**Цель работы:** Ознакомиться с технологиями Java MDB и Java JMS, научиться работе с ними; освоить принцип асинхронности запросов и способа работы с ним; освоить работу с J2EE и проектами Enterprise.

**Задача:** По имеющемуся примеру создать простейшее веб-приложение J2EE Enterprise с поддержкой MDB и JMS. Приложение должно работать с базой данных (осуществлять ввод и вывод данных), и реализовывать возможность асинхронности запросов.

**Ход работы:**

1. Открыл сайт с примером, создал веб-проект J2EE.
2. По примеру спроектировал EJB часть. Создал класс сущности, установил соединение с JavaDB, компонент-управляемый-сообщениями и фасад сеанса.
3. По примеру спроектировал web часть. Создал единичный сеансовый компонент, имплементирущий SessionListener, сервлеты ListNews и PostMessage с необходимыми аннотациями и импортами.
4. При попытке запустить проект возникла ошибка glassfish: jms/NewMessage команда не определена. На сайте был оставлен намек на то, что проблему можно решить в консоли администратора сервера glassfish.
5. Покопавшись в консоли, я нашел ресурсы JMS и создал новую connection factory – jms/newMessageFactory, и новый пункт назначения сообщений – jms/newMessage, после чего проект заработал.

**Скриншоты:**

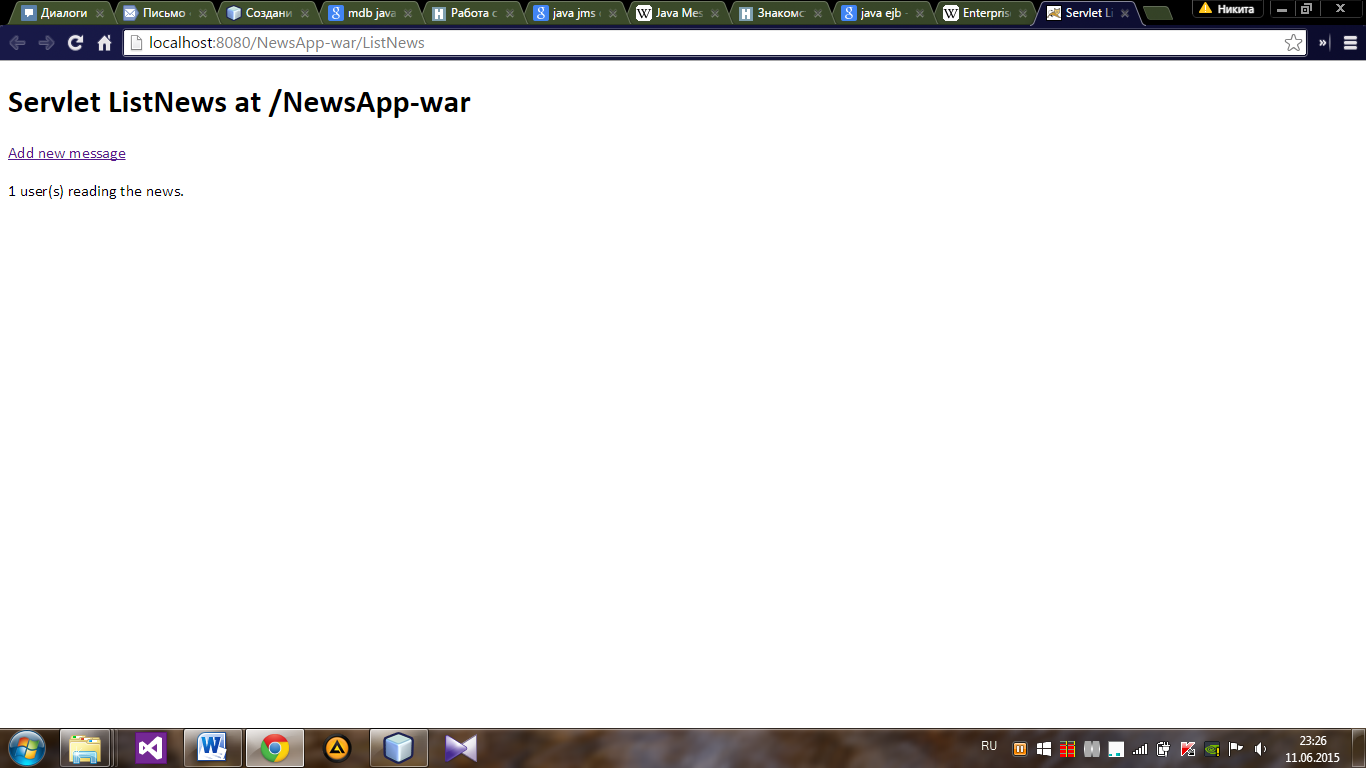


Рисунок 1 – Начальная (главная) страница приложения

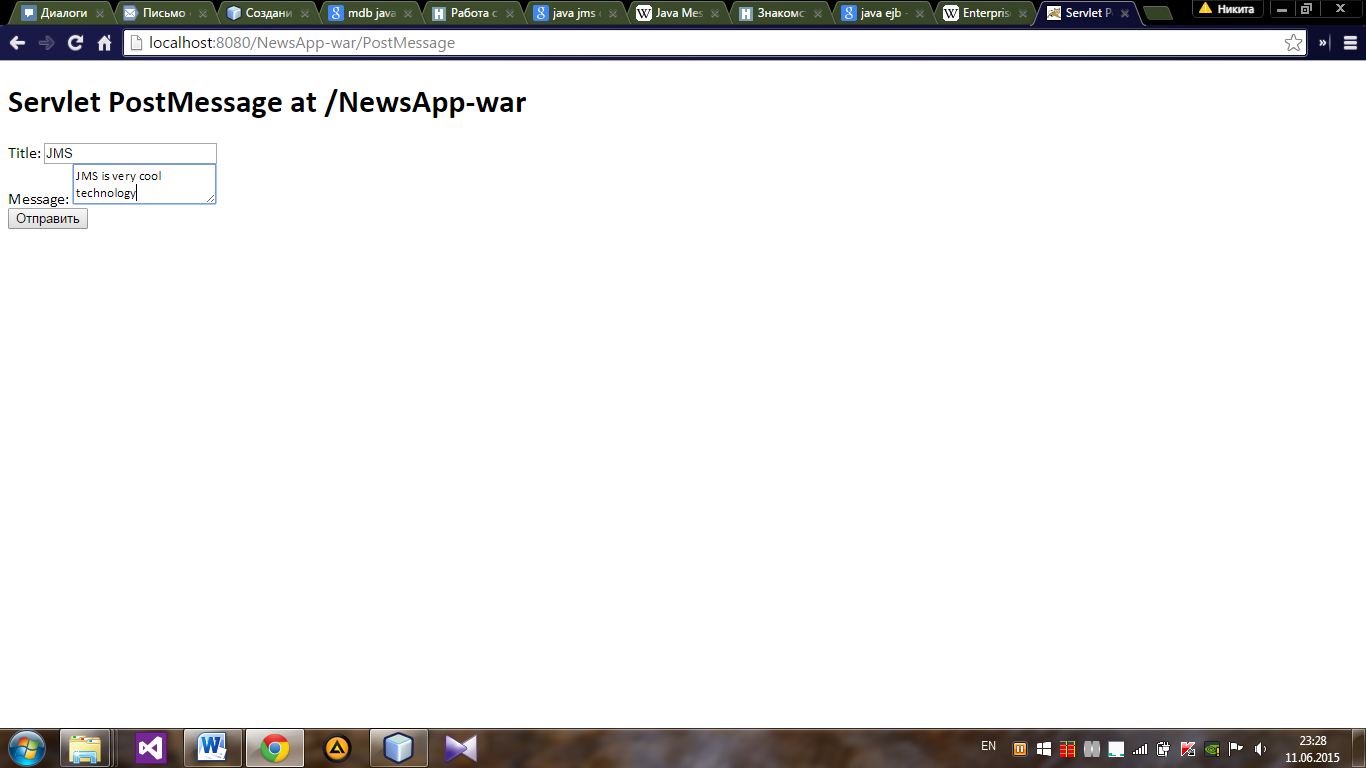


Рисунок 2 – Страница добавления сообщений с заполненной формой

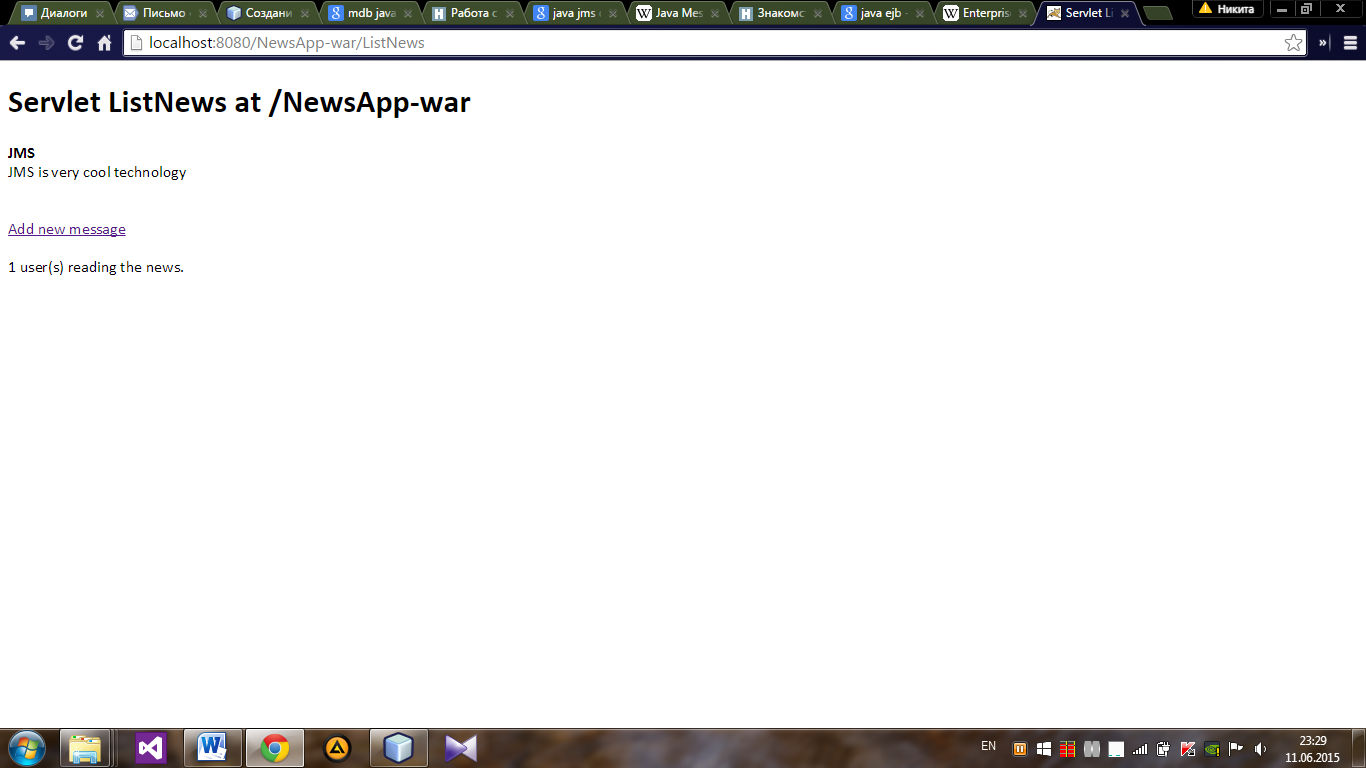


Рисунок 3 – Результат нажатия кнопки «Отправить». Сообщение сохранено в базе и выведено на jsp (сервлет)

