• This is a training **NOT** a presentation
• Please ask questions
Outline

• Ehcache
• Spring Cache Abstraction
• Declarative Caching
• Distributed Caching
Ehcache

- Popular open source cache
- Most widely-used Java-based cache
- Scales from in-process cache to various types of distributed cache
- Actively developed and maintained by Terracotta
Ehcache features

- In-memory
- Can offload to disk
- Built for high concurrency systems
- Can scale to hundreds of nodes
- Supports various cache eviction policies
- Extensible
- JMX Enabled
Configuring Ehcache

• Add Maven dependency

<dependency>
   <groupId>org.lds.stack</groupId>
   <artifactId>stack-ehcache</artifactId>
</dependency>
Configuring Ehcache

• Add to Spring config

<stack-ehcache:cache-manager>
  <stack-ehcache:cache name="test" />
</stack-ehcache:cache-manager>

• See on-line Stack documentation for details.
Stack cache-manager

- What does stack-cache:cache-manager actually do?
  - Configures an Ehcache cache manager
  - Creates a Spring cache manager wrapper
  - Configures individual caches within the cache manager
Lab 1

http://tech.lds.org/wiki/Application_Level_Caching#Lab_1
Spring Cache Abstraction

- Introduced in Spring Framework 3.1
- Includes a simple caching abstraction
- Provides declarative annotation-based caching
  - Annotate methods with @Cacheable to automatically cache their results
@Cacheable example

@Cacheable("books")
public Book findBook(ISBN isbn) {...}
@Inject CacheManager cacheManager;

public Book findBook(ISBN isbn) {
    Cache cache = cacheManager.getCache("books");
    ValueWrapper value = cache.get(isbn);
    if (value != null) {
        return (Book)value.get();
    }
    Book book = ...
    . . .
    Cache.put(isbn, book);
    return book;
}
@Cacheable key generation

• Cacheable methods need a suitable key for caching values
• By default, keys are based on the following rules:
  – If no params, use 0
  – If one param, use param as key
  – If more than one param, compute hash from all parameters
Explicit @Cacheable keys

@Cacheable(value="books", key="#isbn")
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@Cacheable(value="books", key="#isbn.rawNumber")
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)

@Cacheable(value="books", key="T(someType).hash(#isbn)")
public Book findBook(ISBN isbn, boolean checkWarehouse, boolean includeUsed)
@Cacheable(value="book",
condition="#name.length < 32")
public Book findBook(String name)
Other Caching Annotations

- **@CachePut**
  - Adds something to the cache
- **@CacheEvict**
  - Removes something from the cache
- Custom annotations may be used by putting **@Cacheable** on the custom annotation
Enabling caching annotations

• Add the following to your Spring config:

<cache:annotation-driven />

• Uses a Spring CacheManager instance named “cacheManager” to back the annotations
http://tech.lds.org/wiki/Application_Level_Caching#Lab_2
“There are only two hard things in Computer Science: cache invalidation, naming things, and off-by-1 errors.”

– Phil Karlton
Distributed caching

• The Stack uses Oracle Streams AQ for distributed caching transport.
• What the heck? Why AQ?
  – Secure
  – Does not require any changes to prod environments
  – Works well enough (3ms for full cache propagation in 7 node cluster.)
Configuring distributed cache

• Create AQ queue table, queue, and agent in Oracle database.
• See on-line documentation for details.
Configuring distributed cache

• Add the following to Spring configuration:
  – Data source for database
  – Concurrency Executor

• Change stack-cache:cache-manager to stack-ehcache:distributed-aq-cache-manager
Distribution Modes

• Invalidation
  – Nodes are notified of cache entry changes and remove the entry from their local cache.

• Replication
  – Cache entry changes are copied to other nodes.
Lab 3

http://tech.lds.org/wiki/Application_Level_Caching#Lab_3
Mixing local and distributed caches

• To mix local and distributed caches, use the CompositeCacheManager.
• By default, Stack configured cache managers have the Spring bean id "cacheManager"
• To give cache managers a unique name, use the id-suffix attribute.
Cache managers with id-suffix

```xml
<stack-ehcache:cache-manager id-suffix="Local">
  <stack-ehcache:cache name="local1"/>
</stack-ehcache:cache-manager>

<stack-ehcache:distributed-aq-cache-manager id-suffix="Distributed">
  <stack-ehcache:cache name="distributed2"/>
</stack-ehcache:distributed-aq-cache-manager>
```
<bean id="cacheManager" class="org.springframework.cache.support.CompositeCacheManager">
  <property name="cacheManagers">
    <list>
      <ref bean="cacheManagerLocal" />
      <ref bean="cacheManagerDistributed" />
    </list>
  </property>
</bean>
Future Caching Features

- JMX Support for monitoring distributed caches
- Improved Ehcache configuration integration
- Possibly use AQ for discovery, direct sockets for transport.