МИНИСТЕРСТВО образования РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

УНИВЕРСИТЕТ информатики и радиоэлектроники»

Факультет компьютерного проектирования

Контрольная работа №2

По дисциплине: «Реализация EJB- приложения c использованием entity компонентов»

Тема: «Учет выплат по кредитам»

Выполнил студент 4-го курса факультета инновационного непрерывного образования Группа [894351](https://iis.bsuir.by/#/schedule;groupName=493551)

Галкин Илья Викторович

Зачетная книжка №6535021

Дата сдачи: 13.01.2022

Проверил: Унучек Е.Н.

 Минск 2022

**Оглавление**

[1. Краткие теоретические сведения 3](#_Toc92908031)

[2. Скриншоты работы программы 5](#_Toc92908032)

[3. Листинг программного кода 8](#_Toc92908033)

# Краткие теоретические сведения

**Java Server Pages** представляет технологию, которая позволяет создавать динамические веб-страницы. Изначально JSP (вместе с сервлетами) на заре развития Java EE являлись доминирующим подходом к веб-разработке на языке Java. И хотя в настоящее время они уступило свое место другой технологии - JSF, тем не менее JSP продолжают широко использоваться.

По сути Java Server Page или JSP представляет собой html-код с вкраплениями кода Java. В то же время станицы jsp - это не стандартные html-страницы. Когда приходит запрос к определенной странице JSP, то сервер обрабатывает ее, генерирует из нее код html и отправляет его клиенту. В итоге пользователь после обращения к странице JSP видит в своем браузере обычную html-страницу.

Как и обычные статические веб-страницы, файлы JSP необходимо размещать на веб-сервере, к которому обычные пользователи могут обращаться по протоколу http, например, набирая в адресной строке браузера нужный адрес. Однако чтобы сервер мог обрабатывать файлы JSP, он должен использовать движок JSP (**JSP engine**), который также называют JSP-контейнером. Есть множество движков JSP, и все они реализуют одну и ту же спецификацию и в целом работают одинаково. Однако тем не менее при переносе кода с одного веб-сервера на другой могут потребоваться небольшие изменения.

Hibernate – это ORM фреймворк для Java с открытым исходным кодом. Эта технология является крайне мощной и имеет высокие показатели производительности.

Hibernate создаёт связь между таблицами в базе данных (далее – БД) и Java-классами и наоборот. Это избавляет разработчиков от огромного количества лишней, рутинной работы, в которой крайне легко допустить ошибку и крайне трудно потом её найти.

Схематично это можно изобразить следующим образом:

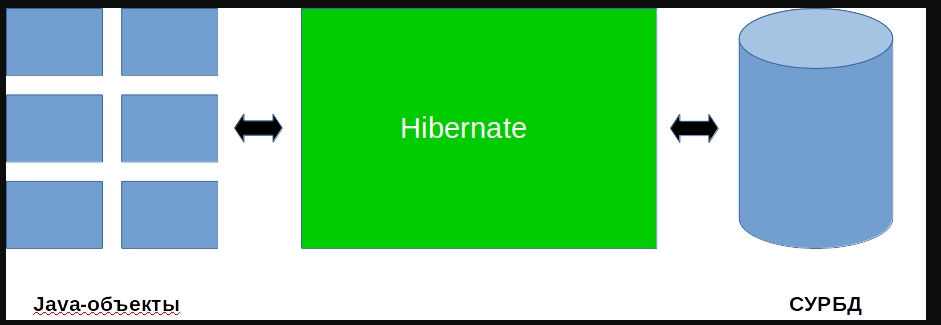


Рисунок 1 – схема работы hibernate

Какие же преимущества даёт нам использование Hibernate?

* Обеспечивает простой API для записи и получения Java-объектов в/из БД.
* Минимизирует доступ к БД, используя стратегии fetching.
* Не требует сервера приложения.
* Позволяет нам не работать с типами данных языка SQL, а иметь дело с привычыми нам типами данных Java.
* Заботится о создании связей между Java-классами и таблицами БД с помощью XML-файлов не внося изменения в программный код.
* Если нам неоходимо изменить БД, то достаточно лишь внести изменения в XML-файлы.

Hibernate поддердивает все основные СУБД: MySQL, Oracle, PostgreSQL, Microsoft SQL Server Database, HSQL, DB2.  
Hibernate также может работать в связке с такими технологиями, как Maven и J2EE.