**Принцип асинхронной обработки запроса:** Работу с сообщениями поддерживает технология Java JMS. Java Message Service (JMS) — стандарт [промежуточного ПО](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для рассылки [сообщений](https://ru.wikipedia.org/wiki/%D0%A1%D0%BE%D0%BE%D0%B1%D1%89%D0%B5%D0%BD%D0%B8%D0%B5), позволяющий приложениям, выполненным на платформе [Java EE](https://ru.wikipedia.org/wiki/Java_EE" \o "Java EE), создавать, посылать, получать и читать сообщения.

Компоненты системы могут посылать сообщения (producer) и получать их (consumer). Сообщение отправляет producer на пункт назначения (destination), являющимся на сервере queue или topic, после чего consumer может забрать оттуда сообщение. В зависимости от того, какой тип имеет destination, разделяют две модели работы с сообщениями.

Первая модель — Point-to-Point.

В случае если на сервере destination имеет тип queue, то сообщение, которое отправил producer, получает единственный consumer. Если на эту очередь сообщений подписано несколько получателей, то сообщение получит только один из них.

Вторая модель — Publish-subscribe.

В случае если на сервере destination имеет тип topic, то одно сообщение может быть прочитано неограниченным количеством consumer, подписанных на этот на этот destination.

Таким образом, наше приложение реализует модель Point-to-Point, что логично, т.к. у нас только один сервлет PostMessage принимает сообщения.

##### Структура JMS сообщения

Сообщение состоит из заголовка, поля свойств и тела.  
Заголовок хранит мета информацию сообщения, заполняемую автоматически.  
