

Практическое применение Redis в Spring Boot для ускорения обработки запросов и хранения временных данных

Саухин Сергей Иванович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В статье рассматривается практическое применение Redis в приложениях на Spring Boot для ускорения обработки запросов и организации хранения временных данных. Описаны архитектурные особенности Redis как in-memory хранилища, показаны основные сценарии его использования в прикладной системе и раскрыта роль TTL при управлении жизненным циклом записей. Практическая часть включает схему подключения Redis к Spring Boot-приложению, организацию сервисного слоя и реализацию REST API для записи и чтения данных. Результаты работы показывают, что использование Redis упрощает обработку повторяющихся запросов, уменьшает нагрузку на основное хранилище и делает приложение более управляемым

Ключевые слова: Redis, Spring Boot, Spring Data Redis, API, TTL, кэширование данных, временные данные, RedisTemplate, REST API, серверное приложение.

Practical use of Redis in Spring Boot for faster request processing and temporary data storage

Saukhin Sergey Ivanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The article considers the practical use of Redis in Spring Boot applications for faster request processing and temporary data storage. It describes Redis as an in-memory data store, outlines the main application scenarios, and explains the role of TTL in record lifecycle management. The practical part includes Redis connection setup, service-layer organization, and REST API implementation for reading and writing values. The results show that Redis simplifies processing of repeated requests, reduces the load on the primary data store, and improves application manageability.

Keywords: Redis, Spring Boot, Spring Data Redis, API, TTL, data caching, temporary data, RedisTemplate, REST API, server-side application.

1 Введение

1.1 Актуальность

В современных серверных приложениях производительность определяется не только качеством бизнес-логики, но и способом работы с часто запрашиваемыми данными. Значительная часть обращений в прикладных системах связана с повторным чтением ранее вычисленных значений, временных токенов, пользовательских сессий и промежуточных результатов обработки. Передача всех таких запросов только в реляционную базу данных увеличивает задержки и создает дополнительную нагрузку на основное хранилище [1, 4, 6].

Redis применяется как инструмент снижения этой нагрузки за счет хранения данных в оперативной памяти и поддержки операций доступа по ключу. Документация Redis и публикации по кэшированию показывают, что данная система используется для реализации кэша, хранения временных сущностей и ускорения чтения в высоконагруженных приложениях [1, 2, 4].

Практическая значимость темы определяется тем, что связка Redis и Spring Boot позволяет быстро встроить слой кэширования и хранения временных данных в типовую архитектуру серверного приложения. Такой подход упрощает обработку повторяющихся запросов, сокращает время отклика и делает взаимодействие с данными более предсказуемым [8, 9].

1.2 Обзор исследований

В работах М. В. Ступиной, И. П. Костенко и публикациях по in-memory СУБД Redis рассматривается как инструмент повышения производительности web-приложений. Эффективность Redis связывается с хранением данных в оперативной памяти, быстрым доступом по ключу и возможностью разгрузить основное хранилище [1, 2, 3].

Д. Н. Сомов и И. В. Князев анализируют кэширование в распределенных web-приложениях и оптимизацию повторных запросов. В этих работах подчеркивается, что кэширование особенно эффективно в сценариях, где одни и те же данные многократно запрашиваются различными компонентами приложения [4, 5].

Исследования высоконагруженных систем показывают, что кэширование и снижение числа обращений к основному источнику данных являются важными условиями масштабируемости. Поэтому Redis целесообразно рассматривать как инфраструктурный компонент, который повышает устойчивость серверного приложения при росте нагрузки [6, 7].

В работах, посвященных современным архитектурам REST API и технологиям web-приложений, Redis упоминается как практический инструмент ускорения запросов и организации промежуточного хранения данных. Это позволяет связать применение Redis со Spring Boot не только с кэшированием, но и с общей архитектурой обработки прикладных запросов [8, 9].

Таким образом, анализ публикаций показывает, что Redis следует рассматривать не только как быстрое in-memory хранилище, но и как

практический инструмент повышения производительности приложений на Spring Boot. Эффективность такого подхода определяется корректным выбором сценариев использования, организацией TTL и местом Redis в общей архитектуре приложения [1, 4, 10].

1.3 Цель исследования

Цель исследования — изучение возможностей использования Redis в приложениях на платформе Spring Boot, анализ особенностей его интеграции в архитектуру информационной системы и оценка влияния на производительность обработки данных и запросов.