# Скриншоты работы программы

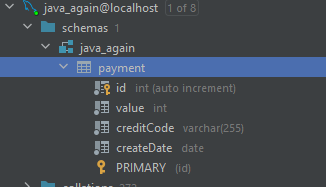


Рисунок 2 – схема базы данных приложения

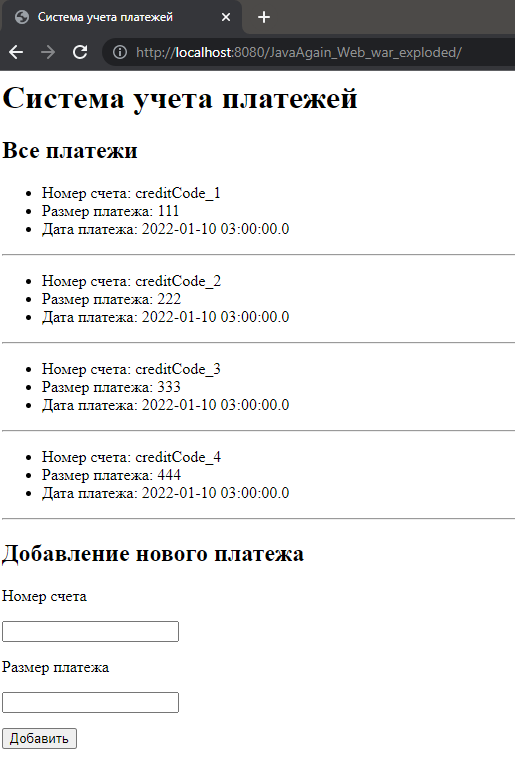


Рисунок 3 – домашняя страница приложения

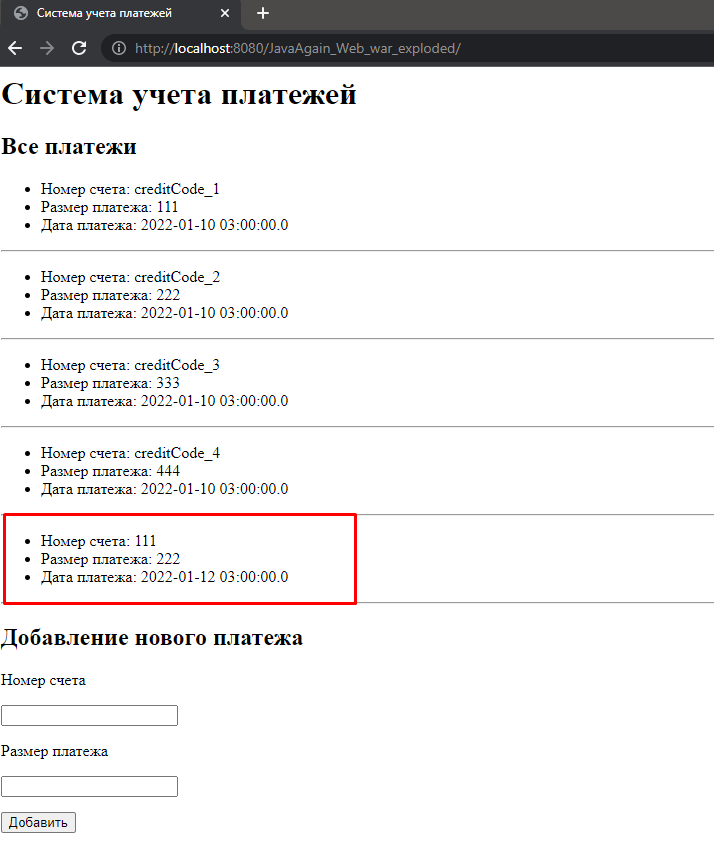


Рисунок 4 – результат добавления записи

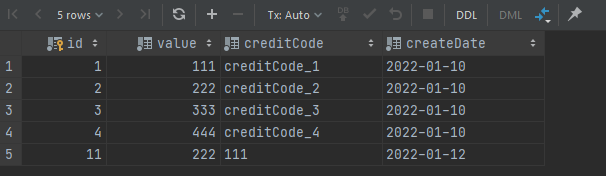


Рисунок 5 – выгрузка из базы данных

# Листинг программного кода

**PaymentBean.java**

**package** beans;

**import** **entity.Payment**;

**import** **javax.ejb.EJB**;

**import** **javax.ejb.LocalBean**;

**import** **javax.ejb.Stateless**;

**import** **javax.persistence.EntityManager**;

**import** **javax.persistence.EntityManagerFactory**;

**import** **javax.persistence.Persistence**;

**import** **javax.persistence.PersistenceContext**;

**import** **javax.persistence.criteria.CriteriaQuery**;

**import** **java.util.ArrayList**;

/\*\*

\*

\* @author ILans

\*/

**@Stateless**

**public** **class** **PaymentBean** {

**public** EntityManager **getEntityManager**() {

EntityManagerFactory emf = Persistence.createEntityManagerFactory("unit");

