УДК 004

Обработка часовых поясов в веб-приложении Java

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут представлены примеры обработки часовых поясов в веб-приложении. Работа будет происходить в среде программирования IntelijIdea с использованием языка программирования Java.
Ключевые слова: Java, Время, Веб

Handling Time Zones in a Java Web Application

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will provide examples of handling time zones in a web application. The work will take place in the IntelijIdea programming environment using the Java programming language.
Keywords: Java, Time, Web

Время – одно из самых сложных понятий. Разделение на часовые пояса, переход на летнее и зимнее время, различные дни выходных в разных странах – это все время. Если писать программу, в которой должно отображаться местное время для пользователя того часового пояса, где он живет, то будет необходимость объединить несколько уровней: браузер, веб-сервер, база данных.

Цель работы – реализовать пример обработки часовых поясов в веб приложении Java.

Исследованиями в данной теме занимались следующие авторы.

А.А.Федоренко провела объяснение концепции часовых поясов и сравнила различные способы для работы с датой и временем на языке

программирования Java [1]. А.А.Симаков реализовал удобную трассировку JVM программ для обратного проектирования и анализа ПО [2]. В.П.Великов, К.С.Добрева обсудили в своей работе некоторые проблемы автоматизированной генерации ПО [3]. С.В.Мельников в своей статье описал принцип работы с отладочным интерфейсом Java [4]. М.К.Ермаков и С.П.Вартанов рассмотрели при помощи динамического анализа подход к поиску состояний гонки в многопоточных программах на языке Java. [5].

Базы данных могут использовать локальный часовой пояс базовой операционной системы или пользовательский часовой пояс.

Для согласованности лучше всего, если все временные метки базы данных хранятся в формате UTC, потому что таким образом будет намного проще вычислять интервалы временных меток, поскольку все пользовательские временные метки относятся к одному и тому же часовому поясу. Более того, если пользователь перемещается в другой часовой пояс, нет необходимости изменять уже сохраненную информацию о дате и времени «user=specific», поскольку преобразование в любом случае будет выполнено на веб-слое.

Для различных БД используются свои команды для установки формата времени (рис.1-2).

```
default_time_zone='+00:00'
```

Рисунок 1 – MySQL

```
timezone = 'UTC'
```

Рисунок 2 - PostgreSQL

По умолчанию приложение Java использует системный часовой пояс. Но при использовании AWS, часовой пояс по умолчанию установлен в формат «UTC». Это можно увидеть при запросе журналов приложений, там временная метка сообщений относится к UTC.

Если JVM не использует часовой пояс UTC, а база данных использует, то имеется несколько вариантов

Можно установить часовой пояс операционной системы в UTC или, если его изменить нельзя, можно установить часовой пояс JVM по умолчанию, для этого необходимо будет установить свойство «user.timezone» (рис.3).

```
java -Duser.timezone="UTC" com.revogain.RevoGainApplication
```

Рисунок 3 – Установка часового пояса

При использовании фреймворка Spring Boot необходимо установить в файле «application.properties» свойство конфигурации (рис.4).

```
spring.jpa.properties.hibernate.jdbc.time_zone=UTC
```

Рисунок 4 – Установка свойств конфигурации

Этот параметр будет указывать «Hibernate» использовать предоставленный часовой пояс при чтении и записи значений столбца метки времени.

В браузере JavaScript хранит дату и время в часовом поясе локальной системы. Поэтому можно использовать часовой пояс пользователя для преобразования всех временных меток UTC, хранящихся в базе данных или созданных в JVM. Для этого нужно сохранить часовой пояс пользователя во время аутентификации.

На странице «login» форма аутентификации содержит «timeZoneOffset» скрытое поле (рис.5).

```
<form th:action="@{/login}" method="post" class="form-signin" >


    <input type="hidden" id="timeZoneOffset" name="timeZoneOffset" value=""/>

    <button class="btn btn-lg" type="submit">Login</button>
</form>
```

Рисунок 5 – Страница login

Здесь можно прочитать значение «timeZoneOffset», используя веб-фильтр (рис.6).

```
public class TimeZoneOffsetFilter implements Filter {

    @Override
    public void doFilter(
            ServletRequest request,
            ServletResponse response,
            FilterChain chain)
                throws IOException, ServletException {
        TimeZoneOffsetContext.set(request.getParameter("timeZoneOffset"));

        chain.doFilter(request, response);

        TimeZoneOffsetContext.reset();
    }
}
```