Для достижения поставленной цели в работе решаются следующие задачи:

- рассмотреть архитектурные особенности Redis как in-memory хранилища;
- описать практические преимущества интеграции Redis со Spring Boot;
- показать схему подключения Redis в приложении;
- представить пример записи и чтения данных через REST API;
- раскрыть роль TTL в управлении жизненным циклом временных записей;
- определить прикладной эффект использования Redis в серверной разработке.

2 Материалы и методы

Теоретическую основу работы составляют публикации по кэшированию, проектированию серверных приложений и применению Redis в прикладных системах, а также официальные материалы Redis и Spring Data Redis. Такой набор источников позволяет рассмотреть Redis одновременно как программный инструмент и как архитектурный элемент системы [1, 5, 8].

Практическая часть связана с демонстрационным приложением на Java с использованием Spring Boot и Gradle. В проект включены зависимости `spring-boot-starter-web` и `spring-boot-starter-data-redis`, после чего выполнена настройка подключения к Redis. Для работы с ключами и значениями используется `StringRedisTemplate`, обеспечивающий доступ к операциям записи и чтения [8, 9].

В рассматриваемой модели REST-контроллер принимает клиентский запрос, передает данные в сервисный слой, а затем сервис сохраняет значение в Redis с заданным временем жизни. Корректность работы решения определяется сценариями сохранения, получения и автоматического удаления записи после истечения TTL [2, 8].

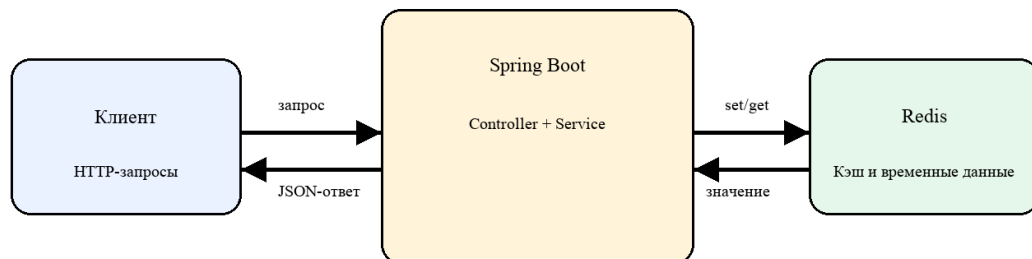
3 Результаты и обсуждение

3.1 Общая архитектура решения

На первом этапе определяется базовая архитектура приложения. Клиент взаимодействует с REST API Spring Boot, а сервисный слой обращается к Redis для чтения и записи значений. Такая схема обеспечивает разделение

ответственности между интерфейсом приложения, прикладной логикой и слоем хранения временных данных [8, 9].

Общая схема использования Redis в Spring Boot



Spring Data Redis позволяет быстро встроить Redis в слой сервисов приложения.

Рис. 1. Общая схема применения Redis в приложении на Spring Boot

3.2 Подключение Redis к проекту

Следующим этапом является подключение Redis к проекту и настройка параметров соединения в конфигурационном файле `application.yml`. Такой подход позволяет встроить Redis в приложение без ручной реализации сетевого взаимодействия и использовать его как отдельный инфраструктурный компонент [8, 9].

Подключение Redis в проект Spring Boot

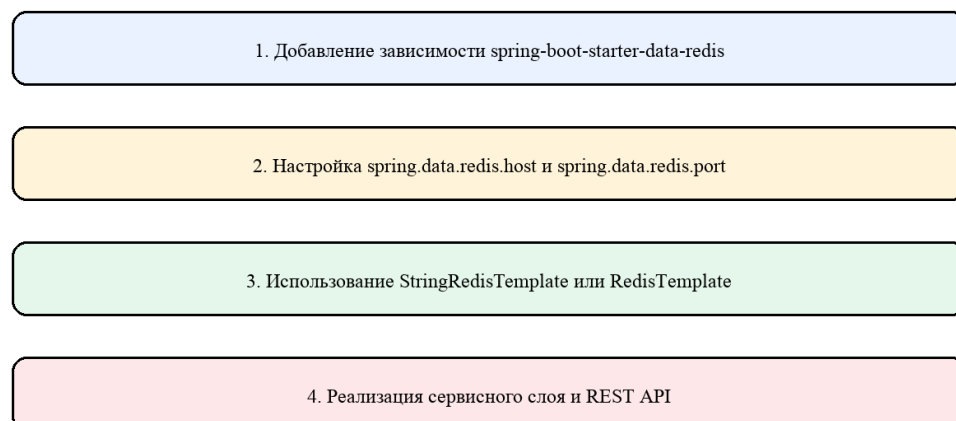


Рис. 2. Схема подключения Redis к приложению на Spring Boot

3.3 Реализация сервисного слоя

Для выполнения операций чтения и записи формируется отдельный сервисный компонент на основе `StringRedisTemplate`. В методе сохранения значение записывается по ключу с TTL, а метод чтения возвращает текущее

значение по ключу либо `null` при отсутствии записи. Выделение сервисного слоя упрощает сопровождение кода и отделяет работу с Redis от логики контроллера [8].

