СОЗДАНИЕ СИСТЕМЫ МОНИТОРИНГА РЕСУРСОДОСТУПНОСТИ СЕРВЕРА

Баркалов Л.Д.

Во время работы приложений на сервере могут возникнуть случаи утечки ресурсов (памяти, загруженность ЦП, и т.д.), которые, в некоторых случаях, могут повлечь за собой неправильную работу сервера приложений, либо выход его из строя. Чтобы избежать таких ситуаций требуется вести мониторинг сервера. Но одного мониторинга недостаточно. В случае возникновения нештатной ситуации в работе какого – либо приложения, нужно незамедлительно оповестить разработчика о существующей неисправности.

В настоящее время не существует подобных встраиваемых решений в открытом доступе.

Целью данной работы является разработка приложения, предназначенного для мониторинга сервера по выбранным параметрам, и оповещения разработчиков в случае нарушения мониторируемым приложением установленных границ этих параметров (система триггеров).

В процессе работы необходимо было решить следующие задачи:

- разработать систему мониторинга состояния ресурсодоступности JVM - сервера;
- разработать редактор системы оповещения разработчика, о существующих проблемах на сервере;
- разработать систему оповещения разработчиков;
- разработать интерфейс взаимодействия разработчиков с системой триггеров;

Основными языками для разработки были выбраны Java и Kotlin. Kotlin – современный язык программирования более лаконичный и типобезопасный, чем Java, работающий поверх установленной JVM.

Для получения данных о ресурсодоступности сервера было решено использовать технологию JMX. JMX позволяет создавать многоуровневую архитектуру, где управляемые ресурсы и управляющее приложение могут быть встроены в одно решение. Ресурсы представляют из себя Java-объекты – ManagedBeans (MBeans), которые регистрируются на основном сервере –MBean-server. Этот сервер действует как агент и может быть запущен практически везде, где есть JVM. После обработки данных, при выполнении условия, заданного разработчиком будет отправлено оповещение на почту о существующей проблеме.

С помощью метода queryNames были получены данные о всех MBeans объектах для mbeans – сервера.

С помощью метода getAll осуществлялся доступ к базе данных, который получен из унаследованного от класса BaseDao класса TriggersDao.

Создание задач из базы было осуществлено с помощью шедулера[3], из класса ThreadPoolTaskSchedulerX унаследованного от ThreadPoolTaskScheduler из спринг фреймворка, описанного в собственной библиотеке ООО «СибирьСофтПроект». Проверка условий проводится с помощью метода Condition, в котором формируется сообщение, которое в дальнейшем будет отправлено разработчику, для его оповещения о существующих проблемах.

Отправка сообщения осуществляется с помощью метода MessageSupport, в котором используются различные классы из пакета Util и методы из класса javax.mail.internet.

В рамках создания системы мониторинга был создан модуль для платформы Flowdox, в котором разработчик может выбрать интересующий его параметр мониторинга, задать его название, выбрать условие для выполнения Action и период, в течение которого будет проверяться условие.

Приложение имеет web – интерфейс, и может быть встроено в уже существующие проекты, как модуль. Так же приложение можно использовать как отдельный проект, создав подключение к исполняющей среде.

Графический интерфейс был создан с помощью технологии JSP, а различные объекты и операции над ними были выполнены с помощью Javascript. Пользователь может выбрать

интересующий его Aggregator, условие Comparator и действие Action. Графический интерфейс встроенного модуля представлен на рисунке 1.

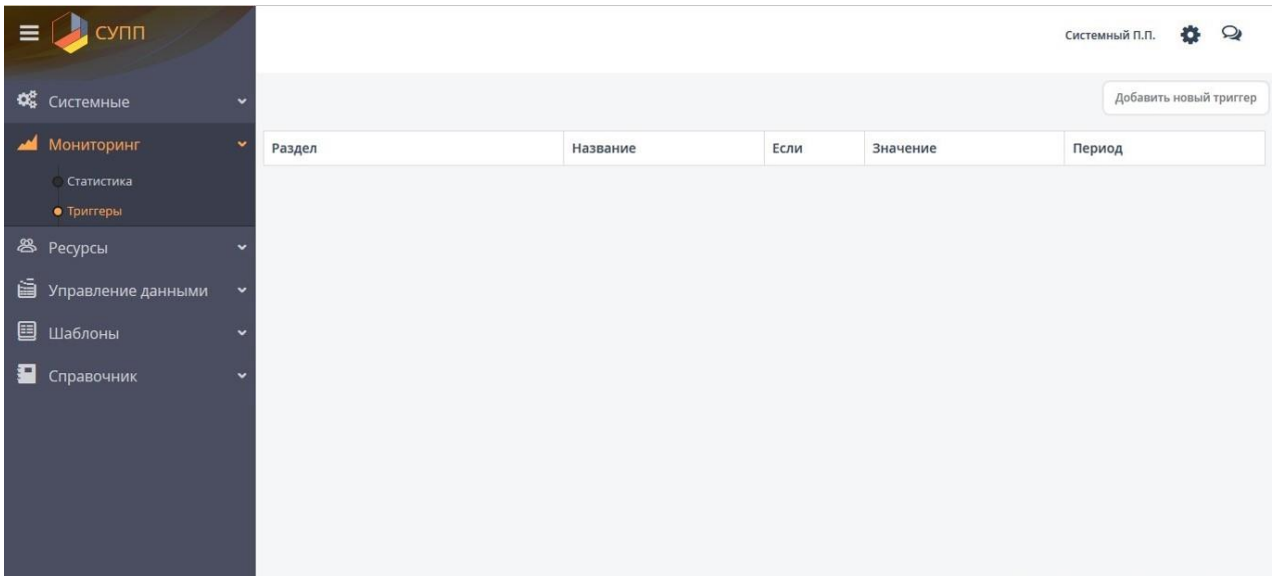

Рисунок 1 – Графический интерфейс системы мониторинга

В результате работы было реализовано приложение для мониторинга сервера и своевременного оповещения разработчика о проблемах.

Данный проект предполагает дальнейшее развитие, в рамках которого будет реализована система мониторинга «в реальном времени», с использованием современных технологий разработки web-сервисов, таких как Comet, Rest и др., а также взаимодействие с другими платформами (CLR и др.).

Структурная модель модернизированного приложения, включающая различные платформы, мониторинг в режиме «реального времени», различные способы оповещения представлена в приложении 1. Описание модели представлено в таблице 1.

Таблица 1 – Логическая структура системы мониторинга

№	Блок управления	Описание
1	Error filter	Фильтр ошибок, например Not Connection
2	API	Программный интерфейс взаимодействия, например построение списка из множества объектов
3	Triggers converter	Преобразование management - объектов
4	Triggers service	Сервис обработки пользовательских триггеров
5	Action service	Сервис проверки результата действия
6	Messages service	Сервис отправки оповещений
7	Subscribes channel	Сервис распределения контента обновляемой информации
8	Data access	Слой доступа к данным
9	Connect service	Сервис создания подключений и каналов распределения
10	Data base	База данных
11	DPS	Data Provide Servise – сервис распределения действий
12	General form	Представление данных, полученных от различных платформ
13	… srv.	Сервис подключения к платформе
14	JVM	Одна из платформ

Приложение 1