.sql.Time"/>
  <field name="exhibitionCount" class="java.lang.Integer"/>
  <title>
    <band height="60">
      <image>
        <reportElement x="0" y="0" width="50" height="50"/>
        <imageExpression><![CDATA[$P{LogoPath}]]></imageExpression>
      </image>
    </band>
  </title>

```

```

<textField>
  <reportElement x="60" y="0" width="742" height="50"/>
  <textElement textAlignment="Center">
    <font size="18" isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
  </textElement>

<textFieldExpression><![CDATA[$P{ReportTitle}]]></textFieldExpression>
</textField>
</band>
</title>
<columnHeader>
  <band height="30">
    <rectangle>
      <reportElement x="0" y="0" width="810" height="30"
backcolor="#CCCCCC" mode="Opaque"/>
    </rectangle>
    <staticText>
      <reportElement x="0" y="0" width="40" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
      </textElement>
      <text><![CDATA[ID]]></text>
    </staticText>
    <line>
      <reportElement x="40" y="0" width="1" height="30"/>
    </line>
    <staticText>
      <reportElement x="41" y="0" width="140" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
      </textElement>
      <text><![CDATA[Name]]></text>
    </staticText>
    <line>
      <reportElement x="181" y="0" width="1" height="30"/>
    </line>
    <staticText>
      <reportElement x="182" y="0" width="210" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">

```

```

        <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
        </textElement>
        <text><![CDATA[Description]]></text>
    </staticText>
    <line>
        <reportElement x="392" y="0" width="1" height="30"/>
    </line>
    <staticText>
        <reportElement x="393" y="0" width="150" height="30"/>
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
            </textElement>
            <text><![CDATA[Address]]></text>
        </staticText>
        <line>
            <reportElement x="543" y="0" width="1" height="30"/>
        </line>
        <staticText>
            <reportElement x="544" y="0" width="100" height="30"/>
            <textElement textAlignment="Center" verticalAlignment="Middle">
                <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
                </textElement>
                <text><![CDATA[Time Opening]]></text>
            </staticText>
            <line>
                <reportElement x="644" y="0" width="1" height="30"/>
            </line>
            <staticText>
                <reportElement x="645" y="0" width="100" height="30"/>
                <textElement textAlignment="Center" verticalAlignment="Middle">
                    <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
                    </textElement>
                    <text><![CDATA[Time Closing]]></text>
                </staticText>
                <line>
                    <reportElement x="745" y="0" width="1" height="30"/>
                </line>

```

```

<staticText>
  <reportElement x="746" y="0" width="65" height="30"/>
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font isBold="true" fontName="DejaVu Sans"
pdfFontName="DejaVuSans-Bold.ttf" pdfEncoding="Identity-H"
isPdfEmbedded="true"/>
  </textElement>
  <text><![CDATA[Exhibitions]]></text>
</staticText>
</band>
</columnHeader>
<detail>
  <band height="30">
    <rectangle>
      <reportElement x="0" y="0" width="810" height="30"
backcolor="#F0F0F0" mode="Opaque"/>
    </rectangle>
    <textField>
      <reportElement x="0" y="0" width="40" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
      </textElement>
      <textFieldExpression><![CDATA[$F{id}]]></textFieldExpression>
    </textField>
    <line>
      <reportElement x="40" y="0" width="1" height="30"/>
    </line>
    <textField>
      <reportElement x="41" y="0" width="140" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
      </textElement>
      <textFieldExpression><![CDATA[$F{name}]]></textFieldExpression>
    </textField>
    <line>
      <reportElement x="181" y="0" width="1" height="30"/>
    </line>
    <textField>
      <reportElement x="182" y="0" width="210" height="30"/>
      <textElement textAlignment="Center" verticalAlignment="Middle">
        <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
      </textElement>

```

```

<textFieldExpression><![CDATA[$F{description}]]></textFieldExpression>
</textField>
<line>
  <reportElement x="392" y="0" width="1" height="30"/>
</line>
<textField>
  <reportElement x="393" y="0" width="150" height="30"/>
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
  </textElement>
  <textFieldExpression><![CDATA[$F{address}]]></textFieldExpression>
</textField>
<line>
  <reportElement x="543" y="0" width="1" height="30"/>
</line>
<textField>
  <reportElement x="544" y="0" width="100" height="30"/>
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
  </textElement>
  <textFieldExpression><![CDATA[new
SimpleDateFormat("HH:mm:ss").format($F{timeOpening})]]></textFieldExpres
sion>
</textField>
<line>
  <reportElement x="644" y="0" width="1" height="30"/>
</line>
<textField>
  <reportElement x="645" y="0" width="100" height="30"/>
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
  </textElement>
  <textFieldExpression><![CDATA[new
SimpleDateFormat("HH:mm:ss").format($F{timeClosing})]]></textFieldExpress
ion>
</textField>
<line>
  <reportElement x="745" y="0" width="1" height="30"/>
</line>
<textField>
  <reportElement x="746" y="0" width="65" height="30"/>

```

