# Caching

## **Spring Boot Caching:**

1. What is Cache?
2. Cache is processing that temporarily store the data in cache, so that can access quickly the data, Caching improve the performance and reduce the time taken access the data

* **How does caching work?**

1. Caches are used CPU, Web browser, apps and operating System.
2. Caches are used to store frequently accessed data, such as app-related data, or instructions
3. Caches can be used store web pages and content in your browser so they load faster the next time to vist.
4. Why Use Caching?
5. Reduce lantency : Faster access to data compared to fetching from a database or an external services.
6. Decrease Load : Reduce the number of calls to the backend systems or database.
7. Improve scalability: Helps applications handle higher traffic loads efficiently
8. Cache in Spring Boot
9. Types of caching?
10. In Memory
11. Distributed

* **Annotations used in Caching:**

@EnableCaching

@Cacheable

@CachePut

@CacheEvict

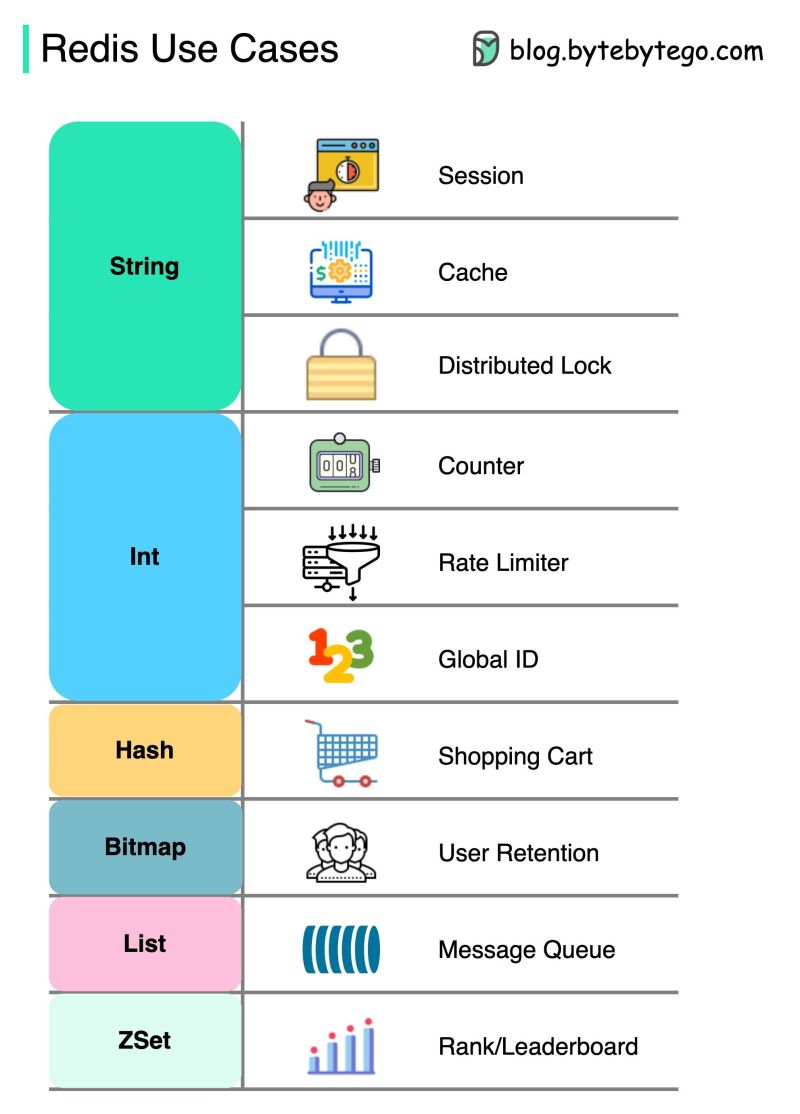
@Caching

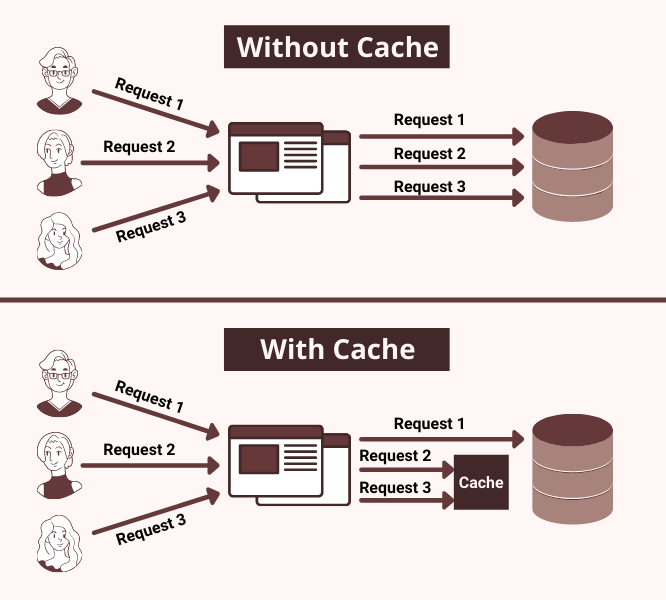
**Redis Port:**

6379

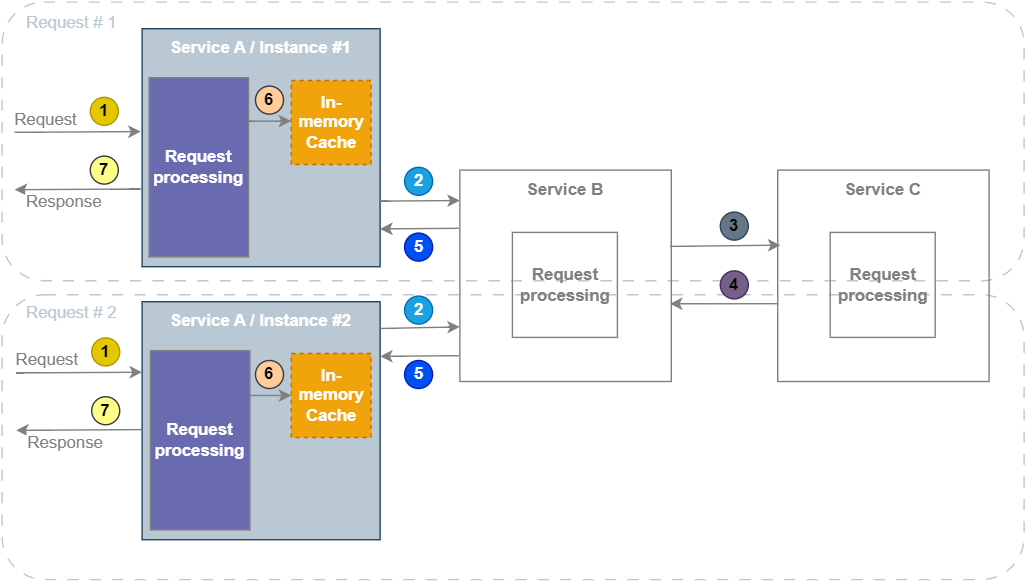
127.0.0.1:6379>

**Popular Redis Use Cases**  
  
1. Caching  
The most common use case is to utilize Redis for caching. This helps protect the database layer from overloading. Redis offers fast lookup for cached data and can help improve application performance.  
  
2. Session Store  
We use Redis to share user session data among stateless servers. Redis provides a centralized place to store session data and makes it easy to scale out servers.  
  