Поле свойств схоже с заголовком, но оно заполняется программно, и позже получатель сможет прочитать эту информацию.  
Тело содержит полезную нагрузку сообщения. Тип нагрузки определяется при создании сообщения. Конкретные типы унаследованы от интерфейса [javax.jms.Message](http://docs.oracle.com/javaee/6/api/javax/jms/Message.html).

Для обработки входящих сообщений используется MDB. Асинхронная обработка реализуется через пул объектов, из которых на обработку сообщения сервер выделят объекты при необходимости.  
  
Для реализации MBD достаточно унаследовать бин от интерфейса javax.jms.MessageListener, реализуя метод onMessage(), и аннотировать соответствующим образом класс.

В onMessage метод добавляется необходимая бизнес логика, в зависимости от типа сообщения, его содержания и тд.

Синхронный запрос - это запрос с ожиданием ответа. Т.е. скрипт создал объект, а объект послал запрос на сервер, это сразу, мгновенно не происходит. И пока этот запрос обрабатывается на сервере скрипт "висит", скрипт просто "висит", он ждет ответ с сервера. Дальше скрипт не выполняется, браузер ничего не делает, он ждет ответ с сервера. Если сервер работает долго или связь с сервером медленная, то это "зависание" может быть даже заметно на глаз. И только тогда, когда сервер все наконец-то пришлет, только тогда скрипт пойдет выполняться дальше. Т.е. скрипт в момент обмена данными с сервером он "висит".

Асинхронный запрос - это запрос без ожидания ответа от сервера. Т.е. скрипт создал объект, а объект послал запрос на сервер. При этом скрипт продолжает выполняться дальше и не ждет ответа от сервера. В какой-то момент данные вернуться.