3.4 Организация REST API

Для демонстрации взаимодействия с Redis реализуется REST-контроллер с двумя основными операциями: `POST /redis` для сохранения пары ключ–значение и `GET /redis/{key}` для чтения сохраненного значения. Такая схема позволяет проследить путь данных от клиентского запроса до временного хранилища и обратно [8, 9].

Сценарий обращения к REST API и Redis

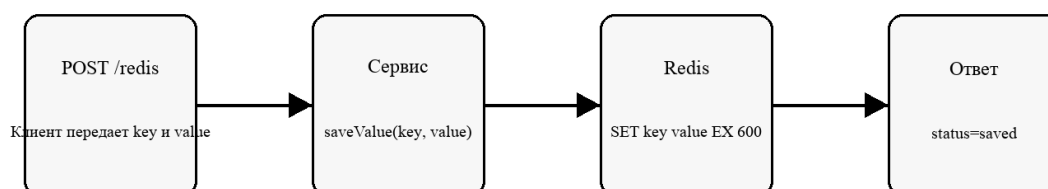
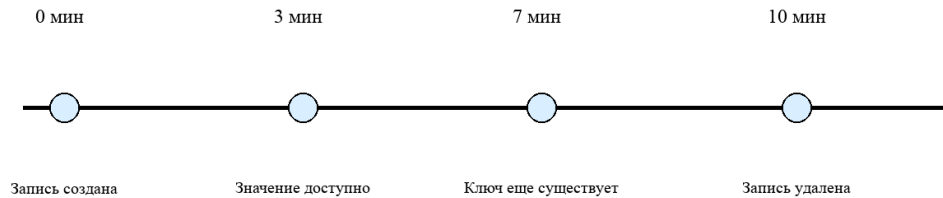


Рис. 3. Схема обращения к REST API и Redis

3.5 Использование TTL

Одним из ключевых механизмов Redis является TTL, позволяющий задавать время жизни записи. Этот механизм делает Redis удобным для хранения временных токенов, кодов подтверждения, сессионных данных и короткоживущих кэшей. Автоматическое удаление устаревших записей упрощает сопровождение системы и уменьшает объем служебной логики в приложении [1, 3, 10].

Жизненный цикл записи с TTL

TTL позволяет Redis автоматически очищать временные данные без отдельного планировщика.

Рис. 4. Жизненный цикл записи при использовании TTL

3.6 Сценарии cache hit и cache miss

При использовании Redis как кэша возникают два типовых сценария. В случае cache hit нужное значение уже находится в Redis, поэтому приложение возвращает ответ без обращения к основному источнику данных. В случае cache miss данные запрашиваются из основного хранилища, после чего могут быть сохранены в Redis для повторного использования. Сопоставление этих сценариев показывает, каким образом Redis уменьшает нагрузку на более медленные подсистемы хранения [4, 8, 9].

Сценарий cache hit и cache miss

Рис. 5. Сценарии cache hit и cache miss при работе с Redis

3.7 Практическая значимость интеграции

Рассматриваемая реализация показывает, что Redis особенно полезен в приложениях с повторяющимися операциями чтения и временными данными, не требующими постоянного хранения в реляционной базе. Его можно использовать для кэширования результатов запросов, хранения сессий, реализации rate limiting и работы с одноразовыми токенами. В сочетании со Spring Boot это формирует предсказуемую структуру приложения и упрощает внедрение механизмов оптимизации [1, 4, 6].

Дальнейшее развитие проекта связано с использованием аннотаций `@Cacheable`, `@CachePut` и `@CacheEvict`, а также настройкой `RedisCacheManager` для прозрачного кэширования результатов методов. Такой подход сокращает объем шаблонного кода и делает работу с кэшем частью общей инфраструктуры приложения [9].