3. Distributed lock  
We use Redis distributed locks to grant mutually exclusive access to shared resources. This prevents race conditions in distributed systems. Redis locks are easy to implement and automatically expire.  
  
4. Counter and Rate Limiter  
We use Redis to track like counts, view counts etc on social media apps. Redis counters provide atomic increments/decrements. We also use Redis to enforce rate limits on our API endpoints. This helps prevent abuse.  
  
5. Leaderboard  
Sorted sets make it easy to implement gaming leaderboards in Redis. We can add, update, or remove users from the leaderboard and query ranges efficiently.





In Memory Caching:



**Title: In-Memory Caching vs. Distributed Caching in Spring Boot Microservices**

**Key Points:**

1. **In-Memory Caching**:
   * **Single Application Instance**: Cache stored in the memory of one instance.
   * **High-Speed Access**: Extremely fast data retrieval.
   * **Low Latency**: Ideal for quick lookups.
   * **Single Point of Failure**: Data loss if the instance goes down.
   * **Example**: Caching user session data.
   * **Spring Boot Annotation**: @Cacheable(value = "sessions")
2. **Distributed Caching**:
   * **Multiple Nodes**: Data spread across several nodes in a cluster.
   * **Scalability**: Handles larger datasets and high traffic.
   * **Redundancy**: Data replication ensures availability.
   * **Latency**: Slightly higher due to network overhead.
   * **Example**: Caching product details in an e-commerce application.
   * **Spring Boot Integration**: Use Redis or Hazelcast.

**Distributed Memory:**