Рисунок 6 – Чтение часового пояса пользователя

«TimeZoneOffsetContext» просто утилита-заполнитель, которая хранит информацию о часовом поясе, чтобы можно было прочитать ее после того, как «Spring Security» аутентифицирует запрос пользователя на вход (рис.7).

```
public class TimeZoneOffsetContext {

    private static final ThreadLocal<String> timeZoneOffsetHolder =
        new ThreadLocal<>();

    public static String get() {
        return timeZoneOffsetHolder.get();
    }

    public static void set(String timeZoneOffset) {
        timeZoneOffsetHolder.set(timeZoneOffset);
    }

    public static void reset() {
        timeZoneOffsetHolder.remove();
    }
}
```

Рисунок 7 – Класс для обработки часового пояса

Так же можно установить часовой пояс пользователя в «UserDetails» объекте «Spring Security», который связан с текущим зарегистрированным пользователем (рис.8).

```
@Service
@Transactional(readOnly = true)
public class UserService implements UserDetailsService {

    ...

    @Override
    public UserDetails loadUserByUsername(String username)
            throws UsernameNotFoundException {
        User user = userRepository.findByEmail(username);

        if (user == null) {
            throw new UsernameNotFoundException("""
                This email or password are invalid.
                Please review them and try again.
                """
            );
        }

        return new ApplicationUserDetails(user)
            .setTimeZoneOffset(
                TimeZoneOffsetContext.get()
            );
    }

    ...
}
```

Рисунок 8 – Передача часового пояса пользователя

«ApplicationUserDetails» хранит информацию о часовом поясе и предоставляет возможность форматирования (рис. 9).

```
public class ApplicationUserDetails
        implements UserDetails {

    public static final DateTimeFormatter DATE_TIME_FORMATTER =
        DateTimeFormatter.ofPattern(
            "dd/MM/uuuu HH:mm:ss"
        );

    private User user;

    private ZoneOffset zoneOffset;

    public ApplicationUserDetails(User user) {
        this.user = user;
    }

    ...

    public ZoneOffset getZoneOffset() {
        return zoneOffset;
    }

    public ApplicationUserDetails setTimeZoneOffset(String timeZoneOffset) {
        if (timeZoneOffset != null) {
            int offsetMinutes = Integer.valueOf(timeZoneOffset) * -1;
            this.zoneOffset = ZoneOffset.ofTotalSeconds(offsetMinutes * 60);
        }
        return this;
    }

    public String getFormattedDateTime(LocalDateTime dateTime) {
        if(zoneOffset != null) {
            OffsetDateTime serverOffsetDateTime = dateTime.atZone(
                ZoneId.systemDefault()
            ).toOffsetDateTime();

            OffsetDateTime clientOffsetDateTime = serverOffsetDateTime
                .withOffsetSameInstant(zoneOffset);

            return DATE_TIME_FORMATTER.format(clientOffsetDateTime);
        }
        return dateTime.format(DATE_TIME_FORMATTER);
    }
}
```

Рисунок 9 – Класс для обработки часового пояса для каждого пользователя

Теперь можно конвертировать метки времени в часовой пояс пользователя. Например, при отображении журнала активности можно сместить временную метку операции в часовой пояс пользователя (рис.10).

```
<tr th:each="op, status : ${operationsPage}"
    th:style="${status.odd}? 'font-weight: normal;'">
    <td th:text="${op.id}"></td>
    <td th:text="${userDetails.getFormattedDateTime(op.createdOn)}"></td>
    <td th:text="${op.credits}"></td>
    <td th:text="${op.type.label}"></td>
</tr>
```

Рисунок 10 – Отображение времени у пользователя

При работе с часовыми поясами лучше всего использовать UTC как можно чаще и преобразовывать метку времени только в часовой пояс текущего пользователя при отрисовке пользовательского интерфейса.

Так же в веб-приложении Java можно использовать Spring Security и метод «UserDetails» для хранения часового пояса пользователя и переноса временных меток UTC, извлеченных из базы данных или созданных на веб-сервере, в локальные часовые пояса каждого зарегистрированного пользователя.

Библиографический список

1. Федоренко А.А. Средства работы с датой и временем в языке программирования java // Актуальные научные исследования в современном мире. 2021. №7-1(51). С. 65-69.
2. Симаков А.А. Java tracer. Программное средство для трассировки java программ // Заметки по информатике и математике. 2019. №3. С. 51-57.
3. Великов В.П., Добрева К.С. Генератор из диаграмм классов java в исходный код java // Информационные системы и технологии: управление и безопасность. 2014. №3. С. 14-23.
4. Мельников С.В. Обзор и применение отладочного интерфейса java (jdi) для обратимой модификации программных продуктов // Современные проблемы науки и образования. 2018. №8. С. 8-19.
5. Ермаков М.К. и Вартанов С.П. Подход к проведению динамического анализа java-программ методом модификации виртуальной машины java // Труды института системного программирования ран. 2015. №2. С. 39-52.