4 Выводы

Проведенный анализ показывает, что Redis является эффективным инструментом для кэширования и хранения временных данных в приложениях на Spring Boot. Использование Redis обеспечивает быстрый доступ к значениям, позволяет управлять жизненным циклом записей при помощи TTL и уменьшает нагрузку на основное хранилище [1, 2, 4].

Практическая реализация демонстрирует, что интеграция Redis в Spring Boot может быть выполнена с минимальными трудозатратами за счет применения Spring Data Redis. Реализованный пример REST API показывает, каким образом Redis включается в типовую архитектуру серверного приложения и упрощает обработку прикладных запросов [8, 9].

Проведенное исследование выполнено в рамках изучения дисциплины «Архитектура информационных систем» и позволило на практике рассмотреть особенности интеграции Redis в многоуровневую архитектуру современных веб-приложений, а также оценить влияние механизмов кэширования на производительность и масштабируемость информационных систем.

Таким образом, применение Redis в приложениях на Spring Boot упрощает организацию работы с временными данными, повышает производительность системы и делает архитектуру обработки повторяющихся запросов более предсказуемой. Эффективность такого подхода определяется корректным выбором сценариев использования Redis и согласованной интеграцией этого инструмента в структуру приложения [1, 3, 10].

Библиографический список

1. Ступина М. В., Костенко И. П. Повышение производительности web-приложений средствами СУБД Redis // Молодой исследователь Дона. 2022. № 4(37). URL: <https://elibrary.ru/item.asp?id=82994127> (дата обращения: 10.06.2026).
2. Костенко И. П., Ступина М. В. О преимуществах СУБД Redis // Актуальные

- проблемы науки и техники. 2022. С. 344–345. URL: <https://elibrary.ru/item.asp?id=49165452> (дата обращения: 10.06.2026).
3. Левин А. А., Семичев С. А., Казакова И. А. Обзор рынка In-Memory систем управления базами данных // Вестник Пензенского государственного университета. 2024. № 4(48). С. 87–89. URL: <https://elibrary.ru/item.asp?id=80462811> (дата обращения: 10.06.2026).
 4. Актуальные проблемы науки и техники. 2022: материалы Всероссийской (национальной) научно-практической конференции, Ростов-на-Дону, 16–18 марта 2022 года / отв. ред. Н. А. Шевченко. Ростов-на-Дону: Донской государственный технический университет, 2022. 1015 с. URL: <https://elibrary.ru/item.asp?id=49373652> (дата обращения: 10.06.2026).
 5. Князев И. В. Продвинутое кеширование и оптимизация веб-приложений с помощью технологии SWR // Sciences of Europe. 2021. № 73-1(73). С. 47–49. URL: <https://elibrary.ru/item.asp?id=46264133> (дата обращения: 10.06.2026).
 6. Мирзапулатов Р. У., Сейдаметова З. С. Особенности разработки высоконагруженных систем // Информационно-компьютерные технологии в экономике, образовании и социальной сфере. 2018. № 2(20). С. 6–13. URL: <https://elibrary.ru/item.asp?id=36399900> (дата обращения: 10.06.2026).
 7. Топалов Н. К. Анализ производительности современных систем управления базами данных // Молодой ученый. 2024. № 49(548). С. 19–22. URL: <https://elibrary.ru/item.asp?id=75217775> (дата обращения: 10.06.2026).
 8. Салтанов Д. С., Арсентьева Н. В. Сравнительный анализ архитектурных подходов к разработке REST API для высоконагруженных систем // Форум молодых ученых. 2025. № 6(106). С. 329–340. URL: <https://elibrary.ru/item.asp?id=82633693> (дата обращения: 10.06.2026).
 9. Щеглеватых А. А., Аристова Е. М. Анализ существующих технологий для реализации веб-приложений с клиент-серверной архитектурой // Актуальные проблемы прикладной математики, информатики и механики: сборник трудов Международной научной конференции, Воронеж, 13–15 декабря 2021 года. Воронеж: ООО «Вэлборн», 2022. С. 1104–1112. URL: <https://elibrary.ru/item.asp?id=49273772> (дата обращения: 10.06.2026).
 10. Боярко М. В., Волик А. Г. Проблема построения архитектуры веб/мпприложения для бронирования столиков в ресторанном бизнесе // Современные исследования в области техники и технологий: сборник статей Международной научно-практической конференции, Симферополь, 22 апреля 2025 года. Симферополь: ООО «Издательство Типография "Ариал"», 2025. С. 184–197. URL: <https://elibrary.ru/item.asp?id=88838481> (дата обращения: 10.06.2026).