**return** emf.createEntityManager();

}

**public** **void** **addPayment**(Payment payment) {

EntityManager em = getEntityManager();

em.getTransaction().begin();

**if** (!em.contains(payment)) {

em.persist(payment);

em.flush();

}

em.getTransaction().commit();

}

**public** ArrayList<Payment> **getAllPayments**() {

EntityManager entityManager = getEntityManager();

CriteriaQuery criteriaQuery = entityManager.getCriteriaBuilder().createQuery();

criteriaQuery.select(criteriaQuery.from(Payment.class));

**return** **new** ArrayList<Payment>(entityManager.createQuery(criteriaQuery).getResultList());

}

}

**PaymentServlet.java**

**package** servlets;

**import** **beans.PaymentBean**;

**import** **constants.Constants**;

**import** **entity.Payment**;

**import** **javax.ejb.EJB**;

**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.ArrayList**;

**@WebServlet**(name="PaymentServlet", value = "/")

**public** **class** **PaymentServlet** **extends** HttpServlet {

// NOTE: Uncomment if server is TomEE

// @EJB

PaymentBean paymentBean = **new** PaymentBean();

**@Override**

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

ArrayList<Payment> payments = paymentBean.getAllPayments();

request.setAttribute("payments", payments);

request.getRequestDispatcher(Constants.IndexPath).forward(request, response);

}

**@Override**

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

request.setCharacterEncoding("UTF8");

System.out.print("1");

**if** (!validateRequest(request)) {

System.out.print("2");

doGet(request, response);

}

**final** String code = request.getParameter("code");

**final** String value = request.getParameter("value");

**final** Payment payment = **new** Payment(Integer.valueOf(value), code);

System.out.print("333");

paymentBean.addPayment(payment);

System.out.print("444");

doGet(request, response);

}

**private** **boolean** **validateRequest**(**final** HttpServletRequest req) {

**final** String code = req.getParameter("code");

**final** String value = req.getParameter("value");

**return** code != **null** && code.length() > **0** &&

value != **null** && value.length() > **0** &&

value.matches("[+]?\\d+");

}

}

**Index.jsp**

<%--

Document : index

Created on : Jan 10, 2022, 9:48:37 PM

Author : ILans

--%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<html>

<head>

<title>Система учета платежей</title>

</head>

<body>

<h1>Система учета платежей</h1>

<h2>Все платежи</h2>

<c:forEach var="payment" items="${requestScope.payments}">

<ul>

<li>Номер счета: <c:out value="${payment.creditCode}"/></li>

<li>Размер платежа: <c:out value="${payment.value}"/></li>

<li>Дата платежа: <c:out value="${payment.createDate}"/></li>

</ul>

<hr />

</c:forEach>

<h2>Добавление нового платежа</h2>

<form method="post">

<p>Номер счета</p>

<label>

<input type="text" name="code">

</label>

<p>Размер платежа</p>

<label>

<input type="number" name="value">

</label>

<div style="margin-top: 15px">

<input type="submit" value="Добавить" name="Ok">

</div>

</form>

</body>

</html>

**web.xml**

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"

version="4.0">

<servlet>

<servlet-name>PaymentServlet</servlet-name>

<servlet-class>servlets.PaymentServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>PaymentServlet</servlet-name>

<url-pattern></url-pattern>

</servlet-mapping>

</web-app>

**Db\_create.sql**

**CREATE** **TABLE** IF **NOT** **EXISTS** `payment` (

id INT AUTO\_INCREMENT **PRIMARY** **KEY**,

value INT **NOT** **NULL**,

creditCode VARCHAR(**255**) **NOT** **NULL**,

createDate DATE **NOT** **NULL**

);

-- test data

**insert** **into** payment(value, creditCode, createDate)

**values**

(**111**, "creditCode\_1", now()),

(**222**, "creditCode\_2", now()),

(**333**, "creditCode\_3", now()),

(**444**, "creditCode\_4", now())