```

        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
        </textElement>

<textFieldExpression><![CDATA[$F{exhibitionCount}]]></textFieldExpression
>
    </textField>
</band>
</detail>
<pageFooter>
    <band height="30">
        <textField>
            <reportElement x="0" y="10" width="802" height="20"/>
            <textElement textAlignment="Right">
                <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
            </textElement>
            <textFieldExpression><![CDATA["Report generated on: " + new
SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format($P{CurrentDate})]]></textFieldExpression>
        </textField>
        <textField>
            <reportElement x="0" y="10" width="100" height="20"/>
            <textElement textAlignment="Left">
                <font fontName="DejaVu Sans" pdfFontName="DejaVuSans.ttf"
pdfEncoding="Identity-H" isPdfEmbedded="true"/>
            </textElement>
            <textFieldExpression><![CDATA["Page " +
$V{PAGE_NUMBER}]]></textFieldExpression>
        </textField>
    </band>
</pageFooter>
</jasperReport>

```

## web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
id="WebApp_ID" version="3.1">
    <display-name>Museum Web Application</display-name>

```



```

<context-param>
  <param-name>jdbcURL</param-name>
  <!-- <param-
value>jdbc:sqlserver://10.4.130.105:1433;database=B2;trustServerCertificate=tru
e;</param-value> -->
  <param-
value>jdbc:sqlserver://localhost:1433;database=museums;trustServerCertificate=t
rue;</param-value>
</context-param>

<context-param>
  <param-name>jdbcUsername</param-name>
  <param-value>sa</param-value>
</context-param>

<context-param>
  <param-name>jdbcPassword</param-name>
  <param-value>123</param-value>
</context-param>

<!--<servlet>
  <servlet-name>ControllerServlet</servlet-name>
  <servlet-class>Servlet.servlet.museum_management.ControllerServlet</servlet-
class>
</servlet>-->

<!--  <servlet-mapping>
  <servlet-name>ControllerServlet</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>-->

  <error-page>
    <exception-type>java.lang.Exception</exception-type>
    <location>/Error.jsp</location>
  </error-page>
</web-app>

```

## Error.jsp

```

<% @ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isErrorPage="true" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>

```

```

    <title>Error</title>
</head>
<body>
<center>
    <h1>Error</h1>
    <h3>
        <%=exception.getMessage() %>
        <br/>
    </h3>
</center>
</body>
</html>

```

## **pom.xml**

```

<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>
    <groupId>servlet</groupId>
    <artifactId>museum_management</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>

    <properties>
        <tomcat.version>9.0.88</tomcat.version>
    </properties>

    <dependencies>
        <!-- POI dependencies -->
        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi</artifactId>
            <version>5.2.3</version>
        </dependency>

        <dependency>
            <groupId>org.apache.poi</groupId>
            <artifactId>poi-ooxml</artifactId>
            <version>5.2.3</version>
        </dependency>

        <dependency>

```

```

    <groupId>org.apache.xmlbeans</groupId>
    <artifactId>xmlbeans</artifactId>
    <version>5.1.1</version>
</dependency>

<dependency>
    <groupId>org.apache.poi</groupId>
    <artifactId>poi-ooxml-schemas</artifactId>
    <version>4.1.2</version>
</dependency>

<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-collections4</artifactId>
    <version>4.4</version>
</dependency>

<!-- Jasper dependencies -->
<dependency>
<groupId>net.sf.jasperreports</groupId>
<artifactId>jasperreports</artifactId>
<version>6.16.0</version>

<exclusions>
    <exclusion>
        <groupId>com.lowagie</groupId>
        <artifactId>itext</artifactId>
    </exclusion>
</exclusions>
</dependency>

<dependency>
<groupId>net.sf.jasperreports</groupId>
<artifactId>jasperreports-fonts</artifactId>
<version>6.16.0</version>
</dependency>

<dependency>
    <groupId>org.eclipse.jdt.core.compiler</groupId>
    <artifactId>ecj</artifactId>
    <version>4.6.1</version>
</dependency>

<dependency>
<groupId>com.lowagie</groupId>

```

```
<artifactId>itext</artifactId>
<version>[1.02b,2.1.7]</version>
<scope>compile</scope>
</dependency>
```

```
<!-- Other dependencies -->
```

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
```

```
<dependency>
  <groupId>jstl</groupId>
  <artifactId>jstl</artifactId>
  <version>1.2</version>
</dependency>
```

```
<dependency>
  <groupId>javax.servlet.jsp</groupId>
  <artifactId>javax.servlet.jsp-api</artifactId>
  <version>2.3.1</version>
  <scope>provided</scope>
</dependency>
```

```
<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-servlet</artifactId>
  <version>9.4.44.v20210927</version>
</dependency>
```

```
<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-server</artifactId>
  <version>9.4.44.v20210927</version>
</dependency>
```

```
<dependency>
  <groupId>org.eclipse.jetty</groupId>
  <artifactId>jetty-webapp</artifactId>
  <version>9.4.44.v20210927</version>
</dependency>
```

```

    <dependency>
      <groupId>com.microsoft.sqlserver</groupId>
      <artifactId>mssql-jdbc</artifactId>
      <version>8.2.2.jre8</version>
    </dependency>

  </dependencies>

  <build>
    <sourceDirectory>src</sourceDirectory>
    <plugins>

      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.5.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>

      <plugin>
        <artifactId>maven-war-plugin</artifactId>
        <version>3.4.0</version>
        <configuration>
          <warSourceDirectory>WebContent</warSourceDirectory>
          <failOnMissingWebXml>>false</failOnMissingWebXml>
        </configuration>
      </plugin>

      <plugin>
        <groupId>org.eclipse.jetty</groupId>
        <artifactId>jetty-maven-plugin</artifactId>
        <version>9.4.54.v20240208</version>
      </plugin>

    </plugins>
  </build>
</project>

```